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
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

1Information and Communication Theory Group, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands
2Bioinformatics 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
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 an intersecting set of Entrez identifiers common to both platforms, and described in van Vliet et al. All datasets except for the Wang dataset were preprocessed as algorithms, and evaluation procedure.

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

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 \( G = \{g_1, g_2, \ldots, g_p\} \). A gene expression dataset such as Vijver can be defined as \( X = \{x_{i,j}\}_{i \times n} \) with labels \( Y = \{y_1, y_2, \ldots, y_n\} \), \( y_j \in \{0, 1\} \) and genes \( G_X = \{g_1, g_2, \ldots, g_p\} \subset G \). 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 \( W = \{W_1, W_2, \ldots, W_p'\} \), where \( W_k \subset G_X, p' \) denotes the total number of predictor gene sets 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 \( X' = \{x'_{i,j}\}_{i \times n} \) is the transformed dataset such that:

\[
x'_{i,j} = \sum_{k \in W_k} \frac{x_{i,j}}{\sqrt{p_k}}
\]

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:

\[
a_k = \sum_{i \in M_k} \frac{x_{i,j}}{\sqrt{p_k}}
\]

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 $S(M_k)$:

$$ S_{MI}(M_k) = \sum_{x \in a_k} \sum_{y \in \{0, 1\}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} $$

(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 candidate 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 $P = \{g_1, g_2, ..., g_{pp}\}$, 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:

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

(4)

where $\mu_1$ and $\mu_2$ 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 a pathway $P$, its average t-score is:

$$ t_{avg} = \frac{t_i}{pp} $$

(5)

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

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 a final potential $S(M_k) = \text{Mutual Information score between } a_k \text{ and } Y$.

![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).]
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.

### 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: $M_{i,1} = \{g_1, g_2, ..., g_p\}$ and at level $i = p$, $p$ clusters are returned, each consisting of a single gene: $M_{i,1} = \{g_1\}, M_{i,2} = \{g_2\}, ..., M_{i,p} = \{g_p\}$. At level $i$, the clusters are given by $M_i = \{M_{i,1}, M_{i,2}, ..., M_{i,p}\}$. 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 $a_i = \{a_{i,1}, a_{i,2}, ..., a_{i,p}\}$.

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 $j$ 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, $X_{\text{train}}$, would perform on a second dataset, $X_{\text{test}}$, the second dataset would be transformed using the set of markers $W_{\text{train}}$ 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 \( M_i = \{M_{i1}, ..., M_{in}\} \), 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.

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 \( y \). 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, \( x'_i = f_{\text{Boolean}, i}(H) \), as is the outcome activity \( y' = f_{\text{Boolean}, y}(H) \). These discrete variables are transformed into continuous variables using two normal distributions. That is, if \( x'_i = 0 \), then \( x_i \sim N(\mu_{x'_i=0}, \sigma_{x'_i=0}^2) \), and if \( x'_i = 1 \), then \( x_i \sim N(\mu_{x'_i=1}, \sigma_{x'_i=1}^2) \). Outcome variable \( y \) is modeled similarly, but discretized similarly to the linear model such that \( y \in \{0, 1\} \). If \( y' = 0 \), then \( y \sim \text{sign}(N(\mu_{y'=0}, \sigma_{y'=0}^2)) \), and if \( y' = 1 \), then \( y \sim \text{sign}(N(\mu_{y'=1}, \sigma_{y'=1}^2)) \). Just as for the linear model, a logical model \( \theta_2 \) is designed which has a set of default values. Formally, the logical model \( \theta_2 \) is defined as:

\[
Pr(h = 0) = 0.5, \quad Pr(h = 1) = 0.5
\]

\[
y' = h_0
\]

\[
y \sim \left\{ \begin{array}{ll}
\text{sign}(N(\mu = -1, \sigma^2 = 1)) & \text{if } y' = 0 \\
\text{sign}(N(\mu = 1, \sigma^2 = 1)) & \text{if } y' = 1
\end{array} \right.
\]

\[
x'_i \sim \left\{ \begin{array}{ll}
\{ h_0 \wedge h_i & \text{if } i = 1, ..., 100 \\
h_{i-100} & \text{if } i = 101, ..., 200 \\
h_{i-200} & \text{if } i = 201, ..., 3000 \\
\vdots & \\
h_{i-900} & \text{if } i = 901, ..., 1000
\end{array} \right.
\]

\[
x_i \sim \left\{ \begin{array}{ll}
N(\mu = -1, \sigma^2 = 1) & \text{if } x'_i = 0 \\
N(\mu = 1, \sigma^2 = 1) & \text{if } x'_i = 1
\end{array} \right.
\]

The logical model is depicted in Figure 6.

2.5.2 The logical model. The logical model generates 101 latent variables \( H = \{ h_0, h_1, ..., h_{100} \} \) to work better than just using all single genes on the logical model. We expect the gene set searching algorithms to 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 \( \theta_2 \) with less noise and where \( y \) depends on fewer latent variables is shown in Figure 7.
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 $\theta_1$. 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.

$$M_1 = \{g_1, g_2, \ldots, g_{10}\}$$

$$M_2 = \{g_{11}, g_{12}, \ldots, g_{20}\}$$

...  

$$M_{100} = \{g_{991}, g_{992}, \ldots, g_{1000}\}.$$  

- **Correlated, signal only.** This set consists of 10 gene sets. Same as the ‘Correlated’ gene sets, but only the signal genes are involved.

$$M_1 = \{g_1, g_2, \ldots, g_{10}\}$$

$$M_2 = \{g_{11}, g_{12}, \ldots, g_{20}\}$$

...  

$$M_{10} = \{g_{91}, g_{92}, \ldots, g_{100}\}.$$  

- **Uncorrelated.** This set consists of 100 gene sets. In every gene set, the genes don’t depend on the same latent variable $h$.

$$M_1 = \{g_1, g_2, \ldots, g_{10}\}$$

$$M_2 = \{g_{11}, g_{12}, \ldots, g_{20}\}$$

...  

$$M_{100} = \{g_{991}, g_{992}, \ldots, g_{1000}\}.$$  

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 $\theta_1$ and $\theta_2$ 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.

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 (≈ 10000 sets) is much larger than the number of possible gene sets (≈ 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 \( \theta_1 \) and \( \theta_2 \). 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 DLCCV performance when noise features are added to the marker set. In our real data experiments, we have relaxed the restriction that only the most
A comparison of supervised gene set searching algorithms

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.

3.2.1 Experiment 1. In our first experiment, for both \( \theta_1 \) and \( \theta_2 \) 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 \( \theta_1 \) worsens, in contrast to model \( \theta_2 \). This is probably due to a better contrast in t-scores of the features in model \( \theta_2 \), 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 \( \theta_1 \) and only the signal genes, i.e. \( x_1, x_2, ..., x_{100} \).

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
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’, \( z_1, z_2, \ldots, z_{10} \), where \( z_k = \sum_{i=1}^{10} x_{(k-1) \times 10+i} / \sqrt{10} \). 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, \( x_1, x_{11}, x_2, \ldots, x_{20} \) are evaluated as single (not combining uncorrelated features) or after being combined (combining uncorrelated features) as meta-gene. In Step 2 \( x_2, x_{12}, x_3, \ldots, x_{20} \) 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 \( \{x_1, x_2, \ldots, x_{10}\}, \{x_{11}, x_{12}, \ldots, x_{20}\} \) and \( \{x_{21}, x_{22}, \ldots, x_{30}\} \) 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 \( \theta_1 \) using various values for the noise added to the features, \( \sigma_x \). 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 \( \theta_1 \) 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 (‘Uncorrelated, 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 \( \theta_2 \), 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, both models, Lee results in improvement given good sets. Park only results in a possible improvement in model \( \theta_1 \).

### 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...
A comparison of supervised gene set searching algorithms

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 $z_1 = \frac{(x_1 + x_2 + \ldots + x_{10})}{\sqrt{10}}$ 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 algorithms.

11
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 ~500 markers are found, consisting of ~500 genes, in contrast to the PPI-based method that returns ~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 Vijiyer 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 ~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


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

Raul Kooter

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 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 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

Outer Loop

- Split in 5 parts
- Repeat \( r \) of 20 repeats
- Dataset
- Test set
- Training set
- Fold \( j \) of 5-fold CV
- Inner Loop
- Validation of predictor
- AUC \( v_{r,j} \)
- Set of 100 AUC values

<table>
<thead>
<tr>
<th>Optimum predictor ( C_{n^*} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimum number of features ( n^* )</td>
</tr>
</tbody>
</table>

Train NMC on \( n^* \) features

Combined learning curve \( l_{c_{r,j}} \)

Inner Loop

- Split in 4 parts
- Repeat \( r \) of 3 repeats
- Dataset
- Test set
- Training set
- Fold \( j \) of 4-fold CV
- Validation of predictor
- AUC learning curve \( l_{c} \)
- Optimum number of features \( n^* \)
- Train NMC on \( n^* \) features
- Optimum predictor \( C_{n^*} \)

Fig. 1. Double-loop-cross-validation strategy.
Algorithm 1 Chuang’s algorithm

for Node ∈ NodesPPI do
    Subnetwork ← Node
    Score ← MI-Score(XSubnetwork, Y)
repeat
    Candidates ← Neighbours(Subnetwork, PPI) ∩ MaxDistance2(Node, PPI)
    for Candidate ∈ Candidates do
        ScoreCandidate ← MI-Score(XSubnetwork ∪ Candidate, Y)
        if ScoreCandidate > Score ∗ 1.05 then
            Subnetwork ← Subnetwork ∪ BestCandidate
        end if
    end for
until Subnetwork hasn’t changed
Subnetworks ← Subnetworks ∪ Subnetwork
end for
return Subnetworks

