What does that gene do?

Gene function prediction by machine learning with applications to plants

Makrodimitris, S.

DOI
10.4233/uuid:6744b23a-aec6-4852-837a-6ffd466ca24d

Publication date
2021

Document Version
Final published version

Citation (APA)
Makrodimitris, S. (2021). *What does that gene do? Gene function prediction by machine learning with applications to plants*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:6744b23a-aec6-4852-837a-6ffd466ca24d

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

WHAT DOES THAT GENE DO?

GENE FUNCTION PREDICTION BY MACHINE LEARNING
WITH APPLICATIONS TO PLANTS

WHAT DOES THAT GENE DO?

GENE FUNCTION PREDICTION BY MACHINE LEARNING
WITH APPLICATIONS TO PLANTS

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir.T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
maandag 1 maart 2021 om 15:00

door

Stavros MAKRODIMITRIS

Master of Science in Computer Science,
Technische Universiteit Delft, Delft, Nederland,
geboren te Cholargos, Griekenland.

Dit proefschrift is goedgekeurd door de

promotor: prof. dr. ir. M.J.T. Reinders
promotor: prof. dr. R.C.H.J. van Ham

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. ir. M.J.T. Reinders	Technische Universiteit Delft
Prof. dr. R.C.H.J. van Ham	Technische Universiteit Delft
Onafhankelijke leden:	
Prof. dr. L.F.A. Wessels	Technische Universiteit Delft
Dr. I. Friedberg	Iowa State University
Prof. dr. M.A. Huijnen	Radboud Universiteit Medisch Centrum
Prof. dr. ir. B.J.G Scheres	Wageningen University and Research
Prof. dr. K. Vandepoele	Universiteit Gent
Prof. dr. ir. R.L. Lagendijk	Technische Universiteit Delft, reservelid

Keywords: bioinformatics, gene function prediction, gene ontology, machine learning, plant biology
Printed by: ProefschriftMaken
Front & Back: Lu Lu

ISBN 978-94-6423-151-9

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

CONTENTS

SUMMARY

Billions of people world-wide rely on plant-based food for their daily energy intake. As global warming and the spread of diseases (such as the banana Panama disease) is substantially hindering the cultivation of plants, the need to develop temperature- and/or disease-resistant varieties is getting more and more pressing. The field of plant breeding has been revolutionized by the use of molecular biology methods, such as DNA and RNA sequencing, which substantially accelerated the finding of genes that are likely to influence a trait of interest. The outcome of such experiments is typically a long list of candidate genes whose involvement in the trait needs to be experimentally validated. Prioritizing these experiments, i.e. testing the most promising genes first, can save a lot of time, effort and money, but is often hindered by the fact that the cellular roles (functions) of plant genes and the corresponding proteins is often unknown.

Experimentally discovering the functions of genes is equally time-consuming and costly, so it is crucial to have computer algorithms that can automatically predict gene or protein functions with high accuracy. After decades of research on this field, considerable progress has been made, but we are still far from a widely-acceptable and accurate solution to the problem.

This thesis explores different research directions to improve protein function prediction, by developing new machine learning algorithms. These directions include new ways to represent proteins, exploiting semantic relationships among functions, and function-specific feature selection. The thesis also deals with the problem of missing protein interaction data for non-model species and quantifies its effect on protein function prediction. All in all, it provides novel insights to the problem that future work can build upon to lead to new breakthroughs.

SAMENVATTING

Miljarden mensen wereldwijd zijn voor hun dagelijkse energie-inname afhankelijk van plantaardig voedsel. Omdat de opwarming van de aarde en de verspreiding van ziekten (zoals de Panamaziekte bij bananen) de teelt van planten aanzienlijk belemmeren, wordt de noodzaak om temperatuur- en / of ziekteresistente rassen te ontwikkelen steeds dringender. Het onderzoeksveld van plantenveredeling is gerevolutioneerd door het gebruik van moleculaire biologietechnieken, zoals DNA- en RNA-sequencing. Deze technieken hebben het vinden van relevante genen aanzienlijk versneld. Het resultaat van dergelijke experimenten is doorgaans een lange lijst van kandidaatgenen, waarvan de betrokkenheid bij de eigenschap van interesse uiteindelijk ook experimenteel bevestigd moet worden. Het prioriteren van deze experimenten, d.w.z. eerst de meest veelbelovende genen testen, kan veel tijd, moeite en geld besparen, maar wordt vaak gehinderd door het feit dat de cellulaire rollen (functies) van plantgenen en de bijbehorende eiwitten vaak onbekend zijn.

Een experimentiele bepaling van genfuncties is echter even tijdrovend en kostbaar. Het is dus cruciaal om computeralgoritmen te hebben die automatisch gen- of eiwitfuncties met hoge nauwkeurigheid kunnen voorspellen. Na decennia van onderzoek op dit gebied is er aanzienlijke vooruitgang geboekt, maar we zijn nog verre van een algemeen aanvaardbare en nauwkeurige oplossing voor het probleem.

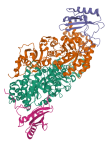
Dit proefschrift verkent verschillende onderzoeksrichtingen om de voorspelling van eiwitfuncties te verbeteren en introduceert nieuwe algoritmen gebaseerd op machine learning technieken. Enerzijds worden nieuwe manieren besproken om eiwitten weer te geven, om gebruik te maken van semantische relaties tussen functies en om functiespecifieke kenmerken te selecteren. Anderzijds wordt ook het probleem van ontbrekende eiwitinteracties voor niet-modelsoorten besproken en een poging gedaan om het effect hiervan op de voorspelling van eiwitfuncties te kwantificeren. Alles overziend, worden er nieuwe inzichten in het probleem geboden waarop toekomstige werk kan voortbouwen om tot nieuwe doorbraken te leiden.

1

INTRODUCTION

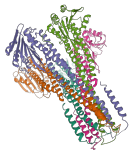
1

Proteins are crucial for all living organisms and viruses, as they are responsible for the vast majority of functions that are needed for their survival, thereby deserving the name "the workhorses of the cell". For example, proteins catalyze most chemical reactions in a cell, transport molecules in and out of it, regulate gene expression and participate in DNA packing, replication, and repair. Proteins are also responsible for cell defence, recognizing and neutralizing intruders, and even make sure that other proteins are constructed and folded properly. To accommodate such enormous diversity, proteins come about in a wide variety of sizes and three-dimensional structures (e.g. Figure 1.1), each specifically fine-tuned by evolution to fit a specific purpose [1].



(a) Structure of Rubisco large chain. (PDB ID: 5IUO, [2])

(b) Structure of a fungal peptide. (PDB ID: 5OD4, [3])



(c) Structure of an ion transporter. (PDB ID: 2BBJ, [4])

Figure 1.1: Examples of proteins that perform different functions. (a) Large chain of ribulose biphosphatase carboxylase (Rubisco), from *Arabidopsis thaliana*. (b) An effector protein from the fungus *Fusarium oxysporum f. sp. lycopersici* that causes the tomato wilt disease. (c) A Magnesium ion (Mg^{2+}) transporter from the bacterium *Thermotoga maritima*.

Understanding the exact role(s) that each protein plays in the cell will help us to better understand not only how living organisms work, but also how a change in a protein (e.g. a mutation, a change in quantity or even complete absence of it) can either disrupt the normal cellular programming and lead to disease or have beneficial effects, for instance by making the carrier immune to a particular pathogen.

However, experimental function discovery typically involves costly, laborious and time-consuming processes. For example, catalytic activity can be demonstrated using an enzymatic assay [5]. Binding-related functions require interaction assays, such as two-hybird screens [6]. Involvement in a certain process or phenotype can be discovered in multiple ways, including the co-occurrence of mutations with the phenotype, knock-out or over-expression experiments. Condition-specific functions, such as developmental processes or stress responses are typically discovered using differential expression analyses. The use of tissue-specific targeted mutagenesis with the CRISPR-Cas9 method is expected to greatly increase the throughput of function discovery in vivo [7]. A comprehensive description of how functions are assigned to proteins (either experimentally or computationally) can be found at `http://geneontology.org/docs/guide-go-evidence-codes/`.

In the past years, there has been an overwhelming increase in available genome and protein sequences. As of July 2020, UniRef90 - the set of proteins with at most 90% sequence identity to each other - comprises more than 110 million sequences. As a result, the experimental determination of function is falling severely behind the discovery of genes with unknown functions, especially for non-model species. For example, according to data from ensembl [8], in Arabidopsis thaliana, still only 61% of the protein-coding genes have been assigned at least one function. For mouse, a model animal species, this

fraction is about 50%. However, for rice and chicken, species of much larger economic importance, the fractions of genes with experimental GO terms are below 15% [8].

Considering the amount of experiments that need to be carried out on different species, tissues, environmental and experimental conditions, and so on, to discover all protein functions in all species, one can quickly realize the need for computational methods that predict protein functions from the currently available knowledge. As a result, a large number of Automatic Function Prediction (AFP) methods have been proposed and are still being proposed. However, despite the growing interest, a globally applicable solution to the problem of AFP has eluded researchers for decades. This thesis deals with AFP from different perspectives and proposes new algorithms to predict protein function.

1.1. REPRESENTATIONS OF PROTEIN FUNCTION

To be able to model and/or predict protein functions with an algorithm, we need to have a formal definition of protein functions. Several such systematic definitions of functions have been proposed and they typically make use of a hierarchical structure (taxonomy) to describe functions at different levels of granularity. The most widely-used functional representations are:

1. the Functional Categories (FunCat) [9],
2. the Enzyme Commission numbers [10],
3. BRENDA [11],
4. the Kyoto Encyclopedia of Genes and Genomes (KEGG) [12],
5. Reactome [13], and
6. the Gene Ontology (GO) [14, 15].

FunCat is a two-level taxonomy [9]. The first level of FunCat 2.0 contains six general functions such as "metabolism", "transport" and "development" as well as a category that denotes that a protein's function is unknown. The second level provides more detailed functional descriptions such as "cell fate", "tissue differentiation" and "energy". FunCat is no longer maintained, but a set of FunCat annotations of the yeast genome has been uploaded to MULAN (`http://mulan.sourceforge.net/index.html`), a database of machine learning datasets. This has lead to FunCat being still in use, mainly by the machine learning community when they want to test a novel method on a variety of datasets.

As the name suggests, the Enzyme Comission provides a system description of the function of enzymes. It is organized in four levels, again with the first level describing general chemical reactions (hydrolysis, oxydoreduction, ligation etc.) and deeper levels providing further details such as the type of substrate. Enzyme comission annotations are usually given in a form of four numbers, one for each level. For example, the mitochondrial gene ALAAT1 from *Arabidopsis thaliana* codes for a protein called Alanine aminotransferase 1, which is annotated with Enzyme Comission number 2.6.1.2. This is read as follows: the first '2' denotes the level 1 class "transferase". The level two annotation 2.6 denotes "transferase transferring nitrogenous groups". 2.6.1 further specifies this to mean transfer of an amino group and the last level (2.6.1.2) specifies that this enzyme acts on the amino acid alanine. The BRENDA resource [11] provides a more elab-

orate functional description of enzymatic function, as it complements Enzyme Commission numbers with additional information such as kinetics, stability and specificity. Although these resources are manually curated and very informative, they are restricted to only enzymes and cannot be used to describe all cellular functions.

KEGG [12] and Reactome [13] provide descriptions of pathways as well as chemical reactions in which a protein can be involved. Reactome is mainly focused on human pathways, but also uses homology to transfer annotations to other species. On the other hand, KEGG contains information on many species from different domains of life.

By far the most elaborate and widely-used functional representation is the Gene Ontology (GO) [14, 15]. GO describes function using three different aspects: molecular function (MF), biological process (BP) and cellular component (CC), though one could argue that the latter also describes protein localization and not only function.

1.1.1. DETAILS ABOUT THE GO

This thesis exclusively uses the Gene Ontology as function representation due to its universality. Therefore, we present some of its important properties here. The GO is structured as a Directed Acyclic Graph (DAG) in which every node corresponds to a term (function) and edges denote relationships between terms. Each sub-ontology (MF, BP and CC) has a root term (GO:0003674, GO:0008150 and GO:0005575 respectively) that has the same name as the sub-ontology and can be perceived as the 0-th level in a hierarchy. Every next level describes more and more specific functions, as shown in the example in Figure 1.2. The GO release of June 2020 comprises over 44,000 different functional labels (called GO terms): 29,112 in BP, 11,118 in MF and 4,181 in CC. The root of each ontology is an ancestor of all terms in that ontology. Figure 1.3 shows the per-level distribution of all terms and leaf terms (terms with no descendants) in the three sub-ontologies.

DAG nodes (terms) can be connected with different types of edges, which denote different types of relationships between the concepts. The most common edge type is the "is_a" edge, which connects a term to a broader one from the same sub-ontology. For example, "DNA-templated transcription" (GO:0006351) is a "nucleic acid templated transcription" (GO:0097659) which is a "RNA biosynthetic process" (GO:0032774). Regulatory relations are represented by "regulates" edges, e.g. "regulation of DNA-templated transcription" (GO:0006355) regulates "DNA-templated transcription" (GO:0006351). Part-whole relations are expressed with "part_of" edges, e.g. "DNA-templated transcription" (GO:0006351) is a part of "gene expression" (GO:0010467). Finally, "capable_of" and "occurs_in" edges connect terms across the three different GO components. For instance "Cry-Per complex" (GO:1990512, from CC) is capable of "zinc ion binding" (GO:0008270, from MF) and "chloroplast mRNA modification" (GO:1900871, from BP) occurs in "chloroplast" (GO:0009507, from CC).

An important feature of the Gene Ontology is the so-called "true path" rule, which states that if a GO term is assigned as functional annotation to a protein, then all the ancestors of that term must be assigned as well. This is because ancestors are broader descriptions that encompass all concepts of their descendants. As a result, all proteins are by definition annotated with the ontology root. More generally, the true path rule implies that not all combinations of GO terms are feasible and automatic GO term as-

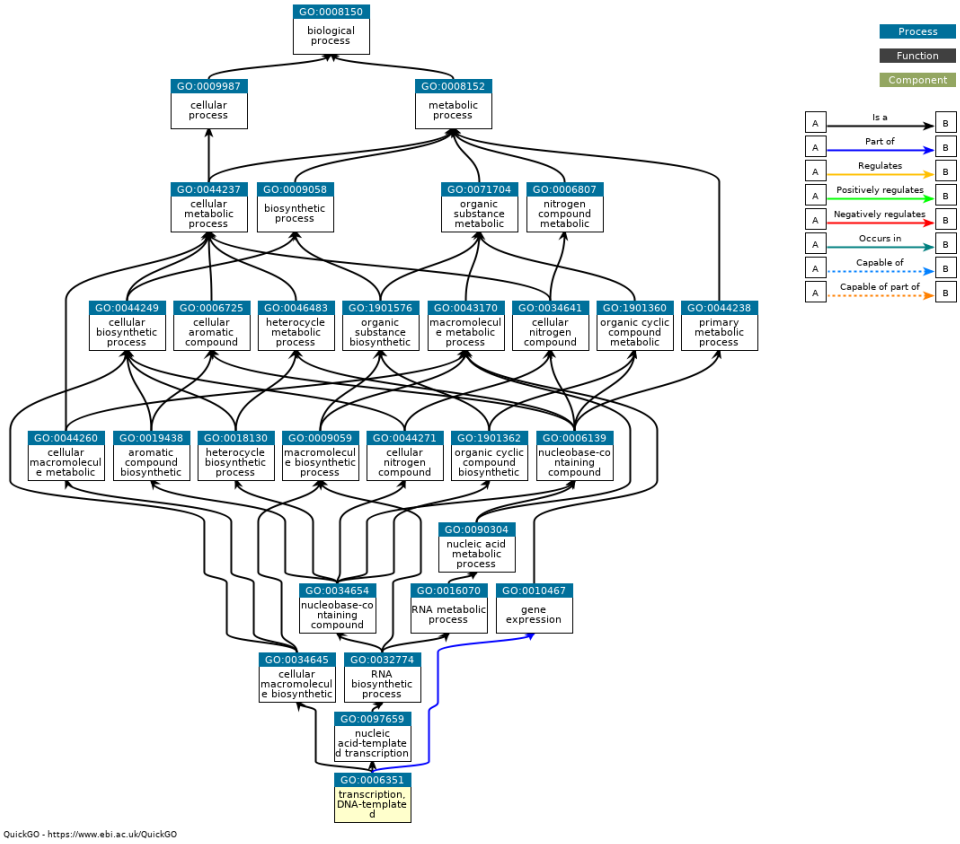


Figure 1.2: Example sub-graph of the Gene Ontology. Image from QuickGO [16]

signment algorithms should account for that.

1.1.2. QUANTIFYING THE IMPORTANCE AND RELATEDNESS OF GO TERMS

It is desirable to determine a protein's function as specifically as possible, which means that we are mainly interested in terms that are as far away from the root as possible. For example, the term "catalytic activity" (distance 1 from the root) simply tells us that a protein is an enzyme, whereas "deoxynucleotide 3'-phosphatase activity" (distance 6 from the root) specifically describes a type of reaction the enzyme catalyzes.

This example highlights the utility of being able to measure how "important" GO terms are. Based on that example, it is straightforward to define a term's importance using its path length to the ontology root. Due to the complicated structure of the GO DAG, for many terms there are multiple such paths. We can therefore measure importance as both the shortest and the longest path length to the ontology root (also known as the level and depth of a term respectively) [17].

Although level and depth are very intuitive, they can be misleading. For instance, leaf

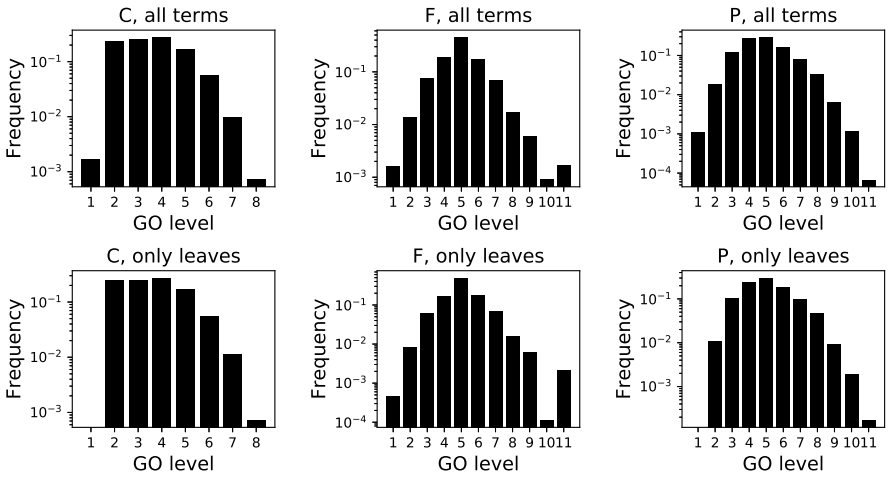


Figure 1.3: Fraction of terms (y-axis, in log scale) that have a specific shortest path length to the ontology root (x-axis) for the cellular component (C, left), molecular function (F, middle) and biological process (P, right). The top row shows the distribution of all terms and the bottom row of only leaf terms.

terms, i.e. terms without descendants, which represent the most specific description of a function available, can be found at varying depths (e.g. "epinephrin binding" has depth 4, "deoxynucleotide 3'-phosphatase activity" 6 and "D-ribose-importing ATPase activity" 8). To counter this, alternative measures of importance have been proposed based on information theory. Such measures assume that general functions are assigned to a large number of proteins, whereas the most specific functions are assigned much more rarely. According to information theory, rare events are more "surprising", thus observing them gives us more information about a system. Similarly, we define rare GO terms as more informative than frequent ones. Resnik [18] defined the Information Content (IC) of a term t as minus the logarithm of $P(t)$, the fraction of total annotations in a corpus that include t (equation 1.1).

$$IC_R(t) = -log_2(P(t)) \tag{1.1}$$

IC_R relies on the frequency of each term compared to the others, but ignores the DAG structure. Clark et al. proposed an alternative definition of Information Content based on the rationale that if a term has exactly one descendant, then the presence of the term is very good indication of the presence of its descendant too. That means that observing the descendant is not really "surprising", regardless of how (in)frequent it is, so it carries little information. Conversely, for a term with many siblings (terms with the same parents) it is harder to infer its presence even if we have demonstrated that its parents are present and that makes such terms more informative. The Clark Information Content (IC_C) [19] models that using equation 1.2, where $\mathbb{P}(t)$ denotes the set of parents of term t.

$$IC_C(t) = -log_2(P(t|\mathbb{P}(t))) \tag{1.2}$$

The probability $P(t|\mathbb{P}(t))$ is estimated as the number of proteins annotated with t divided by the number of proteins annotated with all terms in $\mathbb{P}(t)$.

Apart from a more objective representation of term specificity, the notion of Information Content additionally enables us to measure pairwise similarities of GO terms more effectively. Terms that are directly connected by an edge are obviously similar as they describe related concepts, but extending that is less straightforward. To do so, for a pair of terms t_1, t_2 we define the concept of the Most Informative Common Ancestor ($MICA(t_1, t_2)$), i.e. the term from the set of common ancestors with the highest information content. A related concept is the common ancestor with the largest depth, also known as Lowest Common Ancestor ($LCA(t_1, t_2)$). Note that for IC_R, the $MICA$ and LCA is always the same term, but for IC_C this is not necessarily the case. The similarity between two terms - also known as Semantic Similarity ($SS(t_1, t_2)$) - can then be defined using these concepts. For example, Resnik defined semantic similarity as the information content of the $MICA$ (equation 1.3) [18] and Lin scaled this calculation by the information content of the two terms themselves [20] (equation 1.4). More recently, another definition of semantic similarity was proposed that has similar properties but is only computed based on the DAG topology and does not require an annotated corpus, which accelerates the computation [21].

$$SS_R(t_1, t_2) = IC(MICA(t_1, t_2)) \tag{1.3}$$

$$SS_{Lin}(t_1, t_2) = 2\frac{IC(MICA(t_1, t_2))}{IC(t_1) + IC(t_2)} \tag{1.4}$$

1.2. FORMAL PROBLEM DEFINITION

THE problem of AFP can be viewed as a classification task, where every protein is a sample and every function (GO term) is a class. The goal is to predict which GO term(s) should be assigned to a protein, i.e. to which class(es) a protein belongs. Given that a protein can belong to multiple classes, this falls under the multi-label classification paradigm.In addition, the hierarchical nature of GO DAG makes it a structured output learning problem [22].

Formally, for a protein i, we represent its GO annotations as a binary vector $\mathbf{y}_i \in \{0,1\}^C$, with C denoting the number of labels (GO terms) and $y_{it} = 1$ if and only if protein i is annotated with term t. We also use $x_i \in \mathbb{D}$ - for some (typically multi-dimensional) input domain $\mathbb{D}$ - to denote a pre-defined protein representation. Ideally, our goal is to find a function $\mathbf{f} : \mathbb{D} \rightarrow \{0,1\}^C$, so that $\hat{\mathbf{y}}_i = \mathbf{f}(x_i) \approx \mathbf{y}_i \; \forall i$. Due to technical reasons, it is difficult to find such a function with discrete outputs, so typically the output domain is changed to $[0,1]^C$, meaning that the function returns a posterior probability that the input protein is associated with each label.

If our goal is to predict which proteins are involved in a predefined function - e.g. in order to prioritize genes for experimental validation - then we are typically not interested in predicting all the GO terms, but only one. This means that the function $\mathbf{f}$ does not

map the input to a vector but a single scalar. Also, we can model this task as a retrieval problem, in which case **f** does not need to produce posterior probabilities, but simply a relative ranking of all input proteins without any restrictions on the output range.

1.2.1. POSSIBLE INPUT DOMAINS

The domain $\mathbb{D}$ of x_i can vary depending on the input data we want to use. The most common representations (also graphically shown in Figure 1.4) are:

1. fixed-length feature vectors,
2. variable-length feature vectors that depend on the protein length,
3. (dis)similarity-based representations, and
4. network representations.

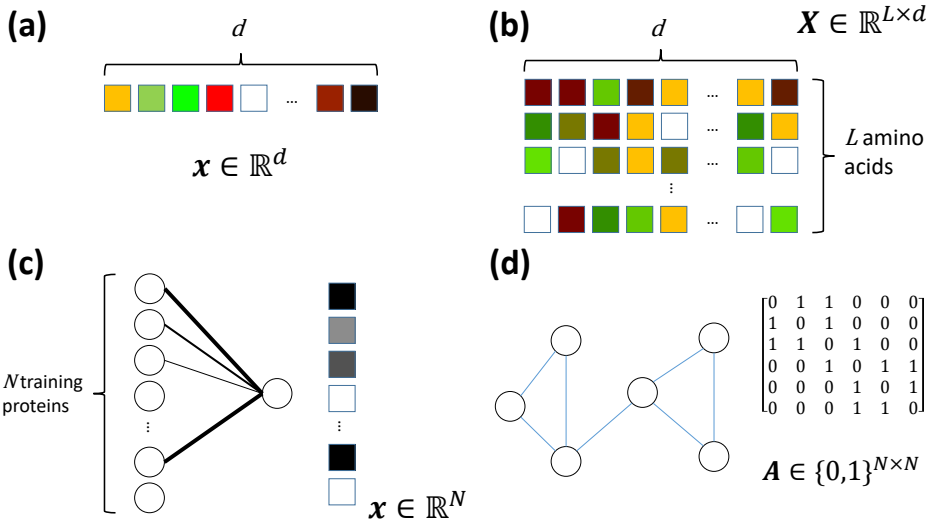


Figure 1.4: Schematic overview of the four most common protein representation used by function prediction methods: a protein feature vector (a), an array of amino acid feature vectors (b), similarity representation (c), and network representation (d).

If we choose to represent a protein of length L by a fixed-length feature vector of dimension d, then $\mathbb{D} = \mathbb{R}^d$. This feature vector can contain some protein attributes that we believe are descriptive of its function (e.g. counts/fractions of amino acids, length, mass, net charge, presence or absence of some protein domains etc). It can also contain the expression values of the gene that codes for that protein measured in d different samples. An example prediction function **f** for this representation can be any standard machine learning algorithm, such as a Support Vector Machine (SVM) or a Multi-Layer Perceptron (MLP) [23].

Alternatively, we can define a $d-$dimensional amino acid feature vector and describe each protein as a sequence of its amino acids. That would mean that $\mathbb{D} = \mathbb{R}^{L\times d}$, so that x_i is a matrix with L rows and d columns. This implies that the first dimension of x_i

is different for each protein. Here, the prediction function $\mathbf{f}$ can be a Recurrent Neural Network (RNN) [24].

Another way to represent proteins is by describing how similar they are to other proteines, which is known as a (dis)similarity-based representation [25]. Doing so requires a set of N_{tr} training proteins with known functions and a measure of (dis) similarity between two proteins, such as the percent sequence identity or the bit score calculated by aligning the sequences [26]. We then represent each protein by its (dis)similarities to the training proteins. As an example, if we use the pairwise percent sequence identity as a similarity measure, then $\mathbb{D} = [0,100]^{N_{tr}}$. In this case, a widely-used prediction function $\mathbf{f}$ is $\mathbf{f}(x_i) = \mathbf{y}_h$, where $h = \arg\max(x_i)$ and the j-th element of vector x_i contains the similarity between proteins i and j. That way we find the most similar training protein and transfer its GO annotations to the query. The vector of similarities can also be used as a feature vector for the protein and fed to classifier.

Network-based representations also fall under the (dis)similarity paradigm. Proteins are represented as nodes of a graph and edges reflect some type of similarity between them, such as co-expression of the corresponding genes, affinity for mutual interaction etc. Gene co-expression is typically calculated using the Pearson or Spearman correlation of the expression values of two genes across a variety of conditions, though more elaborate co-expression measures have been proposed [27, 28]. Protein-Protein Interaction networks usually consist of binary edges connecting proteins that are known to interact, although if information on the strength of the interaction is available, it can be incorporated as edge weights (e.g. [29]). The main difference of network representations from other (dis)similarity-based representations is that they are typically restricted to proteins from one species, unless a list of pairwise homologous genes is available, meaning that such methods cannot be readily applied to newly sequenced species without any proteins of known function. As we are restricted to one species, one can reasonably assume that all proteins for which the method will have to make predictions are known at training time, making this a case of transductive learning [30, 31]. Simply put, a transductive learner does not need to generalize beyond a pre-defined test set (because all the possible test samples are assumed to be already known), meaning that the input features of test samples can be incorporated into model training. For instance, they can be used during pre-processing (e.g. normalization or dimensionality reduction) and/or feature generation (e.g. if node degree is a useful feature, one can include edges from/to test proteins in the calculation).

1.2.2. DETAILS ABOUT THE LABEL VECTOR

As far as the label vector ($\mathbf{y}_i$) is concerned, the number of labels, C, is at least 4,181 (for the CCO), so the total number of different label combinations could in theory reach $2^{4,181} \approx 10^{1,258}$ for the smallest of the three ontologies, which is an incomprehensibly large number.

Fortunately, the true path rule of the Gene Ontology imposes constraints on $\mathbf{y}_i$, so that most of the theoretical label combinations are not possible. For example, each ontology has a root term that is ancestor to all other terms. Clearly, this term cannot be absent and that already eliminates half of the label combinations. Similarly, for every other label with descendants all the combinations where the label is absent and at least

one of its descendants is present are impossible due to the true path constraints. Furthermore, biological constraints, such as the fact that biological process can be mutually exclusive [32], reduce the number of possible combinations even more. Also, due to the DAG structure related labels co-occur which further reduces possibilities. To make this more concrete, as of July 2020, there are 54,100 proteins from the SwissProt database with at least one experimental GO annotation in CCO. Although the total number of GO terms present in that corpus is 2,750, the rank of the label matrix Y is 2,511 ($<< 2^{2,750}$), meaning that there are only 2,511 linearly independent label combinations. In MFO, for 53,592 proteins and 6,638 terms, the number of independent functional components is 6,061. This shows that GO terms are indeed correlated. Such correlations can be exploited to improve performance [33, 34].

1.3. BRIEF LITERATURE REVIEW OF PROTEIN FUNCTION PREDICTION

FOR many years, nearest neighbor methods - also called "Guilt-By-Association" (GBA) - have been the cornerstone of AFP. The most famous instance of this class of methods is homology search: If two proteins have similar sequences, then we can deduce that they will also have the same or similar function. The Basic Local Alignment Search Tool (BLAST) is a fast homology search algorithm that can facilitate this procedure [26] by finding sequences in a database that are similar to a query. Given these sequences, one could directly transfer the functions of the top hit, as illustrated in Figure 1.5, or use a weighted approach where annotations are transferred from all hits, but with a posterior probability that depends on the similarity between the query and the hit [35]. As sequences can accumulate mutations over the course of evolution without changing their function, for some sequences it may be required to search for more remote homologs. This can be achieved by Position-Specific Iterative BLAST (PSI-BLAST) [36] or by building Hidden Markov Models (HMM) of protein families and performing protein-profile alignment, which have been shown to be better at finding distant homologs [37]. Nearest neighbor methods for AFP have also been applied to other data types than sequence, such as gene co-expression networks (e.g. [38]).

1.3.1. MACHINE LEARNING METHODS

On the other hand, machine learning methods try to explicitly learn what makes a protein have a particular function instead of looking for similar proteins. Such methods can be broadly classified into the categories (Figure 1.6):

1. Single-Label methods,
2. Multi-Label methods and
3. Hierarchical methods.

Single-Label methods learn an independent classifier for each GO term. This approach is often called "Binary Relevance" in the machine learning multi-label literature, because it splits the problem into a series of binary classification tasks. This is not a popular approach, because it cannot exploit label correlations and/or mutual exclusions which in this case are imposed by either the ontology graph or the biology. As a result,

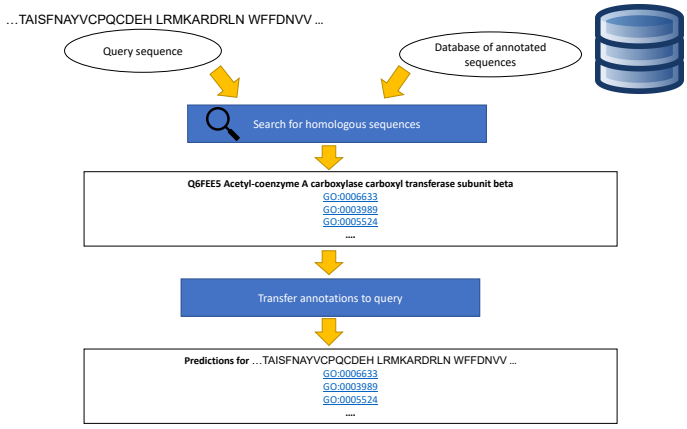


Figure 1.5: Overview of functional annotation transfer using sequence homology search.

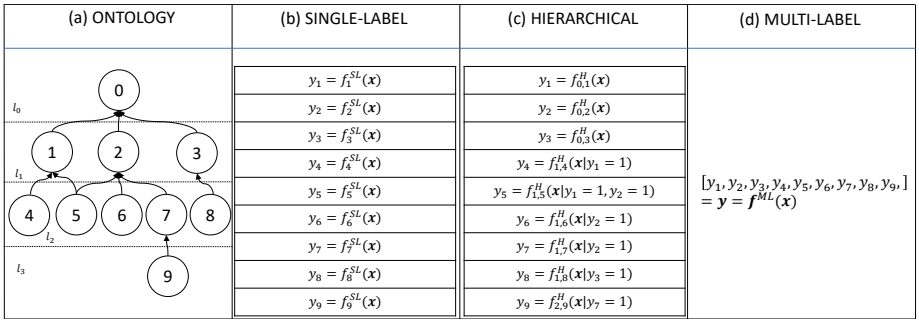


Figure 1.6: Explanation of the Single-Label (b), Multi-Label (c), and Hierarchical (d) learning paradigms through an example ontology (a).

it is an inefficient approach that can work at most as good as multi-label ones, but requires more parameters [39]. However, it can be useful for certain tasks, such as gene prioritization for specific functions [39, 40].

Apart from Binary Relevance, the most naive way to handle multi-label data is the so-called Label Powerset method. This method transforms a multi-label problem with C labels into a multi-class problem where every possible subset of the labels is treated as a separate class. The advantage of this is that the problem can be approached as a standard multi-class classification, but it has a few big disadvantages: First, as previously discussed, the resulting number of classes is very big, even after ignoring impossible label combinations. Second, it makes all label combinations equally dissimilar to each other, which is not true as some of them share labels. This is problematic during learning, as it makes it difficult for the learning algorithm to "know" when it is on the right track. For example, imagine a protein that has to be annotated with terms A, B and C and a learning algorithm that is being trained. If at some stage of the training, the current parameters of the algorithm are such that for terms A and B the posterior probability is large, but for

C it is close to zero, for the multi-class objective function this is considered completely wrong (because it is not exactly A, B and C). Even worse, the current algorithm parameters are as bad as another set of parameters that give low posteriors to A, B, and C and predict another term D with high confidence. As a result, the gradients may push the model away from predicting A and B, which is obviously undesirable.

Therefore, it makes more sense to approach the problem as purely a multi-label task, where the goal is to predict the entire label vector in one go. Although not explicitly stated before, GBA methods such as the k-NN classifier are inherently multi-label. Another approach that has recently gained in popularity is to train a model, such as a neural network with the aim of minimizing a loss over all output labels. A widely-used loss function is the average cross-entropy over all labels [41, 42]. Note that using this loss function with a linear model is equivalent to training a linear classifier - namely logistic regression - independently for each class, making it a Binary Relevance approach. But a non-linear model, such as a multi-layer perceptron, trained with this loss, is indeed jointly learning all labels, as errors in one of them are back-propagated to neurons of the hidden layer, thereby influencing all other terms. Fa et al. have applied a multi-task approach where the first hidden layers are shared for all terms and the latter ones are term-specific, leading to improved performance [43].

Multi-task networks have also been applied in a hierarchical fashion for predicting function [44], where a neural network is trained on all terms that are of a certain path length to the ontology root, also called a GO layer. One of the early works in this hierarchical framework applied a decision tree for each layer and used as training data for each label only the data points that were annotated with one of its parent labels. This approach was more effective than Binary Relevance on predicting FunCat functions in yeast [45].

1.3.2. NETWORK-BASED METHODS

The simplest AFP method for network data is again the GBA, where GO annotations are transferred to a protein from its annotated interacting partners. More advanced approaches do not use the direct interaction between two nodes but the extent to which they share interacting partners as a similarity measure (e.g. [27, 28]).

The first attempts to use machine learning on biological networks for predicting function made use of graph kernels to extract hand-crafted node features which were then fed to a classifier [46, 47]. More recent machine learning methods use automatically learned features mainly based on the concept of a shared neighborhood described in the previous paragraph. The goal of such approaches is to learn a feature vector for each node, so that nodes with highly similar neighborhoods have large inner products in the learned space. This can be achieved by algorithms such as node2vec [48], which employs random walks to estimate the neighborhood of each node. One of the first methods to use this technique was MASHUP [49]. DeepNF improved upon the performance of MASHUP by replacing the random walks with a graph auto-encoder to learn node features [50].

1.3.3. THE TRUE PATH RULE AND PREDICTED FUNCTIONAL ANNOTATIONS

GO annotations have to follow the true path rule and the same holds for predicted annotations. It is easy to show that nearest neighbor methods that make predictions by transferring annotations from one or more proteins are guaranteed to respect the true path rule. The same holds for hierarchical classifiers as described above. However, single-label or multi-label methods are not guaranteed to respect the true-path rule. The most common way to address this is to post-process the posterior scores, so that no term has a higher posterior probability of being assigned to a protein than any of its ancestors in the ontology [51]. This guarantees that no matter what threshold we choose for the posteriors, the resulting hard predictions will respect the true path rule. This idea can also be directly incorporated into a prediction model by using "merge" layers that combine the outputs of a term and its children and output their maximum [41]. Another study attempted to learn this post-processing step using a linear model instead of just applying the true path rule, which lead to improved performance on a FunCat dataset [52].

1.3.4. INTEGRATION STRATEGIES

Integrating information from multiple data sources is also a useful approach, as it can lead to enhanced performance. For example, DeepGO learns sequence-based features using a neural network and concatenates them with network embeddings into a large feature vector used for classification [41]. A more common strategy is to integrate the posterior probabilities of different methods - leading to so-called ensemble predictors - either by averaging them [38] or combining them using Bayesian methods [53]. This late integration of the posterior probabilities is in principle more effective because data from different sources are very heterogeneous, with different ranges and distributions, as explained above. Moreover, for similarity-based methods, we do not expect that two proteins with high sequence similarity are necessarily co-expressed or that two co-expressed proteins necessarily interact. This implies that simply averaging the similarities might dilute the functional signal.

Recently, a more powerful late integration scheme has been proposed that uses learning to rank [54] to prioritize terms for a given protein based on the prediction scores of several simpler predictors [55, 56]. This approach outperformed simpler integration strategies [55] and proved very successful in the CAFA3 challenge, especially for the MFO [40].

1.4. EVALUATION OF AUTOMATED FUNCTIONAL ANNOTATION

HAVING an appropriate method of evaluating and comparing function prediction algorithms is at least as important if not more important than the algorithms themselves. Such an evaluation strategy should ideally satisfy all of the following three criteria:

- Objectivity, i.e. it should not provide an unfair advantage to certain algorithms.
- Independence, in the sense that the test data and labels are not correlated or dependent in any way on the training data.
- Unbiasedness, meaning here that it gives an accurate estimate of the expected performance on new unseen data.

1.4.1. EVALUATION SET-UPS

There are two main experiment types carried out to evaluate and compare AFP algorithms: Cross-Validation (CV) and Temporal Hold-Out (THO). In CV, all available annotated proteins are randomly split into k equal parts (folds) and each fold acts as a test set exactly once and as part of the training set the remaining $k-1$ times. On the other hand, THO generates a single train-test split, based on a pre-defined time point. All proteins that did not have any annotations at that point but have accumulated some ever since constitute the test set, while proteins annotated already before that time are placed in the training set. The Critical Assessment of Functional Annotation (CAFA) adopts the THO by asking participants to make predictions for unannotated proteins before a specific deadline and then evaluating on the subset of those proteins that accumulate experimental annotations in a period of 9-12 months after the deadline [35, 40, 57].

A comparison of the two set-ups, revealed that CV is too optimistic and often overestimates the true performance on unseen data compared to THO [58], meaning that CV is more biased. This difference is attributed to the lack of independence between the training and the test set (information leakage) induced by the CV. The authors argue that GO term frequencies vary over time and mixing annotations from different time points creates an average term frequency that is more or less similar to the training set. In our experience, based on results presented in the following chapters as well as unpublished results, the two set-ups tend to be correlated in terms of the relative ranking of the methods, implying that they do not differ in terms of objectivity [34], which is contradictory to the observations in [58]. Also, by tracking the performance of the naive method [35] over the years, the CAFA organizers argued that GO term distributions have not significantly changed during the 9 years spanning the first three challenges [40]. Instead, we suspect that the information leak is due to the fact that the experiments carried out to annotate proteins are biased [59] and correlated to each other. For instance, if a function of a protein is discovered in mouse, it is more likely that scientists attempt to validate that for the human homolog than for another randomly chosen human protein. Finally, obtaining variance estimates from a THO experiments is less straightforward than CV, but it is typically done using bootstrapping.

1.4.2. EVALUATION MEASURES

The metrics used to evaluate AFP methods are categorized into protein-centric and term centric. Protein-centric metrics evaluate a method's ability to identify the appropriate GO terms for each test protein, which is a multi-label classification task. On the other hand, term-centric evaluation deals with the task of finding the proteins that should be annotated with each particular function (GO term), which constitutes a series of separate binary classification tasks. In the following, we provide definitions of some of the most commonly-used measures, making use of notation introduced in [57]. We use $T_i = \{t | t \in [1,2,..C] \wedge y_{it} = 1\}$ to denote the indices of the GO terms annotated to a protein i and $P_i(\tau) = \{t | t \in [1,2,..,C] \wedge f_t(x_i) \geq \tau\}$ to denote the set of terms that an AFP algorithm predicts for a protein i with a probability of at least τ. $m(\tau)$ is the number of proteins that get assigned at least one annotation with probability at least τ and n_e is the total number of test proteins.

Protein-centric evaluation is usually done using the F1 score and the Semantic Dis-

tance [19]. The F1 score is calculated from precision (equation 1.5) and recall (equation 1.6) using equation 1.7.

$$pr(\tau) = \frac{1}{m(\tau)} \sum_{i=1}^{m(\tau)} \frac{\sum_{t=1}^{C} I(t \in P_i(\tau) \wedge t \in T_i)}{\sum_{t=1}^{C} I(t \in P_i(\tau))} \tag{1.5}$$

$$rc(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} \frac{\sum_{t=1}^{C} I(t \in P_i(\tau) \wedge t \in T_i)}{\sum_{t=1}^{C} I(t \in T_i)} \tag{1.6}$$

$$F1(\tau) = 2\frac{pr(\tau) \cdot rc(\tau)}{pr(\tau) + rc(\tau)} \tag{1.7}$$

When comparing different AFP methods, it is common in the literature to use the optimal operating point for each method as selected from the test data (as shown in equation 1.8 for the F1 score). This of course provides an optimistic view of each method's performance and means that the observed metrics are upper bounds of the expected values of the metrics on unseen data.

$$F_{max} = \max_{\tau \in [0,1]} F1(\tau) \tag{1.8}$$

Another common metric that is based on precision and recall is the area under the precision-recall curve $AUPRC$, which depends on the performance on all operating points instead of only the optimal one. This metric is also called average precision.

A downside of the F_{max} is that it treats all terms equally, regardless of their specificity. A way to correct for that is to weigh each term in the calculation of precision and recall by its information content (equations 1.9 and 1.10) [19].

$$wpr(\tau) = \frac{1}{m(\tau)} \sum_{i=1}^{m(\tau)} \frac{\sum_{t=1}^{C} IC(t) \cdot I(t \in P_i(\tau) \wedge t \in T_i)}{\sum_{t=1}^{C} IC(t) \cdot I(t \in P_i(\tau))} \tag{1.9}$$

$$wrc(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} \frac{\sum_{t=1}^{C} IC(t) \cdot I(t \in P_i(\tau) \wedge t \in T_i)}{\sum_{t=1}^{C} IC(t) \cdot I(t \in T_i)} \tag{1.10}$$

Semantic Distance [19] is calculated based on Remaining Uncertainty (ru, equation 1.11) and Misinformation (mi, equation 1.12) using equation 1.13. Remaining Uncertainty measures the mean information that a prediction algorithm misses due to type II errors (false negatives) and Misinformation the mean information mistakenly introduced to a protein due to type I errors (false positives). Opposite to the F1 score, a lower Semantic Distance corresponds to better performance, so it is common to report the S_{min}, i.e. the minimum value of the metric over all operating points.

$$ru(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} IC(t) \cdot I(t \in T_i \wedge t \notin P_i(\tau)) \tag{1.11}$$

$$mi(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} IC(t) \cdot I(t \notin T_i \wedge t \in P_i(\tau)) \tag{1.12}$$

$$SD(\tau) = \sqrt{ru(\tau)^2 + mi(\tau)^2} \tag{1.13}$$

$$S_{min} = \min_{\tau \in [0,1]} SD(\tau) \tag{1.14}$$

Since proteins with higher total information content contribute more to the metric, a normalized version of ru (equation 1.15) and mi (equation 1.16) in which each test protein contributes equally to the calculation. The normalized S_{min} can then be calculated by inserting these metrics in equations 1.13 and 1.14.

$$nru(\tau) = \frac{1}{n_e} \frac{\sum_{i=1}^{n_e} IC(t) \cdot I(t \in T_i \wedge t \notin P_i(\tau))}{\sum_{i=1}^{n_e} IC(t) \cdot I(t \in T_i \vee t \in P_i(\tau))} \tag{1.15}$$

$$nmi(\tau) = \frac{1}{n_e} \frac{\sum_{i=1}^{n_e} IC(t) \cdot I(t \notin T_i \wedge t \in P_i(\tau))}{\sum_{i=1}^{n_e} IC(t) \cdot I(t \in T_i \vee t \in P_i(\tau))} \tag{1.16}$$

In term-centric evaluation, the most common metric is the area under the Receiver Operator Characteristic ($ROCAUC$). This is calculated per term and averaged over all terms. It is again possible to calculate a weighted average, using IC values as weights in order to emphasize performance on specific terms. One can also define term-centric precision (equation 1.17) and recall (equation 1.18).

$$pr_t(\tau) = \frac{1}{C} \sum_{t=1}^{C} \frac{\sum_{i=1}^{n_e} I(t \in P_i(\tau) \wedge t \in T_i)}{\sum_{i=1}^{n_e} I(t \in P_i(\tau))} \tag{1.17}$$

$$rc_t(\tau) = \frac{1}{C} \sum_{t=1}^{C} \frac{\sum_{i=1}^{n_e} I(t \in P_i(\tau) \wedge t \in T_i)}{\sum_{i=1}^{n_e} I(t \in T_i)} \tag{1.18}$$

Term-centric F_{max} and $AUPRC$ can then be defined based on those metrics similar to the protein-centric ones.

A comprehensive review and comparison of different AFP evaluation metrics is provided in [60]. In that work the authors used artificial predictions to compare different metrics and found significant differences between the metrics, in terms of sensitivity to false positives and ability to reward correct annotations. They also proposed novel metrics which they showed to be superior on both respects [60]. On the other hand, when comparing the performance values of 146 AFP algorithms using five common metrics on the CAFA3 benchmark [40], we found very high absolute rank correlations between all five of them (Figure 1.7). This was consistently the case for all three GO domains and all correlations were statistically significant, though we did observe that unnormalized S_{min} was the least correlated to the other four. This result implies that despite the differences and limitations of each measure, they all tend to give similar relative rankings of methods.

1.5. SCOPE AND ASSUMPTIONS OF THIS THESIS

THE primary focus of this work is on predicting functions of proteins for which no functional information is available. This is done mainly to emulate a situation where a new species has been sequenced and our goal is to find what are the functions of the different newly-discovered proteins. Furthermore, we are mainly interested in plant

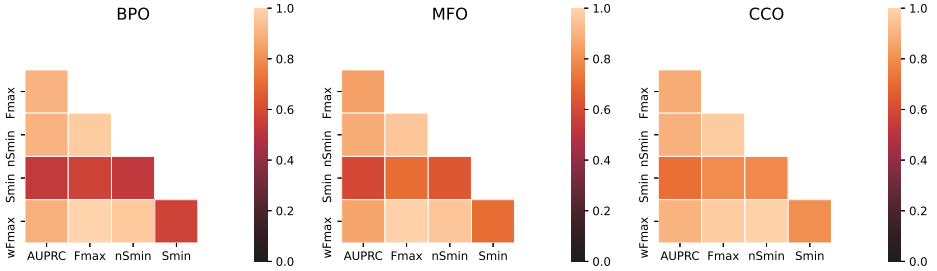


Figure 1.7: Pairwise absolute rank correlations between five different evaluation metrics calculated from the 146 methods that participated in CAFA3 for the biological process (left), molecular function (middle) and cellular component (right) ontologies. Brighter colors denote larger absolute correlations. All pairwise correlations are statistically significant with uncorrected p-values $< 10^{-11}$.

genomes, so we make use of *Arabidopsis thaliana* data, which is the only well-annotated plant species. However, to test the effectiveness of the proposed algorithms on a more general scale, we apply them on the CAFA benchmarks too. These benchmarks mostly include proteins from a handful of eukaryotic model species, with *A. thaliana* being the only plant among them. Only for a specific case in chapter 4, we perform benchmarks on a bacterium. Below, we make explicit certain assumptions that we make throughout the rest of the chapters.

1.5.1. THE "CLOSED-WORLD" ASSUMPTION

First and foremost, we operate in a "closed-world" setting, which essentially means that we assume that the label vectors are correct and complete. This is not correct, as our knowledge of protein function is incomplete. A GO term is assigned to a protein after an experiment, a curator or an algorithm has decided that there is enough evidence that the protein performs that function. On the other hand, the absence of an annotation does not necessarily mean that there is proof/evidence that the protein does not perform the function. Some negative annotations do exist (e.g. [61]), but they are very limited and extremely difficult to obtain. Note, that even experimental negative results cannot guarantee a negative annotation, as a protein could perform the function in question only under certain conditions, different from those used during the experiment, a phenomenon that we have previously brought up [39] for the CAFA-π challenge [40].

This fact about missing annotations is what makes functional annotation an "open world" [62], where a strictly correct label vector should not be binary, but also contain the third option "unknown". Typically, one uses 1 and -1 to denote the confirmed pres-

ence and absence of GO term respectively and 0 to denote lack of evidence for both. This ordering encodes a varying level of certainty about a protein performing a function, from being sure that it does so (1), to not knowing (0) to knowing it does not (-1). Another way to model the open world is by making use of Positive-Unlabeled (PU) learning [63], in which we assume that an unknown fraction of samples of the negative class is actually positive. PU learning algorithms can employ a variety of techniques, such as obtaining a strict negative set using the unlabeled examples that are most dissimilar to the positive ones [63], only focusing on whether new examples are highly similar to existing positive ones (which is similar to one-class classification [64]) or modifying the learning cost function to demote the effect of false positives compared to false negatives. This is done because, in this setting, an unlabeled sample classified as positive should not be considered as big a mistake as a positive example not predicted as such.

Although computational methods have been proposed to predict them [61, 65], the number of available experimental negative annotations is relatively small. Furthermore, we expect that most proteins are highly specialized [1], meaning that GO annotations are expected to be relatively sparse, so that the vast majority of protein-term pairs would really be true negatives.

Operating under the "open world" premise opens another interesting direction, namely the task of completing partial GO annotations for proteins. For instance, Khatri et al. performed Singular Value Decomposition to the label matrix $\mathbf{Y}$ whose rows were the label vectors of human proteins to obtain a lower-dimensional functional representation that explained most of the variance of the functional information, with the assumption that the variance left out would correspond to "noise", i.e. erroneous annotations [33]. Then, they applied the inverse of their transformation to obtain a reconstructed label matrix and compared it with the original [33]. Other work used semantic similarity between genes to identify genes with incomplete annotations and complete them both within and across species [66]. However, as mentioned previously, we do not deal with this problem, as our goal is rather to learn the functions of completely uncharacterized proteins, given some proteins which are assumed to be fully characterized.

1.5.2. CORRESPONDENCE BETWEEN GENES AND PROTEINS

The central dogma of molecular biology, as taught in introductory biology courses, dictates that a specific DNA sequence (a gene) gets transcribed to a messenger RNA (mRNA) which in turn is decoded by ribosomes to synthesize a specific protein. In reality, the picture is a lot more complicated. For example, alternative splicing means that the same gene can give rise to multiple mRNA's and post-translational modifications can alter the behaviour of a protein [1]. In this thesis, we have chosen to simplify the problem and ignore those differences. That means that we do try to predict functions at the gene and protein level and not at the splice variant level. This means that possibly different functions of all splice variants are pooled together as functional annotations of a single gene or protein. We similarly ignore exon-specific gene expression patterns and only look at the total expression at the gene level.

From a practical viewpoint, the way biological data are stored in databases can also create more ambiguities. For example, there are cases where two or more genes (with different gene identifiers) code for two peptides that are then joined leading to one single

protein with one protein identifier. An example for *A. thaliana* is protein HTR2 (UniProt ID: P59226) which is mapped to genes AT1G09200, AT3G27360, AT5G10390, AT5G10400 and AT5G65360. We have removed such ambiguous cases from all the experiments and analyses where they occurred.

1.6. OVERVIEW OF FOLLOWING CHAPTERS

THE rest of this thesis is structured as follows: Chapter 2 deals with obtaining hand-crafted feature representations of both proteins and GO annotations, to deal with the issues of the variable protein length as well as the large number of GO terms. We obtain a fixed-length feature vector for each protein by representing it using its sequence identity to a training set of proteins with known functions (a similarity-based representation) and show that this approach is especially useful in *Arabidopis thaliana*, outperforming a regular sequence similarity search. To deal with the large number of possible label combinations, we map the label vectors into a lower-dimensional space. We test existing mapping methods and propose novel ones that try to preserve the ontology structure and show that they lead to consistent performance improvements on the CAFA benchmarks for several function prediction algorithms.

In Chapter 3, we move from hand-crafted features to deep neural embeddings to represent proteins, motivated by the idea that unsupervised pre-training on millions of sequences can be used to learn a general feature representation that later can be used for other tasks. We use a pre-trained protein language model to generate contextual amino acid embeddings which can be converted to protein-level embeddings by averaging along the L dimension. We show that a simple classifier on these protein-level embeddings achieved competitive performance in predicting terms from the MFO, outperforming deep neural networks that use hand-crafted features. Moreover, we demonstrate that combining protein structure information (in the form of protein distance maps) with these embeddings does not lead to better performance, while it does so for simple hand-crafted features, which hints that perhaps these contextual embeddings also encode information about structure.

In Chapter 4, the focus shifts from protein sequence to gene expression data. More specifically, we deal with the problem of automatically identifying expression samples that are relevant for predicting a specific GO term, using a set of genes that are already annotated with that term. That way we deal with the issue that typically only a few experimental conditions are relevant for predicting a specific function. Nevertheless, expression databases typically contain several different conditions and task-specific manual condition selection selection is time-consuming, requires extensive domain knowledge and is often impossible due to missing metadata. We present and evaluate a metric learning model that learns a term-specific gene co-expression measure by selecting samples from conditions relevant to the prediction of that term. The main advantage of this model over the unweighted Pearson correlation is not the minor performance improvement, but mainly the interpretability of its predictions. We test that model on *A. thaliana* data and the CAFA-π benchmark for predicting cell motility and biofilm formation in *Pseudomonas aeruginosa*.

Chapter 5 continues upon the theme of network-based transductive classification, but here Protein-Protein Interaction (PPI) networks are used. The effectiveness of such

PPI networks for predicting functions has been well-demonstrated in species with a large number of known interactions such as *Homo sapiens* and *Saccharomyces cerevisiae*. In this chapter, we emphasize that such network approaches are not effective in non-model species with many unknown interactions. We then proceed to investigate ways of predicting missing edges and show that predictions from the STRING database are more efficient than a previously published deep learning model that was trained to predict PPIs from protein sequences.

Finally, Chapter 6 provides an outlook on generic short-comings of the AFP field and challenges for the future and sketches possible solutions.

REFERENCES

[1] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. Watson, *Molecular Biology of the Cell*, 4th ed. (Garland, 2002).

[2] K. Valegård, D. Hasse, I. Andersson, and L. H. Gunn, *Structure of Rubisco from Arabidopsis thaliana in complex with 2-carboxyarabinitol-1,5-bisphosphate,* Acta Crystallographica Section D **74**, 1 (2018).

[3] X. Di, L. Cao, R. K. Hughes, N. Tintor, M. J. Banfield, and F. L. W. Takken, *Structureâ€"function analysis of the fusarium oxysporum avr2 effector allows uncoupling of its immune-suppressing activity from recognition,* New Phytologist **216**, 897 (2017), https://nph.onlinelibrary.wiley.com/doi/pdf/10.1111/nph.14733 .

[4] V. V. Lunin, E. Dobrovetsky, G. Khutoreskaya, R. Zhang, A. Joachimiak, D. A. Doyle, A. Bochkarev, M. E. Maguire, A. M. Edwards, and C. M. Koth, *Crystal structure of the cora mg2+ transporter,* Nature **440**, 833 (2006).

[5] J. V. Passonneau and O. H. Lowry, *Enzymatic Analysis* (Humana Press, 1993).

[6] K. H. Young, *Yeast Two-hybrid: So Many Interactions, (in) So Little Time...*, Biology of Reproduction **58**, 302 (1998), https://academic.oup.com/biolreprod/article-pdf/58/2/302/10541339/biolreprod0302.pdf .

[7] F. Port, H.-M. Chen, T. Lee, and S. L. Bullock, *Optimized crispr/cas tools for efficient germline and somatic genome engineering in drosophila,* Proceedings of the National Academy of Sciences **111**, E2967 (2014), https://www.pnas.org/content/111/29/E2967.full.pdf .

[8] A. D. Yates *et al.*, *Ensembl 2020,* Nucleic Acids Research **48**, D682 (2019), https://academic.oup.com/nar/article-pdf/48/D1/D682/31697830/gkz966.pdf .

[9] A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, I. Tetko, U. G?ldener, G. Mannhaupt, M. M?nsterk?tter, and H. W. Mewes, *The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes,* Nucleic Acids Res. **32**, 5539 (2004).

[10] A. Cornish-Bowden, *Current iubmb recommendations on enzyme nomenclature and kinetics,* Perspectives in Science **1**, 74 (2014), reporting Enzymology Data – STRENDA Recommendations and Beyond.

[11] L. Jeske, S. Placzek, I. Schomburg, A. Chang, and D. Schomburg, *BRENDA in 2019: a European ELIXIR core data resource,* Nucleic Acids Research **47**, D542 (2018), https://academic.oup.com/nar/article-pdf/47/D1/D542/27437170/gky1048.pdf .

[12] M. Kanehisa and S. Goto, *KEGG: Kyoto Encyclopedia of Genes and Genomes,* Nucleic Acids Research **28**, 27 (2000), https://academic.oup.com/nar/article-pdf/28/1/27/9895154/280027.pdf .

[13] B. Jassal *et al.*, *The reactome pathway knowledgebase,* Nucleic Acids Research **48**, D498 (2019), https://academic.oup.com/nar/article-pdf/48/D1/D498/31697544/gkz1031.pdf .

[14] M. Ashburner *et al.*, *Gene ontology: tool for the unification of biology. The Gene Ontology Consortium,* Nat. Genet. **25**, 25 (2000).

[15] T. G. O. Consortium, *The Gene Ontology Resource: 20 years and still GOing strong,* Nucleic Acids Res. **47**, D330 (2019).

[16] R. P. Huntley, D. Binns, E. Dimmer, D. Barrell, C. O'Donovan, and R. Apweiler, *QuickGO: a user tutorial for the web-based Gene Ontology browser,* Database **2009** (2009), 10.1093/database/bap010, bap010, https://academic.oup.com/database/article-pdf/doi/10.1093/database/bap010/16728485/bap010.pdf .

[17] D. V. Klopfenstein, L. Zhang, B. S. Pedersen, F. Ramírez, A. Warwick Vesztrocy, A. Naldi, C. J. Mungall, J. M. Yunes, O. Botvinnik, M. Weigel, W. Dampier, C. Dessimoz, P. Flick, and H. Tang, *Goatools: A python library for gene ontology analyses,* Scientific Reports **8**, 10872 (2018).

[18] P. Resnik, *Using Information Content to Evaluate Semantic Similarity in a Taxonomy,* roceedings of the 14th international joint conference on Artificial intelligence - Volume 1 - IJCAI'95 **1**, 6 (1995), arXiv:9511007 [cmp-lg] .

[19] W. T. Clark and P. Radivojac, *Information-theoretic evaluation of predicted ontological annotations,* Bioinformatics **29**, i53 (2013), https://academic.oup.com/bioinformatics/article-pdf/29/13/i53/18535367/btt228.pdf .

[20] D. Lin, *An information-theoretic definition of similarity,* in *ICML* (1998).

[21] C. Zhao and Z. Wang, *GOGO: An improved algorithm to measure the semantic similarity between gene ontology terms,* Sci Rep **8**, 15107 (2018).

[22] G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan, eds., *Predicting Structured Data*, Neural Information Processing (MIT Press, Cambridge, MA, 2007).

[23] S. Theodoridis and K. Koutroumbas, *Chapter 4 - nonlinear classifiers,* in *Pattern Recognition (Fourth Edition)*, edited by S. Theodoridis and K. Koutroumbas (Academic Press, Boston, 2009) fourth edition ed., pp. 151 – 260.

[24] Z. C. Lipton, J. Berkowitz, and C. Elkan, *A critical review of recurrent neural networks for sequence learning,* (2015), arXiv:1506.00019 [cs.LG] .

[25] E. Pękalska and R. P. Duin, *Dissimilarity representations allow for building good classifiers,* Pattern Recognition Letters **23**, 943 (2002).

[26] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, *Basic local alignment search tool,* J. Mol. Biol. **215**, 403 (1990).

[27] A. M. Yip and S. Horvath, *Gene network interconnectedness and the generalized topological overlap measure,* BMC Bioinformatics **8**, 22 (2007).

[28] V. Freschi, *A graph-based semi-supervised algorithm for protein function prediction from interaction maps,* in *Learning and Intelligent Optimization*, edited by T. Stützle (Springer Berlin Heidelberg, Berlin, Heidelberg, 2009) pp. 249–258.

[29] Z. Zhang, J. Song, J. Tang, X. Xu, and F. Guo, *Detecting complexes from edge-weighted ppi networks via genes expression analysis,* BMC Systems Biology **12**, 40 (2018).

[30] V. N. Vapnik, *Statistical Learning Theory* (Wiley-Interscience, 1998).

[31] T. Joachims, *Transductive learning via spectral graph partitioning,* in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03 (AAAI Press, 2003) p. 290–297.

[32] V. Wood *et al.*, *Term matrix: A novel gene ontology annotation quality control system based on ontology term co-annotation patterns,* bioRxiv (2020), 10.1101/2020.04.21.045195, https://www.biorxiv.org/content/early/2020/04/23/2020.04.21.045195.full.pdf .

[33] P. Khatri, B. Done, A. Rao, A. Done, and S. Draghici, *A semantic analysis of the annotations of the human genome,* Bioinformatics **21**, 3416 (2005).

[34] S. Makrodimitris, R. C. H. J. van Ham, and M. J. T. Reinders, *Improving protein function prediction using protein sequence and GO-term similarities,* Bioinformatics **35**, 1116 (2019).

[35] P. Radivojac *et al.*, *A large-scale evaluation of computational protein function prediction,* Nat. Methods **10**, 221 (2013).

[36] S. F. Altschul, T. L. Madden, A. A. Sch?ffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs,* Nucleic Acids Res. **25**, 3389 (1997).

[37] L. S. Johnson, S. R. Eddy, and E. Portugaly, *Hidden Markov model speed heuristic and iterative HMM search procedure,* BMC Bioinformatics **11**, 431 (2010).

[38] L. Lan, N. Djuric, Y. Guo, and S. Vucetic, *MS-kNN: protein function prediction by integrating multiple data sources,* BMC Bioinformatics **14 Suppl 3**, S8 (2013).

[39] S. Makrodimitris, M. J. T. Reinders, and R. C. H. J. van Ham, *Metric learning on expression data for gene function prediction,* Bioinformatics **36**, 1182 (2020).

[40] N. Zhou *et al.*, *The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens,* Genome Biol. **20**, 244 (2019).

[41] M. Kulmanov, M. A. Khan, R. Hoehndorf, and J. Wren, *DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier,* Bioinformatics **34**, 660 (2018).

[42] M. Kulmanov and R. Hoehndorf, *DeepGOPlus: improved protein function prediction from sequence,* Bioinformatics **36**, 422 (2019), https://academic.oup.com/bioinformatics/article-pdf/36/2/422/31962785/btz595.pdf .

[43] R. Fa, D. Cozzetto, C. Wan, and D. T. Jones, *Predicting human protein function with multi-task deep neural networks,* PLoS ONE **13**, e0198216 (2018).

[44] A. Sureyya Rifaioglu, T. Doğan, M. Jesus Martin, R. Cetin-Atalay, and V. Atalay, *DEEPred: Automated Protein Function Prediction with Multi-task Feed-forward Deep Neural Networks,* Sci Rep **9**, 7344 (2019).

[45] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, *Decision trees for hierarchical multi-label classification,* Machine Learning **73**, 185 (2008).

[46] K. M. BORGWARDT, H.-P. KRIEGEL, S. VISHWANATHAN, and N. N. SCHRAUDOLPH, *Graph kernels for disease outcome prediction from protein-protein interaction networks,* in *Biocomputing 2007*, pp. 4–15, https://www.worldscientific.com/doi/pdf/10.1142/9789812772435_0002 .

[47] X. Li, H. Chen, J. Li, and Z. Zhang, *Gene function prediction with gene interaction networks: A context graph kernel approach,* IEEE Transactions on Information Technology in Biomedicine **14**, 119 (2010).

[48] A. Grover and J. Leskovec, *node2vec: Scalable feature learning for networks,* (2016), arXiv:1607.00653 [cs.SI] .

[49] H. Cho, B. Berger, and J. Peng, *Compact Integration of Multi-Network Topology for Functional Analysis of Genes,* Cell Syst **3**, 540 (2016).

[50] V. Gligorijevic, M. Barot, and R. Bonneau, *deepNF: deep network fusion for protein function prediction,* Bioinformatics **34**, 3873 (2018).

[51] G. Valentini, *True path rule hierarchical ensembles for genome-wide gene function prediction,* IEEE/ACM Trans Comput Biol Bioinform **8**, 832 (2011).

[52] L. Zhang, S. Shah, and I. Kakadiaris, *Hierarchical multi-label classification using fully associative ensemble learning,* Pattern Recognition **70**, 89 (2017).

[53] D. Cozzetto, D. W. Buchan, K. Bryson, and D. T. Jones, *Protein function prediction by massive integration of evolutionary analyses and multiple data sources,* BMC Bioinformatics **14 Suppl 3**, S1 (2013).

