Master's Thesis

A comparison of supervised gene set searching algorithms for outcome prediction of breast cancer

Thesis Committee:
Prof.dr.ir. M.J.T. Reinders
Dr. L.F.A. Wessels
Ir. M.H. van Vliet
Dr. E.A. Hendriks
Dr. G.W. Klau

Author	**Raul Kooter**
Email	Raul@xs4all.nl
Student number	1150162
Thesis supervisor	Dr. L.F.A. Wessels Ir. M.H. van Vliet
Date	September 23, 2009 - 14:00

Information and Communication Theory Group [I,C)^T

TU Delft

Faculty of Electrical Engineering, Mathematics and Computer Science

Delft University of Technology

Preface

Before you lies the final report of the project I've worked on for my Master's Thesis, consisting of an article, supplementary material to the article, and a work document. In short, last year I've implemented various algorithms from literature which should improve the diagnosis of breast cancer, and compared them using a standardized evaluation protocol. This work was conducted under the roofs of both the Delft University of Technology and the Netherlands Cancer Institute.

A few acknowledgments are in order to those who have somehow helped me during this project. The ones who have assisted me the most are my direct supervisors, Lodewyk, for his feedback and for making my graduation possible in the first place, and Martin, for his feedback and for providing me with all the necessary code, data and other necessary background knowledge. My gratitude also goes out to all those at the DUT who have made the Master Bioinformatics possible. A 'thank you' is also in order to Han-Yu Chuang for personally answering my inquiries about her paper.

I would also like to thank my fellow bioinformatics students for accompanying me while being stranded with me on a deserted computer island somewhere on the 11th floor: Bas, Jelle, Jeroen, Onno, Patrick and Tisha.

Another thanks goes to the friends I've made during my Computer Science study and my time at the Christiaan Huygens study association, which are unfortunately too many to mention. A few names I would like to mention are Thomas and Gerardo, who have kept in touch since the beginning of the study and of course my CH board members: Jasper, René, Shiraz, Mike and Bas.

Finally, a big thanks to my parents and Ricardo, for supporting me always.

R.P. Kooter
September 14, 2009

> Computers can figure out all kinds of problems, except the things in the world that just don't add up.
>
> James Magary

A comparison of supervised gene set searching algorithms for outcome prediction of breast cancer

Raul Kooter [1,2], Martin van Vliet [1,2], Lodewyk Wessels [1,2]

[1]Information and Communication Theory Group, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands
[2]Bioinformatics and Statistics group, Department of Molecular Biology, Netherlands Cancer Institute, Amsterdam, The Netherlands

ABSTRACT

Motivation: Determining whether a tumor is likely to metastasize is a task that helps selecting the correct treatment for a patient. In breast cancer research, traditional classification of tumors depends on evaluating clinical risk factors, which has led to over-treatment in the past. High-throughput technologies such as mRNA microarrays have generated large amounts of data on tumors from patients, making it possible to perform classification using machine learning techniques, achieving higher accuracy than the traditional classification methods. While early methods have selected an optimal set of single genes as features, newer methods have attempted to find groups of genes that classify accurately. By combining the gene expressions according to these groups a new set of features is determined. The goal of this work is to analyze the classification performances using the latter technique.

Results: In this work various genes set searching algorithms will be reviewed on simulated data, indicating under which conditions these algorithms generate a better feature set than the original feature set. The methods are also applied on actual breast cancer data, indicating that very few of these methods are convincingly able to generate an improved feature set.

Contact: r.p.kooter@student.tudelft.nl

1 INTRODUCTION

The determination of the aggressiveness of tumors using clinical risk factors has led to less mortalities but also cases of overtreatment (van 't Veer *et al.* (2002)). By assessing the gene expression levels of a tumor using mRNA microarrays, a new set of features arises from which a more accurate prediction of metastasis development can be made. Recent research has focused on selecting a set of prognostic markers from these features, or genes, which are able to discriminate between the two classes of tumors: tumors that do and do not metastasize.

Two studies which have compiled breast cancer datasets and selected a set of prognostic markers by machine learning are those by van 't Veer *et al.* (2002) and Wang *et al.* (2005). Van 't Veer determined a 70-gene signature, while Wang determined a 76-gene signature. The set of markers determined by Van 't Veer was later evaluated on a larger breast cancer dataset by van de Vijver *et al.* (2002). While they both improved prediction accuracy, the two sets of markers only had three genes in common and a decreased performance when a set of markers of one study was tested on the dataset of the other study.

Hoping to find a more robust and accurate set of prognostic markers, recent methods have used additional data sources and exploited the structure induced by these to find groups of genes which, when their gene expression is combined, will transform into new features that are more accurate and robust predictors. The underlying hypothesis here is that the activities of pathways are more powerful predictors for cancer classification than activities of single genes, and that the newly found features will represent the activities of these pathways more accurately. These methods include the work of Chuang *et al.* (2007), that defines groups by searching for predictive subnetworks in a protein-protein interaction network; the work of Lee *et al.* (2008), that defines groups by searching predefined gene sets from databases such as MSigDB or GO for predictive subsets; and the work of Park *et al.* (2007), that uses hierarchical clustering and LASSO to define predictive groups (clusters) in a data-driven way. We will refer to these methods as *Chuang*, *Lee*, and *Park*, respectively.

Chuang's and Lee's algorithms differ from Park's in that the latter doesn't use additional predefined biological data, thus being data-driven rather than knowledge-driven. On the other hand, the approaches of Chuang and Lee may be biased by the data contained in the pathway databases or suffer from the noise in the PPI datasets. All these approaches (*Chuang*, *Lee* and *Park*) claim a performance improvement over predictors based on single genes. However, an extensive comparison of the performances of the three existing gene set searching algorithms on multiple breast cancer datasets has not been performed. In addition, none of the existing studies provide insight in the type of effect that underlies the gain in performance.

In this work a comparison and analysis will be made of algorithms which attempt to find prognostic markers by combining gene expression. A standardized and comprehensive comparison will be made between *Chuang*, *Lee* and *Park*, along with a few minor variations. Artificially generated datasets will also be employed to further analyze the methods. By doing so we hope to find out how finding prognostic markers by combining gene expression can be optimized, whether groups of genes actually improve classification significantly over single genes and how this possible improvement can be explained.

2 APPROACH

Three gene set searching algorithms, *Chuang*, *Lee* and *Park* were implemented with a few differences from their original design, along with some variations in implementation. Seven breast cancer

Table 1. Collection of datasets used in this work. The samples column indicates the number of samples that could be assigned to the poor/good group.

Source	Samples n	Notes
Wang *et al.* (2005)	286	Different normalization
van de Vijver *et al.* (2002)	248	Agilent platform
Miller *et al.* (2005)	193	Used in SOS
Pawitan *et al.* (2005)	142	Used in SOS
Desmedt *et al.* (2007)	120	Used in DMFS
Chin *et al.* (2006)	97	Used in DMFS
Loi *et al.* (2007)	120	Used in DMFS
SOS (Specific Overall Survival)	335	Miller and Pawitan
DMFS (Distant Metastasis Free Survival)	337	Desmedt, Chin and Loi

datasets were employed to evaluate the algorithms. A selection of algorithms were used in an all-against-all comparison, in which each of the algorithms was used to select markers from one dataset and tested on the remaining datasets. After performing this comparison, conclusions were drawn about the efficiency of the algorithms across all datasets. We will now give an overview of the datasets, algorithms, and evaluation procedure.

2.1 Datasets

All datasets except for the Wang dataset were preprocessed as described in van Vliet *et al.* (2008). For an overview see Table 1. As described in their paper, in order to combine six Affymetrix datasets and one Agilent dataset, the feature set was taken to be the intersecting set of Entrez identifiers common to both platforms, and the corresponding probe for each Entrez identifier was selected to be the one with the highest variance and with a '_at', '_s_at' or '_x_at' extension, resulting in 11601 features. The same normalization procedure was applied to these six datasets. Since Wang's dataset was compiled using a similar Affymetrix platform, it was easily added by selecting the corresponding Affymetrix probes, although a similar normalization procedure could not be applied since the raw data was not available.

The Miller and Pawitan datasets were combined into the SOS (Specific Overall Survival) dataset and the Desmedt, Chin and Loi datasets were combined into the the DMFS (Distant Metastasis Free Survival) dataset. This grouping of datasets was performed since these were the only clinical endpoints available for these datasets. Even though used by van Vliet *et al.* (2008), the Minn dataset was removed from the collection due to low quality. The final four datasets, Vijver, Wang, SOS and DMFS were z-normalized such that each gene had a mean of zero and a standard deviation of one.

Based on the survival time and censoring for the respective clinical endpoints, the samples in the datasets were assigned to either the good or poor outcome group. Samples were labeled 'poor' if, in the Vijver and DMFS dataset, metastasis was detected within five years or if, in the SOS dataset, death occurred within five years. Samples were labeled 'good' if no event occurred and the patient had a follow-up of at least five years. Samples which did not belong in either category were discarded. For the Wang dataset, the labels assigned were identical to the labels assigned by Chuang *et al.* (2007).

2.2 Gene sets and protein-protein interaction networks

The protein-protein interaction (PPI) network was downloaded from the supplementary information of Chuang *et al.* (2007). This PPI network is a combination of interactions derived from literature, yeast-two hybrid experiments and mass spectrometry data. The network consists of 57235 interactions for 11203 nodes represented by Entrez identifiers. The PPI network and the gene expression data had 8572 identifiers in common.

To test the pathway-based algorithms, the MSigDB C2 pathways were downloaded. Even though at the time of writing version 2.5 was available of MSigDB C2 gene sets, version 1.0, as used by Lee *et al.*, was primarily used and downloaded from an online archived version at http://www.broadinstitute.org/gsea/msigdb/. It consists of 522 pathways, with a total of 4915 genes. There is an overlap of 4554 identifiers with the gene expression datasets.

2.3 Predictor gene set searching algorithms

We define the set of genes which can either be encountered in the datasets or in the additional external data as $G = \{g_1, g_2, ..., g_{p^*}\}$. A gene expression dataset such as Vijver can be defined as $X = \{x_{i,j}\}_{p \times n}$ with labels $Y = \{y_1, y_2, ..., y_n\}, y_j \in \{0, 1\}$ and genes $G_X = \{g_1, g_2, ..., g_p\} \subset G$. Here p is the number of genes and n is the number of samples. We assume that all the datasets X are already z-normalized per gene. A predictor gene set searching algorithm attempts to find a set of predictor gene sets $W = \{W_1, W_2, ..., W_{p'}\}$, where $W_k \subset G_X$, p' denotes the total number of predictor gene sets and and p_k denotes the number of genes in the k'th gene set.

This set W can be used to perform the following mapping: $X \xrightarrow{W} X'$ where $X' = \{x'_{k,j}\}_{p' \times n}$ is the transformed dataset such that:

$$x'_{k,j} = \sum_{i \in W_k} \frac{x_{i,j}}{\sqrt{p_k}} \tag{1}$$

Predictor gene sets can come in the form of subnetworks (eg. in *Chuang*), condition-responsive genes (eg. in *Lee*) or clusters (eg. in *Park*). In order to define a gene set an algorithm may use external data such as protein-protein interaction data (PPI) or biological gene sets (GS), such as MSigDB. Note that this external data isn't restricted to using the genes represented in the dataset X, the PPI for example contains genes which aren't represented in G_X.

2.3.1 Chuang's algorithm. *Chuang* attempts to find W using a dataset X and the PPI. In a PPI network consisting of m nodes, a greedy search (see below) is performed starting from each node resulting in m predictor gene sets. A subnetwork M_k is a subset of p_k genes which form a subnetwork according to the PPI network. (Note that we will be employing W to denote the final set of gene sets produced by the gene set searching algorithm and M as a set of candidate gene sets being evaluated by the algorithm). For every subnetwork, a subnetwork activity vector a_k can be calculated using an equation similar to Equation 1:

$$a_{k,j} = \sum_{i \in M_k} \frac{x_{i,j}}{\sqrt{p_k}} \tag{2}$$

In order to find discriminative subnetworks, a score for discriminative potential (S) is introduced. This score can be derived for each possible subnetwork M_k by calculating a measure of

association between Y and a_k. Chuang *et al.* (2007) employed mutual information to calculate $S(M_k)$:

$$S_{MI}(M_k) = \sum_{x \in a'_k} \sum_{y \in 0,1} p(x,y) log \frac{p(x,y)}{p(x)p(y)} \tag{3}$$

where $p(x,y)$ is the joint probability distribution of a'_k and y, $p(x)$ and $p(y)$ are the marginalized probability distribution of $p(x,y)$ and a'_k is a discrete version of a_k, obtained by binning the values of a_k in 9 linearly spaced bins.

Subnetworks maximizing $S_{MI}(M_k)$ are found by performing a greedy search starting each of the nodes in the PPI network. At each step of the greedy search, candidates for addition include nodes which are direct neighbors to the existing subnetwork and which are within a distance of two PPI hops from the starting node. The canditate which increases $S_{MI}(M_k)$ maximally is considered for addition. If it increases $S_{MI}(M_k)$ by at least five percent, it is added to the subnetwork, otherwise the search stops.

Chuang's original implementation included three additional steps to select only the most significant subnetworks, but for a consistent comparison with the different methods, these steps are not executed in our experiments. Rather, we employ feature selection in the double-loop-cross-validation procedure to select predictive subnetworks.

An overview of *Chuang* is depicted in Figure 1.

2.3.2 Chuang algorithm.* *Chuang** is a modified version of *Chuang* designed to work on predefined biological gene sets rather than a protein-protein interaction network. Just as *Chuang*, it employs greedy forward selection and mutual information as a scoring measure. The greedy search is performed for all biological gene sets, i.e. the final number of predictive gene sets equals the number of biological gene sets.

For each biological gene set M_k, the first gene added to the predictor gene set W_k is the gene with the highest mutual information score. Then, from the remaining genes, the gene which, together with the already selected genes, maximizes the mutual information, is selected. This process terminates when the set is exhausted or the performance score does not increase.

2.3.3 Lee's algorithm. *Lee* finds W by iterating over a set of predefined biological gene sets and selecting an appropriate subset of genes in each of these gene sets, referred to as 'condition-responsive genes' (CORGs) in Lee *et al.* (2008).

In a predefined biological gene set, or pathway $P = \{g_1, g_2, ..., g_{p_P}\}$, the genes are ranked according to their t-score. For a gene $g_i \in P$, its t-score, t_i, is the Student's t-statistic measuring the association between the gene expression and the class label:

$$t_i = \frac{\mu_1 - \mu_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{4}$$

where μ_1 and μ_2 are the means of x_{i,J_0} and x_{i,J_1}, s_1 and s_2 are the standard deviations of x_{i,J_0} and x_{i,J_1} and n_1 and n_2 are the number of samples in x_{i,J_0} and x_{i,J_1} respectively. We define x_{i,J_0} and x_{i,J_1} as the vector of values of all samples with class label 0 and 1 respectively, i.e. $J_0 = \{j | y_j = 0\}$ and $J_1 = \{j | y_j = 1\}$. For

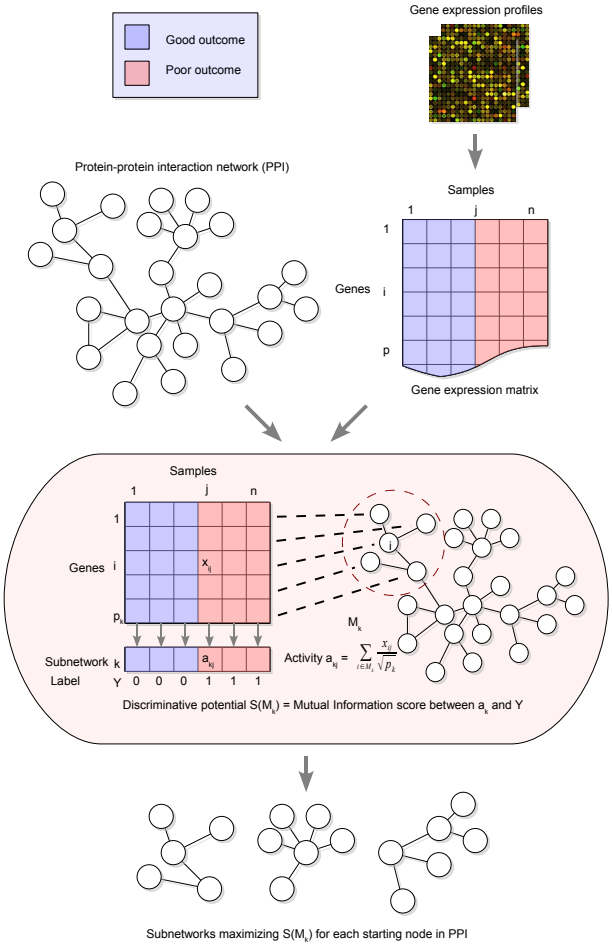


Fig. 1. Overview of Chuang's algorithm. Using the PPI network and the dataset X, a greedy search is performed at each node of the PPI network, which searches for a subnetwork maximizing the mutual information between the network activity and the class label. Shown in the figure is a single step of the greedy search where the mutual information is calculated for an intermediate subnetwork. At each step of the greedy search, the neighboring node that maximizes the mutual information is considered for addition. Only neighboring nodes which lie within a distance of two PPI links from the starting node are considered. If the mutual information increases by at least five percent, the associated node is added to M_k. This figure is adapted from Chuang *et al.* (2007).

a pathway P, its average t-score is:

$$t_{avg} = \frac{t_i}{p_P} \tag{5}$$

If $t_{avg} < 0$, the genes in P are re-ordered such that $t_i \leq t_{i+1}, i = 1, 2, ..., p_P - 1$, otherwise the genes are ordered according to descending t-score: $t_i \geq t_{i+1}, i = 1, 2, ..., p_P - 1$.

Given this ranking, a subset M_k of P can be defined as the first k genes $M_k = \{g_1, g_2, ..., g_k\}$. Similar to *Chuang*, for such a subset M_k an activity vector can be calculated using Equation 2. The discriminative score $S(M_k)$, however, is calculated using the absolute value of the t-score. Only one of these subsets is returned as

being the CORG set of pathway P, in this case the smallest possible k for which $S(M_{k+1}) \leq S(M_k)$.

Lee's original implementation included steps to select the most promising subset of pathways to feed into the algorithm and a filtering step to select only the most significant subset of CORG sets returned by the algorithm. These selection steps are not included in our experiments.

For an overview of *Lee*, see Figure 2.

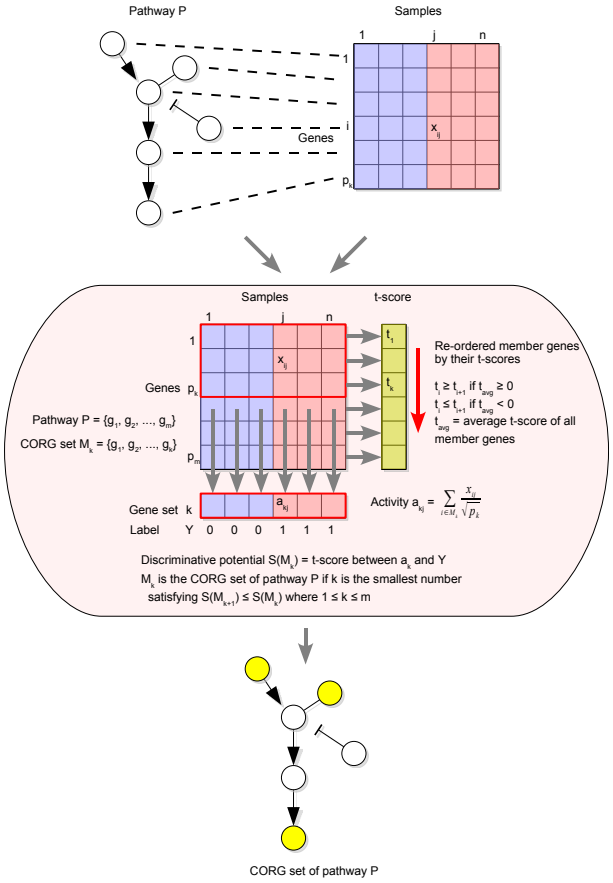


Fig. 2. Overview of Lee's algorithm. Shown here is the algorithm to determine the condition-responsive gene set (CORG set) given a pathway P and its corresponding gene expression dataset. For each node in the pathway P, the t-score (t_i) can be calculated which measures the capability of the corresponding gene to discriminate between the two phenotypes. All genes in the pathway are ordered based on their t-scores. After ordering, a gene set M_k consists of the genes 1 through k. The combined z-score of the individual genes of the gene set make up the activity vector a_k. For a pathway P consisting of m genes, m possible gene sets are considered, each with a discriminative potential $S(M_k)$. The smallest gene set with the first maximum discriminative score is taken to be the CORG set of the pathway P. This algorithm can be repeated for a set of pathways to determine a set of CORGs. This figure is adapted from Lee *et al.* (2008).

2.3.4 Lee(PPI).* To see how well the *Lee* search approach would perform in combination with the PPI data, a set of gene sets were generated (from the PPI). We will refer to these gene sets as $PPI*$. This set of gene sets was generated by iterating over every node in the PPI and defining a gene set as all genes corresponding to PPI nodes within a maximal distance of two from each starting node. Similar to *Chuang*, the nodes which had more than 300 connections were ignored. This helps keep the gene sets in $PPI*$ reasonably small. $PPI*$ consists of 11203 gene sets.

2.3.5 Park's algorithm. In contrast to the previous methods, *Park* does not use additional biological data to find predictor gene sets, but attempts to find them by analyzing the structure induced by the gene expression.

Park performs hierarchical clustering of the genes of the dataset X, using average linkage and Pearson correlation as distance measure. The resulting dendrogram can be cut at p possible levels. Each cut results in a set of clusters. Let's define the j'th cluster at level i as $M_{i,j}$. At the lowest cut off level, $i = 1$, there is one cluster consisting of all genes: $M_{1,1} = \{g_1, g_2, ..., g_p\}$ and at level $i = p$, p clusters are returned, each consisting of a single gene: $M_{p,1} = \{g_1\}, M_{p,2} = \{g_2\}, ..., M_{p,p} = \{g_p\}$. At level i, the clusters are given by $M_i = \{M_{i,1}, M_{i,2}, ..., M_{i,i}\}$. Given the clusters obtained at a given level, Equation 2 is employed to compute the predictor gene set activities, resulting in the activities at level i, given by $a_i = \{a_{i,1}, a_{i,2}, ..., a_{i,i}\}$.

At a level i, the performance of the clusters M_i can be assessed using cross-validation. At every step of this cross-validation, the gene sets are ranked based on the t-score of their gene set activities, a_i, and trained using a Nearest Mean Classifier on the training subset and evaluated on the testing subset, returning a learning curve. These learning curves are combined into a single learning curve l_i across all folds of the cross-validation procedure: $l_i = \{l_{i,1}, l_{i,2}, ..., l_{i,i}\}$ where $l_{i,j}$ is the AUC corresponding to the performance of the best j gene sets. The minimum error over all learning curves indicates the optimum cut off level. The clusters at this optimum cut off level are returned as W. Notice that this algorithm, unlike *Chuang* and *Lee*, has the property that every gene is represented only once in W.

This implementation differs from Park *et al.* (2007) in several ways. First of all, in the original implementation, LASSO was used as the method to both select the optimal cut off level and number of features. The original implementation also selected the 3017 most significant genes before applying the algorithm. For a more detailed overview, see also Figure 3.

2.4 Algorithm comparison

To see how well a set of markers extracted from one dataset, X_{train}, would perform on a second dataset, X_{test}, the second dataset would be transformed using the set of markers W_{train} and subjected to a double-loop-cross-validation (DLCV) procedure to get a set of AUC performance scores v. See part (a) in Figure 4.

A selection of gene set searching algorithms was made and were applied to the four datasets (Vijver, Wang, SOS and DMFS). Each of these marker sets were tested on each of the remaining three datasets. For a single gene set searching algorithm, S, these performance scores may be concatenated to get a larger set of performance scores v_S. See part (b) in Figure 4 (note that in Figure 4 we restrict ourselves to three datasets).

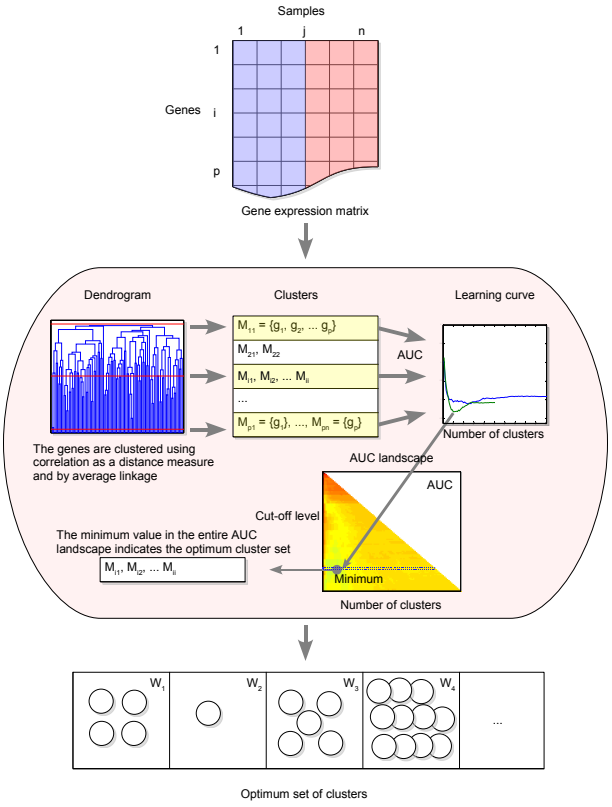


Fig. 3. Overview of Park's algorithm. The genes of a gene expression dataset are clustered by Pearson correlation and average linkage. The resulting dendrogram can be cut off at m levels, meaning that there are m possible ways in which the m genes can be split up. Given a clustering $M_i = \{M_{i,1}, ..., M_{i,i}\}$, a cross-validated learning curve l_i can be generated. The global minimum error found in all l_i's indicates the preferred cut off level, the algorithm returns the corresponding M_i. For Park's original implementation, see Park *et al.* (2007).

If four datasets would be used for selection and testing of the markers, and the DLCV procedure returns 100 AUC scores, then the entire procedure would yield 1200 AUC values per gene set searching algorithm. To test whether one algorithm outperforms another algorithm, a one-sided paired Wilcoxon test was performed on the set of AUC values. The resulting p-value would indicate whether one algorithms outperforms the other. See part (c) in Figure 4.

After inspection of the AUC values, it appeared the Wang dataset produced significantly lower AUC values, regardless of which set of markers were used. Therefore the set of prognostic markers produced by Wang and the Wang dataset were removed from the analysis, leaving us with 600 AUC values per algorithm.

The algorithms used in the comparisons were *Chuang*, *Lee*, and *Park*. The case where no actual gene set searching algorithm was used, but all single gene markers were used is indicated by *Singles*, which results in 11601 markers. Also, the variation *Chuang** and *Lee (PPI*)* were added. To see if the gene sets produced by *Park* could be improved we fed them into a gene set searching algorithm to obtain the variations *Chuang* (Park)* and *Lee (Park)*. Note that such a variation, e.g. *Lee (Park)*, uses the same dataset to run both *Park* and *Lee (Park)*. Finally, to see whether using the t-score instead of the mutual information would make a large difference, a variation of *Chuang (PPI)*, termed *Chuang (PPI, T-score)* was also used.

To test the predictive power of a set of prognostic markers derived from a dataset, a double-loop-cross-validation (DLCV) procedure is employed, similar to the one described in Wessels *et al.* (2005). We used a Nearest Mean Classifier (NMC) using Euclidian distance measure and the Area Under the Curve (AUC) of the ROC as a scoring measure. A detailed description of the DLCV procedure can be found in the Supplementary Material Section 1.

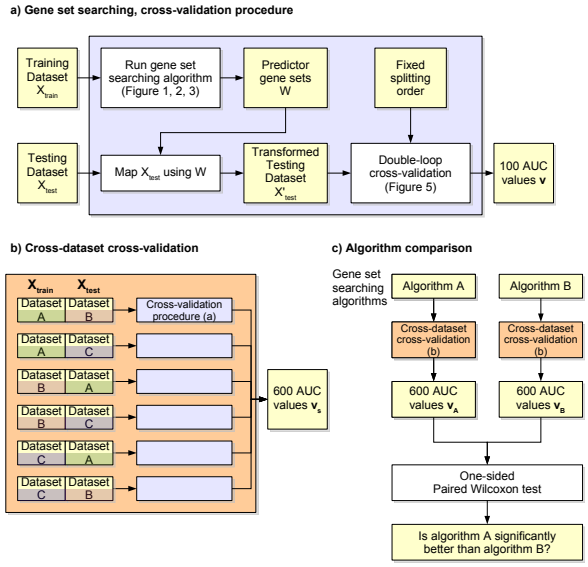


Fig. 4. Algorithm evaluation procedure. (a) The cross-validation procedure checks how well a set of markers trained on one dataset performs on a second set. (b) The cross-dataset cross-validation procedure uses every dataset in turn as a training and testing dataset to return a measure the overall performance of a gene set searching algorithm. This example shows a cross-dataset cross-validation for three datasets, A, B and C. (c) Algorithms are compared by subjecting the cross-dataset performance values to a one-sided paired Wilcoxon test.

2.5 Artificial data

In order to analyze the behavior of the gene set searching algorithms, artificial datasets were generated to evaluate both the basic properties of combining gene expressions and the efficiency of these algorithms. A few criteria had to be met while designing these models: 1) the data had to have relatively many genes while having few samples to simulate the small sample size problem, 2) about ten percent of the genes should carry signal predictive of the outcome, and 3) the data had to simulate the notion of pathways whereby combining gene expression could theoretically result in improved performance. Two models were designed: a linear model

and a logical model. We expect the gene set searching algorithms to work better than just using all single genes on the logical model. *Park*, on the other hand, should perform better on the linear model, since a large degree of redundancy (correlation) is present between genes - a feature exploited by *Park*.

2.5.1 The linear model. The linear model generates of a set of 100 latent variables $H = \{h_1, h_2, ..., h_{100}\}$, a set of 1000 genes $X = \{x_1, x_2, ..., x_{1000}\}$ and a binary outcome variable y, see Figure 5. The genes and the outcome variable have a linear dependence on the latent variables and this dependency is corrupted by adding independent noise. The latent variables are sampled from a normal distribution with parameters μ_{h_i} and $\sigma^2_{h_i}$. The expression of the genes are generated according to $x_i = \beta_i H + \epsilon$, where ϵ is a noise variable sampled from a normal distribution with mean $\mu_{\epsilon_i} = 0$ and variance $\sigma^2_{\epsilon_i}$, and b_i is a vector of real coefficients. The outcome variable y is modeled similarly, but is discretized using $y = sign[\beta_y H + \epsilon]$ where $sign(x) = 0$ if $x < 0$ or $sign(x) = 1$ if otherwise. The outcome variable also has added noise $\epsilon_y \sim N(\mu_{\epsilon_y} = 0, \sigma^2_{\epsilon_y})$.

For our experiments, the linear model θ_1 is employed, with the default values for the parameters, which can be varied depending on the experiment:

$$h \sim N(\mu = 0, \sigma^2 = 1)$$

$$y = sign[h_1 + ... + h_{10} + \epsilon]$$

$$x_i = \begin{cases} h_1 + \epsilon & \text{if } i = 1, ..., 10 \\ h_2 + \epsilon & \text{if } i = 11, ..., 20 \\ \vdots & \\ h_{100} + \epsilon & \text{if } i = 991, ..., 1000 \end{cases}$$

$$\epsilon \sim N(\mu = 0, \sigma^2 = 1)$$

The linear model is depicted in Figure 5.

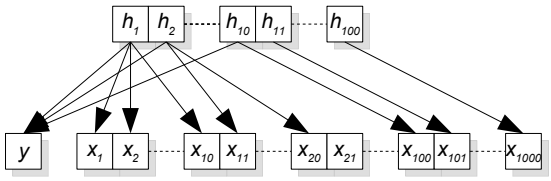


Fig. 5. The linear model θ_1.

In this model, the output (class label) only depends on a limited number of latent variables (h_1, ..., h_{10}). One can think of these hidden variables as 'hallmarks' that are contributing to cancer formation. The genes are organized in correlated groups where all genes in a group depend on the same hidden variable. The genes serve as a read-out of the hallmarks. The model contains 900 noise genes which are independent of the class label.

To get an idea of the actual distribution of the data produced by the linear model, a simplified version of θ_1 with less noise and where y depends on fewer latent variables is shown in Figure 7.

2.5.2 The logical model. The logical model generates 101 latent variables $H = \{h_0, h_1, ..., h_{100}\}$, 1000 genes $X = \{x_1, x_2, ..., x_{1000}\}$ and a binary outcome variable y. We also introduce an extra set of variables, which are the gene activities $X' = \{x'_1, x'_2, ..., x'_{1000}\}$, and the outcome activity y'. This model is designed to be more complex than the linear model (it contains non-linear effects), but also designed to benefit from combining gene expression. The latent variables are sampled from a Bernoulli distribution such that $Pr(h_i = 1) = p_i = 1 - Pr(h_i = 0)$. A gene activity, which is meant to simulate whether a gene is activated or not, is modeled using a Boolean function of the latent variables, $x'_i = f_{Boolean,i}(H)$, as is the outcome activity $y' = f_{Boolean,y}(H)$. These discrete variables are transformed into continuous variables using two normal distributions. That is, if $x'_i = 0$, then $x_i \sim N(\mu_{x'_i=0}, \sigma^2_{x'_i=0})$, and if $x'_i = 1$, then $x_i \sim N(\mu_{x'_i=1}, \sigma^2_{x'_i=1})$. Outcome variable y is modeled similarly, but discretized similarly to the linear model such that $y \in \{0, 1\}$. So if $y' = 0$, then $y \sim sign[N(\mu_{y'=0}, \sigma^2_{y'=0})]$, and if $y' = 1$, then $y_i \sim sign[N(\mu_{y'=1}, \sigma^2_{y'=1})]$.

Just as for the linear model, a logical model θ_2 is designed which has a set of default values. Formally, the logical model θ_2 is defined as:

$$Pr(h = 0) = 0.5, Pr(h = 1) = 0.5$$

$$y' = h_0$$

$$y \sim \begin{cases} sign[N(\mu = -1, \sigma^2 = 1)] & \text{if } y' = 0 \\ sign[N(\mu = 1, \sigma^2 = 1)] & \text{if } y' = 1 \end{cases}$$

$$x'_i = \begin{cases} h_0 \wedge h_i & \text{if } i = 1, ..., 100 \\ h_{i-100} & \text{if } i = 101, ..., 200 \\ h_{i-200} & \text{if } i = 201, ..., 3000 \\ \vdots & \\ h_{i-900} & \text{if } i = 901, ..., 1000 \end{cases}$$

$$x_i \sim \begin{cases} N(\mu = -1, \sigma^2 = 1) & \text{if } x'_i = 0 \\ N(\mu = 1, \sigma^2 = 1) & \text{if } x'_i = 1 \end{cases}$$

The logical model is depicted in Figure 6.

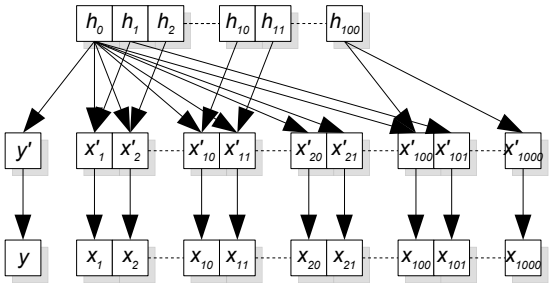


Fig. 6. The logical model θ_2.

A somewhat simpler version of the logical model, where less noise is added and the means of the bimodal distributions are further apart, can be found in Figure 7. Even though the parameters are modified, the notion of logical dependence in this simplified model remains the same, as seen in the left-most panel in the second row, where the output seems to be a logical AND of x_1 and x_2.

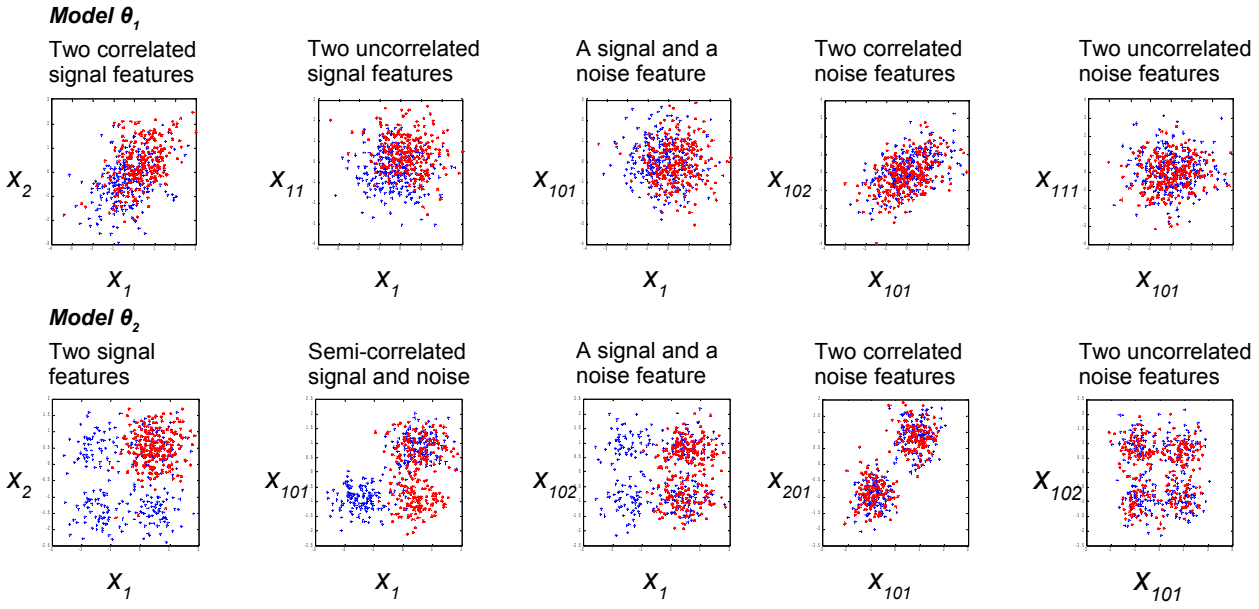


Fig. 7. Scatterplots of instances of the linear model and the logical model. These models are simplified versions of the models shown in Figure 5 and Figure 6, where the parameters are modified to give a better idea of the distribution. Shown in these scatterplots are the various combinations that two genes from the model can have. Signal genes are genes x_i with $i \leq 100$ that contribute to the outcome variable y, while noise genes are genes x_i with $100 < i \leq 1000$ that don't contribute to y. Notice that the logical model is able to model pathway activity as truly activated or deactivated. From a biological point of view, this model may be considered as modeling the pathway activities using the latent variables, while the activation of each gene is some function of the pathway activities.

2.5.3 Artificial gene sets. In order to evaluate how well gene set searching algorithms that depend on predefined data perform on the artificial data, we generated artificial gene sets. For example, *Lee* needs a set of gene sets as input to determine the CORGs. Since we can't use the MSigDB pathways as are previously used for *Lee*, we define our own gene sets.

Of course, the performance of a gene set searching algorithm might just depend on the information given by such a predefined gene set. That's why for our experiments we will consider multiple possible gene sets. The gene sets we will use are based on the design of model θ_1. These sets are:

- **Correlated**. This set consists of 100 gene sets. All genes that are correlated (that are dependent on the same latent variable) are combined.
  $M_1 = \{g_1, g_2, ...g_{10}\}$
  $M_2 = \{g_{11}, g_{12}, ..., g_{20}\}$
  ...
  $M_{100} = \{g_{991}, g_{992}, ..., g_{1000}\}$.
- **Correlated, signal only**. This set consists of 10 gene sets. Same as the 'Correlated' gene sets, but only the signal genes are involved.
  $M_1 = \{g_1, g_2, ...g_{10}\}$
  $M_2 = \{g_{11}, g_{12}, ..., g_{20}\}$
  ...
  $M_{10} = \{g_{91}, g_{92}, ..., g_{100}\}$.
- **Uncorrelated**. This set consists of 100 gene sets. In every gene set, the genes don't depend on the same latent variable h.
  $M_1 = \{g_1, g_{11}, ...g_{91}\}$
  $M_2 = \{g_2, g_{12}, ..., g_{92}\}$
  ...
  $M_{100} = \{g_{910}, g_{920}, ..., g_{1000}\}$.
- **Uncorrelated, signal only**. This set consists of 10 gene sets. Same as the 'Uncorrelated' gene sets, but only the signal genes are involved.
  $M_1 = \{g_1, g_2, ...g_{10}\}$
  $M_2 = \{g_{11}, g_{12}, ..., g_{20}\}$
  ...
  $M_{10} = \{g_{10}, g_{20}, ..., g_{100}\}$.
- **Mixed**. This set consists of 100 gene sets. In every gene set, both signal and noise genes are combined.
  $M_1 = \{g_1, g_{101}, ...g_{901}\}$
  $M_2 = \{g_2, g_{102}, ..., g_{902}\}$
  ...
  $M_{100} = \{g_{100}, g_{200}, ..., g_{1000}\}$.

A graphical representation of these gene sets can be found in the Supplementary Figure 11.

2.5.4 Simulation workflow. For each experiment, the simulation parameters defining the models θ_1 and θ_2 were used to create a set of five artificial datasets, each consisting of 300 samples. These datasets were normalized to have a mean of zero and variance of one per gene. After that, they were subjected to a cross-dataset-cross-validation protocol as shown in Figure 4(b) which returns five possible predictor sets and (5x4=)20 average AUCs.

3 RESULTS AND DISCUSSION

3.1 Real data results

Application of the procedure outlined in section 2.4, for *Singles*, *Chuang (PPI)*, *Chuang (PPI, T-score)*, *Lee (C2 V1.0)*, *Chuang* (C2 V1.0)*, *Lee (PPI*)*, *Lee(Park)*, *Chuang(Park)* and *Park* and the three datasets SOS, DMFS and Vijver resulted in the results presented in Figures 8 and 9. From the performances on the test sets it is clear that in some cases network-based approaches perform significantly better than single singles, while in some cases the performance is (slightly) worse than single genes. To get a more precise indication of the relative performances, the head-to-head statistical comparisons depicted in Figure 9 are very useful. In this figure, whenever a method X significantly outperforms method Y, this is indicated by a box colored red in row X and column Y. Figure 8 presents the results of a detailed comparison of the algorithms. This figure summarizes the marker properties such as subnetwork size and the AUC performances on the test set.

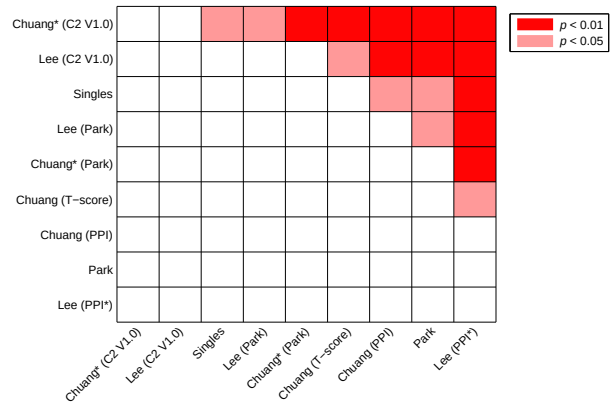


Fig. 9. Comparisons of the various algorithms. Markers were found using the SOS, DMFS or Vijver dataset and tested on the remaining 2 datasets. A p-value of the paired one-sided ranksum test between two sets of AUC values was used to assess the significance. Boxes colored (light) red mean that the algorithm indicated in the row performed significantly better than the algorithm indicated in the column.

A comparison with more variants of the algorithms included, as well as a version where Wang is included in the analysis can be found in the Supplementary Figure 3 and Figure 4.

The two methods that top the list are *Chuang** and *Lee*, which both find the predictive gene sets within the MsigDB C2 pathway, version 1.0. When the *Chuang** and *Lee* gene set searching algorithms are applied to the PPI derived gene sets the performance is significantly worse. Apparently, the use of the C2 pathways works better than using the PPI network. A possible explanation for this effect could be that the number possible gene sets in the PPI ($\sim$ 10000 sets) is much larger than the number of possible gene sets ($\sim$ 500 sets) in the MSigDB database, and that noise genes are selected during the training process.

To get an idea of the effect of the number of features, the evaluation procedure was repeated using only the top-ranking markers. For *Singles*, *Lee*, *Chuang* and *Park*, the top 10, 50, 100, 200 and 500 markers where identified by evaluating the t-score of these markers using the training datasets. Subsequently, only these top gene sets (markers) were employed in the cross-validation procedure on the test set. The results of this comparison is presented in the Supplementary Figure 6. According to this comparison, the overall ranking of the methods remained fairly constant (from stronger to weaker): *Lee*, *Singles*, *Chuang*, *Park*. Also, overall the method seem to perform best when all markers are used. The smaller the number of markers used, the worse the performance.

Furthermore, even though *Park* seems to perform poorly in the results, when the *Park* clusters results are fed in the *Lee* algorithm, it outperforms the original *Park* algorithm. This indicates that *Park* includes too many genes in the clusters it creates.

None of the methods convincingly outperform the single genes case, with the exception of *Chuang**, but only at $p < 0.05$ and not at $p < 0.01$.

Judging from the classification results when the markers and testing datasets are split (see Supplementary Figure 5), the ranking of the algorithms is not consistent. For example, when the markers from Vijver are tested on DMFS, *Park* strangely turns out to be the top-ranking algorithm, but when the Vijver markers are tested on SOS, *Park* becomes the lowest-ranking algorithm. Also, judging from the comparisons separated according to the training and testing datasets, the ranking of algorithms seems to be mainly depending on the testing dataset.

Taking a more in-depth look at the effect of the search algorithm, we note that *Chuang** has forward feature selection incorporated in the gene set search while *Lee* finds gene sets using sequential selection. In Supplementary Figure 4, when comparing *Chuang* (C2 V2.5, T-score)* against *Lee (C2 V2.5)*, we notice that *Chuang** ranks higher, but it doesn't significantly outperform *Lee*. However, *Lee (C2 V1.0)* ranks higher than *Chuang* (C2 V1.0, T-score)*, but again doesn't significantly outperform the other. Our data therefore does not suggest that forward feature selection outperforms sequential selection in the gene set searching algorithms.

Next, we analyze whether the scoring method, such as mutual information or t-score has an effect. In Supplementary Figure 4, we compare *Chuang* (C2 V2.5, T-score)* against *Chuang* (C2 V2.5)*, from which it is apparent that using the t-score works significantly better than the mutual information score, at $p < 0.05$, however in Supplementary Figure 3, where Wang is included, the significance is lost. Also, even though *Chuang (PPI, T-score)* ranks higher than *Chuang (PPI)*, this improvement is not significant.

All in all, in order of importance, the results seem to be mostly influenced by 1) the dataset on which the markers are tested, 2) the external data used in marker searching algorithm 3) the searching algorithm.