Algorithm 2 Lee’s algorithm

for Geneset ∈ Genesets do
    CORG ← ∅
    ScoreCORG ← 0
    for Gene ∈ Geneset do
        ScoreGene ← T-Test(XGene, Y)
    end for
    if Mean(ScoreGeneset) > 0 then
        Geneset ← OrderDescending(Geneset) [Order w.r.t. scores]
    else
        Geneset ← OrderAscending(Geneset)
    end if
    N ← 0
repeat
    N ← N + 1
    Candidates ← Genesets1...N
    ScoreCandidates ← T-Test(XCandidates, Y)
    if ScoreCandidates > ScoreCORG then
        CORG ← Candidates
        ScoreCORG ← ScoreCandidates
    end if
until CORG hasn’t changed
CORGs ← CORG ∪ CORG
end for
return CORGs

Algorithm 3 Park’s algorithm

Dendrogram ← Cluster(X) [Use average linkage]
for Level := 1 to #Features(X) do
    Genesets ← CutOff(Dendrogram, Level)
    LearningCurveLevel ← CrossValidate(XGenesets, Y) [Use area under curve]
    MinErrorLevel ← Min(LearningCurveLevel)
end for
BestLevel ← arg minLevel(MinErrorLevel)
BestGeneSets ← CutOff(Dendrogram, BestLevel)
return BestGeneSets

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.
## Supplementary Material

### DMFS markers tested on VIJVER
- Lee (C2 V1.0)
- Lee (PPI*)
- Chuang* (C2 V2.5)
- Chuang* (C2 V2.5) (T−score)
- Chuang (T−score)
- Chuang* (C2 V1.0)
- Park
- Chuang* (Park)
- Lee* (PPI)
- Lee (PPI*)
- Lee (C2 V1.0)
- Lee (Park)
- Chuang (PPI)
- Lee (C2 V2.5)
- Singles

### SOS markers tested on VIJVER
- Lee (C2 V1.0)
- Chuang* (C2 V1.0)
- Chuang* (C2 V2.5) (T−score)
- Singles
- Lee (Park)
- Chuang* (Park)
- Chuang* (C2 V2.5)
- Lee (C2 V2.5)
- Chuang (PPI)
- Chuang (T−score)
- Lee* (PPI)
- Park
- Lee (PPI*)

### DMFS and SOS markers tested on VIJVER
- Lee (C2 V1.0)
- Chuang* (C2 V1.0)
- Chuang* (C2 V2.5) (T−score)
- Singles
- Lee (Park)
- Chuang* (Park)
- Chuang* (C2 V2.5)
- Lee (C2 V2.5)
- Chuang (PPI)
- Chuang (T−score)
- Lee* (PPI)
- Park
- Lee (PPI*)

### VIJVER markers tested on DMFS
- Park
- Singles
- Lee (Park)
- Chuang* (Park)
- Chuang* (C2 V1.0)
- Chuang* (C2 V2.5)
- Chuang (T−score)
- Chuang (PPI)
- Lee (C2 V1.0)
- Lee (C2 V2.5)
- Lee* (PPI)
- Lee (PPI*)

### SOS markers tested on DMFS
- Lee* (PPI)
- Chuang* (C2 V1.0)
- Lee (C2 V2.5)
- Chuang* (C2 V2.5) (T−score)
- Singles
- Park
- Lee (PPI*)

### VIJVER and SOS markers tested on DMFS
- Park
- Singles
- Lee (Park)
- Chuang* (Park)
- Chuang* (C2 V2.5) (T−score)
- non-

### VIJVER markers tested on SOS
- Park
- Singles
- Lee (Park)
- Chuang* (Park)
- Chuang* (C2 V1.0)
- Chuang* (C2 V2.5)
- Chuang (T−score)
- Chuang (PPI)
- Lee (C2 V1.0)
- Lee (C2 V2.5)
- Lee* (PPI)
- Lee (PPI*)

### DMFS markers tested on SOS
- Lee* (PPI)
- Chuang* (C2 V1.0)
- Lee (C2 V2.5)
- Chuang* (C2 V2.5) (T−score)
- Singles
- Park
- Lee (PPI*)

### VIJVER and DMFS markers tested on SOS
- Park
- Singles
- Lee (Park)
- Chuang* (Park)
- Chuang* (C2 V2.5) (T−score)
- Lee* (PPI)
- Lee (PPI*)
- Lee (C2 V2.5)
- Chuang* (C2 V2.5)
- Chuang (T−score)
- Chuang (PPI)
- Lee (C2 V1.0)
- Lee (PPI*)

---

**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 the 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.
## 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.
<table>
<thead>
<tr>
<th>Table 2. Top 10 subnetworks. Markers from VIJVER, tested on DMFS.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test</strong></td>
<td><strong>Markers</strong></td>
</tr>
<tr>
<td><strong>DMFS</strong> VIJVER</td>
<td>MADD2, MADD1, BUB1, BUB3, PHAG1, CENPA</td>
</tr>
<tr>
<td><strong>DMFS</strong> VIJVER</td>
<td>WEE1, CDCA5, CSN2, PASK1, VYHAB, MAP2K1, C2C12, C2C12</td>
</tr>
<tr>
<td><strong>DMFS</strong> VIJVER</td>
<td>FLJ10259, BAD39R, BAD39, PIMM2, LSM11, UPF2</td>
</tr>
<tr>
<td><strong>DMFS</strong> VIJVER</td>
<td>BVK1, TRNAP, TRNAP, KIAA0494, PREC1</td>
</tr>
<tr>
<td><strong>DMFS</strong> VIJVER</td>
<td>SK2, K-ALPHA-1, PKR, C2C2, DSG</td>
</tr>
<tr>
<td><strong>DMFS</strong> VIJVER</td>
<td>PKS1, TPS2, AURKA, F51, COL1A1, MASP9, LGA103SP, COL1A1, CTPBP4, NFAT1</td>
</tr>
<tr>
<td><strong>DMFS</strong> VIJVER</td>
<td>K-ALPHA-1, CT1C, TREC1, TDG1</td>
</tr>
<tr>
<td><strong>DMFS</strong> VIJVER</td>
<td>LPS1, CT1C, TREC1, TDG1</td>
</tr>
<tr>
<td><strong>DMFS</strong> VIJVER</td>
<td>BAK2, BURI, BURI</td>
</tr>
<tr>
<td><strong>DMFS</strong> VIJVER</td>
<td>EPFR2, AKR2AP5, ATTN, ANK2P1, GRR2, PED1, KNU2, ACTB3</td>
</tr>
</tbody>
</table>
Table 4. Top 10 subnetworks. Markers from VIJVER, tested on SOS.

<table>
<thead>
<tr>
<th>Subnetwork</th>
<th>Markers</th>
<th>PPI Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMFS</td>
<td>SHIP2, MYL2, RAI2, MELK, CD44, NCOR2</td>
<td>1.0</td>
</tr>
<tr>
<td>DMFS</td>
<td>SHIP2, MYL2, RAI2, MELK, CD44, NCOR2</td>
<td>1.0</td>
</tr>
<tr>
<td>DMFS</td>
<td>SHIP2, MYL2, RAI2, MELK, CD44, NCOR2</td>
<td>1.0</td>
</tr>
<tr>
<td>DMFS</td>
<td>SHIP2, MYL2, RAI2, MELK, CD44, NCOR2</td>
<td>1.0</td>
</tr>
<tr>
<td>DMFS</td>
<td>SHIP2, MYL2, RAI2, MELK, CD44, NCOR2</td>
<td>1.0</td>
</tr>
<tr>
<td>DMFS</td>
<td>SHIP2, MYL2, RAI2, MELK, CD44, NCOR2</td>
<td>1.0</td>
</tr>
<tr>
<td>DMFS</td>
<td>SHIP2, MYL2, RAI2, MELK, CD44, NCOR2</td>
<td>1.0</td>
</tr>
<tr>
<td>DMFS</td>
<td>SHIP2, MYL2, RAI2, MELK, CD44, NCOR2</td>
<td>1.0</td>
</tr>
<tr>
<td>DMFS</td>
<td>SHIP2, MYL2, RAI2, MELK, CD44, NCOR2</td>
<td>1.0</td>
</tr>
<tr>
<td>DMFS</td>
<td>SHIP2, MYL2, RAI2, MELK, CD44, NCOR2</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Table 5. Top 10 subnetworks. Markers from DMFS, tested on VIJVER.

<table>
<thead>
<tr>
<th>Subnetwork</th>
<th>Markers</th>
<th>Topology</th>
<th></th>
<th>Probes</th>
<th></th>
<th>Probes</th>
<th></th>
<th>Probes</th>
<th></th>
<th>Probes</th>
<th></th>
<th>Probes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMFS</td>
<td>SLCA2A1</td>
<td>CCNA2 CDC73 CDCN1A CDCN1 BIRC5</td>
<td>PKMYT1 FFRK</td>
<td>E2F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>CCNA2 CDCN1A CDCN1B</td>
<td>PKMYT1 FFRK</td>
<td>E2F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>POLD1 POLD2</td>
<td>PKMYT1 FFRK</td>
<td>E2F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>POLD1 POLD2</td>
<td>PKMYT1 FFRK</td>
<td>E2F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>POLD1 POLD2</td>
<td>PKMYT1 FFRK</td>
<td>E2F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Subnetwork</th>
<th>Markers</th>
<th>Topology</th>
<th></th>
<th>Probes</th>
<th></th>
<th>Probes</th>
<th></th>
<th>Probes</th>
<th></th>
<th>Probes</th>
<th></th>
<th>Probes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOS</td>
<td>POLD1 POLD2</td>
<td>PKMYT1 FFRK</td>
<td>E2F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS</td>
<td>POLD1 POLD2</td>
<td>PKMYT1 FFRK</td>
<td>E2F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS</td>
<td>POLD1 POLD2</td>
<td>PKMYT1 FFRK</td>
<td>E2F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS</td>
<td>POLD1 POLD2</td>
<td>PKMYT1 FFRK</td>
<td>E2F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Supplementary Material
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
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.