[54] C. J. Burges, *From RankNet to LambdaRank to LambdaMART: An Overview*, Tech. Rep. MSR-TR-2010-82 (2010).

[55] R. You, Z. Zhang, Y. Xiong, F. Sun, H. Mamitsuka, and S. Zhu, *GOLabeler: improving sequence-based large-scale protein function prediction by learning to rank,* Bioinformatics **34**, 2465 (2018).

[56] R. You, S. Yao, Y. Xiong, X. Huang, F. Sun, H. Mamitsuka, and S. Zhu, *NetGO: improving large-scale protein function prediction with massive network information,* Nucleic Acids Res. **47**, W379 (2019).

[57] Y. Jiang *et al.*, *An expanded evaluation of protein function prediction methods shows an improvement in accuracy,* Genome Biol. **17**, 184 (2016).

[58] I. Kahanda, C. S. Funk, F. Ullah, K. M. Verspoor, and A. Ben-Hur, *A close look at protein function prediction evaluation protocols,* GigaScience **4** (2015), 10.1186/s13742-015-0082-5, s13742-015-0082-5, https://academic.oup.com/gigascience/article-pdf/4/1/s13742-015-0082-5/25513546/13742_2015_article_82.pdf .

[59] A. M. Schnoes, D. C. Ream, A. W. Thorman, P. C. Babbitt, and I. Friedberg, *Biases in the experimental annotations of protein function and their effect on our understanding of protein function space,* PLOS Computational Biology **9**, 1 (2013).

[60] I. Plyusnin, L. Holm, and P. T?r?nen, *Novel comparison of evaluation metrics for gene ontology classifiers reveals drastic performance differences,* PLoS Comput. Biol. **15**, e1007419 (2019).

[61] N. Youngs, D. Penfold-Brown, R. Bonneau, and D. Shasha, *Negative example selection for protein function prediction: the NoGO database,* PLoS Comput. Biol. **10**, e1003644 (2014).

[62] C. Dessimoz, N. Škunca, and P. D. Thomas, *CAFA and the open world of protein function predictions,* Trends Genet. **29**, 609 (2013).

[63] P. Yang, X.-L. Li, J.-P. Mei, C.-K. Kwoh, and S.-K. Ng, *Positive-unlabeled learning for disease gene identification,* Bioinformatics **28**, 2640 (2012), https://academic.oup.com/bioinformatics/article-pdf/28/20/2640/16909278/bts504.pdf .

[64] D. M. J. Tax, *One-class classification: Concept learning in the absence of counter-examples*, Ph.D. thesis, Technische Universiteit Delft (2001).

[65] G. Fu, J. Wang, B. Yang, and G. Yu, *NegGOA: negative GO annotations selection using ontology structure,* Bioinformatics **32**, 2996 (2016).

[66] G. Yu, W. Luo, G. Fu, and J. Wang, *Interspecies gene function prediction using semantic similarity,* BMC Syst Biol **10**, 121 (2016).

2

Improving Protein Function Prediction Using Protein Sequence and GO-term Similarities

Stavros Makrodimitris, Roeland C.H.J. van Ham and Marcel J.T. Reinders

Parts of this chapter have been published in Bioinformatics Volume 35, Issue 7, 1 April 2019, Pages 1116–1124, https://doi.org/10.1093/bioinformatics/bty751, (2019) [1].

ABSTRACT

Motivation: *Most automatic functional annotation methods assign Gene Ontology (GO) terms to proteins based on annotations of highly similar proteins. We advocate that proteins that are less similar are still informative. Also, despite their simplicity and structure, GO terms seem to be hard for computers to learn, in particular the Biological Process ontology, which has the most terms (>29 000). We propose to use Label-Space Dimensionality Reduction (LSDR) techniques to exploit the redundancy of GO terms and transform them into a more compact latent representation that is easier to predict.*

Results: *We compare proteins using a sequence similarity profile (SSP) to a set of annotated training proteins. We introduce two new LSDR methods, one based on the structure of the GO, and one based on semantic similarity of terms. We show that these LSDR methods, as well as three existing ones, improve the Critical Assessment of Functional Annotation performance of several function prediction algorithms. Cross-validation experiments on Arabidopsis thaliana proteins pinpoint the superiority of our GO-aware LSDR over generic LSDR. Our experiments on A.thaliana proteins show that the SSP representation in combination with a kNN classifier outperforms state-of-the-art and baseline methods in terms of cross-validated F-measure.*

Availability and implementation: *Source code for the experiments is available at https://github.com/stamakro/SSP-LSDR.*

Supplementary information: *Supplementary data are available at Bioinformatics online.*

2.1. INTRODUCTION

Many algorithms have been proposed to predict Gene Ontology (GO) [2] terms for a protein based on defined relationships to a set of proteins already annotated with GO terms. Protein sequence is the most widely-used predictor of function. In fact, over 90% of the methods participating in the second Critical Assessment of Functional Annotation (CAFA) use sequence information [3], by calculating either sequence similarities between proteins (e.g. [4–6]), or other sequence properties, such as k-mer frequencies [6] and enrichment of certain sub-sequences in proteins performing a specific function [7]. Most automatic function prediction (AFP) methods use the "Guilt by Association" principle to predict annotations: assigning a GO term to a query protein if this GO term is present in proteins that are "similar" to the query, by some definition of similarity. For instance, the Multi-Source-kNN (MS-kNN) method [4] applies a weighted averaging of the functions of the k proteins most similar to the query, where similarity is defined as sequence similarity, co-expression or protein-protein interaction. Other methods construct networks with proteins as nodes, where "similar" proteins are connected by an edge, and annotate queries based on the functions of their neighbors in the network (e.g. [8, 9]).

An alternative view is to describe a protein with a predefined set of measurements (a feature vector) and use machine learning techniques to "learn" which proteins perform a specific function. Choosing a meaningful feature representation for proteins is, however, not trivial. The CombFunc method [10] used different data sources to generate features, which include the lowest BLAST E-value for the query, the percent identity between the query and the top BLAST hit, the fraction of co-expressed proteins that perform a given function, etc. These features are then fed to a Support Vector Machine (SVM) classifier. More recently, DeepGO [11] used a deep neural network whose feature vectors consisted of all sub-sequences of length 3 of a protein, as well as features derived by learning an embedding of Protein-Protein Interaction networks [12].

We take a different view on defining features to represent a protein. Instead of describing the protein itself (as, for example, DeepGO does), or extracting features from closely related proteins (BLAST hits or network-based features), we propose to use a feature representation of a protein based on its sequence similarity profile (SSP) to all annotated training proteins, known as a (dis)similarity-based representation in machine learning [13]. The use of similarity-based representations of proteins in the form of an SSP was introduced by Liao and Noble for predicting protein families [14], but has not been used for multi-label AFP before. Although the ability to identify functional similarities decreases with decreasing sequence similarity [15], lower sequence similarity might point towards functional dissimilarity, and thus should lower the likelihood for GO terms associated with low-similarity proteins. Hence, we wanted a feature representation, like the SSP, that exploits functional relationships across all levels of similarity, and not only at high similarity, as current AFP methods do.

In addition, the CAFA benchmarks point out that especially the Biological Process ontology (BPO) is hard to predict for many species [3]. This issue is seen with many AFP algorithms and might be partly due to the nature of the GO itself. Although intuitive for humans, GO terms remain difficult to predict with automatic methods. We suspect that this is because the GO contains so many terms, that it becomes hard to differentiate be-

tween all of them. Also, GO terms are not independent of each other, due to the Directed Acyclic Graph (DAG) structure of the GO.

A way to overcome the complexity of the GO as well as the noise in the annotations might be to reduce the number of target variables that an AFP algorithm needs to predict, by taking advantage of the redundancy imposed by the DAG. Khatri et al. used Singular Value Decomposition (SVD) to obtain principal directions in the GO-term-space and filter out noisy GO annotations of human proteins and predict novel ones [16]. Other methods inspired by text processing, use topic modelling, where GO terms are seen as words that stem from specific latent variables called topics [17]. The topics can be interpreted as a lower-dimensional representation of functions. All the above methods use dimensionality reduction to discover new annotations for proteins that already have some GO terms annotated to them.

To (also) be able to predict annotations for proteins that do not have annotations, we propose to transform the GO terms into a lower-dimensional space using Label-Space Dimensionality Reduction (LSDR) techniques and train a machine learning method in this reduced space. Two such existing LSDR methods, namely Principal Label-Space Transformation (PLST) [18] and Conditional Principal Label-Space Transformation (CPLST) [19], are both based on an SVD of the labels. Chen and Lin showed that, when combined with linear regression, CPLST slightly outperforms PLST in predicting protein localization and protein family [19].

PLST and CPLST are general methods that can in principle be applied to any multi-label problem, including AFP. However, in the case of the GO, further information is available concerning the labels, namely that they are organized in a hierarchy. Exploiting this extra piece of information is likely to lead to better dimensionality reduction and, as a result, better performance. This observation was first made by Bi and Kwok, who introduced an LSDR technique that finds latent representations in which GO terms contribute to each component in a way similar to their ancestors in the DAG [20].

The DAG-aware method by Bi and Kwok has three disadvantages. First, it does not take the distribution of the labels in the training set into account and projects only using information about a term's ancestors. Secondly, it ignores the fact that GO terms can share information even if they are not connected by an edge in the DAG. For instance, two children of a GO term typically describe two related functions. Finally, the method uses a computationally demanding algorithm to ensure that the final predictions respect the GO hierarchy.

We address these disadvantages by introducing an LSDR method that is both GO-aware and label-distribution-aware. Then, we take the notion of GO-awareness further by combining label distributions with GO term semantic similarity [21], so that similar terms are treated similarly regardless of whether they are connected in the DAG. Also, to ensure that our predictions are consistent with the GO, we simply propagate predicted annotations towards the root, which is much more efficient. In summary, we apply dimensionality reduction schemes for the GO label-space to AFP and propose two new reduction schemes that incorporate the structure of the GO. Also, we introduce a new way to represent proteins encompassing the similarity to all other proteins, the SSP.

2.2. METHODS

2.2.1. NOTATION

Let N_{train} denote the number of training proteins. These proteins are represented by a feature matrix $\mathbf{X}_{train} \in \mathbb{R}^{N_{train} \times N_{feat}}$, whose i-th row contains the feature vector $\mathbf{x}_i$ of the i-th training protein, and N_{feat} is the dimensionality of the feature vector. The GO annotations of the i-th protein are represented by a binary label vector $\mathbf{y}_i \in \{0,1\}^L$, where $y_{ij} = 1$, iff protein i is annotated with GO term j, and L is the number of GO terms (labels) in one of the three GO sub-ontologies. Further, $\mathbf{Y}_{train} \in \{0,1\}^{N_{train} \times L}$ represents the corresponding label matrix, whose i-th row contains $\mathbf{y}_i$.

Given a set of N_{test} test proteins, represented by feature matrix $\mathbf{X}_{test} \in \mathbb{R}^{N_{test} \times N_{feat}}$, we define as $\mathbf{Y}_{test} \in \{0,1\}^{N_{test} \times L}$ the matrix containing the corresponding label vectors. For test protein i, the annotations predicted by an AFP algorithm are represented by the label vector $\hat{\mathbf{y}}_i \in \{0,1\}^L$.

2.2.2. SEQUENCE SIMILARITY PROFILE (SSP)

We represent each protein i in the dataset as a vector $\mathbf{x}_i \in \mathbb{R}^{N_{train}}$ whose j-th element contains the sequence identity between i and the j-th training protein. In other words, each protein is represented by a sequence similarity profile (SSP) to all proteins in the training set. This profile is used as a feature representation in combination with a k Nearest Neighbors (kNN) classifier because it is relatively simple and efficient and does not need to be trained separately for each GO term. The posterior for each GO term l then becomes:

$$P(y_{il} = 1 | \mathbf{x}_i) = \frac{\sum_{j \in N_k^{SSP}(i)} y_{jl}}{k} \tag{2.1}$$

where $N_k^{SSP}(i)$ represents the k nearest neighbors in the training set for protein i using Euclidean distance in the SSP space.

2.2.3. LABEL-SPACE DIMENSIONALITY REDUCTION (LSDR)

In general, the LSDR workflow can be summarized as follows:

- Transform the label matrix $\mathbf{Y}$ into a lower-dimensional matrix $\mathbf{Y}' \in \mathbb{R}^{N_{train} \times L'}$, where $L' < L$, using a transformation matrix $\mathbf{T}$, so that $\mathbf{Y}' = \mathbf{Y} \cdot \mathbf{T}$.
- The new label vectors $\mathbf{y}' \in \mathbb{R}^{L'}$ are no longer necessarily binary, so a multi-variate, multi-target regression model f is trained, so that $f(x_i) \simeq \mathbf{y}'_i$.
- For every test vector $\mathbf{x} \in \mathbb{R}^{N_{feat}}$, calculate its predicted latent function representation ($\hat{\mathbf{y}}' = f(\mathbf{x})$).
- Obtain a posterior score for the test protein being annotated with each of the L GO terms using the inverse of the transformation of the first step, $\mathbf{score}(\mathbf{x}) = \hat{\mathbf{y}}' \cdot \mathbf{T}^{-1} \in \mathbb{R}^L$. Each element of the score vector contains a value (not necessarily a probability) that reflects how certain the algorithm is about the assignment of the corresponding GO term to the protein.

LSDR is incorporated into AFP by predicting $\mathbf{y}'$ instead of $\mathbf{y}$. After applying the inverse transformation to $\mathbf{y}'$ to obtain $\mathbf{y}$, to ensure that the predictions respect the GO hierarchy,

we transform the posterior score for a term l as follows:

$$score(x_{il}) = max\{score(x_{il}), \max_{t \in Children(l)} score(x_{it})\} \tag{2.2}$$

The scores are updated per GO level starting from the most distant terms to the root. This guarantees that the scores are consistent with the hierarchy.

We introduce three existing LSDR methods (PLST, CPLST and DAG) and two new ones (GOAT and SEM).

PLST: Principal Label-Space Transformation (PLST) [18] applies SVD on the label matrix $\mathbf{Y}$ (Equation 3), after transforming it so that each column has zero mean:

$$\mathbf{Y} = \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^T, \mathbf{U} \in \mathbb{R}^{N_{train} \times N_{train}}, \Sigma \in \mathbb{R}^{N_{train} \times L}, \mathbf{V} \in \mathbb{R}^{L \times L} \tag{2.3}$$

More specifically, $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices, whose columns contain the eigenvectors of $\mathbf{Y} \cdot \mathbf{Y}^T$ and $\mathbf{Y}^T \cdot \mathbf{Y}$ respectively, also known as left and right singular vectors of $\mathbf{Y}$. Σ is a diagonal matrix, whose diagonal elements correspond to the singular values of $\mathbf{Y}$, which are equal to the square roots of the eigenvalues of $\mathbf{Y} \cdot \mathbf{Y}^T$ and $\mathbf{Y}^T \cdot \mathbf{Y}$. The transformation matrix $\mathbf{T}_{PLST}$ consists of the columns of $\mathbf{V}$ corresponding to the L' largest singular values (L' principal right singular vectors).

CPLST: The Conditional Principal Label-Space Transformation (CPLST) [19] matrix $\mathbf{T}_{CPLST}$ has as columns the L' principal right singular vectors of the matrix $\mathbf{Y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$, where λ is a regularization parameter and $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T = \mathbf{X}^+(\lambda)$ is the left regularized pseudo-inverse of $\mathbf{X}$. By incorporating the term $\mathbf{X}\mathbf{X}^+(\lambda)$, CPLST takes into account the distribution of the labels as well as the distributions of the features, i.e. the labels are projected onto directions where both the variance of the labels (as in the PLST) as well as the correlation between the labels and the features is large.

DAG : In [20] a matrix $\mathbf{G}_{anc} \in \mathbb{R}^{L \times L}$ is defined, such that $G_{anc_{ij}} = 1$ iff $i = j$ or j is an ancestor of i in the DAG defined by the GO. The labels are then transformed based on the matrix $\mathbf{T}_{DAG}$ consisting of the L' principal right singular vectors of $\mathbf{G}_{anc}$. This transformation projects related GO terms towards similar directions. In their testing phase, Bi and Kwok employed an iterative algorithm to ensure that the predictions respect the GO hierarchy. Here, we obtain GO-consistent predictions by applying Equation 2.

GOAT: We defined a matrix $\mathbf{G} \in \mathbb{R}^{L \times L}$ such that $G_{ij} = 1$ iff $i = j$ or i and j are connected by an edge in the DAG and 0 otherwise. The labels are then transformed based on matrix $\mathbf{T}_{GOAT}$, consisting of the L' principal right singular vectors of $\mathbf{Y} \cdot \mathbf{G}$. This transformation attempts to maintain directions of high variance in the label space, while projecting related GO terms towards similar directions.

SEM: As an alternative approach to capture the similarity between GO terms, we used Resnik's definition of semantic similarity [22] to calculate a similarity score between each pair of GO terms (Equation 4).

$$sim(l,l') = \max_{c \in anc(l,l')} -log(P(c)) \tag{2.4}$$

where $anc(l,l')$ is the set of common ancestors of GO terms l and l', and $P(c)$ is the frequency of term c in the training set. All pairwise similarities between GO terms were computed and stored in a similarity matrix $\mathbf{S} \in \mathbb{R}^{L \times L}$. Finally, we applied LSDR using transformation matrix $\mathbf{T}_{SEM}$ whose columns correspond to the L' principal right singular vectors of $\mathbf{Y} \cdot \mathbf{S}$.

2.2.4. BASELINE METHODS, EVALUATION AND DATASETS

We compared SSP to three baseline methods. Two are based on BLAST hits, TRANSFERBLAST and CAFABLAST, and the third is the sequence-based part of MS-kNN [4]. MS-kNN performed best in most CAFA2 benchmarks, using sequence information only for all species but human. Details of these methods are provided in Supplementary Material (SM1). We studied the effects of LSDR on SSP, TRANSFERBLAST and MS-kNN.

We compared the methods using the protein-centric F-measure (F_p), Area Under the Precision-Recall Curve ($AUPRC_p$) and Semantic Distance (SD_p) and the term-centric F-measure (F_t) and Area Under the ROC curve ($AUROC_t$). Definitions of these metrics can be found in SM2.

We tested the methods on three different datasets: the targets of CAFA2 from the 9 species with the most target proteins (SM3), the preliminary CAFA3 A. thaliana targets released by the organizers (SM4), and the dataset containing all A. thaliana proteins with experimental and/or computational annotations (cross-validation dataset, SM5). In all cases we trained and tested all methods only on the target species (intra-proteome annotation), as motivated in SM6 and Figure 2.S16. The parameters of all algorithms were optimized for each dataset using cross-validation (SM7).

2.3. RESULTS

2.3.1. LSDR IMPROVES CAFA PERFORMANCE

We evaluated the effect of the five LSDR techniques on SSP, MS-kNN and TRANSFERBLAST (Table 2.1 and Figures 2.S1-5, SM3) using 860 proteins from the CAFA2 data. LSDR improves the performance of all three algorithms for all metrics except for F_t, where the improvement is evident only for TRANSFERBLAST. GO-aware LSDR techniques achieve slightly better performance on SD_p, a metric that rewards more meaningful predictions and punishes shallow annotations, with SEM achieving the best SD_p for all the methods, although the differences are small.

We also tested our methods on the preliminary CAFA3 dataset for Arabidopsis, which is the species with the most targets in this test set (137) (SM4, Figures 2.S6-S10 and Table 2.S1). The results follow a pattern similar to that of CAFA2, with LSDR considerably improving SD_p and $AUROC_t$ for all three algorithms. For most cases, any AFP algorithm combined with LSDR performs at least on par with, if not better, than the standalone AFP algorithm. It should be noted that SSP – with or without LSDR - achieves much better

Table 2.1: CAFA2 performance (rounded to two decimal places) of the tested algorithms (rows) on the 860 targets based on five different evaluation metrics (columns), as well as the 95% confidence intervals after 1,000 bootstraps. The top performances for each metric are shown in bold. An upwards-pointing arrow next to the metric means that higher values are better and a downwards-pointing arrow that lower values are better.

LSDR	AFP method	$F_p \uparrow$	$AUPRC_p \uparrow$	$SD_p \downarrow$	$F_t \uparrow$	$AUROC_t \uparrow$
-	CAFABLAST	0.14 [0.129, 0.144]	0.15 [0.144, 0.159]	27.36 [29.05, 39.02]	0.04 [0.043, 0.047]	0.75 [0.723, 0.746]
None	SSP	0.27 [0.263, 0.280]	0.29 [0.287, 0.306]	25.07 [27.01, 34.70]	0.04 [0.038, 0.044]	0.55 [0.548, 0.564]
	MS-kNN	0.27 [0.263, 0.279]	0.31 [0.300, 0.321]	25.05 [27.16, 34.93]	0.03 [0.036, 0.040]	0.55 [0.548, 0.563]
	TRANSFERBLAST	0.15 [0.138, 0.156]	0.06 [0.060, 0.072]	37.64 [39.83, 50.89]	0.02 [0.025, 0.027]	0.5 [0.502, 0.507]
PLST	SSP	0.28 [0.272, 0.289]	**0.33** [0.317, 0.336]	24.9 [27.04, 34.78]	0.03 [0.038, 0.043]	**0.77** [0.737, 0.764]
	MS-kNN	**0.29** [0.277, 0.294]	**0.33** [0.320, 0.341]	24.72 [26.83, 34.42]	0.03 [0.034, 0.039]	**0.77** [0.744, 0.768]
	TRANSFERBLAST	0.24 [0.233, 0.251]	0.3 [0.292, 0.310]	26.98 [29.18, 38.01]	0.04 [0.041, 0.046]	0.75 [0.715, 0.738]
CPLST	SSP	0.28 [0.273, 0.289]	**0.33** [0.317, 0.336]	24.87 [27.06, 34.76]	0.04 [0.040, 0.045]	**0.77** [0.742, 0.768]
	MS-kNN	**0.29** [0.277, 0.293]	**0.33** [0.320, 0.341]	24.72 [26.85, 34.45]	0.03 [0.034, 0.039]	**0.77** [0.745, 0.769]
	TRANSFERBLAST	0.24 [0.237, 0.253]	0.3 [0.290, 0.310]	27.11 [28.91, 38.78]	0.04 [0.044, 0.048]	**0.77** [0.739, 0.764]
DAG	SSP	0.27 [0.265, 0.282]	0.32 [0.311, 0.330]	25.15 [27.13, 34.81]	0.03 [0.032, 0.037]	0.76 [0.734, 0.757]
	MS-kNN	0.28 [0.275, 0.291]	**0.33** [0.319, 0.339]	24.76 [26.93, 34.54]	0.03 [0.031, 0.036]	**0.77** [0.741, 0.763]
	TRANSFERBLAST	0.2 [0.190, 0.204]	0.24 [0.230, 0.251]	26.92 [28.95, 38.01]	0.02 [0.025, 0.030]	**0.77** [0.740, 0.762]
GOAT	SSP	0.28 [0.272, 0.289]	**0.33** [0.317, 0.336]	24.9 [27.04, 34.80]	0.03 [0.038, 0.043]	**0.77** [0.737, 0.763]
	MS-kNN	**0.29** [0.277, 0.293]	**0.33** [0.320, 0.341]	24.71 [26.83, 34.44]	0.03 [0.034, 0.039]	**0.77** [0.739, 0.762]
	TRANSFERBLAST	0.24 [0.229, 0.247]	0.3 [0.289, 0.309]	26.97 [29.16, 37.99]	0.04 [0.041, 0.046]	0.75 [0.723, 0.746]
SEM	SSP	0.28 [0.277, 0.293]	**0.33** [0.321, 0.341]	24.68 [26.62, 34.34]	0.04 [0.043, 0.047]	**0.77** [0.739, 0.765]
	MS-kNN	**0.29** [0.277, 0.293]	**0.33** [0.321, 0.341]	**24.65** [26.56, 34.27]	0.03 [0.038, 0.043]	**0.77** [0.747, 0.770]
	TRANSFERBLAST	0.28 [0.275, 0.290]	**0.33** [0.318, 0.337]	24.76 [26.80, 34.37]	0.04 [0.042, 0.048]	**0.77** [0.747, 0.769]

F_p and $AUPRC_p$ than the baselines in this plant-only dataset, while doing slightly worse than MS-kNN on SD_p.

2.3.2. SIMILAR CROSS-VALIDATION AND CAFA PERFORMANCES

To better compare LSDR methods among each other, we used a larger dataset containing 7,834 A. thaliana proteins with experimental or computational BPO annotations and repeated the evaluation using cross-validation (SM5, Figures 2.S11-15, Table 2.S2). SSP is the top-performing non-LSDR method in this dataset based on three out of the five evaluation metrics (F_p, $AUPRC_p$ and F_t) and is on par with the best method based on SD_p. CAFABLAST achieved the top $AUROC_t$ with SSP ranking second. Similar to the CAFA2/3 experiments, the use of LSDR, and especially GO-aware LSDR, also gives a performance boost on this A. thaliana experiment, with the exception of F_t.

Table 2.S4 shows the best performance achieved in each of the three datasets (CAFA2/3, A. thaliana) regardless of the AFP method. The F-measure and the AUPRC are affected most by the fraction of positive examples in each class [23], which varies a lot among the datasets. Therefore, comparing the values of these metrics is not straightforward. The same holds for SD_p, as the Information Content (IC) of terms also changes per dataset. The $AUROC_t$ is the only metric that allows for a fairer comparison. Based on that, CAFA2 and CAFA3 performances are similar to that in the cross-validation dataset as obtained with cross-validation (0.77, 0.7 and 0.73 respectively).

2.3.3. RANKING OF METHODS ROBUST TO EVALUATION SET-UP

Since a direct comparison of the performances of methods across datasets is not trivial, we investigated how sensitive the relative ranking of methods was to different evaluation schemes. We calculated the rank correlation between the performances of all AFP-LSDR combinations across the three different datasets (SM8, Figures 2.S17-2.S21). For all metrics except for F_t, the correlations were positive, although not always statistically significant. The CAFA3 and cross-validation datasets demonstrate the highest similarity, as they contain proteins from the same species. Furthermore, in our CAFA2 results there were a lot of ties which were not observed in the other two datasets and that is bound to have a negative effect on the rank correlations.

Since the CAFA3 dataset (containing only experimental annotations) and the cross-validation dataset (containing in addition computational annotations) gave a similar ranking of the methods, we tested what happens if we even more annotations were used. We additionally included IEA annotations obtained from UniProt keywords in our cross-validation dataset. Adding these annotations increased the performance of all methods. The increases were statistically significant for three out of the five metrics (SM9) but the relative ranking of the methods was not affected (rank correlations > 0.7 SM9), again supporting that LSDR improves AFP.

2.3.4. TUNING OF LSDR PARAMETERS

For all methods and datasets, parameters were tuned to optimize for $AUPRC_p$. We repeated the cross-validation experiments on A. thaliana while optimizing the parameters for term-centric criteria and the ranking of the methods remained similar (Table 2.S5). Again, F_t was an exception to that pattern. The optimal values for parameter k of the standalone SSP and MS-kNN are similar to their LSDR variants (Table 2.S3, SM5). The optimal number of dimensions of the reduced label space (L') was found to be either 500 or 1000, reducing the number of labels at least two-fold (Table 2.S3, SM5), indicating a minimum required number of dimensions to encapsulate the expressiveness of the GO terms.

2.3.5. LSDR USEFUL REGARDLESS OF TERM INFORMATIVENESS

We were interested in whether the improvement of SSP-SEM in SD_p was because SSP-SEM performed better than SSP at more informative GO terms. We used the maximum path length to the ontology root as a proxy for the informativeness of a term. Figure 2.1 shows the distribution of the differences in $AUROC_t$'s between SSP-SEM and SSP for different path lengths of the individual terms (term informativeness). The variance is rel-

atively large at all levels. Nevertheless, SSP-SEM does perform significantly better than SSP for levels 1-11 (FDR < 0.05, Bonferroni-adjusted Wilcoxon rank sum test) with the improvement being consistently around 0.03. The differences are not significant for levels 12 and 13 (FDR = 1.0, Bonferroni-adjusted Wilcoxon rank sum test), but these levels also contain only a few terms. From this we conclude that LSDR is on average useful regardless of the informativeness of the terms. Although intuitive, path length does not accurately capture the information content for a term. Therefore, we also checked how SSP-SEM performs with respect to SSP when ranking terms based on Resnik information content (Figure 2.S22, SM11). From that, the same conclusions can be drawn, i.e. the improvement is not depending on the information content of a term.

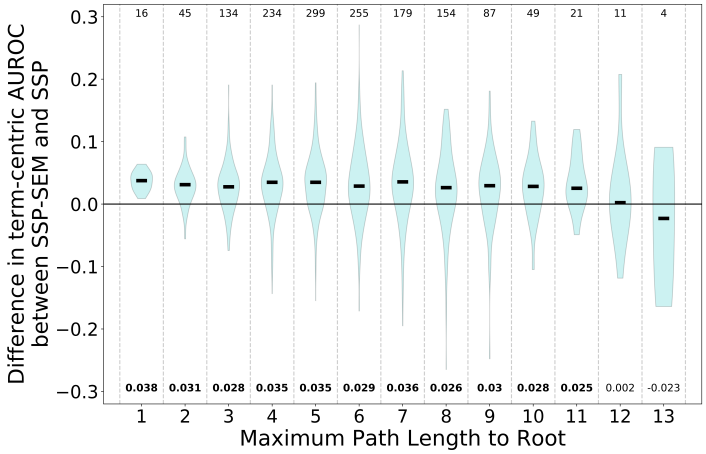


Figure 2.1: Differences in term-centric $AUROC_t$ performance of SSP-SEM with respect to SSP as a function of term specificity. At every path length from the ontology root (x axis), the distribution of the difference in $AUROC_t$ for all terms having that length is plotted using a violin plot. The medians of these differences are designated by the black stripes. The difference between the two medians is shown below the distribution and is marked in boldface if that difference is significant with an FDR of 0.05. The black horizontal line at $y = 0.0$ indicates no difference in performance. The numbers on top of the figure denote the number GO terms with that particular path length.

2.3.6. LSDR CAPTURES GO TERM CORRELATIONS

LSDR reduces the number of dimensions in the label space by exploiting the correlations between the GO terms, which potentially improves the power of the term-specific predictors (they can be learned from more examples in the created meta-terms). A schematic explanation of this is given in Figure 2.S23 (SM12). The predictions in the reduced label space do, however, not guarantee to comply with the GO hierarchy. In the cross-validation dataset, the posterior scores need to be corrected with Equation 2 for 57 to 70% of the cases depending on the LSDR method (SM13). Nevertheless, after this correction, LSDR improves the performance of all AFP methods. Interestingly, all LSDR methods generate a similar fraction of inconsistent parent-child pairs, but GO-aware ones

have a higher total fraction of inconsistencies (Table 2.S6, SM13). This implies that GO-aware LSDR methods tend to produce more confident predictions for the more specific terms than for the generic ones (Table 2.S7, SM13). Also, GO-aware LSDR can identify GO-term correlations, which cannot be done by generic methods (Tables 2.S8-2.S9, SM14).

As an example, consider GO term GO:0009798 (axis specification), which can be predicted reasonably well by both SSP and SSP-SEM, having an $AUROC_t$ of 0.71 and 0.86, respectively (averaged across the three folds). A term related to that is GO:0009956 (radial pattern formation). The two terms share a common ancestor namely GO:0007389 (pattern specification process) (Figure 2.2), which is a parent of the former and a grandparent of the latter. SSP, which essentially treats each term independently, achieves random performance on GO:0009956 ($AUROC_t$ 0.499). However, SSP-SEM performance was 0.79 for this term, as it could exploit the semantic similarity of the term with GO:0009798.

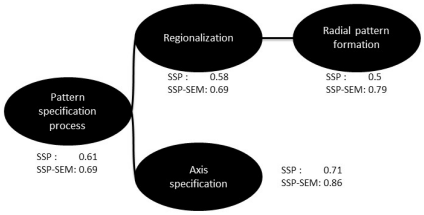


Figure 2.2: Example subgraph of the BPO. SSP achieves reasonable performance for term "axis specification", but its performance is much worse on the other branch of this subgraph. SEM can exploit the similarity of the terms in the right branch to that on the left and achieve reasonable performance for both branches.

2.3.7. SSP SELECTS MORE INFORMATIVE PROTEINS IN ARABIDOPSIS

The optimal value of parameter k, as chosen with a double cross-validation loop, was 1 for both SSP and MS-kNN (SM5, Table 2.S3), which means that both methods predict annotations for a protein based on the single most similar training protein. On average, over the three outer folds in our cross-validation experiment, the training protein that is most similar to the test protein based on the SSP, $N_1^{SSP}(i)$, is different from the protein with the highest sequence similarity, $N_1^{seq}(i)$, in 61.6% of the cases. Importantly, we found that in all three folds, when $N_1^{SSP}(i) \neq N_1^{seq}(i)$, the Jaccard similarity of the GO annotations of i and $N_1^{SSP}(i)$ was significantly larger than the Jaccard similarity of the GO annotations of i and $N_1^{seq}(i)$ (p-values = 10^{-70}, 10^{-78}, 10^{-18} Wilcoxon rank-sum test for each fold, Figure 2.S24 S24, SM15). In other words, the SSP representation selects more informative proteins than the MS-kNN to predict functional annotations in this large plant dataset.

As an example, consider query protein Q9SL78, whose best BLAST hit in Arabidopsis is P33207. These two proteins have 48% sequence identity and a BLAST E-value in the order of 10^{-74}. However, their Jaccard semantic similarity in the Biological Process ontology is 0.0. They are also dissimilar in the other two GO branches, as they are involved in different reactions and are present in different cell compartments. On the other hand, SSP chose Q5EAE9 as the nearest neighbor of Q9SL78. These two proteins have exactly

the same GO annotations: GO:0016567 and GO:0043161, and they both belong to the ATL subfamily of the RING-type zinc finger family. However, they have lower sequence similarity than the query and P33207 (39%). Their profile similarity was more similar, though. Importantly, they are both roughly equally similar (40%) to all the remaining members of the ATL subfamily, 30 of which were present in the training set in this case. As shown in Figure 2.S25 (SM16), the proteins are similar on a specific part of their sequence, which corresponds to a Zinc finger domain. Because SSP looks at the similarity to all proteins and not the most similar one, and because many members of the same subfamily were available in the training set, SSP was able to identify the correct protein in this case.

2.4. DISCUSSION

We presented new AFP methods that exploit the similarities between GO terms (LSDR techniques) and between proteins (SSP representation) to predict GO annotations for new, unannotated proteins. We benchmarked the new methods to MS-kNN [4], one of the top-performing methods in CAFA2, as well as to BLAST-based baseline methods, under different evaluation set-ups and metrics. We showed that LSDR remarkably improves CAFA performance of all tested methods in predicting BPO annotations and that SSP outperforms the baselines in A. thaliana.

2.4.1. LSDR

AFP is a special case of multi-label learning due to the additional constraint that the labels are by definition organized in a DAG structure. This is also known as structured output learning [24]. Several solutions have been proposed to tackle this problem.

Vens et al. train a separate binary decision tree classifier for every GO term, but they use as training examples only the proteins that are annotated with all the parents of the term [25]. This might lead to problems due to lack of negative examples in tree-shaped sub-branches of the DAG (where each node has exactly one parent). Other work by Zhang et al. trains independent binary predictors for each label and then attempts to reconcile conflicting predictions by solving a regression problem from the outputs of the predictors to the true labels using an L2 penalization of predictions that do not respect the hierarchy [26]. Recently, it was proposed to train a multi-label classifier for every level of the GO (i.e. first on all terms of distance 1 to the root, then all terms of distance 2 and so on) [27, 28]. This approach suffers from the lack of negative examples as well. A simpler and faster approach, which was also used here, is to force the posterior probability of a label to be greater or equal to the maximum posterior probability of all its descendants [11].

The machine learning community has produced many interesting methods on how to reduce the number of target variables in a multi-label problem (LSDR), e.g. Zitnik and Zupan used LSDR implicitly to integrate several data sources for AFP [29]. Each data source (including the label matrix Y) is represented by a lower-dimensional representation, whose dimensionality is a parameter of the algorithm. This method has high computational demands as it requires solving an optimization task during training as well as one more for every query protein [29].

The first DAG-aware LSDR work applied to AFP was by Bi and Kwok and was tested on a yeast dataset [20]. Recently, a study by Yu et al. learned a lower-dimensional embedding of the DAG structure and used it to reduce the dimensionality of the label space [30]. They then used semantic similarity in the reduced space to infer new functions for already annotated proteins. Our GOAT approach combines the benefits of the work of Bi and Kwok and those of PLST [18], by finding projections where the variance in the label space is high, but also making sure that terms connected by an edge in the DAG are "pushed" towards similar directions.

These techniques ignore the fact that terms not connected by an edge might still represent related functions. For instance, GO:0036367 (light adaptation) and GO:0009644 (response to high light intensity) are not connected by an edge, but are semantically related and they are both children of GO:0009416 (response to light stimulus). GO term semantic similarity was used by Argot2 to cluster similar GO terms into groups and then to perform predictions per group [31]. It has also been used to transform the label space by Zhang and Dai [32]. However, their transformation does not reduce the dimensionality of the label vectors.

We developed SEM, an LSDR technique that uses semantic similarity of GO terms as well as annotation patterns of proteins to reduce the number of target variables. SEM creates an annotation matrix similar to that of Zhang and Dai, but then reduces its dimensionality with SVD to create a more compact and less noisy functional representation. This method projects semantically similar terms towards similar directions in the reduced label space, even if they are not connected by an edge in the DAG. Although even GO-unaware methods were able to capture similarities between terms that are connected by an edge, SEM captured similarities between non-connected, but related terms much better than any other method (SM14).

The use of all LSDR techniques lead to consistent performance improvements for all tested AFP methods in predicting experimental annotations from CAFA2 and CAFA3. These small datasets do not show significant differences among the different LSDR techniques. However, SEM combined with SSP achieved the best performance in terms of protein-centric Semantic Distance and term-centric AUROC, while it performed on par with the best-performing methods under most of the other evaluation metrics in the much larger cross-validation dataset.

Finally, LSDR is "blurring" and simplifying the GO-term space, which is expected to have a negative effect on term-centric performance. This effect was partially observed for the term-centric F-measure, but not for the AUROC, where three methods combined with different LSDR techniques performed consistently better than the same methods without LSDR across all levels of the GO DAG.

In terms of complexity, all LSDR techniques rely on SVD, which takes $O(L^3)$, so they are fairly efficient. In the training phase, PLST requires only computing the SVD of a matrix. The other methods include an extra matrix multiplication step. The most computationally demanding method is CPLST, which also computes the pseudo-inverse of the feature matrix. The training time of LSDR techniques on the cross-validation dataset never exceeds a few minutes on a single-core machine. In the testing phase, only one matrix multiplication is required ($O(L)$), contrary to more complicated methods that solve optimization problems [20, 29, 33].

Finally, all LSDR techniques tested reduce the number of target variables by at least two-fold with either no significant loss or even gain in performance. This hints at the fact that the inherent dimensionality of a GO annotations matrix is lower than the actual number of GO terms present, which can be expected as the DAG structure enforces constraints on the annotations. It also means that the training time of more complex AFP methods can be reduced without loss of performance by applying LSDR first.

2.4.2. SEQUENCE SIMILARITY PROFILE

The rationale of SSP and MS-kNN is similar. The difference is that MS-kNN finds neighbors based on pairwise similarities, whereas SSP identifies the training proteins based on a similarity profile including all training proteins. We believe that the latter approach can be more informative, as functions can be conserved at medium levels of sequence similarity [15] and really low similarities might indicate low functional similarities. SSP is expected to be particularly useful in species in which the majority of proteins are members of expanded protein families, such as in plants [34]. In such cases, SSP can identify the shared similarity between the query and all members of the family, even if that similarity is not particularly high. Indeed, the proteins chosen by SSP had on average more similar annotations (based on Jaccard similarity) to the queries than those by MS-kNN, also explaining the difference in F-measures between the two methods in our two Arabidopsis (cross-validation and CAFA3) datasets. The two methods performed similarly in the CAFA2 dataset, hinting indeed towards the usefulness of SSP when there are many genes from the same family.

The notion of using distances or similarities as a feature representation of objects is not new [13]. In bioinformatics, the similarity-based representation of proteins is a simple way to convert arbitrary-length sequences to fixed-length vectors. It has been used for predicting protein families [14], antimicrobial peptides [35] and allergen sequences [36], as well as protein-protein interactions [37]. In addition, Li et al. used this representation along with other sequence features to predict functional annotations of viral proteins [38]. Each function was treated as a separate binary classification problem.

All these studies made use of SVMs. The reason the k-NN classifier was employed in this work is its simplicity and efficiency. In the other application domains, the classification task was binary, i.e. only discriminating between two classes. In the case of AFP, we are dealing with thousands of target classes, which would require training one SVM per class. On the contrary, k-NN is inherently multi-label and can still capture non-linear relationships in the data and have reasonable performance in diverse problems [39, 40].

From our parameter tuning, a very small value for k (1-3), the number of neighboring proteins to consider, was shown to give best performances, both for SSP and MS-kNN. This result is contradictory to the original MS-kNN study in which k was set to 20 [4], as well as the work by Yu et al. which predicted novel annotations for a partially annotated protein using the 250 or 500 most semantically similar proteins to the query [41]. Neither of these studies reported tuning of k, using cross-validation or a similar procedure, but rather they set k manually. Another reason for this difference might be the fact that MS-kNN was tested on human proteins [4] and the semantic similarity approach on human and mouse proteins [41]. As the annotations for those species are much denser than for A. thaliana, we expect that including more neighbors would tend to reinforce the

predictions of true annotations. In A. thaliana, many more annotations are missing, so true annotations might be given a low score if k is large, because they are present in only one neighbor. Perhaps this, currently, is an artefact of the fact that the other neighbors have not acquired this annotation yet.

A disadvantage of using a similarity representation is the fact that it suffers from the curse of dimensionality [42], since there are as many features as training examples. As a result, methods that use this representation are more susceptible to overfitting, especially if a complex classifier is used. For binary classification tasks, several methods have been proposed for prototypes selection, i.e. identification of a small number of proteins that are representative of the training set in order to use the similarities to only these prototypes as features [13]. In AFP, we are dealing with thousands of target classes (GO terms) instead of two and different proteins are likely to be informative for different terms. This makes prototype selection more complicated and increases the computational burden during training significantly, so we did not explore this option further. However, we did attempt randomly selecting a smaller number of prototypes [43], leading to a significant deterioration in performance (Figures 2.S26-30, SM17).

2.4.3. IMPORTANCE OF PROPER EVALUATION

Finding proper evaluation practices for AFP remains an open problem [3]. In this paper, we were interested in predicting annotations for completely unannotated proteins; No-Knowledge evaluation [44]. The CAFA challenges [3, 44] provide a unified comparison framework, where target proteins are made available at a certain time point and participants have a few months to submit their predictions on these targets. The evaluation takes place based on the subset of the targets that have accumulated experimental annotations during a predefined period of time after the submission deadline.

Experiments have shown that cross-validation performance is not an informative predictor of performance in the CAFA challenges [45]. However, it remains unclear whether cross-validation or CAFA-like experiments are more informative concerning the performance of a certain method in practice, as both of them have their own limitations. First of all, both tactics suffer from the incompleteness of annotations, which leads to potentially correct predictions of a classifier to be perceived as false positives, if the corresponding annotation has not yet been confirmed by other means. Indeed, we showed that the performance of all methods increases when annotations are more complete, i.e. in our case when we included IEA annotations. CAFA challenges are expected to suffer more from this, as they deal with only newly-derived annotations, whereas in cross-validation the test set consists of randomly selected proteins, regardless of when they received their annotations. These annotations are expected to be “more complete” on average. Also, this leads to a larger number of GO terms being available for evaluation with cross-validation.

A disadvantage of cross-validation is that stratification (i.e. maintaining an equal fraction of positive and negative examples per label across all folds) becomes virtually impossible because of the vast number of labels. As a consequence, some rare terms are left with only a handful or no positive examples. To deal with the lack of examples and its effect on performance estimation, we ignored terms that contained less than 0.1% positive examples in the test set.

Furthermore, we used the inner cross-validation loops to select the optimal parameter set of the tested predictors, using the mean protein-centric AUPRC as a performance criterion. However, due to the aforementioned issues, the AUPRC in different folds was calculated using not always the same target variables or the same target variables but with considerably different priors, which can be very misleading [46]. The different term frequencies also lead to differences in the calculation of term IC among folds. The impact of these differences needs to be investigated further, but we found that optimizing for a term-centric metric give a significantly similar ranking of the methods. We also found similarities between the ranking of methods in CAFA and cross-validation experiments.

The choice of evaluation metrics is another open problem [3], as the ranking of methods can vary based on the choice of evaluation metric [3, 44]. The short-comings of precision, recall and F-measure are well-known. They depend heavily on the class prior, which is different for each GO term and whose true value is not known in this case and can only be roughly estimated based on GO term frequencies in the training set [47]. Moreover, they can give a seemingly high performance for a method that restricts itself in general predictions, near the ontology root. SD [48] was designed to address these limitations, but it still relies on the calculation of IC of terms, which (again) needs to be estimated from the training data and can change significantly from dataset to dataset. Nevertheless, we showed that LSDR is on average useful across all levels of the GO hierarchy, i.e. independent of term frequency or IC. We also observed that term-centric F-measure behaved differently from the other four (both protein- and term-centric) metrics.

2.4.4. CONCLUSION

We used similarity in GO terms space as well as in protein sequence space to build new AFP algorithms. These algorithms improve CAFA and cross-validation performances. Consequently, we combined known concepts from machine learning and bioinformatics into new function prediction methods with encouraging results. Taken together, this shows that AFP can benefit greatly from borrowing ideas from related fields.

REFERENCES

[1] S. Makrodimitris, R. C. H. J. van Ham, and M. J. T. Reinders, *Improving protein function prediction using protein sequence and GO-term similarities,* Bioinformatics **35**, 1116 (2019).

[2] M. Ashburner *et al.*, *Gene Ontology: tool for the unification of biology,* Nature Genetics **25**, 25 (2000), arXiv:10614036 .

[3] Y. Jiang *et al.*, *An expanded evaluation of protein function prediction methods shows an improvement in accuracy,* Genome biology **17**, 184 (2016), arXiv:1601.00891 .

[4] L. Lan, N. Djuric, Y. Guo, and S. Vucetic, *MS-kNN: protein function prediction by integrating multiple data sources.* BMC bioinformatics **14 Suppl 3**, S8 (2013).

[5] Q. Gong, W. Ning, and W. Tian, *GoFDR: A sequence alignment based method for predicting protein functions,* Methods **93**, 3 (2016).

[6] D. Cozzetto, D. W. Buchan, K. Bryson, and D. T. Jones, *Protein function prediction by massive integration of evolutionary analyses and multiple data sources,* BMC Bioinformatics **14**, S1 (2013).

[7] R. Cao and J. Cheng, *Integrated protein function prediction by mining function associations, sequences, and protein-protein and gene-gene interaction networks,* Methods **93**, 84 (2016).

[8] N. Youngs, D. Penfold-Brown, K. Drew, D. Shasha, and R. Bonneau, *Parametric Bayesian priors and better choice of negative examples improve protein function prediction,* Bioinformatics **29**, 1190 (2013).

[9] Y. A. I. Kourmpetis, A. D. J. Van Dijk, M. C. A. M. Bink, R. C. H. J. Van Ham, and C. J. F. Ter Braak, *Bayesian markov random field analysis for protein function prediction based on network data,* PLoS ONE **5** (2010), 10.1371/journal.pone.0009293.

[10] M. N. Wass, G. Barton, and M. J. E. Sternberg, *CombFunc: predicting protein function using heterogeneous data sources,* Nucleic Acids Res **40**, W466 (2012).

[11] M. Kulmanov, M. A. Khan, and R. Hoehndorf, *DeepGO: Predicting protein functions from sequence and interactions using a deep ontology-aware classifier,* arXiv:1705.05919 [q-bio.GN] (2017), arXiv:1705.05919 .

[12] M. Alshahrani, M. A. Khan, O. Maddouri, A. R. Kinjo, N. Queralt-Rosinach, and R. Hoehndorf, *Neuro-symbolic representation learning on biological knowledge graphs,* arXiv:1612.04256 [q-bio.QM] (2016), 10.1093/bioinformatics/btx275, arXiv:1612.04256 .

[13] E. Pękalska and R. P. Duin, *Dissimilarity representations allow for building good classifiers,* Pattern Recognition Letters **23**, 943 (2002).

[14] L. Liao and W. S. Noble, *Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships.* Journal of computational biology : a journal of computational molecular cell biology **10**, 857 (2003).

[15] M. N. Wass and M. J. E. Sternberg, *ConFunc - Functional annotation in the twilight zone,* Bioinformatics **24**, 798 (2008).

[16] P. Khatri, B. Done, A. Rao, A. Done, and S. Draghici, *A semantic analysis of the annotations of the human genome,* Bioinformatics **21**, 3416 (2005).

[17] M. Masseroli, D. Chicco, and P. Pinoli, *Probabilistic latent semantic analysis for prediction of gene ontology annotations,* in *Neural Networks (IJCNN), The 2012 International Joint Conference on* (2012).

[18] F. Tai and H.-T. Lin, *Multilabel Classification with Principal Label Space Transformation,* Neural Computation **24**, 2508 (2012).

[19] Y. Chen and H. Lin, *Feature-aware Label Space Dimension Reduction for Multi-label Classification,* Advances in Neural Information Processing Systems , 1538 (2012).

[20] W. Bi and J. Kwok, *Multi-label classification on tree-and DAG-structured hierarchies,* in *Icml* (2011) pp. 17–24.

[21] C. Pesquita, D. Faria, A. O. Falcão, P. Lord, and F. M. Couto, *Semantic similarity in biomedical ontologies,* PLoS Computational Biology **5**, e1000443 (2009).

[22] P. Resnik, *Using Information Content to Evaluate Semantic Similarity in a Taxonomy,* roceedings of the 14th international joint conference on Artificial intelligence - Volume 1 - IJCAI'95 **1**, 6 (1995), arXiv:9511007 [cmp-lg] .

[23] D. Powers, *Evaluation: From Precision, Recall and F-Measure To Roc, Informedness, Markedness & Correlation,* Journal of Machine Learning Technologies **2**, 37 (2011), arXiv:arXiv:1011.1669v3 .

[24] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, *Support vector machine learning for interdependent and structured output spaces,* In Proc. Intl. Conf. Machine Learning. New York, NY, USA: ACM Press. ISBN **1**, (2004).

[25] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, *Decision trees for hierarchical multi-label classification,* Machine Learning **73**, 185 (2008).

[26] L. Zhang, S. K. Shah, and I. A. Kakadiaris, *Hierarchical Multi-label Classification using Fully Associative Ensemble Learning,* Pattern Recognition **70**, 89 (2017).

[27] R. Cerri *et al.*, *Reduction strategies for hierarchical multi-label classification in protein function prediction,* BMC Bioinformatics **17**, 373 (2016).

[28] A. S. Rifaioglu, T. Doğan, M. J. Martin, R. Cetin-Atalay, and M. V. Atalay, *Multi-task Deep Neural Networks in Automated Protein Function Prediction,* arXiv:1705.04802 [q-bio.QM] (2017), arXiv:1705.04802 .

[29] M. Žitnik and B. Zupan, *Data fusion by matrix factorization,* IEEE Transactions on Pattern Analysis and Machine Intelligence **37**, 41 (2015), arXiv:1307.0803 .

[30] G. Yu, Y. Zhao, C. Lu, and J. Wang, *HashGO: Hashing Gene Ontology for protein function prediction,* Computational Biology and Chemistry **71**, 264 (2017).

[31] M. Falda, S. Toppo, A. Pescarolo, E. Lavezzo, B. D. Camillo, A. Facchinetti, E. Cilia, R. Velasco, and P. Fontana, *Argot2: a large scale function prediction tool relying on semantic similarity of weighted Gene Ontology terms,* BMC Bioinformatics **13**, S14 (2012).

[32] X. F. Zhang and D. Q. Dai, *A framework for incorporating functional interrelationships into protein function prediction algorithms,* IEEE/ACM Transactions on Computational Biology and Bioinformatics **9**, 740 (2012).

[33] D. Hsu, S. M. Kakade, J. Langford, and T. Zhang, *Multi-Label Prediction via Compressed Sensing,* Learning , 1 (2009), arXiv:0902.1284v2 .

[34] S. Lockton and B. S. Gaut, *Plant conserved non-coding sequences and paralogue evolution,* (2005).

[35] X. Y. Ng, B. A. Rosdi, and S. Shahrudin, *Prediction of antimicrobial peptides based on sequence alignment and support vector machine-pairwise algorithm utilizing LZ-complexity,* BioMed Research International **2015** (2015), 10.1155/2015/212715.

[36] H. C. Muh, J. C. Tong, and M. T. Tammi, *AllerHunter: A SVM-pairwise system for assessment of allergenicity and allergic cross-reactivity in proteins,* PLoS ONE **4**, e5861 (2009).

[37] N. Zaki, S. Lazarova-Molnar, W. El-Hajj, and P. Campbell, *Protein-protein interaction based on pairwise similarity.* BMC bioinformatics **10**, 150 (2009).

[38] J. Li, S. K. Halgamuge, C. I. Kells, and S.-L. Tang, *Gene function prediction based on genomic context clustering and discriminative learning: an application to bacteriophages.* BMC bioinformatics **8 Suppl 4**, S6 (2007).

[39] I. Saini, D. Singh, and A. Khosla, *QRS detection using K-Nearest Neighbor algorithm (KNN) and evaluation on standard ECG databases,* Journal of Advanced Research **4**, 331 (2013).

[40] T. Munisami, M. Ramsurn, S. Kishnah, and S. Pudaruth, *Plant Leaf Recognition Using Shape Features and Colour Histogram with K-nearest Neighbour Classifiers,* in *Procedia Computer Science*, Vol. 58 (2015) pp. 740–747.

[41] G. Yu, W. Luo, G. Fu, and J. Wang, *Interspecies gene function prediction using semantic similarity,* BMC Systems Biology **10**, 121 (2016).

[42] M. Köppen, *The curse of dimensionality,* 5th Online World Conference on Soft Computing in Industrial Applications (WSC5) **1**, 4 (2000).

[43] E. Pękalska, R. P. W. Duin, and P. Paclík, *Prototype selection for dissimilarity-based classifiers,* Pattern Recognition **39**, 189 (2006).

[44] P. Radivojac, W. T. Clark, T. R. Oron, A. M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, A. Ben-Hur, G. Pandey, and J. M. Yunes, *A large-scale evaluation of computational protein function prediction,* Nature Methods **10**, 221 (2013), arXiv:1510.05682 .

[45] I. Kahanda, C. S. Funk, F. Ullah, K. M. Verspoor, and A. Ben-Hur, *A close look at protein function prediction evaluation protocols.* GigaScience **4**, 41 (2015).

[46] K. Boyd, V. Santos Costa, J. Davis, and C. D. Page, *Unachievable Region in Precision-Recall Space and Its Effect on Empirical Evaluation.* Machine learning : proceedings of the International Conference. International Conference on Machine Learning **2012**, 349 (2012), arXiv:1206.4667 .

[47] S. Jain, M. White, and P. Radivojac, *Recovering True Classifier Performance in Positive-Unlabeled Learning,* AAAI (2017).

[48] W. T. Clark and P. Radivojac, *Information-theoretic evaluation of predicted ontological annotations,* Bioinformatics **29**, i53 (2013).

2.5. SUPPLEMENTARY MATERIAL

Figures 2.S1-S15 and 2.S26-S30 are available online.

2.5.1. BASELINE METHODS

We compared SSP to three existing function prediction methods:

TRANSFERBLAST: Assign to a test protein i all GO annotations from the training protein with the highest sequence similarity to i. More specifically, all protein sequences are pairwise, globally aligned using the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) to all training sequences. The similarity is then calculated as the ratio of the number of positions in the alignment that are identical (i.e. the same residue is present) over the number of aligned positions, i.e. the positions where there are no gaps. Because annotations are transferred from one hit, for all protein-term pairs, the posterior probability is either 0 or 1.

CAFABLAST: Define the posterior probability $P(y_{il} = 1|x_i)$ as the maximum sequence identity between the test protein i and all training proteins that are annotated with GO term l.

$$P(y_{il}|\mathbf{x}_i) = \max_{j \in trainingset} sim(i,j) \cdot I(y_{jl} = 1) \tag{2.5}$$

with $I(.)$ the identity function (returning 1 when its argument is true), and $sim(i,j)$ being the sequence identity between protein i and j. This baseline method is the same as the BLAST baseline used in the CAFA challenges [44].

MS-kNN : Find the k training proteins with the highest sequence identity to a test protein i, $N_k^{seq}(i)$, and calculate the posterior for each GO term l as follows:

$$P(y_{il}|\mathbf{x}_i) = \frac{1}{k} \cdot \sum_{j \in N_k^{seq}(i)} (sim(i,j) \cdot y_{jl}) \tag{2.6}$$

where $sim(i,j)$ is the sequence identity between proteins i and j, and is used to weigh the contribution of the neighbors based on their similarity to the test protein. Note that the original MS-kNN algorithm combines different data sources but here we use only its sequence similarity component.

2.5.2. DEFINITIONS OF EVALUATION METRICS

In the main document, we use the following evaluation metrics: 1) Protein-Centric F-Measure and AUPRC, 2) Protein-Centric Semantic Distance, 3) Term-Centric F-Measure and 4) Term-Centric AUROC. These metrics are defined in chapter 1:

2.5.3. EVALUATION ON CAFA2 DATA

Method: We tested our methods on the CAFA2 dataset. The test set (target sequences and their annotations) was obtained from the supplemental material of the CAFA2 paper [3]. We restricted ourselves to proteins from the nine species that had at least 15 test sequences in the BP ontology. These are: *Arabidopsis thaliana*, *Danio rerio* (zebrafish), *Dictyostelium discoideum*, *Drosophila melanogaster*, *Escherichia coli*, human, mouse, *Pseudomonas aeruginosa* and rat. We optimized and trained each combination of AFP and LSDR methods separately per species, using as a training set all sequences of that species that were available at the time of release of the CAFA2 targets and had experimental annotations in the BP ontology. We used these training sequences to tune the parameters for all models using 3-fold cross validation. A grid search on the parameter space was performed as described in Supplementary Material (SM7). Then, we trained each tuned model on the whole training set for that species and evaluated it on the 860 unseen test proteins of CAFA2. We also used the latest GO version that was available at that time. Annotations were propagated upwards in the DAG, so that if a protein was annotated with a term, it would be also annotated with all its ancestors. Although most training labels were not present in the test set, they were all used for the LSDR during training. We then pooled all posterior probabilities from the 9 species into one matrix (total #test proteins × total #GO terms) and evaluated jointly across all species. 95% Confidence Intervals were computed for all evaluation metrics using 1000 bootstraps.

Results: The results are summarized on Figures 2.S1-S5. LSDR considerably improves the performance of all tested methods, but the small test set size does not allow for further comparisons.

2.5.4. EVALUATION ON CAFA3 DATA

Method: We tested the same AFP algorithms and dimensionality reduction techniques on A. thaliana data from the CAFA3 challenge. The training sequences and the preliminary targets from the CAFA3 benchmark were downloaded from the CAFA website. The training set contains 6,077 *A. thaliana* protein sequences with experimental annotations in the BP ontology known before September 2016. In total 4,376 GO terms are present, excluding the root. We used these training sequences to tune the parameters for all models and to train our predictors. Parameter tuning, training, predictions and evaluation were done exactly as for CAFA2 (SM3). Finally, we evaluated all models on the 137 unseen test proteins. These were all *A. thaliana* proteins that were included in the preliminary benchmark dataset of CAFA3 released in June 2017. This test set contained 951 BP GO terms (excluding the root), but we report predictions only on the 923 ones that were also available in the training set.

Results: The results are summarized on Figures 2.S6-S10 and Table 2.S1. The general trend of the cross-validation experiments is replicated here as well, with LSDR considerably improving the performance of all tested methods. However, due to the small size of the test set, the confidence intervals are too large to further compare the methods.

Table 2.S1: Evaluation of the algorithms (rows) on predicting experimental annotations using the CAFA3 data based on five different evaluation metrics (columns). The numbers in brackets denote the 95% Confidence Intervals. The top performance under each metric is shown in bold. An upwards-pointing arrow next to the metric means that higher values are better and a downwards-pointing arrow that lower values are better.

LSDR	AFP method	$F_p \uparrow$	$AUPRC_p \uparrow$	$SD_p \downarrow$	$F_t \uparrow$	$AUROC_t \uparrow$
-	CAFABLAST	0.16 [0.140, 0.199]	0.27 [0.236, 0.322]	25.53 [22.38, 28.31]	0.17 [0.132, 0.227]	0.65 [0.602, 0.698]
None	SSP	**0.31 [0.259, 0.374]**	**0.37 [0.315, 0.433]**	25.89 [21.94, 27.89]	**0.28 [0.22, 0.35]**	0.66 [0.62, 0.70]
	MS-kNN	0.19 [0.15, 0.24]	0.26 [0.22, 0.31]	25.68 [22.52, 28.47]	0.12 [0.09, 0.17]	0.56 [0.53, 0.59]
	TRANSFERBLAST	0.19 [0.15, 0.24]	0.26 [0.22, 0.31]	43.29 [34.95, 45.07]	0.12 [0.08, 0.17]	0.65 [0.60, 0.70]
PLST	SSP	**0.31 [0.26, 0.38]**	0.35 [0.30, 0.42]	22.11 [19.68, 24.40]	0.27 [0.21, 0.35]	0.69 [0.63, 0.74]
	MS-kNN	0.21 [0.18, 0.26]	0.25 [0.21, 0.30]	23.12 [20.63, 25.57]	0.12 [0.09, 0.17]	0.56 [0.51, 0.61]
	TRANSFERBLAST	0.18 [0.15, 0.22]	0.22 [0.18, 0.27]	22.29 [19.97, 24.59]	0.10 [0.07, 0.14]	0.57 [0.54, 0.61]
CPLST	SSP	**0.31 [0.26, 0.38]**	0.36 [0.31, 0.42]	22.11 [19.70, 24.40]	0.27 [0.21, 0.35]	0.67 [0.61, 0.73]
	MS-kNN	0.21 [0.17, 0.26]	0.24 [0.20, 0.29]	23.08 [20.53, 25.51]	0.12 [0.08, 0.17]	0.53 [0.50, 0.57]
	TRANSFERBLAST	0.19 [0.16, 0.24]	0.23 [0.19, 0.28]	22.88 [20.22, 25.36]	0.18 [0.08, 0.17]	0.67 [0.51, 0.57]
DAG	SSP	0.29 [0.24, 0.35]	0.31 [0.27, 0.37]	22.28 [19.74, 24.56]	0.18 [0.15, 0.25]	0.67 [0.62, 0.72]
	MS-kNN	0.22 [0.19, 0.26]	0.25 [0.22, 0.29]	22.67 [20.15, 25.08]	0.19 [0.14, 0.24]	0.64 [0.59, 0.68]
	TRANSFERBLAST	0.19 [0.16, 0.23]	0.24 [0.20, 0.29]	22.80 [20.54, 25.14]	0.11 [0.08, 0.15]	0.56 [0.53, 0.60]
GOAT	SSP	0.30 [0.25, 0.36]	0.33 [0.29, 0.40]	**22.01 [19.51, 24.38]**	0.23 [0.18, 0.31]	0.69 [0.63, 0.74]
	MS-kNN	0.21 [0.18, 0.26]	0.24 [0.21, 0.29]	23.04 [20.47, 25.52]	0.19 [0.14, 0.24]	0.64 [0.58, 0.68]
	TRANSFERBLAST	0.19 [0.15, 0.23]	0.23 [0.19, 0.28]	22.57 0.12 [20.10, 24.89]	0.58 [0.09, 0.16]	[0.55, 0.61]
SEM	SSP	0.31 [0.26, 0.37]	0.33 [0.28, 0.39]	22.57 [20.04, 24.86]	0.27 [0.21, 0.35]	**0.70 [0.65, 0.75]**
	MS-kNN	0.21 [0.18, 0.25]	0.23 [0.20, 0.27]	23.04 [20.47, 25.48]	0.17 [0.13, 0.22]	0.64 [0.60, 0.69]
	TRANSFERBLAST	0.19 [0.16, 0.24]	0.22 [0.18, 0.27]	22.54 [20.05, 24.94]	0.12 [0.08, 0.16]	0.60 [0.56, 0.63]

2.5.5. EVALUATION ON CROSS-VALIDATION DATASET

Method: All *A. thaliana* protein sequences with at least 1 GO annotation in the BP ontology were downloaded from SwissProt, along with their corresponding annotations. The root of the BP ontology, GO:0008150, "biological process" was ignored, as it is assigned to all proteins and predicting its presence is trivial. Annotations that were assigned without any review from curators (evidence code IEA – Inferred from Electronic Annotation) were removed as well. Annotations were propagated upwards in the DAG, so that if a protein was annotated with a term, it would be also annotated with all its ancestors. In total, the dataset consisted of N = 7,834 protein sequences, annotated with L = 4,633 different GO terms. In order to be able to properly assess the performance of the algorithms, we excluded all terms that were assigned to less than 0.1% of the proteins in either the training or the test set. Note that the number of excluded terms depends on the random splitting of the data as well as the number of training and test proteins,

which are different in the inner and outer cross-validation loops.

Results: Figures 2.S11-S15 and Table 2.S2 compare the cross-validation performance of the different AFP methods and LSDR techniques on the cross-validation dataset. Table 2.S3 contains the results of parameter selection using a double-loop cross validation.

Table 2.S2: Evaluation of the algorithms (rows) on predicting experimental annotations using the cross-validation data based on five different evaluation metrics (columns). The standard deviation across the three folds is also shown. The top performance under each metric is shown in bold. An upwards-pointing arrow next to the metric means that higher values are better and a downwards-pointing arrow that lower values are better.