3.2 Artificial data results

In this section we describe several experiments on the artificial datasets generated with models θ_1 and θ_2. The purpose of these experiments is to shed light on the gene set searching algorithms and characteristics of the datasets which might help explain the performance of the algorithms on the real data. The first experiment is designed to inspect what happens to the DLCV performance when noise features are added to the marker set. In our real data experiments, we have relaxed the restriction that only the most

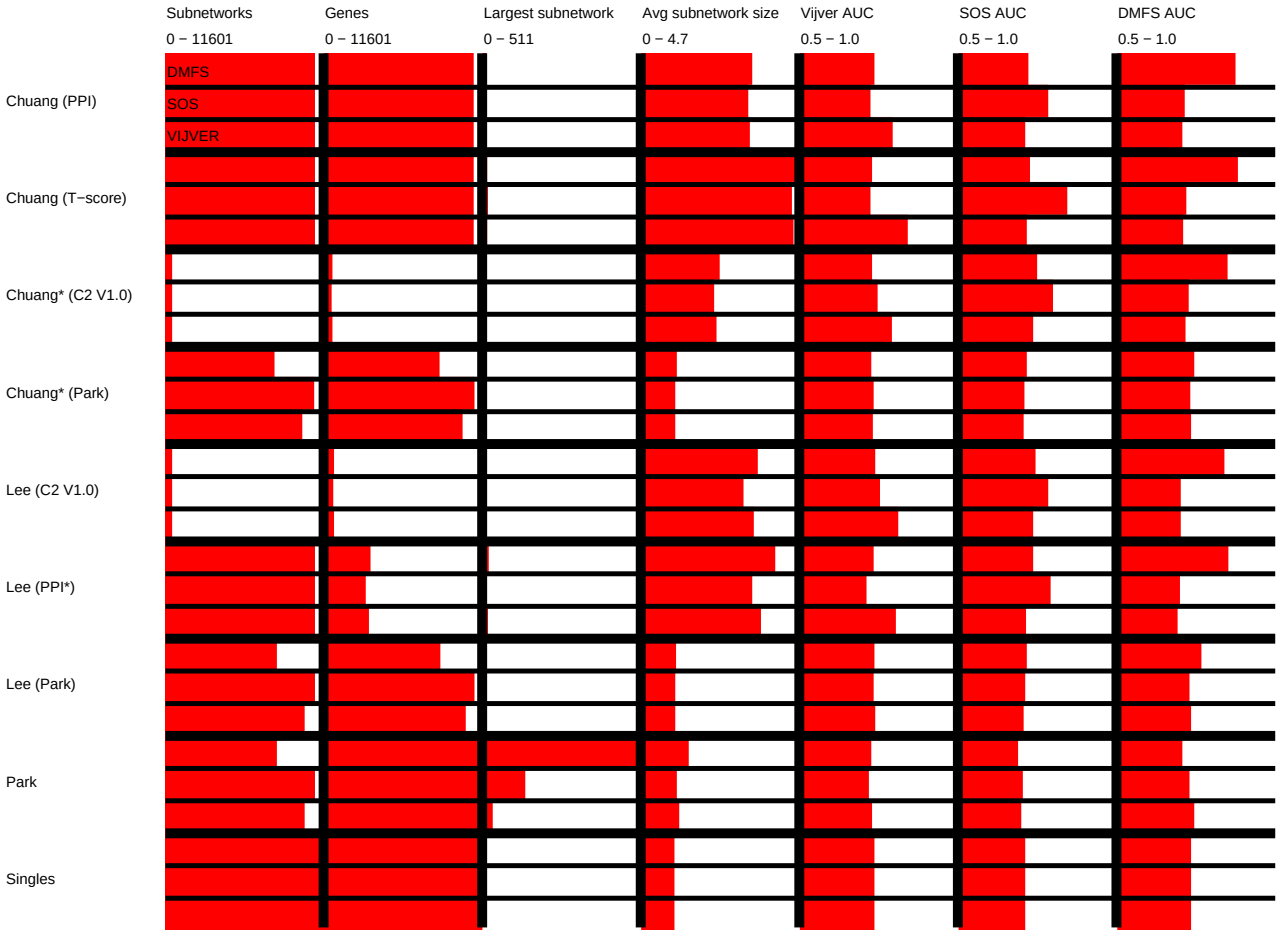


Fig. 8. This figure summarizes the results obtained for all the algorithms on the three datasets. For each gene set searching algorithm, the properties are separated in three rows, each indicating the properties obtained for the DMFS, SOS and Vijver dataset. The area colored red indicates the relative value lying within the boundaries given in the top of the column. 'Subnetworks': the total number of markers found by an algorithm. 'Genes': the total number of genes involved in the union of all the markers. 'Largest subnetwork': the total number of genes in the marker that combines the most genes. 'Avg. subnetwork': the average number of genes combined in the markers. The last three columns indicate the performance of the markers found on either the Vijver, SOS or DMFS dataset, as described in Section 2.4.

significant markers should be used, but instead used all the markers returned by an algorithm, and we have let the DLCV procedure take care of selecting the best markers.

3.2.1 Experiment 1. In our first experiment, for both θ_1 and θ_2 we add genes (x_i) based on their index (i).

So, the first 100 features are signal features, after which only noise features are added. So, no real gene set searching algorithm was used, but instead we've implemented a method that just selects the first n features, regardless of the information in the training set. For the results, see Figure 10.

As can be seen from the figure, both models indeed have a peak performance at 100 features, the point at which all signal features are included. After adding more noise features however, the performance of model θ_1 worsens, in contrast to model θ_2. This is probably due to a better contrast in t-scores of the features in model θ_2, making it easier for the DLCV to select the proper reporter set.

3.2.2 Experiment 2. The second experiment is to verify whether combining genes in a gene set by combining the expression values of these genes really improves the performance. For an overview of the experiment, see Figure 12.

For the first three parts of this experiment we only employ model θ_1 and only the signal genes, i.e. $x_1, x_2, \ldots, x_{100}$.

In the first part we compare the classification performance obtained by two different approaches. In both approaches we evaluate the classification performance (AUC) when the number of genes employed to classify is increased in groups of ten. In other words, we compute the performance obtained with 10, 20, 30 up to 100 genes. However, the genes are entered in the order implied by their indices, i.e. the first group consists of x_1, x_2, ..., x_{10}, the second group of x_{11}, x_{12}, ..., x_{20} etc. This implies that at each step we add all the genes associated with a hidden variable. In the first approach, we enter the genes as single features, i.e. the dimensionality of the problem increases in steps of 10 from 10-dimensional to 100-dimensional. This approach is

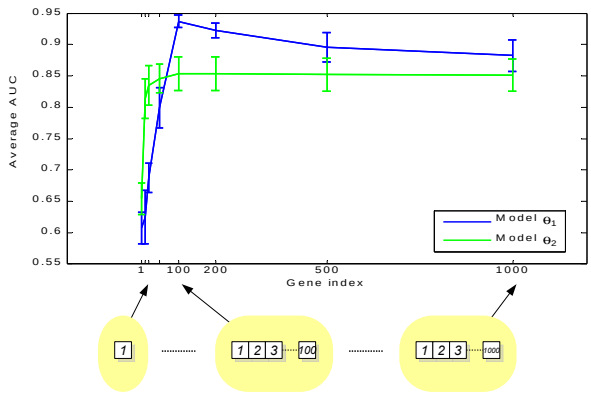


Fig. 10. For both artificial models, multiple datasets were generated, after which the average AUC performance was calculated using cross-dataset-cross-validation. At gene set searching algorithm here is merely taking the first n features in order of index. After adding the first 100 signal features, only noise features are to be added.

referred to as 'not combining correlated features' in Figure 12(a). In the second approach, we combine the genes associated with a hidden variable prior to computing the performance, i.e. we define a new feature space of 'meta-genes', z_1, z_2, ..., z_{10}, where $z_k = \sum_{i=1}^{10} x_{(k-1)*10+i}/\sqrt{(10)}$. This implies that the dimensionality of this problem is one-tenth of the dimensionality of the first approach. The second approach is referred to as 'combining correlated features' in Figure 12(a) and the results obtained with, for example, z_1, z_2 and z_3 are depicted at the x-axis position of 30, denoting the number of genes combined in Figure 12(a). It is surprising to observe from these results that combining of correlated features does not improve the performance at all.

In the second part the experimental setup is exactly the same as the first part, except for the order in which the features are evaluated. While all features associated with a hidden variable were entered simultaneously in the first part, here we enter a set of 10 uncorrelated features at each step. For example, in Step 1, x_1, x_{11}, x_{21}, ..., x_{91} are evaluated as single (not combining uncorrelated features) or after being combined (combining uncorrelated features) as meta-gene. In Step 2 x_2, x_{12}, x_{22}, ..., x_{92} are evaluated, etc. In contrast to the first part where the number of hidden variables being represented by the genes gradually increase from a single hidden variable to all ten at the end, in the second part, we enter genes associated with all hidden variables at each step. Since all hidden variables are required to predict the output (y), this implies that the algorithm theoretically has full information available at each step. This is clearly manifested in the superior performance obtained by (not) combining uncorrelated features over combining correlated features. (See Figure 12(c) where the curves are represented in the same set of axes.) In contrast to combining correlated features, combining uncorrelated features is significantly better than not combining uncorrelated features.

In the third part all signal genes are always employed to evaluate the performance. However, at each step of the process a subset of genes is combined prior to the evaluation of the classification performance. For example, at Step 3, z_1, z_2 and z_3 are obtained by combining $\{x_1, x_2, ..., x_{10}\}$, $\{x_{11}, x_{12}, ..., x_{20}\}$ and $\{x_{21}, x_{22}, ..., x_{30}\}$ respectively, while the rest of the genes are entered as single variables. A similar procedure is followed for the uncorrelated genes. In these experiments full information on the output is always available to the classifier, we merely check under which conditions combining genes helps. Figure 12(d) clearly shows that combining correlated genes does not result in any advantage, while combining uncorrelated genes does result in a performance increase. These results are concordant with the results presented in Figure 12(a-c).

For the last part of this experiment the noise features are also included in the analysis, and the procedure employed in the third part is repeated for this enlarged set. Note that only the signal genes are involved in uncorrelated and correlated combining. Considering the performance curves in Figure 12(d) we observe that now combining of uncorrelated features clearly helps, since it enables the classifier to better distinguish the signal genes from the noise genes. However, combining uncorrelated features is much more advantageous.

3.2.3 Experiment 3. Finally, *Park*, *Lee* and single gene features were evaluated on model θ_1 using various values for the noise added to the features, σ_x. The results are shown in Figure 13.

As can be seen from Figure 13(a), *Park* indeed gives a significant improvement in performance in model θ_1 over single genes, but only if the noise is sufficiently low so *Park* can build and select the proper marker set.

According to Figure 13(b), using mixed sets results in the worst performance. *Lee* ('Correlated, signal only') performs worse than *Lee* ('Uncorrelated, signal only'), which is in accordance with Experiment 2. *Lee* ('Correlated') with noise performs worse than *Lee* ('Correlated, signal only') as expected, but also performs worse than single genes for low noise levels. *Lee* ('Uncorrelated, signal only') performs best, as expected, since there is no noise and we combine genes in sets which work best according to Experiments 2. *Lee* ('Uncorrelated') with noise works worse for higher noise levels but similar for low noise levels, as expected, but still significantly outperforms the single genes. From these observations we conclude that good quality gene sets can help if it contains independent information, poor quality gene sets can hurt the performance. Uncorrelated gene sets result in a performance significantly better than single genes.

In model θ_2, as expected, *Park* doesn't give an improvement, see Figure 13(c).

In Figure 13(d) we observe that *Lee* ('Uncorrelated') shows a large significant improvement over single genes, which, in turn, outperforms mixed gene sets.

In conclusion, in both models, *Lee* results in improvement given good sets. *Park* only results in a possible improvement in model θ_1.

4 CONCLUSION

We were surprisingly unable to reproduce the previous results in such a way that the gene set searching algorithms (as proposed by Chuang *et al.* (2007) and Lee *et al.* (2008), i.e. *Chuang (PPI)* and *Lee (C2 V1.0)*) would outperform single gene approaches. In fact, single genes significantly outperformed *Chuang (PPI)* ($p < 0.05$) but were not significantly outperformed by *Lee (C2 V1.0)*. Various explanations may be given for this negative result: by omitting

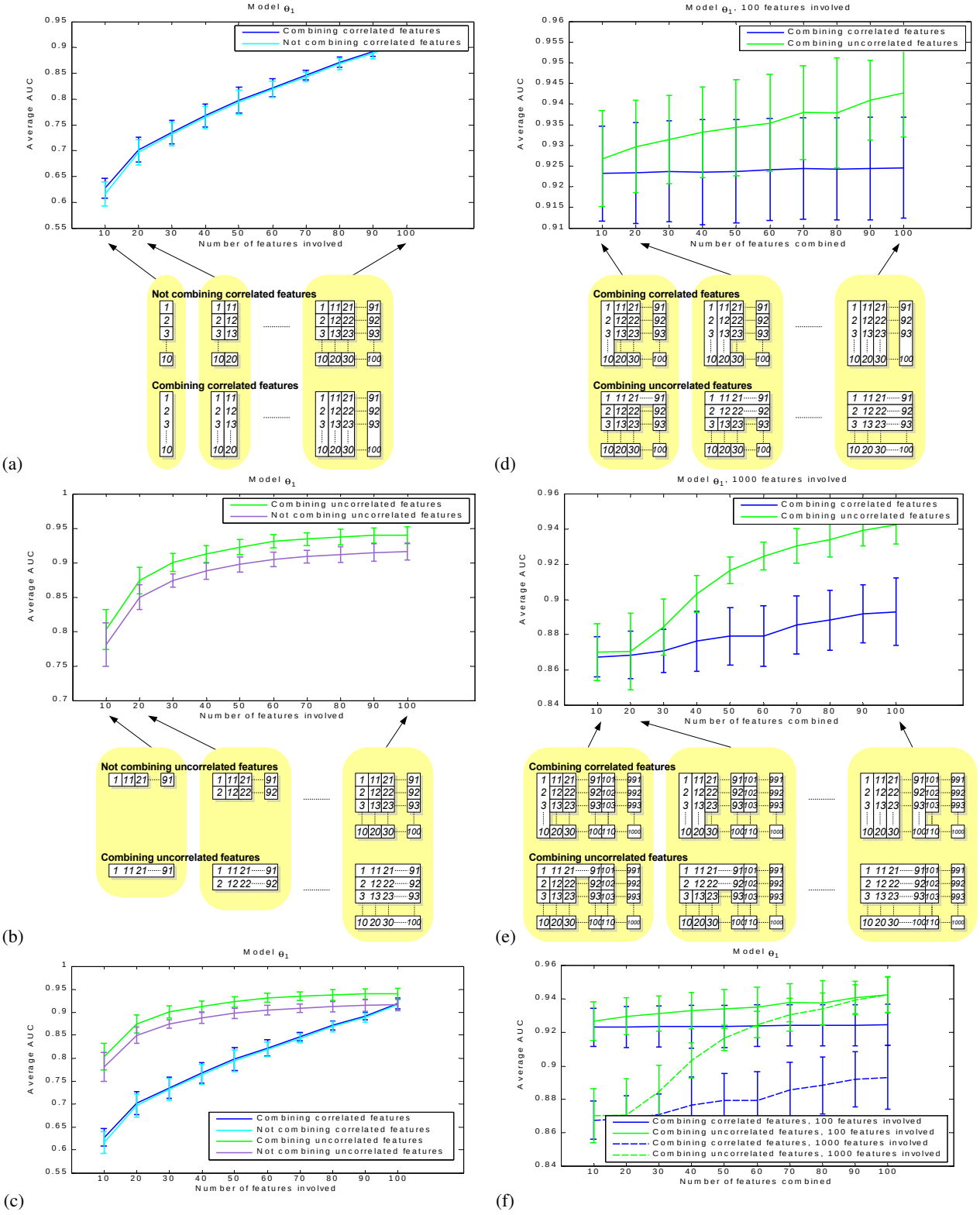


Fig. 11. The numbers in the boxes below the graphs refer to the gene indices. All indices in a box are combined to serve as input to the algorithms. For example, in (a), the genes x_1, x_2, ..., x_{10} are employed to calculate the leftmost point of the 'Not combining correlated features' curve, while $z_1 = (x_1 + x_2 +, ..., +x_{10})/\sqrt{(10)}$ is a single meta-gene which is used to calculate the leftmost point of the 'Combining correlated features' curve. All genes (or combinations of genes) on a yellow region are employed as input to the corresponding on the x-axis.

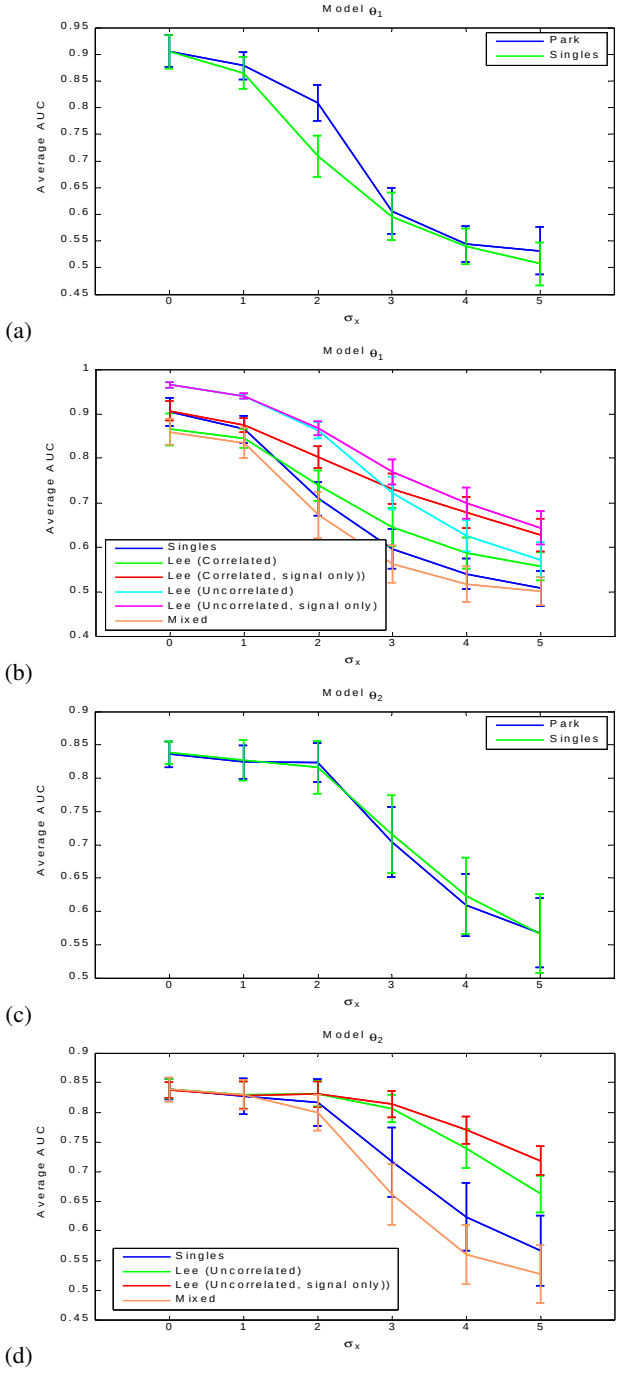


Fig. 12. Average AUC performance when comparing (a) *Park* against singles on model θ_1, (b) *Lee* against singles on model θ_1, (c) *Park* against singles on model θ_2, (d) *Lee* against singles on model θ_2.

the selection of significant markers and using DLCV instead of 5-fold cross-validation we have used a larger marker set which may have solved the small sample size problem only partly. As can be seen from the artificial data experiments, the presence of non-signal features in a dataset may cause the feature selection to perform worse, since it makes it harder to find the correct reporter set.

But the differences could also be due to different preprocessing of the datasets, slight variations in implementation of the algorithms, the cross-validation procedure, the use of Nearest Mean Classifier and AUC for evaluation, etc. However, at one point, to reproduce Chuang's algorithm, the exact same datasets and marker sets as used in their work were downloaded for evaluation, which still did not return the same trend as claimed in their paper, suggesting that the evaluation procedure has a larger effect than expected.

Three gene set searching algorithms were implemented, along with a few variations, and were subjected to the same evaluation protocol. Due to low quality, compared to the other datasets, the Wang dataset was eventually left out of the analysis. Our main observation is that there are few methods that significantly outperform the single genes case, that is, if we apply the evaluation protocol on all genes without the intervention of a gene set searching algorithm, we get a decent performance which is only outperformed by *Chuang**.

However, even though few methods outperform the single genes case, the use of the MSigDB C2 pathways seems to be responsible for the biggest improvement here. The most notable feature of the C2-based methods is that only $\sim$ 500 markers are found, consisting of $\sim$ 500 genes, in contrast to the PPI-based method that returns $\sim$ 10000 markers. To see if this performance could be attributed to the difference in number of features, the top-n ranking markers were selected for a few of these methods. However, the trends among the algorithms remained the same. See Supplementary Figure 6. It should be noted that the performance of the markers depend much on the dataset used to test the performance of these markers. Even though the performance of *Park* is weak, when the *Park* markers obtained using the Vijver dataset are tested on the DMFS datasets, *Park* becomes the highest ranking method. See Supplementary Figure 5.

Artificial data was generated to explore the basic properties of combining gene expression and using gene set searching algorithms. By doing so, we've observed that the presence of non-signal genes may worsen the performance, so perhaps employing all $\sim$ 10000 markers returned by an algorithm should require an additional proper feature selection step, such as employing the LASSO algorithm.

Also, in some cases, combining uncorrelated features may give a better performance than combining correlated features. In some cases, combining correlated features to give a better approximation of the pathway activities only seems to give an advantage when in the context of noise features, since combining correlated features helps to select the proper reporter set rather than really achieving an intrinsic improvement in signal.

To conclude, we recommend always employing the *Chuang** algorithm in conjunction with the C2 database. Single genes should always be included as a simple benchmark and possible fallback.

REFERENCES

Chin, K., DeVries, S., Fridlyand, J., Spellman, P. T., Roydasgupta, R., Kuo, W. L., Lapuk, A., Neve, R. M., Qian, Z., Ryder, T., Chen, F., Feiler, H., Tokuyasu, T., Kingsley, C., Dairkee, S., Meng, Z., Chew, K., Pinkel, D., Jain, A., Ljung, B. M., Esserman, L., Albertson, D. G., Waldman, F. M., and Gray, J. W. (2006). Genomic and transcriptional aberrations linked to breast cancer pathophysiologies. *Cancer Cell*, **10**, 529–541.

Chuang, H., Lee, E., Liu, Y., Lee, D., and Ideker, T. (2007). Network-based classification of breast cancer metastasis. *Mol. Syst. Biol.*, **3**, 140.

Desmedt, C., Piette, F., Loi, S., Wang, Y., Lallemand, F., Haibe-Kains, B., Viale, G., Delorenzi, M., Zhang, Y., d'Assignies, M. S., Bergh, J., Lidereau, R., Ellis, P., Harris, A. L., Klijn, J. G., Foekens, J. A., Cardoso, F., Piccart, M. J., Buyse, M., and Sotiriou, C. (2007). Strong time dependence of the 76-gene prognostic signature for node-negative breast cancer patients in the TRANSBIG multicenter independent validation series. *Clin. Cancer Res.*, **13**, 3207–3214.

Lee, E., Chuang, H. Y., Kim, J. W., Ideker, T., and Lee, D. (2008). Inferring pathway activity toward precise disease classification. *PLoS Comput. Biol.*, **4**, e1000217.

Loi, S., Haibe-Kains, B., Desmedt, C., Lallemand, F., Tutt, A. M., Gillet, C., Ellis, P., Harris, A., Bergh, J., Foekens, J. A., Klijn, J. G., Larsimont, D., Buyse, M., Bontempi, G., Delorenzi, M., Piccart, M. J., and Sotiriou, C. (2007). Definition of clinically distinct molecular subtypes in estrogen receptor-positive breast carcinomas through genomic grade. *J. Clin. Oncol.*, **25**, 1239–1246.

Miller, L. D., Smeds, J., George, J., Vega, V. B., Vergara, L., Ploner, A., Pawitan, Y., Hall, P., Klaar, S., Liu, E. T., and Bergh, J. (2005). An expression signature for p53 status in human breast cancer predicts mutation status, transcriptional effects, and patient survival. *Proc. Natl. Acad. Sci. U.S.A.*, **102**, 13550–13555.

Park, M. Y., Hastie, T., and Tibshirani, R. (2007). Averaged gene expressions for regression. *Biostatistics*, **8**, 212–227.

Pawitan, Y., Bjhle, J., Amler, L., Borg, A. L., Egyhazi, S., Hall, P., Han, X., Holmberg, L., Huang, F., Klaar, S., Liu, E. T., Miller, L., Nordgren, H., Ploner, A., Sandelin, K., Shaw, P. M., Smeds, J., Skoog, L., Wedrn, S., and Bergh, J. (2005). Gene expression profiling spares early breast cancer patients from adjuvant therapy: derived and validated in two population-based cohorts. *Breast Cancer Res.*, **7**, R953–964.

van de Vijver, M., He, Y., van't Veer, L., Dai, H., Hart, A., Voskuil, D., Schreiber, G., Peterse, J., Roberts, C., Marton, M., Parrish, M., and (shortened) (2002). A gene-expression signature as a predictor of survival in breast cancer. *N. Engl. J. Med.*, **347**, 1999–2009.

van 't Veer, L. J., Dai, H., van de Vijver, M. J., He, Y. D., Hart, A. A., Mao, M., Peterse, H. L., van der Kooy, K., Marton, M. J., Witteveen, A. T., Schreiber, G. J., Kerkhoven, R. M., Roberts, C., Linsley, P. S., Bernards, R., and Friend, S. H. (2002). Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, **415**, 530–536.

van Vliet, M. H., Reyal, F., Horlings, H. M., van de Vijver, M. J., Reinders, M. J., and Wessels, L. F. (2008). Pooling breast cancer datasets has a synergetic effect on classification performance and improves signature stability. *BMC Genomics*, **9**, 375.

Wang, Y., Klijn, J., Zhang, Y., Sieuwerts, A., Look, M., Yang, F., Talantov, D., Timmermans, M., Meijer-van Gelder, M., Yu, J., Jatkoe, T., Berns, E., Atkins, D., and Foekens, J. (2005). Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *Lancet*, **365**, 671–679.

Wessels, L. F., Reinders, M. J., Hart, A. A., Veenman, C. J., Dai, H., He, Y. D., and van't Veer, L. J. (2005). A protocol for building and evaluating predictors of disease state based on microarray data. *Bioinformatics*, **21**, 3755–3762.

Master's Thesis

Supplementary Material

A comparison of supervised gene set searching algorithms for outcome prediction of breast cancer

Thesis Committee:
Prof.dr.ir. M.J.T. Reinders
Dr. L.F.A. Wessels
Ir. M.H. van Vliet
Dr. E.A. Hendriks
Dr. G.W. Klau

Author	**Raul Kooter**
Email	Raul@xs4all.nl
Student number	1150162
Thesis supervisor	Dr. L.F.A. Wessels Ir. M.H. van Vliet
Date	September 23, 2009 - 14:00

Information and Communication Theory Group [I,C)^T

TU Delft

Faculty of Electrical Engineering, Mathematics and Computer Science

Delft University of Technology

A comparison of supervised gene set searching algorithms for outcome prediction of breast cancer: Supplementary Material

Raul Kooter [1]

[1]Information and Communication Theory Group, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands

1 CROSS-VALIDATION PROCEDURE

The dataset is split in five stratified folds. Each subset is used in turn as a testing subset, with the other four subsets being the training subset. The training subset is used to both determine the optimal number of features and to train the classifier, which is tested on the the testing subset. This procedure returns five AUC values, and is repeated 20 times, returning 100 AUC values in total. The average of these AUC values is returned as the average performance.

The inner loop of the cross-validation procedure is run to determine the optimal number of features in an unbiased manner. The dataset fed into this inner loop is itself split in four subsets, with each subset being an inner testing subset and the other three being the inner training subset. For every fold, the features are ranked by the magnitude of the t-score of the inner training subset. Given this ranking, a learning curve l was calculated for every fold. This inner loop procedure was repeated three times, after which a combined learning curve was calculated. The optimal number of features $n*$ was determined by taking the position of the minimum error in the combined learning curve. Given this optimal number of features, the entire dataset fed into the inner loop was re-ranked according to t-score, and the NMC was trained on the top ranking features.

To make a fair comparison between sets of markers tested on the same dataset, the information on how the samples were split was stored and reused. The AUC values are then suitable for testing with a paired Wilcoxon test.

An overview of the procedure can be seen in Figure 1.

2 ADDITIONAL TABLES AND FIGURES

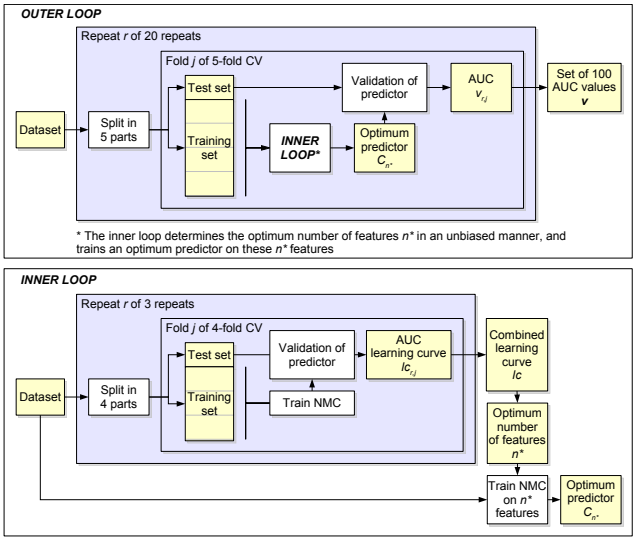


Fig. 1. Double-loop-cross-validation strategy.

Algorithm 1 Chuang's algorithm

for $Node \in Nodes_{PPI}$ **do**
  $Subnetwork \leftarrow Node$
  $Score \leftarrow$MI-Score$(X_{Subnetwork}, Y)$
  repeat
    $Candidates \leftarrow$Neighbours$(Subnetwork, PPI) \cap$MaxDistance2$(Node, PPI)$
    for $Candidate \in Candidates$ **do**
      $Score_{Candidate} \leftarrow$MI-Score$(X_{Subnetwork \cup Candidate}, Y)$
    end for
    $BestCandidate \leftarrow \arg\max_{Candidate}(Score_{Candidate})$
    if $Score_{BestCandidate} > Score \times 1.05$ **then**
      $Subnetwork \leftarrow Subnetwork \cup BestCandidate$
    end if
  until $Subnetwork$ hasn't changed
  $Subnetworks \leftarrow Subnetworks \cup Subnetwork$
end for
return $Subnetworks$

Algorithm 2 Lee's algorithm

for $Geneset \in Genesets$ **do**
  $CORG \leftarrow \emptyset$
  $Score_{CORG} \leftarrow 0$
  for $Gene \in Geneset$ **do**
    $Score_{Gene} \leftarrow$T-Test(X_{Gene}, Y)
  end for
  if Mean$(Score_{Geneset}) > 0$ **then**
    $Geneset \leftarrow$OrderDescending$(Geneset)$ {Order w.r.t. scores}
  else
    $Geneset \leftarrow$OrderAscending$(Geneset)$
  end if
  $N \leftarrow 0$
  repeat
    $N \leftarrow N + 1$
    $Candidates \leftarrow Geneset_{1,\ldots,N}$
    $Score_{Candidates} \leftarrow$T-Test$(X_{Candidates}, Y)$
    if $Score_{Candidates} > Score_{CORG}$ **then**
      $CORG \leftarrow Candidates$
      $Score_{CORG} \leftarrow Score_{Candidates}$
    end if
  until $CORG$ hasn't changed
  $CORGs \leftarrow CORGs \cup CORG$
end for
return $CORGs$

Algorithm 3 Park's algorithm

$Dendrogram \leftarrow$Cluster(X) {Use average linkage}
for $Level := 1$ to $\#Features(X)$ **do**
  $Genesets \leftarrow$CutOff$(Dendrogram, Level)$
  $LearningCurve_{Level} \leftarrow$CrossValidate$(X_{Genesets}, Y)$ {Use area under curve}
  $MinError_{Level} \leftarrow$Min$(LearningCurve_{Level})$
end for
$BestLevel \leftarrow \arg\min_{Level}(MinError_{Level})$
$BestGeneSets \leftarrow$CutOff$(Dendrogram, BestLevel)$
return $BestGeneSets$

1

Fig. 2. Pseudo-code of the three main algorithms used.

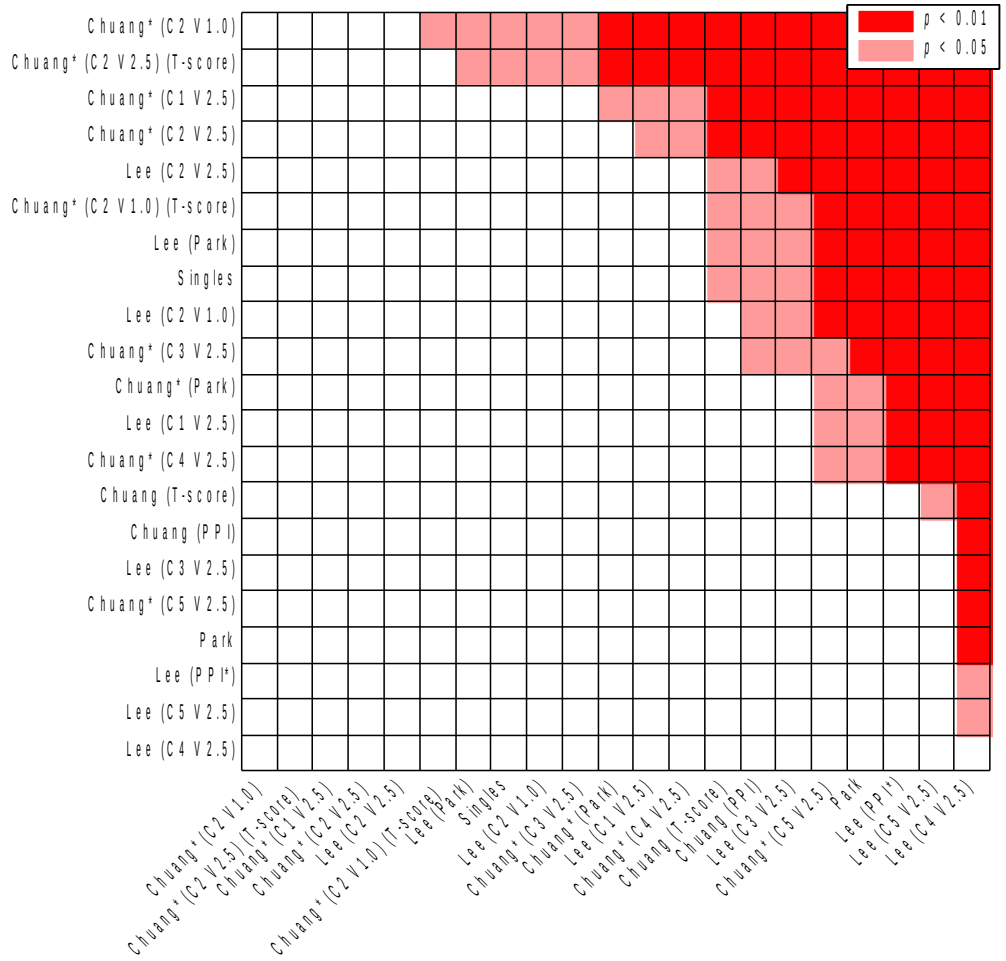


Fig. 3. A comparison of the algorithms. A red box in row X and column Y indicates that the algorithm in row X significantly outperforms the algorithm in column Y. In this set, Wang is included as both a training and testing dataset.

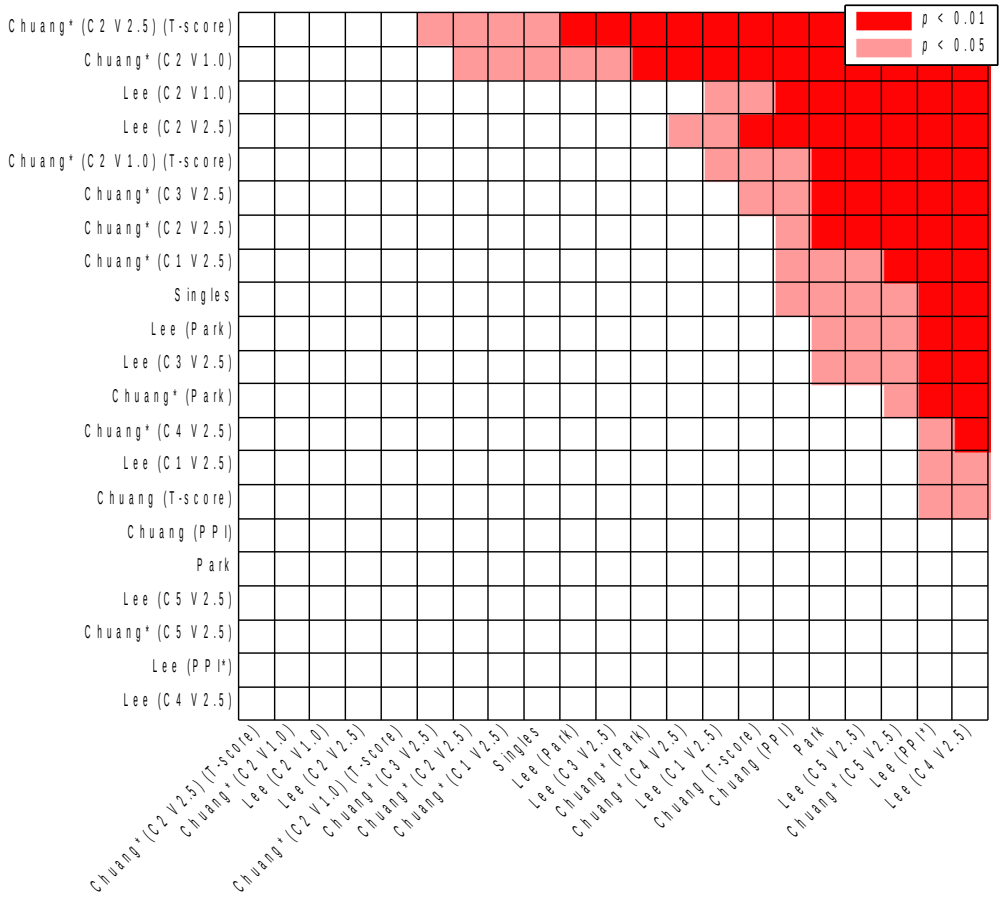


Fig. 4. Same as Figure 3, but Wang was not included for training and testing.

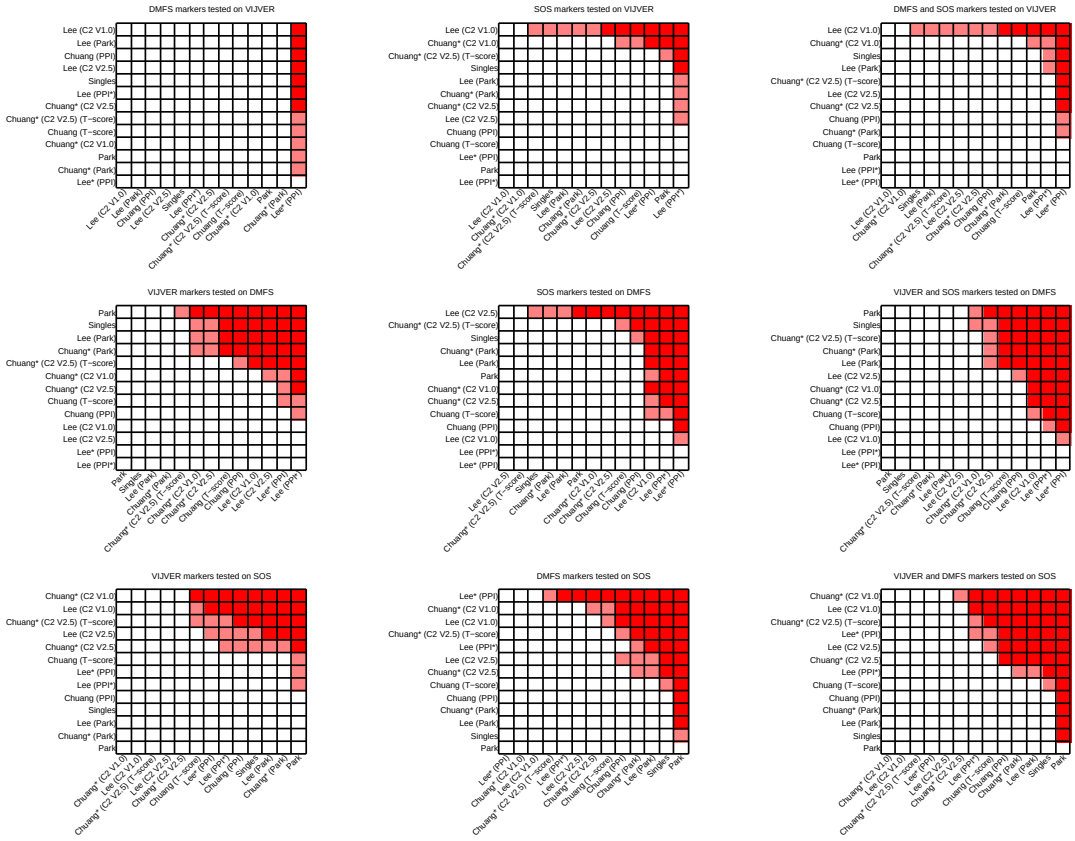


Fig. 5. A comparison of a subselection of the algorithms separated according to training and testing dataset. The same data as used as shown in Figure 4, the results aren't merged together.

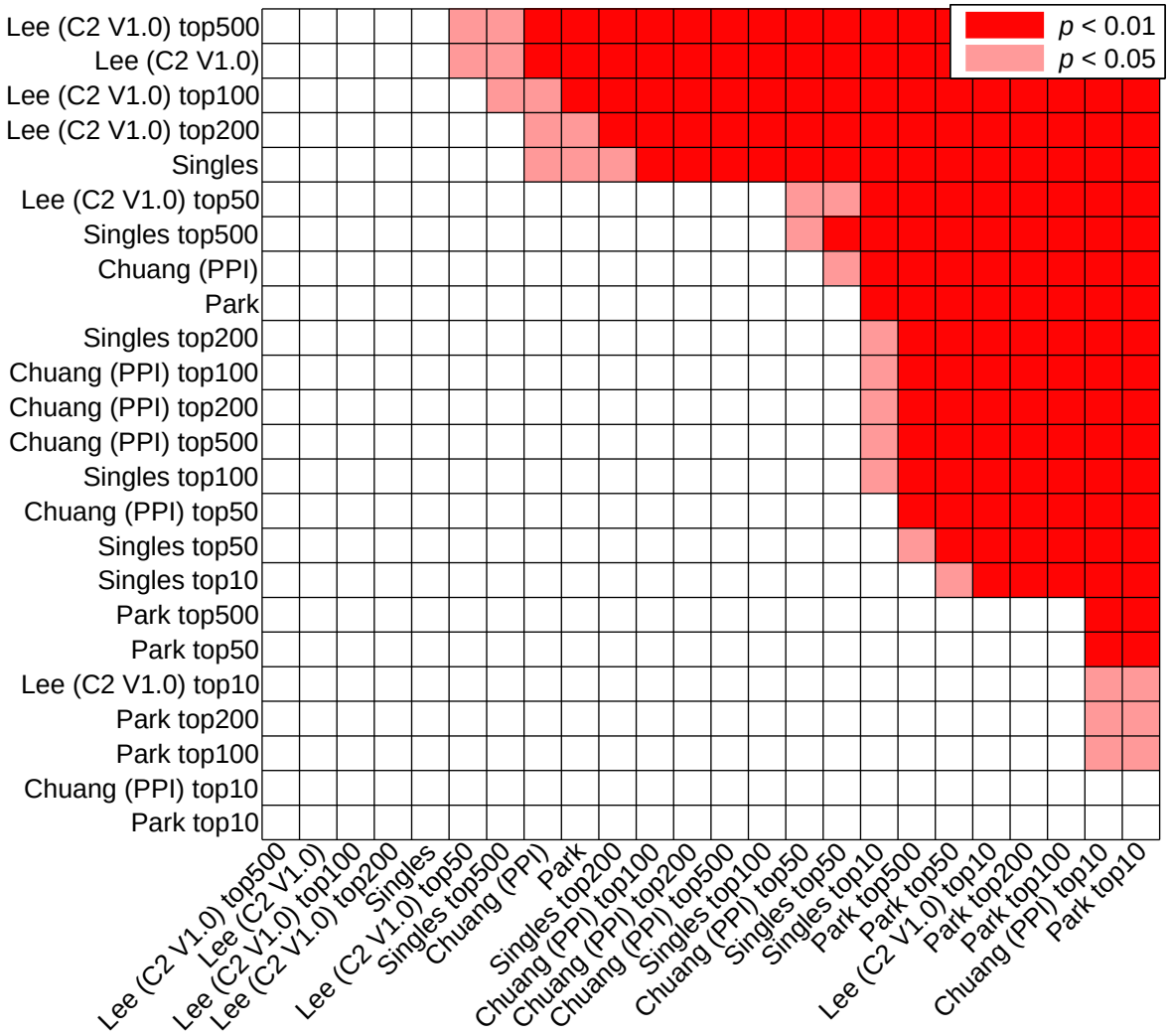


Fig. 6. Similar to Figure 4, but using the top-n ranking markers. Markers were ranked according to absolute t-score using the training dataset, and we highest ranking n number of markers were selected for evaluation.

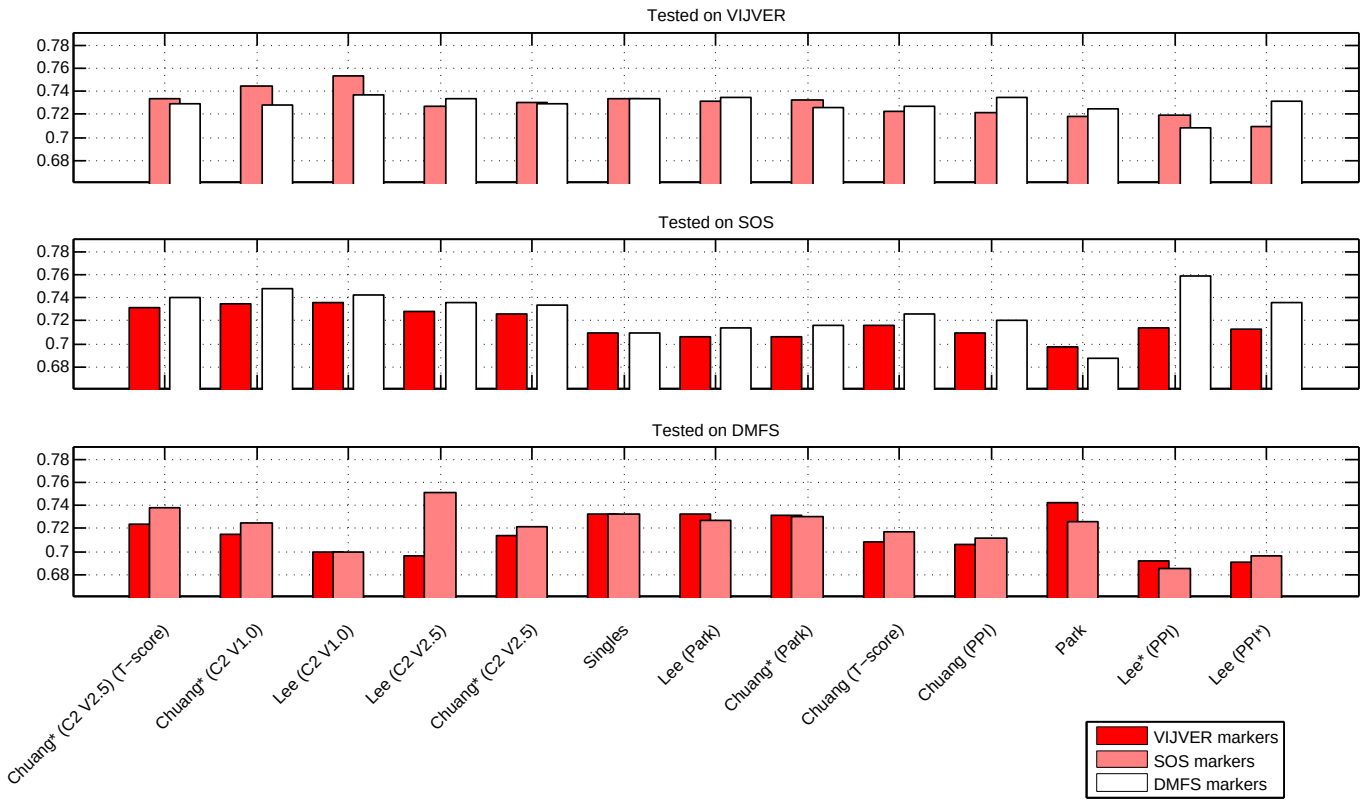


Fig. 7. Barplots of the performances. The same data used to generate Figure 4 is used here.