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

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.
<table>
<thead>
<tr>
<th>Vijver genes</th>
<th>MI Score</th>
<th>T-test</th>
<th>Wang genes</th>
<th>MI Score</th>
<th>T-test</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASPA, ONECUT1, E2F1, FLJ10415, PRK1, POSN, ELOVL1, BM039, MAP2K5, FLJ11029, CRADD</td>
<td>0.14907</td>
<td>-6.9152</td>
<td>DUSP5, MAPK1, COPS5, MAPK9, NER2, MADH3, HEY1, PSMC6, SHC1, ITGAV, RP56K4A, VAV1, CDK2, CDK5, MYPT2</td>
<td>0.16126</td>
<td>-9.3513</td>
</tr>
<tr>
<td>SRC, MATK, EWSN1, ADRBK1, RAD23A, GFAP, MUL, VEGF, YWHAB, VTN</td>
<td>0.1365</td>
<td>-6.1917</td>
<td>NHP2L1, HNRPH2, SNRPA, NCL, GNB2L1, TOP1, POLR2E, SNRP70</td>
<td>0.141</td>
<td>7.8263</td>
</tr>
<tr>
<td>MAN2A2, IKBKAP, KIAA0808, PLP2, ITGA5, GIT1, ADRBK1, PPFIA2, HMOX2, XIP, PXN, PFDNI</td>
<td>0.13392</td>
<td>-7.1414</td>
<td>SNRPA, NHP2L1, HNRPH2, NCL, GNB2L1, TOP1, POLR2E, SNRP70</td>
<td>0.141</td>
<td>7.8263</td>
</tr>
<tr>
<td>PTN, CFP2, PSM2D1, NR4A3, PSM2, P4HB, SLC20A1, CDC6</td>
<td>0.1325</td>
<td>-6.3552</td>
<td>GTF2A1, POLR2E, NHP2L1, HNRPH2, NCL, TOP1, RP55</td>
<td>0.12621</td>
<td>7.5469</td>
</tr>
<tr>
<td>KITLG, EPOR, STAT5A, USP4, FOS, RPL4, MYB, JAK2, RB1, TGFBI1, RPS9</td>
<td>0.1319</td>
<td>-7.045</td>
<td>SNRPA, NHP2L1, CD69, EIF4G1, IFNG, POLR2A, NLI-IF, NONO, FITGB2, GTF2A1</td>
<td>0.12197</td>
<td>7.5406</td>
</tr>
<tr>
<td>APCS, ONECUT1, DKFZP27M231, E2F1, FLJ11029, AKR1C4, ELOVL1, ACPP, NCOA3, HBOA</td>
<td>0.13096</td>
<td>-5.9929</td>
<td>NCL, GNB2L1, NR3C1, SYT1, IFNAR1, TOP1, STAT3, STAT4, RPL6, SP1, TAT, SMARCE1</td>
<td>0.12435</td>
<td>7.4034</td>
</tr>
<tr>
<td>HMMR, MAPK3, GIFT1, HS1, MAPK9, STK3, DKFZP434D156, AKT1, DAPK1, SHC1, PTPR1, MAPK12</td>
<td>0.13055</td>
<td>-5.8541</td>
<td>STAT5A, MYC, TPAP2B, CSF1, USF2, NMI, IL4</td>
<td>0.11996</td>
<td>5.5416</td>
</tr>
<tr>
<td>TNFRSF10C, YWHAZ, LTBH, CDC25B, RGS3, HDAC5</td>
<td>0.12954</td>
<td>-5.8945</td>
<td>PPLA1, PPF3CA, RB1, MAPK9, ASGR2, TMSG1, P84, ABL1, RFP, DKFZP5643J157</td>
<td>0.11659</td>
<td>-3.398</td>
</tr>
<tr>
<td>HFKR, ONECUT1, E2F1, BCR, DRAK1, PCAF, AKR1C4, 7-60, OAZ2, APOH, PRK1</td>
<td>0.1283</td>
<td>-5.8083</td>
<td>HNRPH2, NHP2L1, SNRPA, NCL, GNB2L1, PRKCA</td>
<td>0.11446</td>
<td>6.7924</td>
</tr>
<tr>
<td>PRKDC, HS1, GTF2A2, POLR2C, SUPT5H, TREX1, HCNP</td>
<td>0.1262</td>
<td>-5.4746</td>
<td>ALK, SHC1, INPP11, GHR, PSMC6, KRA52, NRAS, P85PR, SNT1, PRK3CA</td>
<td>0.11071</td>
<td>-6.9089</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Vijver genes</th>
<th>MI Score</th>
<th>T-test</th>
<th>Wang genes</th>
<th>MI Score</th>
<th>T-test</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRC1</td>
<td>0.065814</td>
<td>-6.1666</td>
<td>NR0B1</td>
<td>0.048555</td>
<td>2.9245</td>
</tr>
<tr>
<td>DDXL</td>
<td>0.064802</td>
<td>-5.5205</td>
<td>RAF1</td>
<td>0.046755</td>
<td>1.8241</td>
</tr>
<tr>
<td>DEKPEST</td>
<td>0.061376</td>
<td>-5.4321</td>
<td>NH2P2L1</td>
<td>0.043553</td>
<td>3.8031</td>
</tr>
<tr>
<td>E2F1</td>
<td>0.061079</td>
<td>-5.9182</td>
<td>FLJ19998</td>
<td>0.042217</td>
<td>3.7653</td>
</tr>
<tr>
<td>IL7R</td>
<td>0.058636</td>
<td>-5.665</td>
<td>EBI2</td>
<td>0.040824</td>
<td>0.4984</td>
</tr>
<tr>
<td>KIAA0165</td>
<td>0.057702</td>
<td>-6.65</td>
<td>KIAA0010</td>
<td>0.040279</td>
<td>-3.6297</td>
</tr>
<tr>
<td>T1K</td>
<td>0.057424</td>
<td>-6.125</td>
<td>GPRK8</td>
<td>0.040112</td>
<td>-7.2989</td>
</tr>
<tr>
<td>PGR</td>
<td>0.056666</td>
<td>-5.55</td>
<td>GCP56</td>
<td>0.039936</td>
<td>-3.9462</td>
</tr>
<tr>
<td>BYSL</td>
<td>0.054867</td>
<td>-6.8599</td>
<td>FBP2</td>
<td>0.038862</td>
<td>3.5767</td>
</tr>
<tr>
<td>ODC1</td>
<td>0.053988</td>
<td>-1.4628</td>
<td>ID-GAP</td>
<td>0.037586</td>
<td>-4.4310</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Vijver genes</th>
<th>MI Score</th>
<th>T-test</th>
<th>Wang genes</th>
<th>MI Score</th>
<th>T-test</th>
</tr>
</thead>
<tbody>
<tr>
<td>BHMT, HNF4A, CDC45L, HSPC164, TEAD3, ATF7, L2DLT, HAL, TRPC5, TXNRD1, FLJ10415, POLD4, MAP2K5</td>
<td>0.18343</td>
<td>-9.2905</td>
<td>FNTB, HNF4A, DKFPZp56601646, LOC51631, LOC51707, LIMS1, HSPC160, CDK5, pcap, FLJ10640, SIX2, LOC51096</td>
<td>0.18285</td>
<td>-10.4192</td>
</tr>
<tr>
<td>SLC17A2, HNF4A, FLJ1912, PSMD7, LOC51142, NRAC2, LNR, ADH6, DCK, LOC50834</td>
<td>0.17649</td>
<td>-4.8129</td>
<td>STAM2, HNF4A, HECH, LOC5109, SPOCK, FLJ10460, FLJ10511, HSPC166, DKFPZp434E2220, MGST3</td>
<td>0.17843</td>
<td>-8.7979</td>
</tr>
<tr>
<td>DSCR3, HNF4A, EXO1, PPP6, TRPC5, AF093680, G20RF1, SLPI, NDRG1, COX7A2, GRO3</td>
<td>0.17034</td>
<td>-5.8147</td>
<td>HECH, HNF4A, STAM2, LOC5109, SPOCK, FLJ10460, FLJ10511, HSPC166, DKFPZp434E2220, MGST3</td>
<td>0.17843</td>
<td>-8.7979</td>
</tr>
<tr>
<td>FCYT1A, HNF4A, PSMD7, L2DLT, GKO01, BCS1, GABRE, B3GAT1</td>
<td>0.16811</td>
<td>-6.3851</td>
<td>IRF6, HNF4A, SRFp4, pcap, LIMS1, I326, HECH, MSMB, AKRI1C2</td>
<td>0.17005</td>
<td>-8.0761</td>
</tr>
<tr>
<td>YRT6, HNF4A, EXO1, LOC55972, FLJ30619, RAB2, RIP60, GNG5, TDRKH, FLJ10412, NR2C2</td>
<td>0.16423</td>
<td>-4.125</td>
<td>NPEZL1, HNF4A, DKFPZp434E2220, HECH, STAM2, JM35, CDK5, ACTA2, PMS3, AKRI1C2</td>
<td>0.16579</td>
<td>-7.0062</td>
</tr>
<tr>
<td>TADA3L, USP5, HNF4A, EXO1, CTSZ, CDC45L, SFRP5, ATFF7, TAF2E, CPT2</td>
<td>0.15974</td>
<td>-7.1358</td>
<td>KIAA1226, HNF4A, STAM2, LOC51096, NR4A, TUFT1, pcap, ERO1L, LOC55107, LOC51644, KLRF1, HSU79274</td>
<td>0.16432</td>
<td>-9.3748</td>
</tr>
<tr>
<td>HAAO, HNF4A, RAD21, TRP5, GSK3B, NSEP1, LSM3, NR5A2</td>
<td>0.15899</td>
<td>-5.6782</td>
<td>HNF4, HNF4A, STAM2, HECH, HSPC164, HBSIL, DKFPZp434E2220, GCHFR, NDRG1</td>
<td>0.16692</td>
<td>-9.0979</td>
</tr>
<tr>
<td>ATP10C, HNF4A, PSM7, FEN1, PSMD1, CTSZ, FLJ20619, SUDD, SF3B4</td>
<td>0.15888</td>
<td>-5.8826</td>
<td>DUSP3, MAPK1, COPS8, MAPK9, NEK2, MADH13, HEY1, PSMC6, SHC1, ITGA, RPSK6AK2, AV1, CDK2, CDK5, MYPT2</td>
<td>0.16126</td>
<td>-9.3513</td>
</tr>
<tr>
<td>HSPC072, HNF4A, ASPG2, SREBF2, SCVE1, GSK3B, HSPC160, LRP5, MDFI, UXT, C14ORF1</td>
<td>0.15744</td>
<td>-3.683</td>
<td>HADH2, HNF4A, STAM2, LIMS1, LOC51065, RPSK6KC1, HEY1, KIAA0141, DPM2</td>
<td>0.16112</td>
<td>-7.5496</td>
</tr>
<tr>
<td>MOCS3, MOCS2, HNF4A, NDRG1, TARS, EIF4G1, PSMD7, MRPL4A, EXO1, MTHFR, KIAA0477, PZP</td>
<td>0.15718</td>
<td>-6.635</td>
<td>MGAT3AB, HNF4A, HECH, KRT10, LIMS1, SRP54, VSP54, STAM2, ACTA2, 54TM, VDTH, SSBP</td>
<td>0.16104</td>
<td>-9.1583</td>
</tr>
</tbody>
</table>
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.