LSDR	AFP method	F_p	$AUPRC_p$	SD_p	F_t	$AUROC_t$
-	CAFABLAST	0.31 ± 0.006	0.49 ± 0.001	22.8 ± 0.46	0.32 ± 0.12	0.72 ± 0.003
None	SSP	**0.52 ± 0.003**	**0.57 ± 0.002**	22. ± 0.66	**0.39 ± 0.003**	0.69 ± 0.002
	MS-kNN	0.43 ± 0.003	0.48 ± 0.001	21.8 ± 0.27	0.38 ± 0.010	0.67 ± 0.002
	TRANSFERBLAST	0.43 ± 0.003	0.48 ± 0.001	30.5 ± 1.95	0.37 ± 0.009	0.66 ± 0.003
PLST	SSP	**0.52 ± 0.002**	0.55 ± 0.001	21.9 ± 0.99	0.38 ± 0.002	**0.73 ± 0.004**
	MS-kNN	0.45 ± 0.009	0.50 ± 0.007	20.7 ± 0.39	0.38 ± 0.001	0.71 ± 0.030
	TRANSFERBLAST	0.43 ± 0.003	0.46 ± 0.007	25.0 ± 0.75	0.37 ± 0.009	0.69 ± 0.002
CPLST	SSP	**0.52 ± 0.003**	0.56 ± 0.003	21.1 ± 0.62	0.38 ± 0.003	**0.73 ± 0.002**
	MS-kNN	0.45 ± 0.011	0.50 ± 0.008	21.9 ± 1.53	0.37 ± 0.010	0.70 ± 0.007
	TRANSFERBLAST	0.43 ± 0.003	0.47 ± 0.008	24.6 ± 0.79	0.37 ± 0.10	0.68 ± 0.001
DAG	SSP	0.49 ± 0.024	0.53 ± 0.005	20.6 ± 0.41	0.27 ± 0.030	0.68 ± 0.006
	MS-kNN	0.44 ± 0.007	0.49 ± 0.009	20.7 ± 0.16	0.30 ± 0.009	0.66 ± 0.002
	TRANSFERBLAST	0.42 ± 0.003	0.45 ± 0.006	24.6 ± 0.09	0.28 ± 0.008	0.64 ± 0.004
GOAT	SSP	0.43 ± 0.002	0.53 ± 0.005	20.3 ± 0.54	0.25 ± 0.007	0.72 ± 0.004
	MS-kNN	0.41 ± 0.007	0.46 ± 0.009	22.1 ± 0.10	0.31 ± 0.010	0.67 ± 0.001
	TRANSFERBLAST	0.28 ± 0.170	0.44 ± 0.009	24.8 ± 0.82	0.24 ± 0.140	0.68 ± 0.002
SEM	SSP	0.46 ± 0.002	0.53 ± 0.005	**19.4 ± 0.54**	0.29 ± 0.010	0.72 ± 0.003
	MS-kNN	0.43 ± 0.007	0.46 ± 0.011	21.7 ± 0.39	0.37 ± 0.010	0.67 ± 0.001
	TRANSFERBLAST	0.41 ± 0.005	0.44 ± 0.007	24.9 ± 0.22	0.37 ± 0.009	0.67 ± 0.003

2.5.6. INTER- VS. INTRA-PROTEOME FUNCTIONAL ANNOTATION

Motivation: Usually, functional annotation transfer by sequence similarity takes place from a protein to its orthologs (inter-proteome annotation), i.e. to proteins from other species with which it shares an evolutionary history. On the other hand, proteins from the same species with similar sequences (paralogs) are expected to have divergent functions. However, sequence and annotation databases such as SwissProt are heavily biased towards certain model organisms. Specifically, *Homo sapiens* (human), *Mus musculus* (mouse), *Rattus norvegicus* (rat) and *Saccharomyces cerevisiae* (yeast) comprise over 9% of the total number of sequences. The bias is even more prominent when it comes to the number of GO annotations in SwissProt proteins per organism, where human and mouse proteins combined contain almost 50% of the annotations. Consequently, we expect a large effect on algorithms that use sequence similarity to predict functions in species, such as *A. thaliana*, which is evolutionarily distant to all other model organisms. Due to the biases towards animals in both the number of sequences and the number of annotations, many of the SwissProt BLAST hits for *A. thaliana* proteins are expected to be animal proteins with probably diverged functions. That would increase the number of false positives in the predictions and thereby decrease performance. Also, plant-specific functions are harder to predict for the same reason, and a high false negative rate is ex-

Table 2.S3: Optimal parameters of every combination of LSDR technique and AFP algorithm, determined with cross-validation on the cross-validation dataset.

LSDR	AFP method	k of kNN	Number of labels	CPLST λ
None	SSP	1	2,066	-
	MS-kNN	1	2,066	-
	TRANSFERBLAST	-	2,066	-
PLST	SSP	1	500	-
	MS-kNN	3	1,000	-
	TRANSFERBLAST	-	1,000	-
CPLST	SSP	1	500	10
	MS-kNN	3	1,000	0.5
	TRANSFERBLAST	-	1,00	10
DAG	SSP	1	500	-
	MS-kNN	1	1,000	-
	TRANSFERBLAST	-	500	-
GOAT	SSP	3	1,000	-
	MS-kNN	3	1,00	-
	TRANSFERBLAST	-	1,00	-
SEM	SSP	3	1,00	-
	MS-kNN	1	1,00	-
	TRANSFERBLAST	-	1,00	-

Table 2.S4: The best performance that could be achieved by any method on each dataset for the five evaluation metrics. An upwards-pointing arrow next to the metric means that higher values are better and a downwards-pointing arrow that lower values are better.

Dataset	F_p	$AUPRC_p$	SD_p	F_t	$AUROC_t$
Cross-validation	0.52	0.57	19.4	0.36	0.73
CAFA3	0.31	0.37	22.01	0.28	0.70
CAFA2	0.29	0.33	24.65	0.04	0.77

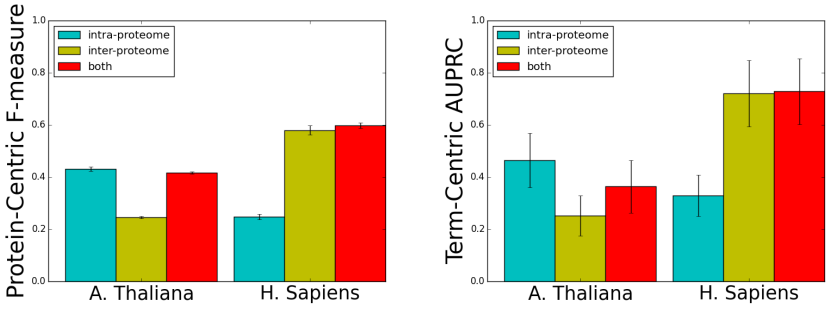
pected for such GO terms. So, although proteins from other species that have sequence similarity give accurate function predictions for animals, this might not be the case for *A. thaliana*. Since *A. thaliana* is relatively well-annotated itself, we choose to predict functions using similar proteins from within *A. thaliana* (intra-proteome annotation), which is expected to result in better predictions.

Method: We tested the annotation transfer using BLAST for *Arabidopsis thaliana* and human protein sequences. The data consisted of all *A. thaliana* proteins in SwissProt with at least one annotation in the BP ontology as of September 2016 and all *H. sapiens* proteins in SwissProt with at least one annotation in the BP ontology as of January 2017. BLAST was applied using the default settings against all entries in the SwissProt database as of January 2017. We randomly selected 3,000 *A. thaliana* and 3,000 *H. sapiens* proteins that had at least one intra- and at least one inter-proteome BLAST hit.

A 3-fold cross-validation scheme was applied. For both species, the proteins were divided into three equal parts. In each fold, one of the three parts was used as the test set.

For intra-proteome annotation transfer, the remaining two thirds of the dataset acted as the training set. For inter-proteome annotation transfer, the training set was, for all folds, all sequences in the SwissProt database except for those that came from the same species as the one that was tested. We also tested the performance when combining inter- and intra-proteome data, in which case the training set was obtained by concatenating the training sets of the two previous experiments.

Results: The inter-proteomic data significantly decreases both the protein-centric F-measure and the term-centric AUPRC in *A. thaliana*, while it significantly increases it in *H. sapiens*, compared to intra-proteomic data (Figure 2.S16). Similar results hold for the Molecular Function ontology (data not shown). Furthermore, the 4 GO terms for which inter-proteomic data achieved the lowest AUPRC in A. thaliana were GO:0009585 (red, far-red light phototransduction), GO:0010451 (floral meristem growth), GO:0048462 (carpel formation) and GO:0048455 (stamen formation), which include 3 plant-related terms. This hints at the difficulty of obtaining interesting predictions for plant-specific functions using inter-proteomic data. Conclusion. These results support our choice to consider AFP from an intra-species perspective for *A. thaliana* proteins.



(a) Protein-Centric F_{max}

(b) Term-Centric AUPRC

Figure 2.S16: Comparison of the performance of inter- and intra-proteome data for GO annotation transfer using BLAST for A. thaliana and human. The y-axis corresponds to the protein-centric F-measure in (a) and the term-centric AUPRC in (b). Intra-proteome performance is designated by the cyan bars, while intra-proteome by the yellow bars. The combined performance of both data types is shown using red bars.

2.5.7. PARAMETER OPTIMIZATION

For the cross-validation dataset, we used a double 3-fold cross-validation loop to tune the parameters of the models and to evaluate their performance on unseen data. The model parameters are: k for the SSP and MS-kNN methods, the number of reduced dimensions L' for each LSDR technique, and λ for CPLST. The inner loop was used to select the optimal parameters of each method and the outer one to evaluate the method's performance on unseen data. The parameter space was explored using a grid search in the range 1-30 for k. For L', a range 10 to 2000 was investigated using a logarithmic grid, and for λ, a range 0.001 to 10, also with a logarithmic grid. The protein-centric Area Under the Precision-Recall Curve (SM2) was used as a criterion to select the optimal parame-

ter settings in the inner loops. In SM10, we show that optimizing for term-centric area under the ROC curve does not influence the relative ranking of the methods.

For each CAFA experiment, we used a single 3-fold cross-validation loop to tune the parameters in the training set. The grid search was performed in the same way as above.

2.5.8. COMPARISON OF EVALUATION SCHEMES

Method: As discussed in the main document, a direct comparison of the performances of different algorithms in different datasets is not straightforward, because changes in the frequencies and the information content of GO terms have an important impact on the values and distribution of metrics such as the F-measure, AUPRC and SD. Instead, we investigated whether the ranking of the algorithms stayed the same across different datasets given a performance metric. For each of the three datasets (cross-validation, CAFA2, CAFA3) we treated the performance of each AFP-LSDR combination as an observation, giving 19 observations in total (3 AFP methods time 6 LSDR techniques – including no LSDR – plus CAFABLAST). Then, we calculated the spearman rank correlation between each pair of datasets across these 19 observations. We repeated this five times, once for each evaluation metric. The p-values obtained from the correlations were corrected for multiple testing using the Benjamini-Hochberg method.

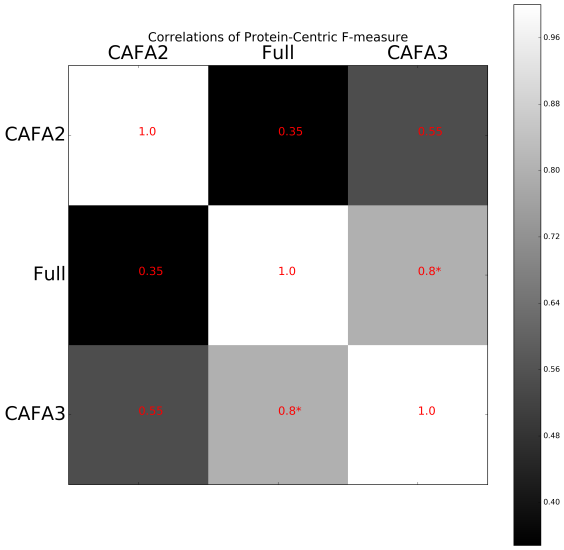


Figure 2.S17: Pairwise spearman rank correlations of the protein-centric F-measures for the three datasets. A brighter color denotes higher correlation. The value of the correlation is shown in red inside each cell. A star (*) next to the number denotes that the correlation is statistically significant with a False Discovery Rate of 0.1.

Results: The correlations are summarized in Figures 2.S17–2.S21. The CAFA3 performances are in most cases highly correlated with the cross-validation ones, although this is not always significant. On the other hand, CAFA2 performances appear to be less correlated to the other two. This is mainly caused by the fact that there many ties in the CAFA2 performances (many methods achieved exactly the same performance, Table 2.1, main document). Also, the CAFA2 data are different in that they come from 9 different species, whereas the other datasets include A. thaliana proteins only. However, most of the correlation values remain positive and there are no significantly anticorrelated cases. Conclusion: Overall, we can conclude that although the exact performance of a method in a CAFA experiment cannot be predicted by cross-validation experiments, the two different evaluation modes give similar results in terms of the ranking of the evaluated methods.

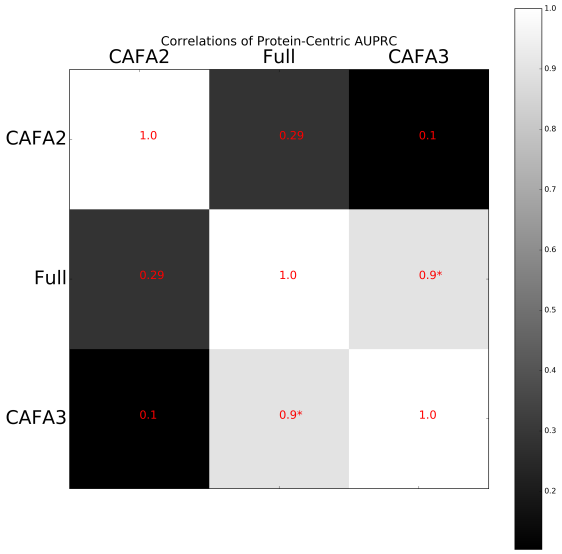


Figure 2.S18: Pairwise spearman rank correlations of the protein-centric AUPRC's for the three datasets. A brighter color denotes higher correlation. The value of the correlation is shown in red inside each cell. A star (*) next to the number denotes that the correlation is statistically significant with a False Discovery Rate of 0.1.

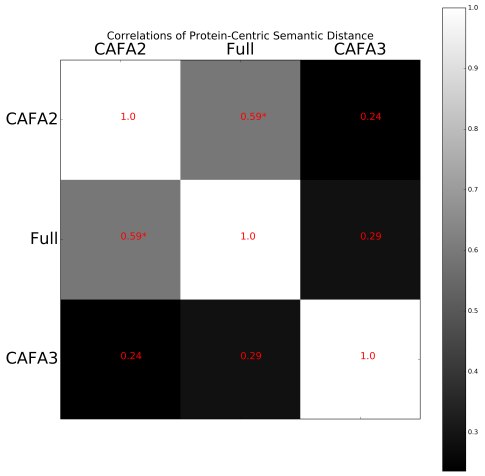


Figure 2.S19: Pairwise spearman rank correlations of the protein-centric Semantic Distances for the three datasets. A brighter color denotes higher correlation. The value of the correlation is shown in red inside each cell. A star (*) next to the number denotes that the correlation is statistically significant with a False Discovery Rate of 0.1.

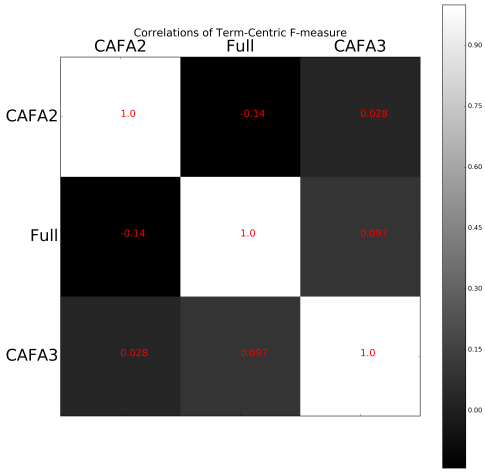


Figure 2.S20: Pairwise spearman rank correlations of the term-centric F-measures for the three datasets. A brighter color denotes higher correlation. The value of the correlation is shown in red inside each cell. A star (*) next to the number denotes that the correlation is statistically significant with a False Discovery Rate of 0.1.

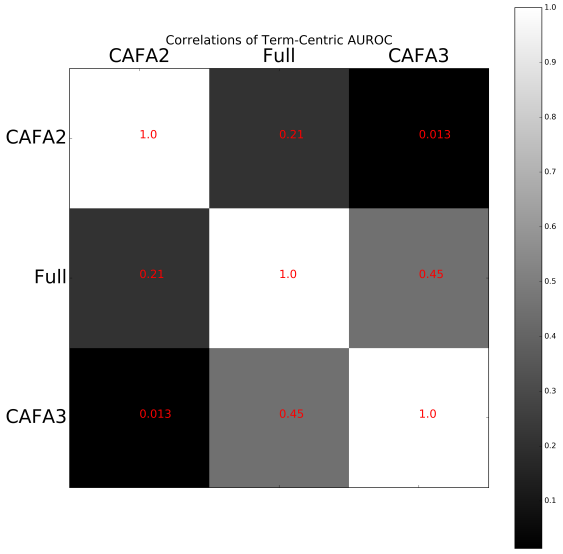


Figure 2.S21: Pairwise spearman rank correlations of the term-centric AUROCs for the three datasets. A brighter color denotes higher correlation. The value of the correlation is shown in red inside each cell. A star (*) next to the number denotes that the correlation is statistically significant with a False Discovery Rate of 0.1.

2.5.9. IEA ANNOTATIONS

Method: LSDR methods are expected to work better if more annotations are available, as with more annotations we can obtain a more accurate overview of the label space, leading to projections that are more robust and less susceptible to overfitting. To validate this hypothesis, we repeated the evaluation of the cross-validation dataset using IEA annotations. More specifically, for the proteins that were already in this dataset, we included IEA annotations that had been obtained by keyword mapping of UniProt keywords. These annotations were propagated upwards towards the ontology root. To save computational effort, we did not repeat the parameter optimization, but used the optimal parameters obtained when excluding IEA annotations (SM5, Table 2.S3).

Results: There is a high rank correlation (0.7–0.93, depending on the evaluation metric) between the methods' performances when using and not using IEA annotations. This means that the ranking of the methods is not significantly altered by adding these extra annotations. Moreover, comparing the performance of all AFP – LSDR method combinations between the two experiments under five evaluation metrics, we observed that the performance is consistently higher in the latter case. The differences are statistically significant based on the Wilcoxon rank sum test for F_p (p-value = 3.38×10^{-5}, SD_p

(p-value = $6.44 \times 10^{-6)}$) and F_t (p-value = 0.0016), but not for $AUPRC_p$ (p-value = 0.79) and $AUROC_t$ (p-value = 0.49).

Conclusion: Including IEA annotations from UniProt keywords does not affect the ranking of the tested methods. Also, since these added annotations do not come from predictions of an AFP method, but from manually curated rules, the increase in performance cannot be attributed to circular reasoning. We believe that the observed increase is composed of two components: a decrease of false positives caused by the decrease in annotation sparsity in the test set, and an overall performance gain caused by reduced overfitting of LSDR methods.

2.5.10. OPTIMIZING FOR TERM-CENTRIC METRICS

Method: All results reported previously have been obtained by tuning the parameters of every method to maximize the $AUPRC_p$. We tested whether the results (and especially protein-centric performance) would be affected if we optimized for a term-centric metric. More specifically, we repeated the double-loop cross-validation experiment, choosing the optimal parameters in the inner loop based on: a) the F_t or b) $AUROC_t$. We then evaluated in the outer loop using the five evaluation metrics, as before. It should be noted that optimizing for the F-measure is more complicated, as it relies on choosing an operating point. Here, we selected the best operating point for each combination of parameters in the inner loop and the (not necessarily the same) operating point that maximizes the test performance in the outer loop.

When optimizing for a specific metric, it is expected that the performance of most methods should increase for that metric, so the absolute values of $AUROC_t$ are expected to be overall higher when we optimize for $AUROC_t$ than when we optimize for $AUPRC_p$. To circumvent this, we did not look at the absolute performance values, but at whether the relative ranking of the different methods remains the same when we use a different optimization criterion. The performance of every combination of AFP method and LSDR technique under a specific metric was treated as a single observation and we used all observations to calculate the Spearman rank correlation between a pair of optimization criteria. That was repeated for all five metrics. We excluded TRANSFERBLAST without LSDR and CAFABLAST from this analysis, because they do not have any parameters to be tuned. The 15 p-values (5 metrics × 3 pairs of optimization criteria) that were obtained from the correlations were corrected for multiple hypotheses testing using the Benjamini-Hochberg FDR method.

Results: Table 2.S5 S5 shows the Spearman correlations of pairs of optimization criteria ($AUPRC_p$ - F_t, $AUPRC_p$ - $AUROC_t$ and $F_t - AUROC_t$) calculated across different methods based on the different metrics. We observe a positive correlation for all three protein-centric metrics (F_p, $AUPRC_p$, SD_p) between optimizing for either of the two term-cen-tric metrics and the protein-centric metric. These correlations are also statistically significant in 7 of the 9 cases. The performances under $AUROC_t$ are also highly positively correlated for all pairs of criteria (significant for 2 out 3 cases). However, the ranking of methods based on F_t when optimizing for F_t is anticorrelated to the rankings

when optimizing for the other two criteria, although these negative correlations are not statistically significant.

Conclusion: The relative comparison of the methods discussed in this work is quite robust to the choice of criterion for the parameter optimization. We have identified that the term-centric F-measure behaves differently from other metrics, but optimizing for $AUPRC_p$ or $AUROC_t$ has similar effects on both protein-centric and term-centric evaluation.

Table 2.S5: Spearman rank correlation of the performance metrics (columns) of the tested methods between two optimization criteria (rows). Statistically significant values (FDR < 0.03) are shown in bold.

	F_p	$AUPRC_p$	SD_p	F_t	$AUROC_t$
$AUPRC_p$ - F_t	**0.91**	**0.88**	**0.67**	-0.12	0.54
$AUPRC_p$ - $AUROC_t$	0.38	**0.79**	**0.66**	0.34	**0.70**
F_t - $AUROC_t$	0.26	**0.73**	**0.77**	-0.46	**0.78**

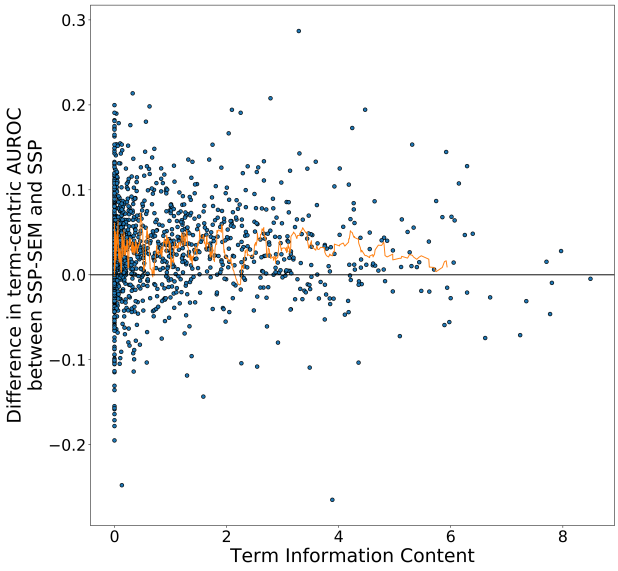


Figure 2.S22: Differences in term-centric $AUROC_t$ performance of SSP-SEM from the standard SSP algorithm as a function of Resnik term information content (blue dots). The orange line shows the local moving average of the differences using a window of 20 terms.

2.5.11. TERM-CENTRIC PERFORMANCE AS A FUNCTION OF INFORMATION CONTENT

Figure 2.S22 shows the difference in $AUROC_t$ between SSP-SEM and SSP as a function of Resnik information content. Similar to Figure 2.1 in the main text, SSP-SEM is on average better than regular SSP regardless of the informativeness of terms. The moving average across 20 nearby terms remains consistently above 0 for most of IC's.

2.5.12. THE EFFECT OF LSDR

LSDR reduces the number of target variables by exploiting the correlations and redundancies between GO terms. To illustrate this in an intuitive way, we use two hypothetical GO terms and show how PLST - which is equivalent to an SVD in label space – converts this 2-dimensional label space into a 1-dimensional one under different assumptions for the relationship between the two terms (Figure 2.S23).

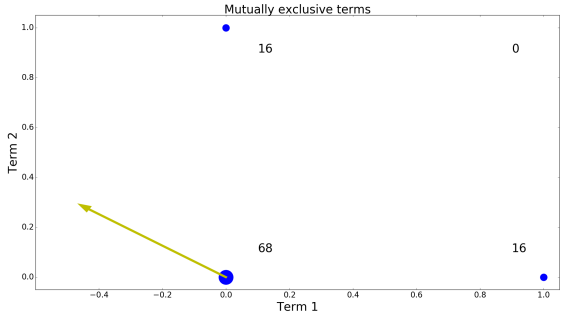
Figure 2.S23 demonstrates that PLST finds different projection vectors for mutually exclusive terms and different for related terms and can also account for differences in the frequency of the terms when doing so. This example uses two GO terms to ease visualization, but the conclusions about the behaviour of the method can be easily extended from this simple case to that of a whole ontology. The remaining LSDR methods described here perform similar transformations with the difference that they weigh label correlations differently based on a certain criterion (e.g. to maximize feature-label correlations in the projected space in the case of CPLST and to emphasize similarities of related GO terms in the case of GOAT and SEM). DAG only takes the structure of the labels into account, ignoring their frequencies and distributions.

2.5.13. ENFORCING GO HIERARCHY

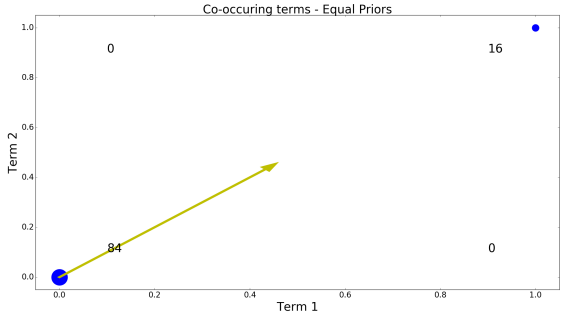
Method: From their definitions, SSP, MS-kNN and TRANSFERBLAST in their standard form are all guaranteed to make predictions that are consistent with the GO hierarchy. However, this does not hold in the cases when these algorithms are combined with LSDR, as their predictions now take place in a latent space. Prediction errors in this latent space, as well as reconstruction errors when transforming the predictions back to GO-term-space (induced by the dimensionality reduction) may lead to inconsistent predictions. To quantify the degree to which each LSDR method leads to inconsistent predictions, we used all the posterior scores of SSP, MS-kNN and TRANSFERBLAST obtained during the three cross-validation folds on the cross-validation dataset. For each LSDR method, we counted how many times a GO term had a higher posterior score than one of its immediate ancestors in the GO hierarchy (we refer to two such terms as a P-pair) across all proteins, classifiers and cross-validation folds.

As predictions get propagated all the way up the hierarchy, we also counted how many times the posterior score of a term had to be adapted to respect the hierarchy, regardless of whether the inconsistency originated at one of its immediate descendants or further down the DAG. Again, we averaged over all proteins, classifiers and cross-validation folds.

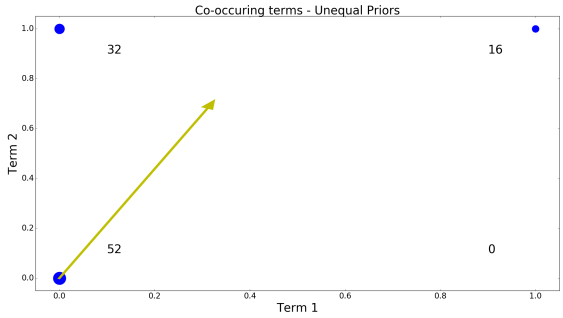
To test if an LSDR method is giving higher posterior scores on more specific terms than another, we looked –for each LSDR method- at the maximum posterior score for each term and AFP method. Then, we averaged these scores over the three AFP methods



(a)



(b)



(c)

Figure 2.S23: 100 hypothetical proteins in a label space with only two GO terms. Each dot corresponds to a set of proteins having a particular annotation (1 means the protein is annotated with the corresponding GO term and 0 that it is not) and the size of the dots reflects the number of proteins that have that annotation. These numbers are also shown in text next to the dots. The yellow arrow shows the direction of highest variation in the dataset, i.e. the projection of vector of PLST. In (a) the two terms are mutually exclusive, in (b) they are always co-occurring and have equal prior and in (c) the terms are co-occurring but term 2 is more frequent that term 1.

to obtain a single confidence score per GO term. Some LSDR methods gave on average

higher scores for all terms than others. We corrected for that by standardizing the confidence scores, so that for each LSDR method they have a mean of 0 and a standard deviation of 1 (z-score transformation). We then compared two LSDR methods by calculating the Pearson correlation between the maximum path length of terms and the difference in their z-score-transformed confidence scores between them. If there is a high positive (negative) correlation, it means that the first (second) method makes on average more confident predictions for terms further away from the root. We repeated this analysis three times, once for each cross-validation fold.

Results: Table 2.S6 summarizes the fractions of inconsistencies. For all methods a GO term has 37% probability on average to be part of an inconsistent P-pair, except for DAG where this probability increases to 43%. Our GO-aware methods (GOAT and SEM) show remarkably lower variance around that estimate than the more generic methods PLST and CPLST. What is also interesting, is that although the fraction of inconsistent pairs is similar, our GO-aware methods produce on average a larger number of total inconsistent predictions ("any pairs", Table 2.S6) than PLST and CPLST (although the variance of the latter estimates is again much higher). That implies that GO-aware methods produce on average more confident predictions for more specific terms, i.e. terms that lie further away from the ontology root.

Table 2.S6: Mean and standard deviation of the fraction of predictions that are inconsistent with the GO hierarchy for each LSDR method. The first row shows the fraction of the times that a GO term gets a higher posterior score than one of its parents. The second row shows the fraction of the times that a GO term gets a higher posterior score than any of its ancestors.

	PLST	CPLST	DAG	GOAT	SEM
P-pairs only	0.37 ± 0.14	0.36 ± 0.12	0.43 ± 0.14	0.37 ± 0.03	0.37 ± 0.02
Any pairs	0.59 ± 0.11	0.57 ± 0.1	0.68 ± 0.07	0.70 ± 0.02	0.69 ± 0.02

Table 2.S7 lists the Pearson correlations of the difference in confidence scores and maximum path length. The confidences of the predictions of GO-aware methods (GOAT, SEM and DAG) are generally higher than those of the generic ones (PLST and CPLST) for more specific terms, as denoted by the negative sign of the correlation of their differences (Table 2.S7). The effect is smaller for DAG than our two GO-aware LSDR techniques.

Conclusion: GO-aware methods make more confident predictions for more specific terms.

2.5.14. CONTRIBUTION OF SEMANTICALLY RELATED GO TERMS

Method: We investigated how LSDR methods treat semantically related terms. More specifically, we looked at pairs of terms connected by an edge in the DAG ("parent – child" pairs, P-pairs) and pairs that are not connected by an edge, but have at least one common parent ("siblings" pairs, S-pairs).

Given an LSDR method with $L' = 1{,}000$ target variables and a transformation matrix $T \in \mathbb{R}^{L \times L'}$ (Section 2.3, main document) we use $T_l \in \mathbb{R}^{L}$ to denote the l-th row of T. For a pair of terms l_1, l_2, we compared the contributions of each term to each of the L' columns

of the transformation matrix T. To assess how similarly the two terms contribute to each component, we used the Spearman correlation coefficient of the two vectors T_{l_1}, T_{l_2}. Since the cross-validation dataset was split into three folds, we averaged each T_{l_i} vector over the folds before computing the correlation. We repeated this process for every LSDR method and for all P- and S-pairs. To test whether between two different LSDR techniques, one produced a reduced space where P-pairs had significantly more highly correlated contributions, we used the paired-samples t-test.

Table 2.S7: Pearson correlation between maximum path length of a GO term and the difference in the confidence scores of that GO term between two LSDR methods (LSDR method 1 – LSDR method 2). The last column shows the FDR-corrected p-value of the correlation. Correlations with FDR < 0.05 are shown in bold.

Fold	LSDR method 1	LSDR method 2	Correlation	FDR
0	PLST	CPLST	-0.024	1.0
	PLST	GOAT	**-0.17**	1.5e-13
	PLST	SEM	**-0.19**	8.2e-17
	PLST	DAG	-0.019	1.0
	CPLST	GOAT	**-0.17**	3.2e-13
	CPLST	SEM	**-0.19**	1.0e-16
	CPLST	DAG	-0.013	1.0
	GOAT	SEM	-0.034	1.0
	GOAT	DAG	**0.16**	2.5e-11
	SEM	DAG	**0.18**	2.9e-15
1	PLST	CPLST	-0.039	0.72
	PLST	GOAT	**-0.21**	7.3e-20
	PLST	SEM	**-0.23**	3.1e-25
	PLST	DAG	-0.052	0.23
	CPLST	GOAT	**-0.2**	5.7e-19
	CPLST	SEM	**-0.23**	7.3e-25
	CPLST	DAG	-0.043	0.49
	GOAT	SEM	-0.033	1.0
	GOAT	DAG	**0.17**	3.3e-13
	SEM	DAG	**0.2**	5.0e-18
2	PLST	CPLST	**0.17**	3.2e-13
	PLST	GOAT	-0.033	1.0
	PLST	SEM	-0.05	0.25
	PLST	DAG	0.033	1.0
	CPLST	GOAT	**-0.14**	1.6e-9
	CPLST	SEM	**-0.17**	7.2e-13
	CPLST	DAG	**-0.073**	0.01
	GOAT	SEM	-0.021	1.0
	GOAT	DAG	**0.08**	0.004
	SEM	DAG	**0.12**	1.4e-6

Results: The subset of the BP ontology that was present in our cross-validation Dataset contained 4,367 P-pairs and 11,351 S-pairs. A lot of the paired tests that we conducted resulted in test statistics with such large absolute values, that the calculation of a p-value often leads to underflow. Therefore, we only report the values of the t statistic. We compared all five LSDR techniques twice with each other (once for P-pairs and once for S-pairs), so in total 20 tests were performed. With an FDR of 0.05 and Bonferroni correction, p-values lower than 0.00125 correspond to significant differences. This p-value cut-off corresponds to t statistic absolute values larger than 3.0251 for a set with 4,367 samples (P-pairs) and larger than 3.024 for a set with 11,351 samples (S-pairs).

The comparisons of LSDR methods are shown on Table 2.S8 for P-pairs and Table 2.S9 for S-pairs. PLST and CPLST exhibit the highest correlation between P-pairs and SEM between S-pairs.

Conclusion: The results point out that even GO-unaware methods can incorporate the annotation similarity between GO terms that are connected by an edge in the DAG (P-pairs). On the other hand, these methods are less successful in capturing the similarities between terms that have a common parent (S-pairs) than GO-aware methods. SEM, which was specifically designed to capture such relationships, showed the highest correlation between contributions of S-pairs.

Table 2.S8: t-statistics for the paired test with null hypothesis that the difference in correlations of contributions of P-pairs to the first 1,000 components of two LSDR methods is not different from 0. Significant values are shown in bold.

	PLST	CPLST	DAG	GOAT	SEM
PLST	-	2.84	**46.41**	**46.7**	**18.83**
CPLST	-2.84	-	**46.38**	**46.72**	**18.67**
DAG	**-46.41**	**-46.38**	-	-0.68	**-34.29**
GOAT	**-46.7**	**-46.72**	0.68	-	**-36.58**
SEM	**-18.83**	**-18.67**	**34.29**	**36.58**	-

Table 2.S9: t-statistics for the paired test with null hypothesis that the difference in correlations of contributions of S-pairs to the first 1,000 components of two LSDR methods is not different from 0. Significant values are shown in bold.

	PLST	CPLST	DAG	GOAT	SEM
PLST	-	1.66	**-14.23**	-2.28	**-32.34**
CPLST	-1.66	-	**-14.53**	-2.59	**-32.68**
DAG	**14.23**	**14.53**	-	**13.33**	**-12.73**
GOAT	2.28	2.59	**-13.33**	-	**-30.69**
SEM	**32.34**	**32.68**	**12.73**	**30.69**	-

2.5.15. SEMANTIC SIMILARITY OF NEAREST NEIGHBORS

For each test protein i for which $N_1^{SSP}(i) \neq N_1^{seq}(i)$, we assessed how similar the GO annotations of i were to those of $N_1^{SSP}(i)$ and $N_1^{seq}(i)$, using Jaccard similarity, defined in Equation S13.

$$J(i,j) = \frac{\sum_{l=1}^{L} I(y_{il} = 1 \wedge y_{jl} = 1)}{\sum_{l=1}^{L} I(y_{il} = 1 \vee y_{jl} = 1} \tag{2.7}$$

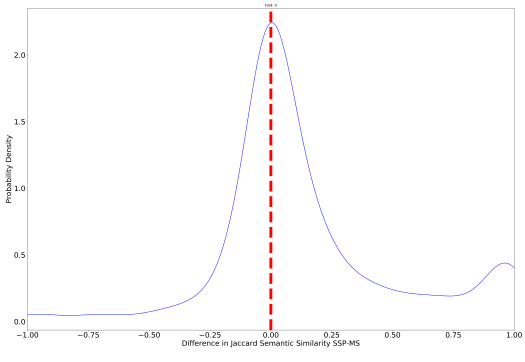
Figures 2.S24a-c show the distribution of the difference $J(i, N_1^{SSP}(i)) - J(i, N_1^{seq}(i))$ for the three cross validation folds. In order to put these results into perspective, Figures 2.S24d-f show the same comparison between the test protein with the most similar profile ($N_1^{SSP}(i)$) and a randomly chosen training protein. As expected, $N_1^{SSP}(i)$ is a significantly better choice than a random protein in the training set (p-value ≈ 0) in all three folds.

2.5.16. EXAMPLE OF SSP OUTPERFORMING BLAST

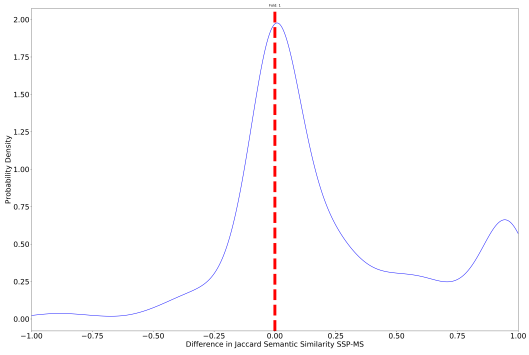
Figure 2.S25 refers to the example of section 3.7 concerning protein Q9SL78. A lot of training proteins align to this protein on a specific part of their sequence (Figure 2.S25a). The same happens for protein Q5EAE9, which has the most similar SSP to Q9SL78 (Figure 2.S25b). This short subsequence corresponds to a Zinc finger domain.

2.5.17. RANDOM PROTOTYPE SELECTION

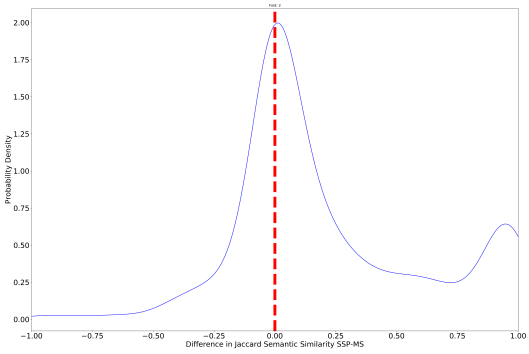
Removing proteins from the profile reduces the dimensionality of the feature vectors and, in theory, makes SSP less susceptible to overfitting. However, different proteins are informative for different GO terms, so standard feature selection methods are not applicable here. We tested the effect of removing proteins at random from SSP using the cross-validation dataset. As seen on Figures 2.S26-30, the more proteins in SSP, the better the performance, despite the curse of dimensionality.


Probability Density
Difference in Jaccard Semantic Similarity SSP-MS

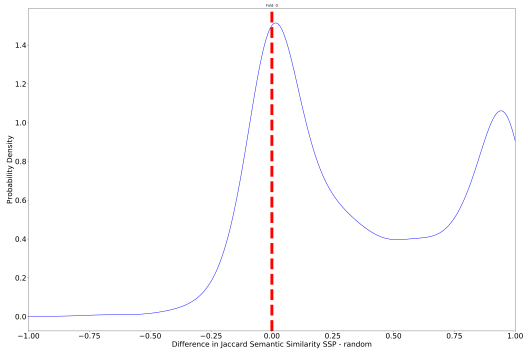
(a)


Probability Density
Difference in Jaccard Semantic Similarity SSP-MS

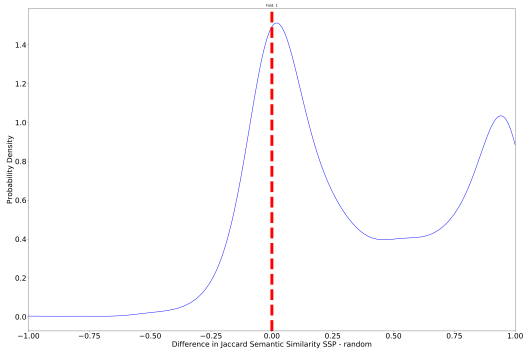
(b)


Probability Density
Difference in Jaccard Semantic Similarity SSP-MS

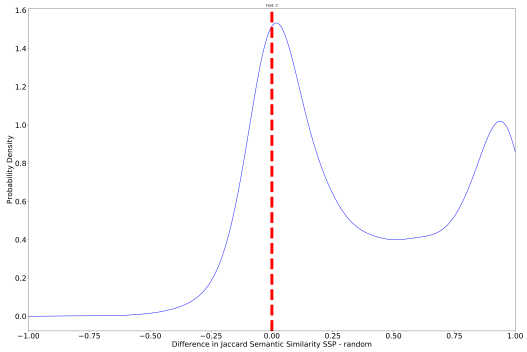
(c)



(d)

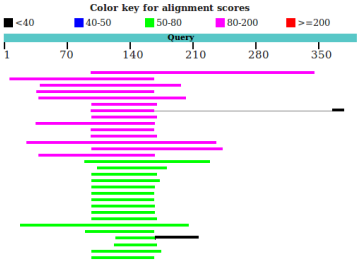


(e)

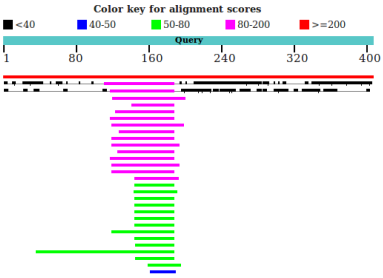


(f)

Figure 2.S24: (a-c) Distribution of the difference of the Jaccard semantic similarity of a test protein and its nearest neighbor in the SSP space minus the Jaccard semantic similarity of that protein and its nearest neighbor in the sequence space. The x = 0.0 line is marked by a red vertical line. The distribution is plotted separately for each of the three cross-validation folds. (d-f) The same but for the nearest neighbour in SSP space and a random training protein.



(a)



(b)

Figure 2.S25: (a) BLAST result of test protein Q9SL78 against the 30 members of its subfamily that were on the training set (b) BLAST result of Q5EAE9, the training protein with the most similar SSP to test protein Q9SL78 against the 30 members of its subfamily that were on the training set (including itself). Images are from https://blast.ncbi.nlm.nih.gov/Blast.cgi.

3

UNSUPERVISED PROTEIN EMBEDDINGS OUTPERFORM HAND-CRAFTED SEQUENCE AND STRUCTURE FEATURES AT PREDICTING MOLECULAR FUNCTION

Amelia VILLEGAS-MORCILLO[1], Stavros MAKRODIMITRIS[1], Roeland C.H.J. VAN HAM, Angel M. GOMEZ, Victoria SANCHEZ and Marcel J.T. REINDERS

Parts of this chapter have been published in Bioinformatics 10.1093/bioinformatics/btaa701, (2020) [1]
[1]These authors contributed equally.

ABSTRACT

Motivation: *Protein function prediction is a difficult bioinformatics problem. Many recent methods use deep neural networks to learn complex sequence representations and predict function from these. Deep supervised models require a lot of labeled training data which are not available for this task. However, a very large amount of protein sequences without functional labels is available.*

Results: *We applied an existing deep sequence model that had been pre-trained in an unsupervised setting on the supervised task of protein molecular function prediction. We found that this complex feature representation is effective for this task, outperforming hand-crafted features such as one-hot encoding of amino acids, k-mer counts, secondary structure and backbone angles. Also, it partly negates the need for complex prediction models, as a two-layer perceptron was enough to achieve competitive performance in the third Critical Assessment of Functional Annotation benchmark. We also show that combining this sequence representation with protein 3D structure information does not lead to performance improvement, hinting that three-dimensional structure is also potentially learned during the unsupervised pre-training.*

Availability: *Implementations of all used models can be found at https://github.com/stamakro/GCN-for-Structure-and-Function.*

Contact: *ameliavm@ugr.es*

Supplementary information: *Supplementary data are available at Bioinformatics online.*

3.1. INTRODUCTION

Proteins perform most of the functions necessary for life. However, proteins with a well-characterized function are only a small fraction of all known proteins and mostly restricted to a few model species. Therefore, the ability to accurately predict protein function has the potential to accelerate research in fields such as animal and plant breeding, biotechnology, and human health.

The most common data type used for automated function prediction (AFP) is the amino acid sequence, as conserved sequence implies conserved function [2]. Consequently, many widely-used AFP algorithms rely on sequence similarity via BLAST [3] and its variants or on hidden Markov models [4]. Other types of sequence information that have been used include *k*-mer counts, predicted secondary structure, sequence motifs, conjoint triad features and pseudo-amino acid composition [5–7]. Moreover, Cozzetto et al. showed that different sequence features are informative for different functions.

More recently, advances in machine learning have partially shifted the focus from hand-crafted features, such as those described above, to automatic representation learning, where a complex model -most often a neural network- is used to learn features that are useful for the prediction task at hand. Many such neural network methods have been proposed, which use a variety of architectures [8].

Some studies combined the two approaches, starting from hand-crafted features that are fed into a multi-layer perceptron (MLP) to learn more elaborate representations [6, 7]. Others apply recurrent or convolutional architectures to directly process variable-length sequences. For instance, [9] used a neural embedding layer to embed all possible amino acid triplets into a 128-dimensional space and then applied a convolutional neural network (CNN) on these triplet embeddings. Moreover, [10] and [11] trained Long Short-Term Memory (LSTM) networks to perform AFP.

The motivation behind these deep models is that functional information is encoded in the sequence in a complicated way. A disadvantage is that complex models with a large number of parameters require a large amount of training examples, which are not available for the AFP task. There are about 80,000 proteins with at least one experimentally-derived Molecular Function Gene Ontology (GO) [12] annotation in SwissProt and 11,123 terms in total.

On the other hand, a huge number of protein sequences of unknown function is available (>175M in UniProtKB). Although these sequences cannot be directly used to train an AFP model, they can be fed into an unsupervised deep model that tries to learn general amino acid and/or protein features. This learned representation can then be applied to other protein-related tasks, including AFP, either directly or after fine-tuning by means of supervised training. Several examples of unsupervised pre-training leading to substantial performance improvement exist in the fields of computer vision [13–15] and natural language processing (NLP) [16–18]. In bioinformatics, pre-training was shown to be beneficial for several deep neural network architectures on protein engineering and remote homology detection tasks [19].

A deep unsupervised model of protein sequences was recently made available [20]. It is based on the NLP model ELMo (Embeddings from Language Models) [17] and is composed of a character-level CNN (CharCNN) followed by two layers of bidirectional LSTMs. The CNN embeds each amino acid into a latent space, while the LSTMs use that embedding to model the context of the surrounding amino acids. The hidden states of

the two LSTM layers and the latent representation are added to give the final context-aware embedding. These embeddings demonstrated competitive performance in both amino acid and protein classification tasks, such as inferring the protein secondary structure, structural class, disordered regions, and cellular localization [20, 21]. Other works also trained LSTMs to predict the next amino acid in a protein sequence using the LSTM hidden state at each amino acid as a feature vector [22, 23]. Finally, a transformer neural network was trained on 250 million protein sequences, yielding embeddings that reflected both protein structure and function [24].

Protein function is encoded in the amino acid sequence, but sequences can diverge during evolution while maintaining the same function. Protein structure is also known to determine function and is -in principle- more conserved than sequence [25, 26]. From an AFP viewpoint, two proteins with different sequences can be assigned with high confidence to the same function if their structures are similar. It is therefore generally thought that combining sequence data with 3D structure leads to more accurate function predictions for proteins with known structure, especially for those without close homologues.

Structural information is often encoded as a protein distance map. This is a symmetric matrix containing the Euclidean distances between pairs of residues within a protein and is invariant to translations or rotations of the molecule in 3D space. One can obtain a binary representation from this real-valued matrix, called protein contact map, by applying a distance threshold (typically from 5 to 20 Å). This two-dimensional representation successfully captures the overall protein structure [27, 28]. The protein contact map can be viewed as a binary image, where each pixel indicates whether a specific pair of residues are in contact or not. Alternatively, it can be interpreted as the adjacency matrix of a graph, where each amino acid is a node and edges represent amino acids that are in contact with each other. In order to extract meaningful information from contact maps, both two-dimensional CNNs [29, 30] and graph convolutional networks (GCNs) [31, 32] have been proposed.

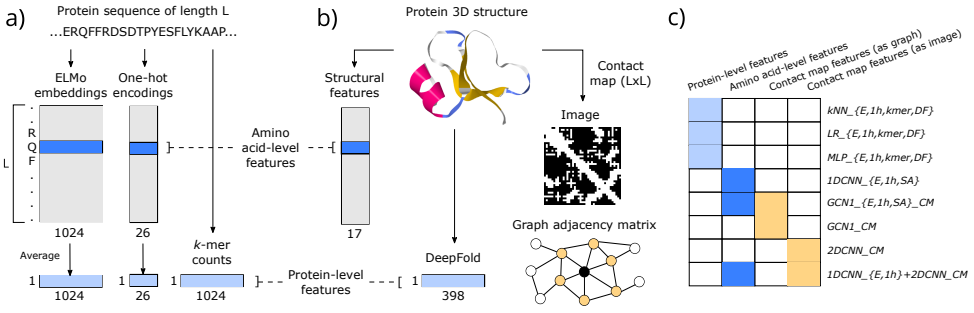


Figure 3.1: Protein representation types considered in this study, which encode (a) amino acid sequence information (ELMo embeddings, one-hot encodings, *k*-mer counts) and (b) 3D structure information encoded by secondary structure and backbone angles, the DeepFold features, or in the form of contact map (as an image or graph adjacency matrix). (c) The protein representations (columns) that are fed as input to each classification model (rows) are indicated by a shaded box, colored blue for amino acid and protein-level features, and orange for contact map representations.

Only [22] have explored the effectiveness of a pre-trained sequence model in AFP, but it was done in combination with protein structure information using a GCN. We suspect

that a deep pre-trained embedding can be powerful enough to predict protein function, in which case the structural information would not offer any significant performance improvement. Therefore, we set out to evaluate pre-trained ELMo embeddings in the task of predicting molecular functions, by comparing them to hand-crafted sequence and structural features in combination with 3D structure information in various forms. We focus on the Molecular Function Ontology (MFO), as it is the most correlated ontology to sequence and structure [33], but also perform small-scale experiments on Biological Process Ontology (BPO) and Cellular Component Ontology (CCO). Figure 3.1 provides an overview of the data and models used in our experiments. We demonstrate the effectiveness of the ELMo model [20] and show that protein structure does not provide a significant performance boost to these embeddings, although it does so when we only consider a simple protein representation based on one-hot encoded amino acids.

3.2. MATERIALS & METHODS

3.2.1. PROTEIN REPRESENTATIONS

We considered two types of representations of the proteins (Figure 3.1). The first one describes the sequence using amino acid features and the second one the three-dimensional structure, in the form of distance maps.

For each sequence of length L, we extracted *amino acid-level features* using a pre-trained unsupervised language model [20]. This model is based on ELMo [17] and outputs a feature vector of dimension d=1,024 for each amino acid in the sequence. We denote this as a matrix $\mathbf{X}^E \in \mathbb{R}^{L\times d}$. As proposed in [20], we also obtained a fixed-length vector representation of each protein (*protein-level features*, denoted as $\mathbf{x}^E \in \mathbb{R}^d$) by averaging each feature over the L amino acids.

To compare ELMo with simpler sequence representations, we used the one-hot encoding of the amino acids, denoted by the matrix $\mathbf{X}^{1h} \in \{0,1\}^{L\times d}$ with d=26. As before, we obtained a protein-level representation $\mathbf{x}^{1h} \in \mathbb{R}^d$, which contains the frequency of each amino acid in the protein sequence, completely ignoring the order. We also used a protein-level representation based on k-mer counts (with k=3,4,5). To reduce the dimensionality of this representation, we applied truncated singular value decomposition (SVD) keeping the first $d \in \{1024, 2000, 3000, 4000, 5000\}$ components ($\mathbf{x}^{kmer} \in \mathbb{R}^d$).

With respect to structural information, we considered the protein distance map. This $L \times L$ matrix contains the Euclidean distances between all pairs of beta carbon atoms (alpha carbon atoms for Glycine) within each protein chain. We used DeepFold [34] to extract a 398-dimensional protein-level feature vector from the distance map ($\mathbf{x}^{DF} \in \mathbb{R}^d$). We also converted the distance map to a binary contact map using a threshold of 10 Å. Finally, we tested an amino acid-level structural representation $\mathbf{X}^{SA} \in \mathbb{R}^{L\times d}$, with d=17 features. These features include the secondary structure (one-hot encoded 8-states 'HBEGITS-') and relative accessible surface area obtained from DSSP (Define Secondary Structure of Proteins) [35], along with the sine and cosine of the backbone angles $[\phi, \psi, \theta, \tau]$ [36].

3.2.2. FUNCTION PREDICTION METHODS

We trained and evaluated several classifiers which use the protein representations defined above (Figure 3.1). Details about the hyperparameters are provided in Supplementary Material (SM) 1 (Tables 3.S1-3.S2).

We first considered methods operating on the protein-level features (either ELMo embeddings $\mathbf{x}^E$, one-hot encodings $\mathbf{x}^{1h}$, k-mer counts $\mathbf{x}^{kmer}$, or DeepFold features $\mathbf{x}^{DF}$). As these feature vectors are of fixed size for all proteins, we can apply traditional machine learning algorithms. Here, we tested the following classifiers: k-nearest neighbors (k-NN), logistic regression (LR), and multi-layer perceptron (MLP) with one hidden layer. We denoted these models as *kNN_{E,1h,kmer,DF}*, *LR_{E,1h,kmer,DF}* and *MLP_{E,1h,kmer,DF}*, respectively.

We also trained several convolutional networks on the amino acid-level representations ($\mathbf{X}^E$, $\mathbf{X}^{1h}$ $\mathbf{X}^{SA}$) and contact maps. The architectures are composed of convolutional layers; either 1D, 2D or graph-based. As the input size is variable in the sequence dimension, these layers are followed by a global pooling operation, to obtain a fixed-size vector for each protein. This embedding vector is then used to predict the corresponding C outputs (GO terms) through fully-connected (FC) layers. In the output layer we applied the sigmoid function, so that the final prediction for each GO term is in the range [0,1]. We tested either one or two FC layers (Table 3.S3, SM1).

The one-dimensional convolutional neural network (1D-CNN) applies dilated convolutions in two layers (Figure 3.S1) and we refer to this model as *1DCNN_{E,1h,SA}*. We also benchmarked DeepGOCNN [37], a 1D-CNN with 8,192 convolutional filters of various sizes operating on one-hot encoded amino acids, followed by one FC layer. We denote this model as *DeepGOCNN_1h*.

To incorporate contact map information, we trained GCN models. In this case the protein 3D structure is viewed as a graph with adjacency matrix $\mathbf{A} \in \{0,1\}^{L\times L}$, where each amino acid of the sequence corresponds to a node and an edge between two nodes denotes that they are in contact. The graph convolution operator that we mainly used was the first-order approximation of the spectral graph convolution defined in [38] as:

$$\mathbf{X}' = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{W}, \tag{3.1}$$

where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with self-loops, $\hat{\mathbf{D}}$ the diagonal degree matrix with $\hat{D}_{i,i} = \sum_{l=1}^{L} \hat{A}_{i,l}$ and $\mathbf{W}$ the weight matrix that combines the node features. Equation (3.1) describes the diffusion of information about each amino acid to the neighboring residues, where the neighborhood is defined by the graph. We tested the model proposed by [22] that has three convolutional layers (*GCN3_{E,1h,SA}_CM*, Figure 3.S2). As we intended to use simple models, we also considered a reduced version of this network, with only one convolutional layer (*GCN1_{E,1h,SA}_CM*, Figure 3.S3).

In order to test the ability of predicting function based on contact maps alone, we evaluated two alternative approaches. The first one is based on the GCN model described above [38] keeping $\mathbf{A}$ as before, but with $\mathbf{X} \in \mathbb{R}^{L\times 1}$ containing the degree of each node as amino acid feature. Therefore, by applying the convolution operation of equation (3.1), the network only learns graph connectivity patterns (*GCN1_CM*). The second approach processes the maps as $L \times L$ images and learns image patterns using a 2D-CNN model with two convolutional layers (Figure 3.S4). We denoted this model as

2DCNN_CM.

Moreover, we investigated alternative ways of combining sequence and structure information, such as a combined 1D-CNN and 2D-CNN model that is simultaneously trained to extract a joint representation (Figure 3.S5). In this case, we concatenated the outputs of the two convolutional parts before the global pooling layer. We refer to this model as *1DCNN_E+2DCNN_CM* and *1DCNN_1h+2DCNN_CM*. As a second approach we concatenated the protein-level ELMo embeddings and the DeepFold features $[\mathbf{x}^E, \mathbf{x}^{DF}]$ in a 1,422-dimensional vector and trained the MLP model (*MLP_E+DF*).

Finally, as baseline methods we used the naive [39] and BLAST [3] methods. The naive method assigns a GO term to all test proteins with a probability equal to the frequency of that term in the training set. BLAST annotates each protein with the GO annotations of its top BLAST hit.

3.2.3. TRAINING DETAILS

For the k-NN classifier, we considered Euclidean distance and k values from {1, 2, 3, 5, 7, 11, 15, 21, 25}. For logistic regression, we trained an independent binary classifier for each GO term using L2 regularization. We used stochastic gradient descent to accelerate the optimization. The optimal value for the penalty coefficient λ was tuned jointly for all terms from the values 10^{-3}, 10^{-4} and 10^{-5}.

The neural network models (MLP, 1D-CNN, 2D-CNN, GCN, and the combined 1D-CNN with 2D-CNN) were trained in a mini-batch mode with a mini-batch size of 64. For the 2D-CNN and the combined 1D-CNN with 2D-CNN models, we grouped protein samples of similar size together into mini-batches of sizes [1, 4, 8, 16, 32, 64] due to memory limitations. We trained all models by minimizing the average binary cross entropy over all GO terms. In order to prevent overfitting, we applied dropout [40] with drop probability 0.3 after the global pooling layer. For parameter updating we used the Adam optimizer [41] with an initial learning rate of $5 \cdot 10^{-4}$, which we reduced by a factor of 10 every time the validation loss did not improve for five consecutive epochs. For DeepGOCNN, we used the hyperparameters reported in [37].

For all classifiers, we used the validation *ROC AUC* to select the optimal set of parameters, epoch and number of FC layers wherever applicable.

3.2.4. DATA

We compared models that only use sequence information to models that also include structure, using proteins from the Protein Data Bank [42]. We refer to this dataset as *PDB*. To better assess the sequence-only models, we also applied them to a larger dataset (referred to as *SP*) that includes all available sequences in the SwissProt database in January 2020. Finally, we also evaluated the ELMo-based models on the CAFA3 benchmark [43] (*CAFA* dataset).

For the *PDB* and *SP* datasets we considered proteins with sequence length in the range [40, 1000] that had GO annotations in the Molecular Function Ontology (MFO) with evidence codes 'EXP', 'IDA', 'IPI', 'IMP', 'IGI', 'IEP', 'HTP', 'HDA', 'HMP', 'HGI', 'HEP', 'IBA', 'IBD', 'IKR', 'IRD', 'IC' and 'TAS'. We used CD-HIT [44] to remove redundant sequences with an identity threshold of 95%. After these filtering steps, we had a total of 11,749 protein chains in *PDB* and 80,176 protein sequences in the *SP* dataset.

On the *PDB* dataset we used 5-fold cross-validation. At each fold, we randomly sampled 10% of the training data to use as a validation set. We excluded GO terms that had fewer than 40 positive examples in the training set or fewer than 5 in the validation or test sets and removed proteins that had no annotations after this filtering. To ensure diversity in the evaluation, we only evaluated on proteins from the held-out test set that had at most 30% sequence identity to the training set, as determined by BLAST. The number of protein chains and MFO GO terms resulting from each cross-validation fold can be found in Table 3.S4 (SM2).

For the *SP* dataset, we randomly split the data into a training (80%), a validation (10%) and a test set (10%). We further defined a subset of the test set using BLAST, in which all proteins had sequence identity smaller than 30% to any of the training proteins. We performed the same GO term filtering steps as before. Finally, we had 63,994 training, 8,004 validation and 3,530 test proteins, annotated with C=441 MFO terms.

The *CAFA* training and test sets were provided by the organizers [43]. The test set contains 454 proteins. We randomly split the given training set into 90% for training (28,286 proteins) and 10% for validation (3,143 proteins), annotated with C=679 MFO GO terms. We did not apply sequence similarity filters on the *CAFA* dataset, as in that case we intend to exploit information present in closely related proteins.

Finally, we evaluated the ELMo embeddings on the Biological Process (BPO) and Cellular Component (CCO) ontologies using the *PDB* and *CAFA* datasets. Here, to save computational time, we did not apply cross-validation on the *PDB* data, but a single train/validation/test split. We ensured that no test protein had more than 30% identity to any training protein and filtered rare terms as for the MFO. For BPO, the *PDB* dataset contained 8,406 training, 1,050 validation and 400 test proteins annotated with C=1,108 terms and for CCO, 7,214 training, 902 validation and 319 test proteins annotated with C=228 terms (Table 3.S5, SM2).

3.2.5. PERFORMANCE EVALUATION

The performance was measured using the maximum protein-centric F-measure (F_{max}), the normalized minimum semantic distance (S_{min}) [45, 46] and the term-centric $ROC\,AUC$. For the *PDB* dataset, we provided the mean and standard deviation of the 5 cross-validated folds. When evaluating one train/test split, we estimated 95% confidence intervals (CI's) using bootstrapping: we drew random samples with replacement from the test set until we obtained a set of proteins with a size equal to the original test set and calculated the metric values in this new set. We repeated this procedure 1,000 and 100 times for the *PDB* and *SP* test sets respectively.

3.2.6. CLUSTERING OF SUPERVISED EMBEDDINGS

We extracted supervised embeddings for each protein chain in the *PDB* dataset from the trained MLP, 1D-CNN, GCN and 2D-CNN models using different input features. As embedding vector, we took the output of the hidden layer for the MLP model, and the output of the global pooling layer for the convolutional models. These embedding vectors were of size 512 in all cases, which we compared to the 1,024-dimensional protein-level ELMo embeddings.

The clustering was done using a single train/test split of the *PDB* dataset. For each

test protein we found its 40 nearest training proteins in the embedding space using the cosine distance as a distance measure. Then we computed the Jaccard distance between the neighborhoods found for each test protein using two different embeddings. This gave us a distribution of neighborhood dissimilarities for each pair of embedding types. We used the median of this distribution as a measure of distance between embeddings and applied hierarchical clustering with complete linkage to group similar embeddings together.

We also tested whether differences in the 40 nearest neighbors also lead to differences in the predictions or performance of the different methods. For each model, we calculated the protein-centric F_{max} for every individual protein and used one minus the Pearson correlation of those values as a distance measure to re-cluster the models. In this experiment, for the ELMo embeddings we used the performance of *LR_E*.

3.3. RESULTS

3.3.1. DEEP, PRE-TRAINED EMBEDDINGS OUTPERFORM HAND-CRAFTED SEQUENCE AND STRUCTURE REPRESENTATIONS

We first compared the unsupervised ELMo embeddings of protein sequences and the DeepFold distance map embeddings to hand-crafted sequence and structure representations at the task of predicting MFO terms. We performed 5-fold cross-validation on the *PDB* dataset with at most 30% sequence similarity between test and training proteins. We used the amino acid-level features ($\mathbf{X}^E$, $\mathbf{X}^{1h}$ and $\mathbf{X}^{SA}$) in a 1D-CNN model and two GCN models, and compared to the k-NN, logistic regression (LR) and multilayer perceptron (MLP) classifiers, which use the protein-level features ($\mathbf{x}^E$, $\mathbf{x}^{1h}$, $\mathbf{x}^{kmer}$ and $\mathbf{x}^{DF}$). As seen in Figure 3.2, the models using pre-trained embeddings significantly outperform their counterparts using other features on all evaluation metrics. In addition, DeepFold outperformed hand-crafted secondary structure and backbone angles features ($\mathbf{X}^{SA}$), but was worse than the ELMo embeddings (Figure 3.2). Also, the protein-level $\mathbf{x}^{1h}$ representation (k-mers with k=1) consistently outperformed $\mathbf{x}^{kmer}$ which uses larger k values.

Furthermore, we benchmarked these representations in predicting BPO and CCO terms on the same *PDB* dataset, this time using a single train/test split (while still ensuring at most 30% similarity between test and training proteins). The results (Figure 3.S6, SM2) show that $\mathbf{x}^E$ features outperformed $\mathbf{x}^{DF}$, $\mathbf{x}^{1h}$ and the baselines in both ontologies. The comparison between DeepFold and one-hot did not yield a clearly superior representation, as the results varied per ontology and classifier.

3.3.2. ELMO FEATURES ARE COMPETITIVE IN MFO AND CCO IN CAFA3

To get an additional evaluation of the ELMo embeddings compared to the state-of-the-art, we used them in the *CAFA* dataset (454 test proteins, 679 MFO terms). Table 3.1 shows the performance of *kNN_E*, *LR_E*, *MLP_E* and *1DCNN_E* in this dataset. All had quite competitive performance, outperforming at least 80% of the methods participating in CAFA3 [43], while having 100% coverage, meaning that they could make predictions for all test proteins. Our top model, *MLP_E*, achieved an F_{max} of 0.55, outperforming all but 4 of the methods that had participated in the challenge, as well as *DeepGOCNN_1h*,

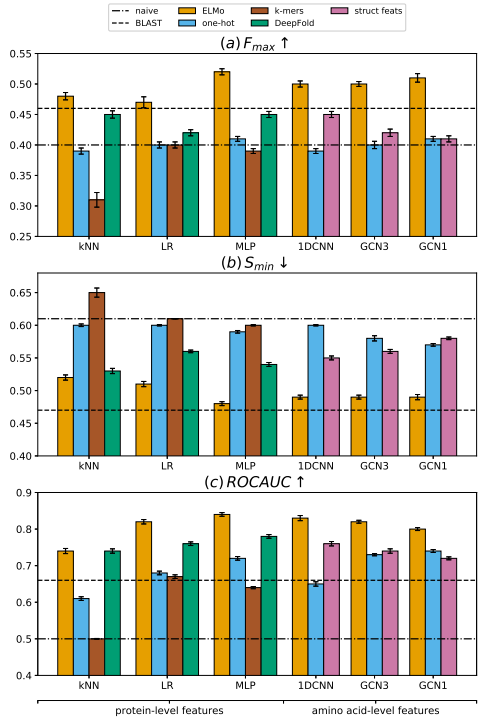


Figure 3.2: F_{max} (a), S_{min} (b) and $ROCAUC$ (c) of models trained using either ELMo embeddings (orange), one-hot encodings (blue), k-mer counts (brown), DeepFold (green) or structural features (pink), averaged over the cross-validated 30% sequence identity PDB test subsets. The arrows denote that lower values (in S_{min}) and higher values (in F_{max} and $ROCAUC$) correspond to better performance. The error bars denote the standard deviation of the cross-validated results. The dashed line corresponds to the performance of BLAST and the dashed dotted line to the naive baseline.

which in our experiments scored an F_{max} of 0.43 (Table 3.1).