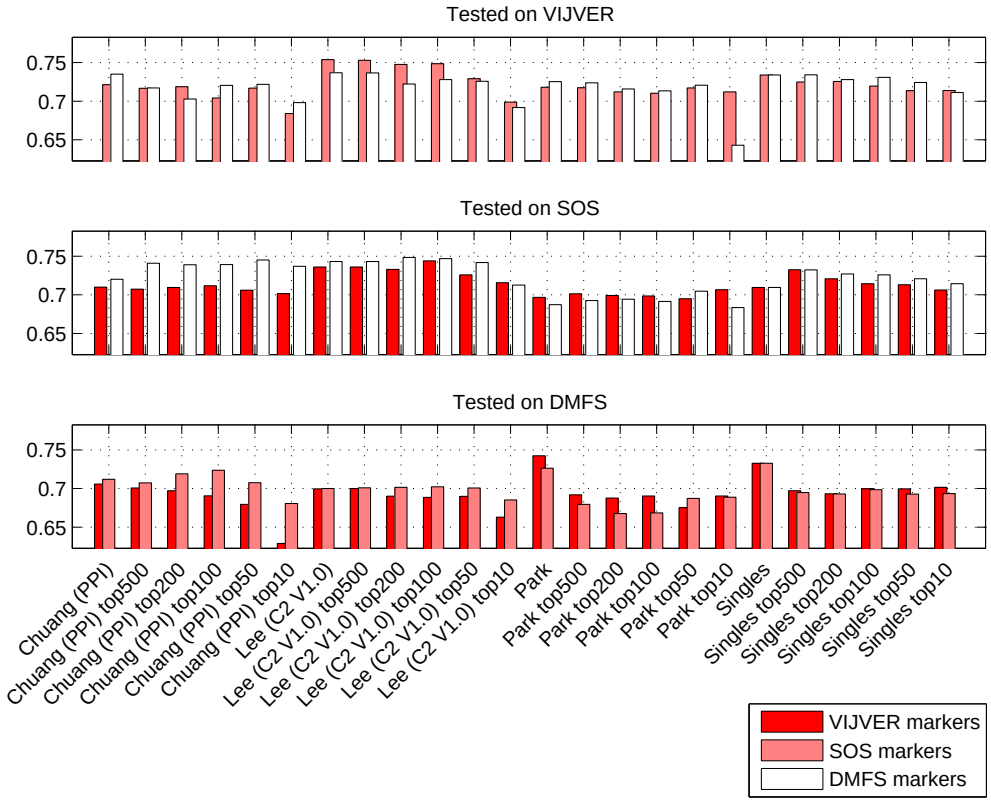


Fig. 8. Barplots of the performances of the top-n ranking markers. The same data used to generate Figure 6 is used here.

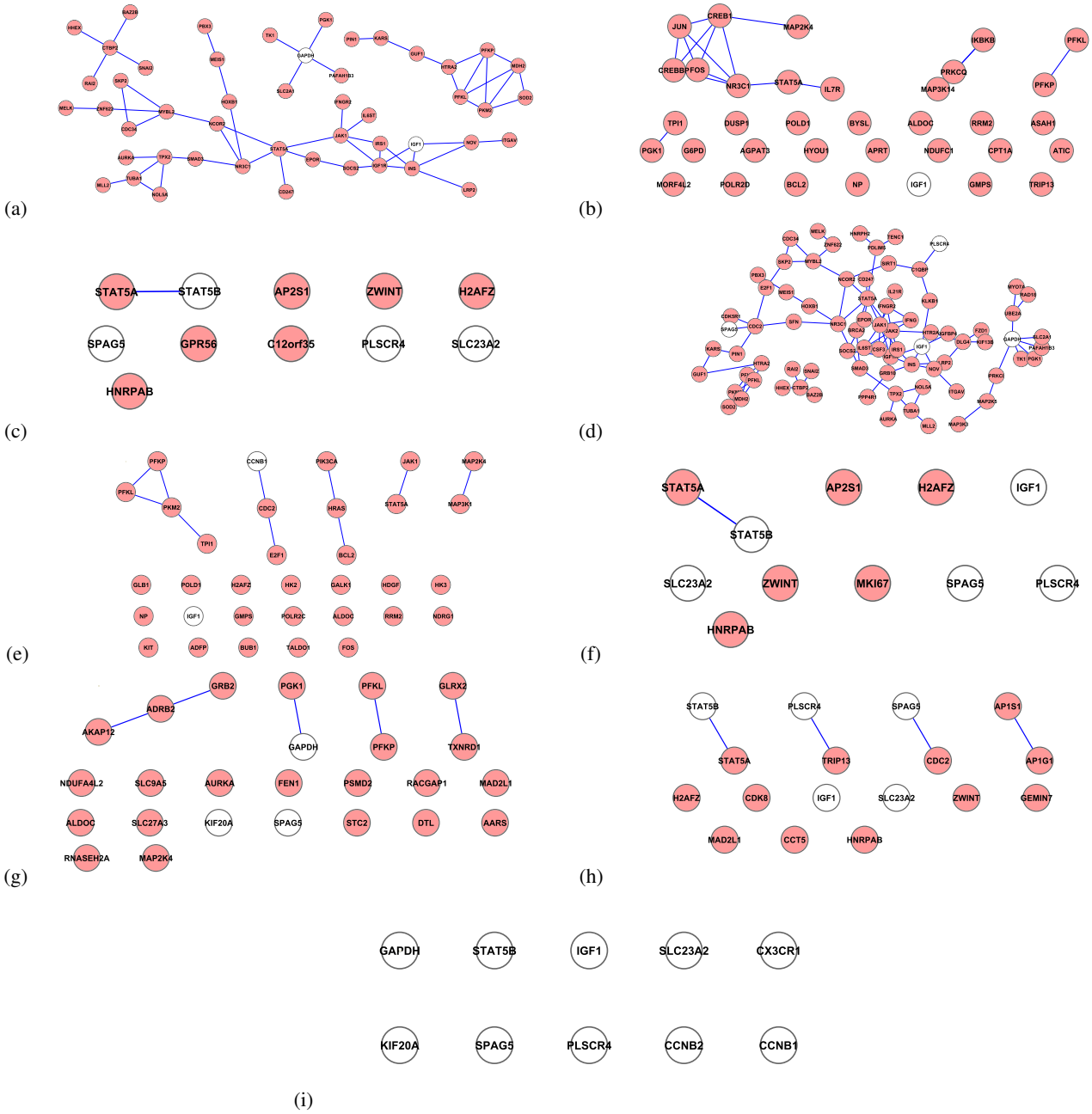


Fig. 9. The genes involved in the top 10 most frequent selected markers for each method, using markers that are found using Vijver, and which are tested on SOS. The methods are: a) Chuang, b) Chuang* (C2 V1.0), c) Chuang* (Park), d) Chuang(PPI, t-score), e) Lee (C2 V1.0), f) Lee (Park), g) Lee (PPI*) h) Park and i) Singles. The markers indicated in white are also the top 10 single gene markers for this setup.

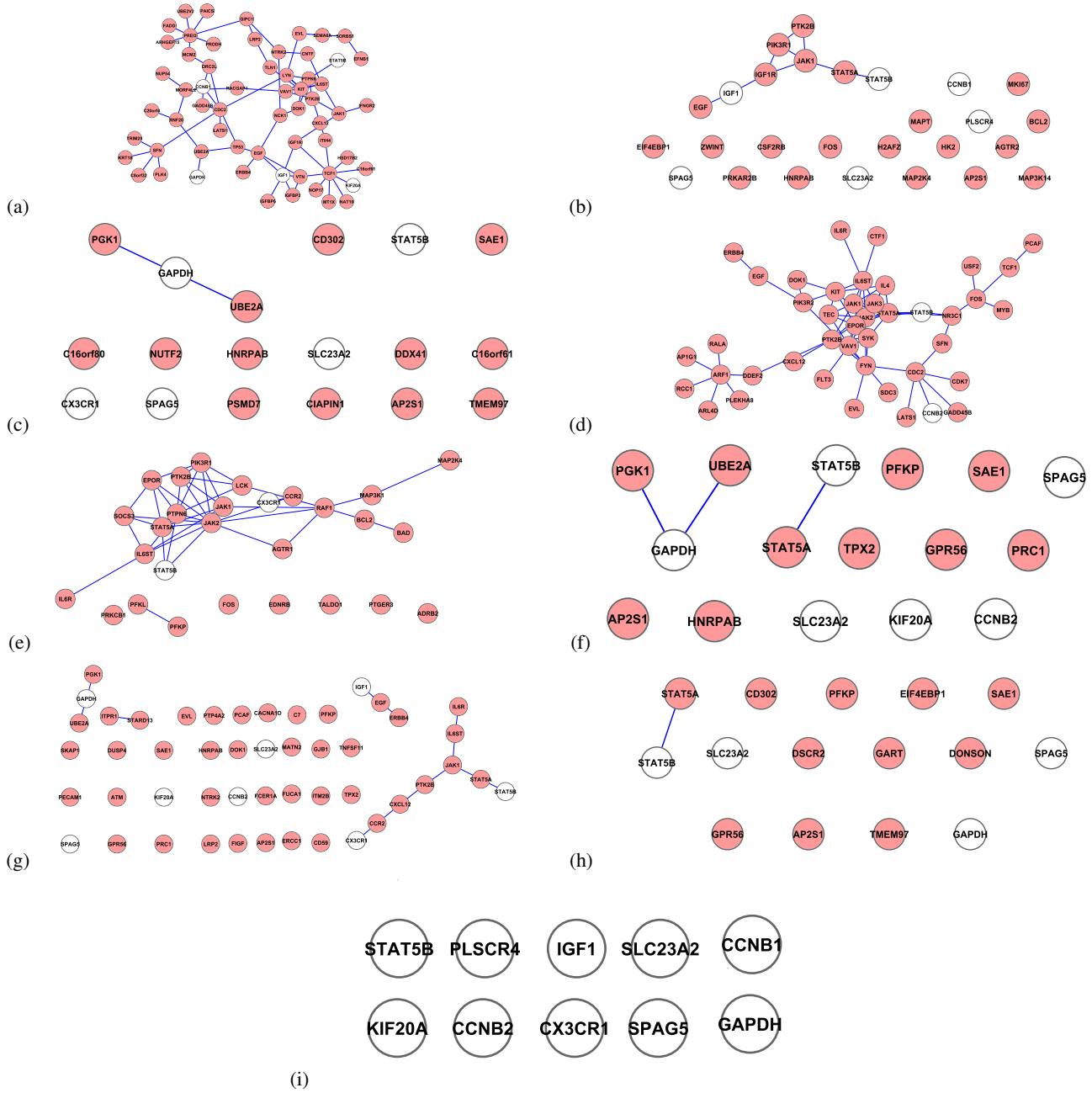


Fig. 10. The genes involved in the top 10 most frequent selected markers for each method, using markers that are found using DMFS, and which are tested on SOS. The methods are: a) Chuang, b) Chuang* (C2 V1.0), c) Chuang* (Park), d) Chuang(PPI, t-score), e) Lee (C2 V1.0), f) Lee (Park), g) Lee (PPI*) h) Park and i) Singles. The markers indicated in white are also the top 10 single gene markers for this setup.

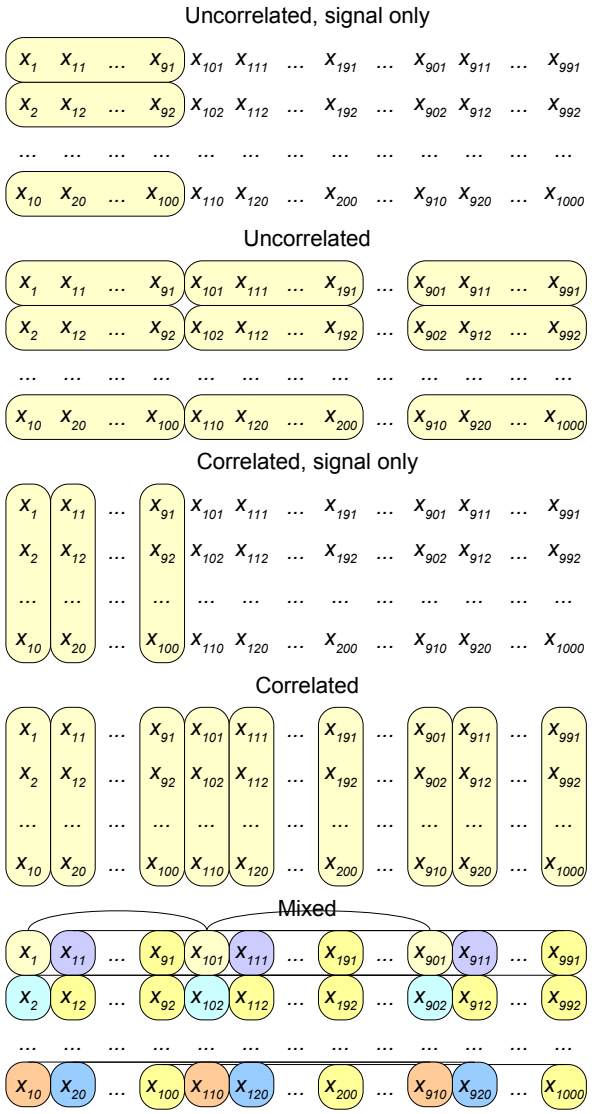


Fig. 11. A graphical representation of the gene sets used for the artificial data experiments.

Tested	Markers	Chuang (PPI)	Chuang (T-score)	Chuang* (C2 V1.0)	Chuang* (Park)	Lee (C2 V1.0)	Lee (PPI*)	Lee (Park)	Park	Singles
DMFS	SOS	ITIH4 CXCL12 PTK2B JAK2	RPS6KA6 MAPK1 HSP90AA1 NEK2 PTPRR HIF1A GRB2	CXCL12 PTK2B	TNFRSF14	ALDH1A1 ABAT ACADS	CDC25A CENPN E2F1 PPFIA1 KRT18 RALA	TNFRSF14	CCT5	TNFRSF14
DMFS	SOS	WDR5 HSP90AA1 EIF2AK2 LSM1 YWHAZ TERT ERBB2	HIF1AN HDAC2 SUV39H1 EED ASH2L HIF1A PGK1 PML	CXCL12 PTK2B	CCT5	CDC2 CCNB1	DLG7 CDC25A CENPN E2F1 PPFIA1 KRT18 HSP90AA1	CCT5	TNFRSF14	EPHX2
DMFS	SOS	RASGRF1 CDC2 CCNB1 CCNB2 CDKN1A CIB2	HAL MAPK1 HSP90AA1 NEK2 PTPRR HIF1A DHPS	PPFIA1 CDH3 CPNE1 BYSL ALCAM	EPHX2	CDC2 CCNB1	STAT5A CXCL12 IL6ST KIT EVL	EPHX2	EPHX2	CCT5
DMFS	SOS	WEE1 CCNB2 CCNB1 GADD45B CDC25B	IER3 MAPK1 NEK2 PTPRR HSP90AA1 HIF1A GRB2 CAMK2D MAPK6	PPP1R12B ADCY1 PTK2B	BTG2	STAT5B STAT5A MAP2K4 FOS BCL2	PDE4A KIT EVL SKAP1 IGJ	TREM1	PARP3	BTG2
DMFS	SOS	VAV1 PTK2B IL6ST CXCL12 PIK3CG	FKBP3 HDAC2 SUV39H1 EED TOP2A PPARD ASH2L GADD45B	STAT5B JAK1	TREM1	TRIP13 CCT5 PGK1 NOL5A PSMD14	IL6ST KIT EVL PDGFRA CSF1 GOLGB1 FLT3 EPOR ARHGEF7 IL6R	PARP3	TREM1	KIF13B
DMFS	SOS	IL2RA STAT5B STAT5A JAK1 PTK2B	CAMK2D MAPK1 HSP90AA1 NEK2 PTPRR HIF1A DHPS	STAT5B JAK1	PARP3	CCNB1 CDC20 TRIP13 HNRPAB ALG3	LRP2 SYNE1 KIF13B ERBB4	BTG2	KIF13B	PARP3
DMFS	SOS	GYS2 CCT2 CCT6A CCT5 GYS1	THAP4 PREI3 FADD GIPC1 MCM2 PAICS	STAT5B JAK1	KIF13B	CCNB1 CDC20 TRIP13	DKFZp762E1312 CCT5 AP1G1	BTD	BTG2	TREM1
DMFS	SOS	CBX5 CCT5 MKI67 TCP1	TNFRSF14 TRAF3 NRIP1 PPARG CTBP2 NR3C1	TRIP13 PGK1	SQLE	CDC2 CCNB1 CDC20 BUB1B	DKFZp762E1312 CCT5 AP1G1	KIF13B	SQLE	BTD
DMFS	SOS	CYB5R2 TRIP13 KIAA1609	HDAC1 TOP2A PPARD RUVBL2 EED GADD45B CCNB1 MDM2 S100A9 ASH2L	IGF1 BCL2 ADCY1	BTD	CDC2 CCNB1 CDC20 BUB1B	PSMD14 STMN1 KRT18 VARS HSP90AA1 TMEM132A PFKL HSPA14 NDRG1	SQLE	BTD	SQLE
DMFS	SOS	INPP5D DOK1 KIT IL6ST PDGFRL	C8orf32 TRIP13 SEC24A SFN KRT18 CDC2 KIAA0408	ATM BCL2	ELOVL5	CDC2 CCNB1	DKFZp762E1312 CCT5 ITGB4BP PPFIA1	ABHD14A	ELOVL5	KIF20A

Table 1. Overview of the top 10 subnetworks. Markers from SOS, tested on DMFS. These top ranking features were found by running the DLCV on DMFS using the SOS markers, and keeping track how often a feature was returned as an optimal predictor by the inner loop of the DMFS.

Tested	Markers	Chuang (PPI)	Chuang (T-score)	Chuang* (C2 V1.0)	Chuang* (Park)	Lee (C2 V1.0)	Lee (PPI*)	Lee (Park)	Park	Singles
DMFS	VIJVER	MAD2L2 MAD2L1 BUB1B BAT2 P4HA2 CENPA	IL6R IL6ST JAK2 STAT5A EPOR RPL4 AR JAK1	BYSL PGK1 TRIP13 NP MORF4L2 HYOU1	TNFRSF14	F11R BAIAP2 BYSL GP5	RPL11 KIF13B	TNFRSF14	EPHX2	TNFRSF14
DMFS	VIJVER	WEE1 CDCA3 CCNB2 PKMYT1 YWHAB MAP2K1 CCNE2 CCNA2 ITGB1	KRT18 TROAP HGS SFN CDC2 BIRC5 MAP2K1 CDK5R1	BCL2 FCER1A ICOS	EPHX2	CCNB2	RPL11 KIF13B	EPHX2	TNFRSF14	EPHX2
DMFS	VIJVER	FLJ20254 RAD54L RAD51 PSMD7 LSM1 UPF2	HTR2A JAK2 IL6ST STAT5A CSF3 JAK1 DLG4 FZD1 KIF13B BRCA2	JAK2 CISH JAK1 IL6R	TRIP13 CCT5	E2F1 NDRG1 CDC2	RPL11 KIF13B	BTG2	TRIP13 CCT5	CCT5
DMFS	VIJVER	BYSL TROAP TRIM37 KIAA0408 PRC1	PAK2 ARHGEF6 PDHB TGFBR1 PIK3R1 VAV3 IRS1 INS ARHGAP15	BCL2 IL7R STAT5A	KIF13B	BCL2 IGF1 KIT	K-ALPHA-1 F11R GAPDH RGS19 SLC2A1 GRB2 NFASC KCNA2 ITGA5 CLTC DDEF1	TREM1	BTG2	BTG2
DMFS	VIJVER	SKI K-ALPHA-1 PML CCT2 TDG	MAGEA12 STAT5A JAK2 C10orf86 AR RPL4 EPOR JAK1	PSMD1 ABCF1 CDC20	BTG2	JAK2 STAT5A JAK1	K-ALPHA-1 CCT5 NFASC TAF6	BTD	PARP3	KIF13B
DMFS	VIJVER	PIN4 TPX2 AURKA FN1 COL13A1 MMP9 LGALS3BP COL4A1 GTPBP4 NAT10	ERCC1 CCNH GTF2B RRAD POLR1B RPL5 ESR1 RPL11 CCND2 MRPL2	TPX2 CDK2AP1 VIL2	TREM1	BCL2 JAK2 FOS	KIF20A PSMD7	PARP3	TREM1	PARP3
DMFS	VIJVER	K-ALPHA-1 CCT5 THEG CCT2	LOC158997 KPNA1 NP GAPDH	BTG2 BCL6 IGFBP6 FHL2	PARP3	BCL2 STAT5A JAK1 PIK3CA IL2RG PIK3R1	BTG2	KIF13B	SQLE	TREM1
DMFS	VIJVER	THEG CCT5 K-ALPHA-1 CCT2	ROS1 VAV3 IGF1R JAK1 JAK2 ZYX IL6ST KIT HOXA9	CCNB2 ERBB2 RAD51 CCNE1	BTD	RRM2 PGK1 MARS	BTG2	ABHD14A	KIF13B	BTD
DMFS	VIJVER	RAE1 BUB1 BUB1B	TIAF1 JAK3 IL6ST JAK2 JAK1 STAT5A	E2F1 IL11 BUB1B	SQLE	ALDH3A2 ABAT GAD1 DPYD ALDH2	BTG2	SQLE	COCH	SQLE
DMFS	VIJVER	EPHX2	ARHGAP8 CTTN ANKZF1 GRB2 FGD1 KCNA2 ACTR3	JAK2 CISH JAK1 IFNG	ABHD14A	TPX2	BTG2	CDKN3	ZNF395	KIF20A

Table 2. Top 10 subnetworks. Markers from VIJVER, tested on DMFS.

Tested	Markers	Chuang (PPI)	Chuang (T-score)	Chuang* (C2 V1.0)	Chuang* (Park)	Lee (C2 V1.0)	Lee (PPI*)	Lee (Park)	Park	Singles
SOS	DMFS	MT1X TCF1 C16orf61 KIF20A NOP17 NAT10 HSD17B2	EPOR STAT5A JAK1 PTK2B CXCL12 IL6ST FYN EVL IL6R	BCL2 STAT5A PIK3R1 FOS MAP2K4	STAT5B	STAT5A STAT5B BCL2 FOS JAK2 MAP3K1 PIK3R1 MAP2K4 RAF1	CX3CR1 EVL FUCA1 DOK1 CXCL12 STAT5A SKAP1	STAT5A STAT5B	STAT5A STAT5B	CCNB1
SOS	DMFS	CNTF IL6ST NTRK2 VAV1 JAK1 IFNGR2 PTPN6 KIT STAT5B	STAT5A JAK1 PTK2B IL6ST KIT DOK1 FYN EVL	STAT5A JAK1 MAP2K4 EGF	SLC23A2	STAT5A PTK2B JAK1 STAT5B BCL2 PIK3R1	FIGF IGF1 CACNA1D STAT5A ERBB4 PTK2B ATM	PGK1 UBE2A	SPAG5 TMEM97	STAT5B
SOS	DMFS	SFN CDC2 CCNB1 LATS1 GADD45B KRT18 ORC2L	SFN CDC2 GADD45B CCNB2 LATS1 CDK7	FOS MAP3K14 MAP2K4	PGK1 UBE2A	STAT5A STAT5B FOS JAK2 PTPN6 RAF1 EPOR	DOK1 IGF1 ITPR1 LRP2 STAT5A ITM2B SKAP1 ERBB4 PTK2B JAK1 ATM	SPAG5	SLC23A2	CCNB2
SOS	DMFS	RNF20 UBE2A GAPDH MORF4L2 RACGAP1 C20orf4 NUP54	SYK STAT5A JAK1 IL6ST PTK2B KIT DOK1 FYN EVL SDC3	BCL2 IGF1 CSF2RB PRKAR2B	SPAG5 TMEM97	STAT5A JAK1 STAT5B FOS LCK	C7 DOK1 IGF1 LRP2 STAT5A JAK1 IL6R	GAPDH	GAPDH	KIF20A
SOS	DMFS	TRIM25 SFN PLK4 KRT18 CDC2 C8orf32	PIK3R2 DOK1 KIT JAK2 STAT5B PTK2B EGF ERBB4 TEC JAK1	EIF4EBP1 HK2	HNRPAB DDX41	STAT5A JAK1 STAT5B BCL2 FOS BAD PTPN6 PIK3R1 SOCS3	CX3CR1 EVL FUCA1 DOK1 CXCL12 CCR2 ERCC1 LRP2 STAT5A SKAP1	CCNB2 KIF20A TPX2 PRC1	PFKP	SPAG5
SOS	DMFS	ARHGEF15 PREI3 MCM2 FADD UBE2V2 PAICS PRODH GIPC1	IL4 STAT5A JAK1 PTK2B IL6ST KIT DOK1	FOS IGF1 IGF1R	CX3CR1	STAT5A STAT5B FOS JAK2 PTPN6	FIGF DOK1 CXCL12 STAT5A SKAP1 ERBB4 PTK2B JAK1	SLC23A2	GPR56	GAPDH
SOS	DMFS	SORBS1 SEMA6A EVL EFNB1 LYN	CTF1 IL6ST JAK1 VAV1 KIT IL6R FLT3	CCNB1	GAPDH	JAK1 IL6R FOS IL6ST	EVL GJB1 STAT5A SKAP1 PTK2B IL6R PECAM1 TNFSF11 IL6ST CD59 EGF PTP4A2	HNRPAB	AP2S1 SAE1	PLSCR4
SOS	DMFS	NCK1 DOK1 KIT EGF ERBB4 VTN IGF1 TP53	USF2 FOS NR3C1 MYB STAT5A JAK1 TCF1 PCAF	STAT5B FOS CSF2RB	SAE1 AP2S1	STAT5A STAT5B FOS JAK2 PRKCB1 PIK3R1	NTRK2 MATN2 EVL FUCA1 DOK1 CXCL12 DUSP4 GJB1 STAT5A SKAP1	GPR56	GART DSCR2 DONSON	IGF1
SOS	DMFS	IGFBP2 IGF1 EGF IGFBP6 IGF1R	PLEKHA8 ARF1 RALA DDEF2 AP1G1 ARL4D RCC1	MAPT PTK2B STAT5A AGTR2	CD302	CX3CR1 EDNRB CCR2 PTGER3 ADRB2 AGTR1	MATN2 EVL GJB1 STAT5A SKAP1 FCER1A PTK2B JAK1 IL6R	PFKP	EIF4EBP1	CX3CR1
SOS	DMFS	TLN1 LRP2 PTK2B CXCL12 JAK1 ITIH4 IL6ST	JAK3 STAT5A JAK1 IL6ST VAV1 KIT DOK1	CCNB1	C16orf61 CIAPIN1 PSMD7 C16orf80 NUTF2	PFKL PFKP TALDO1	NTRK2 EVL FUCA1 PCAF DOK1 STARD13 GJB1 STAT5A	SAE1 AP2S1	CD302	SLC23A2

Table 3. Top 10 subnetworks. Markers from DMFS, tested on SOS.

Tested	Markers	Chuang (PPI)	Chuang (T-score)	Chuang* (C2 V1.0)	Chuang* (Park)	Lee (C2 V1.0)	Lee (PPI*)	Lee (Park)	Park	Singles
SOS	VIJVER	ZNF622 MYBL2 SKP2 MELK CDC34 NCOR2	PRKCI GAPDH TK1 PGK1 MAP2K5 MAP3K3 UBE2A	PFKL PFKP ALDOC G6PD	STAT5B	PFKL GALK1 PFKP HK3 HK2 GLB1	AURKA PFKL PSMD2 SPAG5 AARS PGK1	STAT5B	STAT5B	CCNB1
SOS	VIJVER	IFNGR2 JAK1 IGF1R STAT5A IL6ST	PAFAH1B3 GAPDH TK1 PGK1	MAP3K14 NR3C1 DUSP1 IKBKB CREBBP	AP2S1	BCL2 IGF1 KIT	PFKL MAD2L1 PFKP SLC27A3	PLSCR4	PLSCR4	STAT5B
SOS	VIJVER	PAFAH1B3 GAPDH TK1 PGK1 SLC2A1	KLKB1 IGF1 INS C1QBP SIRT1 IGFBP4 PLSCR4	BYSL PGK1 TRIP13 NP MORF4L2 HYOU1	HNRPAB	PFKL TALDO1 PFKP ALDOC	PFKP GAPDH SLC27A3	SLC23A2	SPAG5	CCNB2
SOS	VIJVER	HOXB1 MEIS1 PBX3 NR3C1 STAT5A	RAD18 UBE2A GAPDH	TPI1 CPT1A	SLC23A2	FOS MAP3K1 MAP2K4 STAT5A JAK1 PIK3CA	GAPDH SLC27A3 RNASEH2A ALDOC	SPAG5	AP1G1 CDK8 AP1S1 GEMIN7	KIF20A
SOS	VIJVER	NOL5A TPX2 AURKA SMAD3 TUBA1 MLL2	HTR2A JAK2 IL6ST STAT5A CSF3 JAK1 DLG4 FZD1 KIF13B BRCA2	IGF1	STAT5A	FOS MAP3K1 MAP2K4 STAT5A JAK1 PIK3CA	PFKL FEN1 PFKP	IGF1	STAT5A	SPAG5
SOS	VIJVER	IGF1 NOV INS ITGAV IRS1 LRP2	MYO7A UBE2A GAPDH	POLD1 GMPS NP APRT POLR2D ATIC	SPAG5	CCNB1 CDC2 HRAS	RACGAP1 PFKL PGK1	STAT5A	CDC2 H2AFZ MAD2L1 ZWINT	GAPDH
SOS	VIJVER	SOD2 MDH2 PFKL PFKP	SFN CDC2 E2F1 SPAG5 CDK5R1	MAP2K4 FOS ASAH1 CREB1	PLSCR4	RRM2 POLD1 GMPS NP POLR2C PKM2	DTL KIF20A PGK1	AP2S1	HNRPAB	PLSCR4
SOS	VIJVER	GUF1 HTRA2 PFKL PFKP KARS PKM2 PIN1	GRB10 JAK2 IL6ST INS IRS1 PPP4R1 IFNG STAT5A	RRM2 PGK1 NDUFC1 AGPAT3	ZWINT H2AFZ	E2F1 NDRG1 CDC2	DTL KIF20A PGK1	HNRPAB	SLC23A2	IGF1
SOS	VIJVER	HHEX CTBP2 SNAI2 BAZ2B RAI2	TENC1 PDLIM5 HNRPH2 STAT5A	BCL2 MAP2K4 JUN PRKCQ	GPR56	PFKL TPI1 PFKP HK3 ALDOC	TXNRD1 PFKP GAPDH GLRX2 NDUFA4L2 GRB2	ZWINT H2AFZ	TRIP13 CCT5	CX3CR1
SOS	VIJVER	EPOR STAT5A SOCS2 JAK1 IL6ST CD247	IL21R JAK1 IL6ST JAK2 STAT5A CSF3	BCL2 IL7R STAT5A	C12orf35	H2AFZ BUB1 TALDO1 HDGF ADFP	STC2 MAP2K4 AKAP12 SLC9A5 ADRB2	MKI67	IGF1	SLC23A2

Table 4. Top 10 subnetworks. Markers from VIJVER, tested on SOS.

Tested	Markers	Chuang (PPI)	Chuang (T-score)	Chuang* (C2 V1.0)	Chuang* (Park)	Lee (C2 V1.0)	Lee (PPI*)	Lee (Park)	Park	Singles
VIJVER	DMFS	SLC25A11 COPA ARFGAP1 CDKN1A CCNB1 CCNE2 CCNB2 GADD45B MYC	DIAPH1 CENPA BAIAP2 MAD2L1	PGK1 PFKL	E2F1	H2AFZ MKI67 PSMA7 PSMD2 PSMD1	TMPO HSP90AA1 SPP1 F11R BAT3 ZG16	PKMYT1 E2F1	BIRC5 TK1 HN1	E2F1
VIJVER	DMFS	MYT1L PKMYT1 CCNB2 CCNB1 CCNA1 CCNE2	PRC1	E2F1 CFL1	ADAM8	PFKL PFKP TALDO1	TMPO HSP90AA1 SPP1 F11R EN2 BAT3 ZG16	GLTSCR2 TPT1	E2F1 PKMYT1	TK1
VIJVER	DMFS	PRKCBP1 BIRC5 PSMD2	FBXO32 CUL1 E2F1 CDCA3 CCNA2 CCND1	E2F1	EBP	E2F1 CFL1 ARF3	TMPO HSP90AA1 SPP1 F11R BAT3 ZG16	AURKA	AURKA	PRC1
VIJVER	DMFS	LIMK1 PAK4 YWHAZ RACGAP1 LATS1	GDF8 SGTA TMPO SPP1 HSP90AA1 BAT3 PTN F11R EFEMP2	E2F1	BIRC5	TPI1 HK2 PMM2 PFKL PFKP SORD PFKFB1	TMPO HSP90AA1 SPP1 F11R BAT3 ZG16	EBP	EBP	ESPL1
VIJVER	DMFS	PAK4 RACGAP1 AURKB	RUTBC1 RPS25 EIF3S4 CA12 RPL11	BIRC5	AURKA	CFL1 ACTR3 BAIAP2	TMPO HSP90AA1 SPP1 F11R BAT3	CCNB2 KIF20A TPX2 PRC1	ADAM8	BIRC5
VIJVER	DMFS	WBP2 PSMD2 PSMA7 ORC1L	ANKZF1 AURKB RACGAP1 TACC1 PSMD1 PSMD7	E2F1 CCNA2	PSMD2	CDC2 MAD2L1 ATP2A2 E2F1	TMPO HSP90AA1 SPP1 F11R BAT3 ZG16	HN1 BIRC5	ADRA2B	E2F2
VIJVER	DMFS	RACGAP1 AURKB PAK4	USHBP1 PRC1	PSMA7	RCE1	CCNB2	TMPO HSP90AA1 SPP1 F11R BAT3 ZG16	ADAM8	PFKL	TPT1
VIJVER	DMFS	RGS3 YWHAZ PCTK1 BRAF CDC25B CDC25A PTPN13	UNC84A RRM2 EIF4G1 NEURL	E2F1 ABL1	GLTSCR2 TPT1 RPS27A	PGK1	TMPO HSP90AA1 SPP1 F11R BAT3 GJA8	PSMD2	SLC1A5	CCNB2
VIJVER	DMFS	BRD2 E2F1 CCNA1	RPL12L3 AARS SEC61G RAD51	BCL2 STAT5A PIK3R1 FOS MAP2K4	DDX39	PGK1	CDCA3 E2F1 YWHAZ	WDR62	CENPM	EBP
VIJVER	DMFS	PRC1	POLD1 FEN1 EXO1	E2F1 TIMP3	PGK1 UBE2A	PGK1	DDX39 SNRPA1 PSMA7 PSMD2 POLR2B	STIP1	WDR62	UBE2C

Table 5. Top 10 subnetworks. Markers from DMFS, tested on VIJVER.

Tested	Markers	Chuang (PPI)	Chuang (T-score)	Chuang* (C2 V1.0)	Chuang* (Park)	Lee (C2 V1.0)	Lee (PPI*)	Lee (Park)	Park	Singles
VIJVER	SOS	ARF3 KIF23 AURKB AURKC RACGAP1 ARF5 ARF1	TRIM37 PRC1 DLG7 APEX2 KIAA0408 PNKP	PFKP HK2 GALK1	TK1	PFKP PFKL ARF1	BUB1 ARF1 NDRG1 ARF3 PPP2R1A UTP14A	TK1	TK1	E2F1
VIJVER	SOS	ARHGDIA FEN1 POLD1 CDC42	TK1 GAPDH UBE2A	PGK1 GOT1 ALDOC	DKFZp762E1312 TROAP	E2F1 ARF1 ARF3 CFL1	RRM2 PFKP PFKL	DKFZp762E1312 TROAP	TROAP DKFZp762E1312	TK1
VIJVER	SOS	FEN1 ARHGDIA POLD1 CDC42	TPT1	POLD1 RRM2	E2F1	RRM2 POLD1 TK1	PFKP ACP1 E2F1 PFKL DPM2	BIRC5	BIRC5	PRC1
VIJVER	SOS	PRC1	BLK BCL2 ITM2B SF1 RPS3A BNIP3L	TRIP13 PGK1	BIRC5	PFKP TPI1 PFKL HK2 HK3	PRC1	E2F1	E2F1	ESPL1
VIJVER	SOS	USHBP1 PRC1	EEF1A2 PSMD1 RACGAP1 CDC25A PSMB7 CHRM4	ARF3 E2F1 CFL1	E2F2	PFKP PFKL HK2 HK3	PRC1	PKMYT1	PKMYT1	BIRC5
VIJVER	SOS	GTF2H5 GTF2H4 CDC2 E2F1 TAF13 TAF4	SPG7 RALY WDR62 PLSCR1 CPSF6 VASP NPDC1 TXNL2	GOT1 AARS	AURKA	FEN1 POLD1 MSH6 EXO1	PRC1	TPT1	TPT1	E2F2
VIJVER	SOS	CFL1 TPI1 PGK1	PRC1	CPT1A TPI1	ADAM8	NDRG1 SLC19A1	PRC1	EBP	AURKA	TPT1
VIJVER	SOS	FENL1 FEN1 ARHGDIA POLD1 PRIM2A VCL EXO1	DCAMKL1 GAPDH UBE2A TK1	CPT1A TPI1	TPT1	PGK1 GOT1	PRC1	E2F2	E2F2	CCNB2
VIJVER	SOS	TPI1 PGK1 CFL1	USHBP1 PRC1	POLD1 MSH6 EXO1	PKMYT1	PFKP TALDO1 GPI PFKL	PRC1	AURKA	EBP	EBP
VIJVER	SOS	POLD4 POLD2 RFC2 POLD1 PRIM2A FEN1 EXO1 CDKN1A CCNA2 CDC45L	ARHGDIA FEN1 PRIM2A POLD1	PFKP ARF1	EBP	GOT1 AARS	PRC1	ADAM8	ADAM8	UBE2C

Table 6. Top 10 subnetworks. Markers from SOS, tested on VIJVER.

Master's Thesis

Work Document

A comparison of supervised gene set searching algorithms for outcome prediction of breast cancer

Thesis Committee:
Prof.dr.ir. M.J.T. Reinders
Dr. L.F.A. Wessels
Ir. M.H. van Vliet
Dr. E.A. Hendriks
Dr. G.W. Klau

Author	**Raul Kooter**
Email	Raul@xs4all.nl
Student number	1150162
Thesis supervisor	Dr. L.F.A. Wessels Ir. M.H. van Vliet
Date	September 23, 2009 - 14:00

Information and Communication Theory Group [I,C)^T

TU Delft

Faculty of Electrical Engineering, Mathematics and Computer Science

Delft University of Technology

1 Network-based classification of breast cancer metastasis: changelog

20th January

1.1 Results overview

1.1.1 Subnetwork significance scores

Table 1 shows the number of subnetwork which passed the test and the combined total number of genes.

Dataset	Crit.	Max degree	All	Test 1	Test 1, 2	Test 1, 2, 3	Genes
Vijver	MI	Inf.	8141	332	107	107 (149)	575 (618)
Wang	MI	Inf.	8141	456	206	206 (249)	877 (906)
Vijver	MI	1000	8141	210	100	100	496
			8141	449	163	163	660
Wang	MI	1000	8141	1054	290	290	1138
			8141	378	215	215	878

Table 1: Subnetworks significance test overview. The greedy algorithm was run on both datasets with Mutual Information (MI) criterion and the T-test (not shown). Also, the effect of a restriction on largely connected nodes (max node degree 200) was inspected. Note that the third significance test doesn't seem to have an effect on the selection of significant subnetworks. For the first two experiments, the number of subnetworks and genes according to the work of Chuang are indicated between parentheses. Also, some experiments are repeated.

The results in Table 1 are variable, the final number of subnetworks depends largely on the first significance test. The work of Chuang does not mention a maximum node degree, but a comparison with the cellcircuits website where their subnetworks are posted lead me to believe that a maximum node degree is in order not to grow gene HNF4A, which has more than 1500 connections. In the following work, I will be concentrating on the the results which are most similar to the work of Chuang, that is the 163 subnetworks from Vijver and the 215 subnetworks from Wang (both derived with the max 1000 degree restriction).

1.1.2 Subnetwork size distribution

The distribution of the sizes of all candidates subnetworks and the significant subnetworks are shown in Figure 1.

1.1.3 Top subnetworks

I've also inspected the 10 best subnetworks from the set of subnetworks selected using MI and with the max 1000 nodes restriction. I've ranked the 10 best subnetworks according to Mutual Information score. The top 10 subnetworks are listed in Table 1.1.3 and Figure 2.

The two top 10 best subnetwork only have 5 genes in common.

To compare the selection of subnetworks and the selection of single genes, I've included the top 10 single genes.

For further comparison, I've also inspected the top 10 best subnetworks using MI and without the max 1000 nodes restriction. The top 10 subnetworks are listed in Table 1.1.3 and Figure 3. Here we note that almost all top subnetworks according to MI score have node HNF4A in common. Also, the MI scores are higher than the top subnetworks found in Table 1.1.3.

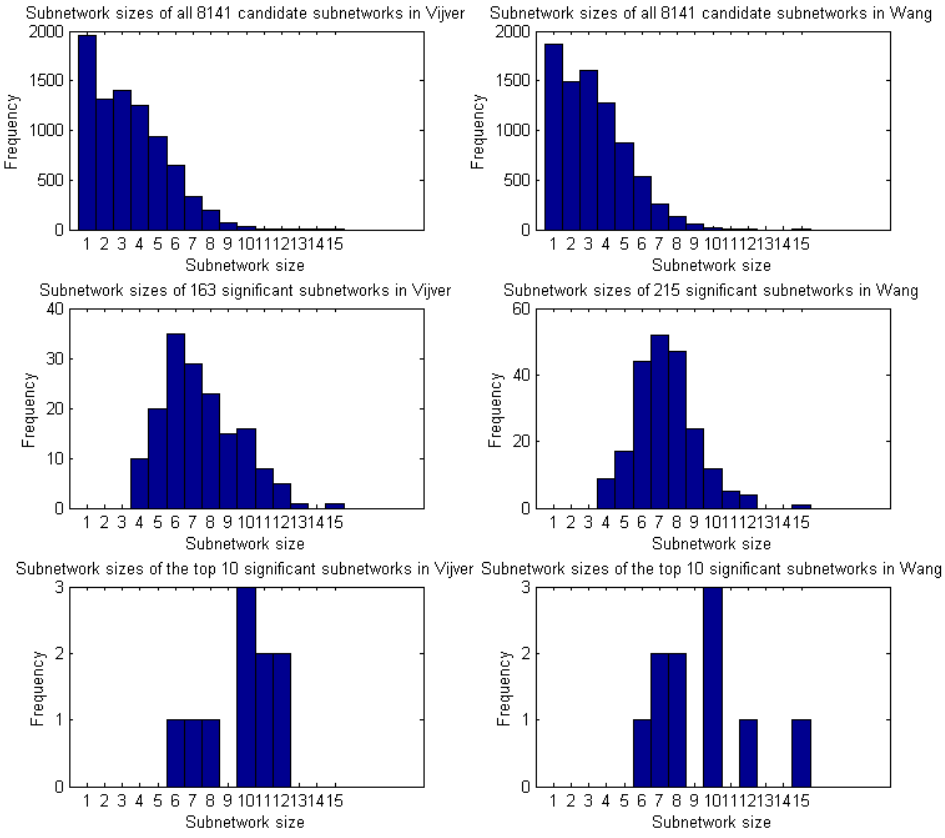


Figure 1: Distribution of the subnetwork sizes from both the Vijver and Wang dataset. The top 10 were selected by ranking the subnetworks on MI score.

Vijver genes	MI Score	T-test	Wang genes	MI Score	T-test
ASPA, ONECUT1, E2F1, FLJ10415, PRR1, PON1, ELOVL1, BM039, MAP2K5, FLJ11029, CRADD	0.14907	-6.9152	DUSP3, MAPK1, COPS5, MAPK9, NEK2, MADH3, HEY1, PSMC6, SHC1, ITGAV, RPS6KA4, VAV1, CDK2, CDK5, MYPT2	0.16126	-9.3513
SRC, MATK, EWSR1, ADRBK1, RAD23A, GFAP, MUL, VEGF, YWHAB, VTN	0.1365	-6.1917	NHP2L1, HNRPH2, SNRPA, NCL, GNB2L1, TOP1, POLR2E, SNRP70	0.141	7.8263
MAN2A2, IKBKAP, KIAA0098, PLP2, ITGA5, GIT1, ADRBK1, PPFIA2, HMOX2, XIP, PXN, PFDN1	0.13392	-7.1414	SNRPA, NHP2L1, HNRPH2, NCL, GNB2L1, TOP1, POLR2E, SNRP70	0.141	7.8263
PTN, CPR2, PSMD2, NR4A3, PSMD8, P4HB, SLC20A1, CDC6	0.13325	-6.3552	GTF2A1, POLR2E, NHP2L1, HNRPH2, NCL, TOP1, RPS5	0.12621	7.5469
KITLG, EPOR, STAT5A, USP4, FOS, RPL4, MYB, JAK2, RB1, TGFB1, RPS9	0.13129	7.7045	SNRP70, NHP2L1, CD69, EIF4G1, IFNG, POLR2A, NLI-IF, NONO, ITGB2, GTF2A1	0.12197	6.3202
APCS, ONECUT1, DKFZP727M231, E2F1, FLJ11029, AKR1C4, ELOVL1, ACPP, NCOA3, HBOA	0.13096	-5.9959	NCL, GNB2L1, NR3C1, SYT1, IFNAR1, TOP1, STAT3, STAT4, RPL6, SP1, TAT, SMARCE1	0.12135	6.1034
HMMR, MAPK3, GIT1, HSF1, MAPK9, STK3, DKFZP434D156, AKT1, DAPK1, SHC1, PTPRR, MAPK12	0.13055	-5.8541	STAT5A, MYC, TFAP2B, CSF1, USF2, NMI, IL4	0.11996	5.5416
TNFRSF10C, YWHAZ, LTBR, CDC25B, RGS3, HDAC5	0.12954	-5.8945	PPIA, PPP3CA, RB1, MAPK9, ASGR2, TMSG1, P84, ABL1, RFP, DKFZP564J157	0.11659	-3.398
H4FK, ONECUT1, E2F1, BCKDHA, PCAF, AKR1C4, 7-60, OAZ2, APOH, PRR1	0.12831	-5.8693	HNRPH2, NHP2L1, SNRPA, NCL, GNB2L1, PRKCA	0.11336	6.7524
PRKDC, HSF1, GTF2A2, POLR2C, SUPT5H, TREX1, HCNP	0.1262	-5.4476	ALK, SHC1, INPPL1, GHR, PSMC6, KRAS2, NRAS, P85SPR, SNT-1, PIK3CA	0.11071	-6.9089

Table 2: Top 10 subnetworks

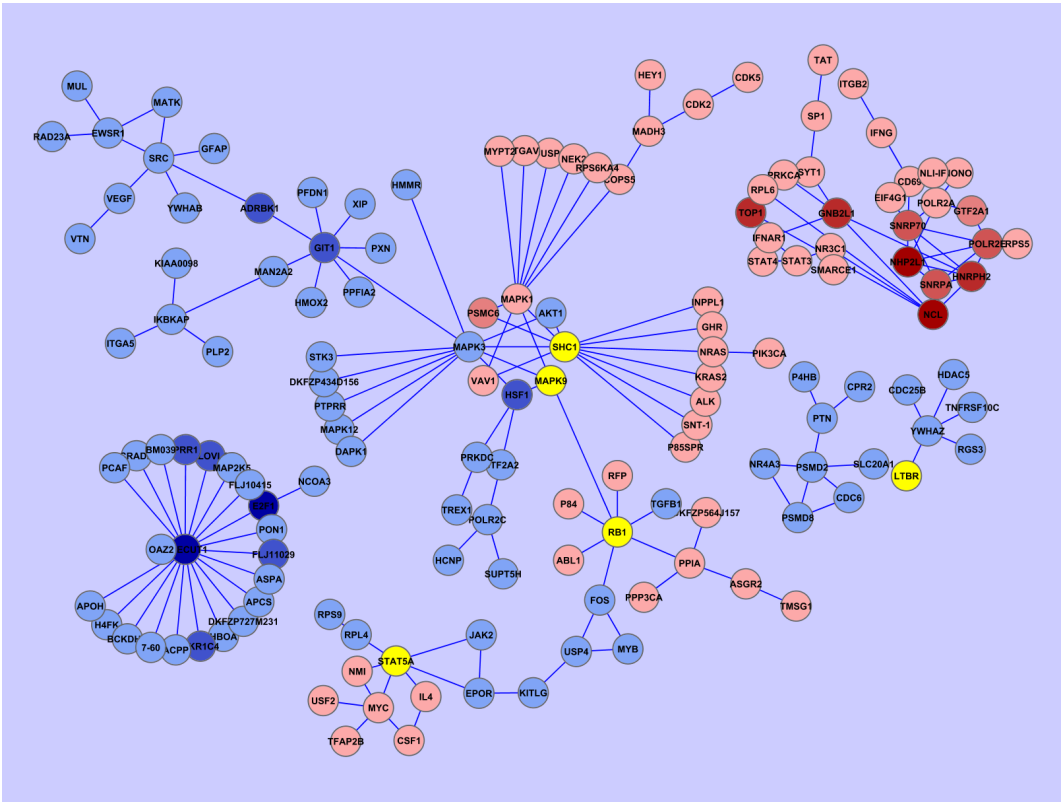


Figure 2: The top 10 subnetworks from both sets combined. The genes from the Vijver subnetworks are indicated in blue, from Wang are indicated in red. The darker a node, the more often it appears. Nodes from both datasets are indicated in yellow. Also, only the within-subnetwork edges are shown, edges which connect genes between different subnetworks a hidden.