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.

<table>
<thead>
<tr>
<th>Input</th>
<th>Transformed using</th>
<th>Error</th>
<th>Optimal feats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang</td>
<td>Vijver significant subnetworks</td>
<td>0.4423</td>
<td>18</td>
</tr>
<tr>
<td>Wang</td>
<td>Vijver top 660 genes</td>
<td>0.4178</td>
<td>10</td>
</tr>
<tr>
<td>Wang</td>
<td>Wang significant subnetworks</td>
<td>0.1642</td>
<td>12</td>
</tr>
<tr>
<td>Wang</td>
<td>Wang top 878 genes</td>
<td>0.2819</td>
<td>17</td>
</tr>
<tr>
<td>Vijver</td>
<td>Wang significant subnetworks</td>
<td>0.3514</td>
<td>21</td>
</tr>
<tr>
<td>Vijver</td>
<td>Wang top 878 genes</td>
<td>0.3748</td>
<td>23</td>
</tr>
<tr>
<td>Vijver</td>
<td>Vijver significant subnetworks</td>
<td>0.2061</td>
<td>20</td>
</tr>
<tr>
<td>Vijver</td>
<td>Vijver top 660 genes</td>
<td>0.2847</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 5: Classification performance
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.

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:

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.
February 5\textsuperscript{th}

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:
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, Logistic and NMC.

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, Logistic 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 
\[(\text{genevector}_2 + \text{genevector}_3 + \text{genevector}_4)/\sqrt{3}\].

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>
<thead>
<tr>
<th>Pathway Name</th>
<th>Frequency</th>
<th># genes</th>
<th>CORGs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclin regulated genes</td>
<td>416/500</td>
<td>2/13</td>
<td>E2F1 CCNE1</td>
</tr>
<tr>
<td>IL7 pathway</td>
<td>200/500</td>
<td>3/16</td>
<td>BCL2 STAT5A IL7</td>
</tr>
<tr>
<td>Cell cycle</td>
<td>197/500</td>
<td>3/84</td>
<td>E2F1 ESPL1 CCNB2</td>
</tr>
<tr>
<td>ActinY pathway</td>
<td>142/500</td>
<td>3/19</td>
<td>PIR PSMA7 ACTR3</td>
</tr>
<tr>
<td>GNF female genes</td>
<td>123/500</td>
<td>3/85</td>
<td>RPS4X RPS6 RPL6</td>
</tr>
</tbody>
</table>

**From Netherlands to USA**

- Cycinin regulated genes
- IL7 pathway
- Cell cycle
- ActinY pathway
- GNF female genes

**From USA to Netherlands**

- Cell Cycle
- Breast cell cycle
- Cyclin regulated genes
- KRAS up-regulated genes
- Cell cycle checkpoint II genes
- Glutamine down-regulated genes
- MMP/Cytokine connection
- Lactate down-regulated genes
- IL22 pathway
- Rapamycin down-regulated genes

*The number of CORGs and member genes are specified.

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

**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.
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.)
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.](image)

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.](image)
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:

<table>
<thead>
<tr>
<th>Label overlap</th>
<th>Genes overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2 v1.0</td>
<td>0.0297</td>
</tr>
<tr>
<td>C2 v2.5</td>
<td>0.1489</td>
</tr>
<tr>
<td>C2 v1.0 (all)</td>
<td>1.0000</td>
</tr>
<tr>
<td>C2 v2.5 (all)</td>
<td>1.0000</td>
</tr>
<tr>
<td>Top 906/618 genes</td>
<td>0.0687</td>
</tr>
<tr>
<td>Chuang</td>
<td>0.0182</td>
</tr>
<tr>
<td>Raul1</td>
<td>0.0739</td>
</tr>
</tbody>
</table>

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.](image)

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.

![Distribution of features per subnetwork using Vyper](image)

![Distribution of features per subnetwork using Wang](image)

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

And the performance...

![Performance of the aucs with the 500 clusters](image)

**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.
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](image)

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

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.

![Classification results, tested on Vijver](image1)

![Classification results, tested on Wang](image2)

![Classification results, tested on DMFS](image3)

![Classification results, tested on SOS](image4)

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 whether 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.

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 expression \( y = (g_1 \text{ OR } g_2) \text{ AND } g_3 \), 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.

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 \( y = f(h) \).

Figure 6: Four basic models in increasing complexity.
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.

![Graph 1](image1)

![Graph 2](image2)

![Graph 3](image3)

![Graph 4](image4)

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 performances climb above 0.7.
- Fisher classifier performs worse when many features are added. NMC seems to be almost immune to this effect.
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
\[ H_i = \text{Normal}(0,3) \]
\[ X_i = \text{Normal}(H_i,3) \]
\[ Y = \text{Normal}(\text{Beta} \times H, 3) \]
\[ \text{Beta} = \text{Normal}(2,3) \]
250 Runs
100 training samples, 100 testing samples
200 features (genes)

In the example run, we see an improvement in classification performance can be gained 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

a) single genes model

b) complex model

c) logics effects model

d) influence model

e) combined model

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)

\[ H_i = \text{Normal(mean}=0, \text{var}=5), \text{for } i=1..n \]
\[ X_i = H_i + \text{Normal(mean}=0, \text{var}=2), \text{for } i=1..n \]
\[ Y = BETA \ast H + \text{Normal(mean}=0, \text{var}=2) \]
\[ BETA = \{1,1,1,..1\} \]
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 upper right is an example distribution of two genes and their outcome of one run. The outcome labels are colored red and green. The lower left graph shows a heatmap of one example run. The heatmap also contains an outcome column. The lower right 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)

\[ H = \text{Normal}(\text{mean}=0, \text{var}=2) \]
\[ BETA = \{1, 1, 1\ldots1\} \]
\[ X_i = BETA*H_i + \text{Normal}(\text{mean}=0, \text{var}=2), \text{ for } i=1..n \]
\[ Y = H + \text{Normal}(\text{mean}=0, \text{var}=2) \]

1.2.2 Example run 2

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

\[ H = \text{Normal}(\text{mean}=0, \text{var}=2) \]
\[ BETA = \{1, 2, 1, 2, 1, 2\ldots2\} \]
\[ X_i = BETA*H_i + \text{Normal}(\text{mean}=0, \text{var}=2), \text{ for } i=1..n \]
\[ Y = H + \text{Normal}(\text{mean}=0, \text{var}=2) \]
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 4: Model B, example 2

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)

\[ H_i = \text{Normal}(\text{mean}=0, \text{var}=2), \text{ for } i=1..n \]

\[ X_i = H_i + \text{Normal}(\text{mean}=0, \text{var}=2), \text{ for } i=1..n \]

\[ Y = \text{min}(H_i) + \text{Normal}(\text{mean}=0, \text{var}=2) \]

![Histograms of error distribution](image1)

![Heat Map](image2)

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

\[ Y = c + H \cdot A + e \]

\[ H = \text{normal}(\text{mean}=0, \text{var}=v) \]

\[ X = c + H \cdot B + e \]

Figure 1: Initial experimental setup.

\[ H = \text{normal}(\text{mean}=0, \text{var}=1) \]

\[ X = H_x + \text{normal}(\text{mean}=0, \text{var}=1) \]

\[ X_{\text{noise}} = \text{normal}(\text{mean}=0, \text{var}=9) \]

\[ Y = -H_1 + H_2 + 2 \cdot H_3 + \text{normal}(\text{mean}=0, \text{var}=9) \]

\[ \text{Outcome} = 1 \text{ if } Y \text{ above 0, 0 if } Y \text{ under 0.} \]

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.

