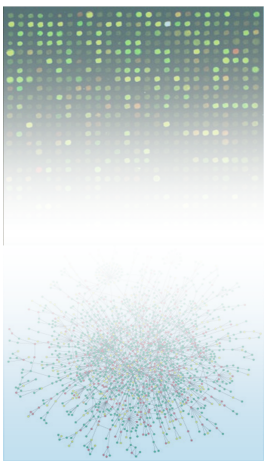
Tel-Aviv University
Raymond and Beverly Sackler
Faculty of Exact Sciences
The Blavatnik School of Computer Science



Prior Knowledge Integration of Gene Networks Data into Gene Expression Analysis

Thesis submitted in partial fulfillment of the requirements for the M.Sc. degree in the School of Computer Science, Tel Aviv University

By

Ofer Lavi

The research work for this thesis has been carried out at Tel Aviv University under the supervision of Prof. Ron Shamir and Prof. Gideon Dror

August 2010

Acknowledgements

When I started my studies in Tel-Aviv University towards my M.Sc. degree, I believed it would take a while until I focus on a specific research subject or even on a general research field. Having no biological background, I was lucky enough to take two of my first year's classes with my advisor, Prof. Ron Shamir, who introduced me to the exciting world of computational biology and allowed me to hop on the research train in his wonderful laboratory in Tel-Aviv. I would like to thank him for the giving me the opportunity, trust and guidance through the hills and valleys we shared during the time. I would also like to thank my co-advisor, Prof. Gideon Dror, for his enlightening insights on machine learning, gluing together learning theory and computational biology in a way that was crucial for the success of this work.

Throughout the time, I was surrounded by a group of great people, who constitute the amazing DNA of Ron Shamir's ***A**lgorithms in **C**omputational **G**enomics in **T**el-Aviv university*. I thank Igor Ulitsky for teaching me most of what I know about the practical side of computational biology, Chaim Linhart for his wonderful methodological advices, Gal Romano for introducing me to real biological experiments, and Michael Gotkin and Didi Amar for sharing with me the work on the GenePark Project. I would also like to thank Sharon Bruckner, Falk Hüffner, Yonit Halperin, Adi Maron-Katz, Seagull Shavit, Michal Ozery-Flato, Renana Meller, Guy Karlebach, Guy Harari, Mukul Bansal, Roye Rozov, Arnon Paz, Eyal David and Akshay Krishnamurthy for always being available to listen, brain-storm together and put in their 2-cents.

I would also like to thank a few people outside of the laboratory, who gave me from their time and contributed in their field of expertise - Prof. Martin Kupiec for guiding us at Gal Romano's yeast QTL project, Jonathan Rosenblatt and Dr. Ronny Luss for their help in statistics and SVM, Haim Avron for linear algebra remarks, and TaeHyun Hwang (University of Minnesota) and Han-Yu Chuang (University of California, San Diego) for the permission and help in running their code for the research purposes. I would also like to thank Gilit Zohar-Oren and Ruth Friedberg for all their administrative and moral help.

Conducting research is only part of the whole story. This work is dedicated to my beloved wife, Keren, who courageously found the strength to push and support me before, and during my studies, sharing a great deal in completing this thesis, and to my wonderful daughters, Sha'ked and Libby. I would like to thank my parents, Yehezkeal and Shlomit for seeding me with the passion for learning, and my in-laws, Yehoshua and Paula for their endless support throughout the time.

This research was supported in part by the GENEPARK project which is funded by the European Commission within its FP6 Programme (contract number EU-LSHB-CT-2006-037544).

Abstract

Analysis of gene expression using microarrays went into clinical use in the recent years using gene signatures as biomarkers for disease prognosis in various cancers. Prediction methods that are based on a small computationally selected set of genes show significantly higher accuracy compared to previous, more traditional methods.

In spite of their higher accuracy, these methods have drawn critique from both biomedical and computational scientists. For the life-scientists, the fact that treatment decisions are made in a ”black box” fashion, treating genes as numbers, and lacking any biological and cellular rationale, is dissatisfying. From a computational perspective, these methods suffer from shortage of training data: thousands of expression values were measured for each patient, but only in a few hundred patients.

In this thesis, we propose to use additional information to overcome the shortcomings of the current methods for selecting diagnostic markers. By collecting published and experimental information, many biological systems can be summarized as signaling and interactions networks. Exploiting the finding that close genes in the network tend to have a similar contribution to learned classification models, we combine expression profiles of cases and controls with large scale protein-interaction networks in order to improve classification results. Our method is based on using SVM with a kernel tailored to integrate expression as well as network topology data. We demonstrate our method on a number of case-control studies from the literature.

Contents

1

Introduction and Summary

This work focuses on methods for incorporating prior knowledge regarding the relations between variables for the purpose of data classification. In particular, we study ways to improve generalization of supervised learning algorithms by exploiting pairwise relations between features as defined by a given network encapsulating prior knowledge in the field of interest. Our main application is in the field of microarray analysis, and more specifically in classification of gene expression profiles with the help of gene networks.

In this chapter we give an overview of this work's topics, informally introduce the problem we tackle, and explain our motivation and the general path we chose towards a solution. The reader should be aware that some of the concepts we present here will only be defined formally later on, as we would like this introduction to be short and concise.

1.1 Classification and Supervised Learning

Classification is a common task in everyday life, as well as in many scientific fields. Consider an office administrator sorting documents by their subjects into corresponding folders, a physician diagnosing a patient's disease or a little kid grouping objects by shape or color. In this framework, we have a set of *samples*, each belongs to one of a set of known *classes*. The task is then, given a new sample, to assign a class (or a *label* - a name tag corresponding to the class) to it. A computer is a natural candidate for performing various classification tasks. It can process large amounts of data in a fraction of the time required for a human for the same task. However, it lacks the knowledge required for taking the decision upon the actual class of a sample.

Supervised learning aims to overcome this lack of knowledge by dividing the task into

two phases. In the first phase, the *learning* (or *training*) phase, the computer is taught by a supervisor. The supervisor presents a set of samples to the computer, coupled with the right class of every sample. During this phase, the computer builds a *model* that will later aid it in taking the right decision. In the second phase, the *prediction* (or *test*) phase, the computer is presented with a new unseen sample, this time without any class associated with it, and its task is to utilize the model in predicting the right class for this sample.

One of the most important roles of the designer of such a framework, is to represent the samples in a form that is readable by the computer, keeping as much of the meaning of the original sample. Usually, one designs a set of *features*, which are individual measurable properties of the samples, allowing for representing each sample as a list of values corresponding to the designed set of features. From that point on, the actual features are meaningless to the computer, as it only treats the measured values.

For instance, in the documents example, one can design a set of features that includes a large number of words, and represent each document as a list of values, one value per word. The value might be 0 if the word does not appear in the document and 1 if it does. The computer now deals with an ordered list of 0's and 1's, regardless of the meaning of the original words. However, it can learn to associate certain features' values with certain classes, as the supervisor also presents it with the right class for every such list of 0's and 1's. For the doctor's example, features can be different clinical measurements results (blood, urine etc.) or different fitness levels in tests conducted by the physician.

1.2 Learning with Prior Knowledge

In some cases, today's technology offers us vast amounts of data, which can hardly be analyzed by humans. DNA Microarrays are one such technology, which will be described in the biological background section hereafter. They are capable of providing thousands to millions of measurements called *gene expression levels*, corresponding to gene activity levels within a living cell or tissue. These measurements have been successfully used as features for the supervised learning task, aiding physicians in diagnosis and treatment selection. Once again, even if a gene's biological function is known, this knowledge is ignored when only the gene expression level is used.

Gene expression analysis poses a set of computational challenges, which are cascaded in turn to the classification task. Some of the reasons for difficulty in microarray analysis include:

1. **Noisy data** - the biochemical measurement process of gene expression levels is known to be inaccurate.

2. **Dimensionality ratio** - the number of samples used in the learning phase is far smaller than number of features.

3. **Feature dependencies** - biologically, genes are known to work together, but these known dependencies are not used if the analysis is based only on a list of independent expression values.

In order to help overcome these challenges, it might be advisable to incorporate prior knowledge regarding the features, instead of discarding information that is available about them. For genes, such information was gathered by curators from the literature and is available in various forms. One available resource is *functional annotation* of genes. Such resource describes the functions of each gene, thus grouping genes according to their function.

Other forms of biological information include small but detailed *gene regulatory pathways* describing pairwise interrelations among a focused group of genes. Such group typically contains dozens to hundreds of genes. Large scale *protein-protein interaction networks* can cover thousands of genes, describing in less detail general interactions or other relations among pairs of genes.