Vijver genes	MI Score	T-test	Wang genes	MI Score	T-test
PRC1	0.065814	-6.1666	NR0B1	0.048555	2.9245
DDXL	0.064802	-5.5205	RAF1	0.045755	1.8241
DEEPEST	0.061376	-5.4321	NHP2L1	0.043553	4.8031
E2F1	0.061079	-5.9182	FLJ10998	0.042217	3.4765
BIRC5	0.058636	-5.665	EBI2	0.040824	0.4984
KIAA0165	0.057702	-5.65	KIAA0010	0.040279	-3.6297
TK1	0.057521	-6.125	GPR48	0.040172	-1.2769
PGR	0.056566	4.55	GCP60	0.039936	-3.9462
BYSL	0.054867	-3.6899	FBP2	0.038862	3.5767
ODC1	0.053988	-1.4628	ID-GAP	0.037586	-4.4431

Table 3: Top 10 single genes in both datasets. Only E2F1 and NHP2L1 also occur in the top 10 subnetworks.

Vijver genes	MI Score	T-test	Wang genes	MI Score	T-test
BHMT, HNF4A, CDC45L, HSPC164, TEAD3, ATF7, L2DTL, HAL, TRPC5, TXNRD1, FLJ10415, POLD4, MAP2K5	0.18343	-9.2905	FNTB, HNF4A, DKFZP566O1646, LOC51631, LOC57107, LIMS1, HSPC160, CDK5, pcnp, FLJ10640, SIX2, LOC51096	0.18285	-10.4192
SLC17A2, HNF4A, FLJ13912, PSMD7, LOC51142, NR2C2, LRN, ADH6, DCK, LOC56834	0.17649	-4.8129	STAM2, HNF4A, HECH, LOC57109, SPOCK, FLJ10640, FLJ10511, HSPC166, DKFZp434E2220, MGST3	0.17843	-8.7979
DSCR3, HNF4A, EXO1, PPGB, TRPC5, AF093680, C2ORF1, SLPI, NDRG1, COX7A2, GRO3	0.17034	-5.8147	HECH, HNF4A, STAM2, LOC57109, SPOCK, FLJ10640, FLJ10511, HSPC166, DKFZp434E2220, MGST3	0.17843	-8.7979
PCYT1A, HNF4A, PSMD7, L2DTL, GK001, BCS1L, GABRE, B3GAT1	0.16811	-6.3851	IRF6, HNF4A, SRP54, pcnp, LIMS1, H326, HECH, MSMB, AKR1C2	0.17005	-8.0761
YKT6, HNF4A, EXO1, LOC55972, FLJ20619, RAB2, RIP60, GNG5, TDRKH, FLJ10142, NR2C2	0.16423	-4.4125	NFE2L1, HNF4A, DKFZp434E2220, HECH, STAM2, JM23, CDK5, ACTA2, PSMB5, AKR1C2	0.16579	-7.7002
TADA3L, USP5, HNF4A, EXO1, CTSZ, CDC45L, SRP54, ATF7, TAF2E, CPT2	0.15974	-7.1358	KIAA1226, HNF4A, STAM2, LOC51096, NRAS, TUFT1, pcnp, ERO1L, LOC57107, LOC51644, KLRF1, HSU79274	0.16332	-9.3748
HAAO, HNF4A, RAD51, TRIP3, GSK3B, NSEP1, LSM3, NR5A2	0.15899	-5.0782	H4F2, HNF4A, STAM2, HECH, HSPC164, HBS1L, DKFZp434E2220, GCHFR, NDRG1	0.16302	-9.0797
ATP10C, HNF4A, PSMD7, FEN1, PSMD1, CTSZ, FLJ20619, SUDD, SF3B4	0.15888	-5.8826	DUSP3, MAPK1, COPS5, MAPK9, NEK2, MADH3, HEY1, PSMC6, SHC1, ITGAV, RPS6KA4, VAV1, CDK2, CDK5, MYPT2	0.16126	-9.3513
HSPC072, HNF4A, ASGR2, SREBF2, SCYE1, GSK3B, HSPC160, LRP5, MDFI, UXT, C14ORF1	0.15744	-3.683	HADH2, HNF4A, STAM2, LIMS1, LOC51659, RPS6KC1, HEY1, KIAA0141, DPM2	0.16112	-7.5496
MOCS3, MOCS2, HNF4A, NDRG1, TARS, EIF4G1, PSMD7, MRPL3, EXO1, MTHFR, KIAA0477, PZP	0.15718	-6.635	MGAT4B, HNF4A, HECH, KRT10, LIMS1, SRP54, VSP45A, STAM2, ACTA2, 54TM, VATD, SSBP	0.16104	-9.1553

Table 4: Top 10 subnetworks without the max node restriction. The MI scores a higher, but HNF4A appears in almost every network.

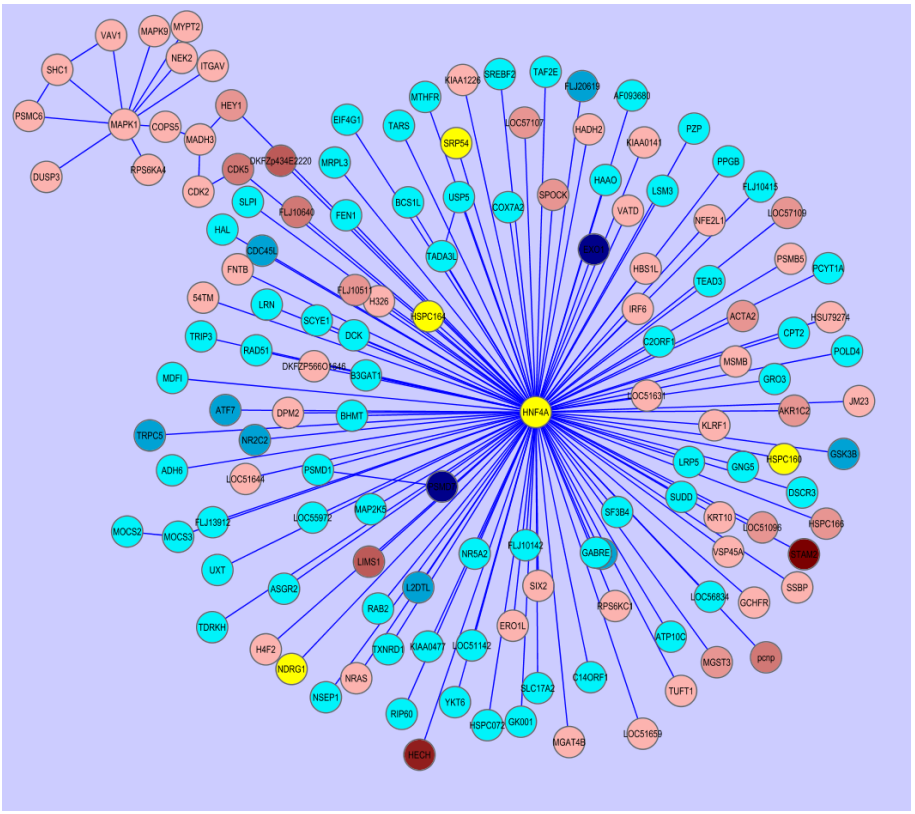


Figure 3: The top 10 subnetworks from both sets combined, without the max node restriction.

1.1.4 Classification

To test the classification performance, I've inspected the method described by Chuang. For now I've skipped the Double Loop Cross Validation strategy, which has a reduced bias, but I don't expect to see large differences in results.

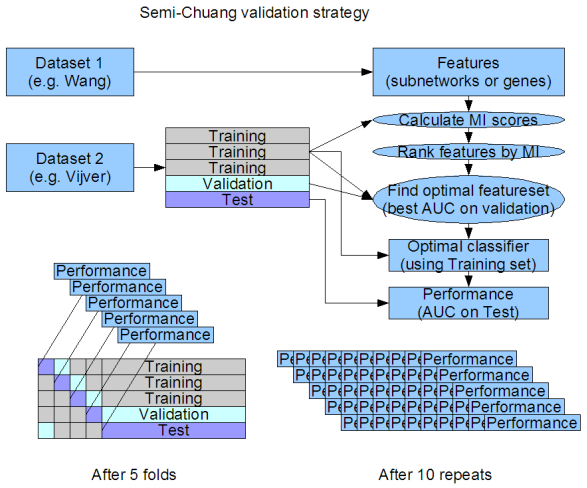


Figure 4: Chuang's classification evaluation method

Chuang's method In Chuang's original paper, the classification performance was measured using a cross validation. Given subnetworks or genes and a dataset, the dataset was transformed. After transformation, the dataset is split into 5 parts: 3 parts make up the training subset, 1 part is the validation subset and 1 part the test subset. The training subset is used to train the logistic regression model. First, the features where ranked according to P3, the P values of the third significance test. The number of features was optimised by evaluating the validation subset (by optimizing the AUC of the ROC), resulting in a optimised classifier. The optimised classifier is then tested on the test subset, also resulting in a AUC of the ROC. This entire procedure is repeated for 5 rotations, so every part gets to be part of the training, validation or test subset. The AUC for every rotation is then averaged resulting in a averaged AUC.

When reproducing the results, I've noticed that all significant subnetworks have P3 = 0, so ranking is meaningless in my results. Instead, I've chosen to rank the features according to Mutual Information score. Chuang also doesn't mention the number of repeats, so I've chosen to repeat the procedure 20 times. Also, when splitting the dataset in 5 parts, I've chosen to make sure this is a stratified split, even though Chuang doesn't mention that. I've limited the maximum number of features to 100 for speed, since it never seems to crosses that limit.

The classification results evaluated according to Chuang's validation strategy are listed in Table 5 and Figure 5. They are not comparable to Chuang's Figure 2B.

Input	Transformed using	Error	Optimal feats
Wang	Vijver significant subnetworks	0.4423	18
Wang	Vijver top 660 genes	0.4178	10
Wang	Wang significant subnetworks	0.1642	12
Wang	Wang top 878 genes	0.2819	17
Vijver	Wang significant subnetworks	0.3541	21
Vijver	Wang top 878 genes	0.3748	23
Vijver	Vijver significant subnetworks	0.2061	20
Vijver	Vijver top 660 genes	0.2847	15

Table 5: Classification performance

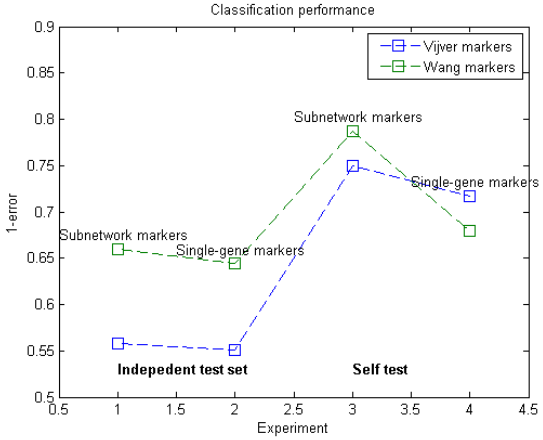


Figure 5: Overview of the classification results

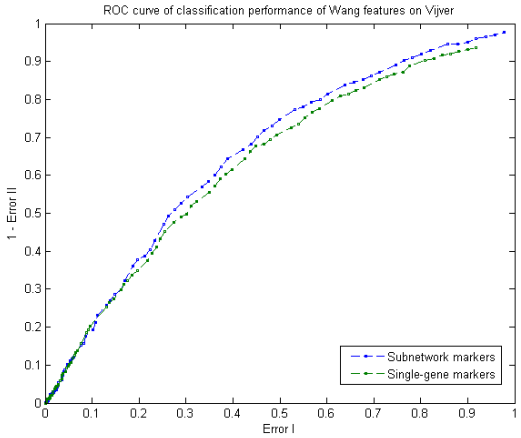


Figure 6: Boxplots the classification results

As an example, I've included the ROC curve of the logistic regression trained on Wang tested on Vijver. This time, a fixed number of features, 19, where chosen from the transformed Wang dataset using a forward selection while optimizing the intra-inter distance. This is the optimal set of features when testing Wang on Wang. Using these features, a Logistic Regression Classifier was trained on these 19 features of the transformed Wang dataset. The Logistic Regression was tested on the transformed Vijver algorithm. See Figure 7.

1.2 Analysis

1.2.1 T-test versus MI scores

When ranking the discriminative power of the subnetworks, both the t-test and the MI score might be used. While the t-test mostly focused on the difference of the mean and the variances, the MI score is able to discriminate samples with the same mean, but with different variances. The relation between MI scores and t-test scores is depicted in Figure 8. The trend between between MI scores and t-test scores is best shown in the Wang subnetworks, where a higher MI score is probably a higher absolute t-test score.

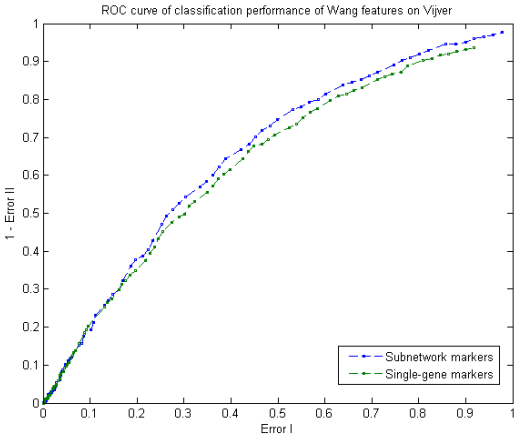
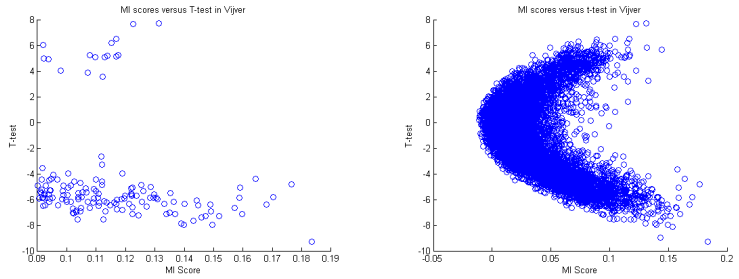
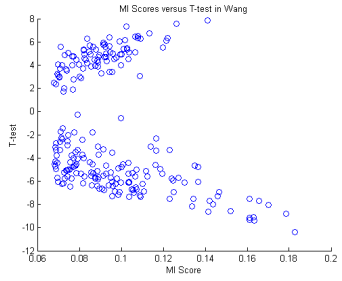


Figure 7: ROC curve of Logistic Regression trained on Wang tested on Vijver.

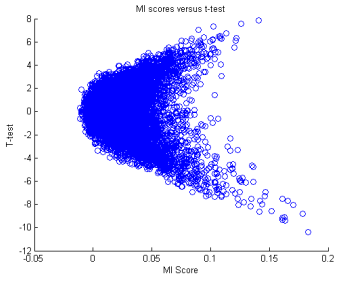


(a) Significant subnetworks (Vijver)

(b) All subnetworks (Vijver)



(c) Significant subnetworks (Wang)

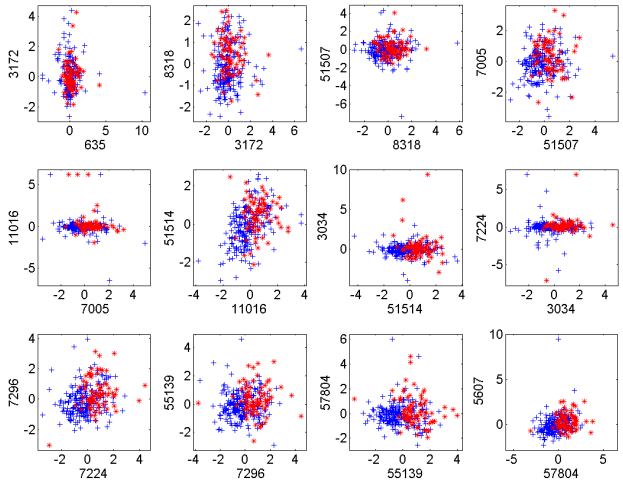


(d) All subnetworks (Wang)

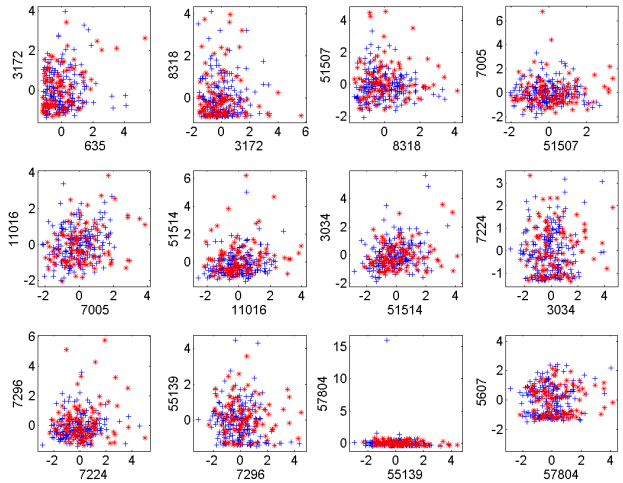
Figure 8: MI scores versus T-scores in the Vijver and Wang subnetworks.

1.2.2 Multivariate effects

The test the effect of multivariate effect inherent in subnetworks, I've tested how the nodes in the best subnetworks correlate. For example, the best subnetwork in Vijver, tested against the Vijver dataset. The axes depict a gene, or a combination of genes, the color depicts the outcome:



(a) The best subnetwork in Vijver tested against itself.



(b) The best subnetwork in Vijver tested against Wang.

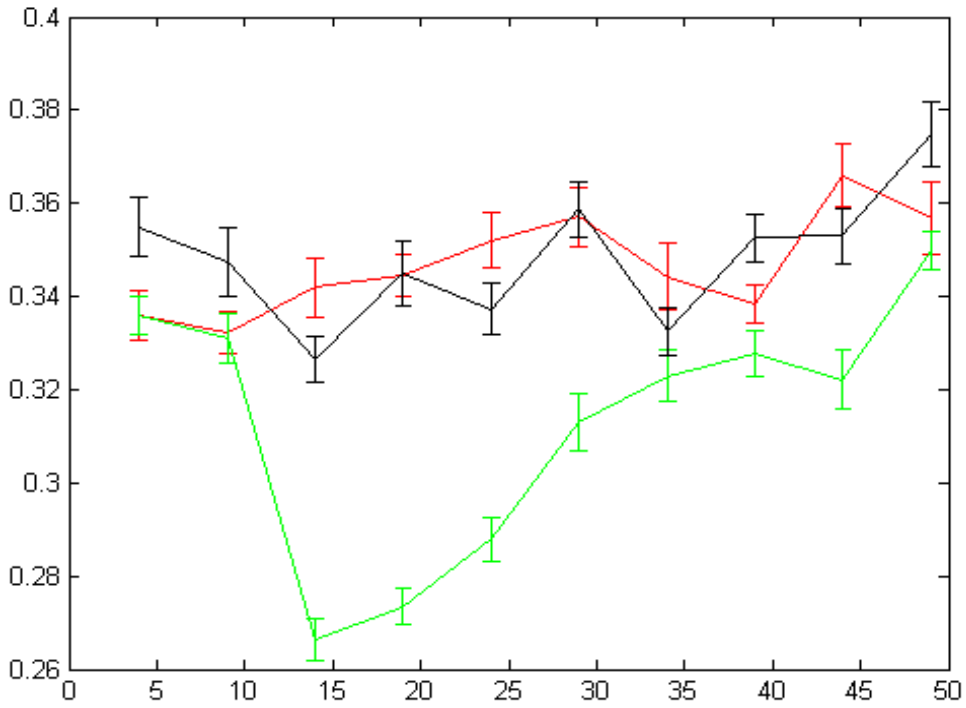
Figure 9: Please note there is a mistake in the axis labeling, the lower label depicts the combination of all previously defined genes, so for example in the third graph, 8318 should be interpreted as 635, 3172 and 8318.

Febuary 5th

Validation method revisited

As mentioned earlier, my validation method gives a worse validation result than mentioned in the paper.
I've attempted to 'fix' the validation procedure. When testing the subnetworks of Chuang on Vijver, I've noticed a strange thing. I've generated the following learning curves by fixing the number of features (no inner loop), and reranking the features.

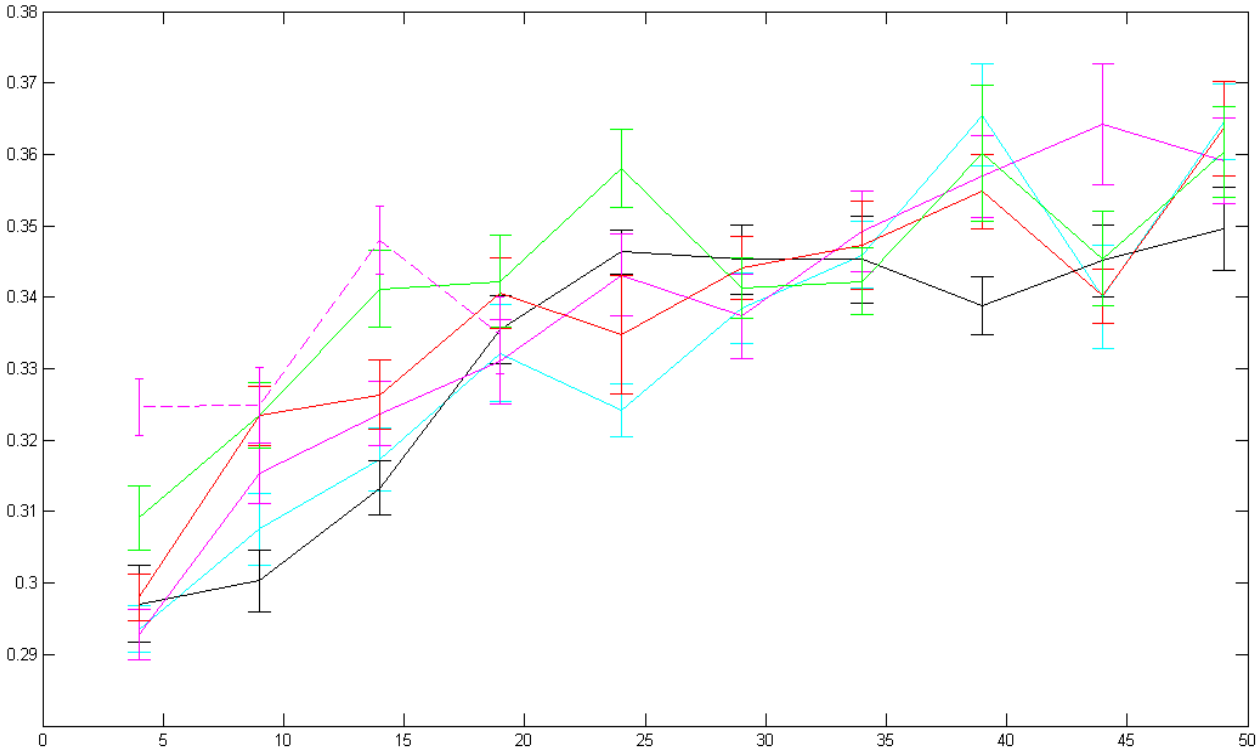
Each point was calculated 50 times. The errorbar indicates the variance, not standard error.



X axis: number of features (4:5:50)
Y: error, lower is better
Black line: random ranking
Red line: ranked in MI
Green line: no ranking (apparently it's alphabetic ranked on starting node)

Since alphabetic ranking seems to perform so well, I've figure it's because it's because it's a random ranking. After all. This is not entirely true. All I can conclude is that alphabetic is just a good ranking by accident.

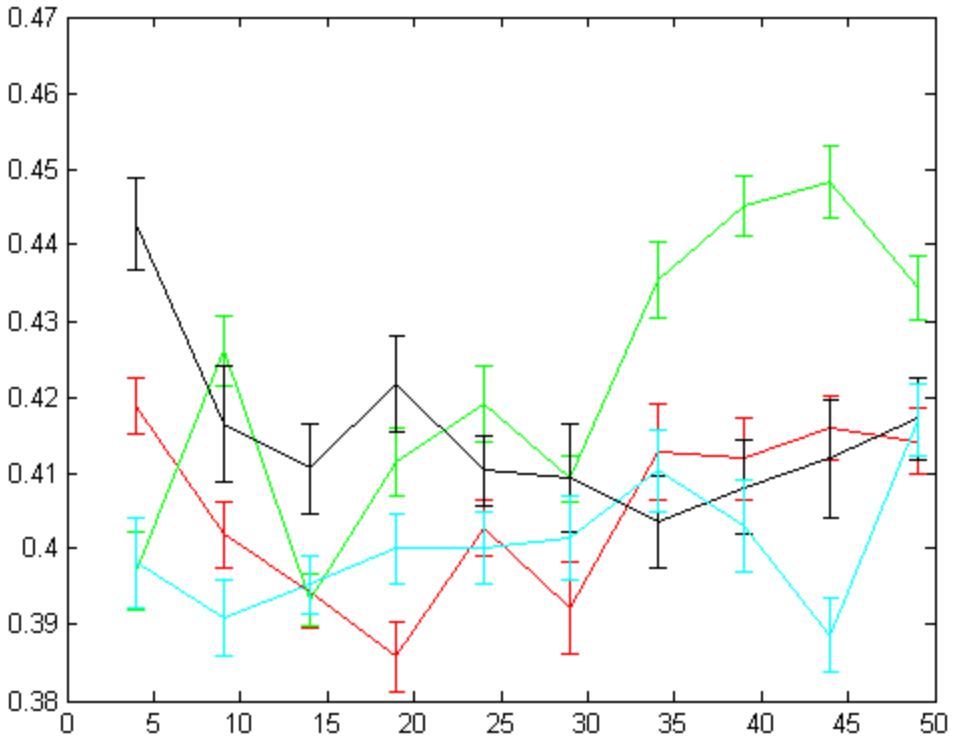
The features ranked by P3 is a different story.



Light blue: P3 test with 50 permutations
Black: P3 test with 100 permutations
Magenta: P3 test with 200 permutations
Red: P3 test with 500 permutations
Green: P3 test with 2000 permutations
Dashed magenta: P3 test with 5000 permutations (maximum of 19 features due to time)

Do less permutations together with few features mean a better ranking?
Probably not. Less permutations causes more p3 values of 0, thereby making the ranking more close the original ranking.
So why is the error around 4 features better than the original ranking? I think this is due to the fact that a few bad genes in the original ranking are moved to the back of the list, thereby combining the goodness of the original ranking and the ranking by p3.

What about the reciprocal test? (Vijver subnetworks tested on Wang?)

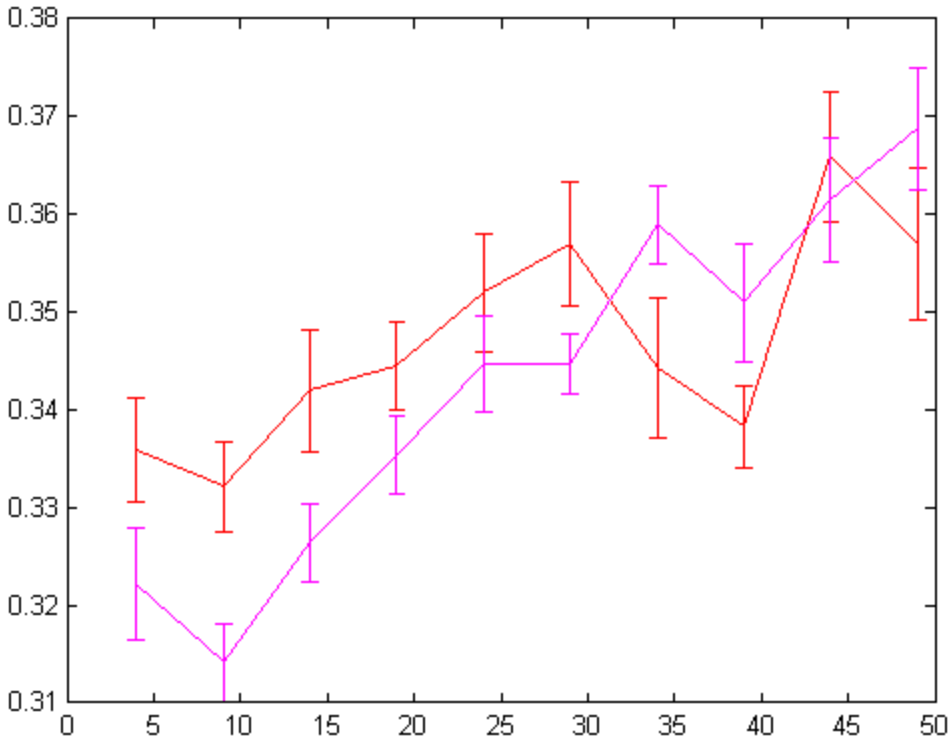


X axis: number of features (4:5:50)
Y: error, lower is better
Black line: random ranking
Red line: ranked in MI
Green line: no ranking (apparently it's alphabetic ranked on starting node)
Light blue: P3 test with 50 permutations

I think from this we can conclude the original ranking, the green line, is not better than random ranking or sorting on MI.

So how about sorting by T?

Going back to the Wang subnetworks tested on Vijver situation:



Red line: ranked by MI
Purple line: ranked by t-test

Overall, I think it is best to use the T-test to rank markers. Probably because mutual information is not good in combination with linear classifiers such as logistic regression.

It would also be nice to look at the P3 value of the T-test, though.

1 Changelog

February friday 13th

1.1 Classification performance

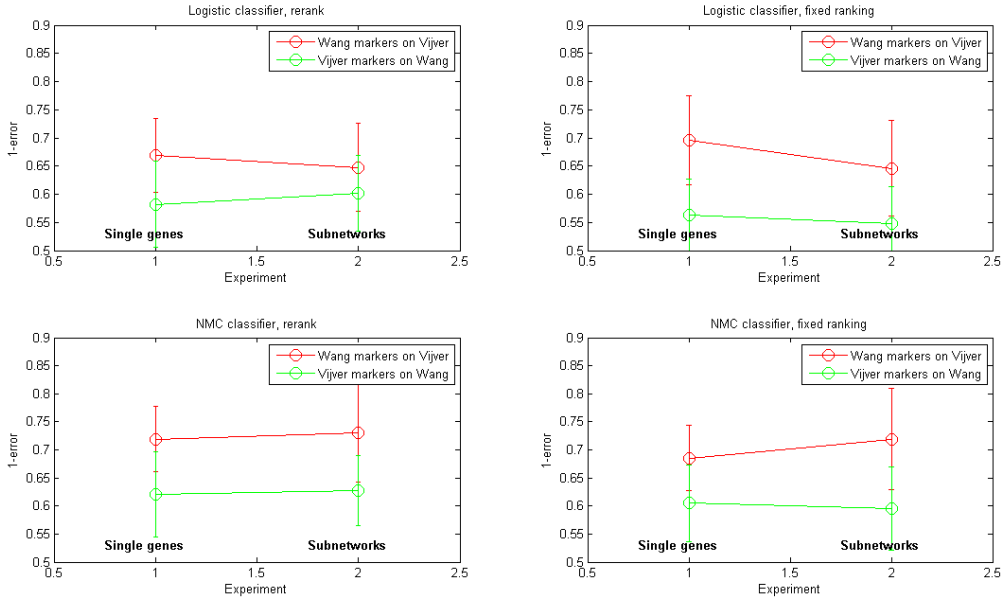


Figure 1: Overview of the classification results. Outer loop: 5 parts (4 training, 1 testing). Inner loop: 4 parts (3 training, 1 testing). In the inner loop, the features where either ranked on the t-test using the training part determine the t-test score, or the the markers had a fixed ranking, by ranking the t-test score of the Wang markers on the entire Wang dataset. Ranking was done by taking the absolute value of the t-test score. The optimum number of features was determined by adding 1 to 100 features and generating a learning curve of the AUC on the test data. This was repeated 8 times, after which the 8 learning curves where averaged to determine the number of features which gives the best AUC. In the outer loop, the features where once again either ranked using the entire training set, or had a fixed ranking. The previously number of markers were limited to the previously determined optimum number of features. The procedure was repeated 10 times, so 50 AUCS where averaged. Two different classifier were used, Loglc and NMC.

Apparently using a Nearest Mean Classifier and using the reranking method instead of a fixed ranking gives an AUC overview which looks most likes Chuang original overview. From now on, this strategy will be used for further evaluations. Furthermore, it is not convincing yet that subnetwork markers perform better than single genes (which where controlled for size in this experiment), as this depends on the method of validation.

1.2 Chuang's response

I asked about the implementation of the cross-validation method. According to Chuang herself, her experiment consisted of only 1 repeat. The inner loop itself was also a 'on-time-shot'. This makes me a little skeptic about her final AUC.

I also asked about how they worked with 'empty nodes'. A gene starting with an empty node was calculated as: (sqrt(2) * genevector2 + genevector3 + genevector4) / sqrt(4). Chuang:

> In such fashion of scoring, we punished a subnetwork of an empty starting node in a less harsh way. Although the PinnacleZ package was not used in our MSB

paper, it implemented what we have done for network search in the paper. However, when we did the classification evaluation, we only took the nodes of expression into account for subnetwork activity inference. Therefore, given the same example here, the activity would be slightly different in classification. It would be imputed as (genevector2+genevector3+genevector4)/sqrt(3).

1.3 Implementing Lee's algorithm

I've implemented the algorithm as described in Lee's paper.

I've downloaded the C2 functional sets from MSigDB (both V1.0, the version they've used and V2.5, the newest.)

The functional sets didn't map perfectly to entrez id's, therefore, my collection pathways only consists of the genes for which I was able to find an entrez id, it didn't matter if that entrez id was in my datasets.

To test the function I've used the same datasets used in Chuang's method, that is, using 8141 genes. This doesn't make quite sense, since Chuang selected the 8141 to be genes in the IPP network, but for a good comparison I will stick to this.

I've implemented Tian's methods to preselect a top 10 percent pathways from the C2 function set V1.0. This was done by calculating the t-score of each gene for which I have expression data in the pathway. I've done this so I have a good comparison with Lee's supplementary figure 2.

Of course there are a few pathways missing, but this might be due to different genesymbol-to-entrez mapping, due to different preparation of the Wang and Vijver datasets or due to different implementation of Tian's method. For now, I think this is a good approximation, even though I'm able to get a better one by playing around with the parameters.

Table S2. Frequently selected pathway markers for breast cancer prognosis.

Pathway Name	Frequency	# genes *	CORGs
From Netherlands to USA			
Cyclin regulated genes	416/500	2/13	E2F1 *CCNE2*
IL7 pathway	200/500	3/16	BCL2 STAT5A IL7
Cell cycle	197/500	3/84	E2F1 *ESPL1* CCNB2
ActinY pathway	142/500	3/19	PIR PSMA7 ACTR3
GNF female genes	123/500	3/85	RPS4X RPS6 RPL6
From USA to Netherlands			
Cell Cycle	500/500	4/84	CCNE2 *ESPL1* MAD2L1 CDK2
Brentani cell cycle	282/500	4/86	CCNE2 MAD2L1 CDK2 MXD1
Cyclin regulated genes	280/500	3/13	*CCNE2* CDK2 CCNA2
KRAS up-regulated genes	202/500	3/84	TUFT1 P4HA2 COL4A1
Cell cycle checkpoint II genes	200/500	2/10	CCNE2 FANCG +RB1
Glutamine down-regulated genes	167/500	6/313	TCEB1 KPNA2 CYCS TMED9 UTP18 MORF4L2
MMP/Cytokine connection	148/500	5/15	~~DEAF1~~ TNFRSF1B CD44 IL1B ~~TGFB2~~
Leucine down-regulated genes	136/500	6/180	TCEB1 KPNA2 CYCS TDG CCT6A CSE1L
IL22 pathway	124/500	3/13	SOCS3 STAT5A STAT3
Rapamycin down-regulated genes	111/500	4/229	STAU1 CYCS RAE1 MORF4L2

* The number of CORGs and member genes are specified.

** Pathways/Genes in italics are shared between datasets

Figure 2: Comparison of with supplementary figure 2. I've marked the pathways that my implementation of Tian is able to find, and the markers that are in the pathway my implementation is able to find.

I've also tested the robustness of the two datasets. This wasn't done using Lee's comprehensive 100-split method, but by comparing the markers derived from the two datasets. Overlap is calculated as number of genes in intersection divided by number in genes of union.

```
Label overlap   Genes overlap
```

C2 v1.0	0.0297	0.0105
C2 v2.5	0.1489	0.0610
Top 906/618 genes	0.0687	0.0687
Chuang	0.0182	0.1265

C2 v2.5 performs better in overlap than C2 v1.0, but this might be due to the fact that the top 10 percent have 52 and 189 subnetworks derived from the v1.0 and v2.5 pathways.

1.4 Testing Lee's subnetworks

I've tested Lee's subnetworks using the classification procedure described above, using a NMC and a reranking method, since that method was best gives me the 'best' result (that is, results I want to see.)

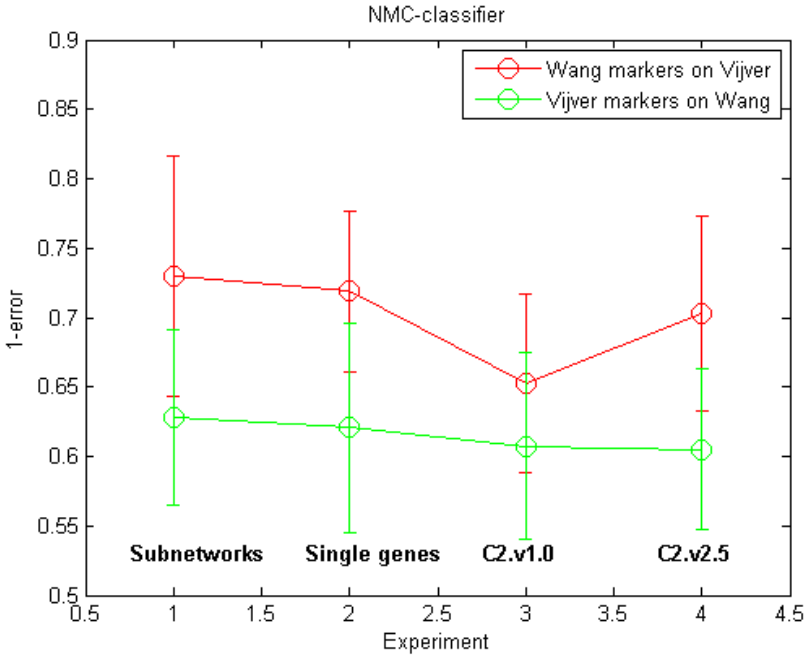


Figure 3: Comparison of all perfomances.

1 Changelog

February 19th

1.1 Changing the t-test

During the programming I noticed that I've used a two-sample t-test assuming equal variances. I will continue the tests using a t-test assuming unequal variances, since that seems more logical. Also, I've implemented an t-test assuming unequal variances which is 40 times as fast than Matlab's original, so obviously I'd prefer my own implementation.

Making this step makes my reproduction of Lee more different than from Lee's original results, unfortunately. This also has an effect on the classification procedure.

So how much does it differ?

For simplicity, let's plot the t-test with unequal variances against t-test with equal variances in subnetwork calculated by Lee from the Wang subnetworks.

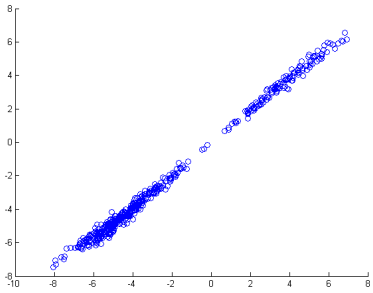


Figure 1: T-test assuming equal variance against T-test assuming unequal variance.

As you can see, it doesn't differ much.

1.2 Updated classification results

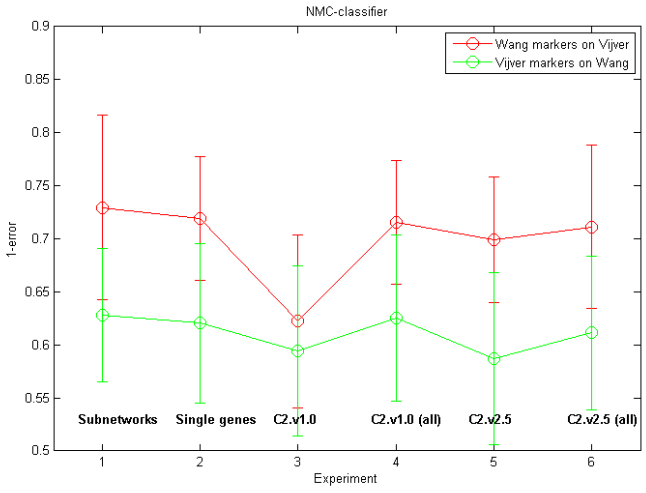


Figure 2: The performance of the Chuang subnetworks and single genes was untouched. The CORGS in the Lee methods and the evaluation procedure where calculated using updated t-test.

1.3 Lee-Chuang comparison: raul1 algorithm

In order to make a good evaluation which searching method works better, Chuang's or Lee's, I've compared them by modifying Chuang's algorithm.

Chuang's algorithm works by adding the best neighbour to a growing network as long as a minimum improvement of 0.05 occurs. This happens in a network distance of 2. In the new algorithm, instead of considering all direct neighbours, I consider all genes in the network distance of 2. The genes are ranked by t-test in a way similar to Lee's, and these are added in the sequential order until there is no improvement.

Effectively this algorithm is the same as Lee's algorithm, where the enforce a starting node, and the 'pathway' given is the set of genes in network distance 2.

One problem that remains is that of feature selection, since I've now ended up with 8141 subnetworks. For simplicity, I will select the top 5 percent of the subnetworks, ranked by absolute t-score, ending up with 407 top ranking subnetworks.

Let's look at the robustness:

	Label overlap	Genes overlap
C2 v1.0	0.0297	0.0105
C2 v2.5	0.1489	0.0610
C2 v1.0 (all)	1.0000	0.1944
C2 v2.5 (all)	1.0000	0.2275
Top 906/618 genes	0.0687	0.0687
Chuang	0.0182	0.1265
Raul1	0.0739	0.0970

Since the number of features differ in all of these marker sets, it is hard to draw a conclusion from this robustness analysis. Let's evaluate the classification performance again:

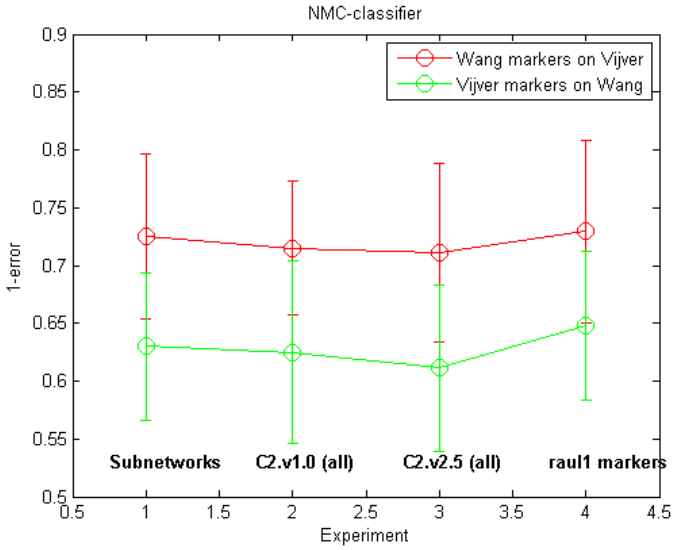


Figure 3: The performances of the aucs.

Strangely, the raul1 markers outperform all the others slightly.

1.4 Park algorithm

In order to simulate the idea of Park, I've employed the following method: I've clustered the gene data in a dataset using correlation as a distance and average linking for clustering. I've set a hard threshold on 500 subnetworks.

The resulting thresholds obviously have all genes in their subnetworks, so robustness analysis is useless here. It is interesting however to inspect the distribution of the number of features in each subnetwork.

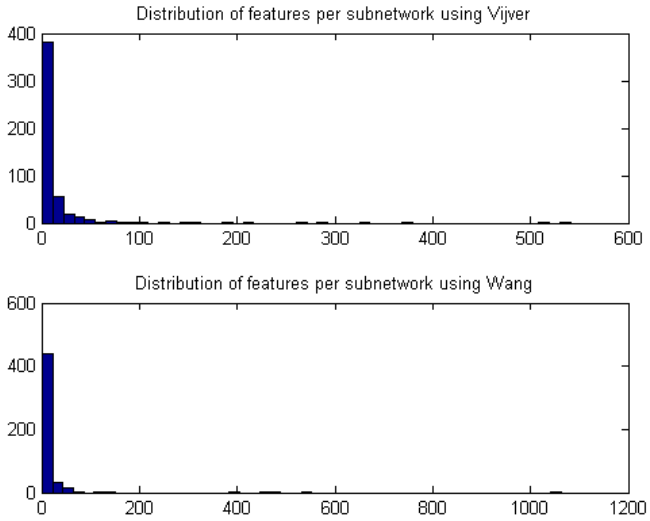


Figure 4: The distribution of the number of features per cluster. Each marker set consists of 500 clusters.

And the performance...

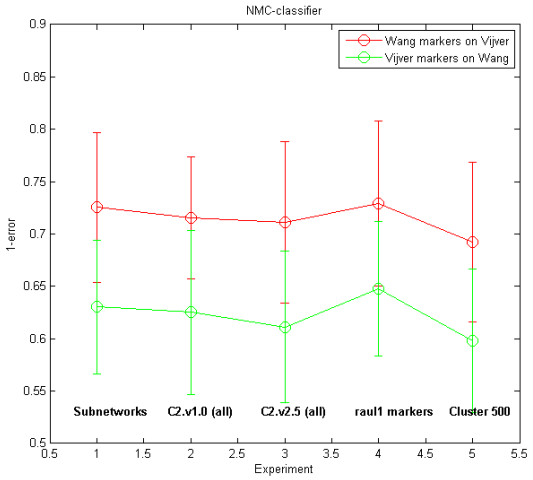
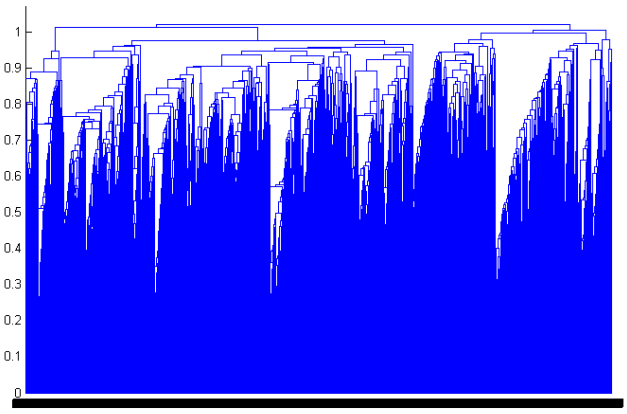
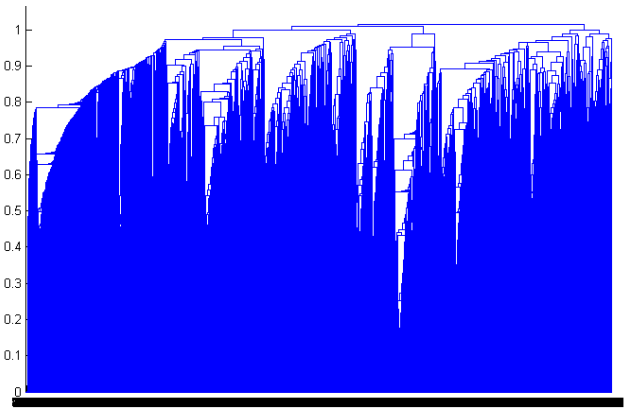


Figure 5: The performances of the aucs with the 500 clusters.

It looks like my first attempt at clustering using has failed. This could be due to the distribution of genes per cluster, the level of hierarchy or perhaps because of the lack of some feature selection method. I've also generated the dendrograms. Since there are 8141 features, it is hard to make a comparison visually.



(a) Dendrogram of Vijver features.



(b) Dendrogram of Wang features.

Figure 6: Dendrogram of the features, average linking, correlation distance.

1 Changelog

March 5th

1.1 Trying out the new datasets

I've received and integrated new datasets. Now I have a collection of 4 datasets, DMFS, SOS, VIJVER and WANG, each having the same 10870 labels. The Vijver dataset is now different from the dataset used in previous analyses, since the outcome labels and number of samples has changed.

1.1.1 Classification performance

I've tested out the different classification performances.

All the subnetworks or genes were ranked by T-test and in advance and the top 500 selection was taken, except for genes1000, where the top 1000 genes were taken.

For Park's algorithm, the complete set of genes was clustered to 8000 subnetworks. Again, from these subnetworks, the top 500 were picked. 8000 was the approximate average of all 4 optimal hierarchy levels.

There is also a new feature, doubles, which a set of subnetworks consisting of all protein-protein pairs.