<table>
<thead>
<tr>
<th>Cutoff</th>
<th>Features</th>
<th>Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>4</td>
<td>[7 8 9] [4 5 6] [10] [1 2 3]</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>[7 8 9] [1 2 3]</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>[7 8 9] [1 2 3]</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>[7 8 9]</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
<td>[7 8] [1 3] [9]</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>[7 8 9] [1 2 3] [10] [4 5 6]</td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>[7] [8 9] [4 6] [5] [1 3]</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>[7 8 9] [4 5 6] [14] [15] [1 2 3]</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>[7 8 9] [4 5 6] [1 2 3] [12] [16] [10]</td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>[7 9] [8 2] [1]</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>[4 5] [7 8 9] [6] [17] [1 2 3] [14]</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>[7 8 9] [1 2 3]</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>[7 8 9] [1 2 3] [4 5 6]</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>[7 8 9]</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Cutoff</th>
<th>Features</th>
<th>Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>8</td>
<td>[7 8 9] [1 2 3] [4 5 6] [12] [17] [18] [10]</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>[7 8 9] [1 2 3] [4 5 6]</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>[7 8 9] [4 5 6] [1 2 3]</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>[7 8 9] [1 2 3] [4 5 6]</td>
</tr>
</tbody>
</table>

1.2 Parameters analysis

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

Once again, here is the basic setup:

\[
\begin{align*}
H &= \text{normal}(\text{mean}=0, \text{var}=1). \\
X &= H_x + \text{normal}(\text{mean}=0, \text{var}=1). \\
X_{\text{noise}} &= \text{normal}(\text{mean}=0, \text{var}=9). \\
Y &= -H_1 + H_2 + 2 \times H_3 + \text{normal}(\text{mean}=0, \text{var}=9). \\
\text{Outcome} &= 1 \text{ if } Y \text{ above 0, 0 if } Y \text{ under 0.}
\end{align*}
\]

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 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

![Simulation setup diagram](image)

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.

<table>
<thead>
<tr>
<th>$H_1$</th>
<th>$H_2$</th>
<th>$X_{1,1}$</th>
<th>$X_{1,2}$</th>
<th>$X_{2,1}$</th>
<th>$X_e$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>$\frac{1}{1+e_x}$</td>
<td>$\frac{1}{1+e_x}$</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{n}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{1+e_x}$</td>
<td>0</td>
<td>$\frac{1}{n(1+e_x)}$</td>
</tr>
<tr>
<td>$X_{1,1}$</td>
<td>0</td>
<td>$\frac{1}{1+e_x}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{n(1+e_x)}$</td>
</tr>
<tr>
<td>$X_{1,2}$</td>
<td>0</td>
<td>$\frac{1}{1+e_x}$</td>
<td>$\frac{1}{1+e_x}$</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{n(1+e_x)}$</td>
</tr>
<tr>
<td>$X_e$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$Y$</td>
<td>$\frac{1}{n}$</td>
<td>$\frac{1}{n(1+e_x)}$</td>
<td>$\frac{1}{n(1+e_x)}$</td>
<td>$\frac{1}{n(1+e_x)}$</td>
<td>0</td>
<td>$\frac{1}{n} + e_y$</td>
</tr>
</tbody>
</table>

1.3 Methods used

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

![Workflow diagram](image)

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.

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 theoretically, 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.

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 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 conclusion 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.)

1 Change log

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:

<table>
<thead>
<tr>
<th></th>
<th>$H_1$</th>
<th>$H_2$</th>
<th>$H_{noise}$</th>
<th>$X_{1,1}$</th>
<th>$X_{1,2}$</th>
<th>$X_{2,1}$</th>
<th>$X_{noise}$</th>
<th>...</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{\sqrt{1+e_y}}$</td>
<td>$\frac{1}{\sqrt{1+e_y}}$</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>$\frac{1}{h_{relevant}}$</td>
</tr>
<tr>
<td>$H_{noise}$</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{\sqrt{1+e_y}}$</td>
<td>0</td>
<td>...</td>
<td>$\frac{1}{h_{relevant}}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$X_{1,1}$</td>
<td>...</td>
<td>...</td>
<td>1</td>
<td>$\frac{1}{1+e_y}$</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>$\frac{1}{h_{relevant}(1+e_y)}$</td>
<td></td>
</tr>
<tr>
<td>$X_{1,2}$</td>
<td>...</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>$\frac{1}{h_{relevant}(1+e_y)}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_{2,1}$</td>
<td>...</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>$\frac{1}{h_{relevant}(1+e_y)}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$X_{noise}$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$Y$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>$\frac{1}{h_{relevant}} + e_y$</td>
</tr>
</tbody>
</table>

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 Chiang-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.
1.2.3 LeeRandom

LeeRandom is Lee’s method using randomized subnetworks. Subnetworks were randomized by randomly switching the features between the different 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 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 result 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.

<table>
<thead>
<tr>
<th></th>
<th>$H_1$</th>
<th>$H_2$</th>
<th>$H_{\text{noise}}$</th>
<th>$X_{1,1}$</th>
<th>$X_{1,2}$</th>
<th>$X_{2,1}$</th>
<th>$X_{\text{noise}}$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_1$</td>
<td>$\frac{1}{n_H}$</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>$\frac{1}{n_H}$</td>
<td>$\frac{1}{n_H}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$H_2$</td>
<td>$\frac{1}{n_H}$</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{n_H}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$H_{\text{noise}}$</td>
<td>$\frac{1}{n_H}$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$X_{1,1}$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$X_{1,2}$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>$X_{2,1}$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>$\frac{1}{n_H \cdot \text{relevant}}$</td>
</tr>
<tr>
<td>$X_{\text{noise}}$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>$Y$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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.
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 an independent dataset. a) Number of hidden nodes = 1, signal error = 0. b) Number of hidden nodes = 1, signal error = 9. c) Number of hidden nodes = 3, signal error = 1. d) Number of hidden nodes = 7, signal error = 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.

![Table of paired t-score vs ranksum comparisons]

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.
<table>
<thead>
<tr>
<th></th>
<th>Chuang* (C2V2.5)</th>
<th>Lee (Park)</th>
<th>Lee (Permuted C2V2.5)</th>
<th>Singles</th>
<th>Lee (KEGG)</th>
<th>Chuang (Permuted PPI)</th>
<th>Lee (C2V1.0)</th>
<th>Chuang (T-score)</th>
<th>Lee (GO)</th>
<th>Chuang (Permuted PPI)</th>
<th>Lee* (PPI)</th>
<th>Park</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0.45</td>
<td>0.45</td>
<td>0.36</td>
<td>0.29</td>
<td>0.2</td>
<td>0.19</td>
<td>0.15</td>
<td>0.047</td>
<td>0.016</td>
<td>0.014</td>
<td>0.0096</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.55</td>
<td>1</td>
<td>0.47</td>
<td>0.39</td>
<td>0.32</td>
<td>0.24</td>
<td>0.2</td>
<td>0.16</td>
<td>0.053</td>
<td>0.014</td>
<td>0.0096</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.55</td>
<td>0.53</td>
<td>1</td>
<td>0.39</td>
<td>0.32</td>
<td>0.24</td>
<td>0.2</td>
<td>0.17</td>
<td>0.052</td>
<td>0.015</td>
<td>0.0096</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.64</td>
<td>0.61</td>
<td>0.61</td>
<td>1</td>
<td>0.43</td>
<td>0.32</td>
<td>0.31</td>
<td>0.24</td>
<td>0.11</td>
<td>0.031</td>
<td>0.0065</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.71</td>
<td>0.68</td>
<td>0.68</td>
<td>0.57</td>
<td>1</td>
<td>0.38</td>
<td>0.4</td>
<td>0.3</td>
<td>0.15</td>
<td>0.061</td>
<td>0.0042</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8</td>
<td>0.76</td>
<td>0.76</td>
<td>0.68</td>
<td>0.62</td>
<td>1</td>
<td>0.5</td>
<td>0.43</td>
<td>0.22</td>
<td>0.1</td>
<td>0.0035</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.81</td>
<td>0.8</td>
<td>0.8</td>
<td>0.69</td>
<td>0.6</td>
<td>0.5</td>
<td>1</td>
<td>0.42</td>
<td>0.24</td>
<td>0.11</td>
<td>0.0076</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.86</td>
<td>0.84</td>
<td>0.83</td>
<td>0.76</td>
<td>0.7</td>
<td>0.57</td>
<td>0.56</td>
<td>1</td>
<td>0.29</td>
<td>0.15</td>
<td>0.046</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.95</td>
<td>0.95</td>
<td>0.94</td>
<td>0.89</td>
<td>0.85</td>
<td>0.76</td>
<td>0.76</td>
<td>0.71</td>
<td>1</td>
<td>0.32</td>
<td>0.28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
<td>0.94</td>
<td>0.94</td>
<td>0.88</td>
<td>0.89</td>
<td>0.85</td>
<td>0.68</td>
<td>1</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.97</td>
<td>0.94</td>
<td>0.91</td>
<td>0.89</td>
<td>0.87</td>
<td>0.72</td>
<td>0.52</td>
<td>0.42</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.99</td>
<td>0.99</td>
<td>0.98</td>
<td>0.96</td>
<td>0.93</td>
<td>0.92</td>
<td>0.9</td>
<td>0.77</td>
<td>0.6</td>
<td>0.58</td>
<td>0.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.99</td>
<td>0.99</td>
<td>0.97</td>
<td>0.97</td>
<td>0.95</td>
<td>0.87</td>
<td>0.74</td>
<td>0.64</td>
</tr>
</tbody>
</table>

Figure 2: Cross-dataset paired ranksums
SOS markers tested on Vijver

Vijver markers tested on DMFS

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 3rd

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.](image)