Various methods use such prior knowledge when analyzing gene expression data, and we shall review some of them later in this study. We chose to use the latter large-scale networks as sources for prior knowledge regarding relations between pairs of genes. We employed such networks as an additional input while building the model during the supervised learning process. The rationale for utilizing this information in classification is that neighbor genes in the network tend to work together (or *co-express*), and hence their joint effect can lead to better classification.

1.2.1 Summary of our Results

Our main contribution is a simple modification to the commonly used *SVM* learning and classification algorithm [74]. We reformulate the SVM problem to take prior knowledge of similarity between features into account. Algorithmically speaking, this modification amounts to running SVM on the result of a linear transformation of the original expression data, a transformation that is based on the network encapsulating the prior knowledge.

This thesis is organized as follows: in Chapter 2 we will briefly review the necessary computational and biological background for this work. In Chapter 3 we map and analyze some extant integration methods of network data to gene expression analysis, and present some results regarding network integration assumptions and its contribution. In Chapter 5 we present a novel algorithm for gene expression classification using network data, and in Chapter 6 we report results of applying the algorithm to a few case-control datasets from the literature. We discuss the results and suggest some future research directions in Chapter 7.

2

Background

This chapter is devoted to laying the background and definitions required for the rest of this thesis. In Section 2.1.1 we will briefly define the problem of supervised learning of a classifier. The SVM algorithm will be described in detail in Section 2.1.2, emphasizing the building blocks used later in this work. In Section 2.2 we will present a short biological background, followed by introduction to DNA microarray analysis in the context of classification. We will end this chapter with a review of the problems in DNA microarray analysis that can benefit from incorporating prior knowledge in the form of gene networks.

2.1 Computational Background

In this section we introduce basic background on classification. For much more on the topics see [76, 51].

2.1.1 Problem Setting

In this section we shall describe the problem of learning a classifier in a supervised manner, and define the notation that will be used throughout this work. As mentioned in the previous section, a classifier should be able to correctly assign a sample to one (or more) predefined classes. Regardless of the specific classification task, we assume that the data are already represented in a format readable by a computer.

A *sample* $\mathbf{x} \in \mathbb{R}^p$ is a p-dimensional vector of values, one for each feature. Let C be a set of class labels. A *classifier* is a function $h : \mathbb{R}^p \rightarrow C$ that takes as input a sample and outputs a prediction regarding the class this sample belongs to.

In a supervised learning setting, the task is to build a classifier using a set of labeled training samples. The input is then a set of pairs $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$ where $y_i \in C$ is the label of sample $\mathbf{x}_i$, and the output is h. The set of training samples is often called a *training set* and the assumption is that these are uniformly sampled from the true population of all samples. The resulting classifier then should fit the true population in its predictions. To this end, a loss function $L : C \times C \rightarrow \mathbb{R}^+$ is defined, which punishes each misclassification. If a sample belonging to class $y \in C$ is assigned by the classifier to class $z \in C$, $z \neq y$, then L assigns a positive cost to such an error. If the classification of the sample is correct, the cost is 0. The objective is to minimize the expected cost over the true population. This expectation is often called the *risk*, and the target of a learning algorithm is to use the given samples in order to build a classifier that minimizes the risk.

2.1.1.1 The Binary Case

In the most common case, the number of possible classes is 2. The task in this case is referred to as *binary classification*. We shall limit our discussion here to the binary case. We shall use $C = \{-1, +1\}$ as the set of labels for the possible classes. Many methods are available for generalization of binary classification to *multiclass classification*.

2.1.1.2 Linear Discriminant Functions

One family of binary classifiers is based on linear discriminant functions. This family of functions include all the functions that are a linear combination of the values of $\mathbf{x}$. That is, they are of the form $g(x) = \mathbf{w}^t\mathbf{x} + w_0$, where $\mathbf{w}$ is a called *weight vector* and w_0 can be considered as a *threshold*. A binary classifier that outputs 1 for a sample $\mathbf{x}$ where $g(\mathbf{x}) > 0$ and -1 otherwise is called a linear classifier.

Geometrically, each sample can be seen as a point in a p-dimensional space. The discriminant function associated with a linear classifier defines a decision boundary which is a surface that separates samples that are assigned by the classifier to one class from those that are assigned to the other class. Since the function is linear, this surface is actually a $(p-1)$-dimensional hyperplane, and the sample points from different classes reside on different sides of the hyperplane. (Compare Figure 2.1 for an example)

The task of learning a linear classifier narrows down to learning a weight vector $\mathbf{w} =$

$w_0, w_1, \ldots, w_p$. Since each component w_j, $1 \leq j \leq p$ corresponds to a single feature, a simple interpretation of the weights indicates that $|w_j|$ corresponds to the importance of feature j as a predictor for one of the classes and against the other. The sign of w_j indicates positive values of feature j contribute to deciding towards the first class or the second class.

2.1.2 Support Vector Machines

Support Vector Machines (SVMs) [74] are among the most popular supervised learning algorithms. Before introducing SVM we need to consider two more simple notions - linearly separable data and margin.

Linearly Separable Data

In the binary setting, if there exists a hyperplane (corresponding to a linear classifier) that correctly separates the samples into two classes, residing from the two sides of the hyperplane, then the data are termed linearly separable. This is not always the case, and in fact, most real world data isn't at all linearly separable. Figure 2.1 shows an example of linearly separated data, with two proposed linear separators.

Margin

If the data are linearly separable, then there could be many hyperplanes that correctly separate the samples into two sets. Let H be one such hyperplane. The classifier's *margin* is defined as the largest distance between two hyperplanes parallel to H that do not contain any sample points between them. Intuitively, we would like to have as large margin as possible on the training samples. This will give us a larger buffer for future unseen samples, which can fall within the margin. Indeed, the margin has been proven to be tightly coupled with the expected generalization error [13]. In Figure 2.1 we can see an example of a separator that maximizes the margin.

While training the classifier, SVM, in its simplest form, finds a linear discriminant function that maximizes the margin. Consider a hyperplane $\mathbf{w}^T\mathbf{x}+w_0^* = 0$ that is indeed a separating hyperplane. The vector $\mathbf{w}$ is perpendicular to this hyperplane, and we can have

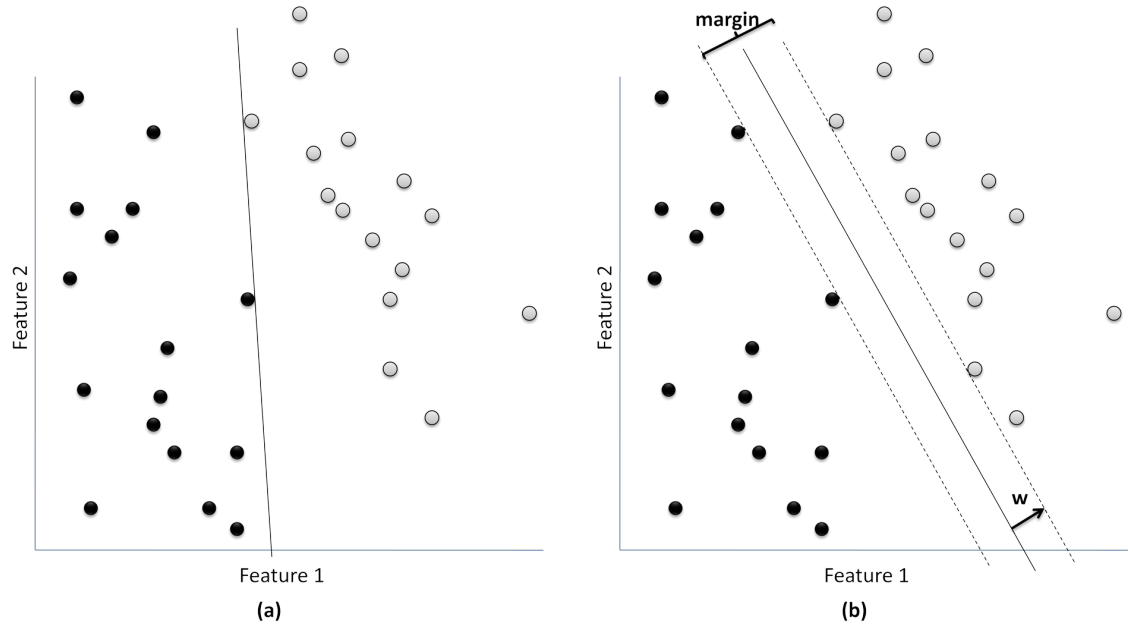


Figure 2.1: **Two linear separators of the same 2-class data**. Each dot is a sample, with features corresponding to its x and y coordinates. The class label in indicated as the dot color. With only two features, the separating hyperplane is actually a line in the 2-dimensional space. **(a) Linearly Separable Data**: The black line is a linear separator between the two classes. This is only one of many possible linear separators of the points in the example. **(b) Max Margin Classifier**: The same data as in (a), this time separated by another line that maximizes the margin between the data points. $\mathbf{w}$ is the vector perpendicular to the separator, and $\frac{2}{\|\mathbf{w}\|}$ is the margin's width.