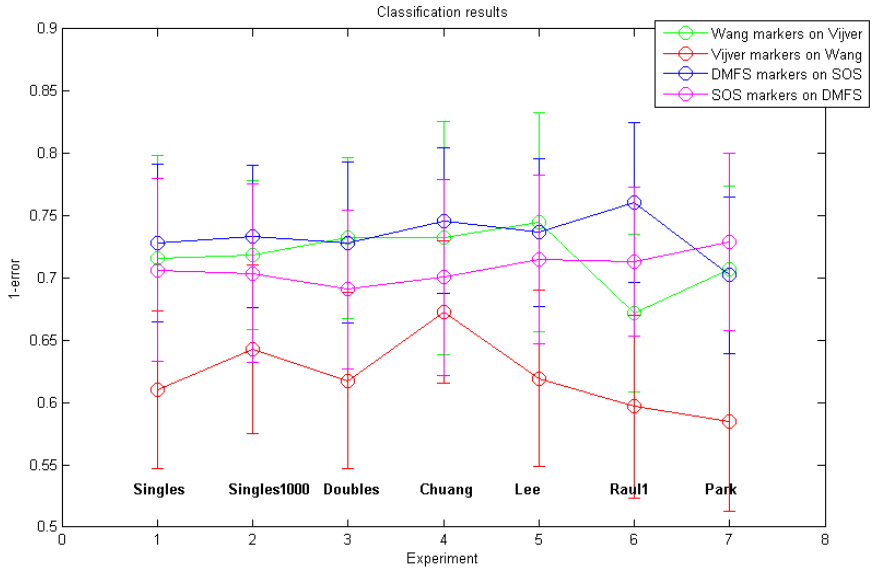


Figure 1: Classification performances

The results differs from previous work. It is hard to see which one really performs best.

1.2 Does Park really work?

I've plotted the cross validation error of Park's algorithm in Figure 2. The horizontal axis depicts the level of hierarchy, growing exponentially, and the vertical axis the number of features. The plot below the curve indicates the lowest possible error for that level of hierarchy. No specific trend can be found, so I'm doubting Park really improves anything here.

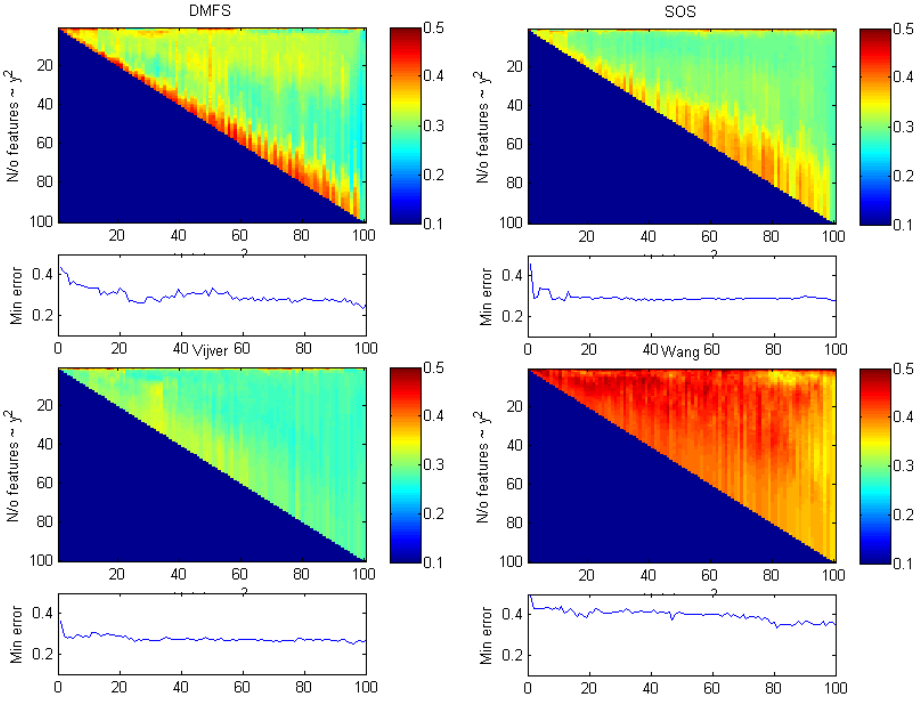


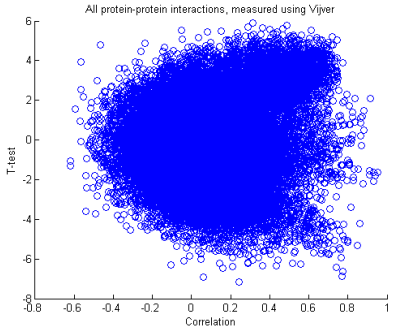
Figure 2: Park curves

1.3 Investigating biological phenomena

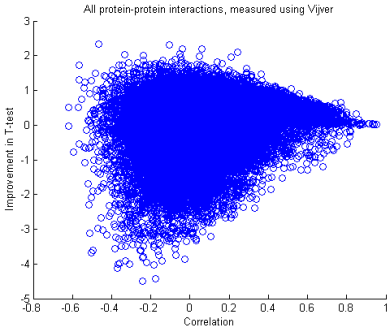
1.3.1 Pairs of genes

An interesting property to examine is whether how much correlation is linked to improvement in t-test score.

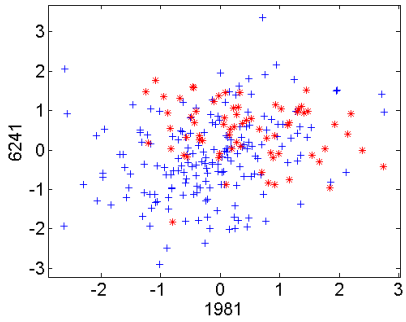
1.4 Suggestion for simulation



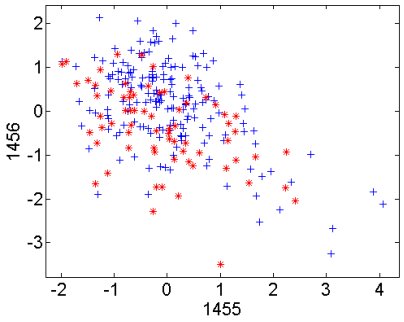
(a) Correlation versus t-test.



(b) Correlation versus improvement t-test.



(c) The protein pair with maximum t-ttest when averaged: EIF4G1, RRM2



(d) The protein pair with maximum improvement when averaged: CSNK1G2, CSNK1G3

Figure 3: Correlation versus t-test or correlation versus improvement in t-test. Improvement is calculated as abs(T-test) - mean(abs(individual t-tests)).

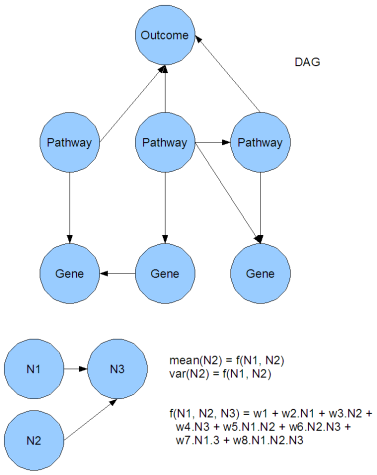


Figure 4: Simulation. Park's paper is a subset of this method.

1 Changelog

March 11th

1.1 Classification performance

Details

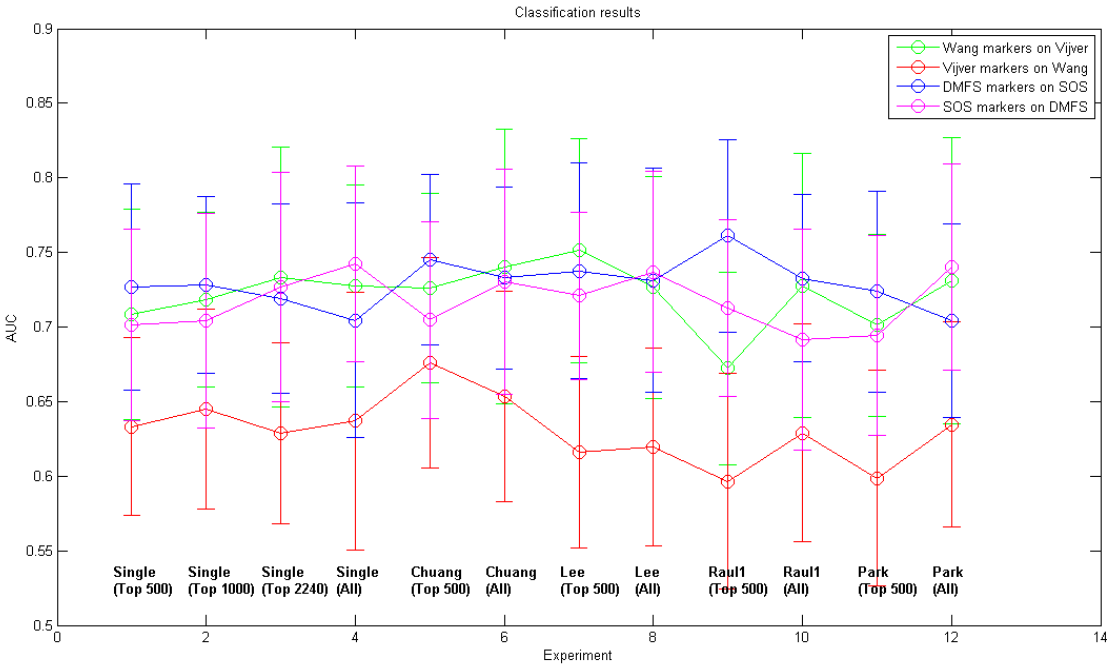


Figure 1: Classification performances

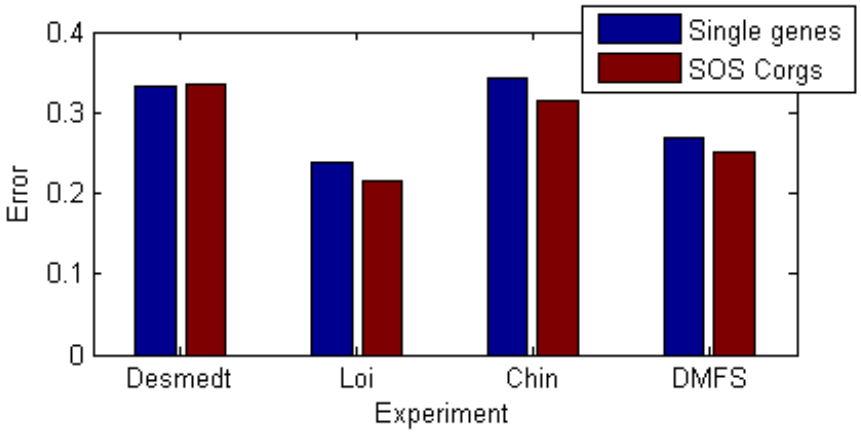


Figure 2: Classification performances, using CORGs trained using SOS, against single genes.

1 Changelog

March 24th

1.1 Classification performance

In order to start making conclusions about the various classification methods, I've performed a few experiments. A complete test can be seen in Figure 1.

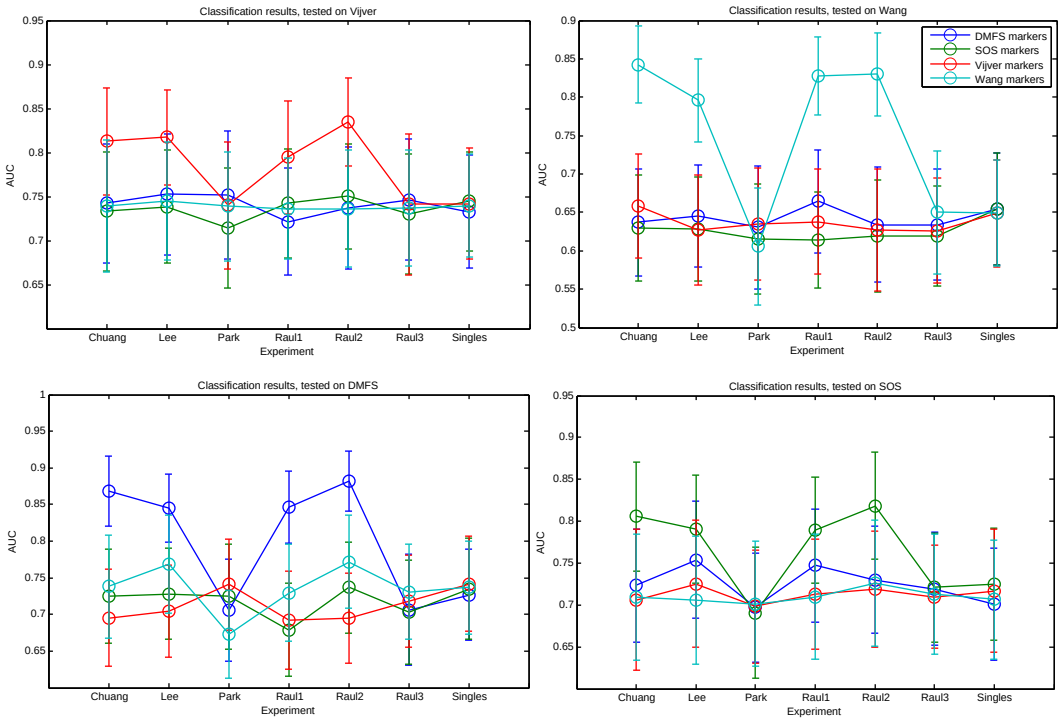


Figure 1: Classification performances, overview of all methods. Raul1 algorithm: Lee-searching in Chuang-search space. Raul2 algorithm: Chuang-style search in Lee-search space. Raul3 algorithm: Chuang's algorithm but tries to maximize average correlation among genes instead of MI-score.

This test uses compares all method without features selection. So in single genes, all 10870 genes are selected. To see the effect of feature selection on single genes, I'd like to refer to a simpler, earlier experiment in Figure 2.

So far, some conclusions I draw from previous observations:

- Park's algorithm doesn't work. I've noticed that the cross-validation method in order to select the optimum number of features prefers a high level, so, the final set of subnetworks are similar to the original set of genes, but even then it still performs worse than single genes. This can be seen clearly when SOS or Wang is the dataset of choice, in Figure 1. Also, even when using the markers trained on the same set, the results don't show the expected biased improvement. I suspect that performing a feature selection prior to Park's algorithm will improve Park's method. They also perform feature selection in their example of Vijver's dataset.
- Raul3 algorithm is a variation of Chuang's algorithm. It doesn't try to improve the MI score, but the average correlation in the subnetwork. This method also fails to show the expected biased improvement. I suspect doing anything with correlation alone won't improve results.

- In the lower-right graph, there is a slight symmetry between Chuang and Lee on one side and Raul1 and Raul2 on the other side. The symmetry can also slightly be seen in the lower-left graph. This might indicate that the searching method has a more important effect than the search space.
- It is hard to conclude wether Chuang or Lee works better. By inspecting the graphs, I would say Lee works better, even tough there are a few exceptions. If we leave out the biased results, so the markers trained on X and tested on X, then Lee often outperforms Chuang.
- None of the methods seem to significantly outperform the single genes. Okay, so in this experiment all genes were selected in single genes, so for a fair comparison you would have to control the genes for size. In Figure 2 the results can be seen when taking the top 2000 genes.

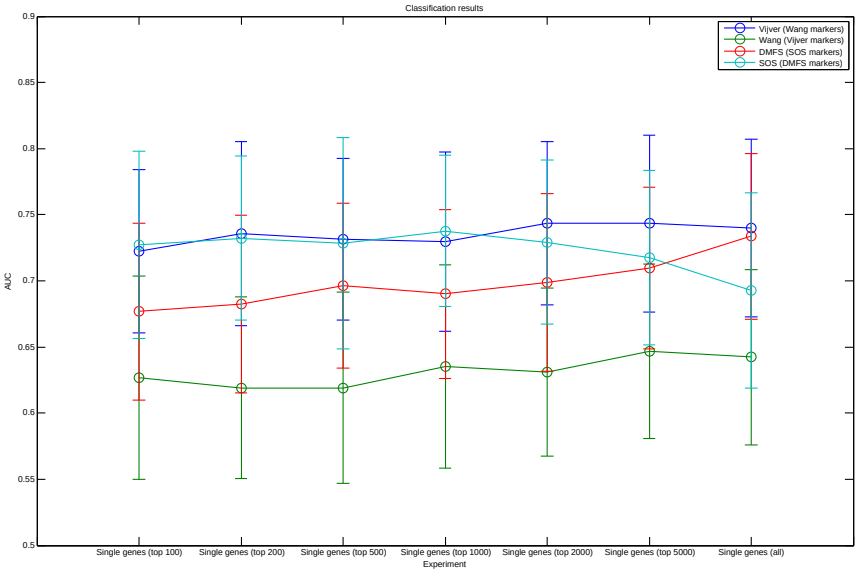


Figure 2: Classification performances for single genes. Here, feature selection was performed using the training set. So for example the top 500 genes were determined using SOS and tested on DMFS.

1.2 Feature selection

In order to investigate the effect of feature selection a little more, I've inspected the learning curves with respect to preselection. The basic feature selection method is ranking by T-score and selection the top-N genes.

From these figures I conclude that a few thousand top ranking genes should be selected for optimum results. Also, the DMFS and SOS learning curves seem to behave different from the Vijver and Wang learning curves.

1.3 Mysterious drop

Last report, I've noticed a sudden drop in performance, using the raul1 algorithm, with the top 500 ranking subnetworks from Wang tested on Vijver.

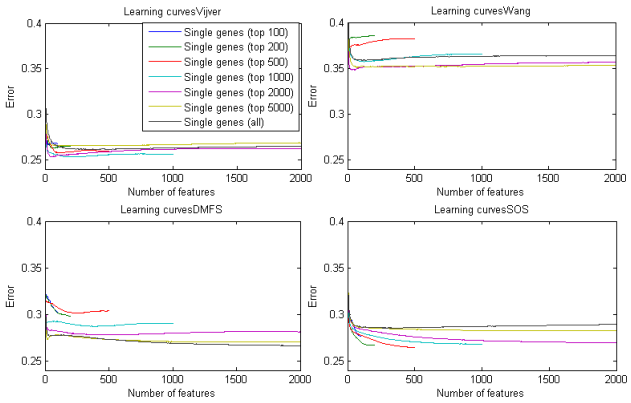


Figure 3: The effect of feature selection investigated. All these figures come where derived by averaging all the inner loop learning curves from the cross validation method. In these figures, Vijver-Wang, Wang-Vijver, SOS-DMFS, DMFS-SOS train-test method was employed. Before each cross validation method, the top-N genes where picked by ranking on T-score.

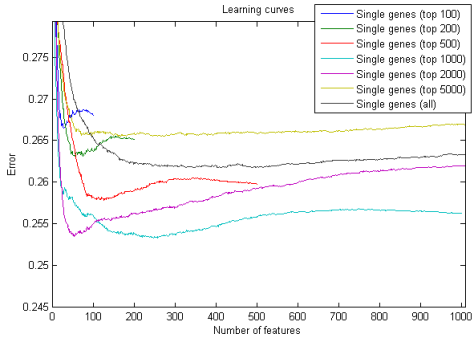


Figure 4: Vijver's learning curves further zoomed in. Once again, feature selection was performed using Wang, and the curves themselves come from Vijver.

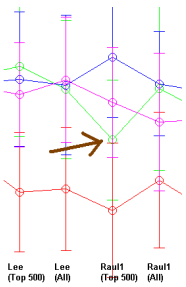


Figure 5: The drop.

Since I've improved the cross validation strategy (faster and now does 20 repeats instead on 10), I was not able to reproduce the drop. So I'm guessing some my previous plots weren't robust enough.

1.4 The use of models

First of all, why would we bother to create models?

- We can try to simulate data given a model and compare the various algorithms on it.
- We can try to analyze why a certain algorithm would work better in certain circumstances compared to other algorithms.
- It may help design improved algorithms.

From an analytic point of view, I think these models can be grouped in 4 groups of increasing complexity. Each model consists of three layers: the outcome (y), the hidden state (h) and the features (x). The outcome and features are functions of what is happening inside the hidden state. These functions are linear functions. See Figure 6. Model A is the single genes model. In this model we assume that the outcome is a simple linear function of a subset of a few genes. Model B is the model which is implied in Chuang's, Lee's an Park analysis. These analyses assume that the hidden state is actually a collection of pathways or complexes. The outcome is a function of the state of

these pathways and the features are correlated to these pathways. A difference between the models is that Park only allows a feature to belong to at most one pathway. Chuang's and Lee's algorithm assume the same underlying model, but have slightly different approaches to find the pathways. Model C tries to add logic effects to previous models. So far, the previous models assumed the function to be linear. By adding extra layers within the hidden layer, we could for example express the outcome variable as a logic expresssion y = (g1 OR g2) AND g3, which would be impossible to do using model B. Model D incorporates interactions between hidden nodes which influence the features. So far, the previous models assumes the feature variables to be independent. With model D, I try to model the dependendancy between the feature values.

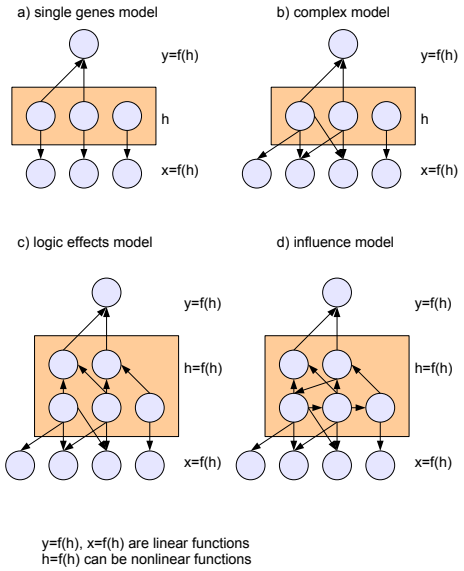


Figure 6: Four basic models in increasing complexity.

Some hypotheses concerning implementations:

- Improved classification can either occur due to grouping genes which estimate pathway values as in model B or by grouping genes to estimate higher level hidden nodes as in model C, or by grouping unwanted genes.
- Subnetworks in the IPP network and C2 pathways give an increased chance of finding pathways which correspond to model B pathways. C2 pathways should work better than IPP subnetworks.
- Chuang and Lee find a combination of improvements due to model B and model C.
- Park's method can be improved by preremoving genes which we know don't interact in complexes, or putting a constraint on which genes may be clustered together.
- Trying to find pathways which only correspond to model B and not model C will improve the results, by putting constraints on correlation for example.
- Trying to estimate higher level nodes in model C using other methods instead of taking means will improve results.
- Using more specific interactions helps finding model B.
- Subtyping improves the function connecting y = f(h).

1 Changelog

April 6th

1.1 Single genes analysis

First of all, let's see what the effect is changing the number of features used in feature selection. In the setup in Figure 1, training sets were used to rank genes by T-score and take the top N genes. A classifier (NMC) was trained using these genes on the training set. This classifier was tested on the training dataset. Note that no cross-validation is used in these methods, so no standard error bars can be generated.

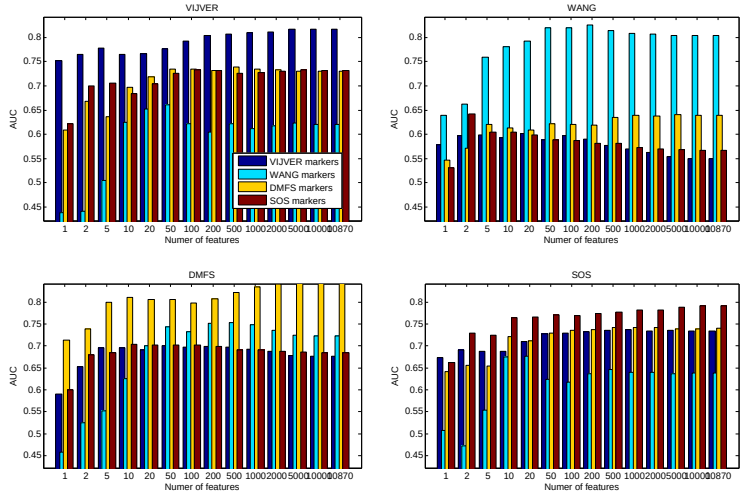


Figure 1: Classification performances using various numbers in feature selection. Each subplot depicts a validation dataset and the bars the various training datasets. Training sets where used for feature selection and training.

The same setup, but this time using Fisher classifier is found in Figure 2. A version using logistic regression is in Figure 3. In logistic regression, the number of features was limited to 1000, but it seems the results are very similar to using a Fisher classifier.

Some conclusions:

- Fisher classifier may be inspected for analysis instead of the much slower logistic regression.
- Using the NMC classifier works better than using a more variant classifier, such as Fisher and logistic regression. When using Fisher, none of the unbiased perfomances climb above 0.7.
- Fisher classifier performs worse when many features are added. NMC seems to be almost immune to this effect.

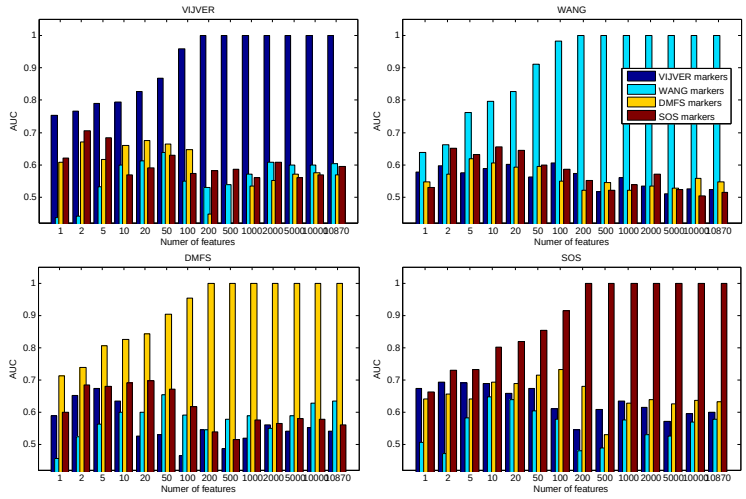


Figure 2: Classification performances using Fisher.

1.2 Simulation

Why bother simulating? In addition to reasons in the previous journal, it gives possible options for answering the questions, why does A work better than B?

Here is an example run:

```
Model a
H_i = Normal(0,3)
X_i = Normal(H_i,3)
Y = Normal(Beta*H, 3)
Beta = Normal(2,3)
250 Runs
100 training samples, 100 testing samples
200 features (genes)
```

In the example run, we see an improvement in classification performance can be gaines using model A, so grouping genes may help improve classification even when there is no coregulation.

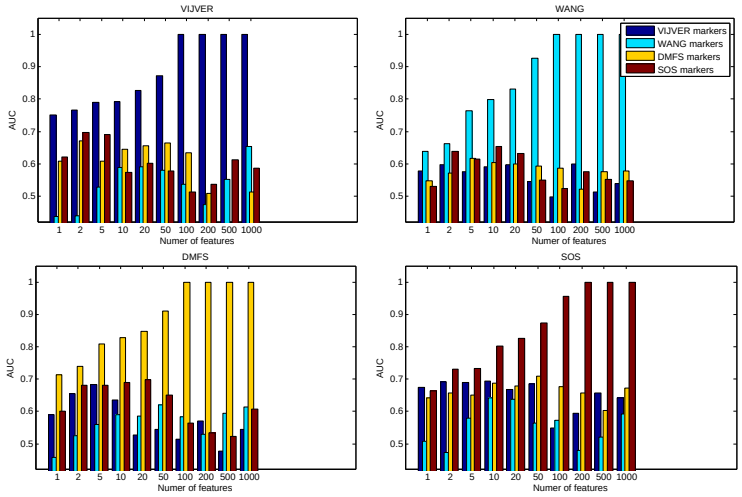


Figure 3: Classification performances using Logistic Regression. Number of features is limited to 1000 since PRtools has troubles with high number of features.

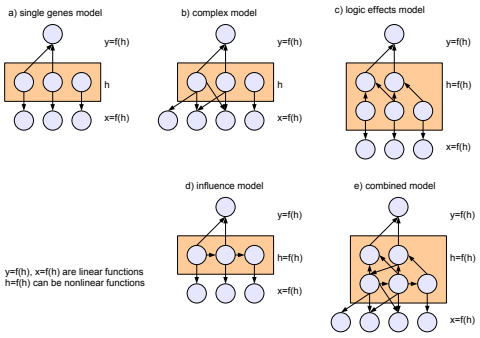


Figure 4: 5 models for simulation in increasing complexity.

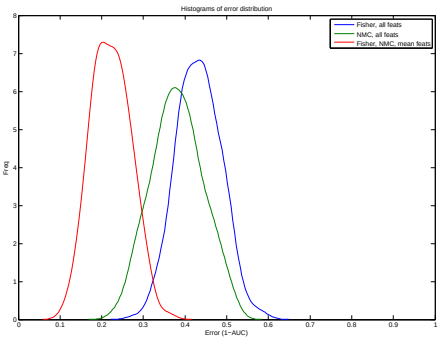


Figure 5: Result of example run.

1 Models

April 13th

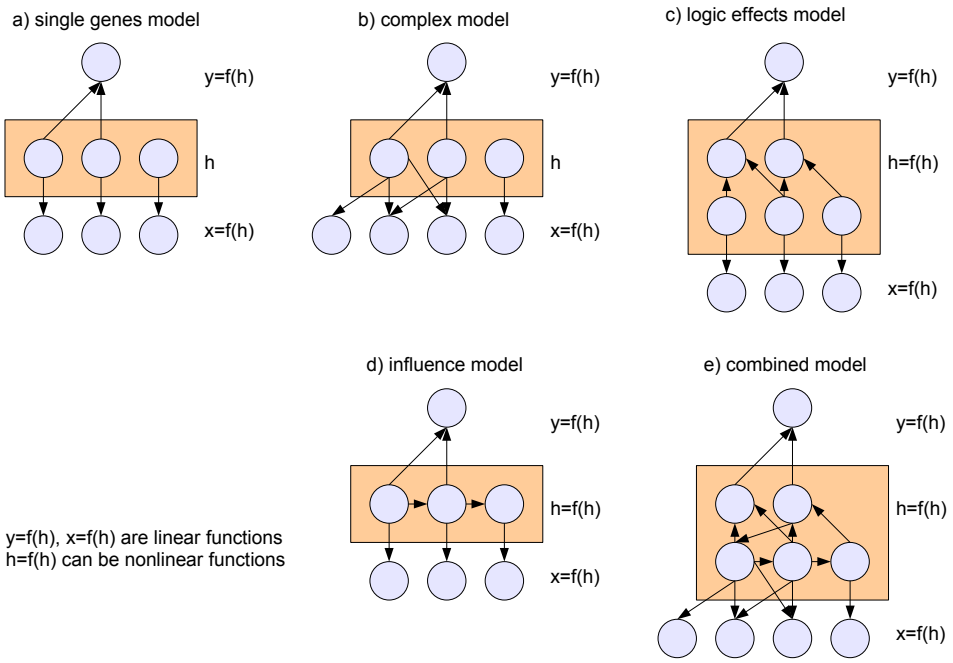


Figure 1: 5 models for simulation in increasing complexity.

1.1 Model A: single genes

In the single genes model, there is no correlation between the measured features. This model might be considered as a sort of null model, wherein there is no knowledge to be exploited, and Chuang's, Lee's and Park's algorithm should have no advantage. This model is meant to explore the basic properties of the various classifiers and taking the average of features.

Note that in this model, the flow of information is from hidden nodes to the outcome label. Also, in these simulations, I haven't put effort in making the training or testing samples stratified, but I expect there to be a 50/50 split in good and poor outcome.

1.1.1 Example run 1

```
100 runs
20 training samples, 80 testing samples
10 features(=n)

H_i = Normal(mean=0, var=5), for i=1..n
X_i = H_i+Normal(mean=0, var=2), for i=1..n
Y = BETA*H + Normal(mean=0, var=2)
BETA = {1,1,1,..1}
```

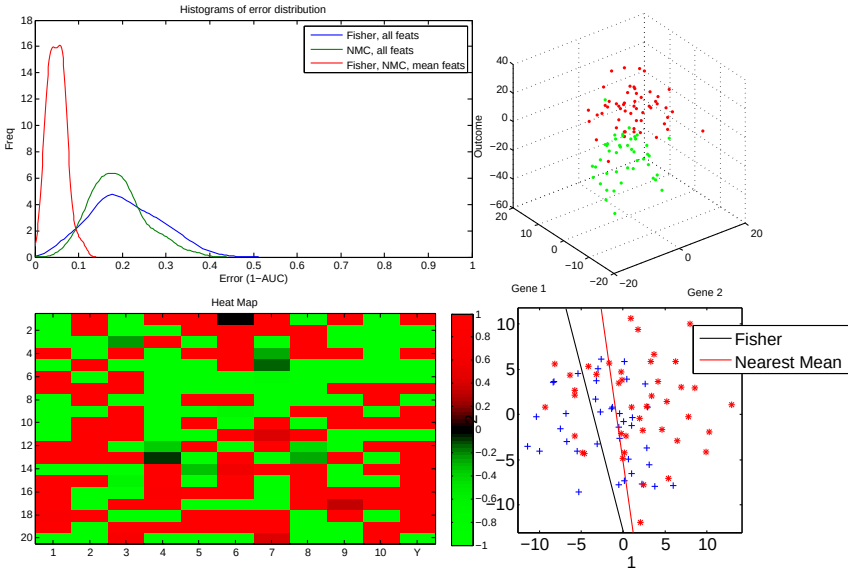


Figure 2: Model A, example. Shown here are a distribution of the errors. The performance of the NMC classifier using all features overlaps with NMC classifier using averaged features. Also, the NMC classifier and fisher classifier using averaged features have the same error distribution when calculating the AUC. Also shown upperright is an example distribution of two genes and their outcome of one run. The outcome labels are colored red and green. The lowerleft graph shows a heatmap of one example run. The heatmap also contains an outcome column. The lowerright graph shows the distribution of two genes for the testing samples, along with the classifier line calculated using the training samples.

1.1.2 Conclusions

In this single genes model, there is a trivial situation wherein improvement in classification by averaging of the features can be achieved if the betas are similar. Averaging the genes has the same effect as training a regression model with all betahats being the same, thus putting the regression line on the 1-vector. Since we're calculating the AUC, further training of this on-dimensional mapping is useless.

1.2 Model B: complex model

In the complex models, features genes are a function of genes in the hidden layer, this way, the features belonging to the same hidden node are forced to correlate. The expected behavior here is of course that classification will improve if the correct genes are averaged. In the following examples, the features belong to at most 1 complex.

1.2.1 Example run

```
100 runs
20 training samples, 80 testing samples
10 features(=n)

H = Normal(mean=0, var=2)
BETA = {1,1,1,..1}
X_i = BETA*H_i+Normal(mean=0, var=2), for i=1..n
Y = H + Normal(mean=0, var=2)
```

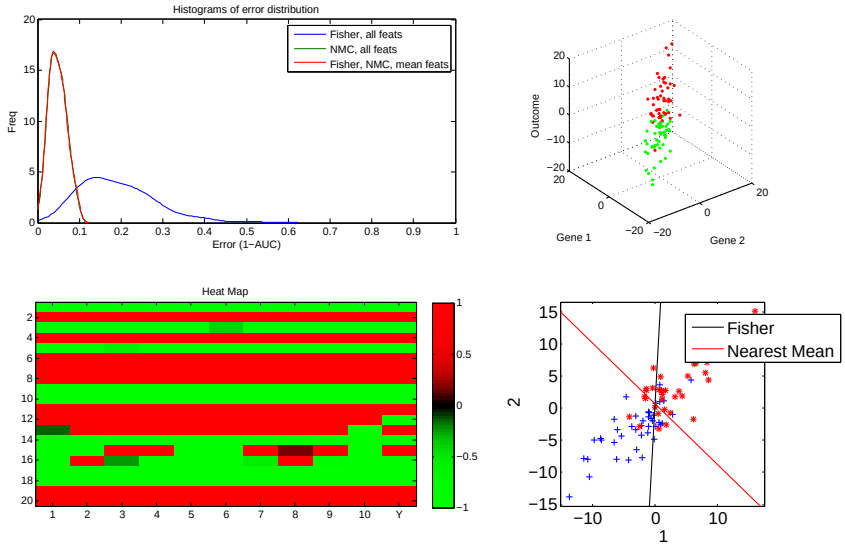


Figure 3: Model B, example

In this example, averaging the gene expression seems to perform just as good as using NMC as a classifier.

1.2.2 Example run 2

```
100 runs
2000 training samples, 80 testing samples
10 features(=n)

H = Normal(mean=0, var=2)
BETA = {1,2,1,2,1,2...,2}
X_i = BETA*H_i+Normal(mean=0, var=2), for i=1..n
Y = H + Normal(mean=0, var=2)
```

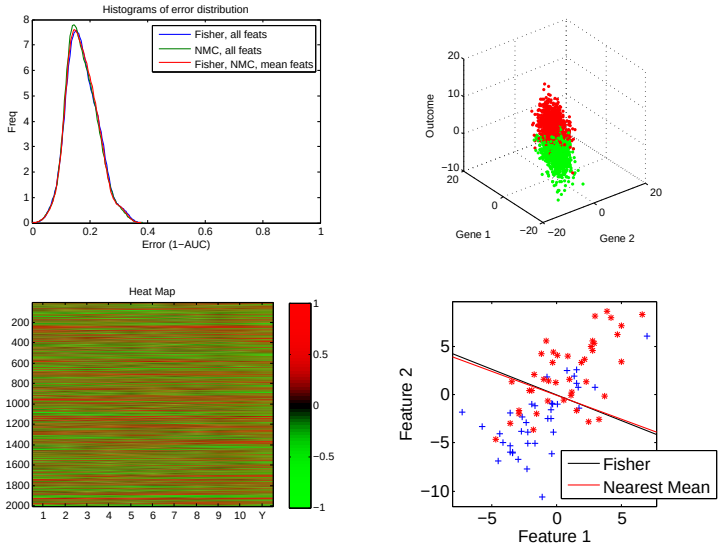


Figure 4: Model B, example 2

In this model, I've added 2000 training samples, and I've altered betas to not lie on the 1-vector, but have different values. I expected the fisher classifier to perform better, but this is not the case.

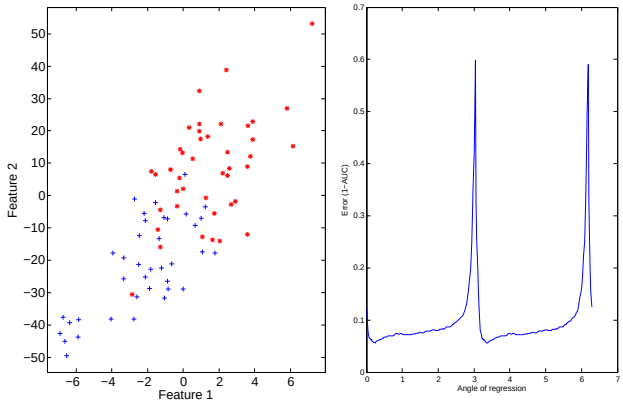


Figure 5: Observation. I've generated a two gene model dataset which are obviously correlated. Averaging genes would put the error of the AUC at angle=$\pi/4$. When training a fisher classifier, the error varies along this graph.

1.2.3 Conclusions

It seems that, when variables are correlated, averaging genes gives the best classifier possible. Intuitively, I would say that given enough samples, a fisher classifier should perform better, but I wasn't able to reproduce this situation leading me to think that if are variables are correlated, any regression line near the 1-vector is optimum. This observation is further examined in Figure 5. Also, from the fact that NMC classifier performs almost the same as when averaging the genes, I'd say most of the improvement in performance is gained by using a NMC.

1.3 Model C: logic effects

1.3.1 Example run

```
100 runs
20 training samples, 80 testing samples
10 features(=n)

H_i = Normal(mean=0, var=2), for i=1..n
X_i = H_i+Normal(mean=0, var=2), for i=1..n
Y = min(H_i) + Normal(mean=0, var=2)
```

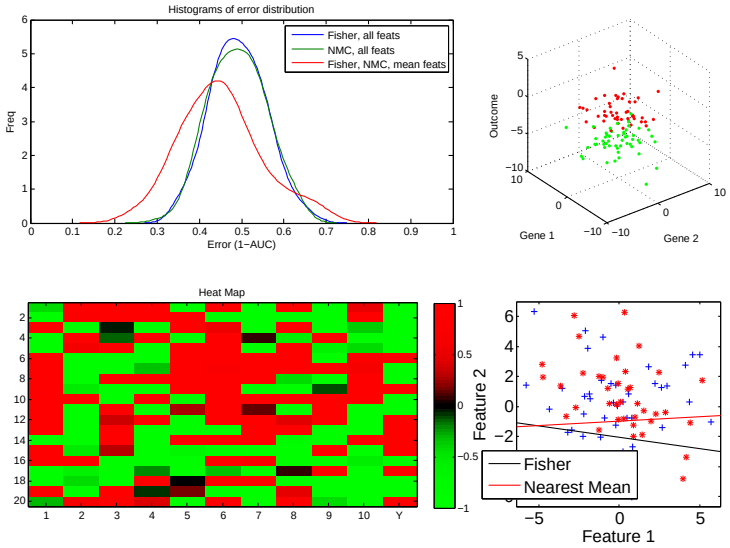


Figure 6: Model C, example 1

1.3.2 Conclusion

When there is a logic effect, averaging genes might also help, but I think the improvement in performance lies is due to the same effect as in the previous sections.

1.4 Further to do

Generate simulation which contain all types of combinations of effects.

We want to find model B type groups of genes. Test how well all combinations of search criteria (MI score, T score, correlations) and search methods (sequential selection, forward selection, clustering) are able to find the correct groups of genes in these simulation.

1 Analyzing Park's efficiency

April 24th

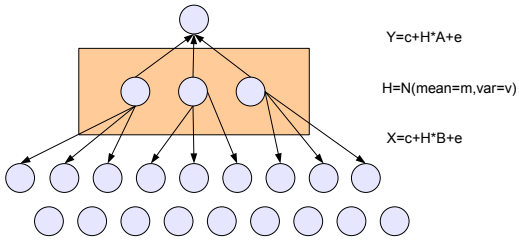


Figure 1: Initial experimental setup.

```
H = normal(mean=0, var=1).
X = H_x + normal(mean=0, var=1).
X_noise = normal(mean=0, var=9).
Y = -H_1 + H_2 + 2 * H_3 + normal(mean=0, var=9).
Outcome = 1 if Y above 0, 0 if Y under 0.
200 training samples
200 testing samples
```

The 200 training and 200 testing samples where generated 200 times from this simulation data to generate Figure 2. Also, the 200 training samples where used to create a clustering, and to train the genes and supergenes in these clustering. The 200 testing samples where used to select the best cutoff level and number of features.

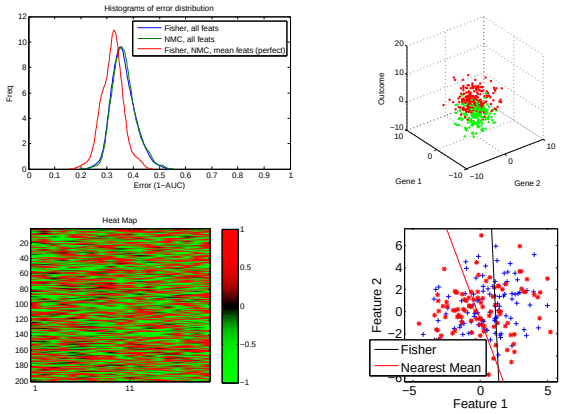


Figure 2: Starting experimental setup results.

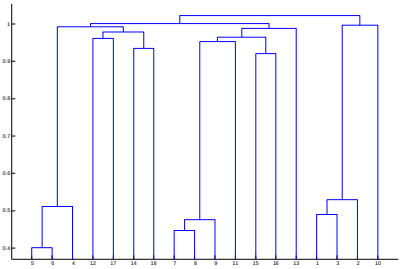


Figure 3: Example dendrogram.

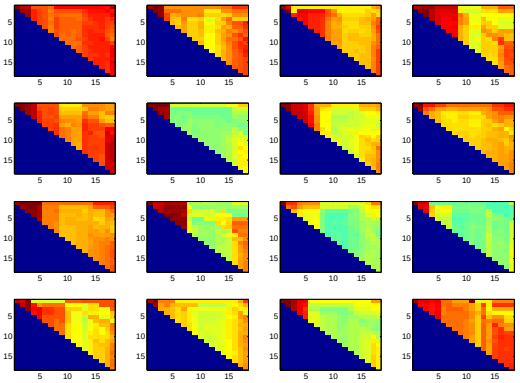


Figure 4: Example learning curves for 16 runs.

Cutoff	Features	Clusters
18	9	[8] [9] [7] [4] [6] [1] [3] [2] [5]
11	4	[7 8 9] [4 5 6] [10] [1 2 3]
11	2	[7 8 9] [1 2 3]
10	2	[7 8 9] [1 2 3]
9	1	[7 8 9]
15	3	[7 8] [1 3] [9]
10	4	[7 8 9] [1 2 3] [10] [4 5 6]
17	7	[9] [8] [7] [3] [4] [2] [5 6]
15	5	[7] [8 9] [4 6] [5] [1 3]
9	5	[7 8 9] [4 5 6] [14] [15] [1 2 3]
11	6	[7 8 9] [4 5 6] [1 2 3] [12] [16] [10]
17	4	[7 9] [8] [2] [1]
13	6	[4 5] [7 8 9] [6] [17] [1 2 3] [14]
10	2	[7 8 9] [1 2 3]
8	3	[7 8 9] [1 2 3] [4 5 6]
12	1	[7 8 9]

Something that's directly apparent from these data is that the cluster [7 8 9] scores better than the other clusters. This is because alfa parameter, which is 2, instead of 1 and -1 for the other clusters. This makes this cluster more important in the prediction of y, so it will get a higher t score.

Also, only 1 out of the 16 runs contains the clustering we would like to see. The clustering seems to work good, but the cutoff selection not. All in all, we need a few scores to measure how well the searching algorithms find the correct clusters.

1.1 Cross validated Park

Using a cross-validated version of Park to find the optimum cutoff level and number of features improves the results since it consists of more robust. To demonstrate, I've combined the 200 training and 200 testing samples in one set and cross validated it in 5 folds for 2 repeats.

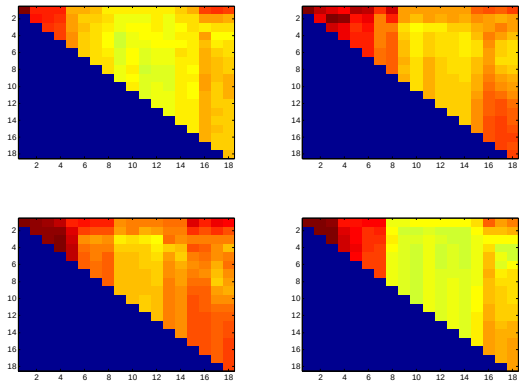


Figure 5: Learning curves for cross-validated Park.

Cutoff	Features	Clusters
11	8	[7 8 9] [1 2 3] [4 5 6] [12] [17] [15] [10]
10	3	[7 8 9] [1 2 3] [4 5 6]
12	3	[7 8 9] [4 5 6] [1 2 3]
14	3	[7 8 9] [1 2 3] [4 5 6]

1.2 Parameters analysis

The basic architecture of the experimental setup is in Figure 1.

Once again, here is the basic setup:

```
H = normal(mean=0, var=1).
X = H_x + normal(mean=0, var=1).
X_noise = normal(mean=0, var=9).
Y = -H_1 + H_2 + 2 * H_3 + normal(mean=0, var=9).
Outcome = 1 if Y above 0, 0 if Y under 0.
200 training samples
200 testing samples
```

First of all, I'd like to know how well Park's method behaves given a different number of noise genes.

The 200 training samples where used to find the best clustering. Park's method was used to find the best cutoff level. All features at this cutoff level where returned. The 200 testing samples where used using a cross validation method to get an average AUC. The DLCV AUC was calculated for all single features and for the features reducted according to the optimal cutoff level.

Park's procedure and the DLCV procedure are the same code I've used before, except for fewer runs.

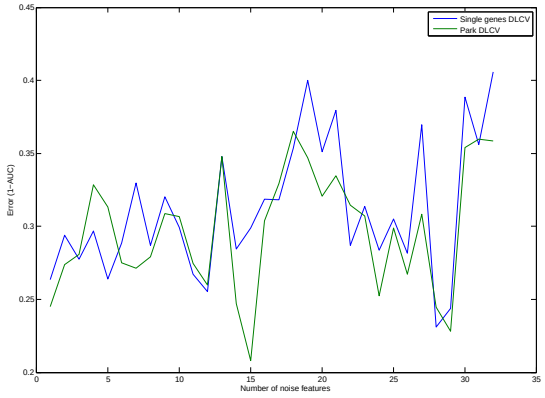


Figure 6: AUCs behavior given a different number of features.