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.
<table>
<thead>
<tr>
<th>Subnetworks</th>
<th>unique genes</th>
<th>largest genes</th>
<th>avg size</th>
<th>doubles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chiang (PPI) top10</td>
<td>DMFS 40</td>
<td>96</td>
<td>14</td>
<td>701</td>
</tr>
<tr>
<td>Chiang (PPI) top10</td>
<td>SOS 40</td>
<td>63</td>
<td>10</td>
<td>73</td>
</tr>
<tr>
<td>Chiang (PPI) top10</td>
<td>VDIVER 10</td>
<td>68</td>
<td>12</td>
<td>78</td>
</tr>
<tr>
<td>Chiang (PPI) top10</td>
<td>WANG 10</td>
<td>88</td>
<td>12</td>
<td>114</td>
</tr>
<tr>
<td>Chiang (PPI) top50</td>
<td>DMFS 50</td>
<td>265</td>
<td>13</td>
<td>471</td>
</tr>
<tr>
<td>Chiang (PPI) top50</td>
<td>SOS 50</td>
<td>247</td>
<td>12</td>
<td>345</td>
</tr>
<tr>
<td>Chiang (PPI) top50</td>
<td>VDIVER 50</td>
<td>202</td>
<td>12</td>
<td>349</td>
</tr>
<tr>
<td>Chiang (PPI) top100</td>
<td>WANG 100</td>
<td>214</td>
<td>12</td>
<td>345</td>
</tr>
<tr>
<td>Chiang (PPI) top100</td>
<td>DMFS 100</td>
<td>436</td>
<td>14</td>
<td>883</td>
</tr>
<tr>
<td>Chiang (PPI) top100</td>
<td>SOS 100</td>
<td>349</td>
<td>12</td>
<td>670</td>
</tr>
<tr>
<td>Chiang (PPI) top100</td>
<td>VDIVER 100</td>
<td>336</td>
<td>12</td>
<td>651</td>
</tr>
<tr>
<td>Chiang (PPI) top100</td>
<td>WANG 100</td>
<td>385</td>
<td>12</td>
<td>704</td>
</tr>
<tr>
<td>Chiang (PPI) top200</td>
<td>DMFS 200</td>
<td>650</td>
<td>13</td>
<td>1659</td>
</tr>
<tr>
<td>Chiang (PPI) top200</td>
<td>SOS 200</td>
<td>595</td>
<td>12</td>
<td>1287</td>
</tr>
<tr>
<td>Chiang (PPI) top200</td>
<td>VDIVER 200</td>
<td>577</td>
<td>14</td>
<td>1269</td>
</tr>
<tr>
<td>Chiang (PPI) top200</td>
<td>WANG 200</td>
<td>592</td>
<td>12</td>
<td>1344</td>
</tr>
<tr>
<td>Chiang (PPI) top500</td>
<td>DMFS 500</td>
<td>1239</td>
<td>13</td>
<td>3888</td>
</tr>
<tr>
<td>Chiang (PPI) top500</td>
<td>SOS 500</td>
<td>1183</td>
<td>12</td>
<td>2971</td>
</tr>
<tr>
<td>Chiang (PPI) top500</td>
<td>VDIVER 500</td>
<td>1115</td>
<td>14</td>
<td>3114</td>
</tr>
<tr>
<td>Chiang (PPI) top500</td>
<td>WANG 500</td>
<td>1141</td>
<td>12</td>
<td>3052</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top10</td>
<td>DMFS 10</td>
<td>54</td>
<td>10</td>
<td>70</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top10</td>
<td>SOS 10</td>
<td>27</td>
<td>7</td>
<td>43</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top10</td>
<td>VDIVER 10</td>
<td>48</td>
<td>8</td>
<td>74</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top10</td>
<td>WANG 10</td>
<td>64</td>
<td>10</td>
<td>74</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top50</td>
<td>DMFS 50</td>
<td>181</td>
<td>14</td>
<td>300</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top50</td>
<td>SOS 50</td>
<td>114</td>
<td>7</td>
<td>214</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top50</td>
<td>VDIVER 50</td>
<td>118</td>
<td>8</td>
<td>218</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top50</td>
<td>WANG 50</td>
<td>221</td>
<td>12</td>
<td>344</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top100</td>
<td>DMFS 100</td>
<td>299</td>
<td>14</td>
<td>595</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top100</td>
<td>SOS 100</td>
<td>191</td>
<td>7</td>
<td>380</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top100</td>
<td>VDIVER 100</td>
<td>223</td>
<td>9</td>
<td>423</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top100</td>
<td>WANG 100</td>
<td>333</td>
<td>15</td>
<td>623</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top200</td>
<td>DMFS 200</td>
<td>421</td>
<td>13</td>
<td>930</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top200</td>
<td>SOS 200</td>
<td>387</td>
<td>9</td>
<td>743</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top200</td>
<td>WANG 200</td>
<td>498</td>
<td>15</td>
<td>1088</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top500</td>
<td>DMFS 500</td>
<td>749</td>
<td>13</td>
<td>1773</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top500</td>
<td>SOS 500</td>
<td>671</td>
<td>9</td>
<td>1352</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top500</td>
<td>VDIVER 500</td>
<td>794</td>
<td>12</td>
<td>1791</td>
</tr>
<tr>
<td>Lee (C2 V1.0) top500</td>
<td>WANG 500</td>
<td>822</td>
<td>15</td>
<td>2014</td>
</tr>
<tr>
<td>Park top10</td>
<td>DMFS 10</td>
<td>15</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>Park top10</td>
<td>SOS 10</td>
<td>51</td>
<td>10</td>
<td>51</td>
</tr>
<tr>
<td>Park top10</td>
<td>VDIVER 10</td>
<td>51</td>
<td>8</td>
<td>51</td>
</tr>
<tr>
<td>Park top10</td>
<td>WANG 10</td>
<td>73</td>
<td>13</td>
<td>73</td>
</tr>
<tr>
<td>Park top50</td>
<td>DMFS 50</td>
<td>162</td>
<td>81</td>
<td>162</td>
</tr>
<tr>
<td>Park top50</td>
<td>SOS 50</td>
<td>106</td>
<td>20</td>
<td>106</td>
</tr>
<tr>
<td>Park top50</td>
<td>VDIVER 50</td>
<td>100</td>
<td>34</td>
<td>100</td>
</tr>
<tr>
<td>Park top50</td>
<td>WANG 50</td>
<td>63</td>
<td>5</td>
<td>63</td>
</tr>
<tr>
<td>Park top100</td>
<td>DMFS 100</td>
<td>262</td>
<td>81</td>
<td>262</td>
</tr>
<tr>
<td>Park top100</td>
<td>SOS 100</td>
<td>193</td>
<td>39</td>
<td>193</td>
</tr>
<tr>
<td>Park top100</td>
<td>VDIVER 100</td>
<td>197</td>
<td>34</td>
<td>197</td>
</tr>
<tr>
<td>Park top100</td>
<td>WANG 100</td>
<td>128</td>
<td>5</td>
<td>128</td>
</tr>
<tr>
<td>Park top200</td>
<td>DMFS 200</td>
<td>411</td>
<td>81</td>
<td>411</td>
</tr>
<tr>
<td>Park top200</td>
<td>SOS 200</td>
<td>284</td>
<td>284</td>
<td>284</td>
</tr>
<tr>
<td>Park top200</td>
<td>VDIVER 200</td>
<td>272</td>
<td>34</td>
<td>272</td>
</tr>
<tr>
<td>Park top200</td>
<td>WANG 200</td>
<td>260</td>
<td>20</td>
<td>260</td>
</tr>
<tr>
<td>Park top500</td>
<td>DMFS 500</td>
<td>855</td>
<td>39</td>
<td>855</td>
</tr>
<tr>
<td>Park top500</td>
<td>SOS 500</td>
<td>606</td>
<td>39</td>
<td>606</td>
</tr>
<tr>
<td>Park top500</td>
<td>VDIVER 500</td>
<td>614</td>
<td>34</td>
<td>614</td>
</tr>
<tr>
<td>Park top500</td>
<td>WANG 500</td>
<td>724</td>
<td>43</td>
<td>724</td>
</tr>
<tr>
<td>Single top10</td>
<td>DMFS 10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Single top10</td>
<td>SOS 10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Single top10</td>
<td>VDIVER 10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Single top10</td>
<td>WANG 10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Single top50</td>
<td>DMFS 50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Single top50</td>
<td>SOS 50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Single top50</td>
<td>VDIVER 50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Single top50</td>
<td>WANG 50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Single top50</td>
<td>DMFS 500</td>
<td>500</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Single top50</td>
<td>SOS 500</td>
<td>500</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Single top50</td>
<td>VDIVER 500</td>
<td>500</td>
<td>500</td>
<td>500</td>
</tr>
</tbody>
</table>

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 unique subnetworks = 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.
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