new hyperplanes that are parallel to our separating hyperplane by sliding the separating hyperplane along this vector $\mathbf{w}$ until the point the hyperplane reaches one of the sample points. Looking at the two such hyperplanes, one touching a positive point and the other touching a negative point, we can pick w_0 for a hyperplane $\mathbf{w}^T\mathbf{x}+w_0 = 0$ that is exactly in the middle in between these two hyperplanes. The two hyperplanes are then $\mathbf{w}^T\mathbf{x} + w_0 = -b$ and $\mathbf{w}^T\mathbf{x} + w_0 = b$. We can scale $\mathbf{w}$, w_0 and b while still keeping the very same two hyperplanes, so w.l.o.g. we can omit b as a parameter and have the two hyperplanes be written now as $\mathbf{w}^T\mathbf{x} + w_0 = -1$ and $\mathbf{w}^T\mathbf{x} + w_0 = 1$.

We seek two parallel hyperplanes that maximize the margin, that is, the distance between them, $\frac{2}{||\mathbf{w}||}$, is maximal. At the same time we want to constrain the hyperplanes such that no sample point resides between them, meaning that the margin region between them is empty. We add then a constraint $\mathbf{w}^T\mathbf{x}_i + w_0 \geq 1$ for each point $\mathbf{x}_i$ of the first class, and $\mathbf{w}^T\mathbf{x}_i + w_0 \leq -1$ for each point $\mathbf{x}_i$ of the second. As mentioned, we have two classes $C = \{-1, +1\}$, so these constraints can be rewritten as $(\mathbf{w}\mathbf{x}_i + w_0) \cdot y_i \geq 1$.

2.1.2.1 Primal Form

The problem is formulated then, as a *Quadratic Programming* optimization problem:

$$\min_{\mathbf{w},w_0} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 \right\}$$

subject to

$$(\mathbf{w}^T\mathbf{x}_i + w_0) \cdot y_i \geq 1 \qquad \text{for each } i = 1, \ldots, n$$

The substitution of $\|\mathbf{w}\|$ with $\frac{1}{2}\|\mathbf{w}\|^2$ is allowed because both functions reach their minimum at the same vector $\mathbf{w}$. The factor $\frac{1}{2}$ is there merely for mathematical convenience.

2.1.2.2 Dual Form

By introducing new variables, the *Lagrange Multipliers*, it is possible to express the optimization problem with an alternative objective function under different constraints.

As we have n original constraints, we need n Lagrange multipliers, denoted α_i for $i = 1, \ldots, n$.

The dual form of the SVM optimization problem is:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_l^T \mathbf{x}_j$$

subject to

$$\alpha_i \geq 0 \qquad \text{for each } i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

Note that in this formulation the input points $\mathbf{x}_i$ only appear as pairwise dot-products $\mathbf{x}_i^T \mathbf{x}_j$

2.1.2.3 Soft Margin

In most cases, the data are not linearly separable. For these cases, the SVM formulation does not have a feasible solution, as no separating hyperplane exists. Suppose we allow some points to fall on the wrong side of some hyperplane with respect to their real class. If we ignore them, this hyperplane cleanly separates the rest of data. However, we do not want to have too many such points, so we penalize every such point according to how deep it is in the wrong side of the hyperplane - how bad the misclassification level of the sample is.

On one hand, we would like to minimize the number of misclassified samples. On the other hand, we might find a better hyperplane in terms of the margin, if we allow for more misclassified samples. This tradeoff is translated to the following soft margin SVM formulation due to [18]. For every point we add a non-negative *slack variable* ξ_i which is 0 if the sample is correctly classified. For the cases where the sample is incorrectly classified, the larger its absolute value, the further away the point is from the hyperplane. In addition, we introduce another *regularization parameter* C, which sets the tradeoff between a large margin and a small total misclassification. The problem is now formulated as:

$$\min_{\mathbf{w},w_0,\xi} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \right\}$$

subject to

$$\begin{aligned} (\mathbf{w}^T\mathbf{x}_i + w_0) \cdot y_i &\geq 1 - \xi_i && \text{for each } i = 1, \ldots, n \\ \xi_i &\geq 0 && \text{for each } i = 1, \ldots, n \end{aligned}$$

If a training point $\mathbf{x}_i$ falls on the wrong side of the hyperplane (misclassified), its corresponding ξ_i must be larger than 0, and the total number of misclassifications is bound by $\sum_{i=1}^{n} \xi_i$. Thus, the larger C is, the higher the cost for total misclassifications in the training set.

This problem can also be expressed in a dual form, with the advantage that all the slack variables vanish, and we are left only with the regularization term C:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \mathbf{x}_l^T \mathbf{x}_j$$

subject to

$$\begin{aligned} C \geq \alpha_i \geq 0 && \text{for each } i = 1, \ldots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$

2.1.2.4 Non-Linear SVM

Often the data are not separable in the original space. Before applying SVM, one can transform the sample points to a space of (possibly) higher dimension using some transformation Φ, where the data might be linearly separable. Then, we can find a maximum margin linear classifier in the transformed space using SVM, and classify every new point by first applying the transformation to it, and then using the classifier. However, the complexity of learning a classifier is now higher, as it depends on the space dimension, which is now higher.

As mentioned, the dual form never takes into account a single sample by itself, but rather a dot product of two samples. This means that instead of considering a single trans-

formed sample point $\Phi(\mathbf{x}_i)$, we only have to look at dot product pairs of two transformed samples $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. If there was a way to efficiently compute such a dot product in the original space (which is of a lower dimension) without going through transforming the points themselves, we could avoid the transformation and the dot product computation in the transformed space. In [12], the authors introduce *non linear kernel functions* $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ that allow for computing the dot product of the transformed sample points in some higher dimensional space, given only the two original points.

Now, given such a kernel function, all we have to do in order to get a non-linear classifier, is to replace the computation of dot products $\mathbf{x}_i^T \mathbf{x}_j$ with that of one of these kernel functions $k(\mathbf{x}_i, \mathbf{x}_j)$.

Two of the kernel functions used in the literature:

Polynomial kernels $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^d$ or $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^d$

Radial Basis Function $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$

The space corresponding to the kernel function can even be of infinite dimension. In this case a straight forward transformation is not applicable at all, but a use of such a space is possible thanks to the kernel functions.

2.1.3 Performance Measurements

Recall that our intention is to improve classification performance by integrating prior knowledge into the learning process. In order to measure such an improvement, we need a way to compare the classification performance of the different models with and without the prior knowledge. Throughout this work we used the standard *area under the ROC curve* score for estimating a classifier's performance. In the binary case, each sample is labeled as *positive* or *negative*. The classifier can classify a sample as either positive or negative, and its decision can be either *True* (right) or *False* (wrong). We now can partition the classifier's decision as for the classes of a given set of samples into four disjoint sets:

True Positives (TP) positive samples correctly classified as positive by the classifier.

False Positives (FP) negative samples incorrectly classified as positive by the classifier.

True Negatives (TN) negative samples correctly classified as negative by the classifier.

False Negatives (FN) positive samples incorrectly classified as negative by the classifier.

There are several common aggregate measures for classification quality. *sensitivity* is defined as the ratio $\frac{|TP|}{|TP|+|FN|}$. A classifier with 100% sensitivity classifies all positive samples as positive (i.e. it has no false negatives). However, one can easily build such a classifier by classifying every sample as positive, regardless of the sample itself, which would create many FP errors. A complementary measure is *specificity*, $\frac{|TN|}{|TN|+|FP|}$. We seek for a classifier with high performance in both measures.

2.1.3.1 AUC - Area Under Receiver Operating Characteristic Curve

Often, classification algorithms return some score for a given sample. By setting a threshold, the score is turned into binary classification of the sample. Every possible threshold, ranging from the minimal to the maximal score, determines the sensitivity and specificity of the classifier. Increased sensitivity comes at the expense of decreased specificity, and vice versa.