From Figure 6 we see that Park's method performs better than single genes most of the time, but the difference is small.

For the next experiment, let's fix the number of noise genes at 9 and let's vary the noise caused by Xnoise with a variance from 0 to 16.

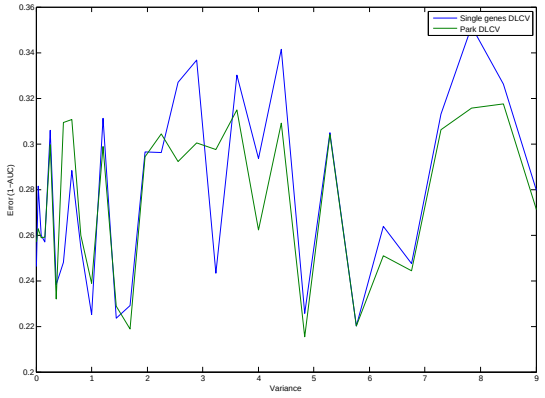


Figure 7: AUCs behavior given a different variance.

A less strong trend can be detected here. Then again, Park's algorithm was run only 30 times to create the graph above. Park still works slightly better.

1 Method comparison using simulations

May 9th

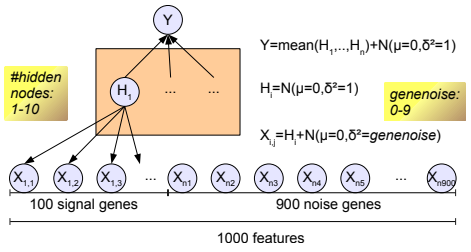


Figure 1: Simulation setup.

The simulation setup can be seen in Figure 1.

1.1 Assumptions for the simulation

One one hand, I want the simulation to be similar to the actual data, but on the other hand, it should be fast enough to work with and allow unknown parameters to be varied.

Here follows a list of parameter choices for the simulation:

- The number of testing and training samples are each 300, since this is the approximate number of samples in each of the DMFS, SOS, VIJVER and WANG dataset.
- The number of features is 1000. This is less than 10 percent of the features available in the actual datasets, but adding more features slows down the simulations.
- The number of informational features is 100. This is 10 percent of the features, and approximation of the actual number of informational features.
- For simplicity, the hidden nodes and noise genes have a mean of 0 and variance of 1. The signal gene has an error noise added and is normalized to have a mean of 0 and variance of 1. This way, all features have a mean of 0 and variance of 1, as in the actual data.

Apart from these assumptions, I want the model to be flexible enough to simulate a range of the following parameters.

- The number of hidden nodes can vary from 1 to 10. The number of signal nodes per hidden node then automatically varies from 100 to 10.
- The noise added to the signal nodes. This variance may vary from 0 to 9.

1.2 A multivariate notation

Since the entire system consists of normal distributions and linear functions, we can represent the system above using a mean vector and covariance vector.

The mean vector consists of zeroes. In the follow covariance matrix, only a few variables of the entire system is indicated. Some of the fractions here are caused by the normalization step. e_x denotes the variance of the signal error, e_y the variance of the outcome error, n the number of hidden nodes, X_e a noise gene.

	H_1	H_2	$X_{1,1}$	$X_{1,2}$	$X_{2,1}$	X_e	Y
H_1	1	0	$\frac{1}{1+e_x}$	$\frac{1}{1+e_x}$	0	0	$\frac{1}{n}$
H_2	0	1	0	0	$\frac{1}{1+e_x}$	0	$\frac{1}{n}$
$X_{1,1}$	$\frac{1}{1+e_x}$	0	1	$\frac{1}{1+e_x}$	0	0	$\frac{1}{n(1+e_x)}$
$X_{1,2}$	$\frac{1}{1+e_x}$	0	$\frac{1}{1+e_x}$	1	0	0	$\frac{1}{n(1+e_x)}$
$X_{2,1}$	0	$\frac{1}{1+e_x}$	0	0	1	0	$\frac{1}{n(1+e_x)}$
X_e	0	0	0	0	0	1	0
Y	$\frac{1}{n}$	$\frac{1}{n}$	$\frac{1}{n(1+e_x)}$	$\frac{1}{n(1+e_x)}$	$\frac{1}{n(1+e_x)}$	0	$\frac{1}{n}+e_y$

1.3 Methods used

Let's call the above setup a dataset generator. See Figure 2.

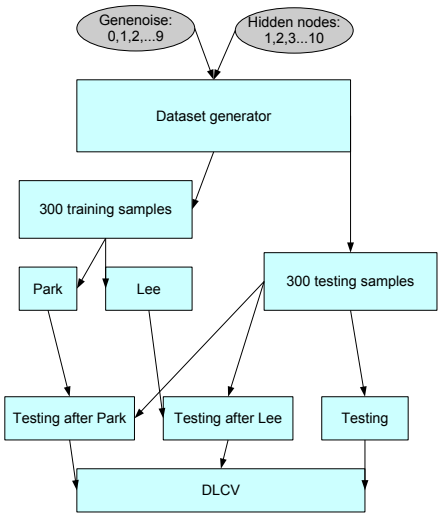


Figure 2: Workflow. Each combination of input variables (genenoise and hidden noides) is fed to the dataset generator to obtain a 'performance' landscape. This entire process is repeated ten times to get an average performance landscape.

1.3.1 Park

A dendrogram was computed of the simulation data and 200 cutoffs of this dendrogram were considered. For each

cutoff/number of features, an approximation of the AUC was calculated by splitting the data in 5 folds just as in the DLCV manner. The final AUCs are an average of 25 AUCs.

An example AUC 'landscape' can be seen in Figure 3.

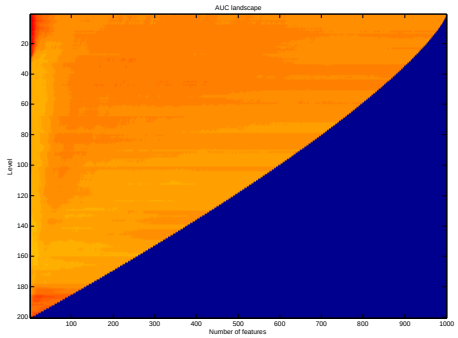


Figure 3: Example AUC landscape of Park. The simulation was run with 5 hidden nodes and a signal error of 1. Darker red indicates a lower error. According to this landscape, the optimum cutoff is 1, meaning that no features are averaged according to this this example.

Even though Park's algorithm determines the optimum number of features, I only use the cutoff level, not the optimum number of features.

1.3.2 Lee

To test Lee's algorithm, the training data was fed to Lee and a perfect set of subnetworks was given. So if we split the training data in 10 parts, the subnetworks given to Lee's algorithm would be those 10 dataset consisting of the corresponding features, without incorrect or missing features. So theorically, this algorithm should be able to return the best feature signature possible.

1.3.3 DLCV

My DLCV code was used, using NMC as the classifier and AUC as performance measure. The outer loop loop was split in 5 folds and repeated 50 times (returning 50 AUC values, not 250). The inner loop was split in 4 folds and was repeated 10 times (returning 10 learning curves, not 40). In the following results, I will use error=1-AUC as a measure.

1.4 Results

Figure 4 shows the average of 10 AUC landscapes.

Figure 5 compares the methods to each other.

If you inspect to experimental setup, increasing the number of hidden nodes basically has the same effect as increasing the signal error and I think the results show this. If there is low signal error, single genes do slightly better. When there is a little higher signal error, Park performs

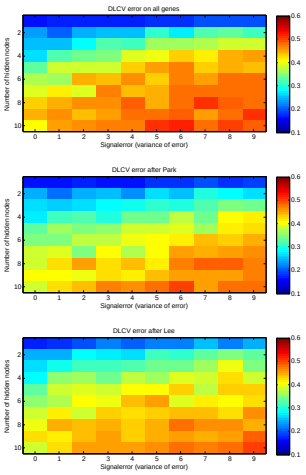


Figure 4: DLCV performances of the various methods. Blue indicates a low error, so a good performance.

better, but when there is too much error, Lee outperforms the rest, probably because of its inherent advantage of being given perfect geneset, rather than the algorithm. The conslusion is that Park works best in this simple setup, but if things get too noisy Lee might work better given perfect pathways.

2 Improved Park on real data

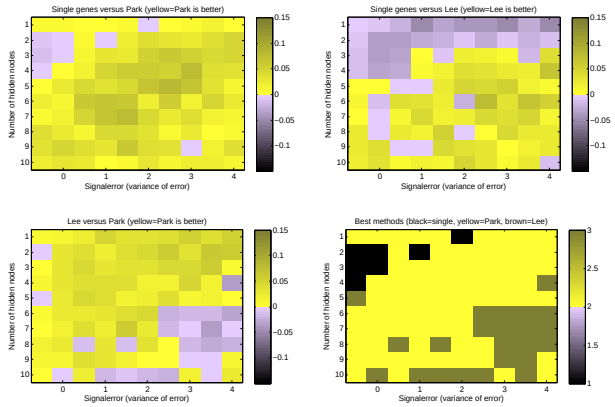


Figure 5: Comparisons. (Last-minute remark: signalerror varies from 0 to 9.

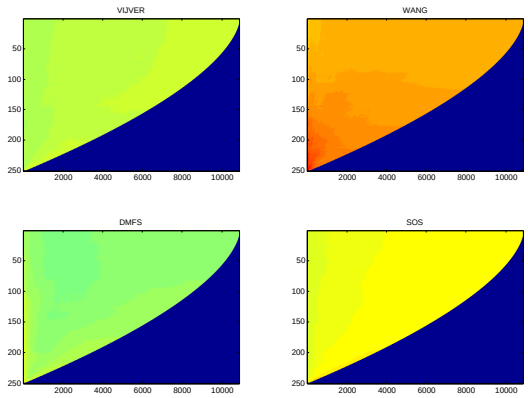


Figure 6: Park on real data. Optimum features/subnetworks. Vijver: 30/580. Wang: 27/7440. DMFS: 7/9206. SOS: 64/1767.

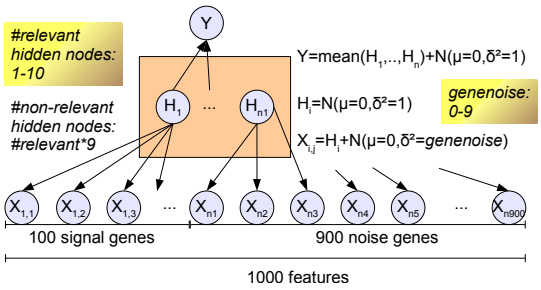


Figure 1: The model

1 Changelog

May 22th

Method comparison using simulations 2

1.1 Updated model

I've updated the model in order to have all features, signal and noise features, to come from a set of hidden nodes. In this model, if there is 1 signal hidden node, there are 9 noise hidden nodes. Each of these hidden noise 'generate' 100 features. If there are 10 signal hidden nodes, then there are 90 noise hidden nodes. Each of these 'generate' 10 features. This way, there are approximately the same number of features per hidden node.

The output, value Y, is a function of only the signal hidden nodes. A visual representation is given in Figure 1.

This model can be expressed as a combination of a mean vector and covariance matrix. This way, the relations between the different variables are clear. The mean vector in this case consists of zeroes, the covariance matrix:

	H_1	H_2	H_{noise}	...	$X_{1,1}$	$X_{1,2}$	$X_{2,1}$	X_{noise}	...	Y
H_1	1	0	0	...	$\frac{1}{\sqrt{1+e_x}}$	$\frac{1}{\sqrt{1+e_x}}$	0	0	...	$1/h_{relevant}$
H_2		1	0	...	0	0	$\frac{1}{\sqrt{1+e_x}}$	0	...	$1/h_{relevant}$
H_{noise}			1	...	0	0	0	$\frac{1}{\sqrt{1+e_x}}$	...	0
...	...	...	...	...	...	...	...	...	...	...
$X_{1,1}$				...	1	$\frac{1}{1+e_x}$	0	0	...	$\frac{1}{h_{relevant}\sqrt{1+e_x}}$
$X_{1,2}$				...		1	0	0	...	$\frac{1}{h_{relevant}\sqrt{1+e_x}}$
$X_{2,1}$				...			1	0	...	$\frac{1}{h_{relevant}\sqrt{1+e_x}}$
X_{noise}				...				1	...	0
...	...	...	...	...	...	...	...	...	...	...
Y				...					...	$\frac{1}{h_{relevant}} + e_y$

Here e_y indicates the variance of the error applied to Y.

1.2 Methods

There are a few methods which attempt to find subnetworks. The single features method, Park and Lee are already discussed in the previous changelog.

1.2.1 Perfect

The perfect method combines all features which belong to the same hidden node. Unlike the name, it is not 100 percent perfect, since it also returns subnetworks derived from the noise hidden nodes. So if the model would have 1 signal hidden node and 9 noise hidden nodes, the 'perfect' methods returns 10 subnetworks.

1.2.2 Raul2

Raul2 method is a Chuang-style searching method in a Lee-style searching space. That is, it uses forward feature selection, unlike the predetermined feature ranking in Lee's algorithm. Also, it uses MI score as a criterium.

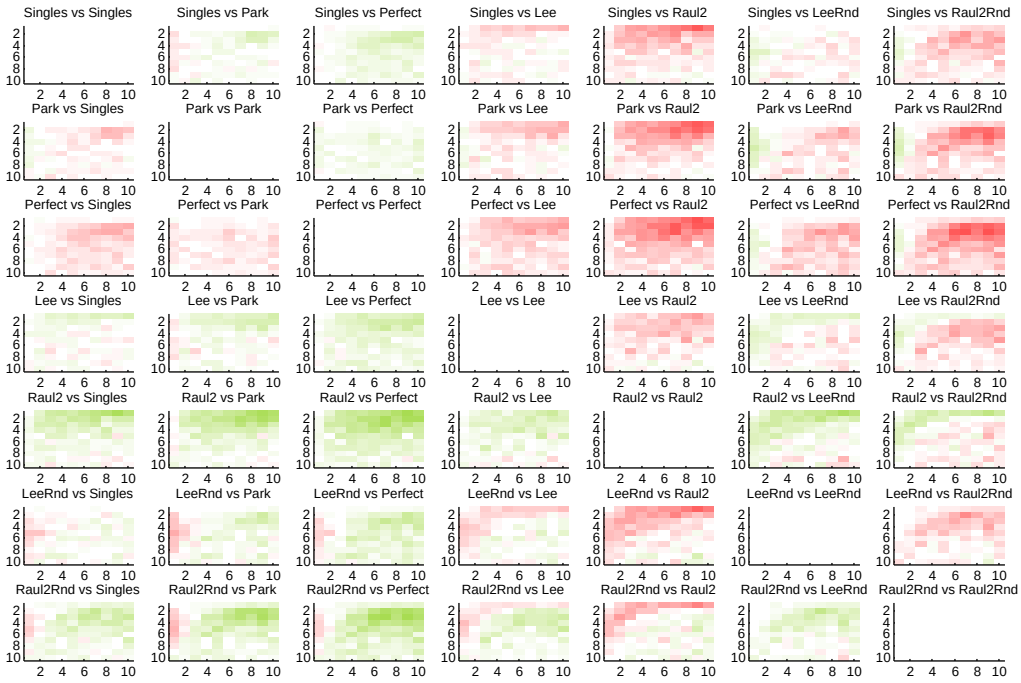


Figure 2: Comparions of the different methods. If X vs Y is indicated, green means Y performs better than X, red means Y performs worse than X.

1.2.3 LeeRandom

LeeRandom is Lee's method using randomnized subnetworks. Subnetworks were randomnized by randomly switching the features between the dierent subnetworks. So if Lee's method would attempt to find CORGs in 1 module full of signal features and 9 modules full of noise features, LeeRandom methods would search in in 10 modules with each a random number of signal features. As we will see later, this method actually outperforms standard Lee's method for this model.

1.2.4 Raul2Random

Similar to LeeRandom.

The best methods are shown in in Figure 3.

2 A Lee-biased model

Most attempts at developing a model which is best solved by Lee's algorithm instead of Park's algorithm failed. It seems that if a model is easily solved by Lee than it can be easily solved by Park.

However, a situation where Lee works better than Park is when we remove the correlation between the features and let the hidden variables be a function of the features instead of the other way around.

Instead of varying the signal error, this time I've varied the error on the outcome label. For the model, see Figure 4 and for the resultse see Figure 5.

This model may also be expressed as a covariance matrix:

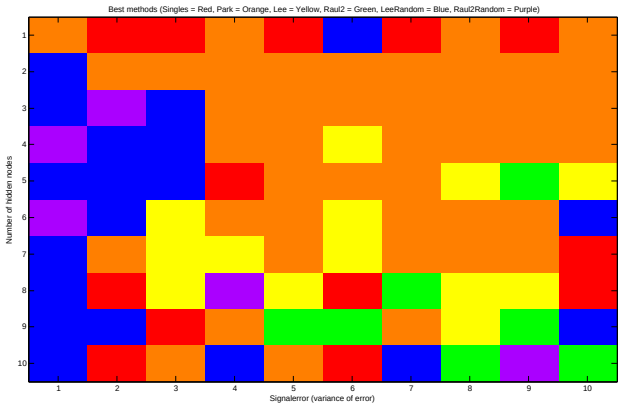


Figure 3: Singles = red, Park = orange, Lee = yellow, Raul2 = green, LeeRandom = blue, Raul2Random = purple

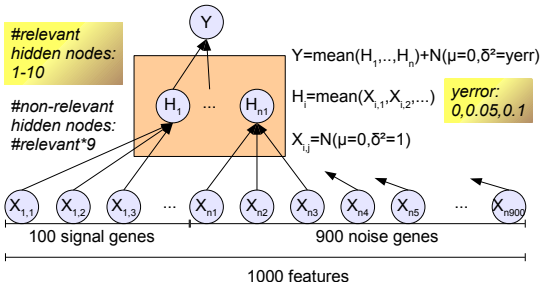


Figure 4: Lee-biased model

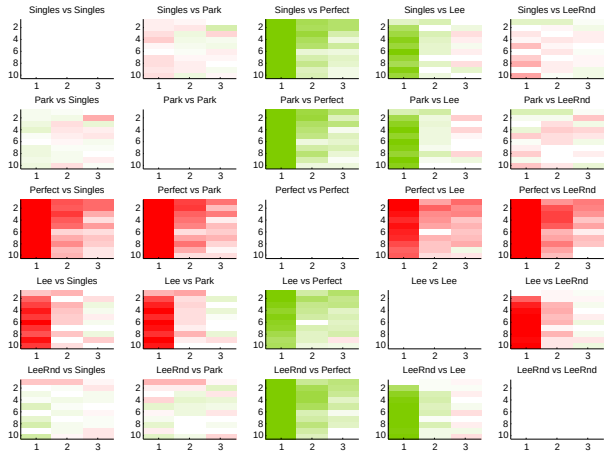


Figure 5: Comparisons of the methods under the Lee-biased model. If X vs Y, then green indicates that Y performs better than X. The vertical axis indicates the number of relevant hidden nodes, varying from 1 to 10. The horizontal axis indicated the error on the outcome. 1 : error $= 0$, 2 : error $= 0.05$, 3 : error $= 0.1$;

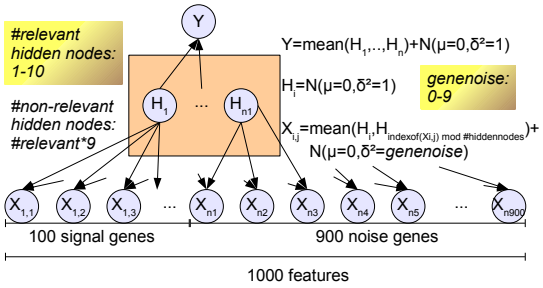
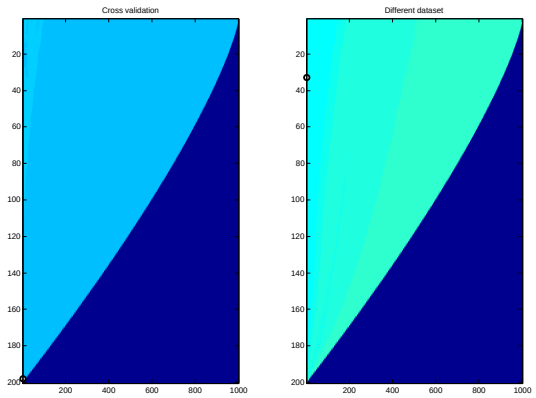


Figure 6: Another attempt at a Lee-biased model.

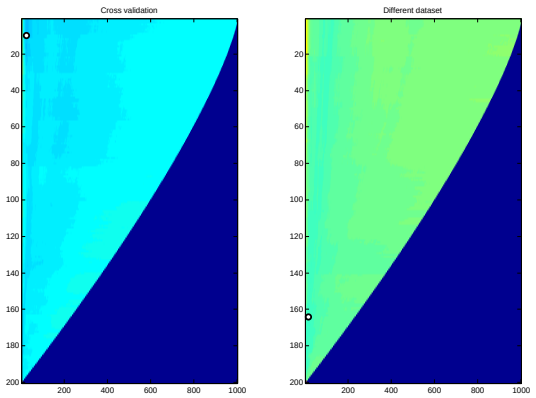
	H_1	H_2	H_{noise}	...	$X_{1,1}$	$X_{1,2}$	$X_{2,1}$	X_{noise}	...	Y
H_1	$\frac{1}{n_H}$	0	0	...	$\frac{1}{n_H}$	$\frac{1}{n_H}$	0	0	...	$\frac{1}{n_H h_{relevant}}$
H_2		$\frac{1}{n_H}$	0	...	0	0	$\frac{1}{n_H}$	0	...	$\frac{1}{n_H h_{relevant}}$
H_{noise}			$\frac{1}{n_H}$	...	0	0	0	$\frac{1}{n_H}$	...	0
...	...	...	...	...	...	...	...	...	...	...
$X_{1,1}$				...	1	0	0	0	...	$\frac{1}{n_H h_{relevant}}$
$X_{1,2}$				...		1	0	0	...	$\frac{1}{n_H h_{relevant}}$
$X_{2,1}$				...			1	0	...	$\frac{1}{n_H h_{relevant}}$
X_{noise}				...				1	...	0
...	...	...	...	...	...	...	...	...	...	...
Y				...					...	$\frac{1}{n_H h_{relevant}} + e_y$

3 Another attempt at a Lee-biased model

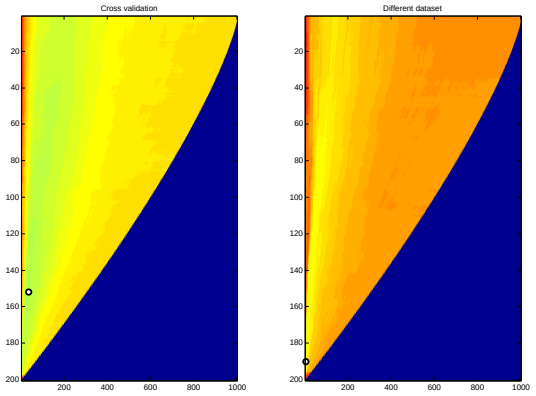
In order to come up with a model which is expected to be more powerful using Lee instead of Park (but isn't), I've tried to make a model which disrupts the dendrogram so that Park can't be effective. See Figure 6. This model is an extension of the first model. Now, more arrows have been added such that each feature node is generated by 2 (or sometimes 1) hidden nodes. This way, the correlation between informative genes is disrupted. However, this model doesn't prefer Lee, as I first expected.



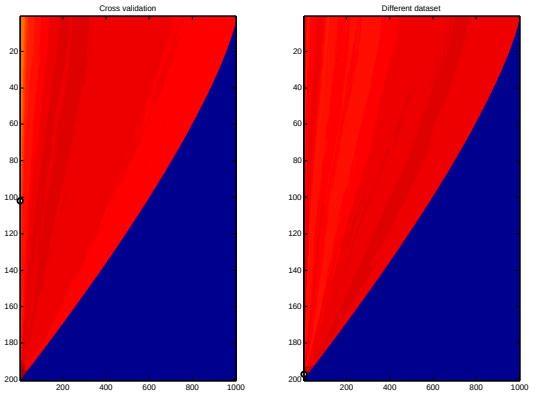
a)



b)



c)



d)

Figure 7: Park's AUC landscape. Shown on the left is the AUC landscape using the first model as in the previous figures, generate by Park's algorithm. On the right is the AUC landscape when a classifier is trained using the training data and testing using all features on a independent dataset. a) Number of hidden nodes = 1, signalerror = 0. b) Number of hidden nodes = 1, signalerror = 9. c) Number of hidden nodes = 3, signalerror = 1. d) Number of hidden nodes=7, signalerror = 0.

4 Some general conclusions regarding the simulations.

- In the first model, when there is a low signal error, then Lee's method using random subnetworks works better than other methods. This is so because every subnetwork Lee's method is likely to have some signal nodes.
- The perfect model works best in all cases, except in the first model, where it is outperformed in a few cases by LeeRandom.
- In the first model, Raul2 performs worse than Lee.
- In the first method, Park outperforms the single genes method when there is a high signal error and a low number of hidden subnetworks.
- In the second method, Lee outperforms the single genes method.

5 More on Park

I've analyzed how efficient Park is at finding the correct cutoff level and number of features. The problem with analyzing this efficiency is that a large range of cutoff levels and number of features could be the best theoretical value. To give an impression of where Park thinks the best cutoff level is and where the best cutoff level is when considering an independent dataset, look at Figure 7.

1 Changelog

July 7th

1.1 Paired T-score versus ranksum

I've tried to compare the paired t-score versus the paired ranksum test to see if it would make a difference. Apparently, the paired ranksum test gives higher P-values.

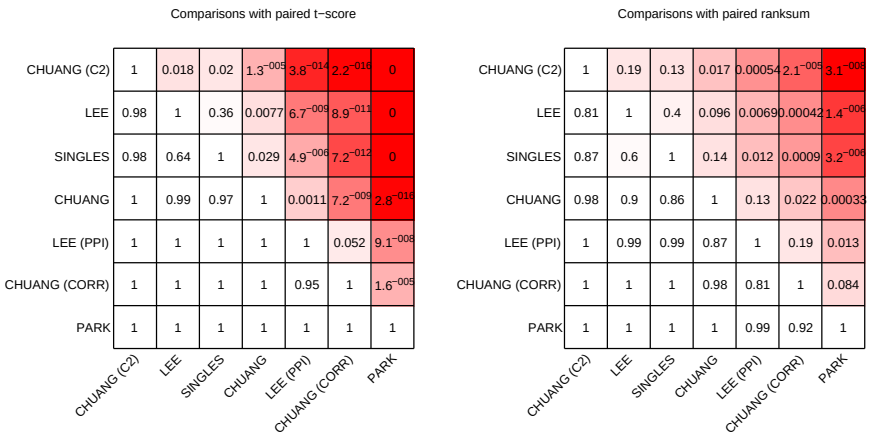


Figure 1: Paired t-score vs ranksum

1.2 Paired ranksum results

Note that to get these results I've recompiled the datasets to have a 11601-gene signature instead of a 10870-gene signature.

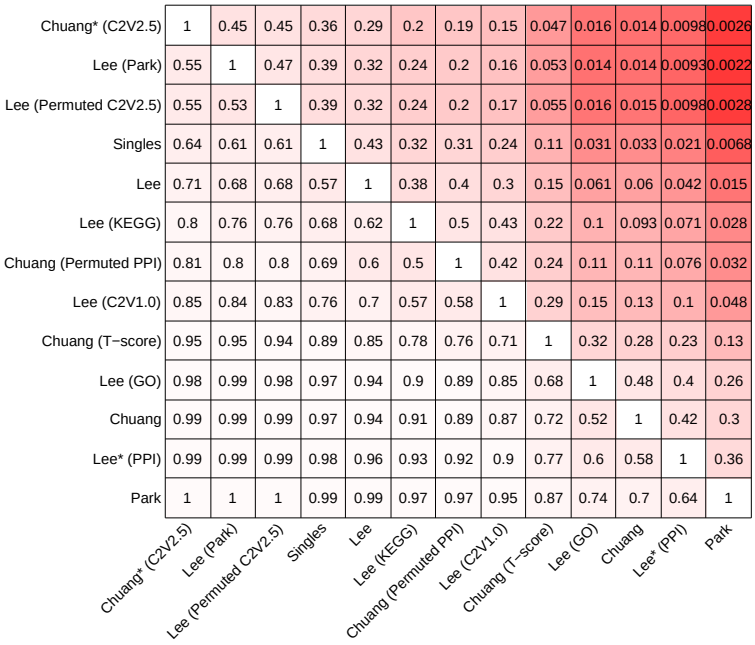


Figure 2: Cross-dataset paired ranksums

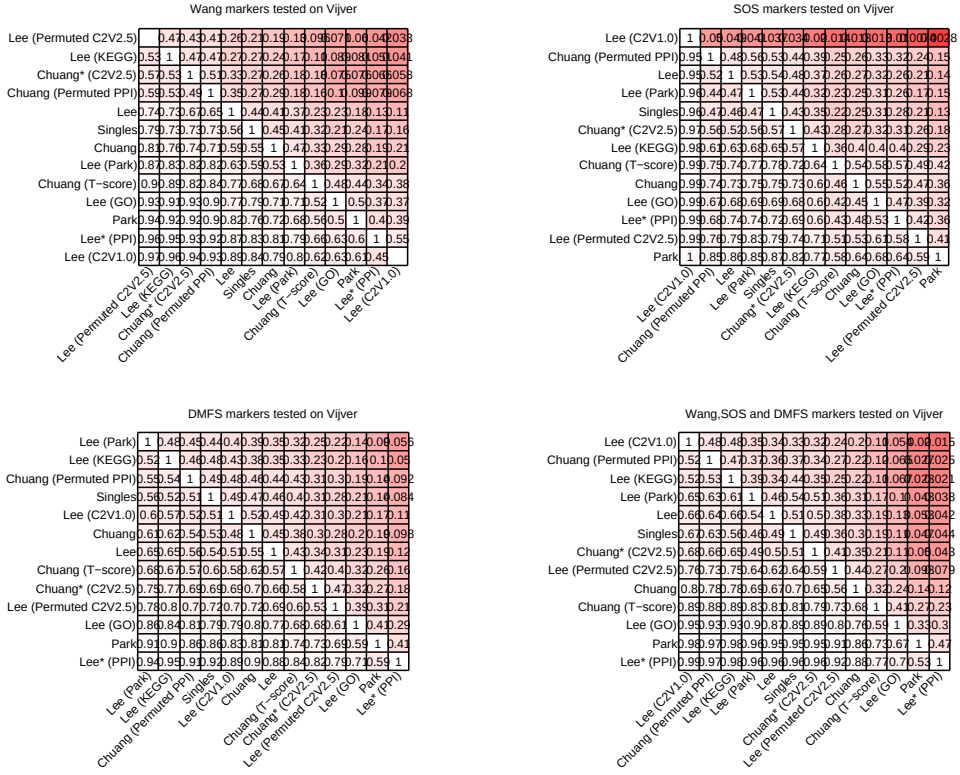


Figure 3: Markers tested on Vijver

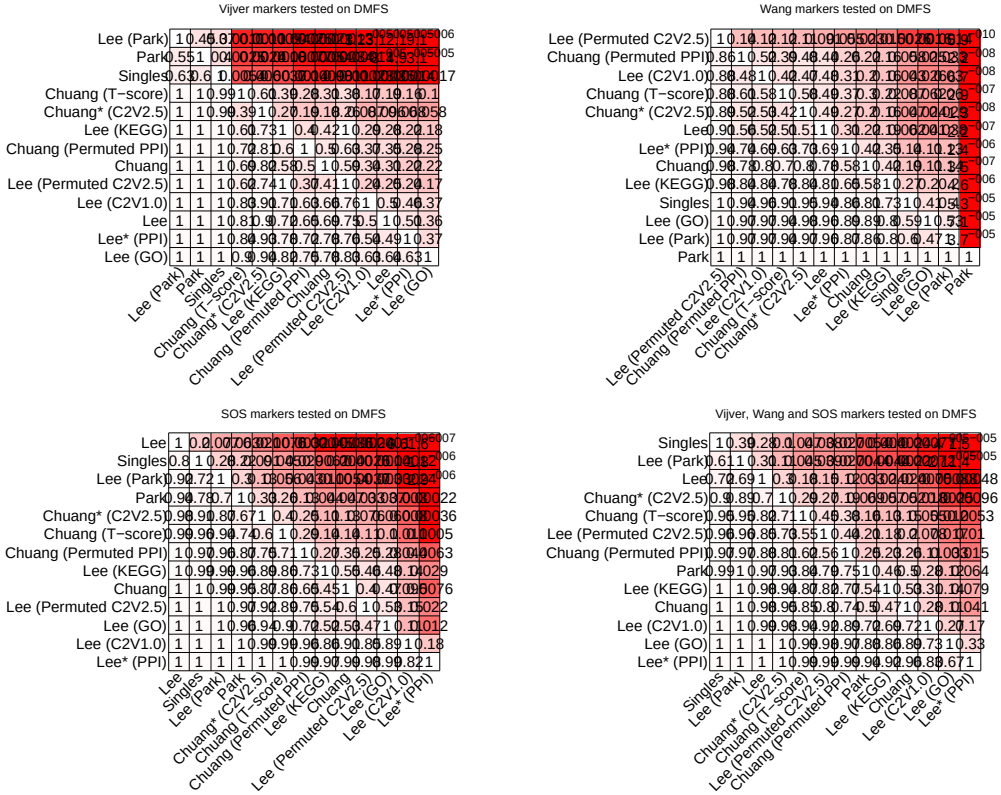


Figure 4: Markers tested on DMFS.

1 Changelog

July 14th

1.1 Figures

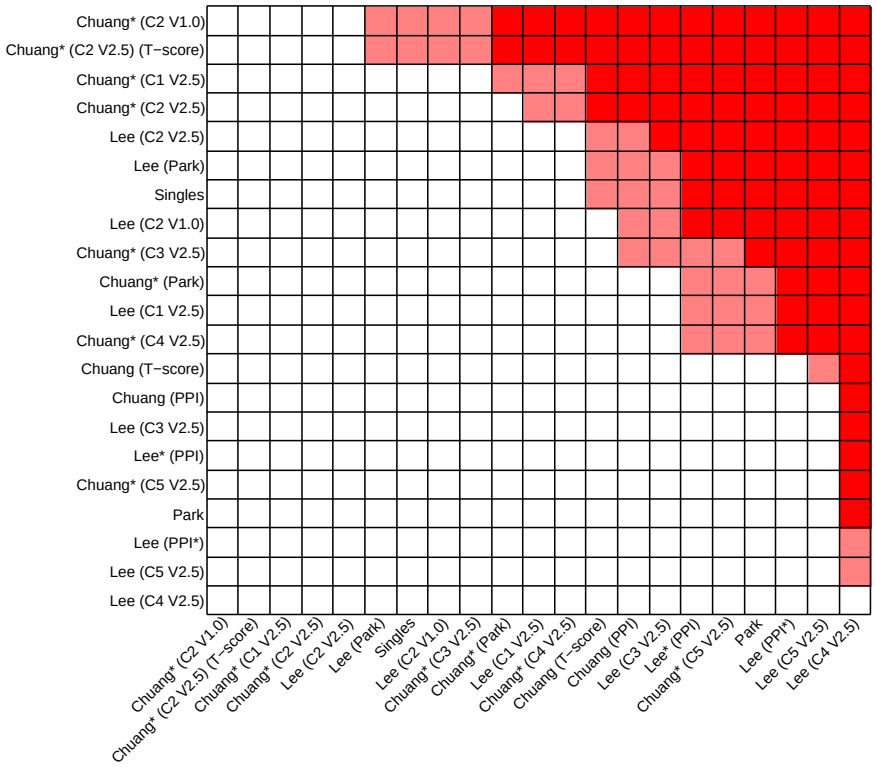


Figure 1: Comparisons. Dark red indicates significance p under 0.01. Light red indicates significance p under 0.05. In this version, Wang is included.

1.2 Conclusions

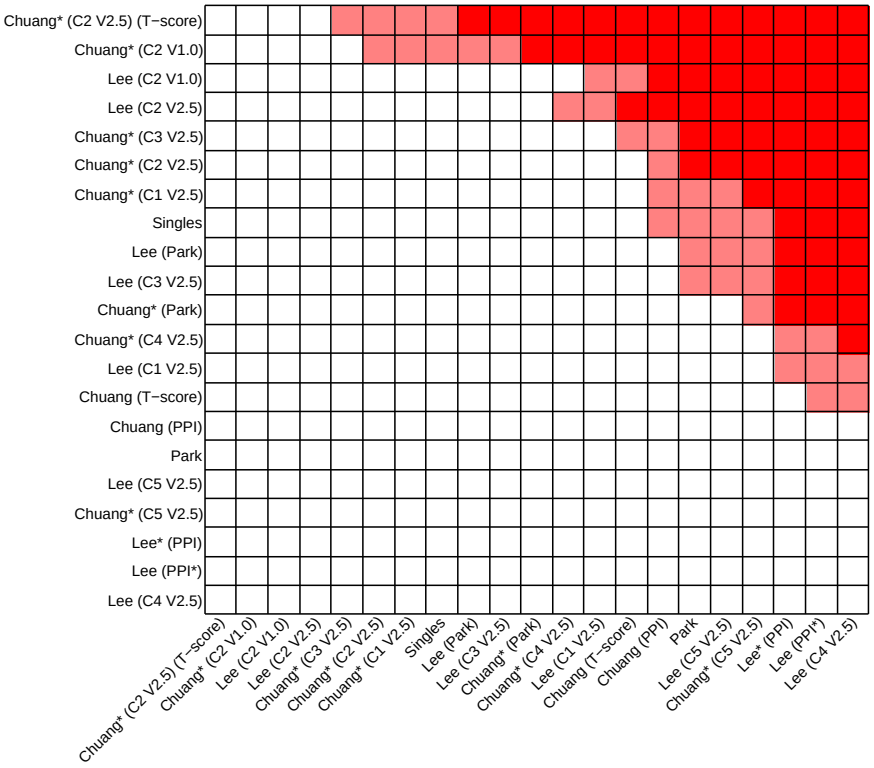


Figure 2: Comparisons. Wang is totally excluded.

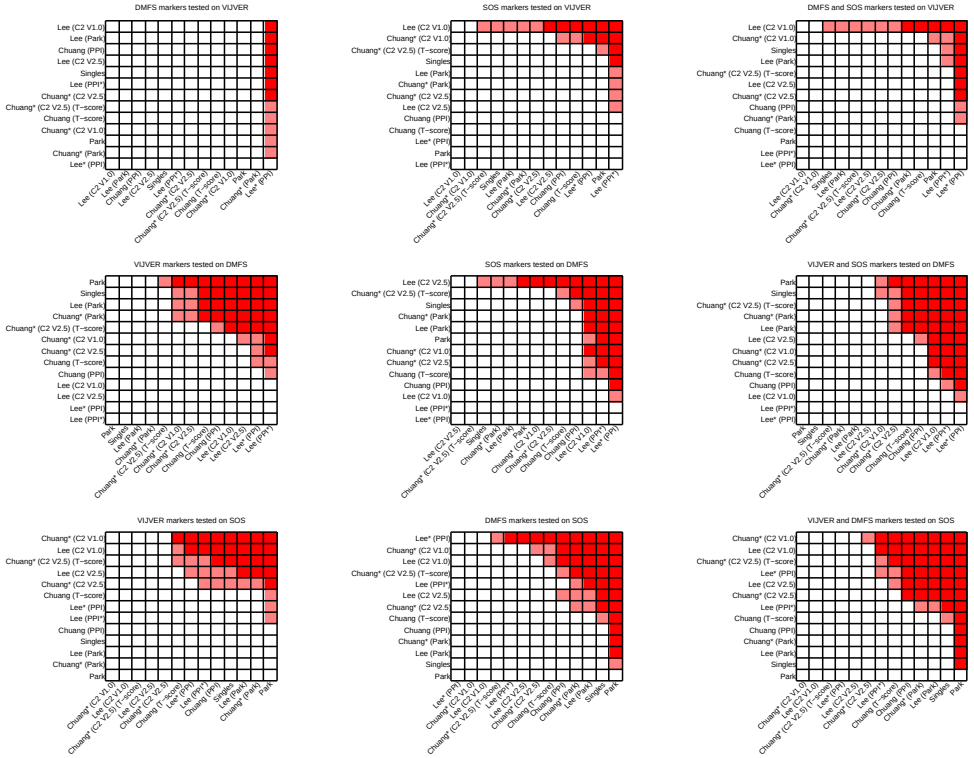


Figure 3: Comparisons split out. Wang is excluded.

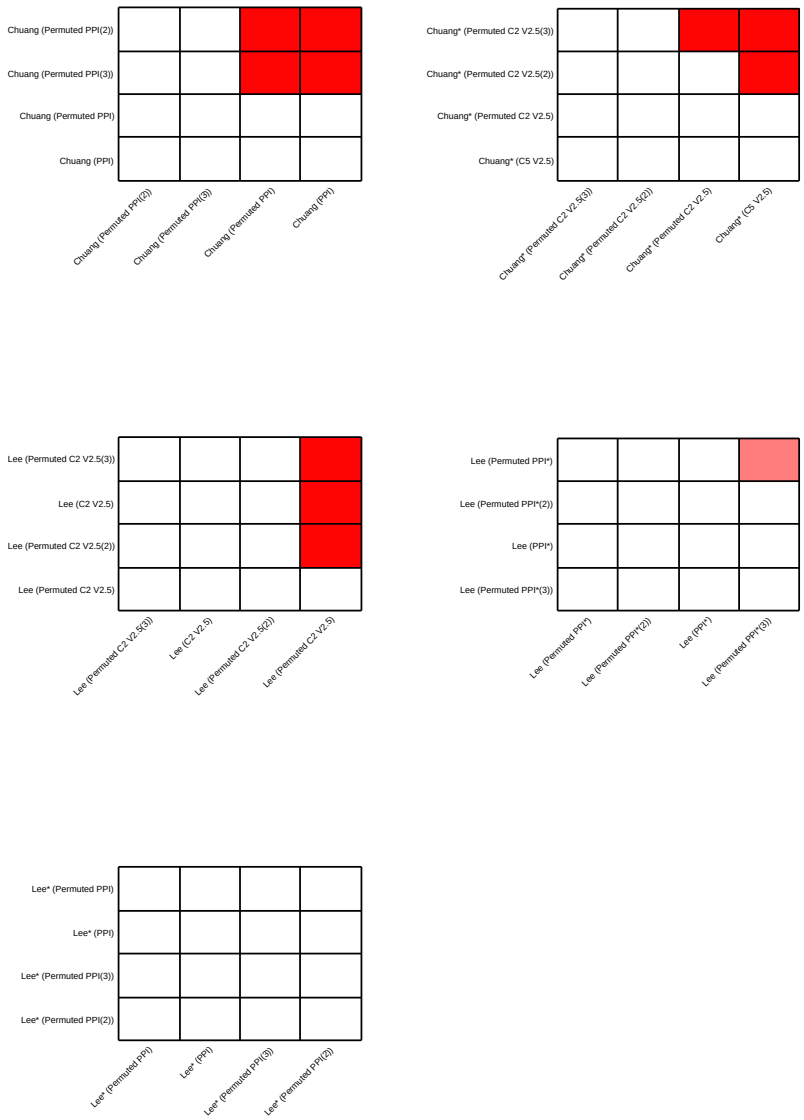


Figure 4: Comparisons using permutations.

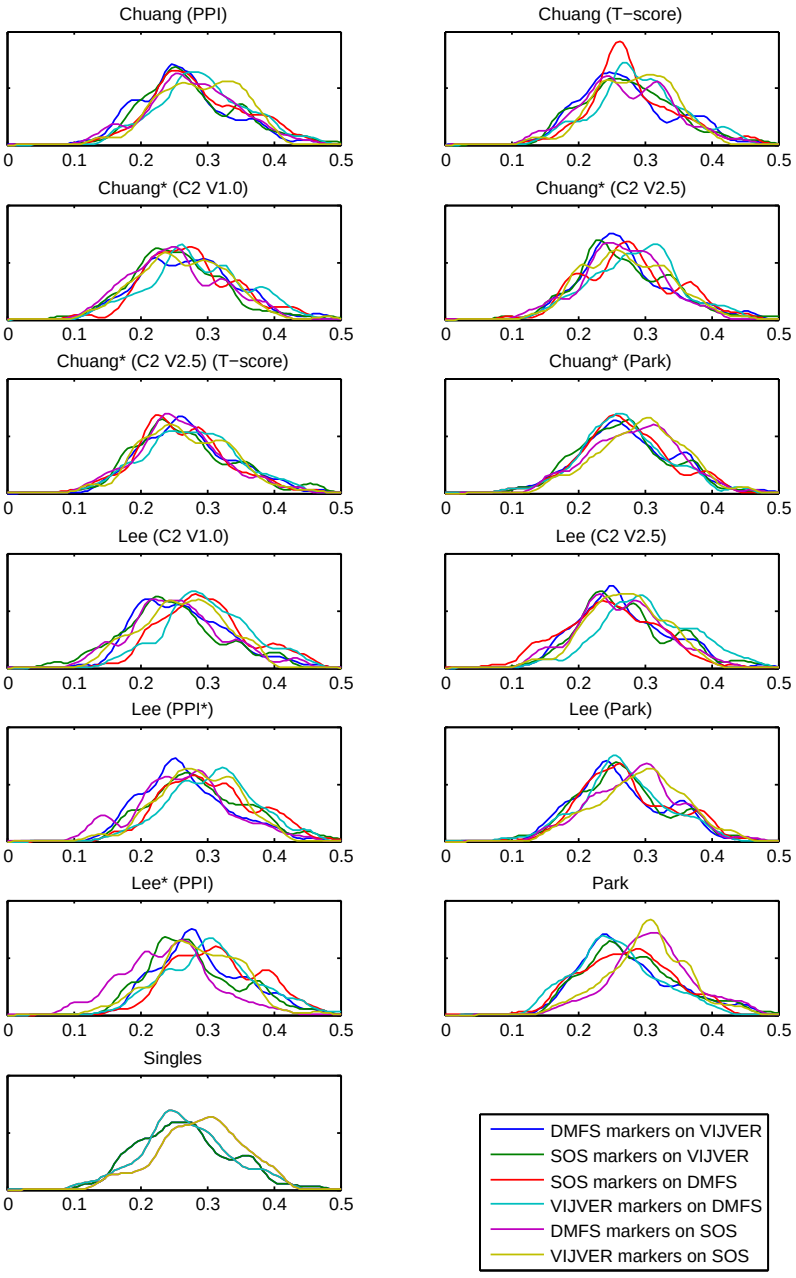


Figure 5: Some histograms.