To gauge the usefulness of ELMo features in BPO and CCO, we also evaluated *kNN_E* on these ontologies as well. We did not tune the parameter k in this experiment but arbitrarily set it to 5. In BPO, *kNN_E* achieved an F_{max} of 0.34 compared to 0.40 for the top method which would place it in the top-50 out of 146 participants. In CCO, it achieved an F_{max} of 0.60 which was close to the top performance of 0.62 and the 0.61 of *DeepGOCNN_1h* [37].

Taken together, these results show that ELMo features are a promising protein representation for AFP.

3.3.3. SIMPLE MODELS WITH GOOD FEATURES BEAT COMPLEX MODELS WITH ONE-HOT ENCODED AMINO ACIDS

In all convolutional networks that we tested, either one or two FC layers were selected, based on the performance on the validation set. In Table 3.S3, we see that there is no performance gain from the second FC layer for models that use ELMo embeddings, while

Table 3.1: F_{max} of sequence-based methods on the *CAFA* test set with 454 proteins and C=679 MFO GO terms. The performance of BLAST and naive baselines is also given, along with that of the five highest scoring models, which were taken from [43].

Method	F_{max} ↑
Naive*	0.33
BLAST*	0.42
kNN_E	0.50
LR_E	0.51
MLP_E	0.55
1DCNN_E	0.53
DeepGOCNN_1h	0.43
CAFA3 rank 1*	0.62
CAFA3 rank 2*	0.61
CAFA3 rank 3*	0.61
CAFA3 rank 4*	0.61
CAFA3 rank 5*	0.54

most of the other models require this extra layer to improve their performance. This implies that classes are more linearly separable in the embedding spaces learnt by models that use ELMo than those that use hand-crafted features.

In Figure 3.2, we compare models that learn convolutional filters to extract patterns from amino acid-level ELMo embeddings to standard classifiers that use the average of these embeddings along the sequence dimension (protein-level), using the *PDB* dataset. We observed that *LR_E* achieved equal *ROCAUC* to the *GCN3_E_CM* (0.82±0.006 and 0.82±0.004, respectively). The *GCN3_E_CM* achieved a better S_{min} (0.49±0.003) than *LR_E* (0.51±0.004), and a better F_{max} (0.50±0.004, compared to 0.47±0.009). The *kNN_E* had comparable S_{min} to *LR_E* and worse *ROCAUC* than all. The two-layer MLP on the protein-level embeddings (*MLP_E*) achieved the best results on all metrics (F_{max}=0.52±0.005, S_{min}=0.48±0.003, *ROCAUC*=0.84±0.005). This model was closely followed by *1DCNN_E* with *ROCAUC*=0.83±0.007 and *GCN1_E_CM* with F_{max}=0.51±0.007. The three convolution-based models provided the same S_{min} (0.49), which is the second best after *MLP_E*.

More importantly, we found a simple logistic regression model combined with pre-trained features learnt by a deep neural network (*LR_E* and *LR_DF*) considerably outperformed all models that used one-hot encodings of amino acids on all three evaluation metrics (Figure 3.2). These models include our custom 1D-CNN, DeepGOCNN [37] and a GCN that also uses protein structure information [22]. These results demonstrate the usefulness of transfer learning in a task with limited labeled training data such as AFP.

3.3.4. GCN PERFORMS SIMILARLY TO 1D-CNN WHEN USING ELMO EMBEDDINGS

We then tested whether combining the ELMo embeddings with contact map information in a GCN improves the performance, for which we considered the *PDB* dataset. Figure 3.2 shows the mean and standard deviation of the F_{max}, normalized S_{min} and *ROCAUC*, across 5 cross-validated folds. The 3-layer GCN proposed in [22] trained with

the ELMo embeddings (*GCN3_E_CM*) performed similarly to the *1DCNN_E* model based on the three metrics though *1DCNN_E* had marginally better *ROC AUC* (0.83±0.007 compared to 0.82±0.004). We also tested whether a simpler GCN model would be more efficient and found that just a one-layer graph convolutional network (*GCN1_E_CM*) performed comparably to the more complex GCN model (Figure 3.2), having only 2% worse *ROC AUC*. To ensure that our observation about GCNs does not depend on the choice of the graph convolution operator, we repeated the experiments using three other graph operators and obtained similar results (Table 3.S6, SM3).

On the contrary, when using one-hot encoded amino acids as features, both *GCN3_1h_CM* and *GCN1_1h_CM* clearly outperformed *1DCNN_1h*. We also tested the *DeepGOCNN_1h* model [37], which performed 2-5% better than our custom *1DCNN_1h* depending on the metric (F_{max}=0.41±0.004, S_{min}=0.59±0.002 and *ROC AUC*=0.68±0.004). *DeepGOCNN_1h* also had equal F_{max} with *GCN1_1h_CM*, but the latter had 3% better S_{min} and 9% better *ROC AUC*, making it clearly the best model that uses this representation (Figure 3.2).

3.3.5. PROTEIN STRUCTURE DOES NOT ADD INFORMATION TO THE ELMO EMBEDDINGS

In order to explain the lack of significant improvement when including the contact map information, we investigated the behavior of the GCN further, focusing on the 1-layer model, which was at least as good as the 3-layer one. Keeping the architecture the same, we re-trained and tested the model, replacing each contact map with (a) a disconnected graph, i.e. substituting **A** with the identity matrix (*GCN1_E_I*), and (b) a random undirected graph with the same number of edges as the original (*GCN1_E_R*). As shown in Table 3.2, the performance on a single train/test split of the *PDB* dataset (Table 3.S5) remains the same as that of the original contact map for both perturbations of the graphs, hinting that the sequence embeddings are enough for learning a good functional representation. However, replacing ELMo with one-hot encodings in this experiment (*GCN1_1h_I* and *GCN1_1h_R*) led to a performance drop compared to *GCN1_1h_CM* (Table 3.2).

We then trained a GCN model using the node degrees as features (*GCN1_CM*), "forcing" the network to learn to differentiate among the different GO terms using only the contact map. The performance of that network was remarkably worse than *GCN1_1h_CM*, having F_{max}=0.43, S_{min}=0.60 and *ROC AUC*=0.64. To put these numbers into perspective, the simple BLAST baseline had F_{max}=0.37, S_{min}=0.53 and *ROC AUC*=0.62. On the other hand, modeling the contact maps as images and not as graphs and feeding them into a custom 2D-CNN (*2DCNN_CM*) achieved better performance (F_{max}=0.41, S_{min}=0.58 and *ROC AUC*=0.68), although significantly worse than the models that used sequence or pre-trained DeepFold features. Furthermore, the combined *1DCNN_E+2DCNN_CM* (F_{max}=0.46, S_{min}=0.54 and *ROC AUC*=0.74) did not outperform *1DCNN_E*, and *1DCNN_1h +2DCNN_CM* (F_{max}=0.39, S_{min}=0.60 and *ROC AUC*=0.61) was worse than *2DCNN_CM*. On the other hand, *MLP_E+DF* provided slightly better (<2%) cross-validation results than *MLP_E* with F_{max}=0.52±0.006, S_{min}=0.47±0.003 and *ROC AUC*=0.85±0.005. All these results show that integrating ELMo and structural features is not trivial. Although contact maps can in general be used for MFO prediction, in the presence of ELMo embeddings they are not particularly useful.

Table 3.2: F_{max}, S_{min} and $ROCAUC$ of the 1-layer GCN using the identity or a random matrix as adjacency matrices, and ELMo embeddings or one-hot encodings as node features, compared to the naive and BLAST classifiers. All networks were evaluated using one 30% sequence identity test subset of the *PDB* dataset. The 95% confidence intervals were estimated using 1,000 bootstraps.

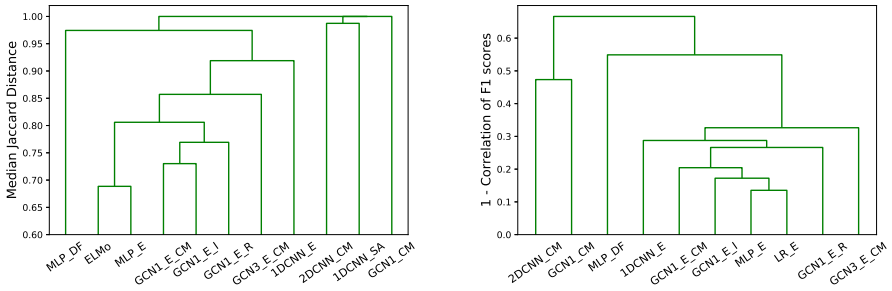
Model	$F_{max} \uparrow$	$S_{min} \downarrow$	$ROCAUC \uparrow$
Naive	0.43 [0.410, 0.451]	0.61 [0.608, 0.620]	0.50 [0.500, 0.500]
BLAST	0.37 [0.346, 0.405]	0.53 [0.512, 0.556]	0.62 [0.597, 0.642]
GCN1_E_CM	0.51 [0.492, 0.540]	0.50 [0.477, 0.515]	0.76 [0.724, 0.789]
GCN1_E_I	0.52 [0.491, 0.541]	0.50 [0.483, 0.519]	0.76 [0.723, 0.793]
GCN1_E_R	0.50 [0.478, 0.527]	0.51 [0.485, 0.524]	0.77 [0.747, 0.798]
GCN1_1h_CM	0.43 [0.407, 0.449]	0.58 [0.567, 0.591]	0.71 [0.672, 0.673]
GCN1_1h_I	0.44 [0.416, 0.457]	0.59 [0.580, 0.600]	0.65 [0.611, 0.683]
GCN1_1h_R	0.43 [0.414, 0.454]	0.59 [0.576, 0.595]	0.70 [0.655, 0.727]

3.3.6. LANGUAGE MODELING LEARNS A COARSE FUNCTIONAL REPRESENTATION

We also evaluated the sequence-only models in the larger *SP* dataset (3,530 test proteins, 441 MFO terms) (Table 3.S7, SM4). The absolute performances are better, but the superiority of ELMo embeddings is evident, as even simple models such as *kNN_E* and *LR_E* outperform all more complex models that use one-hot encodings. *MLP_E* was the top method based on all three metrics in this dataset too (Table 3.S7). Analyzing the performance per GO term, we found that although *kNN_E* has a larger mean $ROCAUC$ than *1DCNN_1h*, its superiority is mainly shown on the most frequent terms (Figs. S7a-b, SM5). On the contrary, all other tested models that use ELMo embeddings tend to have better performance for more specific terms (Figure 3.S8, SM5) and they consistently outperform the one-hot encodings-based models across all levels of the GO graph (Figs. S7c-h, SM5). This shows that the more general functions can be learned during unsupervised pre-training, but further supervised learning is needed for the more specific ones.

3.3.7. SUPERVISED PROTEIN EMBEDDINGS GIVE INSIGHTS INTO THE BEHAVIOR OF THE MODELS

To better understand the differences between the models, we compared the embeddings learned by each of them. We fed all trained models with every protein from our *PDB* dataset and saved the 512-dimensional embedding vectors, which gave us an $11,740 \times 512$ embedding matrix. We then calculated the rank of each of these matrices to assess how "rich" the learned representations are. As shown in Table 3.S8 (SM6), all methods that use the ELMo representation are either full-rank or very close to full-rank (508-512). On the other hand, the models that only operated on contact maps learned much simpler, lower-dimensional representations, with rank 310 for *2DCNN_CM* and 105 for *GCN1_CM*. By applying principal components analysis (PCA) to the *GCN1_CM* embeddings, we found that 3 components explained 99.8% of the total variance (Figure 3.S9,



(a) (b)

Figure 3.3: Hierarchical clustering of the models based on the similarity of the 40 nearest neighbors of each protein in the embedding space (a) and the correlation in protein-centric F_{max} (b).

SM6), suggesting that essentially this network learned a 3-feature representation of the proteins.

We also compared the embeddings of the different supervised models to the unsupervised ELMo embeddings. For every pair of test-training proteins from our *PDB* dataset, we calculated their cosine similarity in the embedding space, as well as a measure of similarity of their GO annotations based on the Jaccard index [47]. For the ELMo embeddings, we found that the two similarity measures were significantly correlated (Figure 3.S10, with Spearman ρ=0.07, permutation p-value < 10^{-4}, SM7). By extracting the embeddings from a supervised model such as *1DCNN_E* and *MLP_E*, the correlation value doubled (ρ=0.14, p-value < 10^{-4}, SM7). For the *GCN1_E_CM*, the correlation value was 0.11 (Figure 3.S11, SM7). This verifies that unsupervised pre-training is able to capture some information about protein function, while additional supervised training provides extra information to the model.

To test to what extent different models learn similar embeddings, we clustered them based on the overlap of their 40 nearest neighborhood graphs, measured using Jaccard distance (Figure 3.3a). We observed that the embeddings of *MLP_E* are the most similar to ELMo (Jaccard distance of 0.68, meaning that about one third of the 40 nearest neighbors are common). The models that used a 1-layer GCN (*GCN1_E_CM*, *GCN1_E_I* and *GCN1_E_R*) learned relatively similar neighborhoods to each other, clustering together at distance 0.77. Moreover, all ELMo-based methods cluster together with *1DCNN_E*, which has the most different representation out of them. In contrast, the models that do not use ELMo features learned very different embeddings, as their neighborhoods have nearly zero overlap both to each other and to the ELMo-based models.

Finally, we investigated whether the observed differences in embeddings imply different performances across proteins. The clustering based on protein-centric performance (Figure 3.3b) was very similar to the one obtained when using embedding similarities (Figure 3.3a). The rank correlation between both similarities was 0.92. However, in absolute numbers, the performance similarities are much higher than the neighborhood similarities (at least 0.35 overall and at least 0.67 among methods that use ELMo embeddings). This shows that ELMo-based models tend to behave similarly.

3.4. DISCUSSION

Our work continues upon two recent studies involving protein representation learning [20] and its combination with protein structure applied to AFP [22]. We confirm the power of the unsupervised ELMo embeddings in capturing relevant biological information about proteins [20]. Simply embedding the proteins into the learned 1,024-dimensional space and applying the k-NN classifier led to better molecular function prediction performance than the two baseline methods (BLAST and naive), as well as several commonly used hand-crafted features such as one-hot encoding of amino acids, k-mer counts, secondary structure and backbone angles. This implies that the ELMo model was able to learn an embedding space in which the similarity between two proteins reflects functional similarity reasonably well, although it was only exposed to amino acid sequences and not to GO annotations. We had similar results with DeepFold embeddings [34] which model protein structures and a similar observation has been recently made for protein domain embeddings [48]. However, the ELMo representation only coarsely reflects protein function, as demonstrated by the poorer performance of the k-NN classifier on the most specific terms.

As expected, we were able to improve the prediction accuracy achieved by the unsupervised embeddings by training supervised AFP methods on the embedding space. A set of logistic regression classifiers trained individually for each GO term achieved comparable S_{min} and F_{max} to the k-NN, while achieving significantly higher $ROCAUC$ in the *PDB* dataset. Contrary to expectation, the GCN and 1D-CNN models trained on the amino acid-level embeddings extracted by ELMo were barely able to outperform the logistic regression model in terms of $ROCAUC$. They did outperform it in terms of S_{min} and F_{max}, though. However, in the *SP* dataset, which is larger and contains more specific GO terms, the differences in S_{min} are less profound (Table 3.S7). Moreover, replacing the linear model (LR) with a non-linear one (MLP) gave a significant performance boost, considerably outperforming all others in $ROCAUC$ and achieving competitive *CAFA* performance. Supervised training also resulted in a more consistent performance across all levels of GO term specificity. In contrast, for DeepFold embeddings, training supervised methods did not improve upon the k-NN performance. This is probably due to the fact that DeepFold is a metric learning model tuned to recognize similar protein structures and not to generally model protein characteristics. All in all, the competitive performance of the protein-level models highlights the power of the unsupervised protein embeddings.

In [22], the authors report on the superiority of a 3-layer GCN using amino acid embeddings from a pre-trained language model based on a LSTM network over BLAST and a 1D-CNN using a one-hot encoded amino acid representation. They attribute this superiority to the use of graph convolutions to model the protein 3D structure represented by contact maps. However, our experiments show that a 1D-CNN with strong amino acid embeddings is competitive with the GCN. Both convolutional models exhibited severe performance decline when replacing the ELMo embeddings with one-hot encoded amino acids. Based on these, we cannot exclude the possibility that the language model of [22] is by itself powerful enough to explain (most of) the increase in performance. If that is indeed the case, it would account for the fact that replacing the true contact map with a predicted one does not cause a significant drop in performance [22]. To support

this claim, we trained another GCN model from scratch, keeping the same architecture as our best GCN, but replacing the contact map by a graph with all nodes disconnected. The performance of that network was similar to that of the original (using the contact map). The same pattern was observed when replacing the contact map with a random graph (both at training and test time), clearly demonstrating that the contribution of the contact maps is rather small. This observation is interesting, as protein 3D structure is much more difficult and expensive to obtain than the sequence.

One of the hyperparameters of our networks was the number of fully-connected (FC) layers between the global pooling layer and the output FC layer for the classification. In our experiments, we tested our models with zero and one intermediate FC layer and used the validation *ROCAUC* to select the optimal for each model. In cases where the performance difference was less than 0.01, we chose to keep the simpler model for testing, as having fewer parameters makes it less prone to overfitting and more likely to better generalize on unseen proteins. A clear pattern emerged from this selection: for both GCN and 1D-CNN networks trained with ELMo embeddings, the extra FC layer was not required. On the other hand, for networks trained with one-hot encoded amino acid features or without any sequence features, the more complicated architecture was always selected. This means that in the feature space learned by the convolutional layers, the different classes (GO terms) are "more linearly separable" when ELMo embeddings are used and learning a simple mapping from that space to the output classes is enough for good performance. In the absence of "good" input features, it is harder for the convolutions to learn a "good" embedding space and as a result a more complex classifier is needed.

One can reasonably assume that also in the case of the one-hot features, it would be possible to learn a better (supervised) embedding space that only requires one linear classification layer. However, that would take a deeper architecture with more convolutional layers to enable us to discover more complicated patterns in protein sequences. This is problematic because the amount of available labeled data is not enough to train deep models with a larger number of parameters. Our experiments showed that two recent models, a 3-layer GCN [22] and a wide 1D-CNN [37], both operating on one-hot encoded amino acids, were remarkably inferior to linear and nearest neighbor methods that operate on pre-trained features. Building a more complex model increases not only training time but also the man-hours spent deciding on the correct architecture and tuning the larger number of hyperparameters. To make matters worse, one would have to repeat almost the whole process from scratch if the task changes e.g. from function prediction to structure prediction. Unsupervised pre-training relieves part of that burden by creating only one complicated deep sequence model to learn a meaningful feature representation of amino acids or proteins, which can then be fed to simpler classifiers to obtain competitive performance in several tasks without much effort [20], as we demonstrated here.

Note that DeepGOCNN combined with other data sources performed better on the CAFA3 MFO benchmark than our *MLP_E* model [37]. Here, we focused on combining ELMo with protein structure information, but other, more diverse data types such as co-expression and protein interactions should be tested in conjuction with these advanced sequence features in ensemble methods. We expect this to be highly beneficial in the

BPO, because ELMo did not achieve high CAFA3 performance in that ontology and the CAFA-π results hinted that achieving good BPO performance using sequence alone is difficult [43].

Our experiments suggest that combining structure information in the form of a contact map with sequence information is not straightforward, when high-quality sequence features are available. Joining a one-dimensional and a two-dimensional CNN that independently extract sequence and contact map features, respectively, did not improve performance over the 1D-CNN applied to sequence data only. It is unlikely that contact maps do not contain any functional information, so our observations could have two possible explanations: either the ELMo embeddings contain three-dimensional structure information or we are still unable to leverage the full potential of contact maps.

To test the first hypothesis one could train a classifier that takes the amino acid-level features as inputs and predicts contacts between amino acid pairs. Such models already exist and do quite well in the CASP challenges by using physicochemical properties, the position-specific scoring matrix (PSSM) and predictions about secondary structure, solvent accessibility and backbone angles [49–51]. By replacing these features with sequence embeddings as in [52], we would expect a considerable improvement in the performance of these models.

On the contrary, finding a more effective way of using distance or contact maps is not trivial. Here, we considered a contact threshold of 10 Å by following previous studies [22], which is a more relaxed threshold than the one used in CASP challenges (8 Å), but also used alternative threshold strategies and obtained similar results. One could argue that the distance matrix is more informative and should be preferred, but our experiments did not confirm that. A different way of using distance maps in a GCN has been proposed by [31] to predict protein interfaces. Firstly, instead of using a fixed distance threshold, Fout et al. define each amino acid as being "in contact" with its k nearest residues, which creates a directed graph as the property of being someone's nearest neighbor is not commutative. Moreover, the distances between the k nearest residues were smoothed with a Gaussian kernel and used as edge features over which a different set of filters was learned [31]. Further research is required to resolve this issue, but our DeepFold results show that unsupervised pre-training is a promising path in this case too.

In conclusion, this study shows that deep unsupervised pre-training of protein sequences is beneficial for predicting molecular function, as it can capture useful aspects of the amino acid sequences. We also showed that combining these sequential embeddings with contact map information does not yield significant performance improvements in the task, hinting that the embeddings may already contain 3D structural information. As language modeling of proteins is a new field with great potential, we think that future work should perform systematic comparisons of those models in AFP, but also other protein-related tasks.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Elvin Isufi and Chirag Raman for their valuable comments and feedback.

FUNDING

This work has been supported by Keygene N.V., a crop innovation company in the Netherlands and by the Spanish MINECO/FEDER Project TEC2016-80141-P with the associated FPI grant BES-2017-079792.

REFERENCES

[1] A. Villegas-Morcillo, S. Makrodimitris, R. C. H. J. van Ham, A. M. Gomez, V. Sanchez, and M. J. T. Reinders, *Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function,* Bioinformatics (2020), 10.1093/bioinformatics/btaa701, btaa701, https://academic.oup.com/bioinformatics/advance-article-pdf/doi/10.1093/bioinformatics/btaa701/33653342/btaa701.pdf .

[2] M. Kimura and T. Ohta, *On some principles governing molecular evolution,* Proc. Natl. Acad. Sci. U.S.A. **71**, 2848 (1974).

[3] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, *Basic local alignment search tool,* J. Mol. Biol. **215**, 403 (1990).

[4] S. R. Eddy, *A new generation of homology search tools based on probabilistic inference.* Genome informatics. International Conference on Genome Informatics (2009), 10.1142/9781848165632_0019.

[5] D. Cozzetto, F. Minneci, H. Currant, and D. T. Jones, *FFPred 3: feature-based function prediction for all Gene Ontology domains,* Sci Rep **6**, 31865 (2016).

[6] R. Fa, D. Cozzetto, C. Wan, and D. T. Jones, *Predicting human protein function with multitask deep neural networks,* PLoS ONE (2018), 10.1371/journal.pone.0198216.

[7] A. Sureyya Rifaioglu, T. Doğan, M. Jesus Martin, R. Cetin-Atalay, and V. Atalay, *DEEPred: Automated Protein Function Prediction with Multi-task Feed-forward Deep Neural Networks,* Scientific Reports (2019), 10.1038/s41598-019-43708-3.

[8] R. Bonetta and G. Valentino, *Machine learning techniques for protein function prediction,* Proteins: Structure, Function, and Bioinformatics (2019), 10.1002/prot.25832.

[9] M. Kulmanov, M. A. Khan, and R. Hoehndorf, *DeepGO: Predicting protein functions from sequence and interactions using a deep ontology-aware classifier,* Bioinformatics (2018), 10.1093/bioinformatics/btx624, arXiv:1705.05919 .

[10] X. Liu, *Deep recurrent neural network for protein function prediction from sequence,* arXiv preprint arXiv:1701.08318 (2017).

[11] R. Cao, C. Freitas, L. Chan, M. Sun, H. Jiang, and Z. Chen, *ProLanGO: Protein function prediction using neural machine translation based on a recurrent neural network,* Molecules (2017), 10.3390/molecules22101732, arXiv:1710.07016 .

[12] M. Ashburner *et al.*, *Gene ontology: Tool for the unification of biology,* (2000).

[13] C. Doersch, A. Gupta, and A. A. Efros, *Unsupervised Visual Representation Learning by Context Prediction,* 2015 IEEE International Conference on Computer Vision (ICCV) , 1422 (2015).

[14] S. Gidaris, P. Singh, and N. Komodakis, *Unsupervised Representation Learning by Predicting Image Rotations,* ArXiv **abs/1803.0** (2018).

[15] A. Mathis, M. Yüksekgönül, B. Rogers, M. Bethge, and M. W. Mathis, *Pretraining boosts out-of-domain robustness for pose estimation,* ArXiv **abs/1909.1** (2019).

[16] B. McCann, J. Bradbury, C. Xiong, and R. Socher, *Learned in translation: Contextualized word vectors,* in *Advances in Neural Information Processing Systems* (2017) arXiv:1708.00107 .

[17] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, *Deep Contextualized Word Representations,* (2018) arXiv:1802.05365 .

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding,* arXiv preprint arXiv:1810.04805 (2018).

[19] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, P. Chen, J. Canny, P. Abbeel, and Y. Song, *Evaluating protein transfer learning with tape,* in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019) pp. 9689–9701.

[20] M. Heinzinger, A. Elnaggar, Y. Wang, C. Dallago, D. Nechaev, F. Matthes, and B. Rost, *Modeling aspects of the language of life through transfer-learning protein sequences,* BMC Bioinformatics **20**, 723 (2019).

[21] H. Kane, M. Coulibali, A. Abdalla, and P. Ajanoh, *Augmenting Protein Network Embeddings with Sequence Information,* BioRxiv , 730481 (2019).

[22] V. Gligorijevic, P. D. Renfrew, T. Kosciolek, J. K. Leman, K. Cho, T. Vatanen, D. Berenberg, B. Taylor, I. M. Fisk, R. J. Xavier, R. Knight, and R. Bonneau, *Structure-Based Function Prediction using Graph Convolutional Networks,* bioRxiv (2019), 10.1101/786236.

[23] E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, and G. M. Church, *Unified rational protein engineering with sequence-based deep representation learning,* Nat. Methods **16**, 1315 (2019).

[24] A. Rives, S. Goyal, J. Meier, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus, *Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences,* bioRxiv (2019), 10.1101/622803.

[25] C. A. Wilson, J. Kreychman, and M. Gerstein, *Assessing annotation transfer for genomics: Quantifying the relations between protein sequence, structure and function through traditional and probabilistic scores,* Journal of Molecular Biology (2000), 10.1006/jmbi.2000.3550.

[26] N. Weinhold, O. Sander, F. S. Domingues, T. Lengauer, and I. Sommer, *Local function conservation in sequence and structure space,* PLoS Computational Biology (2008), 10.1371/journal.pcbi.1000105.

[27] L. Bartoli, E. Capriotti, P. Fariselli, P. L. Martelli, and R. Casadio, *The pros and cons of predicting protein contact maps,* Methods in Molecular Biology (2007), 10.1385/1-59745-574-1:199.

[28] J. M. Duarte, R. Sathyapriya, H. Stehr, I. Filippis, and M. Lappe, *Optimal contact definition for reconstruction of Contact Maps,* BMC Bioinformatics (2010), 10.1186/1471-2105-11-283.

[29] J. Zhu, H. Zhang, S. C. Li, C. Wang, L. Kong, S. Sun, W. M. Zheng, and D. Bu, *Improving protein fold recognition by extracting fold-specific features from predicted residue–residue contacts,* Bioinformatics (2017), 10.1093/bioinformatics/btx514.

[30] W. Zheng, Q. Wuyun, Y. Li, S. M. Mortuza, C. Zhang, R. Pearce, J. Ruan, and Y. Zhang, *Detecting distant-homology protein structures by aligning deep neural-network based contact maps,* PLoS computational biology (2019), 10.1371/journal.pcbi.1007411.

[31] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, *Protein Interface Prediction using Graph Convolutional Networks,* in *Advances in Neural Information Processing Systems 30* (Curran Associates, Inc., 2017) pp. 6530–6539.

[32] R. Zamora-Resendiz and S. Crivelli, *Structural Learning of Proteins Using Graph Convolutional Neural Networks,* bioRxiv , 610444 (2019).

[33] C. B. Anfinsen, *Principles that govern the folding of protein chains,* Science **181**, 223 (1973), https://science.sciencemag.org/content/181/4096/223.full.pdf .

[34] Y. Liu, Q. Ye, L. Wang, and J. Peng, *Learning structural motif representations for efficient protein structure search,* Bioinformatics **34**, i773 (2018).

[35] W. Kabsch and C. Sander, *Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features,* Biopolymers: Original Research on Biomolecules **22**, 2577 (1983).

[36] J. Lyons, A. Dehzangi, R. Heffernan, A. Sharma, K. Paliwal, A. Sattar, Y. Zhou, and Y. Yang, *Predicting backbone Ca angles and dihedrals from protein sequences by stacked sparse auto-encoder deep neural network,* Journal of computational chemistry **35**, 2040 (2014).

[37] M. Kulmanov and R. Hoehndorf, *DeepGOPlus: improved protein function prediction from sequence,* Bioinformatics **36**, 422 (2020).

[38] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks,* in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* (2019) arXiv:1609.02907 .

[39] P. Radivojac *et al.*, *A large-scale evaluation of computational protein function prediction*, Nature Methods (2013), 10.1038/nmeth.2340.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research (2014).

[41] D. P. Kingma and J. L. Ba, *Adam: A method for stochastic optimization*, in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (2015) arXiv:1412.6980 .

[42] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, and I. N. Shindyalov, *The Protein Data Bank (www.rcsb.org)*, Nucleic Acids Research (2000), 10.1093/nar/28.1.235.

[43] N. Zhou *et al.*, *The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens*, Genome Biology **20**, 244 (2019).

[44] L. Fu, B. Niu, Z. Zhu, S. Wu, and W. Li, *CD-HIT: Accelerated for clustering the next-generation sequencing data*, Bioinformatics (2012), 10.1093/bioinformatics/bts565.

[45] W. T. Clark and P. Radivojac, *Information-theoretic evaluation of predicted ontological annotations*, Bioinformatics **29**, 53 (2013).

[46] Y. Jiang *et al.*, *An expanded evaluation of protein function prediction methods shows an improvement in accuracy*, Genome biology **17**, 184 (2016), arXiv:1601.00891 .

[47] C. Pesquita, D. Faria, H. P. Bastos, A. O. Falcão, and F. M. Couto, *Evaluating GO-based Semantic Similarity Measures*, (2007).

[48] D. P. Melidis, B. Malone, and W. Nejdl, *dom2vec: Assessable domain embeddings and their use for protein prediction tasks*, bioRxiv (2020), 10.1101/2020.03.17.995498, https://www.biorxiv.org/content/early/2020/03/18/2020.03.17.995498.full.pdf .

[49] J. Cheng and P. Baldi, *Improved residue contact prediction using support vector machines and a large feature set*, BMC Bioinformatics (2007), 10.1186/1471-2105-8-113.

[50] D. T. Jones, T. Singh, T. Kosciolek, and S. Tetchner, *MetaPSICOV: Combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins*, Bioinformatics (2015), 10.1093/bioinformatics/btu791.

[51] S. Wang, S. Sun, Z. Li, R. Zhang, and J. Xu, *Accurate De Novo Prediction of Protein Contact Map by Ultra-Deep Learning Model*, PLoS Computational Biology (2017), 10.1371/journal.pcbi.1005324, arXiv:1609.00680 .

[52] T. Bepler and B. Berger, *Learning protein sequence embeddings using information from structure*, in *7th International Conference on Learning Representations, ICLR 2019* (2019) arXiv:1902.08661 .

[53] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, *A Comprehensive Survey on Graph Neural Networks,* (2019).

[54] M. Defferrard, X. Bresson, and P. Vandergheynst, *Convolutional neural networks on graphs with fast localized spectral filtering,* in *Advances in Neural Information Processing Systems* (2016) arXiv:1606.09375 .

[55] K. Xu, S. Jegelka, W. Hu, and J. Leskovec, *How powerful are graph neural networks?* in *7th International Conference on Learning Representations, ICLR 2019* (2019) arXiv:1810.00826 .

3.5. SUPPLEMENTARY MATERIAL

3.5.1. NEURAL NETWORK ARCHITECTURES AND HYPERPARAMETERS

Table 3.S1: Neural network hyperparameters for the MLP, 1D-CNN and 2D-CNN models. For each model, input feature information, architecture layers and names are provided. Parentheses indicate an optional intermediate FC layer followed by ReLU. The variable *C* refers to the number of MFO GO terms to be classified by the network (which depends on the dataset).

Model	MLP	1D-CNN	2D-CNN
Input	Protein-level features	Amino acid-level features	Contact map
Layers	FC 512 - ReLU FC *C* - Sigmoid	1Dconv 5x64 - ReLU 1Dconv 5x512 - ReLU Global max pool 512 (FC 256 - ReLU) FC *C* - Sigmoid	2Dconv 5x5x64 - ReLU 2Dconv 5x5x512 - ReLU Global max pool 512 (FC 256 - ReLU) FC *C* - Sigmoid
Names	*MLP_{E,1h,kmer,DF}*	*1DCNN_{E,1h,SA}*	*2DCNN_CM*

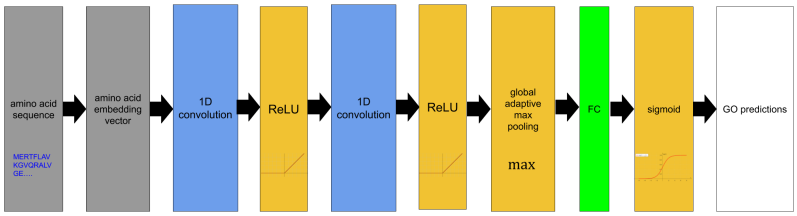


Figure 3.S1: Neural network architecture for the 1D-CNN model.

Table 3.S2: Neural network hyperparameters for the 3-layer GCN, 1-layer GCN and the combined 1D-CNN with 2D-CNN models. For each model, input feature information, architecture layers and names are provided. Parentheses indicate an optional intermediate FC layer followed by ReLU. The symbol + indicates the combination of models. The variable *C* refers to the number of MFO GO terms to be classified by the network (which depends on the dataset).

Model	3-layer GCN	1-layer GCN	1D-CNN + 2D-CNN
Input	Amino acid-level features and contact map		
Layers	Gconv 256 - ReLU Gconv 256 - ReLU Gconv 512 - ReLU Global sum pool 512 (FC 256 - ReLU) FC *C* - Sigmoid	Gconv 512 - ReLU Global sum pool 512 (FC 256 - ReLU) FC *C* - Sigmoid	1Dconv 5x64 - ReLU + 2Dconv 5x5x64 - ReLU 1Dconv 5x512 - ReLU + 2Dconv 5x5x512 - ReLU Global max pool 512 + 512 (FC 256 - ReLU) FC *C* - Sigmoid
Names	*GCN3_{E,1h,SA}_CM*	*GCN1_{E,1h,SA}_CM*	*1DCNN_1h+2DCNN_CM* *1DCNN_E+2DCNN_CM*

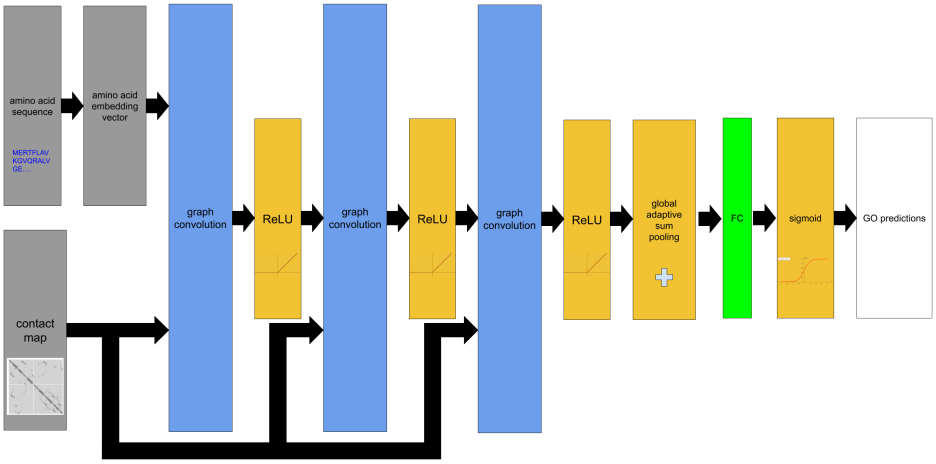


Figure 3.S2: Neural network architecture for the 3-layer GCN model.

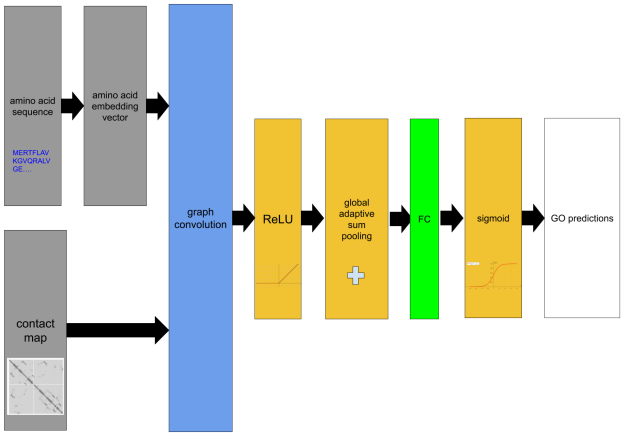


Figure 3.S3: Neural network architecture for the 1-layer GCN model.

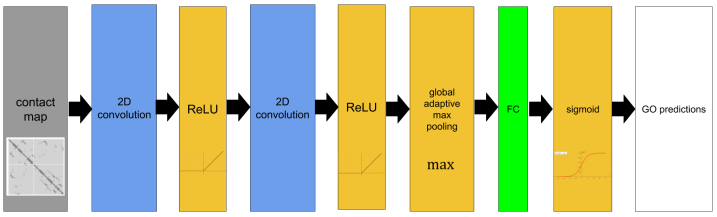


Figure 3.S4: Neural network architecture for the 2D-CNN model.

Table 3.S3: *ROC AUC* of convolutional network models (CNN and GCN) trained with 1 and 2 FC layers on the *PDB* validation set. The results of the selected option for each model are shown in bold.

Method	1 FC layer	2 FC layers
1DCNN_E	**0.929**	0.924
GCN3_E_CM	**0.927**	0.932
GCN1_E_CM	**0.924**	0.930
1DCNN_E+2DCNN_CM	**0.909**	0.893
1DCNN_1h	**0.844**	0.838
GCN3_1h_CM	0.826	**0.873**
GCN1_1h_CM	0.799	**0.857**
1DCNN_1h+2DCNN_CM	**0.835**	0.834
GCN1_CM	0.622	**0.696**
2DCNN_CM	0.795	**0.847**

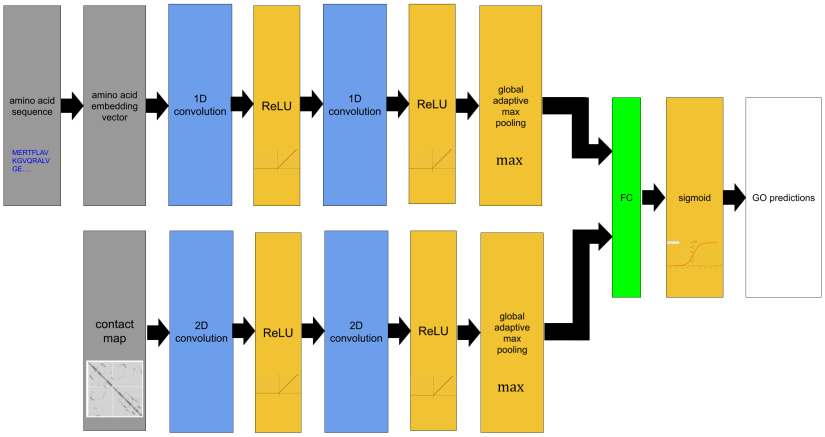


Figure 3.S5: Neural network architecture for the combined 1D-CNN with 2D-CNN model.

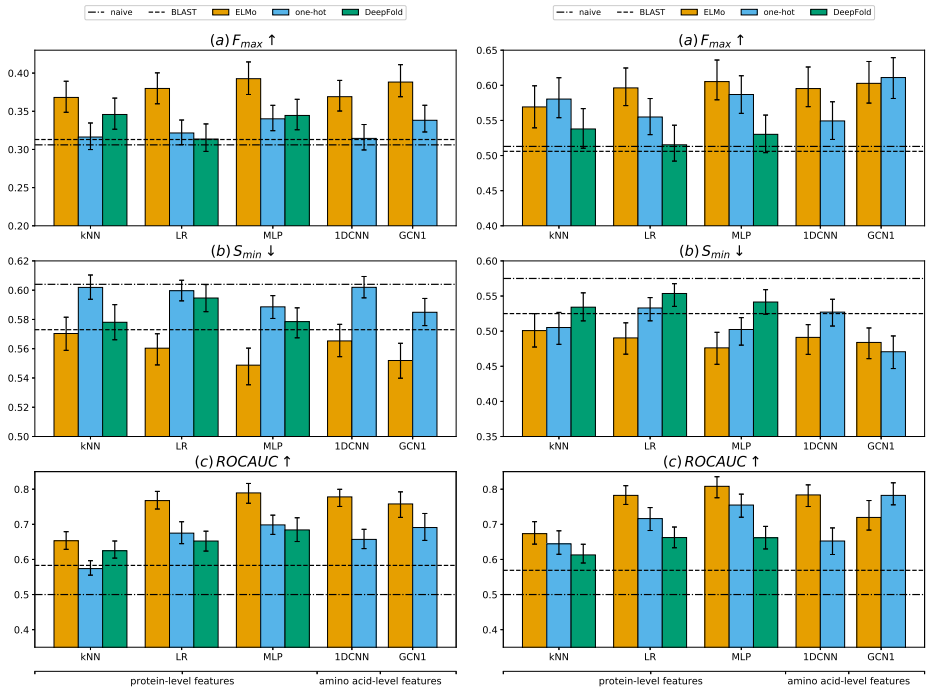
3.5.2. *PDB* CROSS-VALIDATION FOLDS AND BPO/CCO EVALUATION

Table 3.S4: Number of training, validation and test (30%) protein chains, and number of MFO GO terms for each cross-validation fold in the *PDB* dataset.

CV Fold	Training	Validation	Test (30%)	MFO terms
1	8,452	939	480	265
2	8,451	940	516	270
3	8,455	938	495	269
4	8,454	939	519	277
5	8,452	940	584	262

Table 3.S5: Number of training, validation and test (30%) protein chains, and number of terms in the *PDB* dataset for each GO ontology.

Ontology	Training	Validation	Test (<30%)	# terms
MFO	9,395	1,173	450	256
BPO	8,406	1,050	400	1,108
CCO	7,214	902	319	228



(1) BPO (2) CCO

Figure 3.S6: F_{max} (a), S_{min} (b) and $ROCAUC$ (c) of models trained using either ELMo embeddings (orange), one-hot encodings (blue), or DeepFold (green), on the 30% sequence identity PDB test subsets using (1) BPO and (2) CCO terms. The arrows denote that lower values (in S_{min}) and higher values (in F_{max} and $ROCAUC$) correspond to better performance. The error bars denote 95% confidence intervals estimated using 1,000 bootstraps. The dashed line corresponds to the performance of BLAST and the dashed dotted line to the naive baseline.

3.5.3. ALTERNATIVE GRAPH CONVOLUTION OPERATORS

DEFINITIONS OF OPERATORS

The graph convolution operator proposed in [38] is a polynomial approximation of the spectral graph convolution operation [53]. It is a first-order approximation, meaning that it makes use of Chebyshev polynomials up to degree one. That means that when calculating the new feature vector for the node using this approach, we only consider the current features of that node plus the features of its immediate neighbors (nodes that are 1 "hop" away). Higher-order approximations using Chebyshev polynomials were proposed in [54], leading to the Chebyshev spectral graph convolution operator defined as:

$$\mathbf{X}' = \sum_{k=0}^{K} \mathbf{Z}^{(k)}(\mathbf{X}, \hat{\mathbf{L}}) \cdot \mathbf{W}^{(k)} \tag{3.2}$$

In equation 3.2, K is the order of the approximation and corresponds to the number of "hops" from the central node considered and $W^{(k)}$ is the weight matrix corresponding used to combine the features of nodes that are k hops away from the central one.

Finally, $\mathbf{Z}^{(k)}(\mathbf{X},\hat{\mathbf{L}})$ is a Chebyshev polynomial of degree k with $\mathbf{Z}^{(0)}=\mathbf{X}$, $\mathbf{Z}^{(1)}=\hat{\mathbf{L}}\cdot\mathbf{X}$ and $\mathbf{Z}^{(k)}=2\hat{\mathbf{L}}\mathbf{Z}^{(k-1)}-\mathbf{Z}^{(k-2)}$ and $\hat{\mathbf{L}}=\frac{2}{\lambda_{max}}\mathbf{L}-\mathbf{I}$ is the normalized graph Laplacian.

The graph isomorphism network (GIN) is a form of spatial graph convolution [55]. If we use $(\mathbf{X})_i$ to denote the feature vector of node i, i.e. the i-th row of $\mathbf{X}$, the GIN uses the following operator:

$$(\mathbf{X}')_i = h\left((\mathbf{X})_i + \sum_{j\in N(i)} (\mathbf{X})_j, \mathbf{W}\right) \tag{3.3}$$

In equation 3.3, $h(\mathbf{x},\mathbf{W})$ is a multi-layer perceptron with weights $\mathbf{W}$ and $N(i)$ is the set of nodes adjacent to node i.

The Gaussian mixture models (GMM) convolution operation uses a (hand-crafted) d-dimensional edge feature vector $\mathbf{u}(i,j)\in\mathbb{R}^d$ and defines a J-dimensional node similarity function $\mathbf{w}(\mathbf{u})=(w_1(\mathbf{u}),w_2(\mathbf{u}),...,w_J(\mathbf{u}))$ whose m-th element is calculated using equation 3.4 from the learnable parameters $\boldsymbol{\mu}_m$ and Σ_m.

$$w_m(\mathbf{u}) = \exp\left(-\frac{1}{2}(\mathbf{u}-\boldsymbol{\mu}_m)^T\Sigma_m^{-1}(\mathbf{u}-\boldsymbol{\mu}_m)\right) \tag{3.4}$$

EXPERIMENTS AND RESULTS

We tested the Chebyshev operator with $K=2,3,..,10$ using one convolutional layer with 512 filters. In the validation set, we observed similar performance regardless of the value of k, so we only tested the models with $K=2$ and $K=10$. For the GIN model, we used a two-layer perceptron with 512 hidden units. For the GMM, we defined $u(i,j)$ as a single number denoting the linear distance in the sequence between residues i and j, with $u(i,j)>0$ if i precedes j. The dimensionality of w was also set to 1. To achieve the full potential of the GMM model, one needs to learn separate values for the mean and variance for every filter. However, due to memory limitations we were not able to train such a model and restricted ourselves to the simpler and less powerful version of the algorithm where $\boldsymbol{\mu}_m$ and Σ_m are shared across all filters. The performance of the mentioned graph convolution operators is shown in Table 3.S6.

Table 3.S6: Results of different graph convolution operators on the *PDB* test subset with at most 30% sequence identity, using ELMo embeddings or one-hot encoded amino acids. We considered the model with an intermediate FC layer when using one-hot encodings, while the simpler model was chosen when using ELMo embeddings. The performance of the different operators is very similar for the two representations.

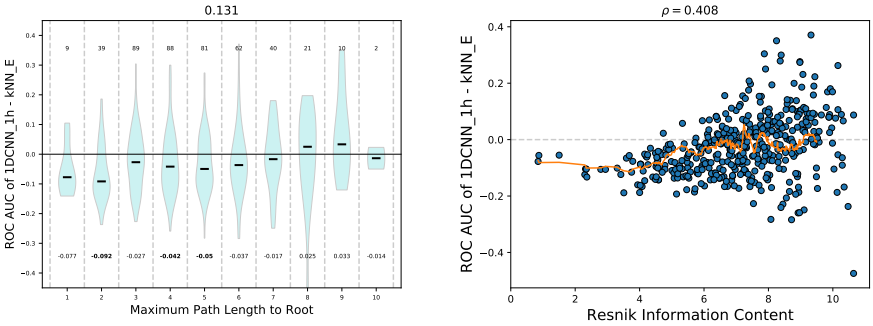
Graph conv operator	ELMo embeddings		One-hot encodings	
	$S_{min}\downarrow$	$ROCAUC\uparrow$	$S_{min}\downarrow$	$ROCAUC\uparrow$
Kipf GCN (1-layer)	0.50	0.76	0.58	0.75
Chebyshev ($K=2$)	0.51	0.75	0.57	0.75
Chebyshev ($K=10$)	0.51	0.74	0.57	0.74
GIN	0.52	0.75	0.59	0.70
GMM	0.51	0.75	0.58	0.73

3.5.4. PERFORMANCE OF SEQUENCE-BASED MODELS ON THE *SP* DATASET

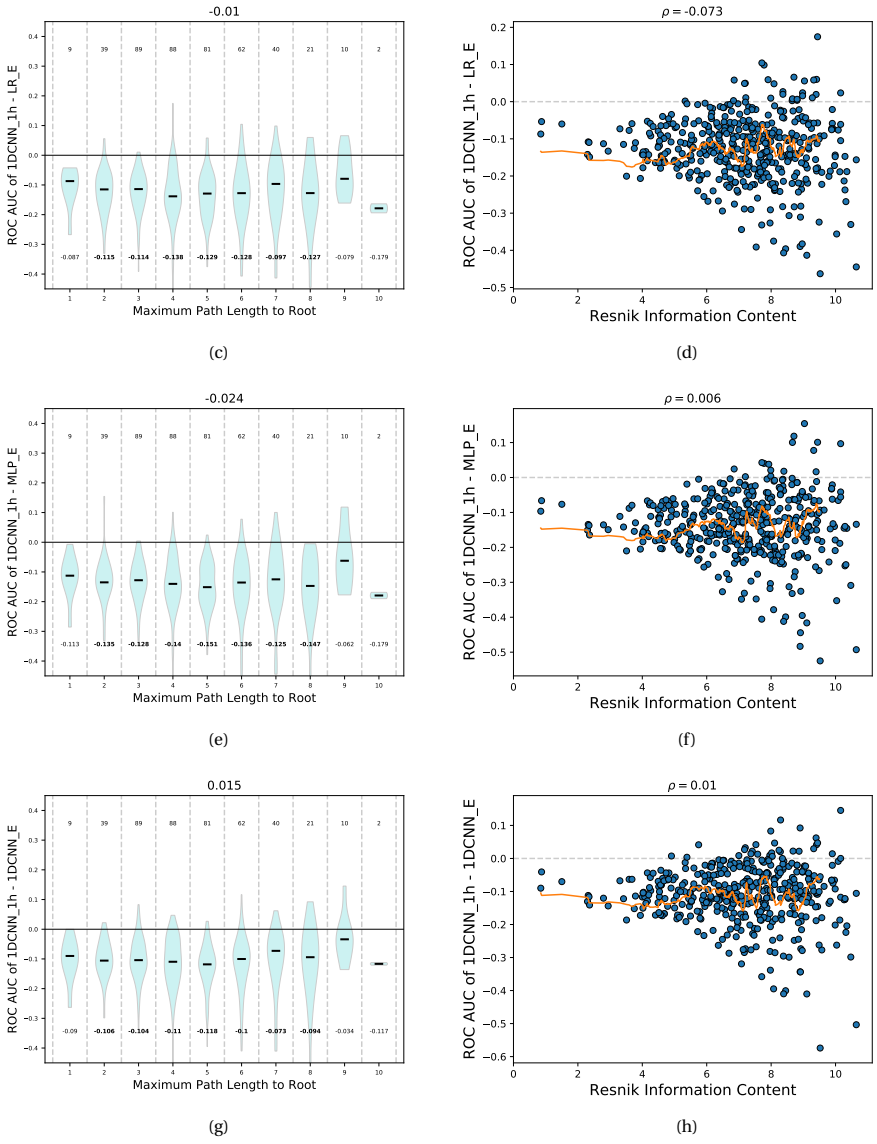
Table 3.S7: F_{max}, S_{min} and $ROCAUC$ of sequence methods on the *SP* test set with 3,530 proteins and C=441 MFO GO terms. The values in brackets indicate 95% confidence intervals estimated using 100 bootstraps. The best performance is indicated in boldface.

Method	$F_{max} \uparrow$	$S_{min} \downarrow$	$ROCAUC \uparrow$
Naive	0.47 [0.463, 0.478]	0.56 [0.548, 0.565]	0.50 [0.500, 0.500]
BLAST	0.39 [0.379, 0.407]	0.52 [0.508, 0.530]	0.67 [0.660, 0.679]
kNN_E	0.52 [0.513, 0.531]	0.46 [0.456, 0.472]	0.76 [0.746, 0.769]
LR_E	0.50 [0.488, 0.507]	0.48 [0.469, 0.484]	0.86 [0.849, 0.867]
MLP_E	**0.56 [0.547, 0.566]**	**0.45 [0.435, 0.452]**	**0.87 [0.860, 0.876]**
1DCNN_E	0.54 [0.527, 0.546]	0.46 [0.448, 0.464]	0.84 [0.832, 0.849]
kNN_1h	0.41 [0.399, 0.415]	0.55 [0.547, 0.558]	0.63 [0.616, 0.637]
LR_1h	0.43 [0.425, 0.441]	0.55 [0.549, 0.560]	0.72 [0.713, 0.731]
MLP_1h	0.43 [0.418, 0.434]	0.55 [0.547, 0.558]	0.75 [0.741, 0.763]
1DCNN_1h	0.43 [0.424, 0.439]	0.54 [0.531, 0.544]	0.73 [0.720, 0.741]

3.5.5. TERM-CENTRIC PERFORMANCE AND TERM SPECIFICITY



(a) (b)



(c) (d) (e) (f) (g) (h)

Figure 3.S7: Difference in term-centric *ROC AUC* between the *1DCNN_1h* model and (a/b) *kNN_E*, (c/d) *LR_E*, (e/f) *MLP_E*, and (g/h) *1DCNN_E* models as a function of term specificity. Left: the maximum path length to the ontology root. For each GO level we plot the distribution of the difference estimated with Gaussian kernels and show its median using a black dash and as text below the plot, where bold values indicate that the difference is significant using a two-sided Wilcoxon rank sum test after Bonferroni multiple testing correction. The numbers above the distributions correspond to the number of GO terms at that level. Right: Resnik information content. Every dot corresponds to a GO term and the orange line shows a local moving average calculated with a window of 30 terms. For both panels, the number at the top denotes the Spearman rank correlation between the difference in *ROC AUC* and the term specificity and a line at $y = 0$ denotes equal performance for both models.

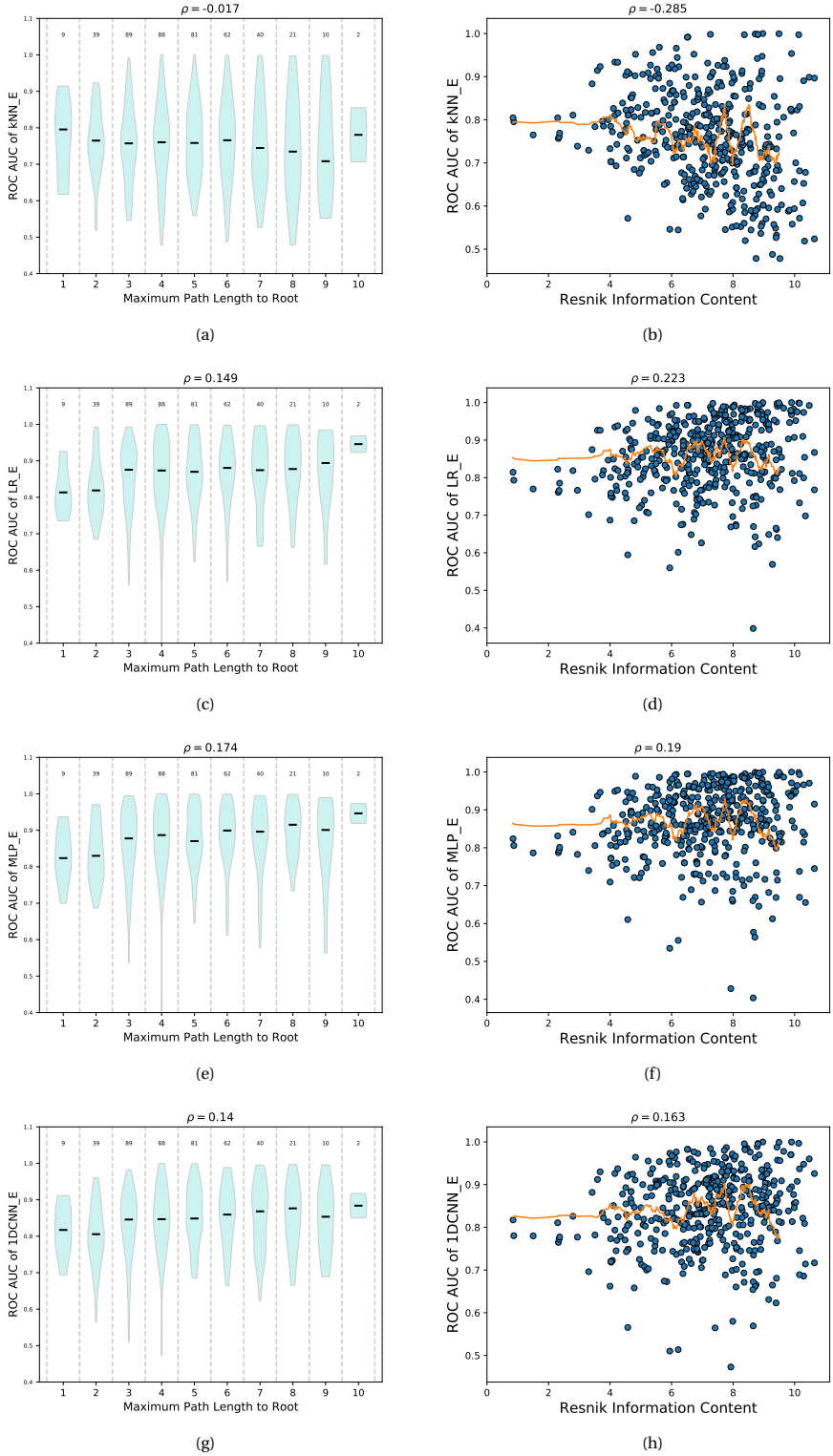


Figure 3.S8: Term-centric *ROC AUC* of (a/b) *kNN_E*, (c/d) *LR_E*, (e/f) *MLP_E*, and (g/h) *1DCNN_E* models as a function of term specificity.

3.5.6. PRINCIPAL COMPONENTS ANALYSIS OF SUPERVISED EMBEDDINGS

Table 3.S8: Rank of the embedding matrices obtained for all available proteins in the *PDB* dataset. The size of the learned embeddings by all methods is 512.

Method	Embedding rank
MLP_E	508
MLP_DF	510
1DCNN_E	511
1DCNN_SA	512
GCN3_E_CM	512
GCN1_E_CM	512
GCN1_CM	105
2DCNN_CM	310

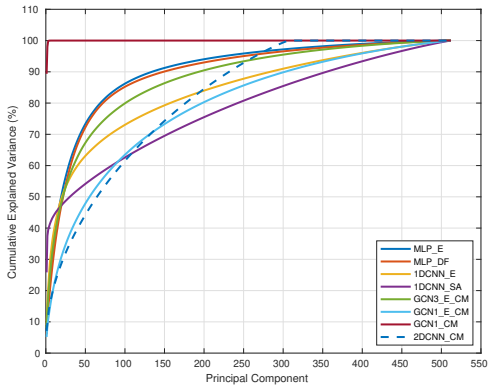


Figure 3.S9: Cumulative percentage of explained variance ($y-$axis) as a function of the number of principal components used ($x-$axis) of the supervised embeddings of different models. Each model is denoted by a different color.

3.5.7. STATISTICAL SIGNIFICANCE OF CORRELATION BETWEEN FUNCTIONAL AND EMBEDDING SIMILARITY

We calculated rank correlations between embedding similarities and functional Jaccard similarities of proteins [47]. Because we are using all pairwise combinations of training and test proteins, the different observations of similarity are not independent, which means that our data do not meet the assumptions for calculating a p-value for the observed correlation. Therefore, we used permutation tests to assess significance.

To test if the observed correlation between embedding similarity and functional similarity is significantly larger than zero, we randomly permuted the GO annotations of the test proteins, by shuffling the rows of the corresponding label matrix. That way, the functional similarity between a training and a test protein is randomized and no longer reflects their functions. We repeated this process 10,000 times, each time calculating the

Spearman correlation between embedding similarity and randomized functional similarity. We thus obtained 10,000 samples from the distribution of the correlation under the null hypothesis, which we then compared to our observed value of $\rho = 0.07$.

To test if correlation values observed for two different embedding similarities (e.g. ELMo and *1DCNN_E*) are significantly different from each other, we randomly exchanged the embedding similarity values for a subset of protein pairs, calculated the two new correlations and kept their signed difference. We repeated this 10,000 times to estimate the null distribution of the correlation difference and compared it to the observed correlation difference.

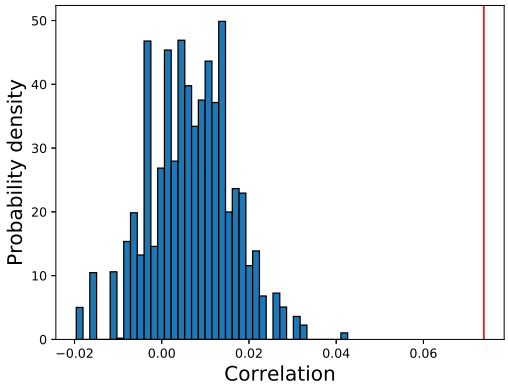


Figure 3.S10: Distribution of Spearman correlations between functional Jaccard similarity and ELMo embedding similarity under the null hypothesis that the two similarities are uncorrelated. The null distribution was estimated using a permutation approach with 10,000 permutations. The red vertical line denotes the truly observed correlation value.

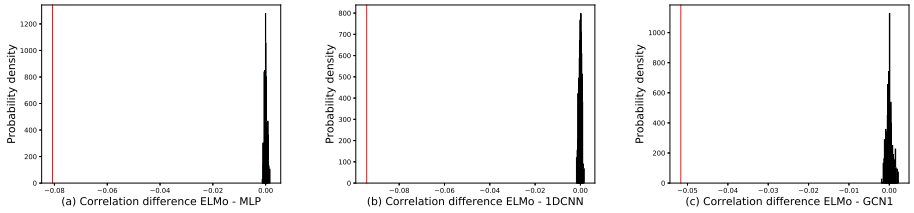


Figure 3.S11: Distribution of the difference in Spearman correlations between functional Jaccard similarity and embedding similarity for a pair of embedding similarities under the null hypothesis that the difference in correlation is zero. The null distribution was estimated using a permutation approach with 10,000 permutations. The red vertical line denotes the truly observed difference in correlation values. In (a), the difference between ELMo and *MLP_E*, in (b) ELMo and *1DCNN_E* and in (c) ELMo and *GCN1_E_CM*.

4

METRIC LEARNING ON EXPRESSION DATA FOR GENE FUNCTION PREDICTION

Stavros MAKRODIMITRIS, Marcel J.T. REINDERS and Roeland C.H.J. VAN HAM

Parts of this chapter have been published in Bioinformatics Volume 36, Issue 4, 15 February 2020, Pages 1182–1190, https://doi.org/10.1093/bioinformatics/btz731, (2020) [1].

ABSTRACT

Motivation: *Co-expression of two genes across different conditions is indicative of their involvement in the same biological process. However, when using RNA-Seq datasets with many experimental conditions from diverse sources, only a subset of the experimental conditions is expected to be relevant for finding genes related to a particular Gene Ontology (GO) term. Therefore, we hypothesize that when the purpose is to find similarly functioning genes, the co-expression of genes should not be determined on all samples but only on those samples informative for the GO term of interest.*

Results: *To address this, we developed MLC (Metric Learning for Co-expression), a fast algorithm that assigns a GO-term-specific weight to each expression sample. The goal is to obtain a weighted co-expression measure that is more suitable than the unweighted Pearson correlation for applying Guilt-By-Association-based function predictions. More specifically, if two genes are annotated with a given GO term, MLC tries to maximize their weighted co-expression, and, in addition, if one of them is not annotated with that term, the weighted co-expression is minimized. Our experiments on publicly available Arabidopsis thaliana RNA-Seq data demonstrate that MLC outperforms standard Pearson correlation in term-centric performance. Moreover, our method is particularly good at more specific terms, which are the most interesting. Finally, by observing the sample weights for a particular GO term, one can identify which experiments are important for learning that term and potentially identify novel conditions that are relevant, as demonstrated by experiments in both A. thaliana and Pseudomonas Aeruginosa.*

Availability: *MLC is available as a Python package at www.github.com/stamakro/MLC*

Contact: *s.makrodimitris@tudelft.nl*

Supplementary information: *Supplementary data are available at Bioinformatics online.*

4.1. INTRODUCTION

Knowing which biological processes and pathways are affected by each gene would be a useful tool for plant biologists and breeders. With this information, they can more easily identify genes that are likely to affect the phenomenon or trait they are studying and prioritize genes for experimental testing. The Biological Process Ontology (BPO) of the Gene Ontology (GO) [2] provides us with a set of terms that describe biological processes at different levels of granularity and can be used to annotate genes from all species in a systematic way. However, the use of computational methods to accurately predict BPO annotations, also known as Automatic Function Prediction (AFP), remains challenging, as demonstrated in the Critical Assessment of Functional Annotation (CAFA) challenges [3].

Most AFP methods use the Guilt-By-Association (GBA) principle. They define a similarity or dissimilarity measure between genes and use it as a proxy for functional similarity. Then, they assign GO annotations to genes of unknown function based on the functions of the genes most similar to them. The choice of similarity measure is always motivated by biology. For instance, sequence similarity points towards a conserved structure which in turn implies similar function. Alternatively, co-expression across different conditions may hint at involvement in the same pathways. Recent results from the third CAFA challenge hint at the great potential of gene expression data to find which genes are involved in a specific biological process [4]. Combining multiple similarity measures in order to better approximate functional similarity is also possible, as done for instance in [5–7].

Genes that are involved in the same biological processes are expected to show similar expression patterns, as they respond similarly to perturbations related to these processes. Discovering BPO annotations for all unannotated genes requires data from a wide range of different experimental conditions. For example, we need samples from different tissues, different time points across development, from wild-type or mutant plants etc. Thanks to world-wide sequencing efforts, more and more RNA-Seq data are becoming available to public databases, such as ArrayExpress [8] and GEO [9].

The Pearson Correlation Co-efficient (*PCC*) is the most widely used measure of gene co-expression similarity and has been largely successful, especially for microarray-derived expression data. For instance, for MS-kNN [5], one of the top-performing methods in CAFA2 [3], the *PCC* was calculated on samples from 392 human microarray datasets to quantify co-expression similarity between genes, outperforming sequence similarity for AFP in BPO [5].