The *Receiver Operating Characteristic* (ROC) curve is a 2D chart reflecting the classifier's behavior on two chosen axes. The y axis measures sensitivity, also termed *TPR - True Positive Rate*, and the y axis measures the *FPR - False Positive Rate*. Similar to TPR for the true positives, FPR is defined as $\frac{|FP|}{|FP|+|TN|}$ for the false positives, which is equivalent to $1-$specificity. The curve represents specificity and sensitivity values of different classifiers obtained by using different thresholds. Regardless of a specific threshold, we can measure the quality of a classifier by the way it ranks the samples. We can do this by looking at pairs of samples, where one sample is negative and the other is positive. Ideally, the classifier would rank the negative sample lower than the positive sample for every such pair. In practice we can look at the probability that for a random pair, the classifier would rank the pair's negative sample lower than its positive sample. The area under the ROC curve (a value ranging from 0 to 1) measures exactly this - it is the probability that the classifier would correctly rank samples from a random positive:negative pair. The larger the area under the curve is, the better the classifier is, and thus the *AUC - area under curve* can be used as a measure for the performance of a given classifier as shown in Figure 2.2.

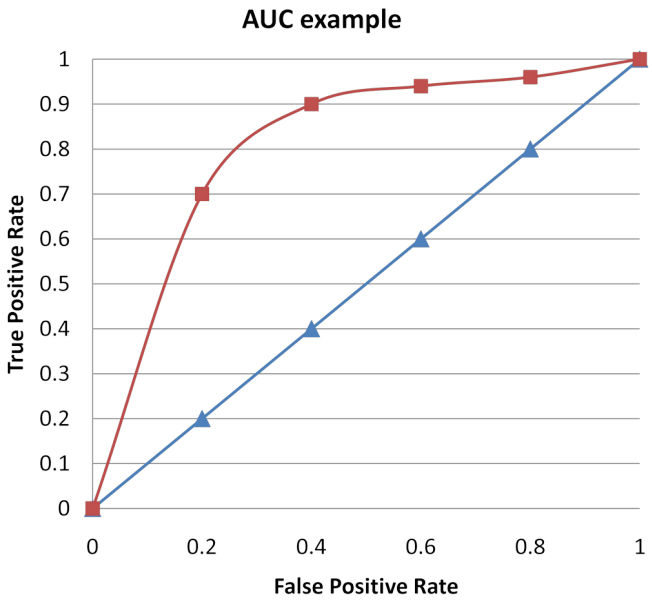


Figure 2.2: **AUC Example.** The curve presents the tradeoff between sensitivity and specificity (increasing sensitivity will be followed by a possible decrease in specificity). The blue diagonal (marked with triangles) serves as a lower bound to the AUC curve - an average behavior of a random classifier. The red curve (marked with rectangles) represents the performance of some classifier, reaching 70% true positives at 20% false positives and 90% true positives while maintaining under 40% false positive. The area under the random lower bound is 0.5. The area under the red curve integrates over the different tradeoffs of TPR and FPR to produce a comparable average performance measure. The closer the curve to the top left corner of the chart, the more accurate the classifier. The closer the curve to the diagonal, the less accurate the classifier.

2.1.3.2 Assessment of Generalization Performance

In order to assess how the results of a classifier will generalize to an independent data set, one needs first to train a classifier on some data. The trained classifier cannot be tested on the same data as its parameters were fitted using this data and may fail to generalize to new datasets while perfectly fitting the original data. This problem is known as overfitting.

Ideally, we would like to build a model by training a classifier on one dataset - a *training set*, and test its performance on another - a *test set*. However, due to shortage of data, one does not always have an independent data set on top of the original training data set. To overcome this shortage while still avoiding the bias, it is possible to resample the data - create multiple partitions of the samples. Each partition divides the samples into two subsets, one serving as a training set for the current partition, and the other as a test set for the current partition. Performance measures are taken for each partition independently, and the values are averaged over all partitions to produce a final value.

A number of strategies may be used for creating the different partitions. In *Leave-One-Out (LOO) cross validation*, the number of partitions is equal to the number of samples, and at each partition, one sample is left out for testing, while the rest of the samples are used as a training set. Another strategy used, called *bootstrapping*, randomly samples the dataset with replacement to partition the data, training on the sampled data and testing on the complement. In this strategy, samples may be sampled multiple times, while others might not be sampled at all. Throughout this work we used a strategy called K*-fold cross-validation* [41], which is described in the next section.

2.1.3.3 K-fold Cross Validation

In K-fold Cross Validation samples are randomly partitioned into K (approximately equally sized) subsets. Of the K subsets, one is set aside as the test set for testing the model, and the remaining $K-1$ subsets are used as training data. The process is repeated K times, where each time one of the K subsets is used as the testing data. Each run corresponds to a *fold*, and in total each subset is used as a test set exactly once. The K results are averaged to provide an unbiased performance measure for the classifier. 5 or 10-fold cross-validation are commonly used [48]. Eventually, estimation of AUC with

K-fold cross validation is used to estimate the performance.

2.1.4 Dimensionality Reduction and Feature Selection

Often, the data present a large number of features relative to a small number of samples. In these cases, it is difficult to build a good classifier that will generalize to an independent dataset based on the whole set of features. One of the problems is that due to the large number of features in such datasets one can find features that discriminate two classes of samples in the dataset merely by chance. Such features will only fit the given dataset, and will probably be worthless in other datasets.

In addition, real life data may include redundant features, and irrelevant features. It might be advisable to remove redundant, irrelevant and noisy features prior to learning a classifier, and rely only on a subset of the original features for the supervised learning task. The process of filtering out those features is called *Feature Selection*. In turn, it also reduces the dimension of the learning problem (*Dimensionality Reduction*) and helps improve the performance of learning models.

Dimensionality reduction is usually obtained by one of two approaches:

1. Feature Selection - selecting a subset of relevant features as mentioned above.
2. Feature Extraction - constructing a new set of features, each of which constructed from a combination of a number of original features.

Both approaches aim at solving some of the problems related to having a large number of features while building a model using machine learning:

1. Generalization improvement - with a large number of features, it is possible to build a model that is too tailored to the training samples, but performs poorly on newly exhibited samples (overfitting).
2. Performance improvement - by removing irrelevant and/or noisy features, it is easier for algorithms to build a model that is more relevant and accurate.
3. Computational efficiency - by reducing the size of the input to the learning algorithms, both time and space consumption can be reduced significantly.

Once a small number of features has been selected, only these features need to be measured. This can also be helpful in two more aspects:

1. Intelligibility - A combination of values from thousands of features can hardly be interpreted to give insight on the effect of each feature. On the other hand, a small number of features can help us to acquire better understanding as for our data, and serve as a good lead for downstream research by letting us know which are the important features and how they relate to each other.

2. Cost - It is generally cheaper to collect and analyze a small number of features.

In the next subsections we will cover families of feature selection algorithms and briefly describe a few methods that are used throughout our work.

2.1.4.1 Filters

Filters [28] comprise a family of feature selection methods where features are considered one by one, and based on a predefined criterion, a feature is either kept or filtered out from the set of features. The criterion is often related to a feature's relevance, and thus only the most relevant features are selected. Filters are usually simple to implement and employed prior to learning the classifier so they are independent of the classification algorithm. They are often based on ranking the features by assigning each feature a score independently of other features, and as such they are scalable in the number of features. Their main downside is that they are univariate - that is, each feature is considered separately, independent of all other features, although such dependencies are possible. As a result, they do not account for redundancy in the feature set, or for advantages of using multiple features together, but merely for a single feature's relevance.

2.1.4.2 Student's t-Test

One of the simplest filters available for binary-class data tests for the difference between two means in the feature value in the populations. It is a statistical hypothesis test where the null hypothesis is that the means of the feature values in the two populations is equal. It is based on the assumptions that feature values follow a normal distribution with

equal variance in both populations, and that samples are independent from each other. For a specific feature and two given vectors X_1 and X_2 corresponding to the feature values in the two populations, the t-statistic is defined as:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{S_{X_1X_2} \cdot \sqrt{\frac{2}{n}}}$$

where

$$S_{X_1X_2} = \sqrt{\frac{S^2_{X_1} + S^2_{X_2}}{2}}$$

is an estimator of the common standard deviation of the samples, and $S^2_{X_i}$ is the variance of X_i.

The t-statistic values follow the *Student's t-distribution* and so statistical significance (p-value) can be calculated for each value. In classification, t-Test is only used to rank the features, and one needs to choose a cutoff point in order to choose the highest scoring features. This may be done either by setting a constant number, or using a p-value cutoff, or determined based on the data using cross-validation techniques.

2.1.4.3 Embedded Selection

Embedded feature selection methods rely on a specific learning algorithm, and use the model provided by it to reduce the original set of features. For example, any linear discriminant function assigns a coefficient (or weight) to each feature during the learning phase. Features that are assigned a near zero weight are meaningless to the classification, and thus can be discarded.