<table>
<thead>
<tr>
<th>Columns</th>
<th>Markers</th>
<th>Chiang (PPI)</th>
<th>Chiang (T-score)</th>
<th>Lee (C2 V1.0)</th>
<th>Lee (PPI*)</th>
<th>Lee (Park)</th>
<th>Park</th>
<th>Singles</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMFS</td>
<td>SOS</td>
<td>10974</td>
<td>10974</td>
<td>13</td>
<td>36219</td>
<td>3</td>
<td>766</td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>SOS</td>
<td>10974</td>
<td>10974</td>
<td>14</td>
<td>35292</td>
<td>3</td>
<td>749</td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>10974</td>
<td>10974</td>
<td>15</td>
<td>51650</td>
<td>4</td>
<td>375</td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>10974</td>
<td>10974</td>
<td>16</td>
<td>49474</td>
<td>4</td>
<td>362</td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>10974</td>
<td>10974</td>
<td>17</td>
<td>1213</td>
<td>2</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>DMFS</td>
<td>522</td>
<td>647</td>
<td>8</td>
<td>1132</td>
<td>2</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>DMFS</td>
<td>522</td>
<td>625</td>
<td>9</td>
<td>1380</td>
<td>4</td>
<td>122</td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>DMFS</td>
<td>522</td>
<td>785</td>
<td>13</td>
<td>43</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>DMFS</td>
<td>522</td>
<td>691</td>
<td>9</td>
<td>1581</td>
<td>6</td>
<td>522</td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>DMFS</td>
<td>522</td>
<td>735</td>
<td>12</td>
<td>1748</td>
<td>3</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>DMFS</td>
<td>522</td>
<td>10149</td>
<td>20</td>
<td>43572</td>
<td>3</td>
<td>4141</td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>DMFS</td>
<td>362</td>
<td>10974</td>
<td>14</td>
<td>10974</td>
<td>4</td>
<td>4862</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Top 10 subnetworks. Markers from SOS, tested on DMFS.
<table>
<thead>
<tr>
<th>Tested</th>
<th>Markers</th>
<th>Obtained (PPI)</th>
<th>Obtained (v1.0)</th>
<th>Obtained (PPI)</th>
<th>Obtained (v1.0)</th>
<th>Lee (PPI)</th>
<th>Lee (v1.0)</th>
<th>Park</th>
<th>Singles</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>MAD2L1, MAD2L4, BUB1B, SAT2, PHA2, CENDA</td>
<td>HBD, IL2ST, JAK2, STAT5A, EPOK, RPL4, AB, JAK1</td>
<td>BDL, PKR1, TRIP13, NP, MORF4L2, HYOU1</td>
<td>TNRFSF14, P11, BA, IAP2, BVSL, GP5</td>
<td>RPL11, KIF13B</td>
<td>TNRFSF14, EBDX, TNRFSF14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>CDC2A3, CCNB2, PKMYT1, YWHAH, MAP2K1, CCNE2, CCNA2, ITGB1</td>
<td>TROAP, RGS, SPF, Cdc2, BIRC5, MAP2K1, CDK5R1</td>
<td>BCL2, FCER1A, Icos</td>
<td>EPHX2, CCNE2, RPL11, KIF13B</td>
<td>TNRFSF14, EBDX, TNRFSF14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>BUB1B, RAD54L, RAD51, PSMD7, L5M1, UF2</td>
<td>BUB1B, RAD54L, RAD51, PSMD7, L5M1, UF2</td>
<td>BCL2, IL2R, JAK1, IL6R</td>
<td>TRIP13, EAF1, RPRM2, CCT5</td>
<td>RPL11, KIF13B, BTG2, TRIP13, CCT5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>BYSL, TRGAP, TRIM37, KIAA0088, PRC1</td>
<td>AR, BIRC5F6, PDHB, TGFBR1, IKB1, VAV3, IRS1, INS, ARHGAP15</td>
<td>BCL2, IL2R, STAT5A</td>
<td>KIF13B, BCL2, IGF1, KIT</td>
<td>TNRFSF14, EBDX, BTG2, BTG2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>JK1, K-ALPHA-1, PSM, CCT2, TDG</td>
<td>MAGA4A2, STAT5A, JAK2, C10orf86, AR, RPL4, EPOK, JAK1</td>
<td>PSMD1, ABCF1, CDC20</td>
<td>BTG2, JAK2, STAT5A, JAK1</td>
<td>K-ALPHA-1, CCT5, NFASC, TAP6</td>
<td>BTG2, PARP3, KIF13B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>PSMD4, AURKA, FN1, COL13A1, M5P9, LGALS3BP, COL4A1, GTFB4, NAT10</td>
<td>TIA2, TIA2, CCNH, GTF2B, RRAA, POLR1B, RPL5, ESR1, RPL11, CCN52, MRP12</td>
<td>K-ALPHA-1</td>
<td>TNRFSF14, TME1, BTG2, PARP3, PSMD7</td>
<td>TME1, PARP3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>K-ALPHA-1, CCT5, THEG, CCT2</td>
<td>LOC105509, KPNPA1, NP, GAPDH</td>
<td>BTG2, BCL2, IGFBP6, PHL2</td>
<td>PARP3, BCL2, STAT5A, JAK1, PIK3CA, IL6R, IKB1, BIRC3</td>
<td>BTG2, KIF13B, SLC7, TME1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>THEG, CCT5, K-ALPHA-1, CCT2</td>
<td>RUS1, VAV2, IGF1R, JAK1, JAK2, ZXY, IL6ST, KIT</td>
<td>LCRBB2, ERBB2, RAD51, CCN5E1</td>
<td>BTG2, KIF13B, MAR, BTG2</td>
<td>BTG2, KDDH14A, KIF13B, BTG2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>K-ALPHA-1, CCT2</td>
<td>JAK3, JAK2, IGF1R, JAK1, JAK2, STAT5A</td>
<td>JIA2, IL14, BUB1B, SQLE</td>
<td>AKD1, GADI1, DPYD, ALDH2</td>
<td>BTG2, SLC7, COCH, SLC7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMFS</td>
<td>VIJVER</td>
<td>EPHX2</td>
<td>ARHDA1A, CTTN, ANK2F1, GRR2, PGD1, CCNA2, ACTR3</td>
<td>JAK2, CDH, JAK1, IFNG</td>
<td>ABHD14A, TPX2</td>
<td>BTG2, CDKN3, ZNF795, KIF20A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Top 10 subnetworks. Markers from VIJVER, tested on DMFS.
### Table 5: Top 10 subnetworks. Markers from DMFS, tested on SOS.

<table>
<thead>
<tr>
<th>tested</th>
<th>DMFS</th>
<th>Markers</th>
<th>Chaining (PPI)</th>
<th>Chaining (T-score)</th>
<th>Chaining (C2 V1.0)</th>
<th>Lee (C2 V1.0)</th>
<th>Lee (PPI*)</th>
<th>Lee (Park)</th>
<th>Park</th>
<th>singles</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOS</td>
<td>DMFS</td>
<td>ST1X TC2</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL IL6R</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL IL6R</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL IL6R</td>
<td>CXCA1D STAT5A STAT5B</td>
<td>C16orf61</td>
<td>STAT5B</td>
<td>STAT5A</td>
<td>CCNB1</td>
</tr>
<tr>
<td>SOS</td>
<td>DMFS</td>
<td>CNIF IL6ST NTBK2</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>CXCA1D STAT5A STAT5B</td>
<td>C16orf61</td>
<td>STAT5B</td>
<td>STAT5A</td>
<td>CCNB1</td>
</tr>
<tr>
<td>SOS</td>
<td>DMFS</td>
<td>SPN CDC2</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>CXCA1D STAT5A STAT5B</td>
<td>C16orf61</td>
<td>STAT5B</td>
<td>STAT5A</td>
<td>CCNB1</td>
</tr>
<tr>
<td>SOS</td>
<td>DMFS</td>
<td>PFP2</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>CXCA1D STAT5A STAT5B</td>
<td>C16orf61</td>
<td>STAT5B</td>
<td>STAT5A</td>
<td>CCNB1</td>
</tr>
<tr>
<td>SOS</td>
<td>DMFS</td>
<td>SFB20</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>CXCA1D STAT5A STAT5B</td>
<td>C16orf61</td>
<td>STAT5B</td>
<td>STAT5A</td>
<td>CCNB1</td>
</tr>
<tr>
<td>SOS</td>
<td>DMFS</td>
<td>TRM35</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>CXCA1D STAT5A STAT5B</td>
<td>C16orf61</td>
<td>STAT5B</td>
<td>STAT5A</td>
<td>CCNB1</td>
</tr>
<tr>
<td>SOS</td>
<td>DMFS</td>
<td>ARGBEF15</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>CXCA1D STAT5A STAT5B</td>
<td>C16orf61</td>
<td>STAT5B</td>
<td>STAT5A</td>
<td>CCNB1</td>
</tr>
<tr>
<td>SOS</td>
<td>DMFS</td>
<td>SORAB1</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>CXCA1D STAT5A STAT5B</td>
<td>C16orf61</td>
<td>STAT5B</td>
<td>STAT5A</td>
<td>CCNB1</td>
</tr>
<tr>
<td>SOS</td>
<td>DMFS</td>
<td>SRR4</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>CXCA1D STAT5A STAT5B</td>
<td>C16orf61</td>
<td>STAT5B</td>
<td>STAT5A</td>
<td>CCNB1</td>
</tr>
<tr>
<td>SOS</td>
<td>DMFS</td>
<td>SRR4</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>STAT5A JAK1 PTK2B CXCL12 IL6ST PYN EVL</td>
<td>CXCA1D STAT5A STAT5B</td>
<td>C16orf61</td>
<td>STAT5B</td>
<td>STAT5A</td>
<td>CCNB1</td>
</tr>
</tbody>
</table>

---

8
<table>
<thead>
<tr>
<th>Tested Markers</th>
<th>Chuang (PPI)</th>
<th>Chuang (Park)</th>
<th>Chuang (C2 V1.0)</th>
<th>Lee (C2 V1.0)</th>
<th>Lee (PPI)</th>
<th>Lee (Park)</th>
<th>Park</th>
<th>Singles</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOS VJIVER</td>
<td>MAP62</td>
<td>MYBL2</td>
<td>SKP2</td>
<td>MELK</td>
<td>CDC34</td>
<td>NCO2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PFCG4</td>
<td>GADPH</td>
<td>TRK1</td>
<td>PGK1</td>
<td>MAP45</td>
<td>MAP3K1</td>
<td>UBE2A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS VJIVER</td>
<td>PAPARH3</td>
<td>GADPH</td>
<td>TRK1</td>
<td>PGK1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS VJIVER</td>
<td>PAPARH3</td>
<td>GADPH</td>
<td>TRK1</td>
<td>PGK1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS VJIVER</td>
<td>MEE51</td>
<td>PKX3</td>
<td>NSC31</td>
<td>STAT5A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS VJIVER</td>
<td>SOLSA</td>
<td>TXP2</td>
<td>AURKA</td>
<td>SMAD3</td>
<td>TURA1</td>
<td>MLL2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS VJIVER</td>
<td>TGF1</td>
<td>NOV</td>
<td>INS</td>
<td>TGAV</td>
<td>HSL1</td>
<td>LRBP2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS VJIVER</td>
<td>SOD2</td>
<td>MSH2</td>
<td>PFKL</td>
<td>PFKP</td>
<td>KARS</td>
<td>PKM2</td>
<td>PIN1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS VJIVER</td>
<td>GFP1</td>
<td>NOV</td>
<td>INS</td>
<td>TGAV</td>
<td>HSL1</td>
<td>LRBP2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS VJIVER</td>
<td>HHRK</td>
<td>CTBP2</td>
<td>SNAI2</td>
<td>BAZ2B</td>
<td>RA1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOS VJIVER</td>
<td>BPH</td>
<td>STS75A</td>
<td>SOCS2</td>
<td>JAK1</td>
<td>IL6ST</td>
<td>CD247</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Top 10 subnetworks. Markers from VIJVER, tested on SOS.
### Table 7: Top 10 subnetworks. Markers from DMFS, tested on VIJVER.