PCC might, however, not be the optimal co-expression measure due to the diversity of biological processes and heterogeneity of public expression datasets. This means that only a subset of all available experimental conditions is likely to be truly informative about a specific GO term. For example, let us assume that we are looking for genes involved in plant immune response. Using the *PCC* across all possible conditions, we implicitly expect that all such genes are expressed similarly not only during immune response, but across all conditions and tissues. However, differential co-expression analysis has shown that several *Arabidopsis thaliana* immune genes, such as FLS2, ADR1 and JAR1, change co-expressed partners before and after infection with *Pseudomonas syringae* [10]. A gene that is co-expressed with immune genes during (only) infection

is still a good candidate gene for immune response, even if it has different expression patterns to the immune genes in other tissues or developmental stages. Including many unrelated expression samples, essentially adds noise to the correlations. According to this reasoning, we should be able to improve the performance of co-expression-based gene function prediction by calculating co-expression only over the samples that are relevant for each term.

This insight that the *PCC* might be suboptimal is not new. For example, Jaskowiak et al. showed that k-means clustering of gene expression data heavily relies on the choice of similarity measure (*PCC*, Spearman correlation, Euclidean distance etc.) and that the most suitable measure varies across different datasets [11]. As another example, Hu et al. showed that using an inappropriate distance metric can really harm the performance of the *k*-Nearest Neighbors (*k*-NN) classifier in biomedical datasets [12].

Adapting a distance measure is a subfield within machine learning that is called metric learning: learning a distance function from a dataset of examples that can most effectively be utilized to perform a task, e.g. discriminating between two classes. It is most explored in combination with the *k*-NN classifier [13]. In the context of AFP, Ray and Misra developed a metric learning method called Genetic Algorithm for Assigning Weights to Gene Expressions using Functional Annotations (*GAAWGEFA*) that learns a weighted *PCC* on microarray data using a genetic algorithm to find the optimal values for the weights [14]. They showed that their weighted correlation increases the protein-centric precision compared to *PCC* in a yeast dataset. Metric learning has also been applied to AFP combined with multiple-instance learning [15]. In that work, each protein is viewed as a "bag of domains" and metric learning is used to learn a distance function between proteins (based on their domains) that is representative of functional similarity.

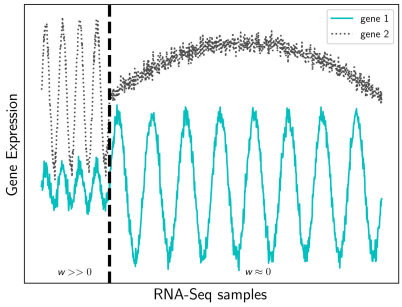


Figure 4.1: Illustrative example of the expression of two hypothetical genes (y-axis, solid and dashed lines) involved in the same biological process over a large set of samples (x-axis). The total Pearson correlation between the genes is 0.09. *MLC* sets large weight values (w_m) for the samples left of the vertical dashed line (where the unweighted correlation is 0.92) and small or zero weights for the samples on the right (unweighted correlation = 0.002).

Here, we use metric learning to identify the most informative conditions for a given GO term. Similar to *GAAWGEFA*, our goal is to assign a weight to every RNA-Seq sample. *GAAWGEFA* learns one weighted *PCC* for all GO terms [14]. On the contrary, our

approach, Metric Learning for Co-expresssion (*MLC*), optimizes the weights per term. Our philosophy (graphically shown in Figure 4.1) is that weights should be chosen in such a way that a pair of genes annotated with the same term should have maximally similar expression profiles, i.e. comply with our assumption that these genes should be co-expressed. On the other hand, when one gene is annotated with the term and the other not, we expect that such a pair should not have high co-expression. In other words, we would like to select weights that minimize the co-expression for these pairs. For pairs of genes both not annotated with the term, we cannot say anything about the co-expression since they might be annotated with another term, and thus be co-expressed too (albeit for other conditions/samples). Consequently, the co-expression of these pairs should be ignored when optimizing weights for the GO term of consideration. A high weight for a sample will put emphasis on that sample when calculating the co-expression over all samples, whereas a low weight for a sample will reduce the influence of that sample. When a weight becomes zero the sample is even ignored. To enforce selecting informative samples, we additionally apply an L1 sparsity constraint on the weights, which will set a weight to zero when a sample is uninformative [16]. In contrast to *GAAWGEFA*, where they have used a genetic algorithm to find the weights, we are able to pose the weight optimization in an elegant mathematical formulation that can be minimized efficiently using standard methods. To reduce the computational burden even further, we use the weighted inner product as a similarity function instead of the weighted *PCC*. We evaluate our algorithm on public RNA-Seq data from *A. thaliana* and on the microarray data from *Pseudomonas aeruginosa* that were used for the CAFA-π challenge, which included experimental validation of gene functions. [4].

4.2. METHODS

4.2.1. DATA AND PREPROCESSING

We used the API of the European Bioinformatics Institute (EBI) [17] to download all *A. thaliana* RNA-seq studies available at ArrayExpress [8]. All samples had been processed using the same pipeline and expression was measured using raw read counts. We restricted our dataset to samples that used the latest version of the *A. thaliana* genome (TAIR10) and had fewer than 10% unmapped reads. After removing duplicate experiments, we had 4,215 samples from 298 different studies (batches) for 32,833 genes, 26,925 of which were protein-coding. We used a preprocessing pipeline similar to the one used to construct the ATTED-II RNA-Seq co-expression network [18]. We first removed samples with fewer than 10,000,000 mapped reads. Then, we removed lowly expressed genes (genes with maximum expression over the remaining samples less than 100). To diminish the zero-inflation of the dataset, we also removed all genes that were not expressed in at least half of the samples (median expression smaller than 1). Finally, we log-transformed the expression counts using a pseudocount of 0.125. Supplementary Material 1 and Figure 4.S1 describe the results when excluding the filtering of low-coverage samples.

We then mapped the TAIR gene ID's to UniProt ID's. BPO annotations were downloaded from GOA (https://www.ebi.ac.uk/GOA) in September 2016 and annotations with the IEA evidence code were removed. A total of 2,978 samples and 6,013 genes with

BPO annotations remained after these filtering steps. We applied ComBat [19] to remove unwanted variation stemming from the fact that the different samples come from different studies (batch effects). ComBat uses a Bayesian method to standardize the mean and the variance of each gene in each study (batch). In order to be able to estimate within-batch variances, we removed all studies that had only one sample, leaving us with 2,959 samples (Supplementary Material 2).

For the experiments on *P. aeruginosa* we used a pre-processed microarray expression compendium from [20]. This dataset contains the expression of 5,549 genes measured over 1,051 samples. The gene annotations were downloaded from the Supplementary Material of the CAFA-π paper [4].

4.2.2. NOTATION

We use $\mathbf{x}_i \in \mathbb{R}^f$ to denote the expression of gene i across all $f = 2,959$ samples. x_{im} is the expression of gene i at sample m. $\overline{x}_i$ is the mean of gene i across all samples. Given N genes and a GO term l, we denote as $\mathbf{y}(l) \in \{0,1\}^N$ the vector with the class labels of the genes, with $y(l)_i = 1$ iff gene i is annotated with l. The sample weights are represented by a vector with f non-negative elements $\mathbf{w}(l) \in [0, +\infty)^f$.

4.2.3. WEIGHTED AND UNWEIGHTED MEASURES OF CO-EXPRESSION

The most widely-used measure of co-expression between two gene expression vectors $\mathbf{x}_i, \mathbf{x}_j$ is the Pearson Correlation Coefficient (*PCC*), which is defined as follows:

$$PCC(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{m=1}^{f}(x_{im} - \overline{x}_i)(x_{jm} - \overline{x}_j)}{\sqrt{\sum_{m=1}^{f}(x_{im} - \overline{x}_i)^2}\sqrt{\sum_{m=1}^{f}(x_{jm} - \overline{x}_j)^2}} \tag{4.1}$$

Note that the numerator is the covariance between $\mathbf{x}_i$ and $\mathbf{x}_j$ and the denominator is the product of the standard deviations of the two vectors.

A related, but simpler measure is the inner product similarity (S), which, on the contrary, is sensitive to the mean expression of both genes:

$$S(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \cdot \mathbf{x}_j = \sum_{m=1}^{f} x_{im} x_{jm} \tag{4.2}$$

If two vectors $\mathbf{x}_i$, $\mathbf{x}_j$ both have zero mean and unit L2-norm, then their *PCC* is equal to their inner product. This equality does not hold anymore if we weigh each vector element (sample) differently. However, since the two metrics are related, we chose to use the weighted inner product similarity instead of the weighted *PCC* as our expression similarity function in order to simplify the problem. We center and scale our data so that the (unweighted) mean of every gene is zero across all conditions and its (unweighted) L2-norm is equal to one:

$$\tilde{\mathbf{x}}_i = \frac{\mathbf{x}_i - \overline{x}_i}{\|\mathbf{x}_i - \overline{x}_i\|} \tag{4.3}$$

Then, we define our similarity function as the weighted inner product of the two

scaled expression vectors ($S_{\mathbf{w}}$):

$$S_{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_j) = \tilde{\mathbf{x}}_i^T \cdot W \cdot \tilde{\mathbf{x}}_j = \sum_{m=1}^{f} w_m x_{im} x_{jm} \tag{4.4}$$

Where $W = diag(\mathbf{w})$ is a diagonal matrix containing the sample weights.

4.2.4. METRIC LEARNING FOR CO-EXPRESSION (*MLC*)

The rationale for learning the weights is to maximize the performance of the k-NN classifier. For this purpose, we want the expression similarity between two genes that are both annotated with a given GO term l to be higher (on average) than the similarity between a gene that is annotated with l and a gene that is not. We group each gene pair into one of the following three categories:

1. both genes are annotated with l (we call these "positive-positive pairs" or "p-p"),
2. exactly one of the two genes is annotated with l ("positive-negative pairs" or "p-n"), and
3. neither gene is annotated with l ("negative-negative pairs" or "n-n").

Our goal is to find the weight values w_m that maximize the separability between "p-p" and "p-n" pairs. Let $\mu_{p-p}, \sigma^2_{p-p}$ denote the mean and variance of the weighted similarity value S_w of all "p-p" gene pairs and, similarly, μ_{p-n}, σ^2_{p-n} for all "p-n" gene pairs. Let also N_{p-p} and N_{p-n} denote the number of gene pairs in each category. We use Welch's two-sample t-statistic with unequal variances to quantify the notion of separability:

$$t(\mathbf{w}) = \frac{\mu_{p-p} - \mu_{p-n}}{\sqrt{\frac{\sigma^2_{p-p}}{N_{p-p}} + \frac{\sigma^2_{p-n}}{N_{p-n}}}} \tag{4.5}$$

Note that μ and σ^2 are functions of $\mathbf{w}$, but this dependence is not shown explicitly in equation 5 to keep the notation simple. Maximizing $t(\mathbf{w})$ is equivalent to minimizing $-t(\mathbf{w})$. In order to enable sample selection, we also added an L1 regularization term that forces the weights of uninformative samples to zero (Tibshirani, 1996). Our optimization problem then becomes:

$$\min_{\mathbf{w}}[-\alpha\, t(\mathbf{w}) + (1-\alpha) \sum_{m=1}^{f} w_m], \quad s.t. \quad w_m \geq 0 \forall m \tag{4.6}$$

Parameter α controls the trade-off between the actual cost and the regularization. The minimization of Equation 4.6 is done with the Broyden-Fletcher-Goldfarb-Shanno method [21].

GLOBAL *MLC*

To investigate the effect of creating GO-term specific predictors, we also implemented a version of *MLC* that is applicable to all terms simultaneously. To this purpose, we redefined "p-p" gene pairs as pairs of two genes which share at least one GO term and "p-n" pairs as pairs of two genes that share no GO annotations. All the ensuing steps remain the same as for the term-specific *MLC*. We call this method "Global *MLC*" (MLC_G).

4.2.5. EXPERIMENTAL SET-UP

We compared *MLC* to the unweighted *PCC* baseline. To investigate the effect of the use of a term-specific classifier, we created term-specific classifiers from the *PCC* by tuning the classifier parameter k individually per GO term and not globally over all terms. We called this approach *PCC(k)*. We also compared to *GAAWGEFA* which, like *MLC*, learns a weighted co-expression measure [14]. *GAAWGEFA* is not GO-term-specific and optimizes the mean protein-centric precision using a genetic algorithm, so we also constructed a non-term-specific version of *MLC* (MLC_G) to compare against. Another way to measure co-expression is the Mutual Rank (*MR*) [18] which is used in the ATTED-II database. Although *MR* neither selects samples nor weighs samples differently, it has been shown to outperform the *PCC* for function prediction [18], so we included it in the comparison as a stronger baseline. Input to *MR* are typically the *PCC* values of all gene pairs, although it can be applied to any co-expression measure. More details on the definition and implementation of each of these methods are given in Supplementary Material 3.

We evaluated the methods in three ways: 1) A cross-validation experiment using all *A. thaliana* genes with at least 1 GO annotation (reported as "CV results"), 2) a time-course experiment using the preliminary test set from CAFA3 containing 6,077 training and 90 test genes also from *A. thaliana* (reported as "CAFA3 results") and 3) the dataset of CAFA-π from bacterium *P. aeruginosa*, where the goal was to make term-specific predictions for biofilm formation and motility [4]. For the cross validation experiment, we used nested cross-validation [22], using the inner loop to optimize the parameters and the outer loop to evaluate performance on previously unseen genes (Figure reffig:s2, Supplementary Material 4). As in this work we are dealing with the problem of identifying which genes should be annotated with a specific GO term, we focus on term-centric evaluation using the mean $ROC\,AUC$. Details about the three evaluation modes are provided in Supplementary Material 4 and details about the used term-centric evaluation metrics as well as protein-centric metrics that we also used are given in Supplementary Material 5.

4.3. RESULTS

4.3.1. ALL METHODS OUTPERFORM THE *PCC*

We compared our metric learning approach (*MLC*), as well as Mutual Rank (*MR*) and *GAAWGEFA* to the standard, unweighted *PCC* using 3-fold cross-validation. *PCC* achieved a mean term-centric $ROC\,AUC$ of 0.69, while the performance of both *MR* and *MLC* with the weighted inner product was 0.72 (Table 4.1). The performance of *GAAWGEFA* was 0.71. MLC_G, the non-term-specific version of *MLC*, also achieved a mean $ROC\,AUC$ of 0.72. Although all methods perform fairly similarly according to protein-centric measures (Tables 4.S1-4.S2, Supplementary Material 6), *PCC* performs significantly worse than the other methods on term-centric $ROC\,AUC$ (False Discovery Rate (FDR) < 0.036, Table 4.S3-4.S6, Supplementary Material 7, effect size 4%). This shows that the *PCC* is indeed a suboptimal co-expression measure. When randomly permuting the GO annotations of the genes, both *PCC* and *MLC* had a mean $ROC\,AUC$ of 0.5, i.e. equal to random guessing, implying that *MLC* does not artificially generate information in the absence of real structure (Supplementary Material 8).

Table 4.1: Mean term-centric *ROC AUC* achieved by the methods under comparison using 3-fold cross-validation (CV, 2nd and 3rd column) and when testing on the CAFA3 dataset (CAFA3, 4th and 5th column). For the cross-validation, we report the average performance over the three folds as well as the corresponding standard error. For the CAFA3 results we report the performance on the test set as well as the 95% Confidence Intervals from doing 1,000 bootstrapped tests.

Method	*ROC AUC* (CV)	Weighted *ROC AUC* (CV)	*ROC AUC* (CAFA3)	Weighted *ROC AUC* (CAFA3)
PCC	0.69 ± 0.003	0.69 ± 0.003	0.68 [0.63, 0.72]	0.68 [0.63, 0.73]
PCC(k)	0.69 ± 0.003	0.69 ± 0.003	0.68 [0.63, 0.71]	0.68 [0.63, 0.72]
PCC + MR	0.72 ± 0.002	0.72 ± 0.002	0.69 [0.65, 0.73]	0.69 [0.65, 0.73]
GAAWGEFA	0.71 ± 0.002	0.71 ± 0.002	0.69 [0.65, 0.73]	0.70 [0.65, 0.74]
MLC (Sw)	0.72 ± 0.003	$\mathbf{0.73 \pm 0.003}$	0.69 [0.65, 0.73]	0.69 [0.65, 0.73]
MLC_G	0.72 ± 0.003	0.72 ± 0.003	**0.71 [0.67, 0.75]**	**0.72 [0.67, 0.76]**
MLC-MR Hybrid	$\mathbf{0.73 \pm 0.005}$	$\mathbf{0.73 \pm 0.005}$	0.69 [0.65, 0.73]	0.69 [0.66, 0.73]

4.3.2. *MLC* IS THE BEST AT PREDICTING SPECIFIC GO TERMS

Although *MR*, *GAAWGEFA* and *MLC* perform equally on average, one is typically not interested in predicting GO terms that are "near" the ontology root, as most of them describe too general biological processes [23]. Therefore, we compared the performances of these methods as a function of term specificity. As measures of specificity, we used the maximum path length to the ontology root and the Resnik Information Content (IC) [24]. One way to take term specificity into account is to calculate the weighted term-centric *ROC AUC*, where each term is weighted by its IC when calculating the average. As shown in Table 4.1, *MLC* achieves the highest weighted *ROC AUC*. The difference is statistically significant for all methods except for *MR* (Table 4.S4, Supplementary Material 7), although the effect size is small (1.5%).

Furthermore, we grouped the GO terms into quintiles (quantiles at 0, 20, 40, 60 and 80%) and plotted the distribution of the percent differences in performance of *MLC* from *MR* for each quintile (Figure 4.2a). We observed that for the first two quintiles (i.e. the 40% most frequent terms), *MLC* performs worse than *MR*, while for the 60% most specific terms, both the mean and the median performance of *MLC* is better. Further analysis showed that for the very general terms, *MLC* makes a lot more type I errors (false positives) than for the more specific ones (Figure 4.S3, Supplementary Material 9) and that makes it underperform with respect to *MR*. The Spearman correlation between percent difference and Resnik IC was 0.26. The same pattern is evident when comparing *MLC* to all other methods (*PCC*, *GAAWGEFA* and MLC_G), as well as when replacing Resnik IC with the path length to the ontology root (Tables 4.S7-4.S8, Figures 4.S4-4.S5, Supplementary Material 10). From that we can conclude that term-specific *MLC* is the preferred method for finding genes belonging to rarer terms.

4.3.3. *MLC* TUNES THE WEIGHTS TO FIND "*p-p*" PAIRS

The goal of *MLC* is to choose the weights so that for test genes that have a particular GO annotation, the learned similarities are higher to training genes that have the same annotation than to genes that do not. As an example, Figures 4.2b and c show the distribution of co-expression similarities between the test genes annotated with term

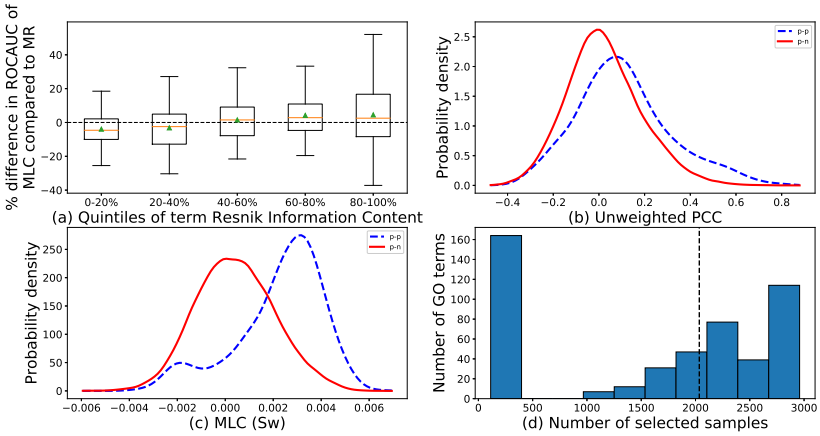


Figure 4.2: (a) Percent increase in *ROC AUC* of *MLC* (S_W) with respect to *MR* as a function of Resnik Information Content. For each set of terms in each quintile of Information Content, the corresponding box includes the two middle quartiles of the percent increase for these terms. An orange line denotes the median. The error bars extend to 1.5 times the range of the two middle quartiles. Note, that the 0-20% quintile corresponds to the 20% least specific terms and the 80-100% quintile to the 20% most specific ones. (b-c) Distributions of co-expressions for genes annotated with term GO:1903047. In dashed blue lines, the co-expression values between a test and a training gene that both are annotated with that term. In solid red lines, the co-expression between test genes annotated with that GO term and training genes that are not. Co-expression is measured as the *PCC* (b) and the S_W trained by *MLC* (c). The *x*-axis shows the co-expression values and the *y*-axis the probability density estimated with Gaussian kernels. Note that the *PCC* and S_W have different ranges due to the weight optimization. (d) Histogram of the number of samples that were selected for each GO term. The *x*-axis corresponds to the number of selected samples and the *y*-axis to how many GO-term-specific similarity functions selected that many samples. The dashed line denotes the median number of non-zero weights.

GO:1903047 (mitotic cell cycle process) and all training genes for the *PCC* and *MLC* similarities respectively. It is clear that for *MLC*, the test genes are a lot more similar to training genes with the same annotation. Note, however, that, for this term, a significant portion of the similarities are negative (small blue peak in Figure 4.2b). This means that some positive genes are anti-correlated to the rest. For these cases *MLC* will make Type II errors (false negatives).

Figure 4.2d shows the distribution of the number of selected samples for each GO term. For about 33% of all GO terms, *MLC* selected less than 9% of the available samples (252 or fewer), setting all other weights to zero. The median number of selected samples was 2,035 out of 2,959 or about 69% of all samples. Randomly selecting samples did not improve the mean performance of the *PCC* (Supplementary Material 11), implying that the correct samples have to be selected for each term. Moreover, for about 23% of the terms, *MLC* kept all the samples and weighted them more or less equally (maximum standard deviation of weights = 0.006), in which case *MLC* was almost equivalent to the unweighted inner product. As expected, for those terms *MLC* had on average similar performance to the baseline *PCC*. We also found that individually tuning k per GO term for the *PCC* gave on average the same term-centric *ROC AUC* as the baseline *PCC* (*PCC (k)*,

Table 4.1), so the performance improvement is not caused by simply choosing the optimal k value for each GO term. Finally, we observed a small negative correlation between term Information Content and the number of samples selected (Spearman $\rho = -0.09$, p-value = 0.057). This means that *MLC* has a slight tendency to select fewer samples for more specific terms, but this result is not statistically significant.

4.3.4. THE WEIGHTS LEARNED BY *MLC* HELP AT IDENTIFYING RELEVANT EXPERIMENTAL CONDITIONS

The weights learned by *GAAWGEFA* are roughly uniformly distributed between 0 and 1 (Kolmogorov-Smirnov test statistic = 0.011, p-value = 0.847) and are not correlated to any of the term-specific weight profiles of *MLC*, which tend to have an exponential-like distribution (Figures 4.S6-4.S7, supplementary Material 12), as many samples get a weight of zero. Furthermore, samples from the same study (batch) tend to be either selected or not selected together by *MLC* (Table 4.S9, Supplementary Material 13).

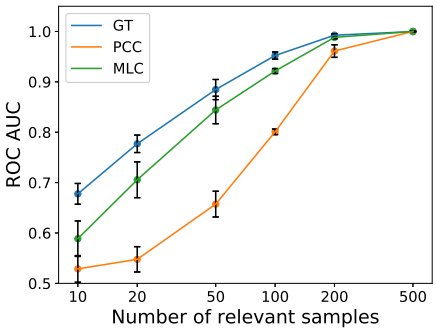


Figure 4.3: *ROC AUC* (y axis) on simulated data as a function of the number of relevant samples (x axis in log scale) for *MLC* (green), *PCC* (orange) and the *PCC* using only the ground-truth samples (blue). The error bars denote the standard deviation over 5 repetitions.

We used simulated data to validate the sample selection of *MLC*. We created an artificial dataset with 7,000 genes, 3,000 samples and three GO terms, where the genes with a particular GO term are correlated over a predefined set of samples. The sets of informative samples are of equal length and do not overlap (Figure 4.S8, Supplementary Material 14). We varied the number of informative samples from 10 to 500 and compared *MLC* to the *PCC* calculated over all samples and the *PCC* calculated only over the informative ones (which can be seen as the optimal, ground-truth performance). As seen in Figure 4.3, when up to 200 samples drive the similarity, *MLC* performs considerably better than the *PCC* and only slightly worse than the ground-truth *PCC* (*GT*). When the number of informative samples increases, both *MLC* and *PCC* increase performance, eventually both converging to *GT*. Yet, *MLC* converges much faster, thus achieving *GT PCC* performance with fewer informative samples. We also found that in all cases, and for all numbers of relevant samples except 500, the *MLC* weights of the ground-truth samples were on average significantly larger than those in the rest of the samples (two-sample t-test, FDR < 0.01, Figure 4.S9a, Supplementary Material 14). Moreover, there was a sig-

nificant enrichment of the informative samples in the samples selected by *MLC* (Fisher's exact test, FDR < 0.01, Figure 4.S9b, Supplementary Material 14). When the number of informative samples increased to 500, *MLC* selected all the samples with rather similar weights (Figure 4.S9) and performed consistently equal to both *PCC* and *GT*.

Continuing on the example from before, for term GO:1903047 (mitotic cell cycle process) *MLC* gives the highest weight to a sample of a plant grown in the absense of phosphorus, which has been shown to restrict the cell division rate [25]. Among the samples with highest weights are also many samples from experiments studying seed germination, a process closely linked to cell cycle [26]. Finally, two *IBM1* mutant samples were selected with very high weights. The *IBM1* gene codes for a histone demethylation protein and has GO annotations that include flower, root and pollen development. The complete weight profile is shown in Figure 4.S10 (supplementary Material 15).

As another example, we looked at "regulation of flower development" (GO:0009909). For this term, *MLC* selected 164 samples and achieved ROC AUC of 0.74 while the *PCC* score was 0.55. The top scoring sample came from an AS1 mutant plant. AS1 is a well-known transcription factor, key in many developmental processes including flowering [27]. Several wild-type samples were selected, many of them from meristems, tissues which contain undifferentiated cells and drive tissue differentiation. In the top-10 samples, we also found a CORYNE mutant (CORYNE is involved in the signaling of cell differentiation in flowers [28]) and an AGO1 mutant, a gene crucial for miRNA-based mRNA splicing [29]. Finally, 3 out of the top-10 samples were from knock-outs of well-known DNA methylation genes. Chen et al. were - to our knowledge - the first to extensively study the interplay between epigenetics, transcription factors and miRNAs in flower development, which is otherwise largely not understood [30]. *MLC* was able to highlight that all these three regulatory mechanisms are at the same time informative for understanding flower development, without using the data from that work or any Chip-Seq or miRNA data.

Together these examples highlight that *MLC* has the ability to identify novel experimental conditions for a specific GO term. Moreover, the weights learned by *MLC* are also consistent with the ontology structure and the existing annotations, as shown in Figure 4.S11 in Supplementary Material 16.

4.3.5. USING ALL SAMPLES OBSCURES CO-EXPRESSION

Next, we investigated the terms for which *MLC* performed sample selection, i.e. assigning a non-zero weight to at most 9% of the samples. We looked at the *PCC* values for "*p-p*" and "*p-n*" gene pairs. Figure 4.4a shows an example of the distributions of the *PCC* values for "*p-p*" and "*p-n*" pairs for term "GO:1903047" (mitotic cell cycle process). Next, we calculated the *PCC* for all "*p-p*" and "*p-n*" pairs, but only considering the samples that were selected by *MLC* (i.e. had a non-zero weight assigned), as well as only considering the samples that were not selected by *MLC* (i.e. were assigned a weight of zero), shown in Figures 4.4b-c respectively. One can notice that the two distributions ("*p-p*" and "*p-n*" *PCC* values) differ more when taking the *MLC* selected samples in consideration (compare Figure 4.4a with Figure 4.4b). When only considering the samples that were not selected, the two distributions differ in a similar way to when all samples are being considered (compare Figure 4.4a with Figure 4.4c).

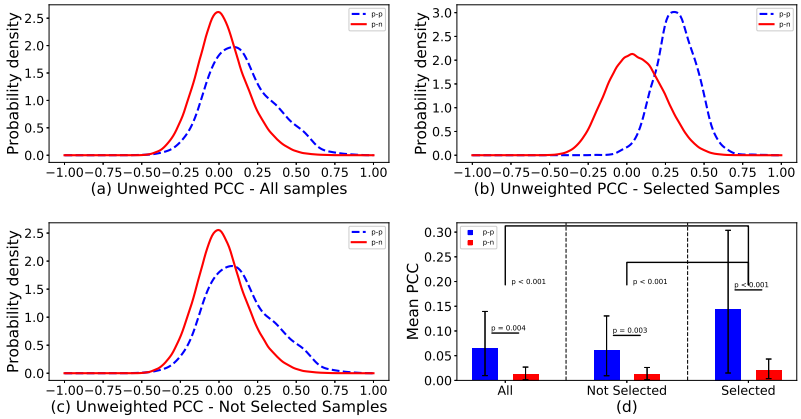


Figure 4.4: (a-c): Distributions of Pearson correlations for pairs of training genes that are both annotated with term GO:1903047 ("*p-p*", blue dashed), and for pairs of training genes of which only one is annotated with that term ("*p-n*", red solid). The correlations are calculated using all samples (a), the samples that were selected by *MLC* (b) and the samples that were not selected (c). (d): The means of the distributions of (a-c), over all GO terms where less than 10% of the samples where used to calculate the co-expression (more than 90% zero weights). Values for "*p-p*" pairs are colored blue and for "*p-n*" pairs red. The error bars show the 95% confidence intervals for the means, calculated with 1,000 bootstraps. Above the bars the bootstrap p-values are shown for the pairwise comparisons.

For every GO term for which *MLC* performs sample selection, we calculated the mean *PCC* of all "*p-p*" and "*p-n*" pairs under the three sample sets (all samples, not selected samples and selected samples). Figure 4.4d shows the average of these values of all GO terms. We also performed 1,000 bootstraps, sampling GO terms with replacement to obtain 95% confidence intervals for these averages. We observed that the difference in mean co-expression between "*p-p*" and "*p-n*" in the samples not selected by *MLC* is similar to the difference in all the samples. Although these differences are statistically significant, they are also significantly smaller than the difference in the *MLC*-selected samples (bootstrap p-value <0.001) (Figure 4.4d). This means that although the whole dataset does contain a few samples that are informative for these GO terms, calculating the co-expression over a larger set of samples can corrupt the "real" co-expression signal, increasing the difficulty of discovering new genes that play a role in these processes.

4.3.6. COMBINING MUTUAL RANK AND *MLC*

The performances of *MLC* and *MR* are positively correlated (Spearman $\rho = 0.13$, p-value =0.003). We also applied *MR* on the co-expression similarities obtained with *MLC*, as *MR* is in principle not restricted to using only the *PCC*. We found a small improvement compared to standalone *MLC*, with a mean ROC AUC of 0.73. Also, the performances of *MLC* and *MLC* + *MR* were highly correlated (Spearman $\rho = 0.97$, p-value $\ll 10^{-20}$).

We tried another approach to combine *MLC* and *MR* depending on the performance of the methods. If the training *ROC AUC* of *MR* was larger than 0.8 for a GO term, we

used the predictions of *MR* for that term, otherwise we used the predictions of *MLC*. This combined classifier had an incrementally larger term-centric *ROC AUC* (0.73, Table 1 - Hybrid), though statistically significant (p-value = 0.008, two-sample t-test). The threshold of 0.8 training *ROC AUC* was chosen arbitrarily and was not tuned to maximize performance. This naïve hybrid classifier shows that there is potential to improve performance by combining *MLC* and *MR* in more sophisticated ways.

4.3.7. CAFA RESULTS

Moreover, we benchmarked *MLC* on 90 temporary *A. thaliana* targets from the CAFA3 competition. The results are similar, both in absolute numbers and the ranking of the methods (Table 4.1), with both *MR* and *MLC* outperforming the *PCC* on average. However, due to the small size of the dataset, the confidence intervals are much wider, so no significant conclusions can be drawn.

Lastly, we compared *MLC* and *MR* to *PCC* in the term-centric challenge CAFA-π for *P. aeruginosa*, where *PCC* on microarray data was the top-performing method [4]. The goal of the challenge was to predict proteins involved in motility and biofilm formation and the ground truth was obtained using genome-wide assays. Note that these two terms are relatively frequent (about 14% and 12% of the tested genes were annotated with biofilm and motility respectively). Comparing them to our experiments in Arabidopsis (Figure 4.2a), they would fall in the left-most, least informative bin, where *MLC* is expected to be on average slightly worse than *MR* and about similar to *PCC*. The results show that *PCC*, *MR* and *MLC* achieve similar *ROC AUC* (around 0.6 for both biofilm and motility, which is also very similar to the performance reported in [4]) (Table 4.S10, Supplementary Material 17). However, *MLC* selected samples from relevant conditions (Supplementary Material 17), enabling interpretability of the predictions. We also examined the genes for which *MLC* was most confident for their involvement in either biofilm formation or motility, but they were not confirmed by the assay (i.e. false positives) and found evidence in the literature that indeed these genes are likely part of these processes under certain biological conditions. For example, two genes in the top-10 for biofilm formation were annotated with the biofilm pathway in KEGG (more detailed discussion and interpretation can be found in Supplementary Material 17). Interestingly, these genes received low scores by *PCC* and *MR* (see for example Figure 4.S12, Supplementary Material 17).

4.4. DISCUSSION

4.4.1. *MLC*

We introduced *MLC*, a metric learning method for building automatic function predictors from a large collection of expression data. *MLC* calculates gene co-expression by assigning GO-term-specific weights to each sample. The weights aim at maximizing the co-expression similarity between genes that are annotated with that GO term. In general, training GO-term specific classifiers (also known as the "Binary Relevance" approach in the machine learning literature) has the disadvantage that individual classifiers fail to see the "bigger picture" and cannot exploit the correlations between terms imposed by the ontological structure. Several works on multi-label classification have shown that Bi-

nary Relevance performs worse than models that incorporate label correlations [31–33]. Despite this, we showed that the weight profiles learned by *MLC* do correlate with real biological knowledge, such as semantic similarity in the ontology graph and gene annotation similarity, meaning that our method is powerful enough to capture at least some of the label similarities even though it was not exposed to them. Due to the use of the L1 regularization, *MLC* can also select informative samples by setting the weights of non-informative samples to zero. Moreover, we showed that the samples that are selected come from biological conditions relevant to the GO term in question.

Our method is designed to work well with a Guilt-By-Association approach like the *k*-NN classifier. This classifier assigns a GO term to a test gene if a large enough fraction of its top co-expressed training genes are annotated with that term. To achieve this, *MLC* tries to maximize the difference between the average co-expression between gene pairs that are both annotated with the GO term of interest ("*p*-*p*" pairs) and the average co-expression between gene pairs only one of which is annotated with the term ("*p*-*n*" pairs). During the training phase, our model ignores gene pairs where neither gene has the term of interest ("*n*-*n*" pairs). Such pairs could either include two genes that have common GO annotations, but different from the GO term of interest or two genes with completely different annotations. For the first case, one might be tempted to think that the co-expression of such pairs should be high. However, if their common function is different from the term of interest, it is likely that they are correlated for another set of samples than the one related to the GO term of interest, and, consequently, are thus uninformative for that GO term. For the second type of "*n*-*n*" pairs, the ones that share no annotations whatsoever, it might make sense to want their co-expression to be 0, as they are expected to be dissimilar over any set of samples. However, we decided to ignore these pairs as they do not add any term-specific information, so it is not clear how they will affect the identification of samples specifically relevant for a specific term. This might be problematic as for a negative test gene (i.e. a gene that should not be annotated with the GO term of interest) we cannot exclude that it can be as highly co-expressed to positive as to negative genes, because we did not tune the co-expression values for "*n*-*n*" pairs. For very frequent terms with a lot of positive training genes this leads to a lot of false positive predictions, which might explain the poor performance of *MLC* for frequent terms.

The similarity function that we used as a basis for *MLC* is the weighted inner product ($S_{\mathbf{w}}$). We chose this measure because its unweighted version is identical to the unweighted *PCC* for centered and scaled data, but it has a simpler form which eases the computational burden. The weighted versions of the inner product and *PCC* are no longer identical, as the data are no longer scaled after weighing the samples. This has as side-effect that the similarity functions that *MLC* learns are not necessarily in the range $[-1, 1]$, like the *PCC*. In most cases, their range is much narrower as can be seen in Figure 4.2b for GO:1903047. Also, because of the range differences, it is not trivial to compare the similarity of two genes across different GO terms. For the purpose of classification with the *k*-Nearest Neighbors classifier, however, the range of the metric is insignificant (only the relevant rankings are important to find the proper neighborhood).

Our model is more general and not restricted to only the inner product, though. The main idea is to maximize the difference between the similarity of *p*-*p* and *p*-*n* pairs. This

is done by maximizing the t-statistic between the two distributions of similarities. This means that *MLC* can also be applied to any measure of similarity such as the weighted *PCC*, weighted Spearman correlation, Euclidean distance etc.. Regardless of the chosen metric, the two classes ("p-p" and "p-n") do not meet the assumptions for applying Student's t-test, as the similarity values are neither normally distributed nor independent. This is not an issue, though, because we do not use the t-statistic to compute a p-value (exploit that the t-statistic is distributed according the Student's t-distribution under these assumptions), but only to quantify the class separability [34]. Equivalently, we could have used any other measure of class separability, for instance the Fisher Discriminant Ratio [35] or the Davies-Bouldin index [36].

4.4.2. COMPARISON TO RELATED METHODS

Our work validates the observation that *PCC* is not the optimal co-expression measure for AFP. The Mutual Rank (*MR*) attempts to obtain more robust and noise-free co-expression values by converting the *PCC* values into ranks and averaging the reciprocal rankings of two genes [18]. *MLC* takes a fundamentally different approach, operating on the sample level rather than the correlation level. First and foremost, as we mentioned above, it removes samples that do not help at discriminating between genes that do or do not perform a certain function. With that *MLC* gives insight into which samples are important for a given GO term, which subsequently can be used to investigate the expression patterns of the GO term related genes across these samples. Weighing samples differently can also be viewed as a way of denoising. For example, it can compensate for the issue that an expression change of 1 unit has a different meaning in different samples due to technical variations, such as differences in sequencing depth or sample preparations. Our results have shown that *MLC* is more beneficial than the *MR* approach for the more specific - and arguably more useful - GO terms.

A similar method to *MLC* is *GAAWGEFA*, which learns a weight for each sample in a dataset and then applies a weighted Pearson correlation. There are two fundamental differences between the two methods. Firstly, *GAAWGEFA* aims at good protein-centric performance, i.e. it tries to do well on average for all genes and therefore learns only one set of sample weights. On the other hand, *MLC* aims at maximizing the performance for each GO term individually. Secondly, *GAAWGEFA* learns the weights using a genetic algorithm. For *MLC*, we used the inner product, which allowed us to have a simple optimization problem that can be solved very efficiently. Even though *MLC* has to be run for each term separately, it is still 67% faster than *GAAWGEFA* and, unlike *GAAWGEFA*, runs for different GO terms can be carried out in parallel to achieve even greater speed-up. Next to those differences, *MLC* makes more accurate predictions for rarer terms and provides interpretability of the predictions by examining the term-specific sample weight distributions.

Furthermore, in the context of selecting expression samples a related technique is biclustering. Biclustering is an umbrella term for a diverse set of algorithms that simultaneously select subsets of genes and samples, so that the genes in the same subset (bicluster) have similar expression to each other within the samples of that bicluster. It is typically expected that each bicluster reflects a biological process and that makes the rationale of *MLC* appear similar to a biclustering approach. Although both approaches

make use of sample selection and aim at discovering genes involved in the same biological processes, they are fundamentally different in the sense that *MLC* is supervised and biclustering unsupervised. Biclustering does not make use of GO annotations, but only of the expression matrix. Often, observing enrichment of certain GO terms or KEGG pathways in the genes of biclusters is one of the ways to validate a biclustering result [37]. On the other hand, *MLC* starts with a set of genes whose GO annotations are known (or at least partly known) and tries to use the expression matrix in order to identify which of the remaining genes participate in a particular biological process by defining a co-expression measure specific to that process.

4.4.3. POSSIBLE EXTENSIONS

MLC learns the sample weights automatically from the available data and does not rely on information about the samples' biological condition or tissue. As curation efforts increase and the amount of well-annotated data in public databases grows larger with time, in the future it might be useful to extend *MLC* to incorporate such knowledge. A possible way to do that would be a group LASSO approach [38]. Group LASSO uses predefined groups of samples and forces the weights of all samples in a group to be equal. Each such group could contain technical and biological replicates, samples from the same tissue or samples from similar knockout experiments and perturbations.

A disadvantage of *MLC* is the fact that it does not account for the possibility that genes that show exactly opposite expression patterns (i.e. genes with large negative correlation) might also be involved in the same biological process. In fact, negative correlations are penalized, as our model explicitly tries to force the signed similarities of *p*-*p* pairs to be larger than those of the *p*-*n* pairs. In Figure 4.2b, we see that large negative *PCC* values are scarce within our dataset, implying that we might not suffer a lot from ignoring negative similarities, at least when considering the *PCC* as a method. The effect might be larger for our *MLC* approach though, which selects a subset of the samples, as in this smaller set negative correlations might be more frequent.

To handle this short-coming, one could directly use the absolute value of the weighted co-expression in the model. Doing this, one is faced with an additional challenge, namely that the absolute value is not differentiable at 0. This can be overcome by approximating the absolute value with a smooth function, such as $\sqrt{x^2+\epsilon}$, where ϵ is a small positive number [39], but this makes the co-expression function non-linear and the calculation of its derivative with respect to $\mathbf{w}$ more costly. More importantly, it makes the optimization problem more difficult, as it adds an extra non-linearity to an already non-convex objective function, meaning that it might be harder to find a good solution for the weights in this problem.

One could also think of alternative formulations of the objective function that would accommodate absolute co-expression values more easily. For instance, it would be possible to minimize the squared difference between the weighted absolute correlations and a target value (e.g. 0 for *p*-*n* pairs and 1 for *p*-*p* pairs). Another possibility would be to use a triplet loss, which has been successfully used in image retrieval [40]. In the triplet loss, we look at sets of three genes at a time instead of two: two positive genes (p_1, p_2) and one negative (n_1). Then, we maximize the difference $S_w(p_1,p_2) - S_w(p_1,n_1)$, where S_w is in this case the absolute weighted similarity.

Finally, in this work, we applied *MLC* on finding candidate genes for GO terms from the BPO. However, it can be useful for any gene annotation problem that can be solved with expression data, such as finding members of KEGG pathways or genes that are likely to influence a given phenotypic trait. As *MLC* is computationally efficient, it can easily be applied to a large number of different terms/phenotypes, offering state-of-the-art performance with the added benefit of allowing users to understand which parts of the dataset influence the predictions.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their valuable suggestions.

FUNDING

This work has been supported by Keygene N.V., a crop innovation company in the Netherlands.

REFERENCES

[1] S. Makrodimitris, M. J. T. Reinders, and R. C. H. J. van Ham, *Metric learning on expression data for gene function prediction,* Bioinformatics **36**, 1182 (2019), https://academic.oup.com/bioinformatics/article-pdf/36/4/1182/32527599/btz731.pdf .

[2] M. Ashburner *et al.*, *Gene Ontology: tool for the unification of biology,* Nature Genetics **25**, 25 (2000), arXiv:10614036 .

[3] Y. Jiang *et al.*, *An expanded evaluation of protein function prediction methods shows an improvement in accuracy,* Genome biology **17**, 184 (2016), arXiv:1601.00891 .

[4] N. Zhou *et al.*, *The cafa challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens,* bioRxiv (2019), 10.1101/653105, https://www.biorxiv.org/content/early/2019/05/29/653105.full.pdf .

[5] L. Lan, N. Djuric, Y. Guo, and S. Vucetic, *MS-kNN: protein function prediction by integrating multiple data sources.* BMC bioinformatics **14 Suppl 3**, S8 (2013).

[6] D. Cozzetto, D. W. Buchan, K. Bryson, and D. T. Jones, *Protein function prediction by massive integration of evolutionary analyses and multiple data sources,* BMC Bioinformatics **14**, S1 (2013).

[7] L. Zhang, S. K. Shah, and I. A. Kakadiaris, *Hierarchical Multi-label Classification using Fully Associative Ensemble Learning,* Pattern Recognition **70**, 89 (2017).

[8] H. Parkinson, M. Kapushesky, M. Shojatalab, N. Abeygunawardena, R. Coulson, A. Farne, E. Holloway, N. Kolesnykov, P. Lilja, M. Lukk, R. Mani, T. Rayner, A. Sharma, E. William, U. Sarkans, and A. Brazma, *ArrayExpress - A public database of microarray experiments and gene expression profiles,* Nucleic Acids Research **35** (2007), 10.1093/nar/gkl995.

[9] E. Clough and T. Barrett, *The Gene Expression Omnibus database,* in *Methods in Molecular Biology*, Vol. 1418 (2016) pp. 93–110.

[10] Z. Jiang, X. Dong, Z. G. Li, F. He, and Z. Zhang, *Differential coexpression analysis reveals extensive rewiring of arabidopsis gene coexpression in response to pseudomonas syringae infection,* Scientific Reports **6** (2016), 10.1038/srep35064.

[11] P. Jaskowiak, R. Campello, and I. Costa, *Evaluating Correlation Coefficients for Clustering Gene Expression Profiles of Cancer,* in *Advances in Bioinformatics and Computational Biology*, Vol. 7409 (2012) pp. 120–131.

[12] L.-Y. Hu, M.-W. Huang, S.-W. Ke, and C.-F. Tsai, *The distance function effect on k-nearest neighbor classification for medical datasets,* SpringerPlus **5**, 1304 (2016).

[13] A. Bellet, A. Habrard, and M. Sebban, *A Survey on Metric Learning for Feature Vectors and Structured Data,* arxiv **1306.6709** (2013), arXiv:1306.6709 .

[14] S. S. Ray and S. Misra, *Genetic algorithm for assigning weights to gene expressions using functional annotations,* Computers in Biology and Medicine (2019), 10.1016/j.compbiomed.2018.11.011.

[15] Y. Xu, H. Min, Q. Wu, H. Song, and B. Ye, *Multi-Instance Metric Transfer Learning for Genome-Wide Protein Function Prediction,* Scientific Reports **7** (2017), 10.1038/srep41831.

[16] R. Tibshirani, *Regression Selection and Shrinkage via the Lasso,* (1996), arXiv:11/73273 [1369–7412] .

[17] R. Petryszak, N. A. Fonseca, A. Füllgrabe, L. Huerta, M. Keays, Y. Amy Tang, and A. Brazma, *The RNASeq-er API - a gateway to systematically updated analysis of public RNA-Seq data,* Bioinformatics , 1 (2017), arXiv:044925 .

[18] T. Obayashi, Y. Aoki, S. Tadaka, Y. Kagaya, and K. Kinoshita, *ATTED-II in 2018: A Plant Coexpression Database Based on Investigation of the Statistical Property of the Mutual Rank Index,* Plant & cell physiology **59**, e3 (2018).

[19] W. E. Johnson, C. Li, and A. Rabinovic, *Adjusting batch effects in microarray expression data using empirical Bayes methods,* Biostatistics **8**, 118 (2007).

[20] J. Tan, G. Doing, K. A. Lewis, C. E. Price, K. M. Chen, K. C. Cady, B. Perchuk, M. T. Laub, D. A. Hogan, and C. S. Greene, *Unsupervised Extraction of Stable Expression Signatures from Public Compendia with an Ensemble of Neural Networks,* Cell Syst **5**, 63 (2017).

[21] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A Limited Memory Algorithm for Bound Constrained Optimization,* SIAM Journal on Scientific Computing **16**, 1190 (1995), arXiv:arXiv:1011.1669v3 .

[22] S. Varma and R. Simon, *Bias in error estimation when using cross-validation for model selection,* BMC Bioinformatics **7**, 91 (2006).

[23] W. T. Clark and P. Radivojac, *Information-theoretic evaluation of predicted ontological annotations,* Bioinformatics **29**, i53 (2013).

[24] P. Resnik, *Using Information Content to Evaluate Semantic Similarity in a Taxonomy,* roceedings of the 14th international joint conference on Artificial intelligence - Volume 1 - IJCAI'95 **1**, 6 (1995), arXiv:9511007 [cmp-lg] .

[25] M. Kavanová, F. A. Lattanzi, A. A. Grimoldi, and H. Schnyder, *Phosphorus deficiency decreases cell division and elongation in grass leaves,* Plant Physiology **141**, 766 (2006), http://www.plantphysiol.org/content/141/2/766.full.pdf .

[26] J. M. Vázquez-Ramos and M. de la Paz Sánchez, *The cell cycle and seed germination,* Seed Science Research **13**, 113 (2003).

[27] B. Xu, Z. Li, Y. Zhu, H. Wang, H. Ma, A. Dong, and H. Huang, *Arabidopsis genes AS1, AS2, and JAG negatively regulate boundary-specifying genes to promote sepal and petal development,* Plant Physiol. **146**, 566 (2008).

[28] R. Muller, A. Bleckmann, and R. Simon, *The receptor kinase CORYNE of Arabidopsis transmits the stem cell-limiting signal CLAVATA3 independently of CLAVATA1,* Plant Cell **20**, 934 (2008).

[29] H. Vaucheret, F. Vazquez, P. Crété, and D. P. Bartel, *The action of argonaute1 in the mirna pathway and its regulation by the mirna pathway are crucial for plant development,* Genes & development **18**, 1187 (2004).

[30] D. Chen, W. Yan, L.-Y. Fu, and K. Kaufmann, *Architecture of gene regulatory networks controlling flower development in Arabidopsis thaliana,* Nature Communications **9**, 4534 (2018).

[31] E. A. Tanaka, S. R. Nozawa, A. A. Macedo, and J. A. Baranauskas, *A multi-label approach using binary relevance and decision trees applied to functional genomics,* Journal of Biomedical Informatics **54**, 85 (2015).

[32] E. Suzuki, M. Gotoh, and Y. Choki, *Bloomy decision tree for multi-objective classification,* in *Principles of Data Mining and Knowledge Discovery*, edited by L. De Raedt and A. Siebes (Springer Berlin Heidelberg, Berlin, Heidelberg, 2001) pp. 436–447.

[33] Y.-K. Li and M.-L. Zhang, *Enhancing binary relevance for multi-label learning with controlled label correlations exploitation,* in *PRICAI 2014: Trends in Artificial Intelligence*, edited by D.-N. Pham and S.-B. Park (Springer International Publishing, Cham, 2014) pp. 91–103.

[34] S. Theodoridis and K. Koutroumbas, *Pattern Recognition* (Academic Press, 2008).

[35] R. A. Fisher, *The Use of Multiple Measurements in Taxonomic Problems,* Annals of Eugenics **7**, 179 (1936), arXiv:arXiv:1011.1669v3 .

[36] D. L. Davies and D. W. Bouldin, *A Cluster Separation Measure,* IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-1**, 224 (1979), arXiv:arXiv:1011.1669v3 .

[37] R. Santamaría, L. Quintales, and R. Therón, *Methods to bicluster validation and comparison in microarray data,* in *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, edited by H. Yin, P. Tino, E. Corchado, W. Byrne, and X. Yao (Springer Berlin Heidelberg, Berlin, Heidelberg, 2007) pp. 780–789.

[38] M. Yuan and Y. Lin, *Model selection and estimation in regression with grouped variables,* Journal of the Royal Statistical Society. Series B: Statistical Methodology **68**, 49 (2006), arXiv:06/68049 [1369–7412] .

[39] C. Ramírez, R. Sanchez, V. Kreinovich, and M. Argáez, $\sqrt{x^2+\mu}$ *is the most computationally efficient smooth approximation to* $|x|$*: a proof,* (2013).

[40] S. S. Husain, E. Ong, and M. Bober, *ACTNET: end-to-end learning of feature activations and multi-stream aggregation for effective instance image retrieval,* CoRR **abs/1907.05794** (2019), arXiv:1907.05794 .

[41] Y. Aoki, Y. Okamura, S. Tadaka, K. Kinoshita, and T. Obayashi, *ATTED-II in 2016: A plant coexpression database towards lineage-specific coexpression,* Plant and Cell Physiology **57**, e5 (2016).

4.5. SUPPLEMENTARY MATERIAL

4.5.1. *MLC* CAN FILTER OUT LOW-QUALITY SAMPLES

As mentioned in section 2.1 of the main document, as a pre-processing step we removed samples with fewer than 10 million reads mapped and applied ComBat to reduce the technical variation among samples. As a side experiment, we ran *MLC* using all the samples, i.e. bypassing the coverage filter, to test whether *MLC* will preferentially set the weights of these supposedly lower-quality samples to 0.

We ran *MLC* again for all GO terms including the whole double-loop cross-validation scheme, but now using all 4,215 samples. We removed 7 samples that were in "singleton" batches in order to be able to run ComBat (s ection 2.1, main document and Supplementary Material 2), leaving us with 4,208 samples. Then, we ranked the samples based on the weight values they get assigned for each term. The sample with lowest weight gets a rank of 0 and the one with the highest 4,207. If multiple samples (for instance say 1,000) get a weight value of 0, then all these samples get a rank of 0 and the immediately next sample gets a rank of 1,000. Other ties are handled similarly. Then, we used the median of each sample's ranks over all GO terms as a measure of how often it is selected by *MLC* (the higher the rank, the more often a sample is selected).

We found that *MLC* preferentially selected samples from the ones we had initially removed (left panel of Figure 4.S1). We found that samples with about 1 million to 100 million reads have a similar distribution of median ranks, but samples with lower coverage tend to have larger median ranks. More specifically, for 188 out of 226 the terms for which *MLC* selected fewer than 500 samples, we found significant enrichment of the low-coverage samples using Fisher's exact test. Despite this, the performance of MLC was not affected (mean of 0.72), neither that of *MR* (also mean of 0.72). On the contrary, the performance of *PCC* slightly increased, from 0.69 to 0.7, remaining significantly worse than the other two methods.

However, when we removed the ComBat from our pipeline and repeated the whole experiment again, we found that the vast majority of samples with fewer than 1 million reads had a median rank of 0 (right panel of Figure 4.S1). In that case, *MLC* suffered a performance drop, performing equal to *PCC* (mean *ROC AUC* of 0.7). Despite this, *MLC* remained superior on specific terms (Spearman ρ between % improvement and term information content = 0.26).

This shows that – indeed – our method can identify poor-quality samples and avoid selecting them.

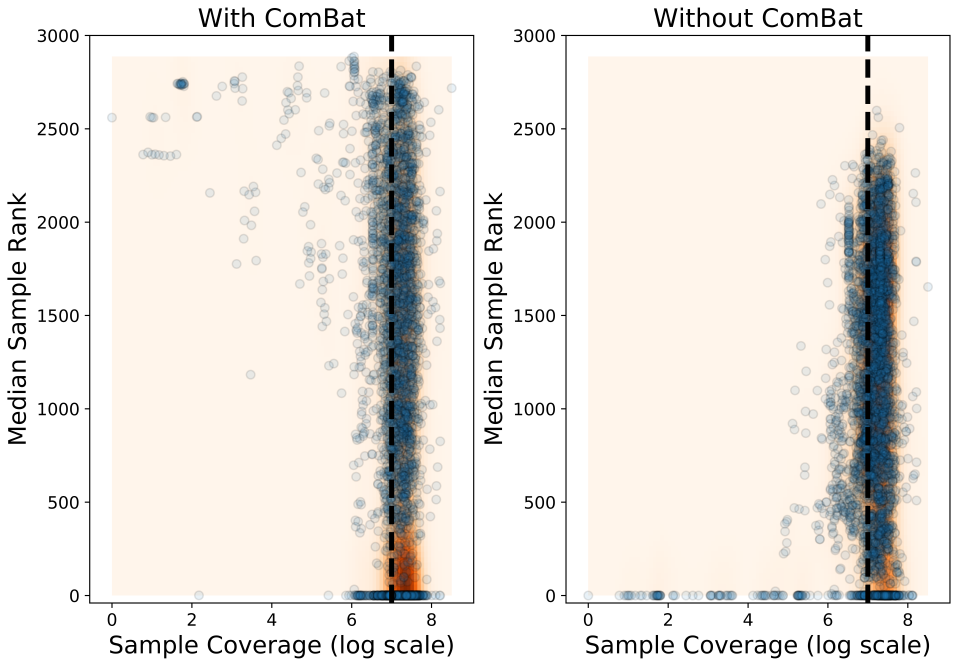


Figure 4.S1: Sample median weight rank (y axis) as a function of the total number of reads (x axis, log_{10} scale) when using ComBat (left) and when not using ComBat (right). The threshold we initially used to filter out samples is shown as black dashed line. Samples are shown as blue dots. For each GO term, we sort the *MLC* weights and convert them to rank values. Then, for each sample, we plot the median of its ranks over all GO terms. The 2-dimensional density of the points as estimated using Gaussian kernels is also shown, with darker areas corresponding to higher density.

4.5.2. USE OF COMBAT WITH BATCHES CONTAINING ONLY 1 SAMPLE

If a batch contains only one sample, the variance of a gene within the samples of the batch is not defined, in which case ComBat only standardizes the means. Since different batches have widely different read counts, samples from batches with higher average coverage will contribute considerably more to the total variance of each gene's expression. This is undesirable, as this variance most probably represents technical and not

biological variation. Therefore, we deemed it necessary to also standardize the variance of each gene within each batch, which meant that we had to remove all studies that had only one sample, leaving us with 2,959 samples.

4.5.3. COMPETING METHODS

MUTUAL RANK (*MR*)

An alternative way to measure co-expression is using the Mutual Rank (*MR*) which has been successfully used in the ATTED-II co-expression database [41]. For every gene i, the co-expression values to all other genes are ranked in descending order. We use $rank_i(j)$ to denote the rank of gene j in terms of similarity to i. Note that $rank_i(i) = 0$. The *MR* value between two genes is calculated as the geometric mean of $rank_i(j)$ and $rank_j(i)$:

$$MR(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{rank_i(j) \times rank_j(i)} \tag{4.7}$$

MR is expected to be more robust to spurious correlations caused by outliers (due to the geometric mean) and can also handle cases where a gene tends to have very high or very low correlations with many other genes, which bias the co-expression ranks leading to poorer performance for the *k*-NN classifier. We downloaded the pairwise *MR* values constructed from public RNA-Seq data for all *A. thaliana* genes from the ATTED-II website (`http://atted.jp/top_download.shtml`).

GAAWGEFA

For *GAAWGEFA* [14] we used the MATLAB code provided by the authors at `http://sampa.droppages.com/GAAWGEFA.html`. Because of the long runtime we did not tune any of the parameters of the algorithm and used the default settings as mentioned in the paper:

- Initial population size: 20
- Crossover Probability: 0.85
- Mutation Probability: 0.1
- Maximum iterations: 1000

After the 150-th iteration we enabled the extra option of early termination, i.e. we stopped the optimization if the fitness function had not increased by at least 1% between two consecutive generations. We did tune the number of nearest neighbors used in the *k*-NN classifier as previously.

4.5.4. EVALUATION MODES

CROSS-VALIDATION EXPERIMENT

We used the *k*-Nearest Neighbors (*k*-NN) classifier to compare the function prediction performance of the different studied co-expression measures on all *A. thaliana* genes with at least one BP annotation. To counter the imbalance in the dataset, we restricted ourselves to GO terms that annotate at least 1% of the genes. Also, for the weight optimization stage of *MLC*, we randomly sampled an equal number of genes with and without each term. The optimal number of nearest neighbors (k) is a parameter of all methods. *MLC* has an extra regularization parameter α (equation 6, main document). We

tuned the parameters of *MLC* independently for each GO term, while we selected the value of k that maximized the mean performance over all tested GO terms for the other methods. Parameter tuning was done in a double 3-fold cross-validation loop [22] (Figure 4.S2) using the term-centric $ROCAUC$ as performance criterion. The inner loop was used to select the optimal parameter values and the outer loop to evaluate the performance of the tuned models on previously unseen genes.

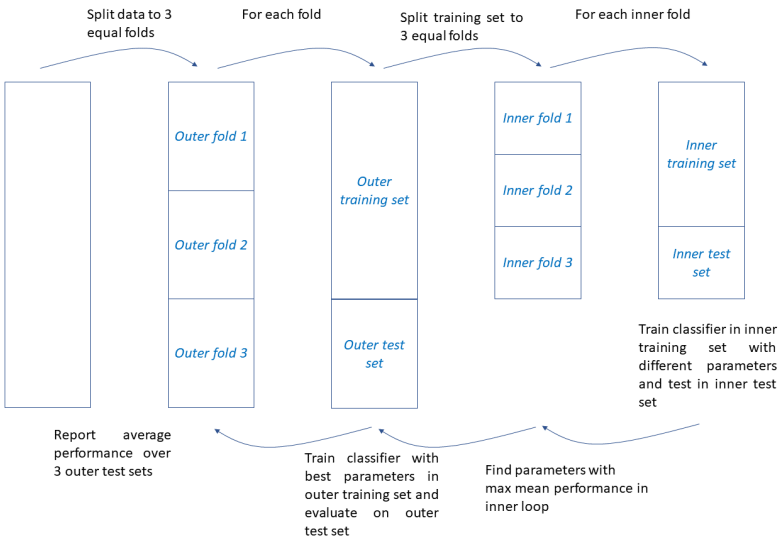


Figure 4.S2: Schematic representation of a nested cross-validation loop.

CAFA3 EXPERIMENT

We also evaluated the same methods on the preliminary test set from CAFA3, released by the organizers in June 2017. This dataset contains 6,077 training and 137 test genes from *A. thaliana*. After ID mapping, we restricted ourselves only to the genes for which we had both expression data from ArrayExpress and *MR* values from the ATTED database and passed the filtering step described in section 2.1 (main document). This left us with 4,889 training genes and 90 test genes, annotated with 707 GO terms. We used the training set to tune the parameters of the tested methods using a 3-fold cross-validation loop. We removed the 10 rarest terms, as there were not enough training and testing genes in all folds. Then, we re-trained each method on the whole training set, using the optimal parameter values found with cross-validation and made predictions for the 697 remaining terms on the 90 test genes (reported as CAFA3 results). To assess the variability of the results, we performed 1,000 bootstraps, choosing at random with replacement 90 genes at each iteration and re-evaluating the mean term-centric performance. We used these bootstraps to construct 95% confidence intervals.

CAFA-π

Finally, we compared *MLC* to *MR* and *PCC* using data from the CAFA-π challenge [4]. The goal of CAFA-π was to make term-specific predictions for biofilm formation and

motility in *Pseudomonas aeruginosa* and *Candida albicans*, which were then tested using genome-wide assays. For pseudomonas, the organizers of the challenge used two baseline methods that made use of an existing microarray dataset with 1,051 samples and *PCC* [20]. These baseline methods turned out to be the top-performing methods in the challenge. We compared our method and *MR* to these *PCC*-based baselines. *GAAWGEFA*, uses all available GO terms and is trained in a protein-centric way, so we did not test *GAAWGEFA* in this strictly term-centric setting.

As starting point, all methods use the genes that were already experimentally annotated with the function of interest (biofilm or motility) before the challenge (training genes). Note that these training genes are only of the positive class, however some of them were not validated as positives by the assay. Nevertheless, we still used all these genes as positive genes, to ascertain a fair comparison with methods implemented before the challenge. For the baseline methods, the posterior probability of a test gene being involved in the function was calculated as either the maximum *PCC* value between that gene and all training genes, or the average value of the top-10 most co-expressed training genes [4]. We applied the same strategy for *MR* and *MLC*. *MLC* requires a negative set during training, i.e. a set of genes known not be involved in the function of interest. In this case, we chose the negative set as a random subset of the genes for which expression data were available, but they were not screened for the function, so their involvement in the function is unknown both before and after the CAFA-π challenge. For example, these could be essential genes whose knock-out is lethal. *MLC* has a parameter α which we previously tuned using cross-validation. This is not possible in this setting due to the small number of training genes, so we further tested MLC with two values of α: one that performs sample selection (0.75, selecting fewer than 10% of the samples) and one that does not (0.25, selecting all samples).

4.5.5. EVALUATION MEASURES

UNWEIGHTED AND WEIGHTED TERM-CENTRIC ROC AUC

In the setting used in this work, we are interested in finding the genes that perform a given function, so term-centric evaluation is the most interesting. As in the CAFA papers, we use the term-centric area under the ROC curve ($ROCAUC$). For every GO term l, we compute a $ROCAUC$ score for the binary problem of finding all test genes that are annotated with that term and then we report the average of that value over all GO terms. We also use the weighted version of this measure ($ROCAUC_w$), in which we calculate a weighted average, weighing each term by its Resnik Information Content ($IC(l)$) [24].

PROTEIN-CENTRIC MEASURES

For protein-centric evaluation, we use the F1 score (F_{max}) and the Semantic Distance (S_{min}) [23]. Similar to the CAFA evaluation, we start with the posterior probabilities for each pair of test gene and GO term. As these measures require hard predictions (i.e. either 0 or 1, not a posterior probability) we compute them for 21 equally-spaced thresholds from 0 to 1 (i.e. in steps of 0.05). We then only report the metric value at the optimal threshold for each measure (the maximum for the F1 and the minimum for the Semantic Distance) and for each evaluated algorithm [3].

4.5.6. PROTEIN-CENTRIC RESULTS

We compared 5 methods: *PCC*, *PCC+MR*, *GAAWGEFA*, *MLC* and MLC_G using 2 term-centric and 2 protein-centric evaluation measures. Term-centric performance is shown in Table 1 in the main document. Tables 4.S1 and 4.S2 show the protein-centric F_{max} and S_{min} for these methods on the cross-validation and CAFA3 datasets respectively. We observe that, according to both metrics, all methods achieve more or less equivalent protein-centric performance, with the exception of the *PCC*, which performs significantly worse (Supplementary Material 4.5.7). The *MR* seems to be consistently the top method, but the differences are very small and not statistically significant (See also Supplementary Material 4.5.7).

Table 4.S1: Comparison of the protein-centric F_{max} and S_{min} of the tested methods using 3-fold cross-validation. For each method and metric the mean and standard error are shown. Higher F_{max} and **lower** S_{min} denote better performance.

Method	F_{max}	S_{min}
PCC	0.34 ± 0.001	19.12 ± 0.11
PCC + MR	0.36 ± 0.001	18.85 ± 0.11
GAAWGEFA	0.35 ± 0.002	18.97 ± 0.10
MLC (S_w)	0.35 ± 0.001	18.89 ± 0.13
MLC_G (S_w)	0.35 ± 0.003	19.00 ± 0.17

Table 4.S2: Comparison of the protein-centric F_{max} and S_{min} of the tested methods using the CAFA3 data. For each method and metric the mean and 95% confidence interval is shown. Higher F_{max} and **lower** S_{min} denote better performance.