Another family of methods, called *wrapper* methods, is characterized by an iterative process, in which a subset of the features is selected, and a learning algorithm is utilized as a black box to evaluate the selected subset and sometimes guide the selection of a better subset for the next iteration. At each iteration one subset is evaluated, and the process may stop with the current subset, or change it in order to try and improve the classification performance. The selected subset of features, whether obtained by an embedded or a wrapper method, is not limited to be used with the specific embedder or wrapped classifier that created it.

We will now present two feature selection methods that are based on SVM. Later on, we shall show an adaptation of one of these methods such that is uses prior knowledge for the selection process.

2.1.4.4 L_1-norm SVM

SVM can be formulated as an optimization problem with a single objective function, incorporating a loss term and a penalty term (In Section 3.6.3 we shall see one method that utilizes this type of formulation for network integration purposes). In the standard (L_2-norm) SVM, the penalty keeps the weight vector norm minimal, in order to minimize the margin:

$$\min_{\mathbf{w},w_0}\left\{\sum_{i=1}^{n}[1-y_i(\mathbf{x}_i^T\mathbf{w}+w_0)]_+ + \lambda\|\mathbf{w}\|_2^2\right\}$$

An alternative is offered by [82], where the L_2-norm penalty is replaced by L_1-norm

$$\min_{\mathbf{w},w_0}\left\{\sum_{i=1}^{n}[1-y_i(\mathbf{x}_i^T\mathbf{w}+w_0)]_+ + \lambda\|\mathbf{w}\|_1\right\}$$

L_1 loss is argued to outperform L_2 when the underlying model is sparse - that is, it only relies on a subset of the features, and in fact the result of the L_1-norm SVM optimization often sets the weights of uninformative features to 0, and can be considered as a feature selection method.

2.1.4.5 L_2-AROM SVM

It is possible to explicitly change the SVM objective function in order to minimize the number of features with non-zero weight. This is termed zero-norm, and defined as $\|\mathbf{w}\|_0^0 = card\{w_i|w_i \neq 0\}$. However, this problem is NP-Hard [4], and an alternative problem that approximates it was offered by Weston et al. [79] together with a solution they call *AROM* (L_2 ***A**pproximation to ze**RO** norm **M**inimization*). Instead of the original SVM objective function

$$\min_{\mathbf{w},w_0} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 \right\}$$

they use the objective function

$$\min_{\mathbf{w}} \left\{ \sum_{i=1}^{p} \ln(\epsilon + |w_i|) \right\}$$

with $0 < \epsilon \ll 1$. This is solved by a simple iterative procedure wrapping the standard L_2-norm SVM.

2.1.4.6 SVM RFE

Guyon et al. [29] offer a simple wrapper method for SVM that uses Sequential Backwards Search (SBS) through the space of feature subsets. They iteratively train L_2-SVM, and use the weight vector to eliminate those features that exhibit the lowest weights according to the model (one or more feature at each iteration). After this elimination, the SVM is trained again, this time with the remaining features only, and the process is repeated until a stopping criterion is met. They name the method ***R**ecursive **F**eature **E**limination*, and compare the final set of features using different types of classifiers. One of their results is that the set of selected features influences the final classification performance more than the actual classifier used.

2.2 Biological Background

In this section we give basic biological background relevant to the classification of gene expression profiles. For more on basic biological notions see, e.g [2]. For more details on expression microarrays see, e.g. [8].

2.2.1 Proteins

Proteins are the main building blocks of cell components and molecular machinery acting within the living cell, and are also used as communication means between and

within cells. At any given moment, proteins take care of maintaining cell form and activity, allowing the cell to cope with external environment, and some of them are used for *signaling*, secreted outside and used within the cell upon demand. There are many types of proteins, so cells of different tissues may hold different mixtures of proteins, and even within the same cell, the protein types and amounts (or *concentrations*) may change from one time to another, depending on its internal state and on outside conditions (or stresses). This is the reason that snapshots of the state of proteins in a cell, such as the concentration of each type of protein, can provide us with a lot of information about processes within a cell and its environment. Even with incomplete understanding of the cell's state and behavior, such a snapshot could help us differentiate one type of cell from another, or, for example, help us differentiate between cells with different levels of malignancy. Today's technologies allow only very partial snapshots of the proteins in a given tissue, ranging from hundreds to a few thousand protein species [80]

From a biochemical point of view, each protein consists of a chain of basic molecules called *amino acids*. The sequence of amino acids differentiates one type of protein from another. The chain of a specific protein is folded into a unique structure, which gives the protein the ability to function. It also sets the protein's ability to bond to other proteins and create *complexes* of proteins. One of the main complexes within a cell is the *ribosome*, which is responsible for synthesis of new proteins when those are required.

2.2.2 Protein synthesis

In order to synthesize a protein, the ribosome uses a basic "recipe" or "blueprint", which tells the ribosome at each point which amino acid it should insert next into the chain. This recipe is written as another chain of chemical units called *nucleotides*. There are four such nucleotides: Adenine, Cytosine, Guanine, and Uracil, abbreviated as A, C, G, and U. A sequence of three nucleotides encodes for exactly one amino acid and a chain of nucleotides is called *RNA*. A *messenger RNA*, or *mRNA*, is a chain of nucleotides that encodes a protein (the recipe of the protein) and the process of synthesizing a protein from mRNA is called *translation*.

The source for RNAs is a much longer chain (or chains) of nucleotides, called DNA. The DNA encodes all required RNAs and subsequently all required proteins of a species.

A *gene* is a part of the DNA chain that codes for a specific protein. Upon demand, parts of the DNA chain are exposed and copied into RNA chains in a process called *transcription*. Hence, a gene is transcribed to RNA and the (mature) RNA is translated into a protein. On a first approximation, the DNA sequence uniquely determines the protein sequence. Like other cellular processes, transcription is also governed by proteins, named *transcription factors*, which regulate the process.

2.2.3 DNA microarrays

It is possible to estimate the concentration and type of proteins within a cell at a given moment by looking at the mRNAs currently present inside the cell. While high-throughput ways to measure and quantify all proteins within a cell are still under development, it is already possible to measure the concentration of all RNA molecules in the cell using *DNA Microarrays*, and use these measurements to estimate current activity of all genes (and indirectly their protein products) in a given point of the cell's life.

A DNA microarray is a small solid surface consisting of thousands to millions of microscopic spots of DNA oligonucleotides called *probes*. Each probe can *hybridize* (bind) to a specific RNA fragment, according to its nucleotide content. In a specific design, the oligonucleotides of each probe are chosen so that they can hybridize to a single gene's mRNA. Extracting RNA material from the cell and creating physical contact between this material and the surface of a microarray allows for the probes to hybridize to the RNA that was present in the cell at the time of extraction. Later on, the microarray is scanned, and the amount of RNA bound to each probe is measured. This allows for measuring the activity levels of all genes within a living cell or tissue, and quantifies the expression level of each gene. A *gene expression profile* is the set of all levels associated with all genes. As mentioned before, the gene measurements are valuable for estimating the levels of active proteins within the cell, providing a lot of information regarding the cell's state.

2.2.4 Next Generation Sequencing

Today, a new generation of DNA sequencing technologies called Next Generation Sequencing (NGS) or Deep Sequencing is being developed. These methods read millions

of short DNA chains. These chains can be compared to a known sequence of a *reference genome* that is close to the one being sequenced. For a short review of these methods and additional references see [62]. Lately, these methods were applied also to RNA sequencing (a technique called RNA-Seq), allowing for high throughput measurement of gene expression profiles. Producing such profiles is becoming cheaper and more accurate, but the challenges of analyzing these profiles are the same as in microarrays. All methods and algorithms covered throughout this work are also applicable to expression profiles obtained by next generation sequencing.

2.2.5 Classification of Gene Expression Profiles

Using a gene expression profile, the genes (or probes) can be used as features for the supervised learning task. Hence, the profile of an individual in a classification task will be a vector of circa $25,000$ features (the number of genes in the human genome), or even more if several probes per gene are used in the mircorarray. In a binary classification task, two sets of such profiles correspond to samples from two populations (classes), a *case* class and a *control* class. Each sample is labeled, i.e., is known to belong to one of the classes. The objective is to create a model using supervised learning, such that given a new unlabeled gene expression profile, it can predict whether its sample belongs to the case class or to the control class.