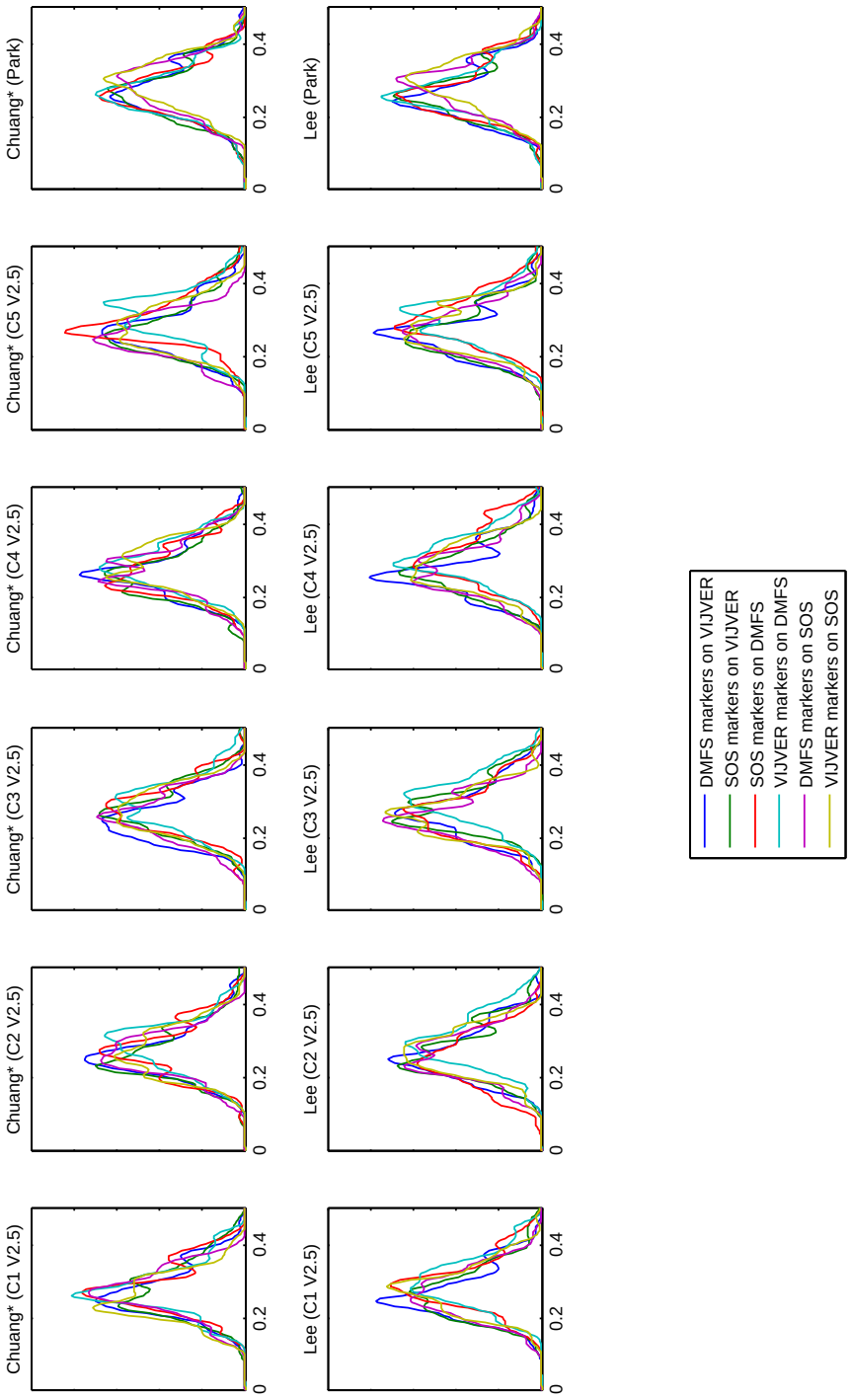


Figure 6: Histograms for comparing search method against search space..

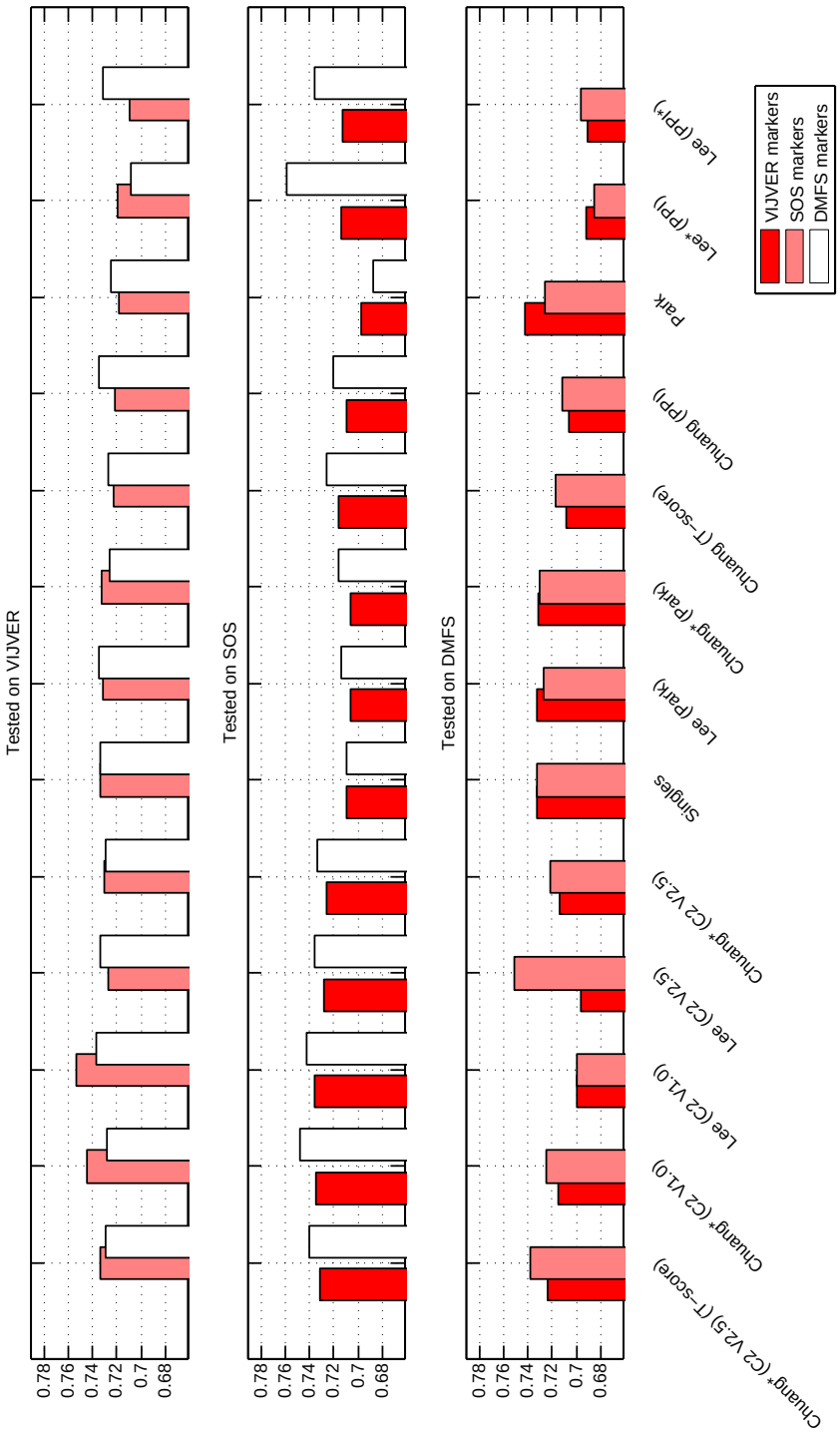


Figure 7: Some barplots.

1 Changelog

August 3st

1.1 Effect of number of features

Since MsigDB works better than PPI in the algorithm comparison, it seems that a W with a lower number of subnetworks performs better than a W with a large number of subnetworks. To see how large W would do with less subnetworks the top n-subnetwork were taken. If a dataset X was used to obtain W, the markers in W were ranked according to absolute t-score calculated in X'. Even though it's biased, it seems that taken the top n subnetworks in this manner overall decreases the performance, and the overall ranking between methods stays the same. See Figure 1 for a comparison and Figure 2 for the barplots. Table 1.1 gives a more general idea on how these subnetworks are composed of.

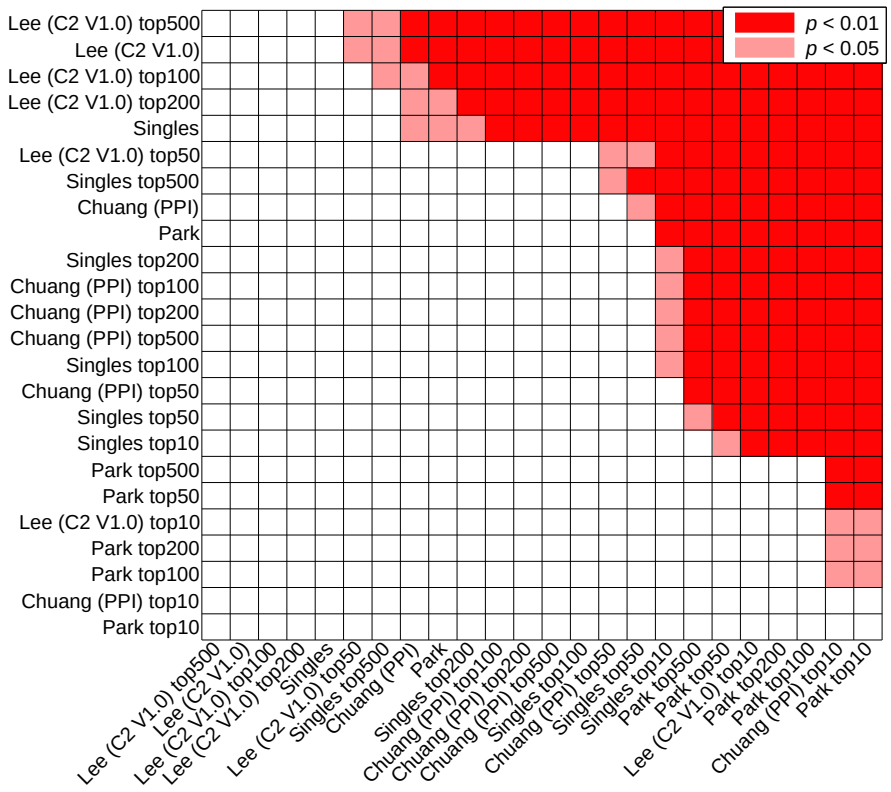


Figure 1: Comparisons algorithm by taking the top n subnetworks. The overall ranking between the type of algorithms stays the same, and it seems that more subnetworks is better.

1.2 Chuang*(C2 V1.0)(T-score)

Since Chuang*(C2 V1.0) performed pretty well, as well as Chuang*(C2 V2.5)(T-score), a logical choice would be to inspect Chuang*(C2 V1.0)(T-score). It's performance is plotted in Figure 3. It doesn't seem to improve over Chuang*(C2 V1.0).

1.3 More insight in the cross validation

Figure 4 is reshown here for comparison between algorithm. The inner loop of the double-loop-cross-validation selects an optimum number of features. If the DLCV calculates the average of 100 AUCs, 100 numbers of features are calculated. To get a feeling of how many numbers are selected, Figure 5, Figure 6 and Figure 7 shown the distribution of markers trained on X_1 tested on X_2, sorted by size. Tables 1.1 gives an idea of how the properties of the subnetworks.

Table 1.3 thru Table 1.3 shown the top 10 subnetworks for each marker set on a testing dataset. The top 10 was compiled by running the DLCV, and counting how many times in the 100 loops of the DLCV the subnetwork was selected by the inner loop. Some observations here are that Lee(PPI*) contains quite a few redundant subnetworks. The top 10 VIJVER markers tested on DMFS has, for Lee(PPI*) 5 out of 10 redundant subnetworks in its top 10. The top 10 DMFS markers tested on VIJVER has for Lee(PPI*) 8 subnetworks which are very similar, but not quite redundant. The top 10 SOS markers tested on VIJVER has 7 instances of PRC1 for Lee(PPI*).

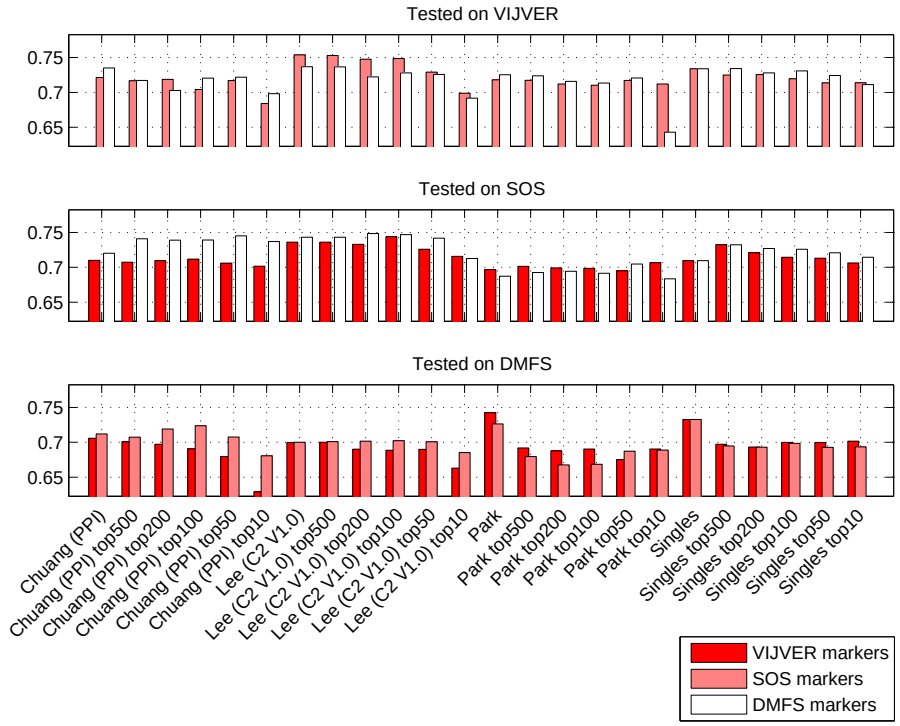


Figure 2: Barplots of top n subnetworks.

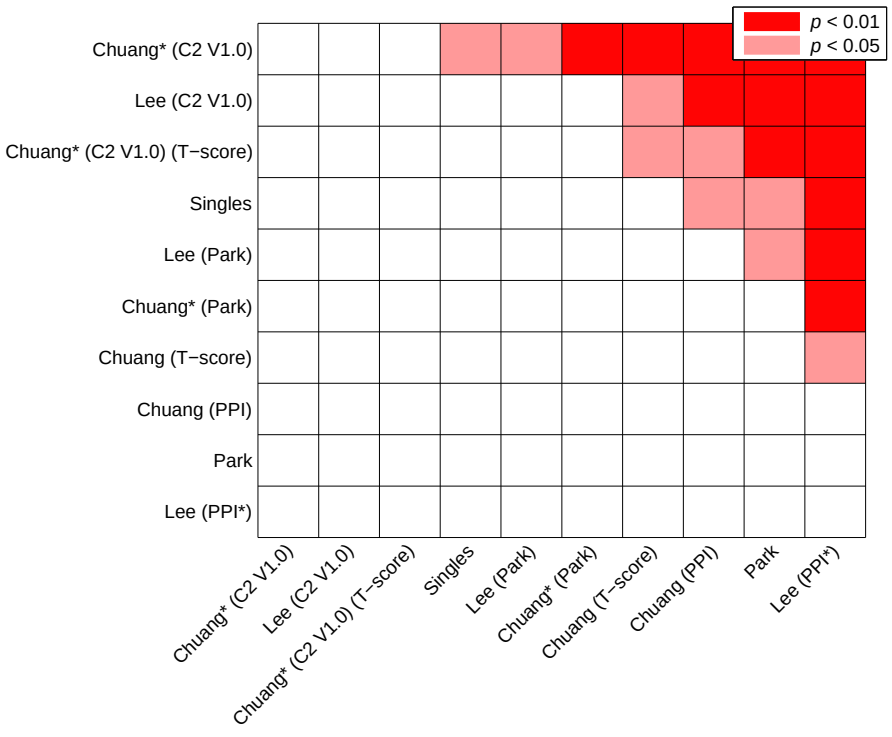


Figure 3: Chuang*(C2 V1.0)(T-score) included.

		subnetworks	unique genes	largest	genes	avg size	doubles
Chuang (PPI) top10	DMFS	10	86	13	101	10	0
Chuang (PPI) top10	SOS	10	63	10	73	7	0
Chuang (PPI) top10	VIJVER	10	68	12	78	7	1
Chuang (PPI) top10	WANG	10	38	10	72	7	2
Chuang (PPI) top50	DMFS	50	265	13	471	9	2
Chuang (PPI) top50	SOS	50	227	12	345	6	1
Chuang (PPI) top50	VIJVER	50	202	12	349	6	10
Chuang (PPI) top50	WANG	50	214	12	394	7	3
Chuang (PPI) top100	DMFS	100	436	13	883	8	4
Chuang (PPI) top100	SOS	100	349	12	670	6	3
Chuang (PPI) top100	VIJVER	100	336	12	651	6	14
Chuang (PPI) top100	WANG	100	338	12	704	7	7
Chuang (PPI) top200	DMFS	200	650	13	1639	8	11
Chuang (PPI) top200	SOS	200	595	12	1287	6	9
Chuang (PPI) top200	VIJVER	200	577	14	1269	6	20
Chuang (PPI) top200	WANG	200	592	12	1344	6	9
Chuang (PPI) top500	DMFS	500	1239	13	3688	7	19
Chuang (PPI) top500	SOS	500	1153	12	2951	5	27
Chuang (PPI) top500	VIJVER	500	1115	14	3104	6	34
Chuang (PPI) top500	WANG	500	1141	12	3052	6	33
Lee (C2 V1.0) top10	DMFS	10	54	10	70	7	0
Lee (C2 V1.0) top10	SOS	10	27	7	43	4	0
Lee (C2 V1.0) top10	VIJVER	10	38	8	54	5	1
Lee (C2 V1.0) top10	WANG	10	64	10	73	7	0
Lee (C2 V1.0) top50	DMFS	50	181	13	300	6	3
Lee (C2 V1.0) top50	SOS	50	114	7	214	4	6
Lee (C2 V1.0) top50	VIJVER	50	130	9	218	4	7
Lee (C2 V1.0) top50	WANG	50	221	12	344	6	1
Lee (C2 V1.0) top100	DMFS	100	289	13	535	5	5
Lee (C2 V1.0) top100	SOS	100	191	7	380	3	18
Lee (C2 V1.0) top100	VIJVER	100	223	9	420	4	15
Lee (C2 V1.0) top100	WANG	100	333	15	623	6	1
Lee (C2 V1.0) top200	DMFS	200	421	13	930	4	12
Lee (C2 V1.0) top200	SOS	200	357	9	745	3	26
Lee (C2 V1.0) top200	VIJVER	200	400	12	871	4	19
Lee (C2 V1.0) top200	WANG	200	498	15	1088	5	3
Lee (C2 V1.0) top500	DMFS	500	749	13	1773	3	43
Lee (C2 V1.0) top500	SOS	500	671	9	1552	3	58
Lee (C2 V1.0) top500	VIJVER	500	704	12	1701	3	51
Lee (C2 V1.0) top500	WANG	500	822	15	2014	4	28
Park top10	DMFS	10	15	4	15	1	0
Park top10	SOS	10	51	39	51	5	0
Park top10	VIJVER	10	51	34	51	5	0
Park top10	WANG	10	13	3	13	1	0
Park top50	DMFS	50	162	81	162	3	0
Park top50	SOS	50	105	39	105	2	0
Park top50	VIJVER	50	100	34	100	2	0
Park top50	WANG	50	63	5	63	1	0
Park top100	DMFS	100	262	81	262	2	0
Park top100	SOS	100	170	39	170	1	0
Park top100	VIJVER	100	157	34	157	1	0
Park top100	WANG	100	128	5	128	1	0
Park top200	DMFS	200	411	81	411	2	0
Park top200	SOS	200	284	39	284	1	0
Park top200	VIJVER	200	272	34	272	1	0
Park top200	WANG	200	260	20	260	1	0
Park top500	DMFS	500	893	81	893	1	0
Park top500	SOS	500	606	39	606	1	0
Park top500	VIJVER	500	614	34	614	1	0
Park top500	WANG	500	724	43	724	1	0
Singles top10	DMFS	10	10	1	10	1	0
Singles top10	SOS	10	10	1	10	1	0
Singles top10	VIJVER	10	10	1	10	1	0
Singles top10	WANG	10	10	1	10	1	0
Singles top50	DMFS	50	50	1	50	1	0
Singles top50	SOS	50	50	1	50	1	0
Singles top50	VIJVER	50	50	1	50	1	0
Singles top50	WANG	50	50	1	50	1	0
Singles top100	DMFS	100	100	1	100	1	0
Singles top100	SOS	100	100	1	100	1	0
Singles top100	VIJVER	100	100	1	100	1	0
Singles top100	WANG	100	100	1	100	1	0
Singles top200	DMFS	200	200	1	200	1	0
Singles top200	SOS	200	200	1	200	1	0
Singles top200	VIJVER	200	200	1	200	1	0
Singles top200	WANG	200	200	1	200	1	0
Singles top500	DMFS	500	500	1	500	1	0
Singles top500	SOS	500	500	1	500	1	0
Singles top500	VIJVER	500	500	1	500	1	0
Singles top500	WANG	500	500	1	500	1	0

Table 1: Some numbers to give an idea on how these subnetworks are built. Shown are the number of subnetworks (subnetworks), the total number of unique genes in these subnetworks (unique genes), the genes largest subnetwork (subnetwork), the sum of all the subnetwork sizes (genes), the average size of a subnetwork rounded down (avg size), number of subnetworks that are already represented (doubles). The unique number of subnetworks would be $uniquesubnetworks = subnetworks - doubles$.

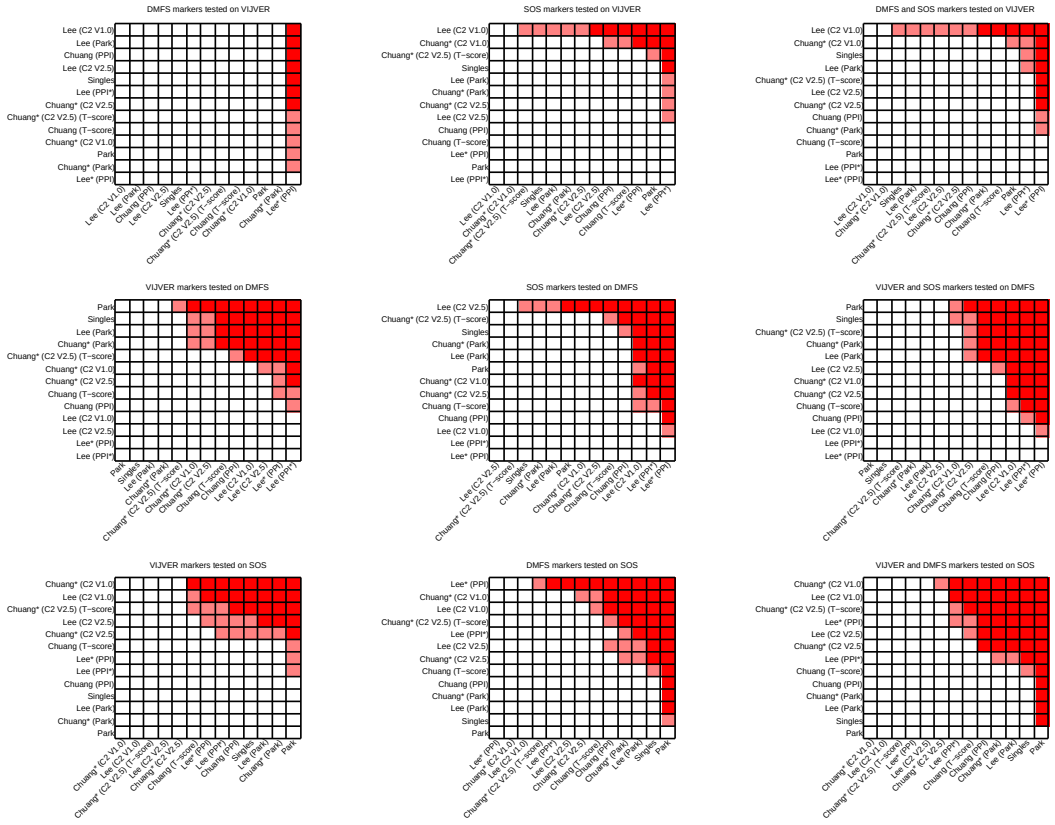


Figure 4: Algorithm comparisons split out in different training-testing combination.

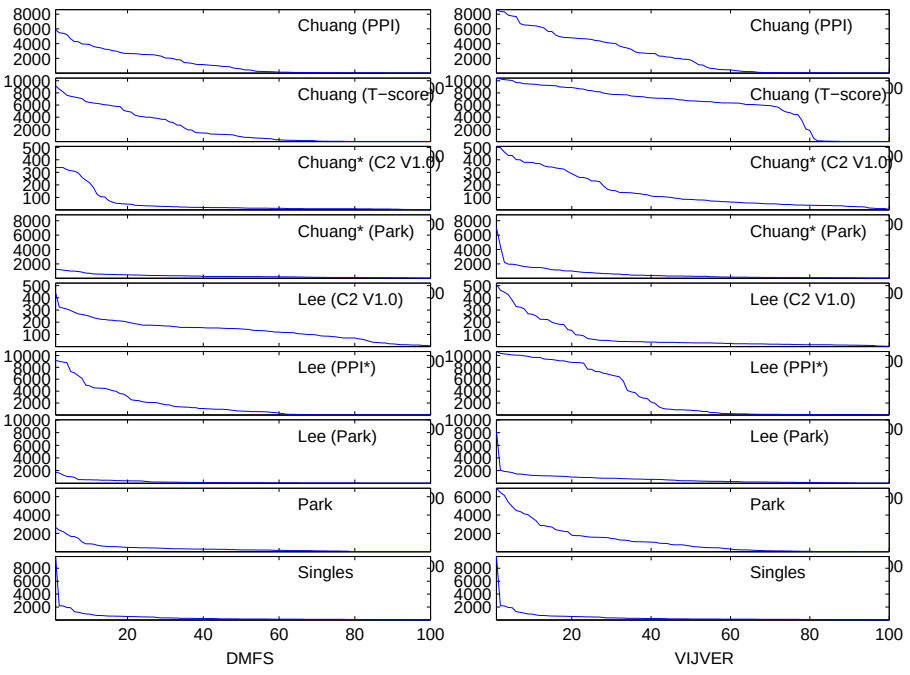


Figure 5: The markers trained on DMFS and VIJVER were tested on SOS in a DLCV. The number of optimum features are sorted according to size.

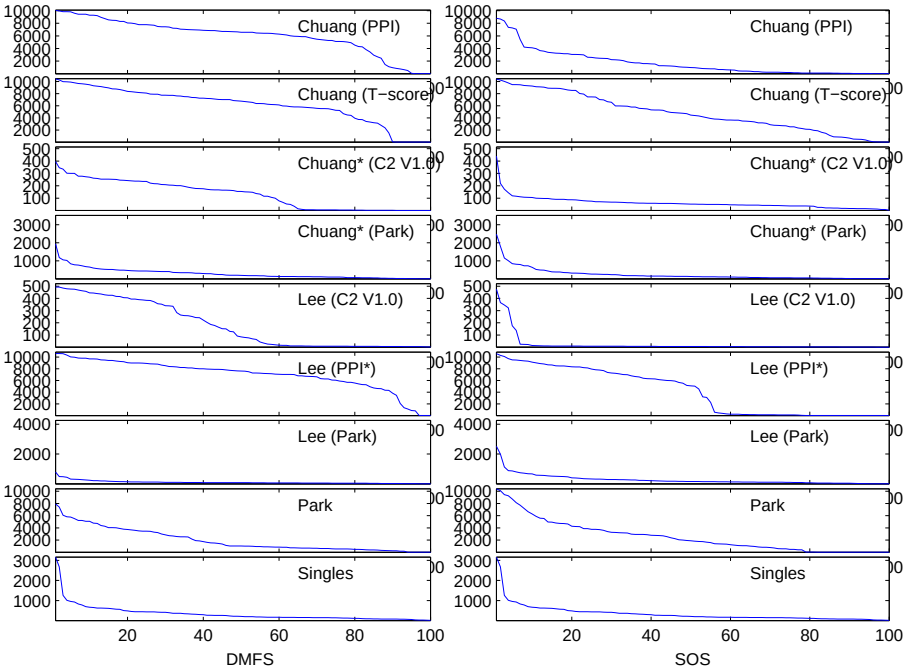


Figure 6: The markers trained on SOS and DMFS were tested on VIJVER in a DLCV. The number of optimum features are sorted according to size.

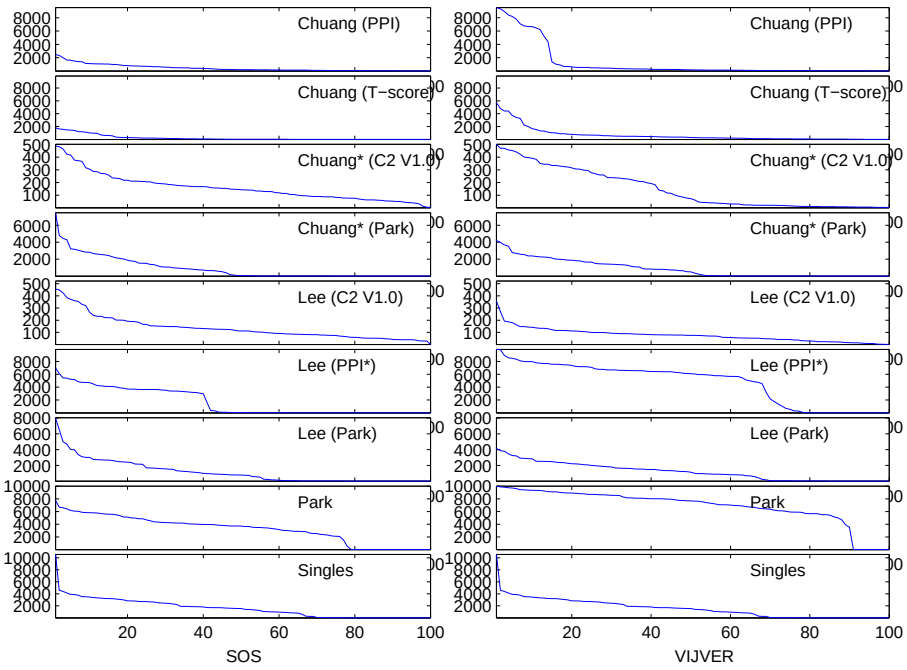


Figure 7: The markers trained on SOS and VIJVER were tested on DMFS in a DLCV. The number of optimum features are sorted according to size.

		subnetworks	unique genes	largest	genes	avg size	doubles
Chuang (PPI)	DMFS	10974	10974	13	36210	3	766
Chuang (PPI)	SOS	10974	10974	13	34917	3	758
Chuang (PPI)	VIJVER	10974	10974	14	35292	3	749
Chuang (T-score)	DMFS	10974	10974	15	51650	4	373
Chuang (T-score)	SOS	10974	10974	17	49151	4	362
Chuang (T-score)	VIJVER	10974	10974	16	49472	4	355
Chuang* (C2 V1.0)	DMFS	522	647	8	1213	2	99
Chuang* (C2 V1.0)	SOS	521	600	6	1132	2	109
Chuang* (C2 V1.0)	VIJVER	519	625	6	1158	2	122
Chuang* (Park)	DMFS	8028	8440	5	8440	1	0
Chuang* (Park)	SOS	10931	11001	3	11001	1	0
Chuang* (Park)	VIJVER	10010	10149	3	10149	1	0
Lee (C2 V1.0)	DMFS	522	775	13	1804	3	43
Lee (C2 V1.0)	SOS	522	691	9	1581	3	64
Lee (C2 V1.0)	VIJVER	522	735	12	1748	3	52
Lee (PPI*)	DMFS	10974	3416	20	43572	3	4141
Lee (PPI*)	SOS	10974	3095	14	36127	3	4862
Lee (PPI*)	VIJVER	10974	3314	17	39065	3	4548
Lee (Park)	DMFS	8186	8523	7	8523	1	0
Lee (Park)	SOS	10962	11036	4	11036	1	0
Lee (Park)	VIJVER	10215	10378	4	10378	1	0
Park	DMFS	8186	11601	511	11601	1	0
Park	SOS	10962	11601	138	11601	1	0
Park	VIJVER	10215	11601	34	11601	1	0
Singles	DMFS	11601	11601	1	11601	1	0
Singles	SOS	11601	11601	1	11601	1	0
Singles	VIJVER	11601	11601	1	11601	1	0

Table 2: Some numbers to give an idea on how these subnetworks are built. Shown are the number of subnetworks (subnetworks), the total number of unique genes in these subnetworks (unique genes), the genes largest subnetwork (subnetwork), the sum of all the subnetwork sizes (genes), the average size of a subnetwork rounded down (avg size), number of subnetworks that are already represented (doubles). The unique number of subnetworks would be $uniquesubnetworks = subnetworks - doubles$.

Tested	Markers	Chuang (PPI)	Chuang (T-score)	Chuang* (C2 V1.0)	Chuang* (Park)	Lee (C2 V1.0)	Lee (PPI*)	Lee (Park)	Park	Singles
DMFS	SOS	ITIH4 CXCL12 PTK2B JAK2	RPS6KA6 MAPK1 HSP90AA1 NEK2 PTPRR HIF1A GRB2	CXCL12 PTK2B	TNFRSF14	ALDH1A1 ABAT ACADS	CDC25A CENPN E2F1 PPFIA1 KRT18 RALA	TNFRSF14	CCT5	TNFRSF14
DMFS	SOS	WDR5 HSP90AA1 EIF2AK2 LSM1 YWHAZ TERT ERBB2	HIF1AN HDAC2 SUV39H1 EED ASH2L HIF1A PGK1 PML	CXCL12 PTK2B	CCT5	CDC2 CCNB1	DLG7 CDC25A CENPN E2F1 PPFIA1 KRT18 HSP90AA1	CCT5	TNFRSF14	EPHX2
DMFS	SOS	RASGRF1 CDC2 CCNB1 CCNB2 CDKN1A CIB2	HAL MAPK1 HSP90AA1 NEK2 PTPRR HIF1A DHPS	PPFIA1 CDH3 CPNE1 BYSL ALCAM	EPHX2	CDC2 CCNB1	STAT5A CXCL12 IL6ST KIT EVL	EPHX2	EPHX2	CCT5
DMFS	SOS	WEE1 CCNB2 CCNB1 GADD45B CDC25B	IER3 MAPK1 NEK2 PTPRR HSP90AA1 HIF1A GRB2 CAMK2D MAPK6	PPP1R12B ADCY1 PTK2B	BTG2	STAT5B STAT5A MAP2K4 FOS BCL2	PDE4A KIT EVL SKAP1 IGJ	TREM1	PARP3	BTG2
DMFS	SOS	VAV1 PTK2B IL6ST CXCL12 PIK3CG	FKBP3 HDAC2 SUV39H1 EED TOP2A PPARD ASH2L GADD45B	STAT5B JAK1	TREM1	TRIP13 CCT5 PGK1 NOL5A PSMD14	IL6ST KIT EVL PDGFRA CSF1 GOLGB1 FLT3 EPOR ARHGEF7 IL6R	PARP3	TREM1	KIF13B
DMFS	SOS	IL2RA STAT5B STAT5A JAK1 PTK2B	CAMK2D MAPK1 HSP90AA1 NEK2 PTPRR HIF1A DHPS	STAT5B JAK1	PARP3	CCNB1 CDC20 TRIP13 HNRPAB ALG3	LRP2 SYNE1 KIF13B ERBB4	BTG2	KIF13B	PARP3
DMFS	SOS	GYS2 CCT2 CCT6A CCT5 GYS1	THAP4 PREI3 FADD GIPC1 MCM2 PAICS	STAT5B JAK1	KIF13B	CCNB1 CDC20 TRIP13	DKFZp762E1312 CCT5 AP1G1	BTD	BTG2	TREM1
DMFS	SOS	CBX5 CCT5 MKI67 TCP1	TNFRSF14 TRAF3 NRIP1 PPARG CTBP2 NR3C1	TRIP13 PGK1	SQLE	CDC2 CCNB1 CDC20 BUB1B	DKFZp762E1312 CCT5 AP1G1	KIF13B	SQLE	BTD
DMFS	SOS	CYB5R2 TRIP13 KIAA1609	HDAC1 TOP2A PPARD RUVBL2 EED GADD45B CCNB1 MDM2 S100A9 ASH2L	IGF1 BCL2 ADCY1	BTD	CDC2 CCNB1 CDC20 BUB1B	PSMD14 STMN1 KRT18 VARS HSP90AA1 TMEM132A PFKL HSPA14 NDRG1	SQLE	BTD	SQLE
DMFS	SOS	INPP5D DOK1 KIT IL6ST PDGFRL	C8orf32 TRIP13 SEC24A SFN KRT18 CDC2 KIAA0408	ATM BCL2	ELOVL5	CDC2 CCNB1	DKFZp762E1312 CCT5 ITGB4BP PPFIA1	ABHD14A	ELOVL5	KIF20A

Table 3: Top 10 subnetworks. Markers from SOS, tested on DMFS.

Tested	Markers	Chuang (PPI)	Chuang (T-score)	Chuang* (C2 V1.0)	Chuang* (Park)	Lee (C2 V1.0)	Lee (PPI*)	Lee (Park)	Park	Singles
DMFS	VIJVER	MAD2L2 MAD2L1 BUB1B BAT2 P4HA2 CENPA	IL6R IL6ST JAK2 STAT5A EPOR RPL4 AR JAK1	BYSL PGK1 TRIP13 NP MORF4L2 HYOU1	TNFRSF14	F11R BA-IAP2 BYSL GP5	RPL11 KIF13B	TNFRSF14	EPHX2	TNFRSF14
DMFS	VIJVER	WEE1 CDCA3 CCNB2 PKMYT1 YWHAB MAP2K1 CCNE2 CCNA2 ITGB1	KRT18 TROAP HGS SFN CDC2 BIRC5 MAP2K1 CDK5R1	BCL2 FCER1A ICOS	EPHX2	CCNB2	RPL11 KIF13B	EPHX2	TNFRSF14	EPHX2
DMFS	VIJVER	FLJ20254 RAD54L RAD51 PSMD7 LSM1 UPF2	HTR2A JAK2 IL6ST STAT5A CSF3 JAK1 DLG4 FZD1 KIF13B BRCA2	JAK2 CISH JAK1 IL6R	TRIP13 CCT5	E2F1 NDRG1 CDC2	RPL11 KIF13B	BTG2	TRIP13 CCT5	CCT5
DMFS	VIJVER	BYSL TROAP TRIM37 KIAA0408 PRC1	PAK2 ARHGEF6 PDHB TGFBR1 PIK3R1 VAV3 IRS1 INS ARHGAP15	BCL2 IL7R STAT5A	KIF13B	BCL2 IGF1 KIT	K-ALPHA-1 F11R GAPDH RGS19 SLC2A1 GRB2 NFASC KCNA2 ITGA5 CLTC DDEF1	TREM1	BTG2	BTG2
DMFS	VIJVER	SKI K-ALPHA-1 PML CCT2 TDG	MAGEA12 STAT5A JAK2 C10orf86 AR RPL4 EPOR JAK1	PSMD1 ABCF1 CDC20	BTG2	JAK2 STAT5A JAK1	K-ALPHA-1 CCT5 NFASC TAF6	BTD	PARP3	KIF13B
DMFS	VIJVER	PIN4 TPX2 AURKA FN1 COL13A1 MMP9 LGALS3BP COL4A1 GTPBP4 NAT10	ERCC1 CCNH GTF2B RRAD POLR1B RPL5 ESR1 RPL11 CCND2 MRPL2	TPX2 CDK2AP1 VIL2	TREM1	BCL2 JAK2 FOS	KIF20A PSMD7	PARP3	TREM1	PARP3
DMFS	VIJVER	K-ALPHA-1 CCT5 THEG CCT2	LOC158997 KPNA1 NP GAPDH	BTG2 BCL6 IGFBP6 FHL2	PARP3	BCL2 STAT5A JAK1 PIK3CA IL2RG PIK3R1	BTG2	KIF13B	SQLE	TREM1
DMFS	VIJVER	THEG CCT5 K-ALPHA-1 CCT2	ROS1 VAV3 IGF1R JAK1 JAK2 ZYX IL6ST KIT HOXA9	CCNB2 ERBB2 RAD51 CCNE1	BTD	RRM2 PGK1 MARS	BTG2	ABHD14A	KIF13B	BTD
DMFS	VIJVER	RAE1 BUB1 BUB1B	TIAF1 JAK3 IL6ST JAK2 JAK1 STAT5A	E2F1 IL11 BUB1B	SQLE	ALDH3A2 ABAT GAD1 DPYD ALDH2	BTG2	SQLE	COCH	SQLE
DMFS	VIJVER	EPHX2	ARHGAP8 CTTN ANKZF1 GRB2 FGD1 KCNA2 ACTR3	JAK2 CISH JAK1 IFNG	ABHD14A	TPX2	BTG2	CDKN3	ZNF395	KIF20A

Table 4: Top 10 subnetworks. Markers from VIJVER, tested on DMFS.

Tested	Markers	Chuang (PPI)	Chuang (T-score)	Chuang* (C2 V1.0)	Chuang* (Park)	Lee (C2 V1.0)	Lee (PPI*)	Lee (Park)	Park	Singles
SOS	DMFS	MT1X TCF1 C16orf61 KIF20A NOP17 NAT10 HSD17B2	EPOR STAT5A JAK1 PTK2B CXCL12 IL6ST FYN EVL IL6R	BCL2 STAT5A PIK3R1 FOS MAP2K4	STAT5B	STAT5A STAT5B BCL2 FOS JAK2 MAP3K1 PIK3R1 MAP2K4 RAF1	CX3CR1 EVL FUCA1 DOK1 CXCL12 STAT5A SKAP1	STAT5A STAT5B	STAT5A STAT5B	CCNB1
SOS	DMFS	CNTF IL6ST NTRK2 VAV1 JAK1 IFNGR2 PTPN6 KIT STAT5B	STAT5A JAK1 PTK2B IL6ST KIT DOK1 FYN EVL	STAT5A JAK1 MAP2K4 EGF	SLC23A2	STAT5A PTK2B JAK1 STAT5B BCL2 PIK3R1	FIGF IGF1 CACNA1D STAT5A ERBB4 PTK2B ATM	PGK1 UBE2A	SPAG5 TMEM97	STAT5B
SOS	DMFS	SFN CDC2 CCNB1 LATS1 GADD45B KRT18 ORC2L	SFN CDC2 GADD45B CCNB2 LATS1 CDK7	FOS MAP3K14 MAP2K4	PGK1 UBE2A	STAT5A STAT5B FOS JAK2 PTPN6 RAF1 EPOR	DOK1 IGF1 ITPR1 LRP2 STAT5A ITM2B SKAP1 ERBB4 PTK2B JAK1 ATM	SPAG5	SLC23A2	CCNB2
SOS	DMFS	RNF20 UBE2A GAPDH MORF4L2 RACGAP1 C20orf4 NUP54	SYK STAT5A JAK1 IL6ST PTK2B KIT DOK1 FYN EVL SDC3	BCL2 IGF1 CSF2RB PRKAR2B	SPAG5 TMEM97	STAT5A JAK1 STAT5B FOS LCK	C7 DOK1 IGF1 LRP2 STAT5A JAK1 IL6R	GAPDH	GAPDH	KIF20A
SOS	DMFS	TRIM25 SFN PLK4 KRT18 CDC2 C8orf32	PIK3R2 DOK1 KIT JAK2 STAT5B PTK2B EGF ERBB4 TEC JAK1	EIF4EBP1 HK2	HNRPAB DDX41	STAT5A JAK1 STAT5B BCL2 FOS BAD PTPN6 PIK3R1 SOCS3	CX3CR1 EVL FUCA1 DOK1 CXCL12 CCR2 ERCC1 LRP2 STAT5A SKAP1	CCNB2 KIF20A TPX2 PRC1	PFKP	SPAG5
SOS	DMFS	ARHGEF15 PREI3 MCM2 FADD UBE2V2 PAICS PRODH GIPC1	IL4 STAT5A JAK1 PTK2B IL6ST KIT DOK1	FOS IGF1 IGF1R	CX3CR1	STAT5A STAT5B FOS JAK2 PTPN6	FIGF DOK1 CXCL12 STAT5A SKAP1 ERBB4 PTK2B JAK1	SLC23A2	GPR56	GAPDH
SOS	DMFS	SORBS1 SEMA6A EVL EFNB1 LYN	CTF1 IL6ST JAK1 VAV1 KIT IL6R FLT3	CCNB1	GAPDH	JAK1 IL6R FOS IL6ST	EVL GJB1 STAT5A SKAP1 PTK2B IL6R PECAM1 TNFSF11 IL6ST CD59 EGF PTP4A2	HNRPAB	AP2S1 SAE1	PLSCR4
SOS	DMFS	NCK1 DOK1 KIT EGF ERBB4 VTN IGF1 TP53	USF2 FOS NR3C1 MYB STAT5A JAK1 TCF1 PCAF	STAT5B FOS CSF2RB	SAE1 AP2S1	STAT5A STAT5B FOS JAK2 PRKCB1 PIK3R1	NTRK2 MATN2 EVL FUCA1 DOK1 CXCL12 DUSP4 GJB1 STAT5A SKAP1	GPR56	GART DSCR2 DONSON	IGF1
SOS	DMFS	IGFBP2 IGF1 EGF IGFBP6 IGF1R	PLEKHA8 ARF1 RALA DDEF2 AP1G1 ARL4D RCC1	MAPT PTK2B STAT5A AGTR2	CD302	CX3CR1 ED-NRB CCR2 PTGER3 ADRB2 AGTR1	MATN2 EVL GJB1 STAT5A SKAP1 FCER1A PTK2B JAK1 IL6R	PFKP	EIF4EBP1	CX3CR1
SOS	DMFS	TLN1 LRP2 PTK2B CXCL12 JAK1 ITIH4 IL6ST	JAK3 STAT5A JAK1 IL6ST VAV1 KIT DOK1	CCNB1	C16orf61 CIAPIN1 PSMD7 C16orf80 NUTF2	PFKL PFKP TALDO1	NTRK2 EVL FUCA1 PCAF DOK1 STARD13 GJB1 STAT5A	SAE1 AP2S1	CD302	SLC23A2

Table 5: Top 10 subnetworks. Markers from DMFS, tested on SOS.

Tested	Markers	Chuang (PPI)	Chuang (T-score)	Chuang* (C2 V1.0)	Chuang* (Park)	Lee (C2 V1.0)	Lee (PPI*)	Lee (Park)	Park	Singles
SOS	VIJVER	ZNF622 MYBL2 SKP2 MELK CDC34 NCOR2	PRKCI GAPDH TK1 PGK1 MAP2K5 MAP3K3 UBE2A	PFKL PFKP ALDOC G6PD	STAT5B	PFKL GALK1 PFKP HK3 HK2 GLB1	AURKA PFKL PSMD2 SPAG5 AARS PGK1	STAT5B	STAT5B	CCNB1
SOS	VIJVER	IFNGR2 JAK1 IGF1R STAT5A IL6ST	PAFAH1B3 GAPDH TK1 PGK1	MAP3K14 NR3C1 DUSP1 IKBKB CREBBP	AP2S1	BCL2 IGF1 KIT	PFKL MAD2L1 PFKP SLC27A3	PLSCR4	PLSCR4	STAT5B
SOS	VIJVER	PAFAH1B3 GAPDH TK1 PGK1 SLC2A1	KLKB1 IGF1 INS C1QBP SIRT1 IGFBP4 PLSCR4	BYSL PGK1 TRIP13 NP MORF4L2 HYOU1	HNRPAB	PFKL TALDO1 PFKP ALDOC	PFKP GAPDH SLC27A3	SLC23A2	SPAG5	CCNB2
SOS	VIJVER	HOXB1 MEIS1 PBX3 NR3C1 STAT5A	RAD18 UBE2A GAPDH	TPI1 CPT1A	SLC23A2	FOS MAP3K1 MAP2K4 STAT5A JAK1 PIK3CA	GAPDH SLC27A3 RNASEH2A ALDOC	SPAG5	AP1G1 CDK8 AP1S1 GEMIN7	KIF20A
SOS	VIJVER	NOL5A TPX2 AURKA SMAD3 TUBA1 MLL2	HTR2A JAK2 IL6ST STAT5A CSF3 JAK1 DLG4 FZD1 KIF13B BRCA2	IGF1	STAT5A	FOS MAP3K1 MAP2K4 STAT5A JAK1 PIK3CA	PFKL FEN1 PFKP	IGF1	STAT5A	SPAG5
SOS	VIJVER	IGF1 NOV INS ITGAV IRS1 LRP2	MYO7A UBE2A GAPDH	POLD1 GMPS NP APRT POLR2D ATIC	SPAG5	CCNB1 CDC2 HRAS	RACGAP1 PFKL PGK1	STAT5A	CDC2 H2AFZ MAD2L1 ZWINT	GAPDH
SOS	VIJVER	SOD2 MDH2 PFKL PFKP	SFN CDC2 E2F1 SPAG5 CDK5R1	MAP2K4 FOS ASAH1 CREB1	PLSCR4	RRM2 POLD1 GMPS NP POLR2C PKM2	DTL KIF20A PGK1	AP2S1	HNRPAB	PLSCR4
SOS	VIJVER	GUF1 HTRA2 PFKL PFKP KARS PKM2 PIN1	GRB10 JAK2 IL6ST INS IRS1 PPP4R1 IFNG STAT5A	RRM2 PGK1 NDUFC1 AGPAT3	ZWINT H2AFZ	E2F1 NDRG1 CDC2	DTL KIF20A PGK1	HNRPAB	SLC23A2	IGF1
SOS	VIJVER	HHEX CTBP2 SNAI2 BAZ2B RAI2	TENC1 PDLIM5 HNRPH2 STAT5A	BCL2 MAP2K4 JUN PRKCQ	GPR56	PFKL TPI1 PFKP HK3 ALDOC	TXNRD1 PFKP GAPDH GLRX2 NDUFA4L2 GRB2	ZWINT H2AFZ	TRIP13 CCT5	CX3CR1
SOS	VIJVER	EPOR STAT5A SOCS2 JAK1 IL6ST CD247	IL21R JAK1 IL6ST JAK2 STAT5A CSF3	BCL2 IL7R STAT5A	C12orf35	H2AFZ BUB1 TALDO1 HDGF ADFP	STC2 MAP2K4 AKAP12 SLC9A5 ADRB2	MKI67	IGF1	SLC23A2

Table 6: Top 10 subnetworks. Markers from VIJVER, tested on SOS.