Method	F_{max}	S_{min}
PCC	0.25 [0.22, 0.29]	21.27 [18.78, 29.17]
PCC + MR	0.27 [0.24, 0.30]	21.18 [18.78, 28.99]
GAAWGEFA	0.25 [0.22, 0.29]	21.32 [18.90, 29.10]
MLC (S_w)	0.26 [0.23, 0.29]	21.56 [18.87, 29.11]
MLC_G (S_w)	0.27 [0.24, 0.31]	21.27 [18.81, 29.22]

4.5.7. STATISTICAL SIGNIFICANCE OF DIFFERENCES IN CROSS-VALIDATION PERFORMANCE

We use four evaluation metrics ($ROCAUC$, $ROCAUC_w$, F_{max}, S_{min}) to compare 5 methods (*PCC*, *PCC + MR*, *GAAWGEFA*, *MLC* and MLC_G). We used the paired-sample t-test to compare all 10 different pairs of methods across the three cross-validation folds. This test tests the null hypothesis that the mean difference in performance between two approaches across the three folds is not different from zero. For every combination of two methods and a metric we obtained a p-value, so in total we obtained 40 p-values. We corrected all these p-values jointly for multiple testing using the Benjamini-Hochberg method for controlling the False Discovery Rate (FDR). The results are listed in tables 4.S3 to 4.S6. *MR* and *MLC* are significantly better than *PCC* according to all four metrics.

Table 4.S3: FDR-corrected p-values for the null hypothesis that the cross-validation $ROCAUC$ of two methods is not different from zero. Entries are colored in red if the row method is significantly worse than the column method (FDR < 0.05) and in green if the row method is significantly better than the column method (FDR < 0.05).

	PCC + MR	*GAAWGEFA*	*MLC* (S_w)	MLC_G (S_w)
PCC	0.008	0.038	0.021	0.041
PCC + MR		0.088	0.272	0.180
GAAWGEFA			0.066	0.175
MLC (S_w)				0.372

Table 4.S4: FDR-corrected p-values for the null hypothesis that the cross-validation $ROCAUC_w$ of two methods is not different from zero. Entries are colored in red if the row method is significantly worse than the column method (FDR < 0.05) and in green if the row method is significantly better than the column method (FDR < 0.05).

	PCC + MR	*GAAWGEFA*	*MLC* (S_w)	MLC_G (S_w)
PCC	0.008	0.038	0.016	0.041
PCC + MR		0.088	0.229	0.189
GAAWGEFA			0.041	0.178
MLC (S_w)				0.155

Table 4.S5: FDR-corrected p-values for the null hypothesis that the cross-validation F_{max} of two methods is not different from zero. Entries are colored in red if the row method is significantly worse than the column method (FDR < 0.05) and in green if the row method is significantly better than the column method (FDR < 0.05).

	PCC + MR	*GAAWGEFA*	*MLC* (S_w)	MLC_G (S_w)
PCC	0.038	0.113	0.038	0.155
PCC + MR		0.154	0.302	0.047
GAAWGEFA			0.258	0.180
MLC (S_w)				0.180

Table 4.S6: FDR-corrected p-values for the null hypothesis that the cross-validation S_{min} of two methods is not different from zero. Entries are colored in red if the row method is significantly worse than the column method (FDR < 0.05) and in green if the row method is significantly better than the column method (FDR < 0.05).

	PCC + MR	*GAAWGEFA*	*MLC* (S_w)	MLC_G (S_w)
PCC	0.021	0.066	0.038	0.155
PCC + MR		0.119	0.223	0.138
GAAWGEFA			0.246	0.181
MLC (S_w)				0.229

4.5.8. *MLC* DOES NOT PICK UP ARTIFICIAL INFORMATION

We started from the label matrix $Y \in \{0,1\}^{N \times L}$, where L is the total number of GO terms. Column l of Y represents to the vector $\mathbf{y}(l)$ that contains a 1 at the rows that correspond to the genes annotated with l. We randomly permuted the rows of the label matrix, so that each gene is assigned to a random set of GO terms, but both the consistency of the ontology graph and the GO term frequencies are preserved. Then we ran both *PCC* and term-specific *MLC* on this random dataset, including the tuning of the parameters using a double-loop cross-validation as described in Supplementary Material 4. We found that both *PCC* and *MLC* achieved a mean term-centric *ROC AUC* of 0.5 (i.e. equal to random guessing). This shows that in absence of real structure in the data, sample selection or re-weighing does not artificially generate information (i.e. false positives are controlled).

4

4.5.9. FALSE POSITIVE RATES FOR GENERAL AND SPECIFIC TERMS

To explain why *MLC* performs better for specific terms than for general ones, we compared the ROC curves of the 20% most specific terms to those of the 20% most general ones. Term specificity was measured by the Resnik Information Content [24]. To get an indication of the general behaviour pattern of *MLC*, we averaged all ROC curves in each of the two groups of GO terms. Figure 4.S3 shows these two average curves. We observed that near the point $(0,0)$, which corresponds to the genes that *MLC* classified as positive with high posterior probability, the average ROC curve of the specific terms is increasing a lot more sharply than the one of the general terms which is smoother. This means that genes for which *MLC* is most confident that are positive are indeed enriched with positive labels for the specific terms. The curve of the general terms is increasing more smoothly, meaning that for those terms, *MLC* is scoring a lot of negative genes highly, resulting in more false positive predictions.

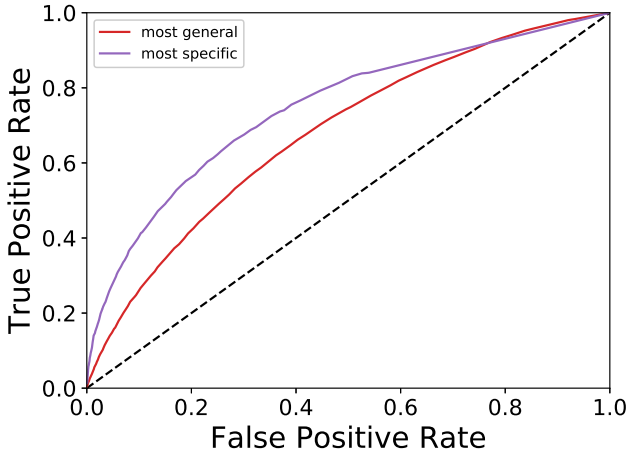


Figure 4.S3: Term-centric ROC curves averaged over the terms with the 20% highest IC (purple) and with the 20% lowest IC (red). The $y = x$ line which represents the expected ROC of a random classifier is shown with a black dashed line.

4.5.10. PERFORMANCE AS A FUNCTION OF TERM SPECIFICITY

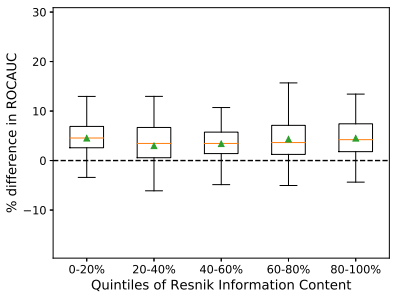
For each pair of methods, we calculate the percent difference in *ROC AUC* between them for each GO term. Tables 4.S7 and 4.S8 show the Spearman correlation of these values with the Information Content and the maximum path length to the ontology root of each term respectively. Moreover, in Figures 4.S4 and 4.S5 we plot these differences for each pair. From these we can clearly conclude that term-specific *MLC* is the best of all tested methods at predicting specific terms.

Table 4.S7: Spearman correlation of the % difference in performance between the method in each row and the one each column with term Resnik Information Content. Statistically significant values (FDR < 0.05) are shown in bold.

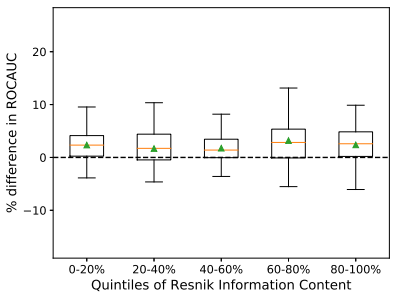
	PCC	*PCC + MR*	*GAAWGEFA*	*MLC*	MLC_G
PCC	-	-0.028	-0.041	**-0.174**	-0.055
PCC + MR	0.028	-	0.016	**-0.165**	0.029
GAAWGEFA	0.041	-0.016	-	**-0.163**	0.003
MLC	**0.174**	**0.165**	**0.163**	-	**0.164**
MLC_G	0.055	-0.029	-0.003	**-0.164**	-

Table 4.S8: Spearman correlation of the % difference in performance between the method in each row and the one each column with term path length to the ontology root. Statistically significant values (FDR < 0.05) are shown in bold.

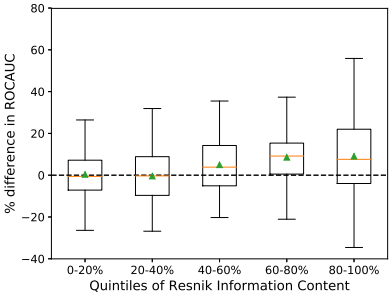
	PCC	*PCC + MR*	*GAAWGEFA*	*MLC*	MLC_G
PCC	-	-0.007	-0.035	**-0.259**	-0.035
PCC + MR	0.007	-	0.022	**-0.246**	0.015
GAAWGEFA	0.035	-0.022	-	**-0.244**	-0.018
MLC	**0.259**	**0.246**	**0.244**	-	**0.255**
MLC_G	0.035	-0.015	0.018	**-0.255**	-



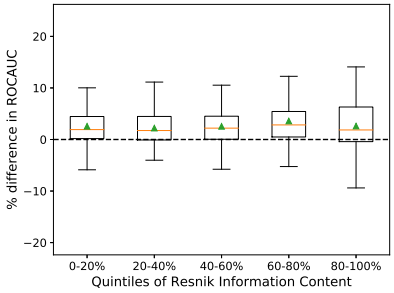
(a) *MR - PCC*



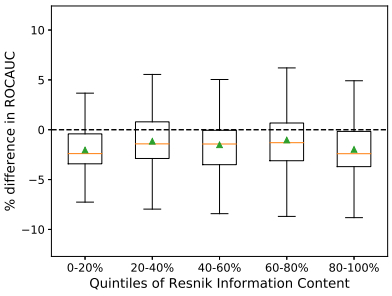
(b) *GAAWGEFA - PCC*



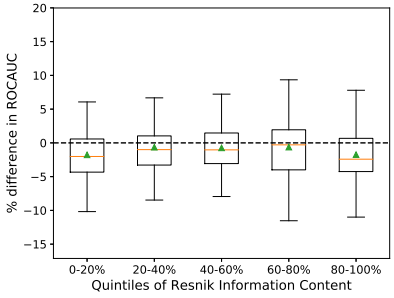
(c) *MLC - PCC*



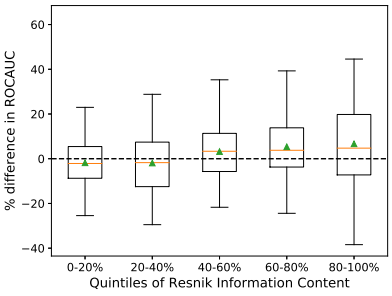
(d) MLC_G - *PCC*



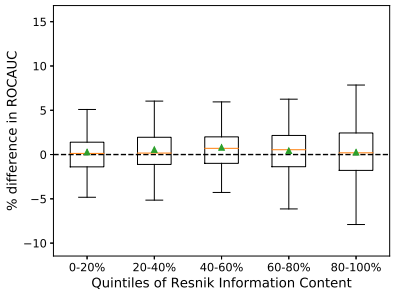
(e) *GAAWGEFA - MR*



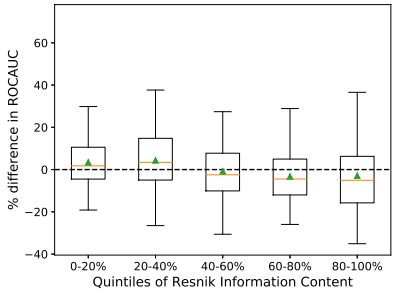
(f) MLC_G - *MR*



(g) *MLC - GAAWGEFA*

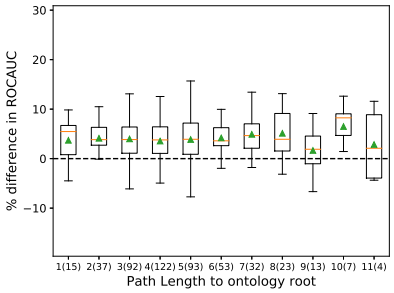


(h) MLC_G - *GAAWGEFA*

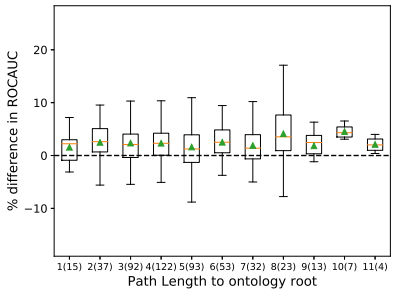


(i) MLC_G - *MLC*

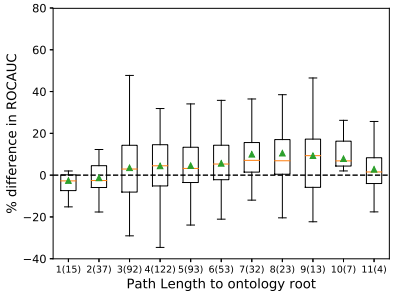
Figure 4.S4: Percent difference in *ROCAUC* of all pairs of methods as a function of Resnik Information Content. For each set of terms in each quintile of Information Content, the corresponding box includes the two middle quartiles of the percent difference for these terms. An orange line denotes the median and a green triangle the mean. The error bars extend to 1.5 times the range of the two middle quartiles.



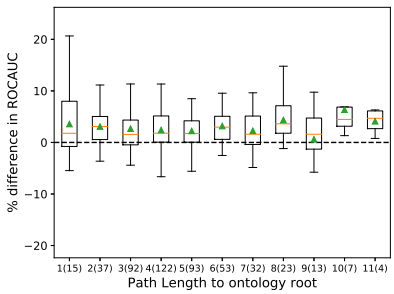
(a) *MR* - *PCC*



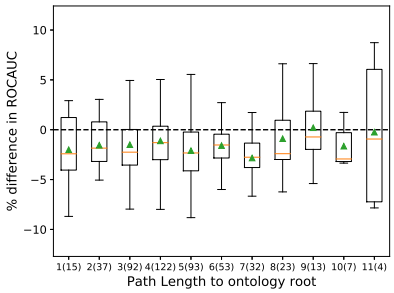
(b) *GAAWGEFA* - *PCC*



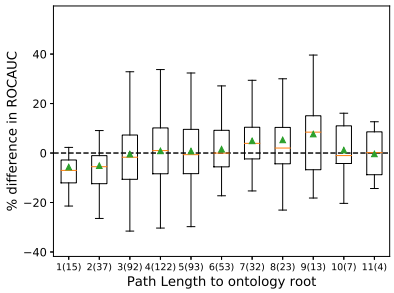
(c) *MLC* - *PCC*



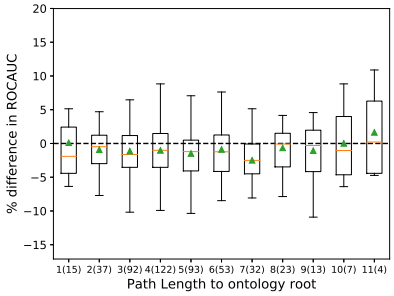
(d) MLC_G - *PCC*



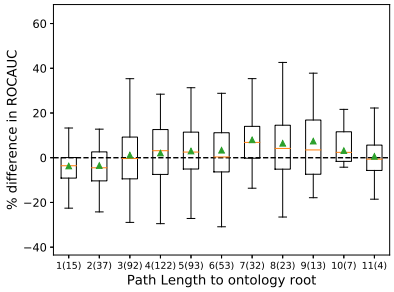
(e) *GAAWGEFA - MR*



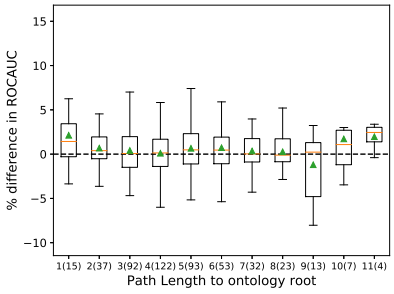
(f) *MLC - MR*



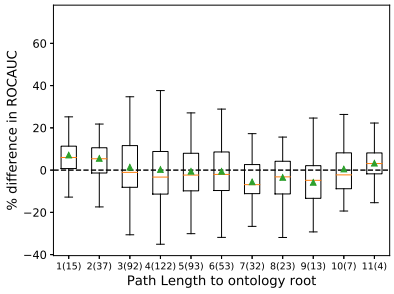
(g) MLC_G *- MR*



(h) *MLC - GAAWGEFA*



(i) MLC_G *- GAAWGEFA*



(j) MLC_G *- MLC*

Figure 4.S5: Percent difference in *ROC AUC* of all pairs of methods as a function of path length to the ontology root. For each set of terms with a given path length, the corresponding box includes the two middle quartiles of the percent difference for these terms. An orange line denotes the median and a green triangle the mean. The error bars extend to 1.5 times the range of the two middle quartiles. The numbers in parentheses next to each path length level denote the number of terms in the dataset at that level.

4.5.11. ONLY REDUCING THE NUMBER OF SAMPLES DOES NOT BOOST *PCC* PERFORMANCE

We tested the performance of the *PCC* on a randomly selected subset of 250 samples out of the original 2,959 samples. We repeated this experiment with 5 different subsets of the same size and in all cases we found the mean $ROCAUC$ to be similar to that of *PCC* with all samples (0.69 ± 0.01). This shows that simply reducing the number of samples is not enough to get a performance boost and that we really need to identify the relevant samples for each GO term.

4.5.12. COMPARISON OF *GAAWGEFA* AND *MLC* WEIGHTS

In Figure 4.S6a we show the distribution of the weight values of the "average" weight profile of *MLC*. To obtain that, we sort all term-specific weight profiles from the smallest to the largest value and then calculate the average weight value at each rank over all term profiles. Figure 4.S6b shows the weight distribution of the *GAAWGEFA*, which is not term-specific. We calculated the *PCC* between each term-specific profile and the *GAAWGEFA* profile and also between each term-specific profile and the MLC_G profile. Figure 4.S7 shows the distribution of these two similarities. In conclusion, the weights learned by *GAAWGEFA* were not correlated to the ones learned by *MLC* or MLC_G.

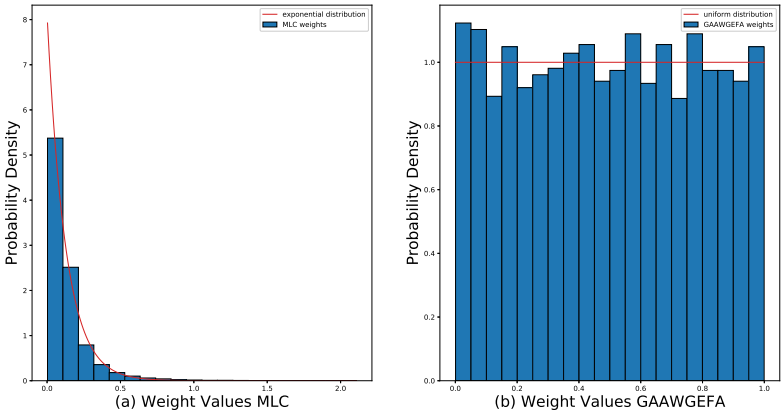


Figure 4.S6: Histogram (y-axis) of the sample weight values (x-axis) for the average *MLC* profile (a) and the GAAWGEFA profile (b). The best fitting exponential distribution (a) and uniform distribution (b) are shown in red.

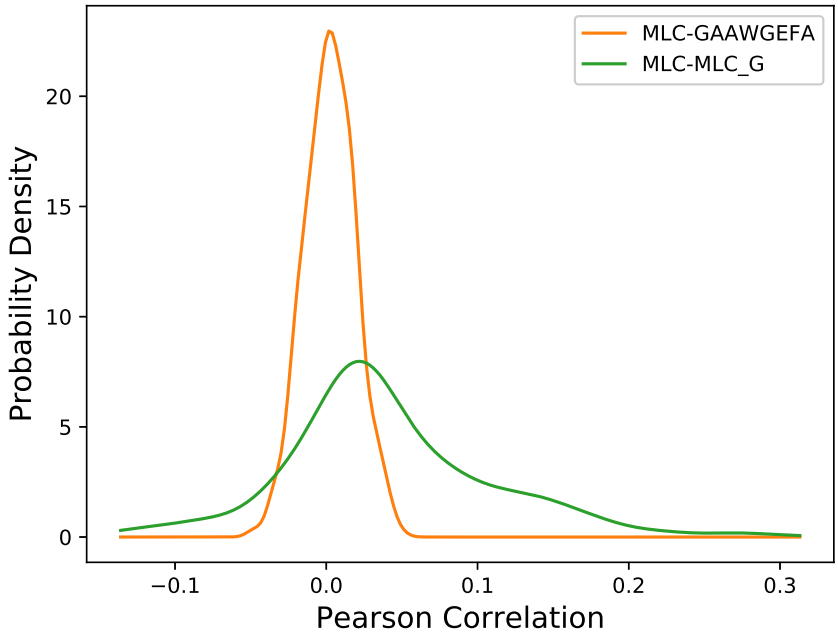


Figure 4.S7: Distribution of the pairwise Pearson correlation values between the *MLC*-derived GO-term-specific weight profiles and the weight profile of MLC_G (green) and *GAAWGEFA* (orange). The *x* axis corresponds to the correlation values and the *y* axis to the probability density.

4.5.13. SAMPLES FROM THE SAME STUDY TEND TO GET EITHER SELECTED OR NOT SELECTED TOGETHER

We tested whether samples from a relevant batch are more often selected together than expected. The null hypothesis in this case is that which batch a sample comes from is independent of whether it gets selected for a particular term. Under the null hypothesis, the number of samples that get selected from each batch are proportional to the total number of samples in the batch. This can be modelled using a multinomial distribution: Each batch is viewed as a categorical variable and under the null hypothesis the probability of observing a sample from that batch is equal to the number of samples in that batch divided by the total number of samples. In each cross-validation fold, we looked at the GO terms for which *MLC* selected fewer than 1,000 samples. For each of those terms, we performed the multinomial test, testing whether more or fewer samples were selected from the same batches than expected under the null hypothesis. To save computation time, we did not perform an exact test, but rather two approximations, one using the chi-squared test and one using Monte Carlo sampling (drawing 1 million samples per test), both implemented in the R package XNomial (https://cran.r-project.org/package=XNomial). With both approximations, we found that for at least 97% of the terms, the observed batch frequencies were signifi-

cantly different from what would be expected by the global batch frequencies with a False Discovery Rate of 0.05 (Table 4.S9). This shows that batches indeed tend to be either enriched or depleted with selected samples for each GO term.

Table 4.S9: Results of two approximations of the multinomial test with null hypothesis that the sample selection is independent of which batches samples come from. For each cross-validation fold, we show the number and fraction (in parentheses) of the terms for which the null hypothesis was rejected with a False Discovery Rate of 0.05.

Fold	Terms tested	Terms significant by Monte Carlo	Terms significant by χ^2
0	176	175 (99.4%)	171 (97.2%)
1	187	186 (99.5%)	183 (97.9%)
2	164	163 (99.4%)	163 (99.4%)

4.5.14. EVALUATION ON SIMULATED DATA

We simulated an expression dataset with 7,000 genes and 3,000 samples, which is similar to the size of our real *A. thaliana* dataset. 15% of these genes were annotated with a GO term of interest (target class). The remaining 85% were evenly split between two other GO terms (background classes 1 and 2). The genes of each of the three classes had a distinct expression pattern: For the target class, the genes within this class have an expression pattern that changes as a noisy sine wave for the first x samples. The expression values for the remaining $n-x$ samples are drawn from a Gaussian noise distribution. The background classes had similar expression patterns, but the starting location, frequency and phase of the sine were different for both classes. Note that the number of informative samples, x, is the same for all three classes, but which samples are informative is different for each of them. To model different basal expressions, each gene was given a random mean expression drawn from a Gaussian distribution (N(0, 3)). Examples of the three classes are shown in Figure 4.S8.

We split the dataset into a training (5,000 genes), validation (1,000 genes) and test set (1,000 genes) in a stratified way. We used the validation set to tune the α parameter of *MLC* and then trained on both the training and validation set and tested on the test set. For the *PCC* (which uses all samples) and the ground-truth *PCC* (which uses only samples 1 to x), we also used the training and validation set as training genes. The number of nearest neighbors (k) was set to 30 for all three methods.

We then varied the number of relevant samples (x) from 10 to 500 and repeated the experiment 5 times for each value of x instantiated, where each time we generate a different simulated dataset.

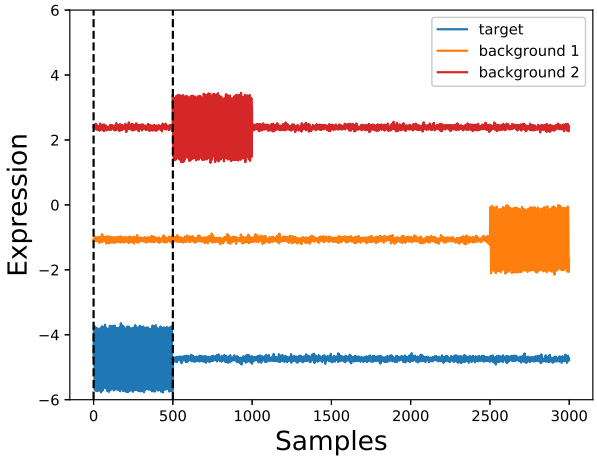


Figure 4.S8: Examples of the generated expression profiles from the three classes, where the relevant number of samples x is equal to 500. The gene expression is on the y axis and the samples on the x axis. The example from the target class is shown in blue and the ones from the other classes in orange and red. Two black dashed lines denote the relevant samples for the target class.

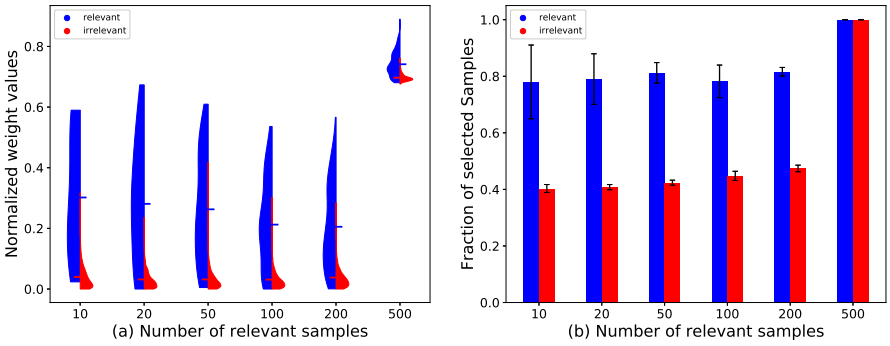


Figure 4.S9: (a) Distribution of weights learned by *MLC* (y axis, scaled in the range [0-1]) for the relevant samples (blue) and the rest of the samples (red) for different numbers of relevant samples (x axis) on simulated data. The mean of each distribution is denoted by a horizontal line. (b) Fraction of the samples that were selected by *MLC* (y axis) for the relevant samples (blue) and the remaining samples (red) for different numbers of relevant samples (x axis). The error bars denote the standard deviation over 5 repetitions.

4.5.15. WEIGHT PROFILE

Figure 4.S10 shows the weight profile that *MLC* learned for term GO:1903047 (mitotic cell cycle process).

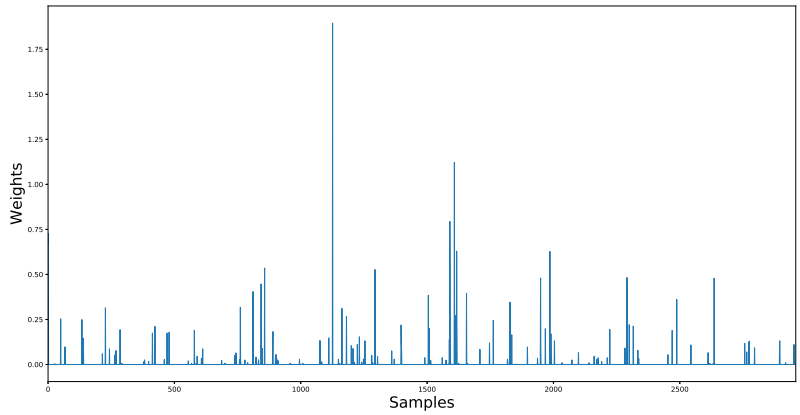


Figure 4.S10: The weight profile learned by *MLC* for term GO:1903047. On the x axis are the 2,959 RNA-Seq samples and on the y axis the weight value of each sample.

4.5.16. THE WEIGHTS LEARNED BY *MLC* ARE CONSISTENT WITH THE ONTOLOGY STRUCTURE AND THE EXISTING ANNOTATIONS

We compared the sample weights learned by *MLC* between parent-child GO term pairs in the following three cases: 1) the parent has only one child term and both parent and child annotate exactly the same genes, 2) the parent has only one child term but annotates more genes than its child, and 3) the parent has exactly two children, meaning that the parent annotates the union of the genes of its children. In the first case, the weight profiles for parent and child are identical (mean Pearson correlation of 1). In the second case, the profiles are similar but not identical (mean Pearson correlation of profiles 0.47). The difference in mean similarity between the two groups is statistically significant (permutation p-value $< 10^{-5}$). Furthermore, the larger the difference in number of extra genes of the parent term, the smaller the profile correlation (Spearman $\rho = -0.53$, $CI_{95\%} = [-0.65, -0.40]$). The profile similarities are even smaller in the third case (mean of 0.36) and significantly smaller than those of case 2 (permutation p-value = 0.0002). This is expected as in this case the parent contains two distinct sets of genes that correspond to two different biological processes. Again, we found a negative correlation between the number of different genes and the profile similarity of pairs (Spearman $\rho = -0.47$, $CI_{95\%} = [-0.61, -0.31]$).

To generalize this finding, we hierarchically clustered the GO terms (complete linkage, Jaccard distance between the gene sets associated with each GO term cutoff of 0.6). The resulting clusters are shown in Figure 4.S11 along with the pairwise distances of the

GO terms. 64 out of 176 clusters contained at least three terms. For each of these 64 clusters, we randomly sampled 10,000 equal-sized sets of GO terms and calculated the mean pairwise similarity in those sets to calculate a permutation p-value. For 62 out of these 64 clusters we found that the pairwise weight-profile similarities of their members (Figure 4.S2, SM2) are significantly higher than random with a False Discovery Rate of 0.05. Also, pairwise profile similarities are positively correlated with the pairwise Resnik semantic similarity of GO terms (Spearman $\rho = 0.16$, $CI_{95\%} = [0.15, 0.17]$). Based on these observations, we conclude that sample weights reflect the gene annotations of each term.

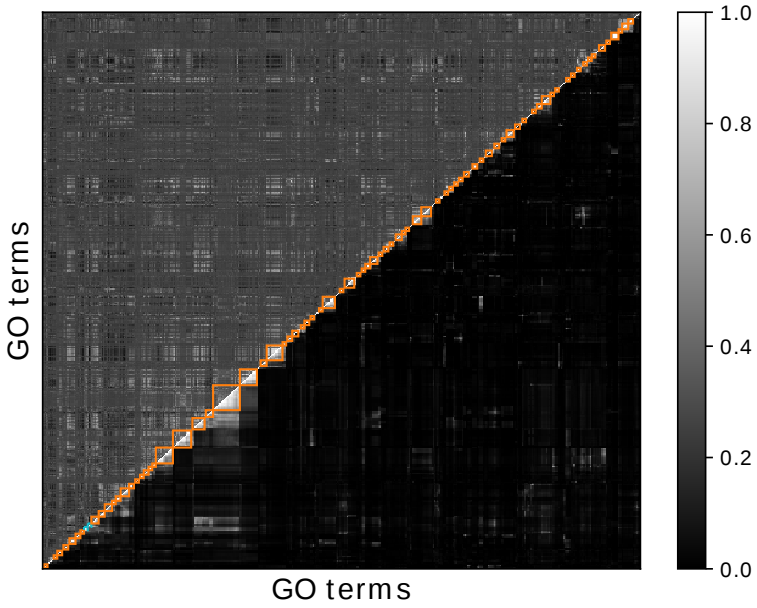


Figure 4.S11: Pairwise Jaccard similarities (below the anti-diagonal) and weight profile similarities (Pearson correlations, above the anti-diagonal) of the tested GO terms. Both the x and y axes show the GO terms ordered so that terms in the same cluster are adjacent. The grey-scale indicates the similarity, with dark being low and bright being high similarity. The profile correlations have been scaled so that they are in the range [0, 1]. The squares highlight the clusters containing at least 3 terms, (cutting the dendrogram at a Jaccard distance threshold of 0.6 when using complete linkage). Light blue boxes indicate the clusters that are not significantly enriched with terms with similar weights and orange-colored clusters are significantly enriched after FDR correction.

4.5.17. CAFA-π RESULTS

PCC, *MR* and *MLC* achieve similar *ROCAUC* (around 0.6 for both biofilm and motility, Table 4.S10)

Again, the samples selected by *MLC* were informative: For motility, the most informative sample was a phhR mutant. phhR is a transcription factor which – under cer-

Table 4.S10: *ROC AUC* achieved by *PCC*, *MR* and *MLC* at the CAFA-π dataset for biofilm formation and motility in *Pseudomonas aeruginosa*. The highest performance per column is shown in bold.

Method	ROC AUC Biofilm	ROC AUC Motility
PCC - top 1	0.56	0.56
PCC - top 10	0.58	**0.60**
MR - top 1	0.57	0.58
MR - top 10	**0.60**	**0.60**
MLC - top 1 ($\alpha = 0.25$)	0.56	0.56
MLC - top 10 ($\alpha = 0.25$)	0.58	**0.60**
MLC - top 1 ($\alpha = 0.75$)	0.57	0.54
MLC - top 10 ($\alpha = 0.75$)	0.59	0.56

tain conditions – regulates the biosynthesis of a molecule called PQS (pseudomonas quinolone signal) [4] which has been shown to repress motility [5]. Among a few wild-type samples, in the top-10 highest scored we found three samples from the same study that studied the effect of Phosphorus on motility [6]. For biofilm formation, 4 of the 10 highest-scored samples were isolates from cystic fibrosis patients, where Pseudomonas is known to produce biofilms [7]. Also samples from a BF8 treatment (inhibits biofilm formation, [8]) and a hydrogen peroxide treatment (promotes biofilm formation in other bacteria,[9]) were highly scored, as well as three sulfur starvation samples from the same study.

Moreover, we had a look at genes that were highly scored by *MLC* for biofilm but were not found by the screen to be involved in it (i.e. false positives of our method). Among the 8 top-scored false positives were 4 genes that are consecutive in the genome (so very likely to be part of the same operon), namely PA0088, PA0089, PA0090 and PA0091. Two out of these genes (PA0089 and PA0090) are annotated with the KEGG pathway for biofilm formation (pae02025). This makes it probable that the other two genes are also members of the same pathway. For instance, it is possible that these genes participate in biofilm formation under different conditions from those used at the experimental screening during CAFA-π. For motility, the highest-scored false positive was PA2496, a gene that has been found to be down-regulated in knockouts of the gene HapZ, which mediates motility [10].

These show that the CAFA-π is not a perfect ground truth, as it reflects only one experimental condition, whereas our method can find good candidate genes for a particular process under various conditions.

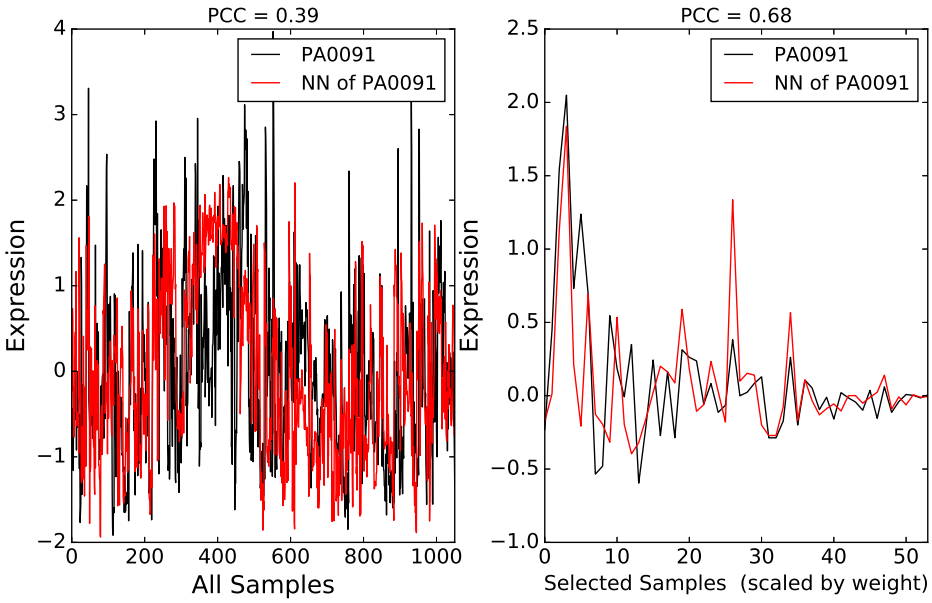


Figure 4.S12: (Left) In black, the expression profile (*y* axis) of gene PA0091 over the 1,051 samples of the dataset (*x* axis). In red, the profile of the training gene with the highest *PCC* to PA0091 over all the samples. The samples were ordered using hierarchical clustering on all genes, so that similar samples are next to each other. (Right) In black, the expression profile of PA0091 (*y* axis) on the samples that were selected by *MLC* (*x* axis). In red, the profile of the training gene with the highest *PCC* to PA0091 over only these samples. The samples are ordered in decreasing weight order and the expression is multiplied by that weight.

5

EXPERIMENTAL, DERIVED AND SEQUENCE-BASED PREDICTED PROTEIN-PROTEIN INTERACTIONS FOR FUNCTIONAL ANNOTATION OF PROTEINS

Stavros MAKRODIMITRIS, Marcel J.T. REINDERS and Roeland C.H.J. VAN HAM

Parts of this chapter have been published in PLOS ONE 15(11): e0242723 25 November as "A thorough analysis of the contribution of experimental, derived and sequence-based predicted protein-protein interactions for functional annotation of proteins" [1].

ABSTRACT

Physical interaction between two proteins is strong evidence that the proteins are involved in the same biological process, making Protein-Protein Interaction (PPI) networks a valuable data resource for predicting the cellular functions of proteins. However, PPI networks are largely incomplete for non-model species. Here, we tested to what extent these incomplete networks are still useful for genome-wide function prediction. We used two network-based classifiers to predict Biological Process Gene Ontology terms from protein interaction data in four species: Saccharomyces cerevisiae, Escherichia coli, Arabidopsis thaliana and Solanum lycopersicum (tomato). The classifiers had reasonable performance in the well-studied yeast, but performed poorly in the other species. We showed that this poor performance can be considerably improved by adding edges predicted from various data sources, such as text mining, and that associations from the STRING database are more useful than interactions predicted by a neural network from sequence-based features.

5.1. INTRODUCTION

One of the main challenges of the postgenomic era is how to extract functional information from the vast amount of sequence data that are available. As the number of known protein sequences grows at a very fast pace (currently >185 million in UniProtKB), experimentally determining the functions of all proteins has become practically infeasible. This creates the need for accurate Automatic Function Prediction (AFP) methods, which can predict a protein's function(s) using the knowledge that has been accumulated in the past. To this end, the Gene Ontology (GO) is a very valuable resource that provides a systematic representation of function in the form of three ontologies: Biological Process (BP), Molecular Function (MF) and Cell Component (CC) [2].

The Critical Assessment of Functional Annotation (CAFA) is a community-driven benchmark study that compares a large number of available AFP methods in an independent and systematic way [3–5]. One of the main conclusions that one can draw from the several editions of CAFA is that top-performing methods tend to use a combination of different data sources and not only the amino acid sequence. For example, MS-kNN, one of the best methods in CAFA2, combined sequence similarity with human gene co-expression and protein-protein interaction (PPI) data [6]. GOLabeler, which was the best in CAFA3, combined six different data sources with a powerful algorithm that predicts how suitable a GO term is for the input protein [7]. More recently, the authors of GOLabeler introduced an extension named NetGO which also uses PPI networks as an extra data source, reporting even better performance than GOLabeler on the CAFA3 dataset [8]. These observations show that PPI networks are informative data sources for AFP, which can be understood, since if two proteins physically interact, they are likely to be involved in the same biological process or pathway.

However, almost all PPI networks are incomplete. The best-characterized model species, *Saccharomyces cerevisiae* (baker's yeast), has one of the densest PPI networks, with 116,209 experimentally-derived, physical interactions in the BIOGRID database [9]. Given the fact that *S. cerevisiae* has about 6,000 protein-coding genes [10], this means that roughly 0.6% of all possible pairs of proteins are known to interact. The human interactome is also quite well characterized, with 424,074 experimental interactions in BIOGRID (about 0.2% of all possible interactions). Moreover, a recent study identified an additional 52,569 high-quality interactions of 8,275 human proteins [11]. On the other hand, in *Arabidopsis thaliana*, the most well-studied plant species, there are about 27,000 protein coding genes and 48,786 experimentally-derived physical interactions in BIOGRID, i.e. only 0.01% of the possible interactions are known. This is not likely due to protein interactions being less common in *A. thaliana*, but rather because it is not as well-studied as yeast.

The number of known edges is orders of magnitude smaller in other plant species, even in important crops. For example, in tomato (*Solanum lycopersicum*), there are only 107 interactions in BIOGRID as of June 2019 (<<0.01% of the total number of possible interactions). In rice (*Oryza sativa japonica*), there are 330 and in corn (*Zea mays*) 13. This phenomenon is not restricted to plants, but is also true for non-model animal species, such as economically important species like cow (*Bos taurus*, 529) and pig (*Sus scrofa*, 88 interactions).

Most methods that employ PPI networks in AFP predict functions by propagating

the GO annotations through the network [6, 8]. The simplest of such methods transfers the annotations of a protein to its immediate neighbors. This is also known as Guilt-By-Association (*GBA*). Figure 5.1a illustrates the *GBA* method in an example network with 6 proteins: Proteins 1 and 2 are annotated with a GO term, while protein 6 is not. We are asked to predict whether proteins 3-5 should be annotated with that GO term. As seen in Figure 5.1a, for all three of these proteins we are at least 66.6% certain that they should be assigned that GO term. Figure 5.1b shows the same example network, assuming that some of its edges are missing. In this case, protein 5 has no known interacting partners, so it is impossible to determine its function. Similarly, protein 1 has a known function, but is disconnected from the rest of the network, so its function cannot be propagated to other proteins. This example shows that when interactions in a PPI network are missing, function prediction cannot benefit from PPI information (as most proteins will have few or no connections to other proteins).

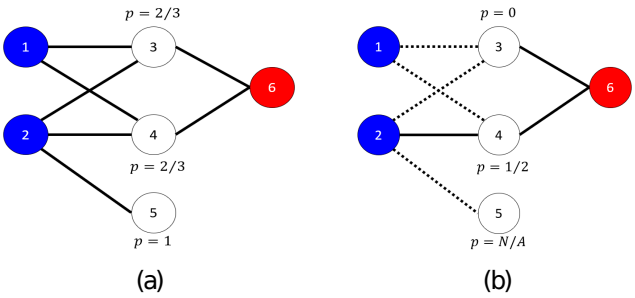


Figure 5.1: **Toy PPI network with 6 nodes.** Nodes annotated with a GO term are shown in blue and nodes not annotated in red. Unlabeled (test) nodes are shown in white. In (a) the entire network is known and the posterior probabilities for each unlabeled node can be calculated accurately. In (b) some of the edges are missing (signified by the dashed lines), making the calculation of posterior probabilities either erroneous or even impossible (e.g. node 5).

A way to counter the lack of edges is to predict them using other data sources. The STRING database contains a large collection of protein associations predicted using different sources, such as gene co-expression and text mining [12]. Moreover, the recent rise in popularity of deep learning has caused an increase in methods that attempt to predict protein-protein interactions purely from protein sequence. One of the first examples was from Sun et al. [13], followed by DPPI [14], PIPR [15] and the work of Richoux et al. [16]. The advantage of predicting edges from sequence is that it is - at least in theory - not biased towards previous experiments. In contrast to, for example, predictions within the STRING database that still require other people to have previously studied a specific protein or its orthologues. Having an accurate sequence-based predictor of PPIs means that for all possible pairs of proteins we can obtain a score for how probable an interaction between each pair of proteins is. This would enable us to find possible interacting partners for proteins that have not been previously studied at all.

In this study, we are interested in quantifying the influence of missing edges in a PPI network on protein function prediction. Moreover, we are interested in how well (deep learning based) sequence-based PPI predictors can recuperate this missing information, and how that translates in improvements of the function prediction. We hypothesize that using such a model to predict interactions would be more effective than STRING in the downstream task of network-based protein function prediction.

5.2. MATERIALS AND METHODS

5.2.1. PROTEIN-PROTEIN INTERACTION NETWORKS

We compared PPI networks in *S. cerevisiae, Escherichia coli, A. thaliana* and *S. lycopersicum* using three types of PPIs: 1) Physical interactions that have been experimentally derived. 2) Predicted interactions based on non-experimental protein association data from the STRING database, and 3) Sequence-based predicted interactions based on the amino acid sequence of two proteins using PIPR.

Physical interactions: For the experimental interactions we used the BIOGRID (version 3.5.171) [9] and STRING databases [12]. We only used physical interactions and ignored the genetic interactions. Of note, the STRING database contains a collection of experimental protein-protein interactions from different databases, including BIOGRID (marked with the "experiments" data source code) and we found edges in BIOGRID that were not present in STRING. From STRING, we only chose experimental protein-protein interactions with association scores larger than the median score over the non-zero scores for each species individually. The node degree distributions of these networks are shown in Figure 5.S1 in S1 Text.

Predicted interactions: Besides the experimental evidence, STRING contains protein associations from 12 data sources in total: "neighborhood", "neighborhood transferred", "co-occurrence", "database", "database transferred", "experiments transferred", "fusion", "homology", "co-expression", "co-expression transferred", "text mining" and "text mining transferred". We use these data as features predictive of two proteins interacting and/or being functionally associated to add edges to the experimental network. We refer to these edges as "predicted edges". Table 5.S1 in S1 Text shows the number of interactions per species and per data type. In each species, we ignored data sources that did not add any new edges. We also removed "database", as it includes protein associations that were identified by using the GO annotations of proteins and these edges would cause circular reasoning if used to predict GO terms, leading to a biased evaluation. This left us with 9 data sources from which we could infer PPIs in yeast, *E. coli* and *A. thaliana* and 8 in tomato (Table 5.S1 in S1 Text). The interaction scores have different distributions in different data sources. Therefore, instead of applying a fixed threshold, we selected the protein pairs with the 50% highest non-zero scores for each data source and species individually. Next to individually using the data sources as proxies for the protein-protein interactions, we also combined data sources. This was done by first integrating the STRING scores from different sources as described in [17] (see S1 Text for more information) and then keeping the 50% top non-zero scores for every combination,

as before. To combine a binary STRING network with the experimental one, we applied an element-wise logical OR to the corresponding adjacency matrices, so an interaction is added to the combined network if it is present in at least one of the original networks.

We also examined the possibility of using all STRING edges by creating weighted graphs whose edge weights correspond to the STRING interaction scores. We then added these weighted graphs to the binary experimental network.

Sequence-based predicted interactions: We used PIPR [15] to predict PPIs from protein sequence. It uses a Siamese twin architecture with both convolutional and recurrent units and three fully connected layers at the end. PIPR also makes use of predefined amino acid embeddings, obtained from both chemical properties of amino acids and their co-occurence in protein sequences. PIPR had an accuracy of about 97% in predicting yeast PPIs when trained on a large, balanced dataset from the DIP database. After having trained the model, we feed it all pairs of proteins. For each pair we get a score in the range [0,1] denoting the probability that these two proteins interact. We add an edge to our predicted PPI network if the score for that edge is greater than or equal to 0.5.

5.2.2. GO ANNOTATIONS

We obtained GO annotations from the GOA website [18] and only used the experimental annotations and curated annotations (evidence codes "EXP", "IDA", "IPI", "IMP", "IGI", "IEP", "IBA", "IBD", "IKR", "IRD" and "TAS"). We used the entire GO graph (not the smaller GO slim versions). Annotations were propagated towards the ontology root, so that when a protein is annotated with a term, it is also annotated with all its ancestors in the GO graph. We focused on the Biological Process Ontology (BPO), as it is the most difficult ontology to predict [4] and also is the most commonly used in further analyses such as gene set enrichment. Table 5.1 gives an overview of the different dataset sizes for the four species.

Table 5.1: Number of proteins and known PPIs per species in BIOGRID. (version 3.5.171)

	Yeast	E. coli	Arabidopsis	Tomato
approximate #protein-coding genes	6,000 [10]	4,400 [19]	27,029 [20]	34,727 [21]
#proteins with BPO annotations (N)	4,997	2,869	10,648	651
#BIOGRID edges between proteins with BPO annotations	149,659	17,540	23,371	57
#pairs of proteins with BPO annotations ($N(N-1)/2$)	12,482,506	4,114,146	56,684,628	211,575
% annotated protein pairs interacting	1.20	0.43	0.04	0.03
% disconnected proteins	0.4	23.1	43.4	96.9

5.2.3. FUNCTION PREDICTION METHODS

We represent the protein-protein interactions as a network with the proteins as nodes and the interactions as binary, undirected edges. Using this network, we can make predictions about the functions of unannotated proteins using the proteins with known function. To do so, we used a simple Guilt-By Assosciation (GBA) method and a more complicated one that uses node embeddings learned using *node2vec* [22]. We compared these methods to the *BLAST* and *naive* baselines, which are commonly used in the CAFA challenges [3, 4]. Each method computes the probability $P(p_i, t)$ that a GO term t should annotate protein p_i. Below we provide details about how each method makes this computation. When $P(p_i, t)$ is undefined, e.g. because a protein has no neighbors in a PPI

network or no significant *BLAST* hits, we set it to zero to indicate that this term cannot be assigned to this protein.

Guilt-By-Association (GBA): This method assigns a GO term to a protein with posterior probability equal to the fraction of the protein's interacting partners annotated with that term. More formally, let A be the network's adjacency matrix, V_{train} a set of training proteins and V_{test} a set of test proteins. Moreover, let $T(p)$ be the set of GO terms assigned to $p \in V_{train}$. For a protein $p_i \in V_{test}$, we define its neighborhood $N(p_i)$ as all its interacting partners that are in the training set:

$$N(p_i) = \{p : p \in V_{train} \wedge A[p, p_i] = 1\} \tag{5.1}$$

For a GO term t, the probability it is assigned to test protein p_i is given by Equation 5.2:

$$P(p_i, t) = \frac{\sum_{p \in N(p_i)} I(t \in T(p))}{|N(p_i)|} \tag{5.2}$$

Where $I(x) = 1\ iff\ x$ is a true statement and $|S|$ denotes the number of elements in set S.

For weighted graphs, Equation 5.2 was adapted so that each neighbor transfers its annotations with a weight equal to the edge weight and we divide by the total sum of the weights instead of the number of neighbors.

node2vec: The *node2vec* algorithm learns a fixed-length embedding for every node, such that the similarity in the embedding space reflects the similarity of neighborhoods in the graph, as defined by random walks [22]. We used these embeddings as feature vectors on which we applied standard machine learning methods; specifically the k-Nearest Neighbors (kNN) and the ridge classifiers. For kNN, we look for the k training proteins with the most similar feature vectors to a query protein p_i and set $P(p_i, t)$ equal to the fraction of these k proteins annotated with t. The ridge classifier models protein function prediction as a multi-output regression problem and learns a linear mapping from the feature space to the label space. We use $\mathbf{X} \in \mathbb{R}^{N \times d}$ to denote the *node2vec* feature matrix, where each row contains the feature vector of one protein, and $\mathbf{Y} \in \{-1, 1\}^{N \times L}$ to denote the label matrix, where each row represents the GO annotations of each protein and a value of 1 in the matrix denotes that the corresponding protein is annotated with the corresponding GO term. The ridge classifier tries to find a linear mapping $W \in \mathbb{R}^{d \times L}$, such that $\mathbf{Y} \approx \mathbf{XW}$. We also add L2 regularization to the model with coefficient λ which leads to the optimal solution $\mathbf{W}^* = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$. To bring the predictions ($\mathbf{XW}^*$) in the range [0,1], we apply a sigmoid function $s(a) = (1 + e^{-a})^{-1}$ to each predicted value a. We did not post-process the predictions of the ridge method so it is possible that it makes predictions that are inconsistent with the GO hierarchy.

Naive: The *naive* method of CAFA [3] assigns a GO term to a protein with probability equal to the fraction of training proteins annotated with that term (Equation 5.3).

$$P(p_i, t) = \frac{|\{p : p \in V_{train} \wedge t \in T(p)\}|}{|V_{train}|} \tag{5.3}$$

This means that all test proteins get the same annotation using this method (making it a quite weak baseline).

BLAST: We ran *BLAST* with default settings and set $P(p_i, t)$ equal to the maximum sequence identity between p_i and its hits annotated with t.

Combining two classifiers: Given the posterior probabilities of two classifiers $P_1(p_i, t)$ and $P_2(p_i, t)$ we combined them using Equation 5.4, which gives a high score for a protein-term pair if at least one of the two methods gives a high score.

$$P_{combo}(p_i, t) = 1 - (1 - P_1(p_i, t)) \cdot (1 - P_2(p_i, t)) \tag{5.4}$$

5.2.4. EXPERIMENTAL SET-UP

Evaluation metrics: To compare function prediction across the differently constructed protein-protein interaction networks, we applied a 5-fold cross-validation. As evaluation metrics we used the protein-centric F_{max} and S_{min} that are extensively used in the CAFA challenges. Definitions for these metrics are provided S1 Text. We also measured the coverage of each algorithm, defined as the fraction of test proteins for which at least one term has a non-zero posterior probability.

As the GO term distributions and frequencies are different in each species, directly comparing the performances across species is not trivial. To counter the effect of GO term frequencies, we use the concept of Prediction Advantage (PA) [23], which is defined as the improvement on the classification loss of a classifier c (L_c) with respect to the *naive* classifier (L_{naive}). The PA, which is defined in Equation 5.5, can be calculated for any classification loss, so here we used $L = 1 - F_{max}$.

$$PA(c, L) = 1 - \frac{L_c}{L_{naive}} \tag{5.5}$$

In each fold, we discarded the GO terms that had no positive examples in either the training or the test set.

Experimental PPI (*EXP*): We started from the experimental PPI network of a given species. This network includes as nodes all proteins that have at least 1 functional annotation, even if they have no interacting partners. Proteins without functional annotations were removed, even if they had known interactions.

node2vec is an unsupervised feature extraction step that only depends on the network and not the functional annotations. We additionally tested whether also including the unannotated proteins as nodes in the network would possibly lead to better features in the first step of the *node2vec* procedure, as it leads to a better neighborhood estimation. To this end, we ran *node2vec* on the entire *EXP* network (including unannotated proteins) and then used the extracted (*unsupervised*) features of the annotated proteins only in the *supervised* phase. We repeated this experiment for all four species and compared the performance with that of the original *node2vec* which learned the (unsupervised) features on a network of only annotated proteins.

Combined experimental and predicted PPI (*EXP+STRING*): We added predicted edges to the experimental network from the different data sources in STRING. We evaluated all possible combinations of the 9 STRING data sources (8 for tomato): First, we added each data source individually. Then, we tested all combinations of 2 data sources (36 possibilities), all combinations of 3 (84 possibilities) and so on, until we have included all 9 data sources. So, in total, we tested $\sum_{i=1}^{9}\binom{9}{i} = 511$ combinations of data sources (255 for tomato) along with the experimental network.

Sequence-based predicted PPI (*EXP+SEQ*): We used edges predicted by PIPR for predicting function. We tested the performance of a network with the experimental edges combined with the PIPR predictions.

OPTIMIZATION OF *node2vec* CLASSIFICATION

node2vec has hyperparameters that can have a large influence on the learned features. We tuned these hyperparameters on the experimental PPI network of each species, by splitting the training set of each cross-validation fold into a new training (80% of initial training set) and a validation set (20% of intial training set). For each hyperparameter combination, we generated node features which we fed to the *k*NN and ridge classifiers for different values of their parameters (k and λ respectively). Finally, for each cross-validation fold, we identified the combination of hyperparameters, classifier and classifier parameter that maximized the F_{max}, trained it on the whole training set and used the trained model to make predictions on the test set. Details about the hyperparameters that were tuned and the values considered are provided in S1 Text.

When running *node2vec* on all proteins with known interactions (and not only the ones with functional annotations), we again used 5-fold cross-validation as before. The training, validation and test splits in each fold were kept identical. We also repeated the hyperparameter optimization step, as changes in the network topology might call for different hyperparameter values.

5.3. RESULTS

5.3.1. ONLY THE YEAST EXPERIMENTAL PPI NETWORK HAS ACCEPTABLE FUNCTION PREDICTION PERFORMANCE

Figures 5.2a-d compare the F_{max} achieved by the *GBA* method on the *EXP* network to the baseline performances in four species using 5-fold cross-validation. In yeast, this simple approach significantly outperforms both *naive* (p-value$< 10^{-5}$, paired t-test, FDR-corrected) and *BLAST* (p-value$=0.5 \cdot 10^{-3}$, paired t-test, FDR-corrected). In *E. coli*, *A. thaliana* and tomato, the picture is quite the opposite, with even the *naive* method largely outperforming *GBA* (p-values=0.026, $0.3 \cdot 10^{-5}$, $0.2 \cdot 10^{-3}$ respectively, paired t-test, FDR-corrected). In tomato, the network is so sparse and disconnected that the maximum F1 score is achieved by assigning all GO terms to all proteins. The Prediction Advantage (PA, see Methods) between *GBA* and *naive* classifier follows a linear trend with respect to the fraction of existing edges. The calculation was based on only four points, but it still lies under the statistical significance threshold of 0.05 (Figure 5.2e, Pearson's ρ=0.98, p-value=0.016).

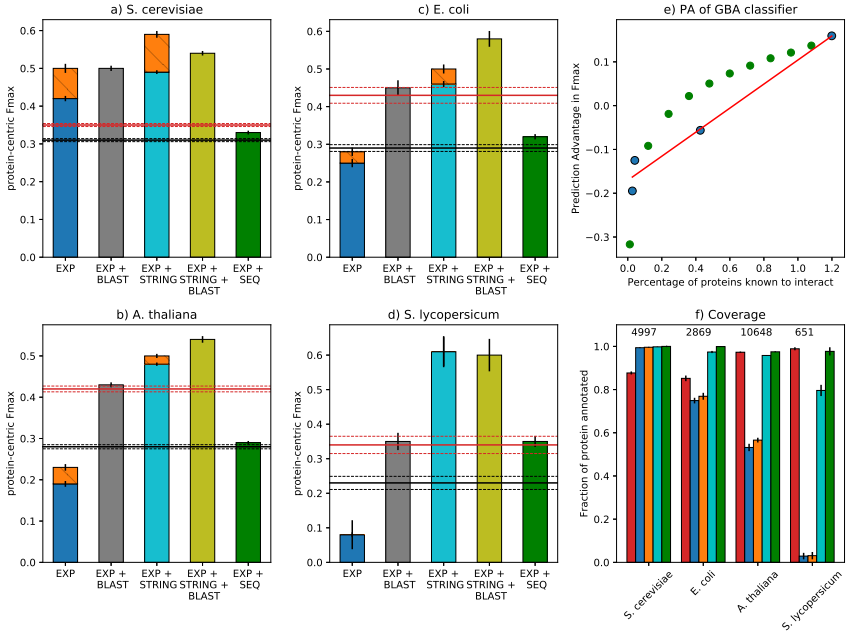


Figure 5.2: **Function prediction performance of PPI networks in four species.** (a-d): On the *x*-axis, are the different PPI networks. The height of the bars denotes the F_{max} in each species. The *naive* and *BLAST* baselines are shown as a red and a black horizontal line respectively, with dashed lines showing the corresponding standard deviations. *EXP, GBA* is shown in blue, *EXP+STRING, GBA* in cyan and *EXP+SEQ* in green. The improvement of *node2vec* on *EXP* and *EXP+STRING* is shown as an orange bar. Absence of an orange bar denotes that the two algorithms performed equally. The combinations of *EXP, GBA* and *EXP + STRING, GBA* with BLAST are shown in gray and yellow respectively. The error bars denote the standard deviation over the 5 cross-validation folds. e) Prediction Advantage (PA) of F_{max} as a function of the fraction of known interactions. Each species is shown as a blue dot and red line shows the least squares linear fit. PA values calculated by downsampling the original yeast network at different levels of missing edges are shown as green dots. f) The fraction of annotated proteins for which each method can make predictions (*y*-axis) for each species (*x*-axis). On top, the number of total proteins is shown. Different methods are shown in the same colors as in a-d. Note that the naive method has a coverage of 100% by design.

To better characterize the effect of missing edges, we simulated the phenomenon in yeast by removing edges either uniformly at random or by an approach that makes nodes with the lowest degree more likely to lose their edges first (S1 Text). We found that the F_{max} is relatively robust to uniform edge removal up to 40-50%, but S_{min} deteriorates more quickly (Figure 5.S2 in S1 Text), meaning that predicting more specific terms suffers even under this simplified missing edges scenario. The coverage also drops very slowly (at least initially), which implies that most edges are removed from "dense" parts of the network so that the remaining edges can partly make up for this loss. In the degree-based sampling strategy, which is more realistic, we observed a much steeper drop for

all three metrics. In this case, poorly-studied proteins lose their connections very quickly making it impossible to make predictions for them, as indicated by the steep decline in coverage. As a result, the average performance also reduces very fast. The PA values calculated from the degree-based downsampling did not confirm the linear relationship between PA and fraction of known edges (green dots in Figure 5.2e).

Combining PPI networks with homology: In many function prediction pipelines, PPI networks are combined with other data sources and used in ensemble algorithms. Experiments with a simple method that fuses the posterior probabilities of *BLAST* with those of the PPI classifier (Equation 5.4) showed minimal performance gains (2-6%) with respect to stand-alone *BLAST*, for all species except for *S. cerevisiae* (43%, Figure 5.2). The difference with respect to *BLAST* was found statistically significant using the paired t-test. However, after correcting for multiple testing using the False Discovery Rate method, the p-values for *E. coli*, *A. thaliana* and tomato lie just below the 5% significance threshold (0.0468, 0.0468 and 0.0486 respectively), whereas for yeast the corrected p-value is $1.5 \cdot 10^{-5}$. These results confirm that using experimental PPI networks with many missing edges is not helpful for function prediction.

***node2vec* results:** The *GBA* method is very simple and therefore unlikely to be able to capture all the functional signal present in complicated biological networks. We therefore tested whether a more complicated classifier based on *node2vec* could outperform it. In the same cross-validation loop, we used a validation set to tune the hyperparameters of *node2vec* and used the same unseen test set as before to evaluate the model. The optimal hyperparameter values varied per cross-validation fold and per species. The *1*NN classifier was the optimal choice in yeast and tomato, while the ridge with moderate regularization in *E.coli* and *A. thaliana*. More importantly, *node2vec* performed better than *GBA* on the *EXP* network in all species except for tomato, where assigning all terms to all proteins still maximizes the F_{max} (Figure 5.2a-d, Table 5.S2 in S1 Text). Evaluation based on S_{min} gave similar results (Table 5.S3 in S1 Text).

We also tested whether including proteins with known interactions but no functional annotations during the feature learning step could improve the performance of *node2vec*. We used the t-test to compare the F_{max}, S_{min} and coverage of these networks to the ones that consist of only annotated proteins. We found that doing so lead to a small but significant increase in coverage in *E. coli* and *A. thaliana* (paired t-test, corrected for the FDR), but there was no significant difference in F_{max} or S_{min} in any of the four species (FDR > 0.05, Table 5.S4 in S1 Text). This means that although we can make predictions for more proteins the predictions become less accurate when including these edges. Therefore, for the rest of our experiments we only refer to *node2vec* trained on the proteins that have GO annotations.

Performance per protein: Comparing the performance for each individual protein, we observed a large non-linear dependency between the performance and the number of annotated neighbors. This dependency was consistently smaller for *node2vec* (a Spearman correlation of 0.30, 0.60 and 0.81 for yeast, *E. coli* and *A. thaliana* respectively) than for *GBA* (0.41, 0.65 and 0.85 for yeast, *E. coli* and *A. thaliana* respectively). We also

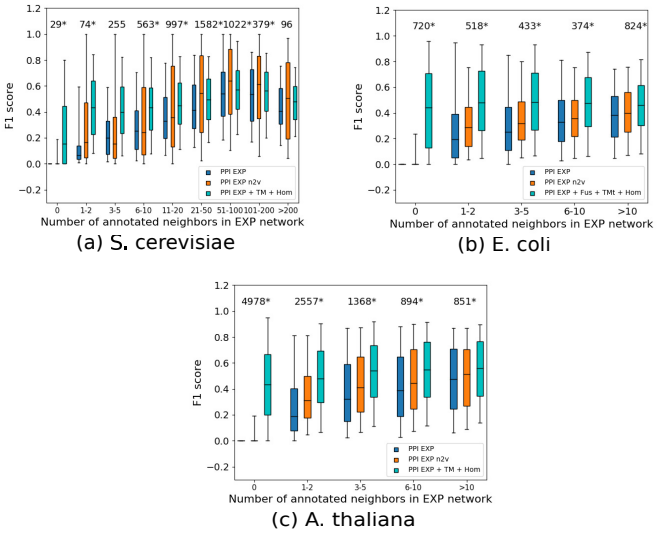


Figure 5.3: **Performance per protein.** F_{max} achieved per protein (*y*-axis) as a function of the number of training neighbors in the *EXP* network (*x*-axis) for *EXP, GBA* (blue), *EXP, node2vec* (orange) and *EXP+STRING, GBA* (cyan). The median of each group is denoted by a horizontal line and the 5^{th} and 95^{th} percentiles by the whiskers. The number of proteins in each group is shown at the top of each group and an asterisk (*) next to the number signifies that the difference between *EXP, GBA* and *EXP, node2vec* is statistically significant at a False Discovery Rate of 5%

. For the *EXP+STRING* network, we show the performance of the combination of data sources that had the best performance in each species.

found that *node2vec* consistently outperforms *GBA* regardless of the number of annotated (training) neighbors in *E. coli* and *A. thaliana* (Wilcoxon rank sum test, FDR < 0.05, Figure 5.3 and Table 5.S4 in S1 Text). In *S. cerevisiae, node2vec* is significantly better than *GBA* for 6 out of 9 bins and significantly worse in 1 bin, while for two bins there were no significant differences (Wilcoxon rank sum test, FDR < 0.05, Figure 5.3 and Table 5.S5 in S1 Text). Finally, *node2vec* can make predictions for proteins that do not have any training neighbors as long as they are not completely disconnected, as its feature vectors are learned in an unsupervised way using the entire network. This means that, for not too sparse networks, *node2vec* is the preferred option compared to *GBA*.