Classification of gene expression profiles is already being used successfully for various clinical purposes. One of the most striking uses is for differentiating breast cancer patients who may benefit from chemotherapy from those who are likely to overcome the cancer following only surgical intervention. This is done by using tumor mRNAs from databanks that follow patients for a long period after the tumor extraction, and dividing the patients into two classes according to their prognosis, as measured by their survival times. Since chemotherapy itself is very toxic, and has a lot of adverse effects, physicians would like to treat with chemotherapy only patients whose state truly requires it.

Gene expression data that are related to good or poor prognosis actually differentiate quite accurately tumors that are more aggressive from less aggressive ones. Visionary researchers kept biopsies from tissues for years after they were taken. Eventually, they could associate a specific person's tissue with his or her condition months or years

after the biopsy. This way two sets of tissues were gathered, corresponding to two patient populations - those who survived after a given time, and those who, unfortunately, did not. Tumors of those who were lucky enough to survive without being treated with chemotherapy indicate that chemotherapy is not required for the body to eliminate them, and a classifier based on these data can predict the need for chemotherapy based on a gene expression profile of a patient's tumor.

Gene expression classification has been researched for more than a decade now for various classification tasks. Some early examples are Alon et al [3], classifying colon cancer types, and Golub et al, who used it to classify leukemia subtypes [27] in 1999. Later, Singh et al [65] applied it for distinguishing prostate tumors from normal samples.

A large portion of the research was devoted to breast cancer, including pioneering research by van 't Veer [72], followed by van de Vijver [71], and Wang [75]. In most cases, feature selection was employed, and often specific feature selection methods were developed for the specific task. Prior to gene expression profiling, oncologists followed consensus criteria based on tumor size, presence of hormonal receptors on the tumor cells' surface, and patient age, for determining whether a patient required chemotherapy or not. These are called St. Gallen criteria in Europe, and the US counterparts are called the NIH criteria.

van 't Veer et al. showed that while between 82% to 92% of the patients were eligible for chemotherapy according to the consensus criteria, only 55% of them were eligible according to their gene expression profile and the classifier criteria. That method correctly identified more than 91% of the poor prognosis women, a fraction comparable to the consensus criteria. Still overall, the method had only 27% false positives, compared to $70\% - 90\%$ false positives of the consensus criteria. The methods have been commercialized and validated in a number of followup studies (see e.g. [56]).

3

Integration of Prior Knowledge

During the recent years, several methods have been proposed for integration of prior knowledge into gene expression analysis. In this chapter, we will review a couple of them, trying to emphasize the basic differences between them. A number of preliminary questions have to be answered before integrating prior knowledge into gene expression analysis, and the different methods differ in the answers they provide to these questions. Posing and answering these questions will help us in creating a taxonomy of the methods, allowing us to select the integration method path we would like to take.

First, in Section 3.1 we will ask ourselves why should we try to integrate prior knowledge at all. Following that, in Section 3.2 we will discuss the limitations of current methods. In Section 3.3 we will discuss the influence of analysis goals on choosing the basic integration model. After that, we shall review different types of available prior knowledge in Section 3.4. We shall cover the basic domain-specific assumptions of prior knowledge integration in Section 3.5, followed by a review of a few extant methods that directly influence our work in Section 3.6. Section 3.7 presents a taxonomy of the properties of the alternative approaches.

3.1 Motivation

Before the era of DNA micorarrays and high throughput gene expression analysis, experts could measure quantities of only a small number of proteins in a cell in one experiment. The expert would choose the proteins suitable for the experiment, and later analyze the results. With high throughput experiments, we are being led by the data. When designing experiments, no choice is made for individual genes will be measured.

With thousands of measurements for every sample, even an expert would find it difficult to analyze the output of such an experiment without the aid of data mining methods.

The basic uses of data mining and machine learning for clustering and classification of high throughput gene expression data, discard a lot of prior biological knowledge, e.g., the gene functions and the relations among them. Genes are treated merely as indices, and no underlying structures known from biology are used. This is somewhat similar to analyzing text as a long string of characters (or tokens) without using any knowledge of the syntax and semantics of the language.

All this would not have bothered us if not for the frequent problematic results of such an analysis. For example, when looking for differentially expressed genes, even following established statistical analysis practices, one is often left with hundreds of genes, a problem known as the *"gene list syndrome"*. These genes might be suitable, up to a certain level, for classification, but the list is often too long, difficult to interpret even by an expert, and contains too many candidate genes for downstream research.

The difficulty in finding meaning and interpreting such gene lists can be shown in the differences among results of different classification studies. Although used on the same type of data and the same disease, these studies often come up with a good set of genes in terms of classification performance, but surprisingly, the overlap between the sets of genes selected as features in different studies is very poor. For example, the overlap between two breast cancer studies, each providing about 70 genes, was only 3 [20]. In addition, the genes selected in one dataset do not perform well on the other datasets [17].

We would like then to find a list of genes that is more meaningful biologically, and thus more useful for downstream biological research. These aspirations were also raised by other researchers, aiming at integrating network data into gene expression analysis. Rapaport et al. [59] state that their motivation is to achieve biologically significant classifiers, and use network data for analysis purposes rather than for a more traditional purpose of statistical validation of results. They also mention that gene expression data were used for network inference in prior works, but there were not many works taking the other direction, using network data for analysis of gene expression. Chuang et al. [17], the authors try to find prognostic markers for cancer that are both more accurate in predicting cancer metastasis, and more directly related to the disease.

The two incentives, namely finding genes that are more directly related to disease, and finding genes that are more accurate in prediction, do not always go hand in hand. Hwang et al [32] provide a list of four genes that are highly related to breast cancer, but were not detected as differentially expressed by assays looking for breast cancer prognostic markers such as van 't Veer's seminal work [72]. There are many biological reasons why a disease's causative gene may not necessarily be highly differentially expressed between the control and case groups. It is possible that a relatively small change in the expression of the gene cascades to larger changes in expression of a large number of genes downstream of this gene. In other cases, a mutation in the gene might cause no production of its protein, or the production of a defective protein. At the transcriptional level, high levels or mRNA are detected both in the case and control groups, and malfunctioning of the protein in the case group can not be detected.

Another reason for the poor overlap and reproducibility in differentially gene expression analysis, is the inherent noise in the basic measurements of mRNA levels using microarrays. Expression levels are highly sensitive to specific probe affinities, and are influenced by differences in platforms and protocols. Due to the large number of probes and the relatively small number of samples it is difficult to eliminate the noise by known methods for noise reduction. This point was demonstrated by Ein-Dor et al. [20] on the same dataset of [72].

3.2 Limitations of Current Methods

Ein-Dor et al. [20] showed that statistically, even after eliminating possible reproducibility difficulties such as different platforms and protocols, the number of genes that correlate with desired phenotypes (such as breast cancer metastatic recurrence) is much larger than the size of reported subsets. They did this by examining only a single dataset [72], and analyzing different subsets of its samples. Of 5852 genes, they found 1234 to be significantly correlated with the phenotype (FDR$< 10\%$).

They ranked all genes by their correlation with the phenotype, and partitioned them into subsets of 70 genes, where the first set contained genes ranked 1 to 70, the second contained genes ranked 71 to 140, and so on. This way they obtained a large number of disjoint subsets of genes which they used as features. They trained many classifiers by

the same classification algorithm, each time using a different subset of the samples, and a different subset of 70 genes. Fluctuations in performance as measured by the number of errors on a given test set were found to be mainly due to the different sample subsets selected for training, and were less influenced by the subset of features. The performance of the 10 classifiers resulting from training on the top 10 subsets of genes was also comparable.

In another test, they showed that some genes ranked low in their correlation with the phenotype (with rank larger than 1000) had non negligible probability (> 0.05) to be selected in the top 70 genes, given a random subset of the samples. Eventually, they state that searching for one "master gene" regulating the phenotype is impossible using the current methods and only a few hundreds of samples, but prognostic tools used for classification can reach fairly good results using many subsets of different genes. Their suggestion is to overcome this problem by enlarging the sample size, or dividing the sample size in advance into known homogenous subsets based on some prior knowledge, and to analyze each subset separately (as done by [67]).

In a later paper, Ein-Dor et al. [21] calculated the number of samples required in order to reach a desired overlap on two different studies using an analytic method. As an example, they give two breast cancer studies, stating that thousands of patients are required in ordet to achieve a typical overlap of 50% between two predictive lists of genes.