<table>
<thead>
<tr>
<th>Subnetwork</th>
<th>Markers</th>
<th>Chiang (PPI)</th>
<th>Chiang (C4 V0.1)</th>
<th>Lee (C4 V1.0)</th>
<th>Lee (PPI)</th>
<th>Lee (Park)</th>
<th>Park</th>
<th>Singles</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIJVER DDFS</td>
<td>SLC25A1</td>
<td>G6P</td>
<td>ORA2</td>
<td>E2F1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER DDFS</td>
<td>MTR</td>
<td>PKMY1</td>
<td>HN</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER DDFS</td>
<td>PERKBP</td>
<td>PISMD2</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER DDFS</td>
<td>UNC44</td>
<td>BIRC5</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER DDFS</td>
<td>RPL12B</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER DDFS</td>
<td>PRKCI</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER DDFS</td>
<td>AURKB</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER DDFS</td>
<td>HN</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Subnetwork</th>
<th>Markers</th>
<th>Chiang (PPI)</th>
<th>Chiang (C4 V0.1)</th>
<th>Lee (C4 V1.0)</th>
<th>Lee (PPI)</th>
<th>Lee (Park)</th>
<th>Park</th>
<th>Singles</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIJVER SOS</td>
<td>ARF3</td>
<td>EIFP2</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER SOS</td>
<td>ABHD5A</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER SOS</td>
<td>FEN1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER SOS</td>
<td>USHBP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER SOS</td>
<td>GTF2H4</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER SOS</td>
<td>CFL</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER SOS</td>
<td>FEN1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
<tr>
<td>VIJVER SOS</td>
<td>TAPF1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
<td>EIFP1</td>
</tr>
</tbody>
</table>

### Table 9: Additional subnetworks.

- **Fen1** is a marker for the TNFRSF1A (CD123) subnetwork.
- **CFL1** is a marker for the TNFRSF1A (CD123) subnetwork.
- **PDK1** is a marker for the TNFRSF1A (CD123) subnetwork.

---

**Note:** The tables are based on the analysis of publicly available datasets and reflect the top 10 subnetworks identified by the indicated methods. The markers are listed along with their corresponding subnetworks and additional markers for further investigation.
1 Models

![Figure 1: Different types of models. Models \(\theta_1\) and \(\theta_2\) use a normally distributed hidden layer where the features and linear combinations of these hidden variables. Models \(\theta_3\) and \(\theta_4\) use binary hidden layer where the features are logical functions of these latent variables. The outcome variable \(Y\) is modeled as being another feature, and discretized to get a 0-1 label. Models \(\theta_2\) and \(\theta_4\) have two separate models for the poor and good case.

In order of increasing complexity, the different models are:

- **Model \(\theta_1\)**. 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 similarly as the a feature of \(X\), but discretized. This is the model described in Park.

- **Model \(\theta_2\)**. 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 \(\theta_3\)**. 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 \(\theta_4\)**. This model is similar to \(\theta_3\), but with separate 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 \(\theta_1\)**. \(\mu_{h_i}, \sigma_{h_i}^2\) for \(h_i \in H\). \(\beta_{x_{i}}, \sigma_{x_{i}}^2\) for \(x_i \in X\). \(\beta_y, \sigma_y^2\).

- **Model \(\theta_2\)**. \(\mu_{h_y,i}, \sigma_{h_y,i}^2\) for \(h_{y=0,i} \in H_{y=0}, h_{y=1,i} \in H_{y=1}\). \(\beta_{x_{y,i}}, \sigma_{x_{y,i}}^2\) for \(x_{y=0,i} \in X_{y=0}, x_{y=1,i} \in X_{y=1}\). \(p_y=0\).

- **Model \(\theta_3\)**. \(p_{hi}\) for \(h_i \in H\). \(B_{x_i}\) for \(x_i \in X\). \(B_y, \mu_{x_{i}}, \mu_{y_{i}}, \Sigma_{x_{i}}, \Sigma_{y_{i}}\) for \(x_i \in X\). \(\Sigma_{y_{i}}=0, \sigma_{y_{i}}=1\).

- **Model \(\theta_4\)**. This model won’t be used since model \(\theta_3\) is complex enough for our purposes.

For the simpler models \(\theta_1\) and \(\theta_2\) 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 \(\theta_1\)**. \(\mu_{x_{i}}, \Sigma_{x_{i}}, \beta_{y}, \sigma_{y}^2\).

- **Model \(\theta_2\)**. \(p_y=0, \mu_{y_{i}}, \Sigma_{y_{i}}\) for \(y_{i}\) and \(x_{i}\) respectively with \(\mu_{y_{i}}=0, \Sigma_{y_{i}}=1\).

Since we’ve obtained simpler models for \(\theta_1\) and \(\theta_2\), 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 \(\theta_3\) and \(\theta_4\), but this would require too complex notation and won’t probably be easily implementable.

- **Model \(\theta_1\)**. \(\hat{y} = 1 \text{ if } (X\beta_y > 0), \hat{y} = 0 \text{ otherwise}\).

- **Model \(\theta_2\)**. \(\hat{y} = 1 \text{ if } g_{y=1} > g_{y=0}, \hat{y} = 0 \text{ otherwise}\). Here \(g_{y=c} = \log(p_{y=c}) - 0.5\log(\Sigma_{y=c}) - 0.5(X - \mu_{y=c})\Sigma_{y=c}^{-1}(X - \mu_{y=c})\).

1.1 Standard model \(\theta_1\)

The standard model \(\theta_1\) 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:

- \(\mu_{h_i} = 0, \sigma_{h_i}^2 = 1\) for \(h_{i=1...h_{100}}\). \(\beta_{x_{1}}(n) = 1\) if \(n = \lfloor i/1000 \rfloor\), otherwise \(0, \sigma_{x_{i}}^2 = 2^2\) for \(x_{i=1...x_{1000}}\). \(\beta_y(n) = 1\) if \(n < 11\) otherwise \(0, \sigma_y^2 = 1\).
1.2 Standard model $\theta_2$

Standard model $\theta_2$ 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.

1.3 Standard model $\theta_3$

Standard model $\theta_3$ ...

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 $\theta_1$ has the important property that the entire model could theoretically be modeled as one big joint multivariate gaussian distribution. This means that if variable 1 is positively correlated with the outcome $y$, and variable 2 is also positively correlated with $y$, then variable 1 must also be correlated with variable 2. This behaviour should be advantageous to an algorithm such as Park, which searches for correlated variables.

However, model $\theta_2$ 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 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 crossvalidation is we would take the first 5, 10, 15 ... 1000 features. So for model $\theta_1$ and $\theta_2$, we would expect the optimum performance when the first 100 features are selected, which are all signal features. For model $\theta_3$ 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](image)

**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 $\theta_1$ and $\theta_2$, 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 $\theta_1$, 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](image)

**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](image.png)

**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](image.png)

**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 $\theta_2$, the variables are correlated in groups of 10, making it similar to model $\theta_1$.

![Figure 10: Performances of Park and Lee, depending on the error introduced on the features.](image)

**Observation 5** Judging from the $S$ shapes, Park’s algorithm seems much more dependent on the $\sigma_x$ than the other algorithms. Since a higher $\sigma_x$ 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 advantageous 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 $\theta_3$.

- 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

\[ y = \text{discrete}[h_1 + h_2 + \ldots + h_{100} + \epsilon], \epsilon \sim N(\mu=0,\sigma^2=1) \]

Figure 1: Standard model 1

\[ y' = \text{discrete}[y' \sim N(\mu=2y'-1,\sigma'=-1)] \]

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 dataset was subjected to a cross-dataset-cross-validation, so each point, and its variance, is based on 5x2=10 average AUCs. Most of the graphs should speak for themselves, although for clarity Figure 7 and Figure 8 are visualized in Figure 6. Figure 9 is 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

- **Uncorrelated, signal only**
  - $x_1$, $x_{12}$, ..., $x_{91}$
  - $x_{100}$, $x_{110}$, ..., $x_{191}$
  - $x_{200}$, $x_{210}$, ..., $x_{291}$

- **Correlated, signal only**
  - $x_1$, $x_{12}$, ..., $x_{91}$
  - $x_{100}$, $x_{110}$, ..., $x_{191}$
  - $x_{200}$, $x_{210}$, ..., $x_{291}$

- **Correlated**
  - $x_1$, $x_{12}$, ..., $x_{91}$
  - $x_{100}$, $x_{110}$, ..., $x_{191}$
  - $x_{200}$, $x_{210}$, ..., $x_{291}$

- **Mixed**
  - $x_1$, $x_{12}$, ..., $x_{91}$
  - $x_{100}$, $x_{110}$, ..., $x_{191}$
  - $x_{200}$, $x_{210}$, ..., $x_{291}$

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

## Figure 7: Model $\theta_1$ versus model $\theta_2$

## Figure 8: Correlated versus uncorrelated
Figure 9: Correlated combined versus uncorrelated combined, but with remaining features included.

Figure 10: Model $\theta_1$

Figure 11: Model $\theta_2$

Figure 12: Model $\theta_1$

Figure 13: Model $\theta_2$