Tested	Markers	Chuang (PPI)	Chuang (T-score)	Chuang* (C2 V1.0)	Chuang* (Park)	Lee (C2 V1.0)	Lee (PPI*)	Lee (Park)	Park	Singles
VIJVER	DMFS	SLC25A11 COPA ARFGAP1 CDKN1A CCNB1 CCNE2 CCNB2 GADD45B MYC	DIAPH1 CENPA BAIAP2 MAD2L1	PGK1 PFKL	E2F1	H2AFZ MKI67 PSMA7 PSMD2 PSMD1	TMPO HSP90AA1 SPP1 F11R BAT3 ZG16	PKMYT1 E2F1	BIRC5 TK1 HN1	E2F1
VIJVER	DMFS	MYT1L PKMYT1 CCNB2 CCNB1 CCNA1 CCNE2	PRC1	E2F1 CFL1	ADAM8	PFKL PFKP TALDO1	TMPO HSP90AA1 SPP1 F11R EN2 BAT3 ZG16	GLTSCR2 TPT1	E2F1 PKMYT1	TK1
VIJVER	DMFS	PRKCBP1 BIRC5 PSMD2	FBXO32 CUL1 E2F1 CDCA3 CCNA2 CCND1	E2F1	EBP	E2F1 CFL1 ARF3	TMPO HSP90AA1 SPP1 F11R BAT3 ZG16	AURKA	AURKA	PRC1
VIJVER	DMFS	LIMK1 PAK4 YWHAZ RACGAP1 LATS1	GDF8 SGTA TMPO SPP1 HSP90AA1 BAT3 PTN F11R EFEMP2	E2F1	BIRC5	TPI1 HK2 PMM2 PFKL PFKP SORD PFKFB1	TMPO HSP90AA1 SPP1 F11R BAT3 ZG16	EBP	EBP	ESPL1
VIJVER	DMFS	PAK4 RACGAP1 AURKB	RUTBC1 RPS25 EIF3S4 CA12 RPL11	BIRC5	AURKA	CFL1 ACTR3 BAIAP2	TMPO HSP90AA1 SPP1 F11R BAT3	CCNB2 KIF20A TPX2 PRC1	ADAM8	BIRC5
VIJVER	DMFS	WBP2 PSMD2 PSMA7 ORC1L	ANKZF1 AURKB RACGAP1 TACC1 PSMD1 PSMD7	E2F1 CCNA2	PSMD2	CDC2 MAD2L1 ATP2A2 E2F1	TMPO HSP90AA1 SPP1 F11R BAT3 ZG16	HN1 BIRC5	ADRA2B	E2F2
VIJVER	DMFS	RACGAP1 AURKB PAK4	USHBP1 PRC1	PSMA7	RCE1	CCNB2	TMPO HSP90AA1 SPP1 F11R BAT3 ZG16	ADAM8	PFKL	TPT1
VIJVER	DMFS	RGS3 YWHAZ PCTK1 BRAF CDC25B CDC25A PTPN13	UNC84A RRM2 EIF4G1 NEURL	E2F1 ABL1	GLTSCR2 TPT1 RPS27A	PGK1	TMPO HSP90AA1 SPP1 F11R BAT3 GJA8	PSMD2	SLC1A5	CCNB2
VIJVER	DMFS	BRD2 E2F1 CCNA1	RPL12L3 AARS SEC61G RAD51	BCL2 STAT5A PIK3R1 FOS MAP2K4	DDX39	PGK1	CDCA3 E2F1 YWHAZ	WDR62	CENPM	EBP
VIJVER	DMFS	PRC1	POLD1 FEN1 EXO1	E2F1 TIMP3	PGK1 UBE2A	PGK1	DDX39 SNRPA1 PSMA7 PSMD2 POLR2B	STIP1	WDR62	UBE2C

Table 7: Top 10 subnetworks. Markers from DMFS, tested on VIJVER.

Tested	Markers	Chuang (PPI)	Chuang (T-score)	Chuang* (C2 V1.0)	Chuang* (Park)	Lee (C2 V1.0)	Lee (PPI*)	Lee (Park)	Park	Singles
VIJVER	SOS	ARF3 KIF23 AURKB AURKC RACGAP1 ARF5 ARF1	TRIM37 PRC1 DLG7 APEX2 KIAA0408 PNKP	PFKP HK2 GALK1	TK1	PFKP PFKL ARF1	BUB1 ARF1 NDRG1 ARF3 PPP2R1A UTP14A	TK1	TK1	E2F1
VIJVER	SOS	ARHGDIA FEN1 POLD1 CDC42	TK1 GAPDH UBE2A	PGK1 GOT1 ALDOC	DKFZp762E1312 TROAP	E2F1 ARF1 ARF3 CFL1	RRM2 PFKP PFKL	DKFZp762E1312	TROAP DKFZp762E1312	TK1
VIJVER	SOS	FEN1 ARHGDIA POLD1 CDC42	TPT1	POLD1 RRM2	E2F1	RRM2 POLD1 TK1	PFKP ACP1 E2F1 PFKL DPM2	BIRC5	BIRC5	PRC1
VIJVER	SOS	PRC1	BLK BCL2 ITM2B SF1 RPS3A BNIP3L	TRIP13 PGK1	BIRC5	PFKP TPI1 PFKL HK2 HK3	PRC1	E2F1	E2F1	ESPL1
VIJVER	SOS	USHBP1 PRC1	EEF1A2 PSMD1 RACGAP1 CDC25A PSMB7 CHRM4	ARF3 E2F1 CFL1	E2F2	PFKP PFKL HK2 HK3	PRC1	PKMYT1	PKMYT1	BIRC5
VIJVER	SOS	GTF2H5 GTF2H4 CDC2 E2F1 TAF13 TAF4	SPG7 RALY WDR62 PLSCR1 CPSF6 VASP NPDC1 TXNL2	GOT1 AARS	AURKA	FEN1 POLD1 MSH6 EXO1	PRC1	TPT1	TPT1	E2F2
VIJVER	SOS	CFL1 TPI1 PGK1	PRC1	CPT1A TPI1	ADAM8	NDRG1 SLC19A1	PRC1	EBP	AURKA	TPT1
VIJVER	SOS	FENL1 FEN1 ARHGDIA POLD1 PRIM2A VCL EXO1	DCAMKL1 GAPDH UBE2A TK1	CPT1A TPI1	TPT1	PGK1 GOT1	PRC1	E2F2	E2F2	CCNB2
VIJVER	SOS	TPI1 PGK1 CFL1	USHBP1 PRC1	POLD1 MSH6 EXO1	PKMYT1	PFKP TALDO1 GPI PFKL	PRC1	AURKA	EBP	EBP
VIJVER	SOS	POLD4 POLD2 RFC2 POLD1 PRIM2A FEN1 EXO1 CDKN1A CCNA2 CDC45L	ARHGDIA FEN1 PRIM2A POLD1	PFKP ARF1	EBP	GOT1 AARS	PRC1	ADAM8	ADAM8	UBE2C

Table 8: Top 10 subnetworks. Markers from SOS, tested on VIJVER.

August 21th

1 Models

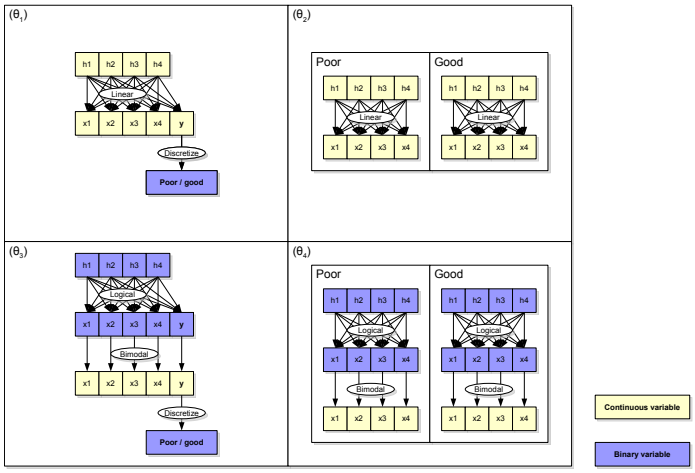


Figure 1: Different types of models. Models θ_1 and θ_2 use a normally distributed hidden layer where the features and linear combinations of these hidden variables. Models θ_3 and θ_4 use binary hidden layer where the features are logical functions of the hidden variables, which in turn are converted in normally distributed features. In models θ_1 and θ_3 the outcome variable is modeled as being another feature, and discretized to get a 0-1 label. Models θ_2 and θ_4 have two separate models for the poor and good case.

In order of increasing complexity, the different models are:

- Model θ_1. In this model, a set of normally independent distributed variables H model the latent variables. The features X are a linear combination of the latent variables with added noise. The outcome variable Y is modeled similar as the a feature of X, but discretized. This is the model described in Park.

- Model θ_2. This model is has a set of normally independent distributed variables H for each of the poor and good case. The features X are a linear combination of each of these latent variables. Also, not shown in the Figure, the model should have a prior probability of good and poor case.

- Model θ_3. In this model, a set of independent Bernoulli variables H model the latent variables. The features $X_{0/1}$ are logical functions of these latent variables. The features X are normally distributed with parameters depending on its corresponding $X_{0/1}$. The outcome variable Y is again regarded as a feature, but discretized.

- Model θ_4. This model is similar to θ_3, but with seperate systems for the poor and good case. This model also should have a prior probability.

More formally, the different models are noted using the following variables. In this notation, B indicates a Boolean function.

- Model θ_1. μ_{h_i}, $\sigma^2_{h_i}$ for $h_i \in H$. $\vec{\beta}_{x_i}$, $\sigma^2_{x_i}$ for $x_i \in X$. $\vec{\beta}_y$, σ^2_y.

- Model θ_2. $\mu_{h_{y,i}}$, $\sigma^2_{h_{y,i}}$ for $h_{y=0,i} \in H_{y=0}$, $h_{y=1,i} \in H_{y=1}$. $\vec{\beta}_{x_{y,i}}$, $\sigma^2_{x_{y,i}}$ for $x_{y=0,i} \in X_{y=0}$, $x_{y=1,i} \in X_{y=1}$. $p_{y=0}$.

- Model θ_3. p_{h_i} for $h_i \in H$. B_{x_i} for $x_i \in X$. B_y. $\vec{\mu}_{x=0,i}, \vec{\mu}_{x=1,i}, \vec{\sigma^2}_{x=0,i}, \vec{\sigma^2}_{x=1,i}$ for $x_i \in X$. $\mu_{y=0}, \mu_{y=1}, \sigma^2_{y=0}, \sigma^2_{y=1}$.

- Model θ_4. This model won't be used since model θ_3 is complex enough for our purposes.

For the simpler models θ_1 and θ_2 an alternative notation exists that directly models the relation between the features and the outcome, which can be used for improved analysis later. These models don't model the hidden layer explicitly.

- Model θ_1. $\vec{\mu_x}$, Σ_x, $\vec{\beta}_y$, σ^2_y.

- Model θ_2. $p_{y=0}$, $\vec{\mu_{y=0}}$, $\Sigma_{y=0}$, $\vec{\mu_{y=1}}$, $\Sigma_{y=1}$.

Since we've obtained simpler models for θ_1 and θ_2, we can easily determine the optimal classifier, or Bayes classifiers for these models. We could also attempt to find the Bayes classifiers for the more complex models θ_3 and θ_4, but this would require too complex notation and won't probably be easily implementable.

- Model θ_1. $\hat{y} = 1$ if $(X\vec{\beta}_y > 0)$, $\hat{y} = 0$ otherwise.

- Model θ_2. $\hat{y} = 1$ if $g_{y=1} > g_{y=0}$, $\hat{y} = 0$ otherwise. Here $g_{y=c} = log(p_{y=c}) - 0.5log(|\Sigma_{y=c}|) - 0.5(X - \mu_{y=c})^T \Sigma^{-1}_{y=c}(X - \mu_{y=c})$.

1.1 Standard model θ_1

The standard model θ_1 has a hidden layer of 100 variables and a feature layer with 1000 features. Every 10 features represent a hidden variable with noise added. The outcome variable y is the mean of the first 10 hidden variables, with noise added to it and discretized. This model has 300 samples.

This model can be described using the following variables:

- $\mu_{h_i} = 0$, $\sigma^2_{h_i} = 1$ for $h_1 ... h_{100}$. $\vec{\beta}_{x_i}(n) = 1$ if $n = \lceil i/1000 \rceil$, otherwise 0, $\sigma^2_{x_i} = 2^2$ for $x_1 ... x_{1000}$. $\vec{\beta}_y(n) = 1$ if $n < 11$ otherwise 0, $\sigma^2_y = 1$.

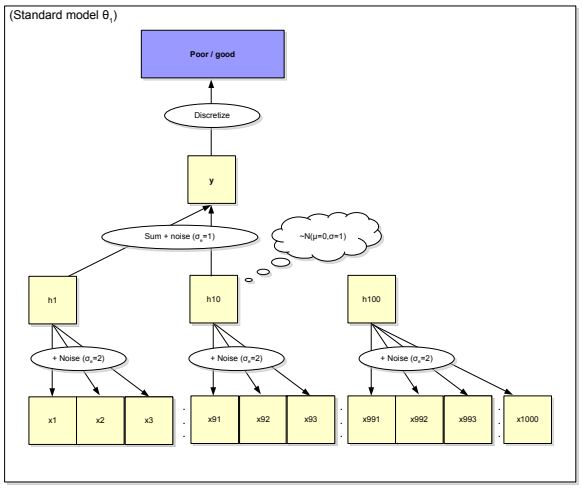


Figure 2: Standard model 1.

1.2 Standard model θ_2

Standard model θ_2 has 100 hidden variables and a 1000 features. There is a system for both the good and the poor case individually. Again every 10 features are represented by a hidden variable with noise added. The difference between the poor and the good case is that the hidden variables have a different mean. Also, a prior probability is defined which determines the approximate number of poor and good cases.

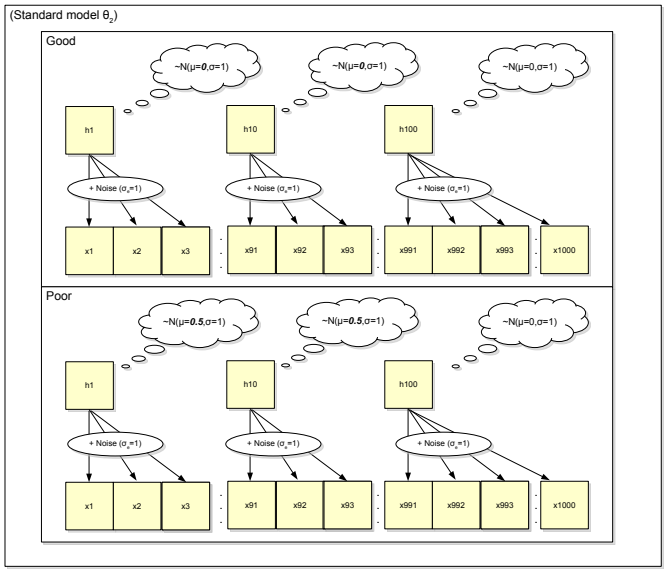


Figure 4: Standard model 2.

1.3 Standard model θ_3

Standard model θ_3 ...

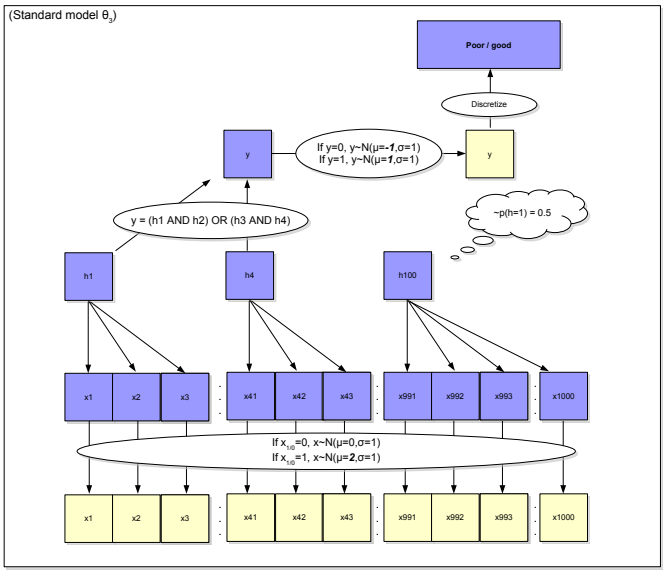


Figure 5: Standard model 3.

1.4 Why these models?

The rationale for choosing these models is that we can have different datasets with different complexity, which as we will hopefully see will have an influence on the efficiency of the gene set searching algorithms. For example, model θ_1 has the important property that the entire model could theoretically be modeled as one big joint multivariate guassian distribution. This means that if variable 1 is possitively correlated with the outcome y, and variable 2 is also possitively correlated with y, then variable 1 must also be correlated with variable 2. This behaviour should be advantegous to an algorithm such as Park, which searches for correlated variables.

However, model θ_2 models the good and poor population as separate systems. We could model the predictors variables in each of these population as independent variables, meaning that two different predictor variables which are predictive for y, will only be slightly correlated to each other. So, these models could be used to show that Park only performs well under certain circumstances.

2 Properties of the DLCV

In this section we will look at the basic performance of running a Double-Loop-Cross-Validation method using T-score ranking and a Nearest Mean Classifier. To do this, we will define three standard models which we use a starting point, which can be varied to observe the change in performance. By doing to so hope to find the basic properties of the DLCV. In the following section, we will look more specifically at what happens when genes are combined, and what the properties are of Park and Lee.

2.1 The DLCV

We will employ a DLCV with a 5 fold outer loop and a 4 fold inner loop. The outer loop is repeated 10 times, returning 50 AUC values to average, while the inner loop has 3 repeats, determining the optimal number of features from the average of 12 learning curves.

2.2 Signal vs noisy features

For each of the standard models, I've inspected the AUC returned by crossvalidaton is we would take the first 5, 10, 15 ... 1000 features. So for model θ_1 and θ_2, we would expect the optimum performance when the first 100 features are selected, which are all signal features. For model θ_3 we would expect this when the first 40 features are selected. Note that when I talk about noisy features, I mean non-signal features, since signal features may also be partly noisy. This experiment leads to our first observation.

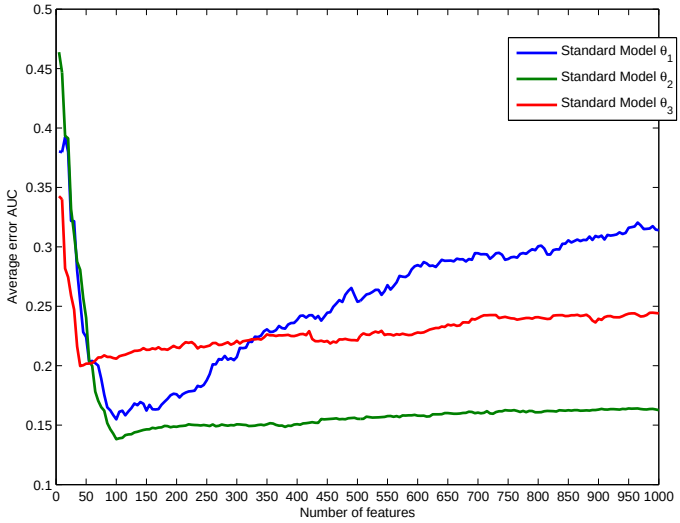


Figure 6: The first n genes where selected in each of the standard models and DLCV was applied after which an average AUC was returned.

Observation 1 *Adding signal genes improves the AUC. Adding noisy signals worsen the AUC. Even though you would expect the DLCV to be able to select the optimum reporter set, the more noisy features are added, the harder this seems to get.*

2.3 Correlated vs uncorrelated signal features

For each standard model, I've inspected whether uncorrelated features are more powerful than correlated features. To inspect this, I've taken, for each model θ_1 and θ_2, the first 10, 20, 30 ... 100 features. The first 10 features are correlated, that is, they come from the same hidden variable. After that, sets of 10 genes are added which are too correlated to each other. This 'learning curve' was compared against a different strategy wherein uncorrelated features were selected first. The first feature of all the 10 hidden variables was used as the starting set, after which the second features of the 10 hidden variables where added, etc. In these models, noise features were left out.

For model θ_1, the first 10, 20, 30 and 40 features where taken to inspect the correlated learning curve, and features [1 5 9 13 17 21 25 29 33 37], [2 6 10 14 18 22 26 30 34 38], etc where used to calculate the uncorrelated learning curve.

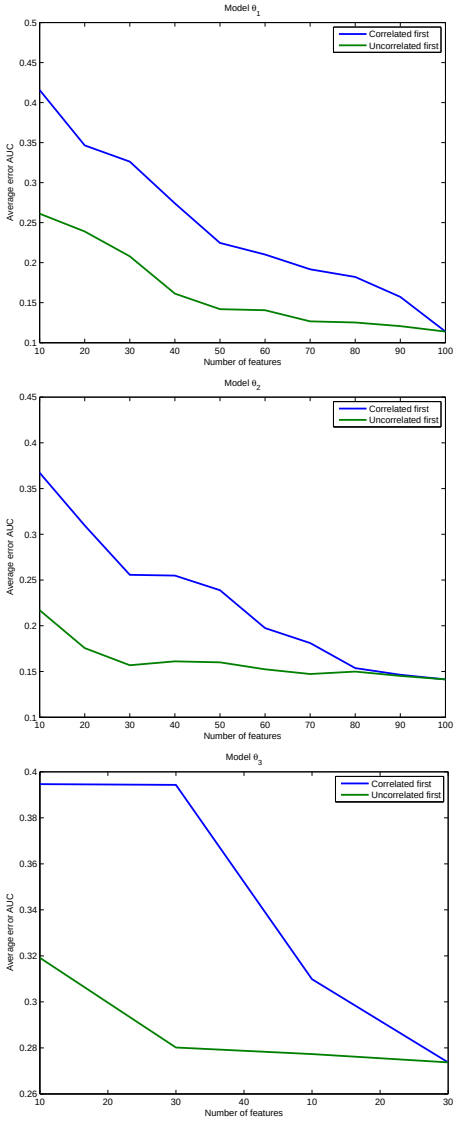


Figure 7: The correlated-first versus the uncorrelated-first learning curve for the three models.

Observation 2 *A subselection of uncorrelated features is more powerful than a subselection of correlated features, even though each feature individually is equally powerful.*

2.4 Duplicated features

Let's see what happens when we introduce duplicated signal features. To do this, we inspected the AUC of all 100 signal features, added these 100 signal features to the dataset and rechecked the AUC, and again added these 100 signal features. This was done for the cases where we only used the signal features, and where the noise features were added.

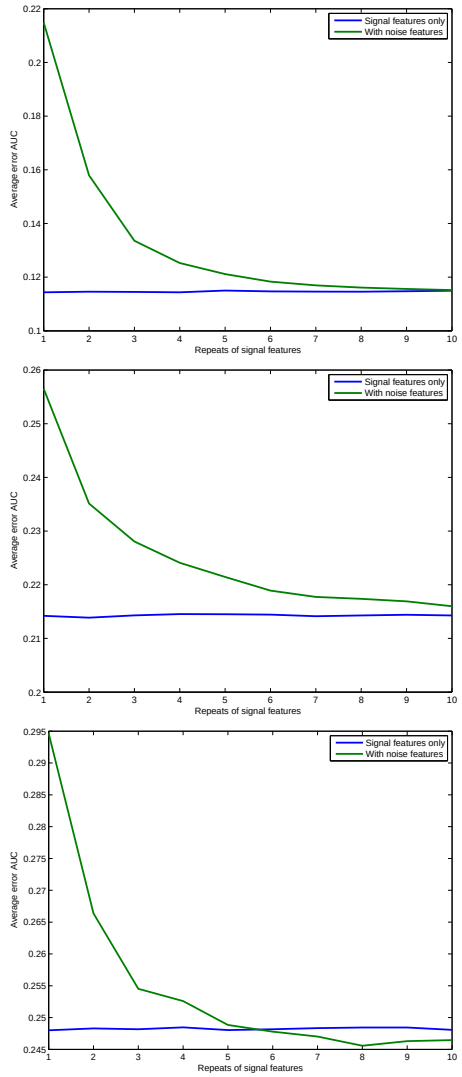


Figure 8: For the three models, the signal genes were duplicated.

Observation 3 *Adding duplicated signal features affect the AUC only if it affects the ratio of signal and noisy features.*

3 Properties of combining features

In this section we will take a look at how combining features affect the AUC.

3.1 Combining correlated features

My first experiment is to see what happens when correlated features are combined. We would expect to see that this improve our AUC's. To do this, I've run the DLCV on all three models. Then, I've taken the first 10 features, combined them, and run DLCV on the new transformed dataset. This was done for both the cases where all 1000 features were used and where only the 100 or 40 signal features are used.

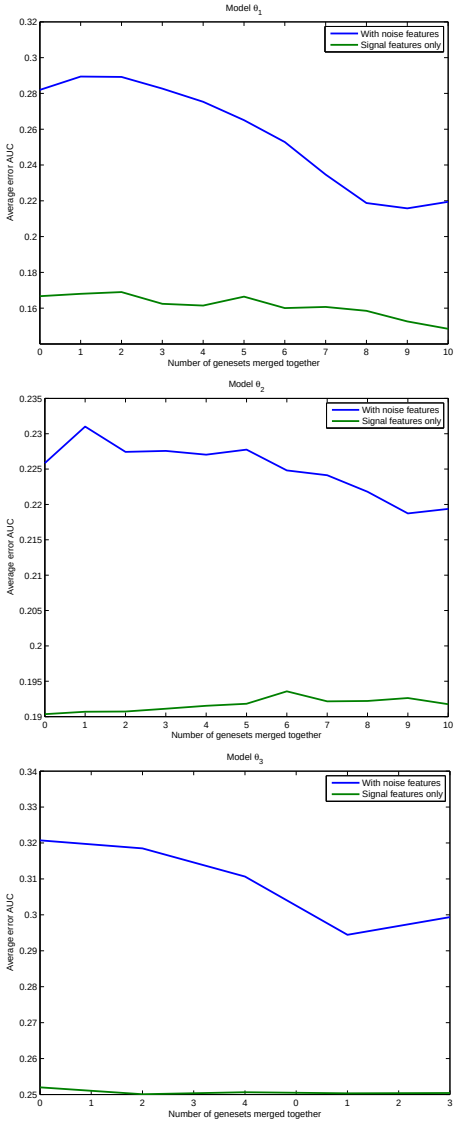


Figure 9: For the three models, groups of features which are correlated were gradually combined.

Observation 4 *Combining features generally improves the AUC. However, the improvement also depends on the presence of noisy features. The best improvement of AUC comes from removal of the noisy features rather than combination of signal genes.*

4 Properties of gene set searching algorithms

4.1 Singles, Park and Lee

We will first take a look at the two basic gene set searching algorithm, Park and Lee. We will vary the error on the features to see how this affects the efficiency of the algorithms. Notice that in standard model θ_2, the variables are correlated in groups of 10, making it similar to model θ_1.

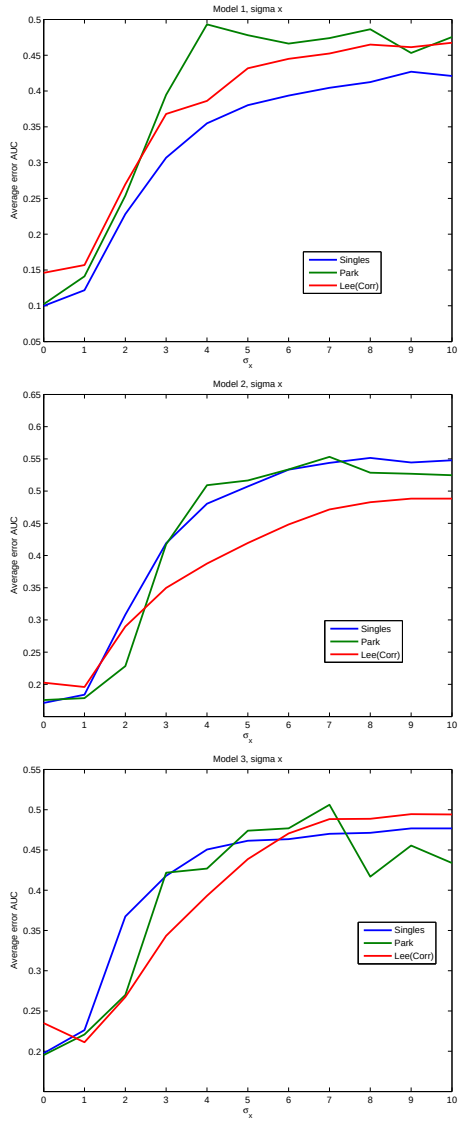


Figure 10: Performances of Park and Lee, depending on the error introduced on the features.

Observation 5 *Judging from the S shapes, Park's algorithm seems much more dependent on the σ_x than the other algorithms. Since a higher σ_x blurs the correlation between the variables, Park will have a harder time building a correct dendrogram.*

5 Conclusions

Basically, an observed performance depends on the following factors:

- The underlying model.
- The DLCV.
- The gene set searching algorithm.

Let's go through these factors.
The underlying model.

- Multiple methods exist in which the data can be modeled. A model which has logical effects incorporated may suffer from bias when a linear classifier is used to predict the outcome. But even within linear models we can find different levels of complexity. For example, the data, along with the outcome labels, may be modeled as a joint multivariate gaussian. If two features depend on the same latent variable, then the two features must be correlated, which is a property that may be advantegeous to a gene set searching algorithm that tries to exploit the correlation structure of the data, such as Park.
- Since our models have a noise incorporated in it, every model brings an irreducible noise with it, which can not be solved with even the best algorithms.

The DLCV

- The DLCV may be regarded as a cross-validation of an advanced classifier, NMC+featsel, which is a NMC with feature selection incorporated in it. So basically, this advanced classifier is a model which itself has a certain bias and variance.
- A bias with the DLCV comes from the fact that it is linear, so it will expectedly perform worse when logical effects are incorporated.
- Another bias is introduced due to the feature selection part. It is expected that the DLCV will work great when the underlying data consists of data has features where the highest ranking features are indeed signal features.
- Also, since my implementation NMC is sensitive to feature scaling and priors, this may cause additional bias. However, due to the normalization and combination of the genes by dividing $\sqrt{(n)}$, the feature scaling shouldn't be a problem. The priors also shouldn't cause trouble since we're mostly working with AUCs.

The gene set searching algorithm

- A gene set searching algorithm returns a mapping W. This mapping may influence the performance of the DLCV in various ways. It could for example remove noise signals, making it less likely for the DLCV to overtrain. It may combine signal features, which produces

new features that are more likely to have a higher t-score and thus will perform better in the featsel ranking of DLCV. However, it may also incorrectly combine wrong signal and/or noise features, which causes a bias due to worsened signal.

- Gene set searching algorithms return a mapping based on a training set. This procedure itself may show high some variance. For example, Park's algorithm selects an optimum cut-off level, which may return a variable number of features, while Lee's algorithm has a predefined number of features wherein each features has a maximum signal. In other words, Park is biased due to the fact that only correlated genes can be combined, while Lee is biased due to the fact that only genes within the same predefined pathways may be combined. This pathway bias may explain why Lee using random sets of pathways may perform better than 'perfect' pathways. In the 'perfect' pathways scenario, we introduce bias since we have a certain number of predefined pathways which have no signal, no matter which subset of CORGs are taken.

5.1 Bias-variance decomposition

I've attempted to make a bias-variance-error decomposition of the observed error in this work. The reason why I've tried to do this is because I believe a well-designed decomposition would give insight in virtually all possible effects than can occur, as can be seen from the above conclusions.

However, while trying to do this, I've encountered a few problems:

- Theoretically, since for all our experimental models we know the underlying 'true' model, we should be able to calculate the bayesian noise analytically. However, this is quite difficult to do for model advanced models such as model θ_3.
- Since this is not a regression problem, but a classification problem with a zero-one loss penalty, the decomposition is not very natural. There is a paper that demonstrates how to calculate this for classification problems, but the variance, bias and error terms may not be summed, making it harder to interpret the results.
- Approximation of the bias and variance terms would require too much artificial data and simulation time, which is infeasible for now.
- Bias-variance-error applies to error per example, not the AUCs.

Still, I would recommend such an approach for analysis of the algorithms on artificial data since it immediately points out the strong and weak points in the design of the model and gene set searching algorithms.

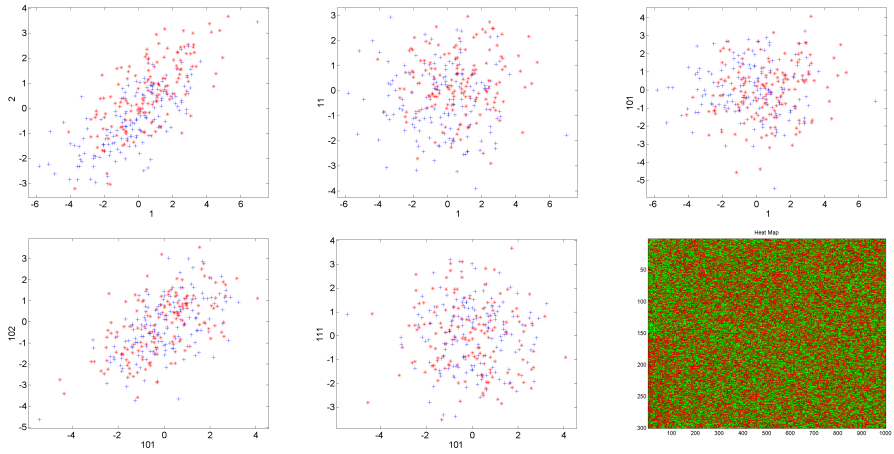


Figure 3: Example of standard model 1. From left right, top to bottom, we see: two correlated and signal features, two uncorrelated signal features, a signal and a noise feature, two correlated noise features, two uncorrelated noise features, a heatmap with the first 150 samples poor and the other 150 samples good.

1 Gene set searching algorithms evaluation

1.1 Models

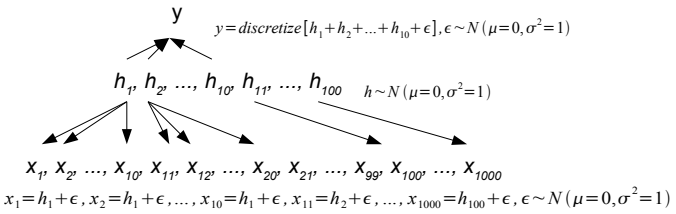


Figure 1: Standard model 1

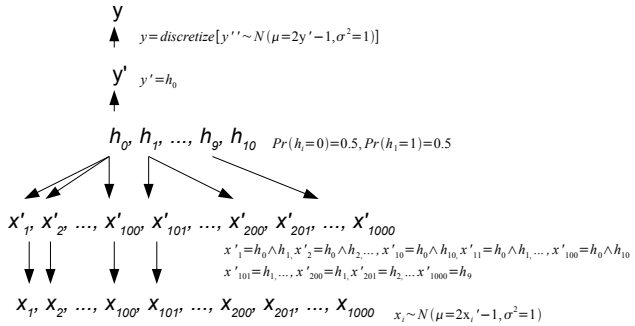


Figure 2: Standard model 2

1.2 Gene sets

See Figure 5.

1.3 Experiments

Each point, along with the errorbar, is generated using 5 datasets. For these 5 datasets, 5 corresponding mappings were generated. This datasets were subjected to a cross-dataset-cross-validation, so each point, and its variance, is based on 5x4=20 average AUCs. Most of the graphs should speak for themselves, altough for clarity Figure 7 and Figure 8 are visualized in Figure 6. Figure 9 are basically 'Correlated first, combined' and 'Uncorrelated first, combined' experiments from Figure 8, but with the remaining 100 or 1000 single genes included.

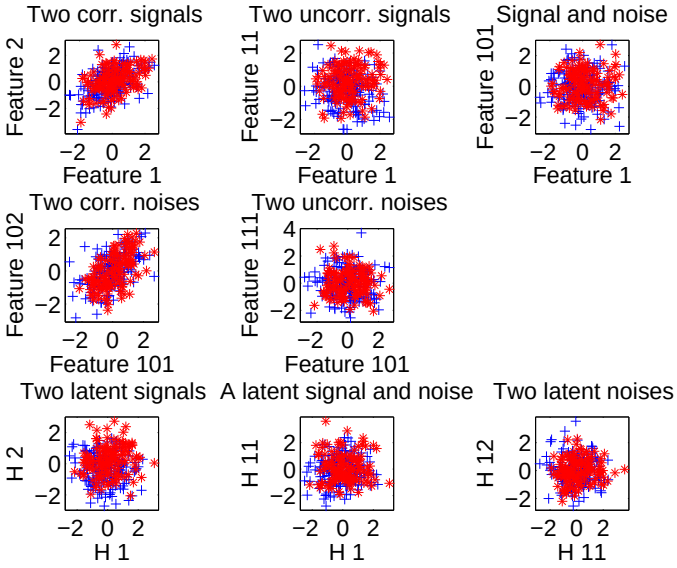


Figure 3: Scatterplots of standard model 1

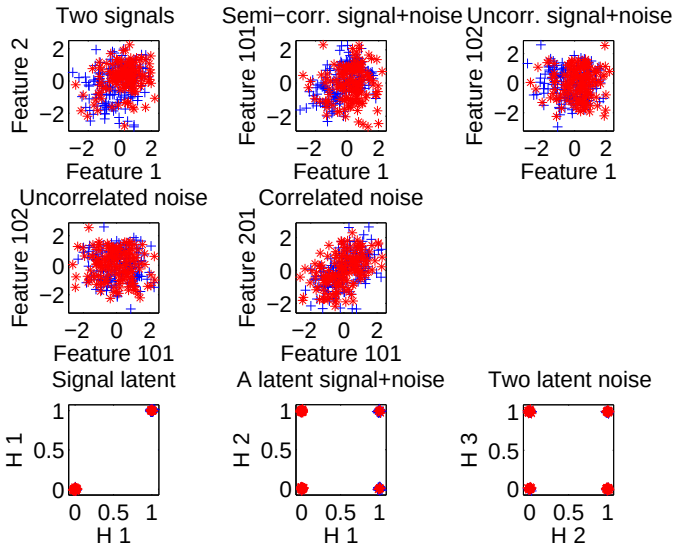


Figure 4: Scatterplots of standard model 2

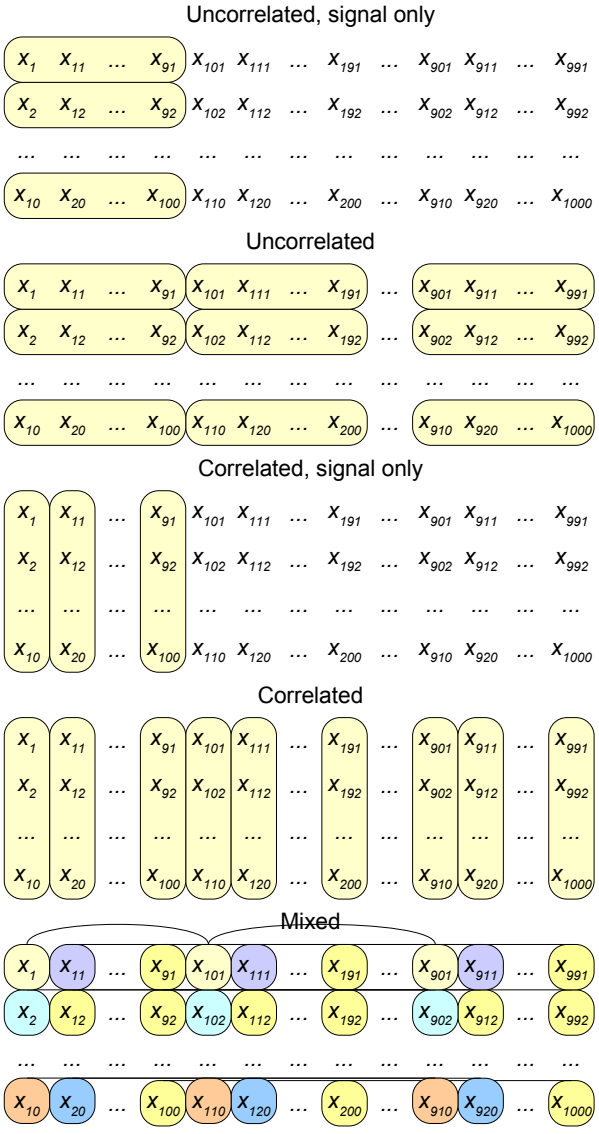


Figure 5: Gene sets

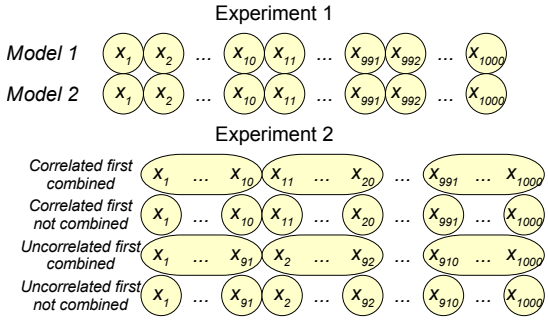


Figure 6: Experiment 1 and 2 visualized. The genesets shown here are added in the order depicted.

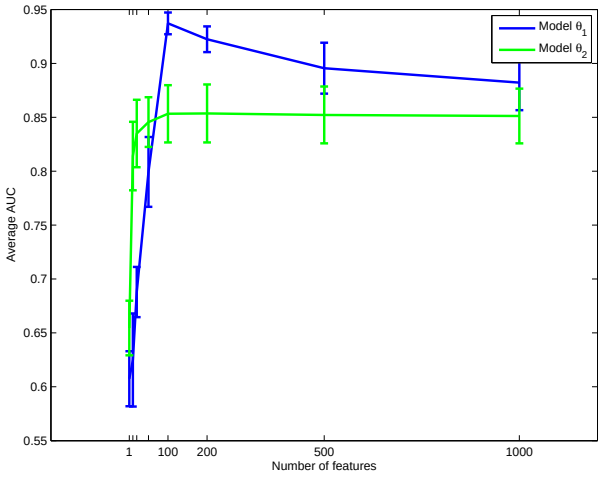


Figure 7: Model θ_1 versus model θ_2

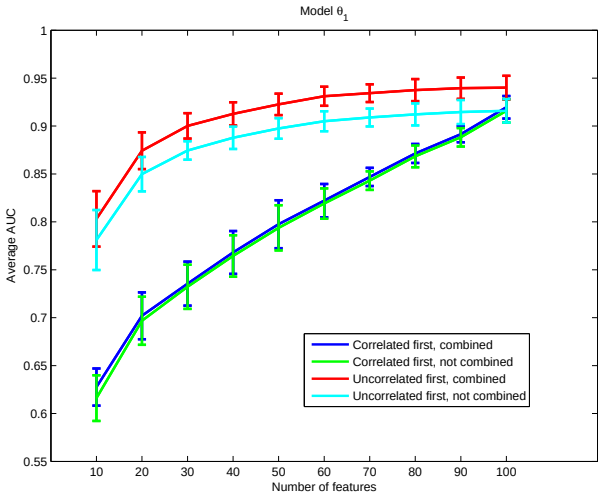


Figure 8: Correlated versus uncorrelated

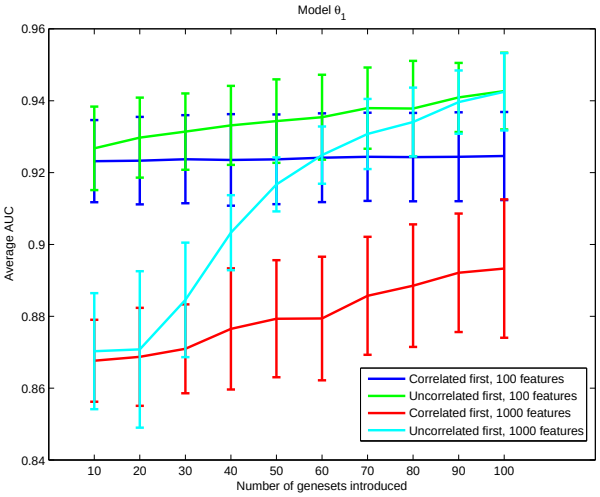


Figure 9: Correlated combined versus uncorrelated combined, but with remaining features included

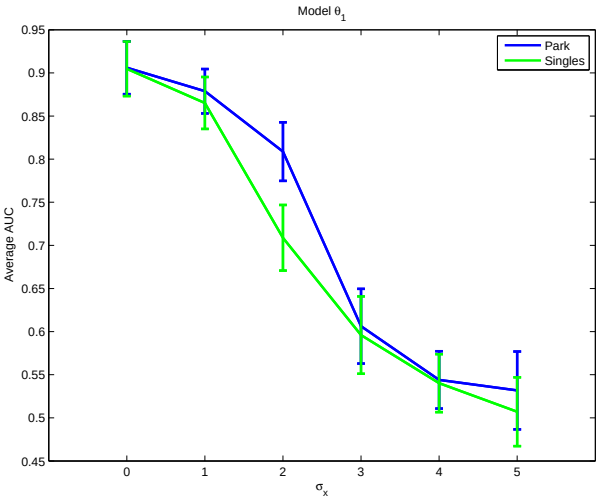


Figure 10: Model θ_1

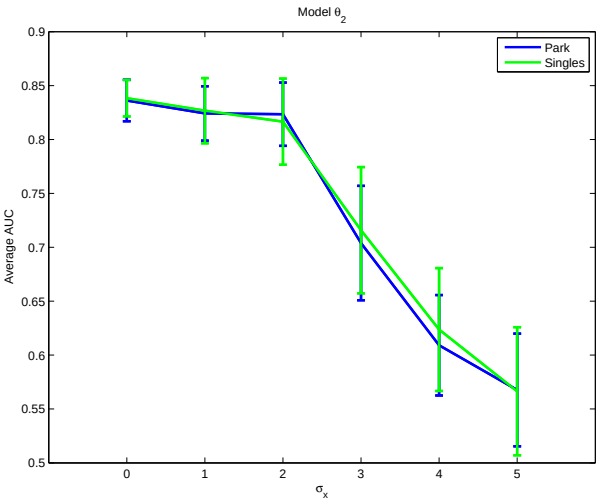


Figure 11: Model θ_2

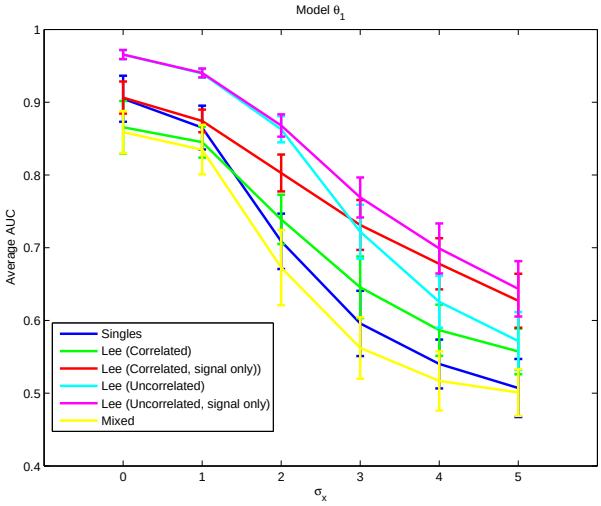


Figure 12: Model θ_1

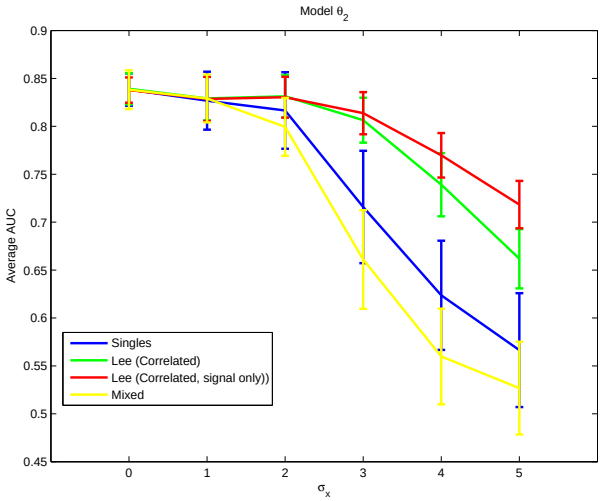


Figure 13: Model θ_2