Another limitation of many standard classification methods is the difficulty to interpret the resulting classifier and its internal behavior. This limitation is presented and tackled in various papers (e.g. [24]). A linear or non-linear combination of more than a few features using feature weights fitted by some learning algorithm is hard to interpret. Often, algorithms output weights of tens to hundreds of features, and although the resulting classifiers may perform well, they are treated as black boxes and their internal logic can not be followed. The problem also appears in other classifiers such as neural networks, and ensembles of decision trees, where although one can follow each set of decisions given a single tree, it is hard to justify the logic of this tree, not to mention the case where voting is used among a large number of such trees. The solution as offered by Geman et al. [24] is a new classification algorithm that is looking at pairs of genes, and has some properties that make it suitable to underlying biology of gene expression

profiles, but do not include any prior external data in the learning process.

To sum up, the reasons for integrating prior knowledge into gene expression analysis, and specifically into detection of differentially expressed genes, are to increase the results' relevance to disease and to improve prediction performance. This may lead to better reproducibility of experiments and analysis, and thus to higher acceptance of such methods by traditional biologists, and to the ability to intelligibly interpret the results and use them for better downstream research.

3.3 Analysis Tasks and Integration Model

Common gene expression analysis tasks include discovery of differentially expressed genes, model creation for gene expression profiles classification, clustering of genes or profiles and bi-clustering of genes and samples. More basic tasks would be noise reduction and data normalization. Integrating prior knowledge into one of those basic tasks usually results in transforming the data in a way that can later be used to solve any of the more advanced tasks, such as classification or clustering. Alternatively, we can choose a specific advanced task and integrate the prior knowledge in a way that is tailored for this task. Among the basic tasks that may be considered as preprocessing steps, a large class of tasks deal with dimension reduction, as. described in Section 2.1.4.

Advantages of knowledge integration in basic tasks could be integration simplicity, and the use of the method for various advanced tasks. On the other hand, integration tailored to one of the more advanced tasks could result with better performance in this specific task. For example, general purpose noise reduction ignores the known samples' classes in a classification task. This might mean that we put effort in noise reduction on data that are irrelevant to our task.

3.3.1 Downstream research

In some cases, one of the research long-term goals is to find leads for deeper downstream research. While this task is important, the target here is somewhat vague, as it is hard to quantify what is a good gene for downstream research. Often, genes that seem to hold informative expression patterns related to the task in hand are also good candidates

for additional research. However, gene expression is only one manifestation of protein activity. There are cases where proteins that are known to play a central role in some disease are not detectable based on their gene expression. Some works state the discovery of disease-related genes as one of the motivations behind the classification task [32], while others focus on forming an accurate classification regardless of the contribution of the classification features in downstream research. There are also studies that explicitly seek for disease-related genes based on gene expression regardless of their actual contribution to the classification task [55, 73, 38].

3.3.2 Use of Statistical Modeling

When performing prior knowledge integration, one can assume an underlying distribution of the data. One option is to assume a model generating the data, which includes the prior knowledge as parameters. This option can fit in basic analysis tasks such as noise reduction. Modeling, however, is not limited to the actual expression levels and the basic tasks. It is also possible to assume an underlying model with hidden variables corresponding to one of the advanced tasks, such as the cluster to which a gene belongs, or the differential state of a gene. Such a model is used, for example, by Wei and Li in [78]. Wei and Li extended a statistical model for the distribution of gene expression values, by adding hidden binary variables for gene states that are either differentially expressed or equally expressed between two populations. This way, their model is suitable for differential expression analysis or supervised classification.

The other integration option, without a statistical model, covers a large range of heuristics. These include greedy search guided by the prior knowledge as performed, for example by Chuang et al [17] and Lee et al [44], or direct optimization, using the prior knowledge as constraints, as formulated by Hwang et al [32] and Tian et al. [70].

Although statistical modeling allows for more rigorous solutions, the model itself often oversimplifies the real nature of data, and does not fit the real data distribution. Inaccurate independence assumptions, binarization of states and bad choice of probability density functions may lead to poor results, and the choice whether to use a model or not should depend on the nature of our problem. In the scope of classification, there is a common distinction between two types of classifiers. The first type includes classifiers

that learn a model that explains how to generate random samples conditioned on the target class and use this model for classification (*generative* models). The second includes those classifiers that learn the distribution of the classes given the training samples, or even learn a distribution-free direct map from samples to their classes (*discriminative* classifiers). The modeling question is somewhat similar to the choice of a generative model vs. use of discriminative classifiers. In our case, the feature dimension is very high relative to the number of samples, and it has been shown that generative models-based classifiers can outperform discriminative classifiers in such situations [54].

The choice whether to use a statistical model or another integration approach is tightly linked to the available prior knowledge, which will be covered in Section 3.4 and to the assumptions regarding the linkage between this knowledge and the expression data, as presented in Section 3.5

3.4 Available prior knowledge

Due to the requirement to automatically analyze the expression data, without manual aid of an expert, we would like to use prior knowledge that is already encapsulated in data that a computer can utilize. Such data should be simple enough so they could be used in a broad manner, capturing biological facts and notions that are valid across domains, and are not too specific. As our task is analysis of gene expression profiles, this data naturally have to do with genes. With this respect, the data can either relate to specific genes, to the relations between genes, or both. Luckily, today's bioinformatics community offers a number such resources, capturing years of biological research in a compact representation that can be used by computers.

One can distinguish between annotation based repositories that annotate individual genes, and network based repositories that link pairs of genes according to various types of relations.

3.4.1 Annotation based repositories

The most widely used gene annotation database is the Gene Ontology, or GO [7]. Roughly speaking, it annotates genes by three categories: Biological Process, Cellular

Component, and Molecular Function. Each category holds thousands of terms, usually grouped in a hierarchical tree structure. The more general the term, the higher it is in the hierarchy. The terms and annotations are gathered by groups of professional curators from published scientific papers. The GO annotation is a suitable candidate for automatic integration of prior knowledge into gene expression analysis, as one can use the terms' identifiers and their hierarchical relationships for her analysis.

GO is frequently used for statistical coherence validation of gene sets, e.g. in the TANGO algorithm [64]. If a large fraction of the genes in a set share a common function, it is a good indicator regarding the set's coherence.

3.4.2 Network data

Network data represents relations between elements, such as genes or proteins. Formally, a network is a graph $G = (V, E)$, where V is the set of nodes (e.g. genes) and E is the set of edges (relations). Edges can be directed or undirected, they may be weighted and it is possible to have different type of edges, corresponding to different types of relations. We shall look at two types of biological networks that capture prior biological knowledge: large scale networks, and pathways.

3.4.2.1 Large-scale networks

Large scale networks contain a large number of elements, with a good coverage of most possible elements. They hold diverse elements, including many that were not deeply researched, and thus, such networks usually contain many elements with few details, and are not very accurate.

Protein-Protein Interaction networks Originally, protein-protein interaction (PPI) networks described physical relations between proteins [33, 68, 40, 5, 61]. That is, each node in the networks represents a protein, and an edge connects two proteins if the two proteins can physically interact. The physical interactions were obtained through a series of experiments where each protein is tested for interactions with a large number of other proteins, and observed interactions are assigned a confidence level. This way high coverage is obtained, but the only available level of detail is the possibility of two proteins to interact.

Reasons for low accuracy include in-vitro experimental results that might differ from in-vivo behavior, and the stochastic nature of protein interactions related to the wide range of bonding affinities. The semantics of an edge can be phrased then as ”protein A and B might interact”. It does not follow that they do interact, and if A and B might interact and B and C might interact, it does not follow that the two interactions occur under the same condition.

Later on, these networks were expanded using more sophisticated high throughput methods, where each experiment tested for multiple proteins interactions. Next, other types of relations were considered [36, 66], among them co-citations of proteins, where an edge is drawn if two proteins are mentioned together in scientific texts, and functional similarity, where an edge is drawn if two proteins putatively share the same function as predicted by function prediction algorithms. STRING [36, 66] is an example for such integrated network developed since 2000 that is also used throughout our work.

3.4.2.2 Small-scale networks

As seen, PPI networks semantics is pretty poor. In order to come closer to the cell's behavior at the genomic level, one needs a richer and more accurate representation model.

Gene Regulatory Networks (GRN) These include both genes and gene products as their basic nodes, and describe signalling relations between these nodes, controlling the production of proteins and protein complexes within the cell. Often they also include external inputs that trigger specific signalling pathways. As these networks describe steps in ongoing process, their edges are directed, and may describe a regulatory relation of either repression (down-regulating) or induction (up-regulating) of the target proteins.

Gene regulatory networks are related to gene expression in a straightforward manner, as gene expression microarrays measure a snapshot of gene levels within a cell, while a GRN can provide for the relations between the different levels. Although this relation is easy to understand, one should note that there are timing issues that such a relation neglects. Since usually a microarray reflects a single time point in a cell's lifecycle, it might so happen that it catches high level of targets, while their regulators are already back to their basal level. Nevertheless, for a steady-state regulatory system, when it is

relatively stable, it is possible to measure gene expression levels that reflect the underlying regulatory network.