5.3.2. ADDING PREDICTED EDGES IS MORE USEFUL THAN USING A COMPLEX CLASSIFIER

We then tested to what extent predicted interactions from STRING can improve upon the protein function prediction performance of the *EXP* networks. As we can see in Fig-

ure 5.2a-d, the *GBA* classifier performed considerably better on the *EXP+STRING* network than on *EXP* for all species. It also significantly outperformed the *naive* and *BLAST* baselines. As shown in Figure 5.3, the STRING edges offer a performance boost for both nodes that have and nodes that do not have annotated neighbors in the experimental network for all species. However, for hub yeast proteins with more than 20 experimental edges, applying *node2vec* on the *EXP* network was more effective than adding predicted edges (Figure 5.3a). The fraction of proteins that can be annotated by the *STRING* networks approaches 100% for *E. coli* and *A. thaliana* and 80% for tomato (Figure 5.2f).

Using a weighted STRING network with all available interactions instead of a binary one lead to small performance improvements, but mainly for the combinations that performed less well (Figure 5.S3-5.S6). The effect sizes were rather small for the top-performing combinations (Table 5.S6). This shows that STRING edges possibly contain useful functional signal even at confidence levels lower than those we considered here.

Combining STRING edges with homology: Moreover, combining the predictions of the *GBA* classifier on this network with *BLAST* predictions (see Methods) leads to significant improvement (28-76%) over *BLAST* for all species (Figure 5.2). The combined model gave significant improvements (10-26%) over its PPI component in yeast, *E. coli* and *A. thaliana* and performed equally well in tomato (Figure 5.2). S_{min} results show similar trends, with the exception that in yeast, the optimal S_{min} is achieved by *GBA* on the *EXP+STRING* network and not by the combination with *BLAST* (Tabe S3 in S1 Text). These show that adding predicted edges is very beneficial for all tested PPI networks.

***node2vec* on STRING edges:** Similar to the *EXP* network, we compared the *GBA* classifier to the one based on *node2vec* on *EXP+STRING*. We again observed that the more complex classifier achieved higher F_{max} in yeast, *E. coli* and *A. thaliana* (Figure 5.2a-d), but in terms of S_{min} only yeast showed an improvement (Table 5.S3 in S1 Text). In addition, Figures 5.2b-d show that in not so well-studied species, using a more complicated classifier on the *EXP* network performs considerably worse than a simple classifier on a more complete network with predicted edges.

Effect of individual STRING data sources: We also examined which *STRING* data sources were responsible for the observed increase in performance. As shown in Figure 5.4 and Figures 5.S7-5.S9 in S1 Text, the vast majority of data sources when individually added to the *EXP* network lead to better function prediction in terms of both F_{max} and S_{min}, with the exception of "experiments transferred" in yeast. Figure 5.4 and Figures 5.S7-5.S9 in S1 Text also show that "text mining" (in *S. cerevisiae* and *A. thaliana*), "text mining transferred" (in *E. coli* and *S. lycopersicum*) and "homology" (in all four) were by far the most useful sources. A more in-depth analysis of the results showed that these three data sources alone are actually enough to obtain the maximum performance of the *GBA* method on the *EXP+STRING* network (Tables 5.S7-5.S14 in S1 Text) and that removing all of them leads a to significant performance drop (Tables 5.S15-5.S16 in S1 Text). Moreover, including all nine data sources (eight for tomato) lead to worse F_{max} and S_{min} in all species (Figure 5.4 and Figures 5.S7-5.S9 in S1 Text).

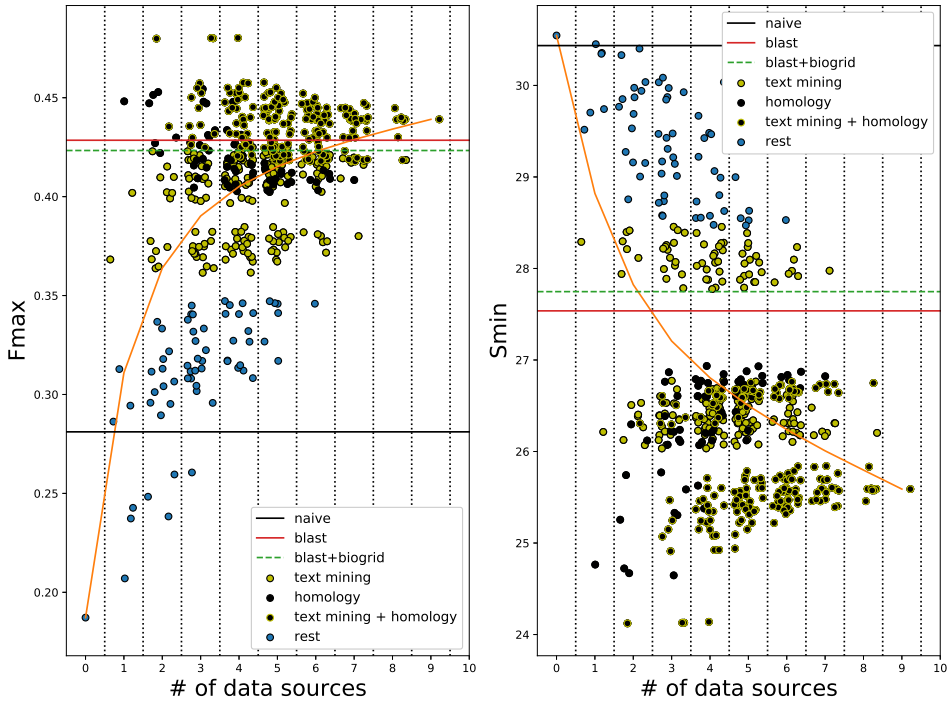


Figure 5.4: **Performance of STRING edges in *A. thaliana*.** F_{max} (left) and S_{min} (right) (*y*-axis) as a function of the number of *STRING* data sources included (*x*-axis). Each dot corresponds to one combination of data sources added to the experimental network. Combinations that include "text mining" and/or "text mining transferred" are shown in yellow, combinations that include "homology" in black and combinations that include both in black with yellow border. The rest of the combinations are shown in blue. To ease visibility, we added a random number in the range [-0.5, 0.5] to each combination of the same number of sources. Zero data sources corresponds to the *EXP* network and the orange line shows the average performance for a specific number of data sources. Horizontal lines denote the performance of the *naive* (black), *BLAST* (red) and the combination of *BLAST* with the *EXP* PPI network (dashed green).

5.3.3. EDGES PREDICTED FROM PROTEIN SEQUENCES BY A NEURAL NETWORK ARE LESS USEFUL THAN STRING EDGES

The *PIPR* model for predicting protein-protein interactions from sequence was reported to have 97% cross-validation accuracy on a balanced dataset with about 11,200 data points from *S. cerevisiae* proteins from the DIP database, a result that we also replicated. This model, however, was not able to generalize to predict BIOGRID edges in yeast, as it achieved an accuracy of 0.59 on a balanced dataset. We also measured the model's recall, i.e. its ability to identify true interacting pairs, and it was comparable to random guessing (0.51).

We therefore set out to train PIPR for predicting BIOGRID edges, keeping the architecture and the training procedure the same. As positive training examples, we used all yeast protein pairs reported to be physically interacting in BIOGRID and as negative examples, an equal-sized set of randomly selected protein pairs that are not reported as

interacting. This proved to be a more challenging task for PIPR, as the best validation accuracy achieved was 0.77 (Table 5.S17 in S1 Text).

The sequence-based predicted PPI network combined with the experimental one (*EXP+SEQ*) hampers the AFP performance in yeast as compared to *EXP* (Figure 5.2a). This is probably due to the addition of many false positive edges, as it predicts that more than 41% of all possible protein pairs are interacting, which is about 10 times more than expected [17]. In contrast, in *E.coli*, *A. thaliana* and tomato the *EXP+SEQ* PPI network seems to be more useful, providing significant improvements over *EXP* (Figure 5.2b-d). However, these improvements are not enough to surpass even the *BLAST* baseline in *E. coli* and *A. thaliana*. Contrary to our expectation, the *EXP+STRING* network performed significantly better than *EXP+SEQ* for all species (Figure 5.2a-d). This was true even when we removed edges from text mining from the *STRING* networks.

In tomato, the *EXP+STRING* network cannot make predictions for roughly one fifth of the proteins (Figure 5.2f). Adding the *SEQ* edges only for these proteins improved the overall F_{max} from 0.61 to 0.67. This shows that *SEQ* edges are useful, but they are surpassed by the higher quality of *STRING* edges.

Finally, we trained *PIPR* on *A. thaliana* edges from BIOGRID and obtained new networks in *A. thaliana* and *S. lycopersicum*. Although this network worked slightly better in tomato than the one trained in yeast data, it was still worse than *BLAST* and *EXP+STRING* (Table 5.S18 in S1 Text).

5.4. DISCUSSION

The aim of this work was to investigate ways of addressing the problem of missing edges in experimental protein-protein interaction networks for the downstream task of genome-wide function prediction. Our main hypothesis was that a deep learning model that can identify interacting proteins from sequence with very high accuracy would be a good solution to this issue.

We demonstrated how the sparsity of experimental PPI networks leads to poor function prediction performance, using the simple *GBA* classifier. We did not compare this classifier to any state-of-the-art methods, such as GOLabeler [7] or INGA [24], but rather to the *naive* and *BLAST* baselines from the CAFA challenges. The *naive* classifier, as its name suggests, does not use any information to relate specific proteins to GO terms, rather it only uses the frequency of each GO term in the training set. In the machine learning literature, this classifier is also called the "Bayesian Marginal Predictor" [23] and is the optimal classifier when the distributions of the classes ($P(y)$) are known, but information about the relationship between the data and the classes ($p(x|y)$) is missing. This means that any classifier that uses any kind of (informative) data is expected to outperform the *naive* one.

However, we clearly demonstrated the failure of the *GBA* classifier in predicting BPO terms in *E. coli*, *A. thaliana* and tomato, as it performed considerably worse than the *naive* method. This was not the case in yeast, where the *GBA* classifier outperformed both baselines. When examining the performance for individual proteins, we found a high correlation between the number of known interacting partners and the prediction accuracy.

The *GBA* method has proven to be very useful in function prediction [6], but it is a

very simple approach and therefore heavily relies on the correctness of the given network. We thus expected that using a more complicated approach that captures broader network patterns might (partly) overcome the sparsity. Several such node classification methods exist [25]. Recently, Graph Convolutional Networks (GCNs) have been shown to be effective in such tasks [26]. We chose to use *node2vec* to generate node features, as it has been successfully applied to protein-protein interaction networks [22] and used these features to train standard classifiers for function prediction. Although, we observed a clear improvement in *A. thaliana* and *E. coli* with respect to *GBA*, the performance remained below that of the baselines, meaning that these models can only partly compensate for missing edges. To make matters worse, we did not observe any improvement in the even sparser tomato network. This difference can be explained by the fact that when tuning the *node2vec* hyperparameters we rely on the performance on a validation set, which in tomato is very small and only includes "easy" proteins, leading to an apparent high performance for a large number of hyperparameter combinations. This makes it hard to select the optimal hyperparameters for *node2vec* in tomato, but it is still possible that an improvement could be observed if the correct parameters were known. Using the optimal hyperparameters from another species with a more complete network, e.g. A. thaliana, might be an alternative. However, since the topologies of the two networks are vastly different, the optimal hyperparameters for one species are not necessarily good for the other. Taken together, these observations validated our hypothesis that a sparse PPI network is detrimental to genome-wide AFP.

It is worth noting that *node2vec* can make predictions for nodes that have no annotated neighbors, as opposed to *GBA*, which helps increase the coverage. Nevertheless, including unannotated proteins with known interactions during the *node2vec* feature learning step did not lead to better function prediction performance. This hints that -apart from the lack of known interactions- the lack of GO annotations for training proteins also has a considerable negative effect on the accuracy of function prediction algorithms.

Many methods have been proposed that try to complete a network by predicting edges. Reviews of such methods can be found in [27] for social and in [28] for biomedical networks. More specifically, the computational prediction of protein-protein interactions has been an active research area for many years [29, 30]. Our work is the first to evaluate the contribution of predicted edges in protein function prediction in a species-specific way. We used the STRING database as a proxy for predicting interaction using omics data such as genome features, homology, co-expression and text mining. In sparse experimental PPI networks, the *STRING*-derived edges contribute a great deal, increasing the performance of the *GBA* classifier 1.8-fold in *E. coli*, more than 2.5-fold in *A. thaliana* and about 30-fold in tomato. They also outperformed the *node2vec* method on the *EXP* network. This is because these extra edges connect proteins that were previously disconnected from the rest of the graph, but also because they can discover new functions for already connected proteins, leading to a performance boost regardless of the number of neighbors. Using these edges was enough to significantly outperform the *naive* and *BLAST* baselines. In the case of yeast, which has a more "complete" network, the *STRING*-derived edges also improved the prediction performance, but to a lesser extent. In fact, in yeast, *node2vec* on the *EXP* network and *GBA* on the

EXP+STRING network performed similarly on average, with *node2vec* being more useful for hub proteins that have a complicated neighborhood. As expected, combining a better network (*EXP+STRING*) with a better classifier (*node2vec*) lead to even better performance, though this was not observed in the small tomato dataset.

To combine the different *STRING* data sources, we used the simple algorithm described in [17]. This algorithm (also described in S1 Text) assumes independence between the data sources and applies a Bayesian framework to join them into a final score for each protein-protein association. Some more advanced methods have been proposed to perform this integration, such as *Mashup* [31] and *deepNF* [32]. Both of these approaches, which are conceptually similar to each other and to *node2vec*, perform a number of random walks separately for each network derived by each data source to estimate the neighborhood similarity of each node to all other nodes. Then, they learn a feature vector for every node (protein) in order to approximate this similarity as closely as possible. The main difference between the two methods is that *Mashup* learns these vectors using matrix factorization [31], while *deepNF* using an autoencoder neural network [32]. Both of these methods outperformed the simple integration strategy in yeast and human PPI networks [32], which means that the performance of the *EXP+STRING* network could be enhanced by using one of these two methods. On the other hand, these methods - and especially *deepNF* that has many parameters to be learned - are not guaranteed to work well in a small dataset such as the tomato one. Furthermore, as *STRING* networks have weighted edges, instead of using thresholds to make them binary, it might be more helpful to employ algorithms that classify nodes directly on weighted graphs, such as those described in [33] and [34]. Our small-scale experiments in that direction gave mixed results, so more research is needed on this issue.

Notably, text mining of scientific literature and homology were the most informative STRING data sources for all species. Although removing the text mining edges did lead to a decrease in the maximum performance of *EXP+STRING* networks, we showed that it did not change the main conclusions of this study. Moreover, we found that edges from "text mining transferred", i.e. associations that have been discovered through text mining in other species and then transferred based on sequence homology, are very useful in *E. coli* and tomato. Given that we did not consider GO annotations inferred automatically due to sequence similarity, it is likely that text mining indeed captures true functional information that is conserved across species. This perhaps means that text mining is an underrated data source for functional annotation. We hypothesize that since scientific knowledge is mainly disseminated by publishing articles, text mining on these articles compiles all of this information into one resource. This would explain why otherwise very informative resources such as gene co-expression or operons (in bacteria) are individually useful when added to the *EXP* network, but are rendered redundant in the presence of text mining edges. Although homology is the most commonly used data source for function prediction, from the descriptions of the methods submitted to the CAFA challenges, we know that only a small minority of them make use of text mining [5]. Two of these methods are described in [35, 36]. A more recent study showed that integrating homology-based predictors with neural-network-based text models leads to a significant performance boost [37], so we expect the role of text mining in function prediction research to be expanded in the future.

We also applied a sequence-based neural network model (PIPR) for PPI edge prediction. Firstly, we noticed that although PIPR was very accurate in predicting edges in one yeast dataset, it did not immediately generalize to another dataset from the same species, performing very close to random guessing. Richoux et al. have reported that overfitting and information leaks from the validation set are common when training protein-protein interaction predictors [16]. Although a certain protein pair from the test set cannot be present in the training set too, the two individual proteins can be in the training set in other pairs. This can have an effect for hub proteins with many interacting partners, as in an extreme case the network could learn to always predict this protein to interact with any other protein [16]. The result of these findings as well as ours is that caution is required when using these deep models, despite their high accuracy in one dataset.

Nevertheless, PPIs predicted from the PIPR model can be useful for the downstream task of network-based function prediction, as they outperformed the *naive* baseline. However, our hypothesis that such a model could accurately produce the entire or a big part of the interactome of a species leading to very accurate predicted annotations was not validated, as *STRING* edges proved more useful. Our experiments in tomato showed that for proteins that were disconnected in the *EXP+STRING* network, adding *SEQ* edges gave a significant performance increase, while this was not the case for combining the *EXP+STRING* network with *BLAST*. This implies that *SEQ* can be a useful resource for species with very few protein associations known in *STRING*.

Another limitation of our study is that except for the variable degree of unknown PPIs among the tested species, there is also a large variability in the amount of missing experimental annotations, with yeast being the most well-characterized species and tomato by far the least. This means that it is much more likely that a correctly predicted protein-GO term pair is flagged as a false positive in tomato than in yeast, simply because that annotation has not been discovered yet. Moreover, the GO terms have different frequencies in the four species, meaning that is virtually impossible to compare performances across species. For example, yeast contains a lot more specific annotations than e.g. tomato. This is also demonstrated by the large differences in S_{min} of the *naive* method, which means that the total information content of the terms present in each species is vastly different. Calculating the Prediction Advantage with respect to the *naive* method [23] can correct for differences in term frequencies, but the different degree of missing annotations is harder to correct for while only using experimental annotations. This is not a big issue in our analyses because we did not focus on the exact performance values, but rather on how the performances of different networks (i.e. networks with different edge types) compare to each other within a species. Also, we have shown that the same conclusions can be drawn when evaluation is done using the semantic distance [38], which punishes shallow predictions.

Although F_{max} and S_{min} are the most widely-used evaluation metrics for function prediction, a recent study has raised concerns about them [39]. The concerns, which were based on artificially generated predicted annotations, mainly have to do with these metrics being overly lenient to false positive predictions. This might not be a big problem, as due to missing annotations most proteins are likely to be under-annotated. The same study showed that both metrics correlate highly with the signal to noise ratio of the

predictions [39]. Based on that we argue that our conclusions do not rely on the choice of evaluation measures, but we believe that proper evaluation of function prediction algorithms is a pressing issue that requires further research.

5.5. CONCLUSION

Our work highlights the difficulty of applying PPI networks in AFP for less well-studied species. We show that predicted PPIs can partially compensate for the sparsity of the networks, with STRING-predicted edges to be the most useful, especially text mining and homology, and sequence-based deep learned predictions mostly to be useful when nodes are still not connected when combining experimental and STRING based PPI edges.

REFERENCES

[1] S. Makrodimitris, M. Reinders, and R. van Ham, *A thorough analysis of the contribution of experimental, derived and sequence-based predicted protein-protein interactions for functional annotation of proteins,* PLOS ONE **15**, 1 (2020).

[2] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, *Gene Ontology: tool for the unification of biology,* Nature Genetics **25**, 25 (2000), arXiv:10614036 .

[3] P. Radivojac, W. T. Clark, T. R. Oron, A. M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, A. Ben-Hur, G. Pandey, and J. M. Yunes, *A large-scale evaluation of computational protein function prediction,* Nature Methods **10**, 221 (2013), arXiv:1510.05682 .

[4] Y. Jiang *et al.*, *An expanded evaluation of protein function prediction methods shows an improvement in accuracy,* Genome biology **17**, 184 (2016), arXiv:1601.00891 .

[5] N. Zhou *et al.*, *The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens,* Genome Biol. **20**, 244 (2019).

[6] L. Lan, N. Djuric, Y. Guo, and S. Vucetic, *MS-kNN: protein function prediction by integrating multiple data sources.* BMC bioinformatics **14 Suppl 3**, S8 (2013).

[7] R. You, Z. Zhang, Y. Xiong, F. Sun, H. Mamitsuka, and S. Zhu, *GOLabeler: Improving sequence-based large-scale protein function prediction by learning to rank,* Bioinformatics (2018), 10.1093/bioinformatics/bty130.

[8] R. You, S. Yao, Y. Xiong, X. Huang, F. Sun, H. Mamitsuka, and S. Zhu, *NetGO: improving large-scale protein function prediction with massive network information,* Nucleic Acids Research (2019), 10.1093/nar/gkz388.

[9] R. Oughtred, C. Stark, B. J. Breitkreutz, J. Rust, L. Boucher, C. Chang, N. Kolas, L. O'Donnell, G. Leung, R. McAdam, F. Zhang, S. Dolma, A. Willems, J. Coulombe-Huntington, A. Chatr-Aryamontri, K. Dolinski, and M. Tyers, *The BioGRID interaction database: 2019 update,* Nucleic Acids Research (2019), 10.1093/nar/gky1079.

[10] S. R. Engel, F. S. Dietrich, D. G. Fisk, G. Binkley, R. Balakrishnan, M. C. Costanzo, S. S. Dwight, B. C. Hitz, K. Karra, R. S. Nash, S. Weng, E. D. Wong, P. Lloyd, M. S. Skrzypek, S. R. Miyasato, M. Simison, and J. M. Cherry, *The Reference Genome Sequence of Saccharomyces cerevisiae: Then and Now,* G3: Genes, Genomes, Genetics (2014), 10.1534/g3.113.008995.

[11] K. Luck *et al.*, *A reference map of the human protein interactome,* bioRxiv (2019), 10.1101/605451.

[12] D. Szklarczyk, A. L. Gable, D. Lyon, A. Junge, S. Wyder, J. Huerta-Cepas, M. Simonovic, N. T. Doncheva, J. H. Morris, P. Bork, L. J. Jensen, and C. Mering, *STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets,* Nucleic Acids Research **47**, D607 (2018), https://academic.oup.com/nar/article-pdf/47/D1/D607/27437323/gky1131.pdf .

[13] T. Sun, B. Zhou, L. Lai, and J. Pei, *Sequence-based prediction of protein protein interaction using a deep-learning algorithm,* BMC Bioinformatics **18** (2017), 10.1186/s12859-017-1700-2.

[14] S. Hashemifar, B. Neyshabur, A. A. Khan, and J. Xu, *Predicting protein-protein interactions through sequence-based deep learning,* in *Bioinformatics*, Vol. 34 (2018) pp. i802–i810.

[15] M. Chen, C. J. Ju, G. Zhou, X. Chen, T. Zhang, K. W. Chang, C. Zaniolo, and W. Wang, *Multifaceted protein-protein interaction prediction based on Siamese residual RCNN,* in *Bioinformatics* (2019).

[16] F. Richoux, C. Servantie, C. Borès, and S. Téletchéa, *Comparing two deep learning sequence-based models for protein-protein interaction prediction,* arxiv (2019), arXiv:1901.06268 .

[17] C. von Mering, L. J. Jensen, B. Snel, S. D. Hooper, M. Krupp, M. Foglierini, N. Jouffre, M. A. Huynen, and P. Bork, *STRING: Known and predicted protein-protein associations, integrated and transferred across organisms,* Nucleic Acids Research (2005), 10.1093/nar/gki005.

[18] R. P. Huntley, T. Sawford, P. Mutowo-Meullenet, A. Shypitsyna, C. Bonilla, M. J. Martin, and C. O'Donovan, *The GOA database: Gene Ontology annotation updates for 2015,* Nucleic Acids Research **43**, D1057 (2015).

[19] M. H. Serres, S. Gopal, L. A. Nahum, P. Liang, T. Gaasterland, and M. Riley, *A functional update of the Escherichia coli K-12 genome,* Genome Biol. **2**, RESEARCH0035 (2001).

[20] P. Lamesch, T. Z. Berardini, D. Li, D. Swarbreck, C. Wilks, R. Sasidharan, R. Muller, K. Dreher, D. L. Alexander, M. Garcia-Hernandez, A. S. Karthikeyan, C. H. Lee, W. D. Nelson, L. Ploetz, S. Singh, A. Wensel, and E. Huala, *The Arabidopsis Information Resource (TAIR): Improved gene annotation and new tools,* Nucleic Acids Research (2012), 10.1093/nar/gkr1090.

[21] B. V. Suresh, R. Roy, K. Sahu, G. Misra, and D. Chattopadhyay, *Tomato genomic resources database: An integrated repository of useful tomato genomic information for basic and applied research,* PLoS ONE (2014), 10.1371/journal.pone.0086387.

[22] A. Grover and J. Leskovec, *node2vec: Scalable feature learning for networks,* (2016), arXiv:1607.00653 [cs.SI] .

[23] R. El-Yaniv, Y. Geifman, and Y. Wiener, *The Prediction Advantage: A Universally Meaningful Performance Measure for Classification and Regression,* arxiv (2017), arXiv:1705.08499 .

[24] D. Piovesan and S. C. E. Tosatto, *INGA 2.0: improving protein function prediction for the dark proteome,* Nucleic Acids Research (2019), 10.1093/nar/gkz375.

[25] S. Bhagat, G. Cormode, and S. Muthukrishnan, *Node classification in social networks,* Social Network Data Analytics , 115–148 (2011).

[26] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks,* (2016), arXiv:1609.02907 [cs.LG] .

[27] B. Pandey, P. K. Bhanodia, A. Khamparia, and D. K. Pandey, *A comprehensive survey of edge prediction in social networks: Techniques, parameters and challenges,* Expert Systems with Applications **124**, 164 (2019).

[28] G. Crichton, Y. Guo, S. Pyysalo, and A. Korhonen, *Neural networks for link prediction in realistic biomedical graphs: a multi-dimensional evaluation of graph embedding-based approaches,* BMC Bioinformatics **19**, 176 (2018).

[29] A. Valencia and F. Pazos, *Computational methods for the prediction of protein interactions,* Current Opinion in Structural Biology **12**, 368 (2002).

[30] R. Jansen, H. Yu, D. Greenbaum, Y. Kluger, N. J. Krogan, S. Chung, A. Emili, M. Snyder, J. F. Greenblatt, and M. Gerstein, *A Bayesian Networks Approach for Predicting Protein-Protein Interactions from Genomic Data,* Science (2003), 10.1126/science.1087361.

[31] H. Cho, B. Berger, and J. Peng, *Compact Integration of Multi-Network Topology for Functional Analysis of Genes,* Cell Syst **3**, 540 (2016).

[32] V. Gligorijevic, M. Barot, and R. Bonneau, *deepNF: deep network fusion for protein function prediction,* Bioinformatics **34**, 3873 (2018).

[33] M. Dallachiesa, C. Aggarwal, and T. Palpanas, *Node classification in uncertain graphs,* in *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*, SSDBM '14 (Association for Computing Machinery, New York, NY, USA, 2014).

[34] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, *K-nearest neighbors in uncertain graphs,* Proc. VLDB Endow. **3**, 997–1008 (2010).

[35] T. De Bie, L. C. Tranchevent, L. M. van Oeffelen, and Y. Moreau, *Kernel-based data fusion for gene prioritization,* in *Bioinformatics* (2007).

[36] S. Jaeger, S. Gaudan, U. Leser, and D. Rebholz-Schuhmann, *Integrating protein-protein interactions and text mining for protein function prediction,* in *BMC Bioinformatics* (2008).

[37] R. You, X. Huang, and S. Zhu, *Deeptext2go: Improving large-scale protein function prediction with deep semantic text representation,* Methods **145**, 82 (2018), data mining methods for analyzing biological data in terms of phenotypes.

[38] W. T. Clark and P. Radivojac, *Information-theoretic evaluation of predicted ontological annotations,* Bioinformatics **29**, i53 (2013).

[39] I. Plyusnin, L. Holm, and P. Törönen, *Novel comparison of evaluation metrics for gene ontology classifiers reveals drastic performance differences,* PLoS Comput. Biol. **15**, e1007419 (2019).

5.6. SUPPLEMENTARY MATERIAL

5.6.1. STATISTICS OF PPI NETWORKS

The experimental PPI networks of the four species have very different degree distributions, as shown in Figure 5.S1. Table 5.S1 lists the number of total and unique protein-protein interactions (PPIs) that each data source includes for *Saccharomyces cerevisae, Escherichia coli, Arabidoisis thaliana* and *Solanum lycopersicum.*

5.6.2. INTEGRATION OF STRING SCORES

For completeness, we describe the algorithm for integrating STRING scores from D different sources. We use s_i to denote the score (probability) of a particular interaction based on data type i. This score is computed by taking into account the prior probability of two proteins interacting (p), for which the value $p = 0.041$ is used. When combining multiple data types, this prior score should be only be incorporated once, so the first step is to remove it from all data types (Eq 5.6):

$$s_{i,noprior} = \frac{s_i - p}{1 - p} \tag{5.6}$$

Scores from different sources are combined using Eq 5.7:

$$s_{combined,noprior} = 1 - \prod_{i=1}^{D} (1 - s_{i,noprior}) \tag{5.7}$$

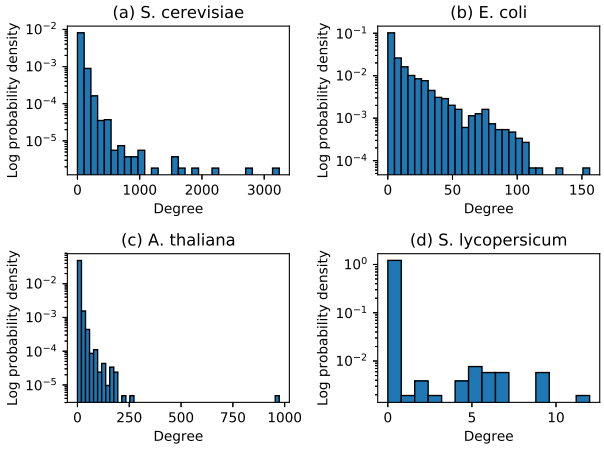


Figure 5.S1: Histogram of node degrees in the experimental PPI networks of yeast (a), *E. coli* (b), arabidopsis (c) and tomato (d). On the *x*-axis node degree and on the *y*-axis the probability density in log scale.

Table 5.S1: Number of interactions added by each data source (rows) in each species (columns). "Database" from STRING was not considered, as it includes protein associations based on known GO annotations.

Data source	*S. cerevisiae*		*E. coli*		*A. thaliana*		*S. lycopersicum*	
	Total	Unique	Total	Unique	Total	Unique	Total	Unique
Physical (BIOGRID)	97,503	51,473	9,734	5,295	21,720	10,826	57	11
Experiments (STRING)	87,378	24,485	11,269	3,944	10,176	746	0	0
Neighborhood (STRING)	0	0	3,644	638	0	0	0	0
Neighborhood transferred (STRING)	55,410	30,241	49,167	27,128	165,627	106,821	117	57
Co-Occurence (STRING)	2,217	498	43,114	29,663	21,283	12,968	1,014	187
Database transferred (STRING)	0	0	0	0	0	0	0	0
Experiments transferred (STRING)	121,858	57,464	13,356	6,175	351,488	236,893	2,878	1,247
Fusion (STRING)	1,784	1,007	1,568	224	2,010	671	2	0
Homology (STRING)	3,700	561	1,083	414	31,983	16,734	1,858	212
Co-Expression (STRING)	71,525	24,635	0	0	285,075	217,555	0	0
Co-Expression transferred (STRING)	242,228	100,849	47,897	21,878	390,727	193,802	3,029	622
Text mining (STRING)	206,721	122,099	7,609	4,798	222,844	171,286	904	595
Text mining transferred (STRING)	211,409	95,945	56,745	30,854	497,495	313,994	3,712	1,412

Finally, the prior probability is incorporated back into the score by Eq 5.8, which is the same as Eq 5.6, but solving for a different variable.

$$s_{combined} = s_{combined,noprior} + p(1 - s_{combined,noprior}) \tag{5.8}$$

5.6.3. EVALUATION MEASURES AND RESULTS

DEFINITIONS

We evaluated using the protein-centric F1 score and Semantic Distance (SD). The definitions of these metrics can be found in Chapter 1.

RESULTS

The performance of all tested methods is listed in Tables 5.S2 and 5.S3.

Table 5.S2: F_{max} of different function prediction methods (rows) in 4 species (columns) estimated using 5-fold cross-validation. The mean ±the standard deviation is shown.

	S. cerevisiae	*E. coli*	*A. thaliana*	*S. lycopersicum*
Naive	0.31±0.004	0.29±0.009	0.28±0.005	0.23±0.019
BLAST	0.35±0.004	0.43±0.021	0.42±0.007	0.34±0.025
EXP, GBA	0.42±0.007	0.25±0.011	0.19±0.007	0.08±0.007
EXP, node2vec	0.50±0.012	0.28±0.011	0.23±0.008	0.08±0.042
EXP, GBA + BLAST	0.50±0.007	0.45±0.019	0.42±0.006	0.33±0.025
EXP + STRING, GBA	0.49±0.005	0.46±0.008	0.48±0.004	0.61±0.045
EXP + STRING, node2vec	0.59±0.009	0.50±0.012	0.50±0.005	0.61±0.042
EXP + STRING, GBA + BLAST	0.54±0.006	0.58±0.021	0.54±0.008	0.60±0.047
EXP + SEQ, GBA	0.33±0.005	0.32±0.008	0.29±0.004	0.33±0.016

Table 5.S3: S_{min} of different function prediction methods (rows) in 4 species (columns) estimated using 5-fold cross-validation. The mean ±the standard deviation is shown.

	S. cerevisiae	*E. coli*	*A. thaliana*	*S. lycopersicum*
Naive	34.67±0.75	21.41±0.20	30.44±0.57	17.67±0.77
BLAST	35.24±0.97	16.85±0.85	27.66±0.31	19.31±0.87
EXP, GBA	31.17±0.55	21.58±0.44	30.55±0.61	19.39±0.91
EXP, node2vec	30.05±0.55	20.37±0.41	28.69±0.59	19.37±0.89
EXP, GBA + BLAST	29.28±0.83	15.57±0.79	27.75±0.55	18.25±0.92
EXP + STRING, GBA	27.42±0.57	17.07±0.22	24.12±0.50	9.08±0.89
EXP + STRING, node2vec	25.65±0.48	17.23±0.34	25.10±0.15	9.20±0.77
EXP + STRING, GBA + BLAST	28.48±0.97	12.94±0.60	23.00±0.56	10.21±1.06
EXP + SEQ, GBA	34.50±0.74	21.02±0.14	30.48±0.57	17.06±0.71

Table 5.S4: F_{max}, S_{min} and coverage of the *node2vec* method on the EXP PPI network in four species when excluding and including proteins without any functional annotations during the feature learning step. Where one of the two approaches is significantly (FDR < 5%) better, it is shown in bold.

Species	F_{max}		S_{min}		Coverage	
	Annotated proteins only	All proteins	Annotated proteins only	All proteins	Annotated proteins only	All proteins
S. cerevisiae	0.50 ± 0.012	0.50 ± 0.010	30.05 ± 0.54	30.13 ± 0.32	0.99 ± 0.002	0.99 ± 0.001
E. coli	0.28 ± 0.011	0.29 ± 0.008	20.37 ± 0.41	20.47 ± 0.31	0.77 ± 0.016	$\mathbf{0.79 \pm 0.017}$
A. thaliana	0.23 ± 0.008	0.23 ± 0.009	28.69 ± 0.49	28.94 ± 0.49	0.57 ± 0.010	$\mathbf{0.58 \pm 0.011}$
S. lycopersicum	0.08 ± 0.007	0.08 ± 0.007	19.37 ± 0.89	19.36 ± 0.89	0.03 ± 0.017	0.03 ± 0.017

COMPARISON OF GBA AND *node2vec* PER PROTEIN

Table 5.S5: P-values before and after multiple testing correction to assess the significance in the median per-protein performances of GBA and *node2vec* at different levels of node degree. The bin numbers correspond to the bins shown in Figure 3 in the main document, with bin 1 denoting the smallest number of neighbors. For species where fewer bins were used, a dash signifies unused bins.

Bin number	*S. cerevisiae*		*E. coli*		*A. thaliana*	
	raw	corrected	raw	corrected	raw	corrected
1	0.028	0.036	7.5×10^{-11}	1.3×10^{-10}	$< 10^{-20}$	$< 10^{-20}$
2	1.6×10^{-5}	2.9×10^{-5}	1.8×10^{-11}	9.0×10^{-11}	$< 10^{-20}$	$< 10^{-20}$
3	0.845	0.845	7.6×10^{-11}	1.3×10^{-10}	$< 10^{-20}$	$< 10^{-20}$
4	0.004	0.006	0.031	0.031	4.4×10^{-16}	5.5×10^{-16}
5	7.4×10^{-6}	1.7×10^{-5}	1.9×10^{-5}	2.4×10^{-5}	0.001	0.001
6	2.5×10^{-21}	2.2×10^{-20}	-	-	-	-
7	5.5×10^{-12}	2.5×10^{-11}	-	-	-	-
8	5.5×10^{-6}	1.6×10^{-5}	-	-	-	-
9	0.109	0.123	-	-	-	-

5.6.4. *node2vec* PARAMETERS AND TUNING

The *node2vec* algorithm has several parameters that can have great influence on the final node embeddings and therefore the function prediction performance. For each species, we tuned these parameters for the experimental network and for the best combination of experimental and STRING edges separately. To do so, we employed a grid search. Below is a list of the parameters we tuned and the ranges of parameter values we considered:

- The number of random walks per node ([5, 10, 15, 20, 30]),
- The parameter p ([0.1, 0.5, 1.0, 1.5, 3.0]),
- The parameter q ([0.1, 0.5, 1.0, 1.5, 3.0]),
- The length of each random walk ([20, 40, 60, 80, 100, 160]), and
- The dimensionality of the embedding vector ([32, 64, 96, 128, 150, 200, 500]).

For each of the combinations above, we tried the kNN and ridge classifiers. The kNN also has one tunable parameter, the number of neighbors k, for which we considered the values [1, 2, 5, 7, 9, 11, 15, 21, 31, 51]. The parameter λ of ridge, which controls the amount of L2 regularization, was tuned among the values [0.0001, 0.001, 0.01, 0.1, 0.5, 1.0, 2.0, 3.0, 5.0, 10.0].

This amounts to 105,000 classification models per network per species. Each model was evaluated in each of the five cross-validation folds, by further subdividing the training set of each fold into a training (80% of original training set) and a validation set (20% of originial training set). The combination of *node2vec* hyperparameters, classifier and classifier parameter that maximized the validation F_{max} in each fold was subsequently trained on the entire training set (of the corresponding fold) and the resulting trained model was tested on the previously unseen test set.

5.6.5. REMOVING EDGES FROM THE YEAST EXP PPI NETWORK

We started from the full *EXP* PPI network of yeast and step-wise removed 10%, 20%, 30%, ..., 90% and 99% of its edges. The edges were removed at random, using two different strategies: In the first ("uniform"), the probability of removing an edge was uniform over all present edges. In the second ("degree"), it was inversely proportional to the smaller degree of the two nodes that the edge connects, so that nodes with the fewest edges are most likely to lose them. This procedure gave us 11 downsampled networks per sampling strategy on which we ran the GBA classifier, using the same cross-validation loop as before. We repeated this 5 times with different random seeds to obtain variability estimates. The performance is plotted against the amount of missing edges in Figure 5.S2.

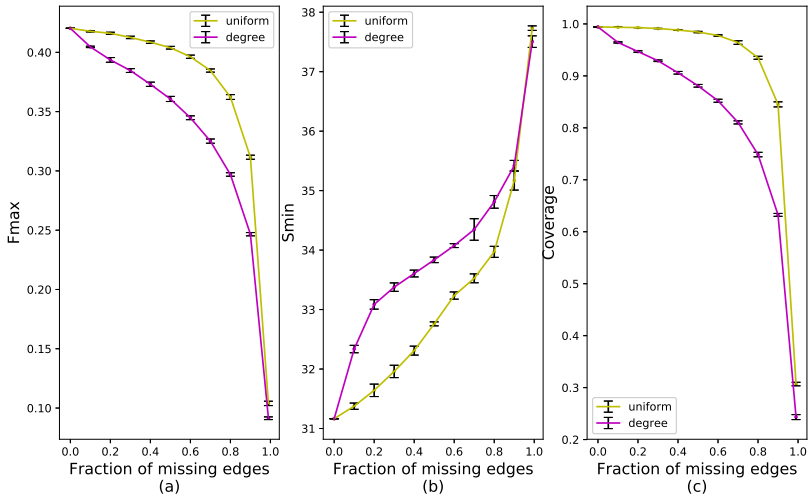


Figure 5.S2: F_{max} (a), S_{min} (b) and coverage (c) of the GBA method on the yeast experimental PPI network (*y*-axis) as a function of the fraction of missing edges (*x*-axis). Edges where removed at random either uniformly (yellow) or inversely proportional to node degree, so that least connected nodes are more likely to lose their edges (purple). Error bars show standard deviation of 5 rounds of random sampling.

5.6.6. USING ALL STRING EDGES IN A WEIGHTED GRAPH

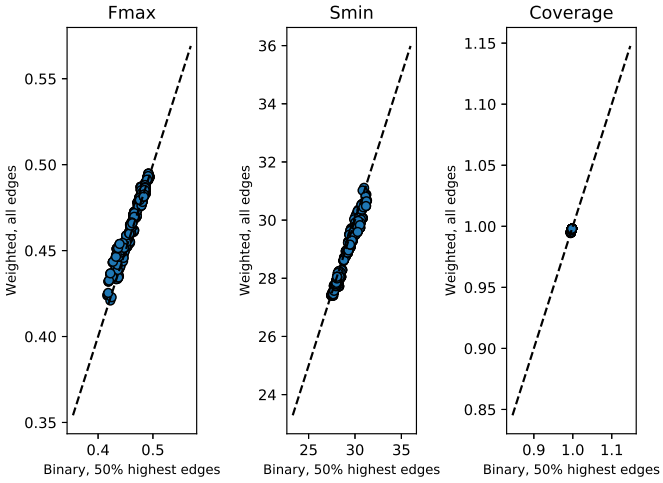


Figure 5.S3: F_{max} (left), S_{min} (middle) and coverage (right) of the binary STRING networks with the 50% top edges (x-axis) against the weighted ones with all edges (y-axis) for *S. cerevisiae*. Each dot corresponds to one of the 511 combinations of the 9 STRING data sources added to the binary experimental network. The black dashed line shows the $y = x$ line to ease comparison.

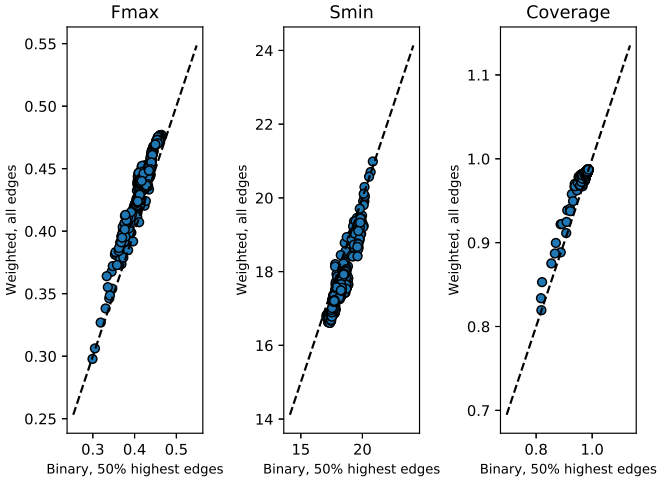


Figure 5.S4: F_{max} (left), S_{min} (middle) and coverage (right) of the binary STRING networks with the 50% top edges (x-axis) against the weighted ones with all edges (y-axis) for *E. coli*. Each dot corresponds to one of the 511 combinations of the 9 STRING data sources added to the binary experimental network. The black dashed line shows the $y = x$ line to ease comparison.

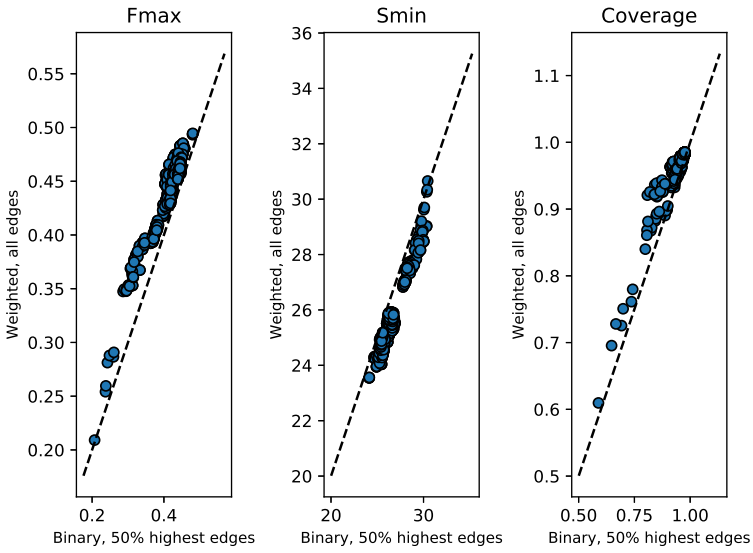


Figure 5.S5: F_{max} (left), S_{min} (middle) and coverage (right) of the binary STRING networks with the 50% top edges (x-axis) against the weighted ones with all edges (y-axis) for *A. thaliana*. Each dot corresponds to one of the 511 combinations of the 9 STRING data sources added to the binary experimental network. The black dashed line shows the $y = x$ line to ease comparison.

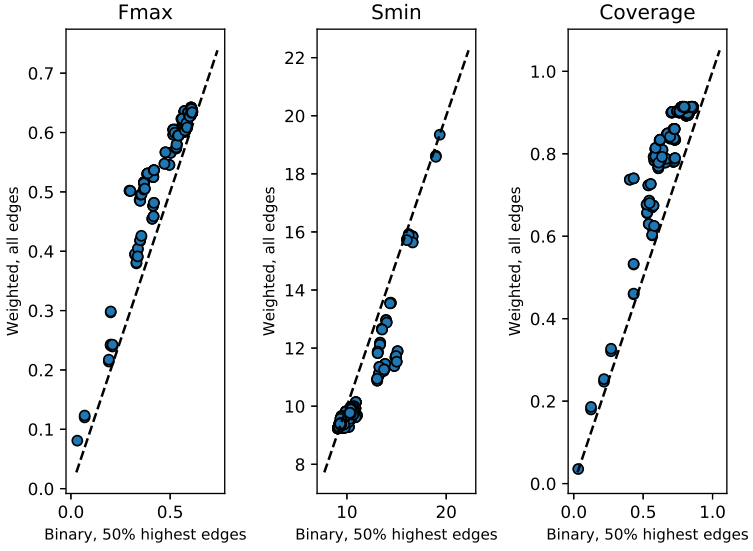


Figure 5.S6: F_{max} (left), S_{min} (middle) and coverage (right) of the binary STRING networks with the 50% top edges (x-axis) against the weighted ones with all edges (y-axis) for *S. lycopersicum*. Each dot corresponds to one of the 255 combinations of the 8 STRING data sources added to the binary experimental network. The black dashed line shows the $y = x$ line to ease comparison.

Table 5.S6: F_{max}, S_{min} and coverage of the *GBA* method on the *EXP+STRING* PPI network in four species when using a binary network with the top 50% STRING edges or a weighted one with all STRING edges. The performance of the best combination for each metric and each species is shown. Statistically significant differences (paired t-test, FDR < 0.05) are shown in bold.

Species	F_{max}		S_{min}		Coverage	
	Binary	Weighted	Binary	Weighted	Binary	Weighted
S. cerevisiae	0.49 ± 0.005	0.49 ± 0.005	27.43 ± 0.57	27.40 ± 0.61	0.99 ± 0.001	0.99 ± 0.001
E. coli	0.46 ± 0.008	$\mathbf{0.48 \pm 0.009}$	17.07 ± 0.22	16.61 ± 0.35	0.99 ± 0.004	0.99 ± 0.003
A. thaliana	0.48 ± 0.004	$\mathbf{0.49 \pm 0.005}$	24.12 ± 0.50	$\mathbf{23.54 \pm 0.50}$	0.98 ± 0.003	$\mathbf{0.99 \pm 0.001}$
S. lycopersicum	0.61 ± 0.045	0.64 ± 0.043	9.08 ± 0.89	9.23 ± 1.00	0.86 ± 0.020	$\mathbf{0.91 \pm 0.018}$

5.6.7. STRING PERFORMANCE PER DATA SOURCE

For every species, we tested all possible combinations of *STRING* data sources and found that certain combinations perform significantly better than others, as certain individual data sources were more informative (Fig 5.S7-5.S9.). To quantify the contribution of each data source we devised a statistical test described below.

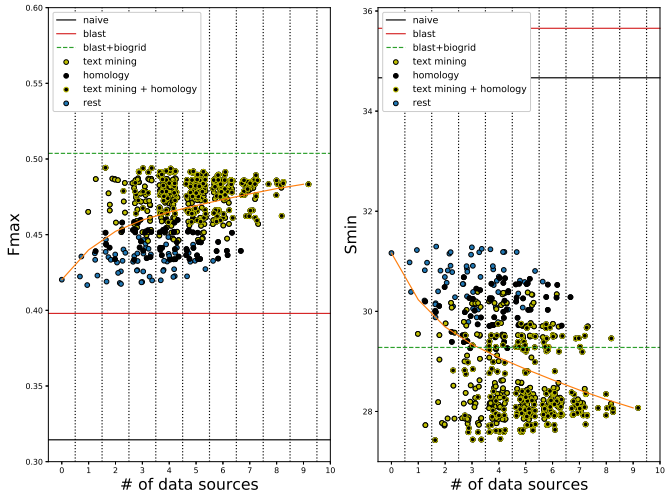


Figure 5.S7: F_{max} (left) and S_{min} (right) in *S. cerevisiae* (*y*-axis) as a function of the number of *STRING* data sources included (*x*-axis). Each dot corresponds to one combination of data sources added to the experimental network. Combinations that include "text mining" and/or "text mining transferred" are shown in yellow, combinations that include "homology" in black and combinations that include both in black with yellow border. The rest of the combinations are shown in blue. To ease visibility, we added a random number in the range [-0.5, 0.5] to each combination of the same number of sources. Zero data sources corresponds to the *EXP* network and the orange line shows the average performance for a specific number of data sources. Horizontal lines denote the performance of the *naive* (black), *BLAST* (red) and the combination of *BLAST* with the *EXP* PPI network (dashed green).

For a species for which *STRING* contains protein associations from N different data sources, there are $2^N - 1$ ways to combine them (in combinations of different lengths). Each data source i is present in 2^{N-1} of these combinations (denoted as P_i) and absent in the remaining ones (denoted as A_i). We define an "irreplaceability" score ($IR(i)$) for

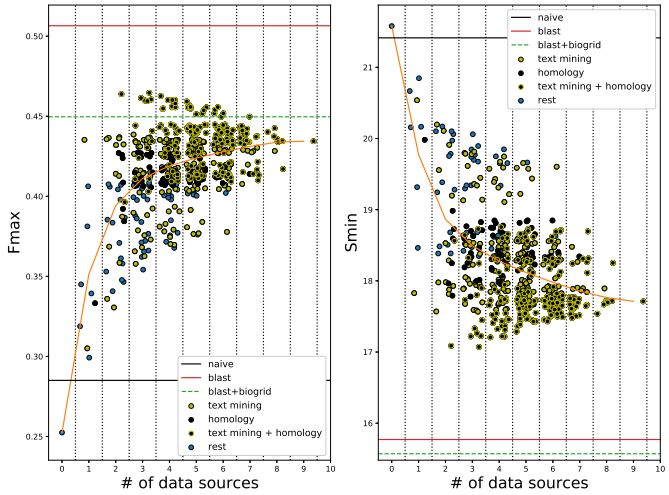


Figure 5.S8: F_{max} (left) and S_{min} (right) in *E. coli* (*y*-axis) as a function of the number of *STRING* data sources included (*x*-axis). Each dot corresponds to one combination of data sources added to the experimental network. Combinations that include "text mining" and/or "text mining transferred" are shown in yellow, combinations that include "homology" in black and combinations that include both in black with yellow border. The rest of the combinations are shown in blue. To ease visibility, we added a random number in the range [-0.5, 0.5] to each combination of the same number of sources. Zero data sources corresponds to the *EXP* network and the orange line shows the average performance for a specific number of data sources. Horizontal lines denote the performance of the *naive* (black), *BLAST* (red) and the combination of *BLAST* with the *EXP* PPI network (dashed green).

i as the increase in the optimal performance of P_i with respect to A_i. For F_{max}, this is formally defined in equation 5.9

$$IR_F(i) = \max_{c \in P_i} F_{max}(c) - \max_{c \in A_i} F_{max}(c) \tag{5.9}$$

An $IR_F(i)$ equal to 0, means that the combination that achieves the maximum performance does not include data source i, which implies that it made redundant by other sources. A positive value for $IR_F(i)$ means that i is necessary for maximizing *STRING* performance and a negative value that i is creating more errors than correct predictions. Since lower S_{min} means better performance, the definition of IR for S_{min} is slightly modified, so that a positive IR still denotes that a data source is useful (equation 5.10).

$$IR_S(i) = \min_{c \in A_i} S_{min}(c) - \min_{c \in P_i} S_{min}(c) \tag{5.10}$$

We compared the observed IR_F and IR_S values to what would be expected by chance to obtain a measure of statistical significance. To do so, we did a permutation test for which we randomly mixed the elements of P_i and A_i and calculated the IR_F and IR_S values again. We repeated this 10,000 times for each data source to obtain a null distribution of the two statistics and counted the fraction of times the permuted statistic was larger than the observed one. Note that this is a one-sided test, meaning that we

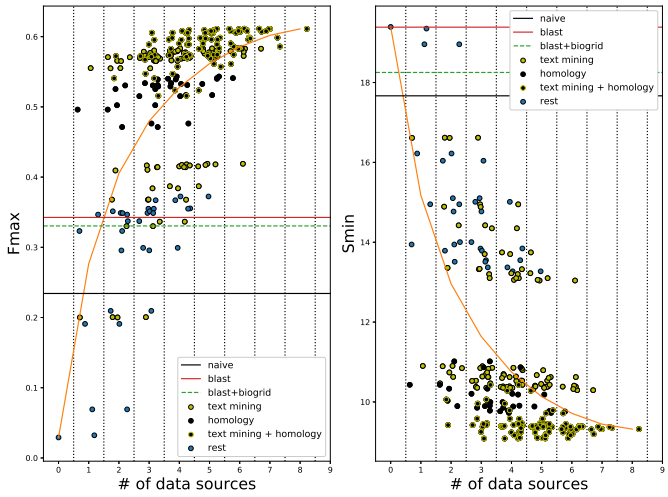


Figure 5.S9: F_{max} (left) and S_{min} (right) in *S. lycopersicum* (*y*-axis) as a function of the number of *STRING* data sources included (*x*-axis). Each dot corresponds to one combination of data sources added to the experimental network. Combinations that include "text mining" and/or "text mining transferred" are shown in yellow, combinations that include "homology" in black and combinations that include both in black with yellow border. The rest of the combinations are shown in blue. To ease visibility, we added a random number in the range [-0.5, 0.5] to each combination of the same number of sources. Zero data sources corresponds to the *EXP* network and the orange line shows the average performance for a specific number of data sources. Horizontal lines denote the performance of the *naive* (black), *BLAST* (red) and the combination of *BLAST* with the *EXP* PPI network (dashed green).

only tested whether a data source was significantly irreplaceable, but not whether it was significantly leading to worse performance. We performed 90 tests in total, so we also applied multiple testing correction using the FDR method. The observed test statistics and the corresponding p-values are listed in Tables 5.S7-5.S8 for yeast, 5.S9-5.S10 for E. coli, 5.S11-5.S12 for arabidopsis and 5.S13-5.S14 for tomato.

Table 5.S7: IR_F scores of STRING data sources, corresponding raw p-values and p-values corrected for multiple tests for *S. cerevisiae*.

Data source	IR_F	p-value	corrected p-value
coexpression	-0.0026	0.9989	1.0
neighborhood transferred	-0.0024	0.9933	1.0
coexpression transferred	-0.0081	1.0	1.0
experiments transferred	-0.0078	1.0	1.0
textmining	0.0154	$< 10^{-4}$	< 0.007
cooccurence	-0.0004	0.8738900	1.0
fusion	-0.0002	0.75490	1.0
textmining transferred	-0.0022	0.9715	1.0
homology	0.0007	$< 10^{-4}$	< 0.007

Table 5.S8: IR_S scores of STRING data sources, corresponding raw p-values and p-values corrected for multiple tests for *S. cerevisiae*.

Data source	IR_S	p-value	corrected p-value
coexpression	-0.1572	0.9832	1.0
neighborhood transferred	-0.1571	0.9707	1.0
coexpression transferred	-0.4923	1.0	1.0
experiments transferred	-0.4663	1.0	1.0
textmining	1.3206	$< 10^{-4}$	< 0.007
cooccurence	-0.0105	0.7568	1.0
fusion	-0.0111	0.8798	1.0
textmining transferred	-0.1867	1.0	1.0
homology	0.2914	$< 10^{-4}$	< 0.007

Table 5.S9: IR_F scores of STRING data sources, corresponding raw p-values and p-values corrected for multiple tests for *E. coli*.

Data source	IR_F	p-value	corrected p-value
neighborhood	-0.0042	0.9695	1.0
neighborhood transferred	-0.0198	1.0	1.0
coexpression transferred	-0.0046	0.9962	1.0
experiments transferred	-0.0021	0.8759	1.0
textmining	-0.0045	0.9919	1.0
cooccurence	-0.0232	1.0	1.0
fusion	0.0008	0.4949	1.0
textmining transferred	0.0303	$< 10^{-4}$	< 0.007
homology	0.0261	$< 10^{-4}$	< 0.007

Table 5.S10: IR_S scores of STRING data sources, corresponding raw p-values and p-values corrected for multiple tests for *E. coli*.

Data source	IR_S	p-value	corrected p-value
neighborhood	-0.1456	0.9418	1.0
neighborhood transferred	-0.5245	1.0	1.0
coexpression transferred	-0.2496	0.9996	1.0
experiments transferred	-0.1348	0.8769	1.0
textmining	-0.1483	0.9839	1.0
cooccurence	-0.3763	1.0	1.0
fusion	0.016	0.4938	1.0
textmining transferred	0.6704	$< 10^{-4}$	< 0.007
homology	0.4951	$< 10^{-4}$	< 0.007

Table 5.S11: IR_F scores of STRING data sources, corresponding raw p-values and p-values corrected for multiple tests for *A. thaliana*.

Data source	IR_F	p-value	corrected p-value
coexpression	-0.0297	1.0	1.0
neighborhood transferred	-0.0225	0.96600	1.0
coexpression transferred	-0.0324	1.0	1.0
experiments transferred	-0.0275	1.0	1.0
textmining	0.0258	0.00010	0.0052
cooccurence	0.0001	0.50100	1.0
fusion	0.0002	0.25070	1.0
textmining transferred	-0.0252	0.9982	1.0
homology	0.0534	$< 10^{-4}$	< 0.007

Table 5.S12: IR_S scores of STRING data sources, corresponding raw p-values and p-values corrected for multiple tests for *A. thaliana*.

Data source	IR_S	p-value	corrected p-value
coexpression	-0.9484	1.0	1.0
neighborhood transferred	-0.7883	0.9984	1.0
coexpression transferred	-1.2361	1.0	1.0
experiments transferred	-1.1204	1.0	1.0
textmining	0.5247	0.0659	1.0
cooccurence	-0.0069	0.8801	1.0
fusion	-0.0066	0.7477	1.0
textmining transferred	-1.0243	1.0	1.0
homology	1.9112	$< 10^{-4}$	< 0.007

Table 5.S13: IR_F scores of STRING data sources, corresponding raw p-values and p-values corrected for multiple tests for tomato.

Data source	IR_F	p-value	corrected p-value
neighborhood transferred	-0.0002	0.98540	1.0
coexpression transferred	0.0019	0.00380	0.1938
experiments transferred	0.0031	$< 10^{-4}$	< 0.007
textmining	0.0128	$< 10^{-4}$	< 0.007
cooccurence	0.0002	0.24900	1.0
fusion	0.0000	0.75380	1.0
textmining transferred	0.0254	$< 10^{-4}$	< 0.007
homology	0.0281	$< 10^{-4}$	< 0.007

Table 5.S14: IR_S scores of STRING data sources, corresponding raw p-values and p-values corrected for multiple tests for tomato.

Data source	IR_S	p-value	corrected p-value
neighborhood transferred	-0.0029	0.93930	1.0
coexpression transferred	-0.0812	1.0	1.0
experiments transferred	-0.2280	1.0	1.0
textmining	0.1536	$< 10^{-4}$	< 0.007
cooccurence	-0.0085	0.9875	1.0
fusion	0.0000	0.75470	1.0
textmining transferred	0.2691	$< 10^{-4}$	< 0.007
homology	1.2114	$< 10^{-4}$	< 0.007

REMOVING THE MOST INFORMATIVE DATA SOURCES

Table 5.S15: Changes in F_{max} of the combined *EXP+STRING* network, when removing text mining and/or homology edges.

Data sources	*S. cerevisiae*	*E. coli*	*A. thaliana*	*S. lycopersicum*
All	0.49±0.005	0.46±0.008	0.48±0.004	0.61±0.045
No text mining/text mining transferred	0.46±0.006	0.43±0.013	0.45±0.007	0.54±0.053
No homology	0.49±0.004	0.44±0.011	0.43±0.006	0.58±0.044
No text mining/text mining transferred/homology	0.45±0.006	0.41±0.010	0.35±0.004	0.37±0.042

Table 5.S16: Changes in S_{min} of the combined *EXP+STRING* network, when removing text mining and/or homology edges.

Data sources	*S. cerevisiae*	*E. coli*	*A. thaliana*	*S. lycopersicum*
All	27.42±0.57	17.07±0.22	24.12±0.50	9.08±0.89
No text mining/text mining transferred	29.24±0.64	17.78±0.43	24.65±0.59	9.72±1.16
No homology	27.72±0.57	17.57±0.26	26.03±0.51	10.30±0.84
No text mining/text mining transferred/homology	29.79±0.63	18.35±0.22	28.47±0.47	13.27±0.65

5.6.8. PIPR TRAINING AND RESULTS

Table 5.S17: Effect of hyperparameters in training of PIPR for predicting yeast PPIs from the BIOGRID database.

Optimizer	**Learning Rate**	**Validation loss**	**Validation accuracy**
ADAM	0.001	0.507	0.771
ADAM	0.0001	0.540	0.754
SGD + learning rate scheduler	0.01	0.525	0.751
SGD + learning rate scheduler	0.001	0.547	0.737

The originally trained PIPR model in yeast generalized poorly in Arabidopsis (accuracy of 51% on a balanced dataset). Therefore, we chose to train PIPR on Arabidopsis by using the original trained model as initial conditions. We trained using Stochastic Gradient Descent with learning rate 0.001 and early stopping based on the validation loss with patience of 40 epochs. As validation set, we randomly selected 10% of the data and as loss function the binary cross-entropy. We did not use RMSprop optimizer, which was used by the authors, as it produced unstable results.

Table 5.S18: Comparison of function prediction performance in arabidopsis and tomato of edges predicted with PIPR trained in either yeast or arabidopsis.

Train species	Test species	F_{max}	S_{min}
S. cerevisiae	*A.thaliana*	0.29 ± 0.004	30.48 ± 0.57
A. thaliana	*A.thaliana*	0.29 ± 0.006	30.16 ± 0.29
S. cerevisiae	*S. lycopersicum*	0.33 ± 0.016	17.05 ± 0.70
A. thaliana	*S.lycopersicum*	0.35 ± 0.017	16.24 ± 0.66

6

Discussion - Automatic Gene Function Prediction in the 2020's

Stavros Makrodimitris, Roeland C.H.J. van Ham and Marcel J.T. Reinders

Parts of this chapter have been published in Genes 2020, 11, 1264 as "Automatic gene function prediction in the 2020's" [1]

ABSTRACT

The current rate at which new DNA and protein sequences are being generated is too fast to experimentally discover the functions of those sequences, emphasizing the need for accurate Automatic Function Prediction (AFP) methods. AFP has been an active and growing research field for decades and has made considerable progress in that time. However, it is certainly not solved. In this paper, we describe challenges that the AFP field still has to overcome in the future to increase its applicability. The challenges we consider are how to: (1) include condition-specific functional annotation, (2) predict functions for non-model species, (3) include new informative data sources, (4) deal with the biases of Gene Ontology (GO) annotations, and (5) maximally exploit the GO to obtain performance gains. We also provide recommendations for addressing those challenges, by adapting 1) the way we represent proteins and genes, 2) the way we represent gene functions, and 3) the algorithms that perform the prediction from gene to function. Together, we show that AFP is still a vibrant research area that can benefit from continuing advances in machine learning with which AFP in the 2020s can again take a large step forward reinforcing the power of computational biology.

6.1. INTRODUCTION

Automatic function prediction (AFP) deals with the algorithmic assignment of functional annotations - usually Gene Ontology (GO) terms - to proteins/genes of unknown function from proteins/genes whose function has already been determined experimentally. In the past two decades, the amount of new protein sequences has been growing at such a fast pace [2] that no experimental screen can keep up, making AFP a necessity for modern biology. In addition to generating fundamental biological knowledge about what proteins do, AFP is crucial for other aspects of research, such as linking genotype to phenotype, by enabling gene set enrichment analyses or facilitating the interpretation of GWAS hits (e.g. [3]). A far from exhaustive list of some of the recent successful AFP models is given in Table 6.1.

Table 6.1: Some of the most important AFP models proposed in the past years.

Name	Reference	Input data	Method
GOLabeler	[4]	Amino acid sequence, GO term frequencies	Learning to rank
FunFams	[5]	Amino acid sequence	Hidden Markov Model
INGA	[6]	Amino acid sequence	Homology search, enrichment analysis
PFP	[7]	Amino acid sequence	Phylogenetics
COFACTOR	[8]	Amino acid sequence, protein structure, protein interactions	Homology search, structural similarity
NetGO	[9]	Amino acid sequence, GO term frequencies, protein interactions	Learning to rank
DeepGOPlus	[10]	Amino acid sequence	Convolutional neural network, homology search

The Critical Assessment of Functional Annotation (CAFA) challenges provide an objective evaluation of modern AFP algorithms on a set of proteins with newly-acquired GO annotations [11–13]. The main finding of these challenges is that methods significantly improved between CAFA1 and CAFA2, but remained rather stagnant in CAFA3, with the exception of one novel method, GOLabeler [4], that outperformed all others, especially in the Molecular Function Ontology (MFO) [13]. Several methods performed rather similarly in the Biological Process Ontology (BPO) and all participating methods failed to outperform a simple co-expression-based baseline method at predicting cell motility and biofilm formation in *Pseudomonas aeruginosa* [13]. Together, these results show that the problem of AFP is far from solved and that perhaps one or several leaps are required to advance the field.

More than a decade ago, in a paper that set the foundation for the CAFA benchmarks, Godzik et al. defined three main challenges for AFP research [14]:

- How to extend AFP beyond homology transfer.
- How to define protein function in a standardized way.
- How to properly evaluate AFP methods.

Since then, all three of these questions have been addressed to varying extents. Several algorithms have been proposed that make use of different data sources, such as sequence features [15], gene expression, and protein-protein interactions [16]. The GO has become the standard vocabulary for describing protein function in the vast majority of AFP models, and the CAFA is a widely-accepted platform to objectively evaluate these models. To inspire the AFP field in realizing new breakthroughs, we have attempted to identify some (new) challenges that we feel are important for advancing the field and we will address them in detail in the next sections. These challenges are:

1. How can we deal with biological function being tissue, cell-type, or condition-specific?
2. How do we predict functions in non-model species?
3. What data sources should be used for predicting function?
4. How does missingness or bias in GO annotations affect the training of AFP models?
5. How can we better exploit the Gene Ontology structure to improve functional annotation?

We will discuss these challenges from three different perspectives of an AFP pipeline (Figure 6.1): a) how proteins are represented, b) how function is encoded, and c) what kind of prediction algorithms one should use. Protein representations refer to the input data types that feed an AFP model, e.g. sequence similarities or raw amino acid sequences. as well as any feature extraction steps. Generally, functions are described using GO terms, but these have limitations or may require adaptations to cope with the challenges identified. Finally, prediction algorithms deal with the mapping between the input protein representation to the target function predictions.

6.2. TISSUE/CONDITION-SPECIFICITY

Simply assigning a set of GO terms to a protein is often not enough. For instance, it does not necessarily provide information about whether the protein performs this function in specific tissues or under specific conditions, which is especially important for the Biological Process Ontology (BPO). For example, from gene expression experiments, it is known that genes can change co-expression partners [17] and regulators [18] from tissue to tissue [17], when under stress [19] or at different developmental stages [20]. Also, tissue-specific protein-protein interactions are known [21]. In other words, although a protein can be involved in multiple biological processes, it doesn't have to execute all these functions at all times, which has two implications. On one hand, it will be difficult to validate predicted annotations when it is not known for which tissues or cell types the protein performs this function. On the other hand, we need to have tissue or cell-type specific information of the activity of the protein (such as mRNA expression levels) to be able to discover these functions. Greene et al. have demonstrated the importance of this issue, by constructing tissue-specific co-functional networks from existing GO annotations and tissue-specific gene expression information, leading to more accurate predictions of response to perturbation and discovery of gene-disease associations [22].

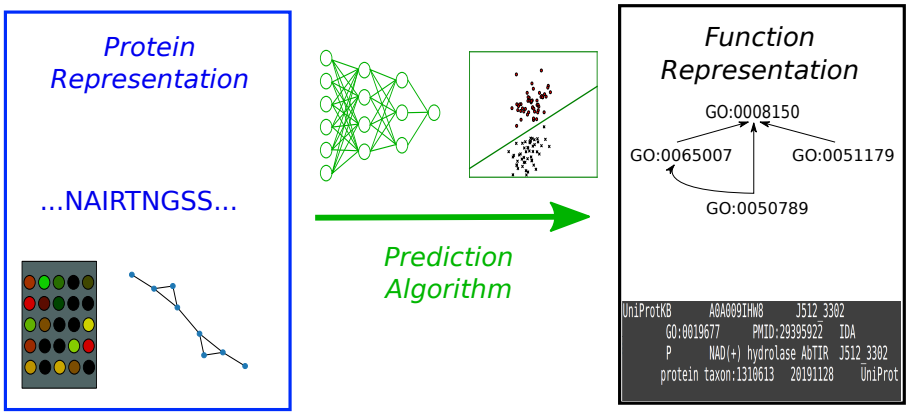


Figure 6.1: AFP algorithms typically consist of a protein representation (protein sequence, expression data, biological networks) (left, in blue), a function representation (often a vector of one-hot encoded GO terms) (right, in black) and a prediction algorithm that connects both (neural networks, support vector machines, Guilt-By-Association methods etc.) (middle, in green).

All of this is made even harder, as, at this point, there is not even a clear definition of a cell type. The Cell Ontology (CO) is a good step towards standardizing cell type definitions, but it is only restricted to animal cells [23]. Despite this, we believe that AFP researchers and curators of the GO need to start preparing for a possible transition to this more specialized AFP phase.

6.2.1. PROTEIN REPRESENTATION

To make protein representation tissue-specific, Zitnik and Leskovec adapted the node2vec method to extract tissue-specific protein embeddings from tissue-specific Protein-Protein Interaction (PPI) networks which were then fed to a linear classifier to predict function [24]. It would be interesting to extend this approach to co-expression networks, which are probably a lot more variable. For cases where co-expression or interaction evolves over time, e.g. for developmental processes or stress responses, it might be helpful to look at dynamic network embedding algorithms, such as dynnode2vec [25] that can learn condition-specific node embeddings without prior knowledge and more efficiently than the approach of [24]. Advances in single-cell sequencing [26] and the generation of cell atlases [27, 28] are expected to elucidate even more subtleties of protein function and thus provide a valuable resource for more fine-grained functional annotation. Being able to represent genes by their expression and/or methylation pattern [26] in millions of cells and not by the average of those quantities, as is done with bulk sequencing, can help us find rare but also specific gene functions. On the other hand, this creates

computational challenges, as one single-cell experiment can nowadays generate data for millions of cells. Integrating data from multiple such experiments will require the use of techniques specialized for processing 'big data'.

6.2.2. FUNCTION REPRESENTATION

Unfortunately, predicting cell-type-specific and/or condition-specific function is a lot harder, as the number of possible outcomes increases in a combinatorial fashion. The number of GO terms is already very large, so creating a separate target variable for each combination of GO term, cell-type and condition would be intractable. Also, this would make the set of existing annotations even sparser posing severe problems for learning algorithms [29]. The Gene Ontology Consortium [30] uses so-called annotation extensions [31] to specify details about an annotation (including that the function occurs at a specific cell type) instead of creating a new GO term for each combination of function and cell type. However, the number of such extensions is also very large, so this does not solve the issue. Perhaps we are in need of a more fundamental representation of BPO functions. For example, for Molecular Function Ontology (MFO) terms, protein domains are often used as clear representatives of function, as specific domains correspond to specific 3D folding patterns that enable specific chemical reactions and therefore can be accurately associated with a molecular function. Certainly, domains can also be associated with BPO terms, but there rarely is a clear causal link. For instance, a DNA-binding domain, might indicate that a protein could be a transcription factor, but that does not provide insights into the genes that this transcription factor regulates or the biological process(es) these genes are involved in. The introduction of Causal Activity Modelling (GO-CAM) [32] tries to address this issue and to unify GO terms by using causal graphs to model their interrelations. Alternatively, markers, such as DNA methylation, chromatin accessibility, and transcription factor binding can be used as a tissue-specific function representation, as they describe a gene's regulation, which might provide information about its functions. The different data types can be integrated with appropriate approaches (e.g. [33]) to identify their common and independent components.

6.2.3. PREDICTION METHODS

When the number of labels increases dramatically, model learning should also be adapted to handle this increase. A promising option would be to move from a discrete to a continuous representation of the conditions. Way and Greene have demonstrated this as a proof of principle by training a Variational Auto-Encoder (VAE) on gene expression data from different cancer types [34]. They then showed that the latent space learned during the unsupervised training contained directions that encoded important information such as gender, tissue of origin and presence or absence of a metastasis [34]. Further work needs to be done on the interpretation of such models, so that we can make use of the latent encoding of different conditions for predicting function. Since VAE's are generative models, they could perhaps also generate predicted gene expression data for combinations of tissues and stresses that are not in the training set. Other generative models, such as Generative Adversarial Networks (GANs) [35] could also be used and specifically conditional GANs [36] are designed to generate data for a specific condition given a (discrete or continuous) numerical representation of that condition. We believe

that this line of research will become popular in the near future. GANs have already been used in AFP to generate artificial data to counter class imbalance, thereby performing data augmentation [37] and leading to increased performance [38]. Incorporating the 'grammar' of GO-CAM relations into prediction models will also be an interesting challenge. For example, genes or proteins can also be viewed as part of the GO-CAM causal graph, thereby transforming the AFP problem into a semi-supervised learning problem of predicting new edges in that graph, specifically edges connecting the genes with the terms.

6.3. GOING BEYOND MODEL SPECIES

The need for accurate AFP is especially pressing for non-model species. Plants are interesting in that regard, as experimentally-derived functions in most plants are either very sparse or non-existent and the huge number of genes per species (e.g. more than 100,000 in wheat [39]) means that genome-wide experimental annotation would require vast amounts of time and resources. On the other hand, a lot of labeled data are required to train, as well as to test an AFP algorithm. Currently, labeled data come mostly from model species, as proteins from ten species account for 86-88% of the experimental GO annotations in UniProtKB, depending on the ontology (Figure 6.2).

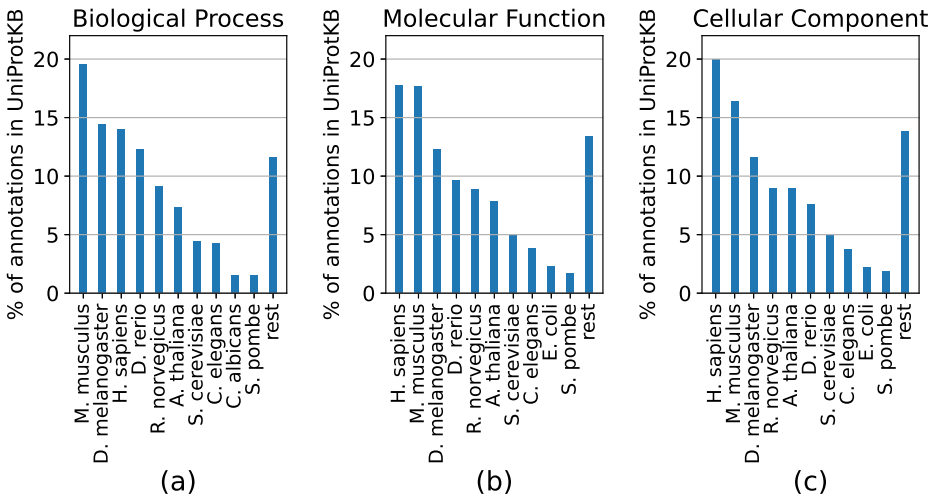


Figure 6.2: Distribution of experimental annotations from the Biological Process (a), Molecular Function (b) and Cellular Component (c) ontologies per species for proteins in UniProtKB. The ten species with the most annotations are shown for each ontology and annotations for all other species are shown in the 'rest' group.

One of the findings of the CAFA challenges [11–13] is that ensemble methods that combine predictions from many data sources tend to perform very well (e.g. MS-kNN [16] in CAFA2; and GOLabeler [4] and INGA [6] in CAFA3). Although CAFA is extremely useful, the evaluations rely on recent experimental annotations and these, by definition, are in their vast majority from 10-15 model species (Figure 6.3), because most experi-

mental biologists work on those species. Besides plants, bacteria and archaea are also largely underrepresented in CAFA benchmarks (Figure 6.3). This focus on model species might hide the fact that perhaps some of the algorithms that are successful in CAFA might not be directly or fully applicable in non-model species, as for newly-sequenced species, typically, only DNA and protein sequences are available. We do by no means attempt to diminish the usefulness and impact of researching multi-omics ensemble methods, but it is important to realize that models that rely on protein-protein interactions or co-expression across different conditions are thus not applicable in the vast majority of species across all domains of life. This is especially important when predicting BPO terms, as these models have been shown to rely more on non-sequence-based data sources [13].

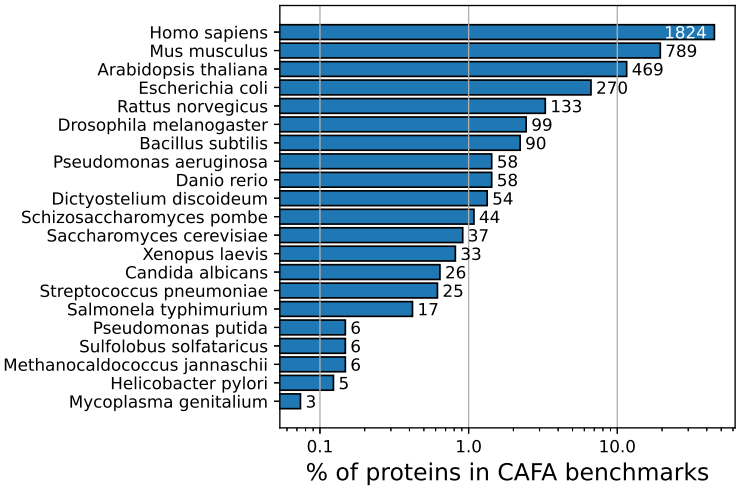


Figure 6.3: Percentage of proteins used in the three CAFA benchmarks [11–13] (x-axis, in log scale) per species. The absolute number of proteins per species is also given next to the bars. Only newly-annotated proteins are included, i.e. proteins that had no GO annotations before the benchmark (referred to as No-Knowledge benchmarks in CAFA).

6.3.1. PROTEIN REPRESENTATION

A big question for AFP is whether sequence-only methods can achieve as high performance as methods that use multi-omic data. Recent work has shown that it is possible to accurately predict PPI's from amino-acid sequence [40] or gene expression from DNA sequence [41], which implies that a big part of the multi-omic data is encoded in the sequence data, albeit in a more complicated manner. In other words, one could improve existing techniques that predict multi-omic data and use those predictions for AFP. But, this might also imply that sequence-only models might have the potential to perform equally well to the ensemble methods. Alternatively, as more and more species

are being sequenced, finding orthologs or constructing high-quality multiple sequence alignments of proteins will get easier and easier. That, in turn, would imply more robust and accurate old-school, sequence-based annotation transfers with ever-increasing coverage. Essential wet-lab experiments could in that scenario be focused on the ever-decreasing set of proteins without close homologues.

6.3.2. FUNCTION REPRESENTATION

Another big challenge of predicting protein functions in other species is that certain functions might be more prevalent or even unique in certain lineages. As an extreme example: it is not trivial to predict photosynthesis-related functions when training only on animal data. It would therefore be beneficial to have a function representation that allows extrapolating to new functions. To do so, we need our representation to be agnostic of the training annotations and capture more general functional aspects. This can be done by learning an embedding for each GO term so that terms that describe related functions have high similarity in the embedding space. That would enable us to extrapolate the meaning of terms beyond the training set. Several such representations have been proposed, one of the first ones being clusDCA [42], which used random walks to learn features that reflect the GO graph topology. More recent approaches make use of advances in Natural Language Processing (NLP) to learn embeddings that reflect semantic meanings based on the term names and/or descriptions [43]. Theoretical work has shown the utility of embedding graph-structured data (as GO terms are) in hyperbolic rather than Euclidean spaces [44].

6.3.3. PREDICTION METHODS

Prediction models that predict across species should mainly deal with two issues: 1) the presence of novel functions, as described above, and 2) the potential differences in distribution of the input data between species. There is a vast machine learning literature on few-shot and zero-shot learning, which deals with classification models that can make predictions for classes for which only very few or even no examples have been seen, respectively [45, 46]. Such methods often tend to use class embeddings [47] and try to leverage prior knowledge on similarities between the classes. A similar approach has been applied in predicting novel cell types from gene expression data using Cell Ontology [23] embeddings [48]. Such approaches can additionally be useful for describing new terms that are occasionally added to the ontology, even before they accumulate many annotations. As for the difference in distributions, also known as domain shift [49], it can lead to large performance loss if not taken into account. As an example, methods that use the frequency of amino acids or amino acid n-grams in a sequence can be sensitive to amino acid frequency differences across lineages (e.g. [50]). Given certain assumptions about the type of domain shift, there exist different approaches for correcting it [49]. Even if theoretical assumptions are not met, however, domain adaptation can still give a performance boost (e.g. [51]), so even then it might still be worth attempting to detect and correct domain shifts.

6.4. OVERLOOKED DATA SOURCES

The amino acid sequence is the most widely-used data source for function prediction, followed by sequence-derived features such as domains as well as other omics data, such as gene expression or protein-protein interactions. There is, however, a wealth of other omics data that has the potential to predict function accurately, but that is currently hardly used. Leveraging these data could also boost performance of AFP algorithms. For the sake of brevity, we focus on three such omics data sources: proteomics, genome proximity, and epigenetic data. In addition, we also address the utility of literature mining.

6.4.1. PROTEIN REPRESENTATION

The power of gene expression data in function prediction has been well-documented in the CAFA challenges, especially for predicting BPO terms [13]. Interestingly, a study from 2017 found that co-expression networks constructed from proteomics rather than transcriptomics were more efficient at predicting both GO terms and pathways from the Kyoto Encyclopedia of Genes and Genomes (KEGG) in *Homo sapiens* [52]. Furthermore, mRNA and proteomics have been shown to contain complementary information [53–55] (Figure 6.4), caused by - amongst others - differences in degradation rates and translation speeds. Hence, protein abundances as measured, for example, by mass spectrometry give a more representative picture of the function of a protein than their mRNA proxies. Integrating these two omics data types could boost prediction accuracy. In addition, the relationship between genes and proteins is not 1-to-1, as RNA splicing [56] as well as post-translational modifications (PTM's) [57] cause the encoding of multiple protein isoforms that have been shown to have different roles and functions [58]. This introduces an additional layer of complexity when trying to predict the function of genes, that can be partly addressed by measuring the expression of isoforms using RNAseq. However, different PTM's can only be measured using proteomics techniques. As the quality, reproducibility, scalability and accessibility of mass spectrometry methods keeps increasing [59], we expect proteomics to start playing a more and more prominent role in AFP.

Proximity between genes has also been shown to be indicative of co-functionality [60]. This is particularly true for bacteria, where genes from the same pathway are often organized in operons, but it was recently shown to be applicable in eukaryotes as well [61]. Except for linear proximity, the same holds for genes that are close in three-dimensional space, as these tend to be co-regulated [62] (Figure 6.4). Chromosome conformation data, generated using e.g. 4C [63] or Hi-C [64] techniques, have been used in a human protein function prediction pipeline [65], but they are certainly not exploited enough.

Epigenetic markers, such as DNA methylation or chromosome accessibility, affect gene regulation, implying that genes that have similar epigenetic patterns across tissues might be regulated jointly. This makes epigenetic data potentially a rich resource for predicting Biological Process terms, although it is not known to what extent this signal is complementary to gene expression. Human and mouse are the two species where this hypothesis can be easily tested as many of their genes have well-documented functions and the ENCODE project has generated large amounts of epigenetic data in both species [66].

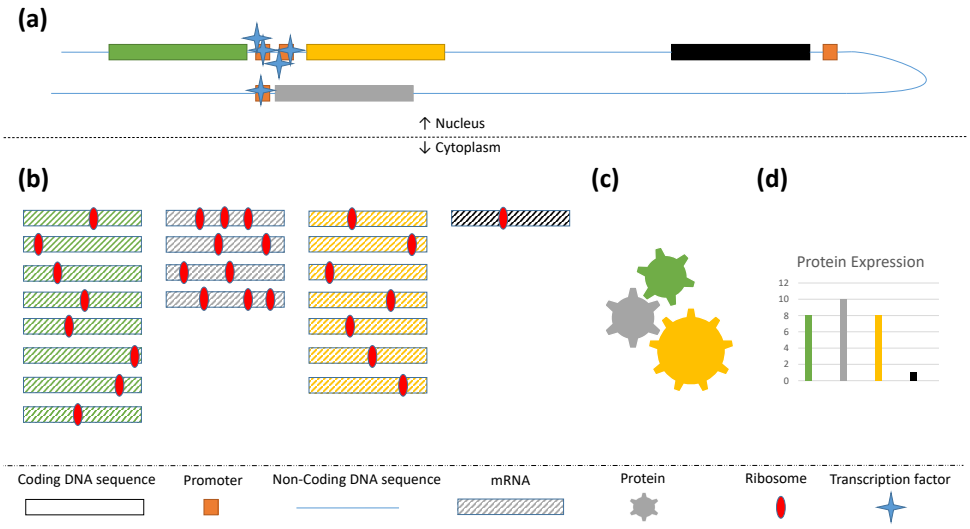


Figure 6.4: Genes that are close to each other in space are likely to be co-expressed as transcription factors that bind on one promoter are likely to also bind to nearby ones (panel a). Differences in expression are indicated by the amount of transcripts in the cytoplasm (panel b). In this case the green, yellow and gray genes code for proteins (shown as gears of the same color) that form a protein complex and perform a function together (panel c). The ribosome occupancy of the mRNAs is used to indicate differences in translation efficiency which result in similar abundances for proteins (panel d) despite different mRNA abundancies.

Finally, mining of scientific literature has been under-used in the past years, but a recent study showed that it can have competitive performance [67]. The idea behind the usage of such text mining methods is that if two genes are often mentioned together in publications, then they are likely to be involved in the same function. In addition, co-occurrence of gene names with other words, such as disease or pathway names [68], can be informative. As successful NLP models are starting to be applied in biomedical literature data [68, 69], we expect the role of text mining in AFP to increase in the immediate future.

6.4.2. FUNCTION REPRESENTATION

Using new data sources in both computational and experimental protein annotation might call for new evidence codes. For instance, if indeed protein co-expression is much more relevant for co-functionality than mRNA co-expression, it might make sense to differentiate between functions discovered using the two technologies by splitting the HEP (High-throughput Expression Pattern) evidence code. For similar epigenetic profiles and co-regulation in general, there is also no appropriate evidence code. The closest one is perhaps IGI (Inferred from Genetic Interaction), but this mainly refers to one gene influencing another gene (e.g. by changes in expression or mutations), while co-regulation implies that both genes are jointly regulated by the same mechanism.

6.4.3. PREDICTION METHODS

Arguably one of the most important recent methodological leaps is representation learning. Advances in machine learning have been quickly adopted by AFP researchers, causing a shift of the research efforts from the guilt-by-association (GBA) paradigm to automatic representation learning. A lot of recent works, often inspired from natural language processing, use convolutional or recurrent neural networks to automatically learn sequence features useful for predicting function. Such methods have been shown to work better than simple homology search (e.g. [70]). Also, several neural embedding methods have been recently proposed which can learn complex features for nodes of networks. Such methods have been applied to both PPI and co-expression networks and shown to be useful for reconstructing functional relationships [71, 72]. However, there is still not enough evidence of whether such methods can outperform simple GBA methods, such as gene co-expression based on Pearson correlation, which is very effective for BPO predictions [13].

6.5. BIASED AND MISSING ANNOTATIONS

Another unresolved question that is important to address is how biases in GO annotations influence AFP. One extreme form of such a bias is missing annotations. Missing annotations in the test set do not have a dramatic effect on the ranking of AFP methods [73], but the effect of missing annotations in the training set has not yet been systematically quantified. We suspect that this effect is larger, especially for machine-learning-based methods that try to learn characteristics of proteins with the same function.

6.5.1. PROTEIN REPRESENTATION

What is often overlooked is that the biased way that annotations are generated might also affect the protein representation. For example, in plant research, scientists are very often interested in stress responses and flowering, so a lot of the available gene expression data come from such conditions, meaning that it might be harder to infer other functions for which very little data are available. The same holds for other species. For example, *Drosophila melanogaster* is typically used to study genetics [74], *Caenorhabditis elegans* for development [75] and so on. The detection of condition-specific or tissue-specific protein-protein interaction suffers from the same issue. Protein sequence data are also not completely 'safe' from this bias, as it is possible that a functional isoform of a protein used in rare, poorly-studied conditions is not known. However, for sequence data the effect of that bias is arguably smaller than for other data types.

6.5.2. FUNCTION REPRESENTATION

Many researchers tend to only include *experimental* evidence codes (EC's) when training and testing AFP methods to avoid biases. This has the downside that a great amount of knowledge about protein function is potentially ignored. Moreover, experimental EC's have also been shown to be highly biased [76], leading to the recent split between "experimental" and "high-throughput experimental" EC's. Also, experimental EC's include co-expression and protein interaction experiments which might introduce circular reasoning for algorithms that use such data types, in the same way that sequence similarity

EC's introduce bias for sequence-based algorithms. On the other hand, annotations that are automatically generated by a curated set of rules are labeled "IEA" and are often ignored, although they are rather reliable, given that the rules are made by expert curators [77]. We cannot convince experimentalists to start randomly selecting a protein and testing it for random functions to obtain an unbiased ground-truth with independent, identically distributed observations. Nor can we ever obtain a reliable, experimentally-derived set of negative annotations, as a protein might have a particular function only under certain conditions. Therefore, it might be interesting to try to further quantify the biases introduced by including certain EC's. Previous work quantified the quality of automatic annotations by measuring to what extent they were later experimentally confirmed [77]. Additionally, one could look for possible changes in the performance of the naïve classifier when adding or removing annotations with a specific EC. Also, one could compare the performance of AFP methods for each evidence code separately to assess these biases better.

Some studies have attempted to tackle the missing annotations problem by generating negative examples [78, 79]. Somewhat surprisingly, these datasets of negative annotations have not gained popularity among AFP researchers, as they are most often not used during the training of new methods. To our knowledge, there is no study providing evidence against using these data, so we believe that this topic deserves further investigation. A recent study showed that ignoring negative annotations at test time can lead to very misleading evaluations [80].

6.5.3. PREDICTION METHODS

Alternatively to generating negative annotations, we could also change the loss functions used to train AFP models. Most recent machine learning models are trained by minimizing a cross-entropy loss, which assumes no missing labels. That could be replaced by a loss for Positive-Unlabeled (PU) learning [81]. Again, these loss functions are well-known in AFP and related fields [78, 82], but they seem not to be very popular. It is possible, that despite their obvious theoretical benefits, such losses do not work in practice for AFP and publication bias has prevented us from seeing these results. A different approach would be to introduce probabilistic labels, where annotations from EC's that are considered reliable are attributed with high certainty, but unreliable ones (e.g. derived automatically by another AFP method and never verified by a curator) are used in the training set but with a low-probability. One could additionally always assign a non-zero probability to all other terms to account for missing annotations. Such a probabilistic ground-truth means that probabilistic learning algorithms will be needed, such as the graphical model proposed in [83].

6.6. GENE ONTOLOGY

GO is a very useful resource that describes biological function in a standardized manner that is human- and computer-readable. This has led to its nearly catholic acceptance as the go-to functional representation, to the point that function prediction is almost a synonym of GO term prediction.

6.6.1. PROTEIN REPRESENTATION

As stated above, automatic representation learning is a very promising direction. It can be further enhanced by unsupervised pre-training, where a generic protein representation is learned [84–87] from all available sequences. This representation can then be used to predict GO terms [88]. But end-to-end training is also possible, where the weights of the unsupervised feature extractor are also fine-tuned to create an ontology-specific feature representation designed for predicting GO terms from that ontology. This can lead to better performance, especially when only few labeled examples are available. Such approaches are currently the state-of-the-art in Natural Language Processing [89].

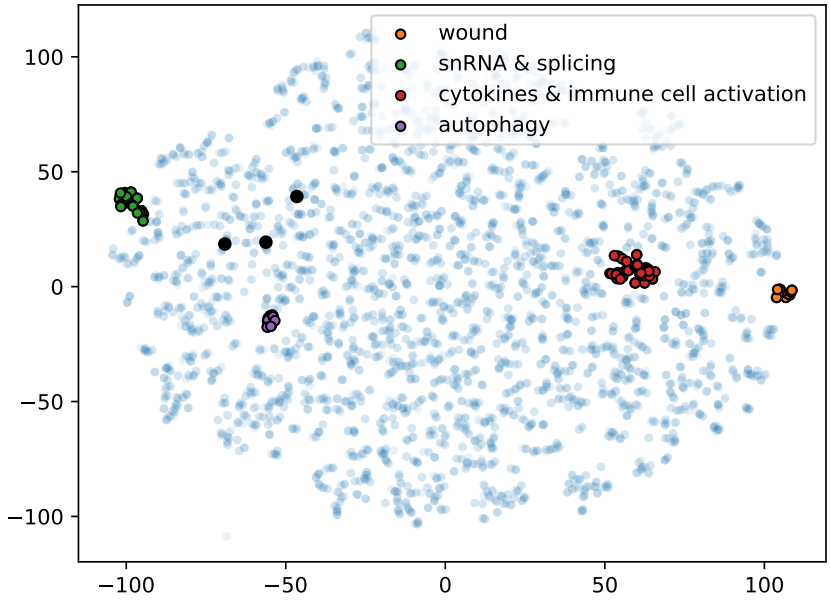


Figure 6.5: Two-dimensional tSNE embedding of GO terms from all three ontologies that annotate at least 0.1% of SwissProt entries. One minus the Pearson correlation of the occurence patterns of terms across SwissProt proteins was used as a distance measure for calculating the embeddings. Inspecting the terms in some of the clusters observed in this 2D space revealed that terms from the same cluster have similar meanings. Examples of clusters with terms that refer to wound healing, small nuclear RNAs and mRNA splicing complexes, cytokines and immune activation, and autophagy are shown in orange, green, red, and purple respectively. Terms "DNA binding", "nucleus" and "DNA-templated regulation of transcription" are shown as larger black dots.

6.6.2. FUNCTION REPRESENTATION

We previously touched upon embedding GO terms in a vector space and its potential in creating a new functional representation that is simpler but reflects the same semantic

relationships as the GO graph. Figure 6.5 shows an example of such an embedding in two dimensions that reflects term co-occurence patterns. The biggest downside of those approaches is the loss of the interpretability and human-readability that GO terms have. It is still not trivial to always provide a biological interpretation for a given set of GO terms (even though visualization techniques are helping in that regard [90]), but it is still relatively easy to understand the meaning of an individual term by its name, description, and connections to ancestors and descendants. On the other hand, representing terms as high-dimensional vectors makes us lose the intuition, which implies that we would like this representation to be invertible, i.e. also provide us with a rule to convert a given vector in this "functional space" back to a term or a set of terms, ideally in a unique way. This is possible for linear mappings and we and others have worked on such approaches [91, 92]. Linear approaches can capture simple relationships between terms such as co-occurrence or mutual exclusivity of a pair of terms [91], but might struggle to find more complicated relationships "hidden" either in the graph or in the semantics of terms. We therefore suspect that non-linear (e.g. neural) term embeddings are required to capture the whole structure. Nevertheless, we think that substantial emphasis and attention should be put on maintaining the interpretability of these models. This is nowadays also a hot topic in machine learning and computer vision [93–95] , which has been dominated by neural networks in the past decade.

6.6.3. PREDICTION METHODS

The three different ontologies contain correlated information. For example, “DNA binding” (MFO) co-occurs with “DNA-templated regulation of transcription” (BPO, $\rho = 0.54$) and “nucleus” (CCO, $\rho = 0.35$) (Figure 6.5). GO curators are well aware of these co-occurrences and have hand-crafted rules to automatically transfer such annotations. However, little computational work has been done to improve upon this rule-based system, although this seems like a very promising direction, especially for the Limited-Knowledge category of CAFA, where the goal is to predict the functions of proteins in one ontology using its functions in others. A possible approach would be to embed the terms of each ontology to a vector space and then try to align the three ontologies, with the aim of discovering new cross-ontology similarities that are not obvious to the curators and could be exploited by function prediction algorithms. In Onto2vec, the authors learned a joint embedding for terms from all three ontologies and proteins [96]. They used it to calculate protein similarities for the downstream task of protein interaction prediction, but it would also be very interesting to examine similarities in the embedding space between pairs of GO terms to identify potentially unknown correlations.

Over time, unannotated or partly annotated genes obtain new GO annotations, i.e. the ground-truth data that can be used to train AFP models changes. Therefore, it would be interesting to have models which can incorporate such new knowledge without having to re-train from scratch. This could be achieved by using online machine learning algorithms [97, 98].

6.7. EVALUATION OF AFP ALGORITHMS

Next to the challenges described above, the problem of properly evaluating AFP models, as already initially identified as one of the three main challenges by Godzik et al. [14], is still lingering. It has been shown that temporal hold-out evaluation strategies, like the CAFA challenges, give more realistic estimates of the performance on new unseen data than cross-validation [99]. Given the success of omics data in CAFA - and specifically in CAFA-π [13]- we believe that a GBA-based baseline should be included in future CAFA editions, for example a GBA based on PPI's from e.g. the STRING database [100]. This is to be preferred over a co-expression-based baseline as for many species this data is not readily available.

By comparing the rankings of methods in CAFA3 [13] across five evaluation metrics, we found that most widely-used metrics are highly correlated, with the Semantic Distance [101] being slightly different from the rest (Figure 1.7). However, a more recent simulation study questioned the validity of these commonly used evaluation measures, such as the F1 score and the Semantic Distance [101]. The authors also proposed novel measures that more accurately reflect the quality of the predictions [102]. The complicated and biased nature of GO annotations can indeed lead to misleading evaluations, so having appropriate evaluation measures is essential to ensure that the field keeps going forward.

6.8. CONCLUSION

To conclude, AFP remains one of the most challenging bioinformatics tasks and despite its growing interest and recent progress, it is far from being completely solved. Here, we addressed some of the current and future challenges of the field. In our view, significant breakthroughs are expected mainly from the use of neural embeddings (for describing both proteins and GO terms) and the use of new technologies, such as proteomics. The problem of predicting function for non-model species is possibly the most challenging, as it may still require generation of experimental data. However, as the community of AFP researchers is growing [13], we are optimistic that these challenges will soon be tackled.

REFERENCES

[1] S. Makrodimitris, R. C. H. J. van Ham, and M. J. T. Reinders, *Automatic gene function prediction in the 2020's*, Genes **11**, 1264 (2020).

[2] A. Bateman, *UniProt: A worldwide hub of protein knowledge*, Nucleic Acids Research **47**, D506 (2019).

[3] T. B. González-Castro, C. A. Tovilla-Zárate, A. D. Genis-Mendoza, I. E. Juárez-Rojop, H. Nicolini, M. L. López-Narváez, and J. J. Martínez-Magaña, *Identification of gene ontology and pathways implicated in suicide behavior: Systematic review and enrichment analysis of gwas studies*, American Journal of Medical Genetics Part B: Neuropsychiatric Genetics **180**, 320 (2019), https://onlinelibrary.wiley.com/doi/pdf/10.1002/ajmg.b.32731 .

[4] R. You, Z. Zhang, Y. Xiong, F. Sun, H. Mamitsuka, and S. Zhu, *GOLabeler: Improving sequence-based large-scale protein function prediction by learning to rank,* Bioinformatics (2018), 10.1093/bioinformatics/bty130.

[5] S. Das, D. Lee, I. Sillitoe, N. L. Dawson, J. G. Lees, and C. A. Orengo, *Functional classification of CATH superfamilies: a domain-based approach for protein function annotation,* Bioinformatics **31**, 3460 (2015), https://academic.oup.com/bioinformatics/article-pdf/31/21/3460/17122354/btv398.pdf .

[6] D. Piovesan and S. C. E. Tosatto, *INGA 2.0: improving protein function prediction for the dark proteome,* Nucleic Acids Research **47**, W373 (2019), https://academic.oup.com/nar/article-pdf/47/W1/W373/28879989/gkz375.pdf .

[7] A. Jain and D. Kihara, *Phylo-PFP: improved automated protein function prediction using phylogenetic distance of distantly related sequences,* Bioinformatics **35**, 753 (2018), https://academic.oup.com/bioinformatics/article-pdf/35/5/753/27994728/bty704.pdf .

[8] C. Zhang, P. L. Freddolino, and Y. Zhang, *COFACTOR: improved protein function prediction by combining structure, sequence and protein–protein interaction information,* Nucleic Acids Research **45**, W291 (2017), https://academic.oup.com/nar/article-pdf/45/W1/W291/23741003/gkx366.pdf .

[9] R. You, S. Yao, Y. Xiong, X. Huang, F. Sun, H. Mamitsuka, and S. Zhu, *NetGO: improving large-scale protein function prediction with massive network information,* Nucleic Acids Research (2019), 10.1093/nar/gkz388.

[10] M. Kulmanov and R. Hoehndorf, *DeepGOPlus: improved protein function prediction from sequence,* Bioinformatics **36**, 422 (2019), https://academic.oup.com/bioinformatics/article-pdf/36/2/422/31962785/btz595.pdf .

[11] P. Radivojac, W. T. Clark, T. R. Oron, A. M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, A. Ben-Hur, G. Pandey, and J. M. Yunes, *A large-scale evaluation of computational protein function prediction,* Nature Methods **10**, 221 (2013), arXiv:1510.05682 .

[12] Y. Jiang *et al., An expanded evaluation of protein function prediction methods shows an improvement in accuracy,* Genome biology **17**, 184 (2016), 1601.00891 .

[13] N. Zhou *et al., The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens,* Genome Biology **20** (2019), 10.1186/s13059-019-1835-8.

[14] A. Godzik, M. Jambon, and I. Friedberg, *Computational protein function prediction: Are we making progress?* (2007).

[15] D. Cozzetto, D. W. Buchan, K. Bryson, and D. T. Jones, *Protein function prediction by massive integration of evolutionary analyses and multiple data sources,* BMC Bioinformatics **14**, S1 (2013).

[16] L. Lan, N. Djuric, Y. Guo, and S. Vucetic, *MS-kNN: protein function prediction by integrating multiple data sources.* BMC bioinformatics **14 Suppl 3**, S8 (2013).

[17] M. Farahbod and P. Pavlidis, *Differential coexpression in human tissues and the confounding effect of mean expression levels,* Bioinformatics **35**, 55 (2019).

[18] A. R. Sonawane, J. Platig, M. Fagny, C. Y. Chen, J. N. Paulson, C. M. Lopes-Ramos, D. L. DeMeo, J. Quackenbush, K. Glass, and M. L. Kuijjer, *Understanding Tissue-Specific Gene Regulation,* Cell Reports **21**, 1077 (2017).

[19] Z. Jiang, X. Dong, Z. G. Li, F. He, and Z. Zhang, *Differential coexpression analysis reveals extensive rewiring of arabidopsis gene coexpression in response to pseudomonas syringae infection,* Scientific Reports **6** (2016), 10.1038/srep35064.

[20] A. J. Singh, S. A. Ramsey, T. M. Filtz, and C. Kioussi, *Differential gene regulatory networks in development and disease,* (2018).

[21] O. Basha, R. Shpringer, C. M. Argov, and E. Yeger-Lotem, *The DifferentialNet database of differential protein-protein interactions in human tissues,* Nucleic Acids Research **46**, D522 (2018).

[22] C. S. Greene, A. Krishnan, A. K. Wong, E. Ricciotti, R. A. Zelaya, D. S. Himmelstein, R. Zhang, B. M. Hartmann, E. Zaslavsky, S. C. Sealfon, D. I. Chasman, G. A. Fitzgerald, K. Dolinski, T. Grosser, and O. G. Troyanskaya, *Understanding multicellular function and disease with human tissue-specific networks,* Nature Genetics **47**, 569 (2015).

[23] A. D. Diehl, T. F. Meehan, Y. M. Bradford, M. H. Brush, W. M. Dahdul, D. S. Dougall, Y. He, D. Osumi-Sutherland, A. Ruttenberg, S. Sarntivijai, C. E. Van Slyke, N. A. Vasilevsky, M. A. Haendel, J. A. Blake, and C. J. Mungall, *The cell ontology 2016: enhanced content, modularization, and ontology interoperability,* Journal of Biomedical Semantics **7**, 44 (2016).

[24] M. Zitnik and J. Leskovec, *Predicting multicellular function through multi-layer tissue networks,* in *Bioinformatics*, Vol. 33 (2017) pp. i190–i198, arXiv:1707.04638 .

[25] S. Mahdavi, S. Khoshraftar, and A. An, *Dynnode2vec: Scalable Dynamic Network Embedding,* in *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018* (2019) pp. 3762–3765, arXiv:1812.02356 .

[26] D. A. Jaitin, E. Kenigsberg, H. Keren-Shaul, N. Elefant, F. Paul, I. Zaretsky, A. Mildner, N. Cohen, S. Jung, A. Tanay, and I. Amit, *Massively parallel single-cell rna-seq for marker-free decomposition of tissues into cell types,* Science **343**, 776 (2014), https://science.sciencemag.org/content/343/6172/776.full.pdf .

[27] I. Papatheodorou, P. Moreno, J. Manning, A. M.-P. Fuentes, N. George, S. Fexova, N. A. Fonseca, A. Füllgrabe, M. Green, N. Huang, L. Huerta, H. Iqbal, M. Jianu, S. Mohammed, L. Zhao, A. F. Jarnuczak, S. Jupp, J. Marioni, K. Meyer, R. Petryszak, C. A. Prada Medina, C. Talavera-López, S. Teichmann, J. A. Vizcaino, and A. Brazma, *Expression Atlas update: from tissues to single cells,* Nucleic Acids Research **48**, D77 (2019), https://academic.oup.com/nar/article-pdf/48/D1/D77/31697837/gkz947.pdf .

[28] P. J. Thul and C. Lindskog, *The human protein atlas: A spatial map of the human proteome,* Protein Science **27**, 233 (2018), https://onlinelibrary.wiley.com/doi/pdf/10.1002/pro.3307 .

[29] N. Japkowicz and S. Stephen, *The class imbalance problem: A systematic study,* Intelligent Data Analysis **6**, 429 (2002), 5.

[30] GO Consortium, *Guide to GO Evidence Codes,* http://geneontology.org/page/guide-go-evidence-codes (2016).

[31] *Annotation extension,* `http://wiki.geneontology.org/index.php/Annotation_Extension`, accessed: 2020-09-12.

[32] P. D. Thomas, D. P. Hill, H. Mi, D. Osumi-Sutherland, K. Van Auken, S. Carbon, J. P. Balhoff, L.-P. Albou, B. Good, P. Gaudet, S. E. Lewis, and C. J. Mungall, *Gene ontology causal activity modeling (go-cam) moves beyond go annotations to structured descriptions of biological functions and systems,* Nature Genetics **51**, 1429 (2019).

[33] E. F. Lock, K. A. Hoadley, J. S. Marron, and A. B. Nobel, *JOINT AND INDIVIDUAL VARIATION EXPLAINED (JIVE) FOR INTEGRATED ANALYSIS OF MULTIPLE DATA TYPES,* Ann Appl Stat **7**, 523 (2013).

[34] G. P. Way and C. S. Greene, *Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders,* Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing **23**, 80 (2018), 29218871[pmid].

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets,* in *Advances in Neural Information Processing Systems 27*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Curran Associates, Inc., 2014) pp. 2672–2680.

[36] M. Mirza and S. Osindero, *Conditional generative adversarial nets,* (2014), arXiv:1411.1784 [cs.LG] .

[37] L. Perez and J. Wang, *The effectiveness of data augmentation in image classification using deep learning,* (2017), arXiv:1712.04621 [cs.CV] .

[38] C. Wan and D. T. Jones, *Protein function prediction is improved by creating synthetic feature samples with generative adversarial networks,* Nature Machine Intelligence **2**, 540 (2020).

[39] R. Appels *et al., Shifting the limits in wheat research and breeding using a fully annotated reference genome,* Science **361** (2018), 10.1126/science.aar7191, https://science.sciencemag.org/content/361/6403/eaar7191.full.pdf .

[40] F. Richoux, C. Servantie, C. Borès, and S. Téletchéa, *Comparing two deep learning sequence-based models for protein-protein interaction prediction,* (2019), arXiv:1901.06268 [cs.LG] .

[41] O. M. Sigalova, A. Shaeiri, M. Forneris, E. E. Furlong, and J. B. Zaugg, *Predictive features of gene expression variation reveal a mechanistic link between expression variation and differential expression,* bioRxiv (2020), 10.1101/2020.02.10.942276, https://www.biorxiv.org/content/early/2020/02/11/2020.02.10.942276.full.pdf .

[42] S. Wang, H. Cho, C. Zhai, B. Berger, and J. Peng, *Exploiting ontology graph for predicting sparsely annotated gene function,* Bioinformatics **31**, i357 (2015), https://academic.oup.com/bioinformatics/article-pdf/31/12/i357/17102342/btv260.pdf .

[43] D. Duong, A. Uppunda, L. Gai, C. Ju, J. Zhang, M. Chen, E. Eskin, J. J. Li, and K.-W. Chang, *Evaluating representations for gene ontology terms,* bioRxiv (2020), 10.1101/765644, https://www.biorxiv.org/content/early/2020/01/31/765644.full.pdf .

[44] B. P. Chamberlain, J. Clough, and M. P. Deisenroth, *Neural embeddings of graphs in hyperbolic space,* (2017), arXiv:1705.10359 [stat.ML] .

[45] X. Li, Z. Sun, J.-H. Xue, and Z. Ma, *A concise review of recent few-shot meta-learning methods,* arXiv preprint arXiv:2005.10953 (2020).

[46] Y. Xian, B. Schiele, and Z. Akata, *Zero-shot learning - the good, the bad and the ugly,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[47] D. Huynh and E. Elhamifar, *Fine-grained generalized zero-shot learning via dense attribute-based attention,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

[48] S. Wang, A. O. Pisco, A. McGeever, M. Brbic, M. Zitnik, S. Darmanis, J. Leskovec, J. Karkanias, and R. B. Altman, *Unifying single-cell annotations based on the cell ontology,* bioRxiv (2020), 10.1101/810234, https://www.biorxiv.org/content/early/2020/02/04/810234.full.pdf .

[49] W. M. Kouw and M. Loog, *An introduction to domain adaptation and transfer learning,* (2018), arXiv:1812.11806 [cs.LG] .

[50] V. Kumar, A. Sharma, R. Kaur, A. K. Thukral, R. Bhardwaj, and P. Ahmad, *Differential distribution of amino acids in plants,* Amino Acids **49**, 821 (2017).

[51] J. Munro and D. Damen, *Multi-modal domain adaptation for fine-grained action recognition,* in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)* (2019) pp. 3723–3726.

[52] J. Wang, Z. Ma, S. A. Carr, P. Mertins, H. Zhang, Z. Zhang, D. W. Chan, M. J. Ellis, R. R. Townsend, R. D. Smith, J. E. McDermott, X. Chen, A. G. Paulovich, E. S. Boja, M. Mesri, C. R. Kinsinger, H. Rodriguez, K. D. Rodland, D. C. Lieblerc, and B. Zhang, *Proteome profiling outperforms transcriptome profiling for coexpression based gene function prediction,* Molecular and Cellular Proteomics **16**, 121 (2017).

[53] T. J. Griffin, S. P. Gygi, T. Ideker, B. Rist, J. Eng, L. Hood, and R. Aebersold, *Complementary profiling of gene expression at the transcriptome and proteome levels in saccharomyces cerevisiae,* Molecular & Cellular Proteomics **1**, 323 (2002), https://www.mcponline.org/content/1/4/323.full.pdf .

[54] X. Wang, Q. Liu, and B. Zhang, *Leveraging the complementary nature of rna-seq and shotgun proteomics data,* PROTEOMICS **14**, 2676 (2014), https://onlinelibrary.wiley.com/doi/pdf/10.1002/pmic.201400184 .

[55] P. Grabowski, G. Kustatscher, and J. Rappsilber, *Epigenetic variability confounds transcriptome but not proteome profiling for coexpression-based gene function prediction,* Molecular & Cellular Proteomics **17**, 2082 (2018), https://www.mcponline.org/content/17/11/2082.full.pdf .

[56] D. Wang, X. Zou, and K. Fai Au, *A network-based computational framework to predict and differentiate functions for gene isoforms using exon-level expression data,* Methods (2020), https://doi.org/10.1016/j.ymeth.2020.06.005.

[57] R. T. Perchey, L. Tonini, M. Tosolini, J.-J. Fournié, F. Lopez, A. Besson, and F. Pont, *Ptmselect: optimization of protein modifications discovery by mass spectrometry,* Scientific Reports **9**, 4181 (2019).

[58] V. Csizmok and J. D. Forman-Kay, *Complex regulatory mechanisms mediated by the interplay of multiple post-translational modifications,* Current Opinion in Structural Biology **48**, 58 (2018), folding and binding in silico, in vitro and in cellula • Proteins: An Evolutionary Perspective.

[59] J. B. Müller, P. E. Geyer, A. R. Colaço, P. V. Treit, M. T. Strauss, M. Oroshi, S. Doll, S. Virreira Winter, J. M. Bader, N. Köhler, F. Theis, A. Santos, and M. Mann, *The proteome landscape of the kingdoms of life,* Nature , 1 (2020).

[60] M. Huynen, B. Snel, W. Lathe, and P. Bork, *Predicting protein function by genomic context: quantitative evaluation and qualitative inferences,* Genome Res. **10**, 1204 (2000).

[61] F. Foflonker and C. E. Blaby-Haas, *Co-locality to co-functionality: Eukaryotic gene neighborhoods as a resource for function discovery,* Molecular Biology and Evolution (2020), 10.1093/molbev/msaa221, msaa221, https://academic.oup.com/mbe/advance-article-pdf/doi/10.1093/molbev/msaa221/33716050/msaa221.pdf .

[62] S. Schoenfelder, T. Sexton, L. Chakalova, N. F. Cope, A. Horton, S. Andrews, S. Kurukuti, J. A. Mitchell, D. Umlauf, D. S. Dimitrova, C. H. Eskiw, Y. Luo, C.-L. Wei, Y. Ruan, J. J. Bieker, and P. Fraser, *Preferential associations between co-regulated genes reveal a transcriptional interactome in erythroid cells,* Nature Genetics **42**, 53 (2010).

[63] Z. Zhao, G. Tavoosidana, M. Sjölinder, A. Göndör, P. Mariano, S. Wang, C. Kanduri, M. Lezcano, K. Singh Sandhu, U. Singh, V. Pant, V. Tiwari, S. Kurukuti, and R. Ohlsson, *Circular chromosome conformation capture (4c) uncovers extensive networks of epigenetically regulated intra- and interchromosomal interactions,* Nature Genetics **38**, 1341 (2006).

[64] N. L. van Berkum, E. Lieberman-Aiden, L. Williams, M. Imakaev, A. Gnirke, L. A. Mirny, J. Dekker, and E. S. Lander, *Hi-c: a method to study the three-dimensional architecture of genomes,* Journal of visualized experiments : JoVE , 1869 (2010), 20461051[pmid].

[65] R. Cao and J. Cheng, *Integrated protein function prediction by mining function associations, sequences, and protein-protein and gene-gene interaction networks,* Methods **93**, 84 (2016).

[66] J. E. Moore *et al., Expanded encyclopaedias of dna elements in the human and mouse genomes,* Nature **583**, 699 (2020).

[67] R. You, X. Huang, and S. Zhu, *Deeptext2go: Improving large-scale protein function prediction with deep semantic text representation,* Methods **145**, 82 (2018), data mining methods for analyzing biological data in terms of phenotypes.

[68] Q. Chen, K. Lee, S. Yan, S. Kim, C. H. Wei, and Z. Lu, *BioConceptVec: Creating and evaluating literature-based biomedical concept embeddings on a large scale,* PLoS Comput Biol **16**, e1007617 (2020).

[69] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, *BioBERT: a pre-trained biomedical language representation model for biomedical text mining,* Bioinformatics **36**, 1234 (2019), https://academic.oup.com/bioinformatics/article-pdf/36/4/1234/32527770/btz682.pdf .

[70] A. S. Rifaioglu, T. Doğan, M. J. Martin, R. Cetin-Atalay, and M. V. Atalay, *Multi-task Deep Neural Networks in Automated Protein Function Prediction,* arXiv:1705.04802 [q-bio.QM] (2017), arXiv:1705.04802 .

[71] A. Grover and J. Leskovec, *node2vec: Scalable Feature Learning for Networks,* Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16 , 855 (2016), arXiv:1607.00653 .

[72] J. Du, P. Jia, Y. Dai, C. Tao, Z. Zhao, and D. Zhi, *Gene2vec: Distributed representation of genes based on co-expression,* BMC Genomics **20** (2019), 10.1186/s12864-018-5370-x.

[73] Y. Jiang, W. T. Clark, I. Friedberg, and P. Radivojac, *The impact of incomplete knowledge on the evaluation of protein function prediction: A structured-output learning perspective,* Bioinformatics **30**, 609 (2014).

[74] K. G. Hales, C. A. Korey, A. M. Larracuente, and D. M. Roberts, *Genetics on the fly: A primer on the drosophila model system,* Genetics **201**, 815 (2015), https://www.genetics.org/content/201/3/815.full.pdf .

[75] P. E. Kuwabara and N. O'Neil, *The use of functional genomics in c. elegans for studying human development and disease,* Journal of Inherited Metabolic Disease **24**, 127 (2001).

[76] A. M. Schnoes, D. C. Ream, A. W. Thorman, P. C. Babbitt, and I. Friedberg, *Biases in the Experimental Annotations of Protein Function and Their Effect on Our Understanding of Protein Function Space,* PLoS Computational Biology **9**, 1 (2013), arXiv:1301.1740 .

[77] N. Škunca, A. Altenhoff, and C. Dessimoz, *Quality of computationally inferred gene ontology annotations,* PLoS Computational Biology **8** (2012), 10.1371/journal.pcbi.1002533.

[78] N. Youngs, D. Penfold-Brown, R. Bonneau, and D. Shasha, *Negative Example Selection for Protein Function Prediction: The NoGO Database,* PLoS Computational Biology **10** (2014), 10.1371/journal.pcbi.1003644.

[79] G. Fu, J. Wang, B. Yang, and G. Yu, *NegGOA: negative GO annotations selection using ontology structure,* Bioinformatics **32**, 2996 (2016), https://academic.oup.com/bioinformatics/article-pdf/32/19/2996/25072589/btw366.pdf .

[80] A. Warwick Vesztrocy and C. Dessimoz, *Benchmarking gene ontology function predictions using negative annotations,* Bioinformatics **36**, i210 (2020), https://academic.oup.com/bioinformatics/article-pdf/36/Supplement_1/i210/33488859/btaa466.pdf .

[81] R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama, *Positive-unlabeled learning with non-negative risk estimator,* in *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017) pp. 1675–1685.

[82] P. Yang, X. L. Li, J. P. Mei, C. K. Kwoh, and S. K. Ng, *Positive-unlabeled learning for disease gene identification,* Bioinformatics **28**, 2640 (2012).

[83] A. Akbarnejad and M. S. Baghshah, *A probabilistic multi-label classifier with missing and noisy labels handling capability,* Pattern Recognition Letters **89**, 18 (2017).

[84] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, P. Chen, J. Canny, P. Abbeel, and Y. Song, *Evaluating protein transfer learning with tape,* in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019) pp. 9689–9701.

[85] M. Heinzinger, A. Elnaggar, Y. Wang, C. Dallago, D. Nechaev, F. Matthes, and B. Rost, *Modeling aspects of the language of life through transfer-learning protein sequences,* BMC Bioinformatics (2019), 10.1186/s12859-019-3220-8.

[86] E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, and G. M. Church, *Unified rational protein engineering with sequence-based deep representation learning,* Nature Methods **16**, 1315 (2019).

[87] A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus, *Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences,* bioRxiv (2020), 10.1101/622803, https://www.biorxiv.org/content/early/2020/08/31/622803.full.pdf .

[88] A. Villegas-Morcillo, S. Makrodimitris, R. C. H. J. van Ham, A. M. Gomez, V. Sanchez, and M. J. T. Reinders, *Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function,* Bioinformatics (2020), 10.1093/bioinformatics/btaa701, btaa701, https://academic.oup.com/bioinformatics/advance-article-pdf/doi/10.1093/bioinformatics/btaa701/33653342/btaa701.pdf .

[89] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding,* (2018), arXiv:1810.04805 [cs.CL] .

[90] Q. Wei, I. K. Khan, Z. Ding, S. Yerneni, and D. Kihara, *Navigo: interactive tool for visualization and functional similarity and coherence analysis with gene ontology,* BMC Bioinformatics **18**, 177 (2017).

[91] S. Makrodimitris, R. C. H. J. Van Ham, and M. J. T. Reinders, *Improving Protein Function Prediction in Ara-bidopsis Using Protein Sequence and GO-term Similarities,* under review (2019), 10.1093/bioinformatics/bty751.

[92] W. Bi and J. Kwok, *Multi-label classification on tree-and DAG-structured hierarchies,* in *Icml* (2011) pp. 17–24.

[93] A. Adadi and M. Berrada, *Peeking inside the black-box: A survey on explainable artificial intelligence (xai),* IEEE Access **6**, 52138 (2018).

[94] L. Longo, R. Goebel, F. Lecue, P. Kieseberg, and A. Holzinger, *Explainable artificial intelligence: Concepts, applications, research challenges and visions,* in *Machine Learning and Knowledge Extraction*, edited by A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl (Springer International Publishing, Cham, 2020) pp. 1–16.

[95] H. J. Escalante, S. Escalera, I. Guyon, X. Baró, Y. Güçlütürk, U. Güçlü, and M. van Gerven, eds., *Explainable and Interpretable Models in Computer Vision and Machine Learning* (Springer International Publishing, 2018).

[96] F. Z. Smaili, X. Gao, and R. Hoehndorf, *Onto2Vec: Joint vector-based representation of biological entities and their ontology-based annotations,* in *Bioinformatics*, Vol. 34 (2018) pp. i52–i60, arXiv:1802.00864 .

[97] R. Venkatesan, M. J. Er, M. Dave, M. Pratama, and S. Wu, *A novel online multi-label classifier for high-speed streaming data applications,* Evolving Systems **8**, 303 (2017).

[98] Z. Ahmadi and S. Kramer, *Online multi-label classification: A label compression method,* (2018), arXiv:1804.01491 [cs.LG] .

[99] I. Kahanda, C. S. Funk, F. Ullah, K. M. Verspoor, and A. Ben-Hur, *A close look at protein function prediction evaluation protocols.* GigaScience **4**, 41 (2015).

[100] D. Szklarczyk, A. L. Gable, D. Lyon, A. Junge, S. Wyder, J. Huerta-Cepas, M. Simonovic, N. T. Doncheva, J. H. Morris, P. Bork, L. J. Jensen, and C. V. Mering, *STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets,* Nucleic Acids Res **47**, D607 (2019).

[101] W. T. Clark and P. Radivojac, *Information-theoretic evaluation of predicted ontological annotations,* Bioinformatics **29**, i53 (2013).

[102] I. Plyusnin, L. Holm, and P. Törönen, *Novel comparison of evaluation metrics for gene ontology classifiers reveals drastic performance differences,* PLOS Computational Biology **15**, 1 (2019).

ACKNOWLEDGEMENTS

I found that doing a PhD can be compared to the Odyssey: it is a long, often lonely journey with many setbacks, and with several reviewing monsters waiting to devour you at your first minor slip-up. My odyssey brought me to places such as Prague, Basel, Athens, Wageningen and the exotic Lunteren. However, although Odysseus's journey companions often got him into trouble, mine were essential for completing this journey while also maintaining my sanity.

First and foremost, I would like to thank my whole family and especially my parents, my wife and my in-laws for their unconditional support and love. Σας ευχαριστώ για τη στήριξη και την κατανόηση όλα αυτά τα χρόνια. Δεν θα τα είχα καταφέρει χωρίς εσάς. Special credits to **Valentina** for proofreading all my manuscripts before submission.

I want to thank my advisors for their patience, support, and guidance. **Roeland**, you taught me the importance of taking a step back and thinking about the bigger picture. For your sake, but also for the sake of the Veluwe wildlife, please be careful with that e-bike. **Marcel**, in line with the Odyssey analogy, thanks for being such a good mentor all these years.

A big thank you also to **Bart**, **Robbert**, **Ruud** and **Saskia** for the huge work they do, making sure that the PRB group runs smoothly.

Certainly I owe many thanks to my friends and colleagues from the PRB group. Let me start with the people from PartyCity 5.920: **Alex**, aka Sally, aka triple OG, my office mate from day one, from whom I learned a lot about how to manage my PhD and my colleagues. **Christian**, I know you were disappointed to go from a private office to having five office mates, but I really enjoyed having you around. That being said, I have still not forgiven you for going to Wageningen instead of playing football with us that Tuesday afternoon three years ago. **Christine**, before you joined the room, I didn't know the name of the guy sitting across from me (and I have forgotten it since), but you made everyone talk to each other and were the best colleague to gossip with. Also, thanks for the Delft Global muffins. **Tom**, you must be the first master student to supervise a post doc. I've known you since my first day at TU Delft and we've had a lot of laughs, for example making fun of deep learning. **Amelia**, you stayed in Delft for only a few months, but really made your mark. You even caused **Chirag** to want to attend the biotalks. Thank you for running my jobs in the cluster :-P and for the many lessons on Spanish swear words. **Ma c***lle**, although only part-time in Delft, we've still had several hours of interesting discussions and brainstorming sessions as well as many fun times. I've learned a lot from you. **Aysun**, thanks for the constant supply of lokum(i) and our great discussions. **Mostafa**, you started your PhD and joined a room with three 4th-year students at their peak of their hate for academia. The fact that you weren't scared away showed me that a real badass is hidden behind this calm exterior.

Haley, you got the office next to PartyCity, sorry for all the noise. **Thomas**, I remember you sitting with me after some of my first lab talks and helping me see the error of my

ways, in a constructive way. Plus, you always have something funny to say. I think whatever I say about **David** and **Marco**, will be an understatement. You were always there to answer my silly math questions and also provided countless amounts of gezelligheid during Thursday borrels.

Ziqi, you are by far my favourite Korean person. **Yeshwanth** and **Attila**, thanks for the board-game nights. **Chirag**, you have many qualities, but I find it unbelievable how you can learn any dance in under 1 minute. My whole family still talks about your moves. **Jose** is a serious guy, who doesn't like to say much, but when on the football pitch, his skill speaks for itself. PR **Tom** and **Arman**, thanks for the great collaboration and all the fun we had during the NLP project. **Mo**, thanks for organizing the Bayesian reading groups and pushing all of us to get our statistics straight. **Ahmed**, I admire your humility and your attitude towards science.

Arlin, I don't know what your standards about collaborators are, but they are about to drop quite a bit. **Ramin**, second-in-command of the Plant Bioinformatics Group, I have really enjoyed our various discussions and your cooking. Given that your recent visit to Aigina was not really a success, I think we should visit a Greek island together. Do you like Kos? **Tamim**, without a doubt you are the second-best basketball player of PRB. Also, you are a great colleague and friend and I had a great time with you in the office, on the court, and during the borrels.

Thanks to the rest of the DBL people, **Nicco**, **Lucas**, **Marc** and **Henne** - who endured me during my MSc thesis -, **Joana** and **Erik** for the useful discussions, **Sven**, **Meng**, and the best T.A.-partner, **Lieke**.

Osman, thanks for the nice collaboration, the in-depth talks about Mediterranean cuisine, and your crucial position in the PRB Sports Committee. **Robert-Jan**, thanks for setting up the Discord server and teaching everyone how to use it. In the beginning I was a bit disappointed, because I was hoping to escape some meetings, but soon I realized how important this was for everyone's mental health. When **Yancong** joined the group, he was a very serious, quiet guy. Until he joined his current room, where he somehow transformed into a very social person and regular attendant of the Thursday drinks. I would like to thank everyone from the PR/CV/Social blalbla labs for the interesting discussions during the coffee talks: **Jan**, **Alexander**, **Seyran**, **Silvia**, **Nergis**, **Jesse**, **Xin**, **Yunqiang**, **Jin**, **Stephanie**, **Bob**, **Hamdi**, **Ombretta**, **Bernd**, **Marian**, **Amogh**, and **Burak**.

Colm, **Gerard**, **Skander**, **Yasin**, DBL **Stephanie**, PR **Ramin**, you joined the PRB recently and during the corona crisis. I hope we get to know each other better in the future.

Finally, thanks to the "older generation" of PhDs: **Thies**, **Wouter**, **Ekin**, **Laura**, **Taygun** (the third-best basketball player of PRB), **Sjoerd**, **Jasper**, **Wenjie**, **Yanxia**, **Yazhou**, and **Amin**. Although we didn't overlap for very long, I have very fond memories from all of you.

Several kids suffered under my supervision. I don't know if they learned something from me, I definitely learned a lot from them, for example how NOT to supervise. Special shout-out to **Rafael**, **Irene** and **Katja**. Among the rest of the kids who were walking around the past four years, I also had some good times with **Bram** and **Ekaterina**.

I would also like to thank **Keygene** and **Arjen van Tunen** for the opportunity to work on this challenging project. A big thank you to **Susan**, **Alex**, **Niek**, **Evert-Jan**, **Raymond**, **Anna**, **Rudi** and **Erwin** for welcoming me into the bioinformatics unit. Friday afternoon

coffee breaks will never be the same.

Lastly, I feel the need to thank the developers of open source libraries such as **numpy**, **scipy**, **scikit-learn**, and **pytorch** and the countless people who help others via the **stack exchange** fora.

CURRICULUM VITÆ

Stavros MAKRODIMITRIS

17-02-1991 Born in Cholargos, Athens, Greece.

EDUCATION

2008–2013 Diploma in Electrical & Computer Engineering
National Technical University of Athens, Greece

2014–2016 MSc of Computer Science – Specialization Bioinformatics
Technical University of Delft & Leiden University

2016–2020 PhD. Bioinformatics
Technical University of Delft & Keygene N.V.
Thesis: What does that gene do?
Promotors: Prof. dr. M.J.T. Reinders & Prof. dr. R.C.H.J. van Ham

AWARDS

2013 3rd prize for Best Greek Bioinformatics BSc Thesis 2012-2013,
IEEE Engineering in Medicine and Biology Society – Greek Chapter.

2019 Student Poster Prize at Crop Innovation and Business conference,
Amsterdam 2019.

List of Publications

1. A. Gastounioti, **S. Makrodimitris**, S. Golemati, N.P.E. Kadoglou, C.D. Liapis, K.S. Nikita, *A novel computerized tool to stratify risk in carotid atherosclerosis using kinematic features of the arterial wall*, IEEE Journal of Biomedical and Health Informatics 19 (3), 1137-1145, 2014.

2. **S. Makrodimitris**, R.C.H.J. van Ham, M.J.T. Reinders, *Improving protein function prediction using protein sequence and GO-term similarities*, Bioinformatics 35 (7), 1116-1124, 2019.

3. **S. Makrodimitris**, M.J.T. Reinders, R.C.H.J. van Ham, *Metric learning on expression data for gene function prediction*, Bioinformatics 36 (4), 1182-1190, 2020.

4. A. Villegas-Morcillo, **S. Makrodimitris**, R.C.H.J van Ham, A.M. Gomez, V. Sanchez, M.J.T. Reinders, *Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function*, Bioinformatics, btaa701, 2020.

5. **S. Makrodimitris**, R.C.H.J. van Ham, M.J.T. Reinders, *Automatic Gene Function Prediction in the 2020's*, Genes, 11(11), 1264, 2020.

6. E.B. van den Akker, **S. Makrodimitris**, M. Hulsman, M.H. Brugman, T. Nikolic, T. Bradley, Q. Waisfisz, F. Baas, M.E. Jakobs, D. de Jong, P. Slagboom, F.J.T. Staal, M.J.T. Reinders, H. Holstege, *Dynamic clonal hematopoiesis and functional T-cell immunity in a supercentenarian*, Leukemia, 2020.

7. **S. Makrodimitris**, M.J.T. Reinders, R.C.H.J. van Ham, *A thorough analysis of the contribution of experimental, derived and sequence-based predicted protein-protein interactions for functional annotation of proteins*, PloS one 15 (11), e0242723, 2020.

8. K.G. Schmahl, T.J. Viering, **S. Makrodimitris**, A. Naseri Jahfari, D. Tax, M. Loog, *Is Wikipedia succeeding in reducing gender bias? Assessing changes in gender bias in Wikipedia using word embeddings*, Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science, 94–103, 2020.