The timing issue is not the main drawback of such networks when planning to integrate them in the analysis of large scale gene expression data. Because these networks hold much more detail, they are usually much smaller than PPI networks, and often, if not always, incomplete. There are small pathways (holding dozens to hundreds of genes) that were deeply investigated, but as of today, a large-scale regulatory network for human is not yet available. Instead, we have a good (but sparse) collection of small networks covering specific processes, or such that all their genes relate to a specific subject, or disease. SPIKE [22] introduced by Elkon et al. includes such a database, together with data manipulation, algorithmic analysis and visualization tools.

The KEGG pathways database [37] is a widely used example for such a collection of small pathways, including many regulatory pathways. Among others, it holds pathways related to human diseases such as various types of cancer, cellular processes such as apoptosis (one type of programmatic cell death), and a pathway of the yeast cell cycle. See Figure 3.1 for an example. KEGG also contains pathways of basic genetic information processing units such as RNA polymerase, and also pathways related to metabolism and to drug development. The database also contains genes annotation according to the various pathways they belong to, and can be used is similar ways to the GO annotation mentioned above.

Metabolic Networks. Similar to gene regulatory networks, metabolic networks are also comprised of set of pathways. A pathways comprises of a sequence of chemical reactions occurring within a cell, describing the step-by-step modification of a chemical to another, by a set of reactions that involve some other molecules. As mentioned in the former paragraph, the KEGG database includes metabolic pathways too, and some of the pathways can include both regulatory and metabolic reactions.

It is worthwhile to note that some highly investigated species, such as the *Saccharomyces Cerevisiae*, already have a large number of both regulatory and metabolic pathways that together cover a large fraction of their genomes, and can be considered as a large-scale network.

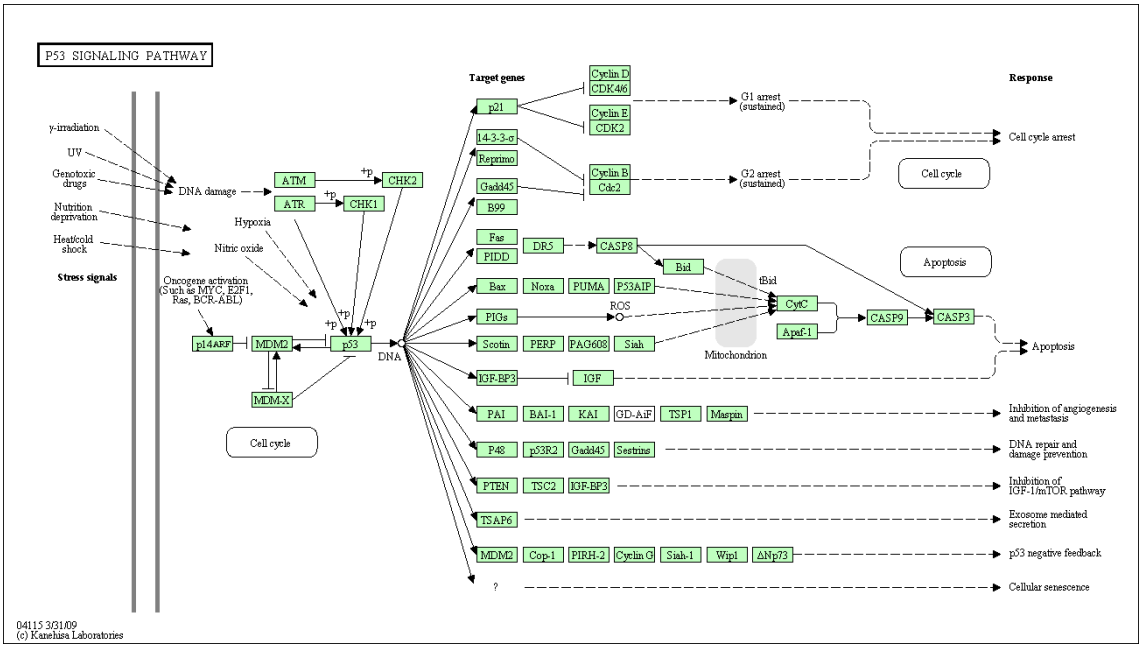


Figure 3.1: **KEGG p53 signaling pathway**. An example of a partly known regulatory network related to the p53 gene, playing a central role in DNA damage repair taken from the KEGG pathway database [37]. Semantics include data about suppression and activation on the RNA and DNA level, external stimulus, and relation to other cellular processes. Taken from `http://www.genome.jp/kegg/pathway/hsa/hsa04115.html`.

So far, we have touched several aspects of prior knowledge integration into gene expression analysis: the analysis purpose, its influence on the modeling choice, and the available knowledge source. We covered two types of available data - annotation sources - ontologies, and relations sources - networks, and looked at two main categories of networks, namely simple large-scale networks, and more detailed small-scale pathways collections. Next, we will try and link the prior knowledge to the expression data, focusing on network based data from now on.

3.5 Basic Integration Assumptions

In this section we summarize the assumptions that we make regarding the relation between gene expression and the network. Our analysis will be based on these assumptions.

Close Genes co-Express

Our first assumption is that genes that are close on the network are likely to have similar expression. This assumption is made for example by Rapaport et al [59], who subsequently hypothesize that noisy measurements of gene expression can be denoised to some extent using the network. They give PPI networks and metabolic networks as an example, and use the *Saccharomyces cerevisiae* metabolic network in their experiments. They justify their choice of a metabolic network, with data that show co-regulation (and thus co-expression) of enzymes required for the same metabolic process.

Using the overall network topology they relax a more strict assumption used before, regarding pathways. The stricter assumption states that genes that are members of the same pathway tend to co-express, treating all genes of a pathway as equal, ignoring the pathway's intra-topology. Apart from adding topological consideration within a pathway, the relaxation overcomes another limitation of the stricter assumption. Pathways may overlap and by considering each pathway separately, the dependency due to this overlap is ignored. The relaxed assumption is key to to analyzing large-scale networks and not only individual pathways.

Although this assumption has been validated to some extent, for example by Jensen et al in [35], it is questionable, especially regarding regulatory networks. Since regulatory

genes can either activate or suppress the transcription of other genes, overexpression of one gene would not necessarily lead to overexpression of a neighboring gene if the former is suppressor of the latter.

Close Genes have Similar Differential Expression

A gene is called differentially expressed if its expression level varies markedly between two sets of samples. Two sample sets can differ, for example, in their phenotype, (e.g. normal-disease, or case-control), the physiological or genetic state of the ces (e.g. starvation or gene knock-out), or in the time they were sampled, (e.g. samples from the same tissue taken in different times). The above assumption states that close genes tend to have similar differential expression pattern: they tend to change or not to change together among the two sets of samples.

Such an assumption, together with the criticism on Rapaport's assumption is presented by Wei and Li at [78]. Although relaxing the co-expression assumption, the drawback of this assumption is that it requires labeled data, and thus is suitable for classification or differential expression analysis, but not for basic tasks such as normalization or noise reduction or for unsupervised tasks such as clustering.

Expression Data reflects Materializations of Interactions

This assumption states that a given regulatory pathway describes the possible interactions among genes, but in a state reflected in real data, an interaction can either occur or not. Efroni et al. [19] say that an iteraction *materialized* if it actually occurred in a s specific sample or set of samples. Such an assumption is closely related to stating that genes in a known pathway tend to change together, so it can capture differences between pathway behavior among two sets of samples. However, this assumption captures more than common change of genes in a pathway: a pathway can generally be active in one state and inactive in another, or, depending on the state of a specific gene in a pathway, the consistency level of the pathways with the expression data may differ among the two sets of samples.

Looking only on the difference of the overall activity levels of genes in a pathway

among two states ignores the internal topology of the pathway, but Efroni et al. overcome this by looking specifically at interactions, and also take into account the difference in consistency levels of a pathway among the two states. However, since the method is tailored to regulatory networks, and large regulatory networks are currently still incomplete, it is currently applicable only for small-scale pathways.

3.6 Prior Work

The actual way the assumptions we reviewed in Section 3.5 are utilized vary from one work to another. For example, the level of differential-expression of a gene can be treated as a binary variable, or as a continuous one. Assumptions on pathways or subnetworks can be formalized as a set of pairwise assumptions on all pairs of the pathway's members, or on neighboring pairs only. In the next subsections we will review several existing methods that tackle this data integration task, emphasizing the differences between them.

3.6.1 Greedy Search for Subnetworks as Markers

Two works from the lab of Trey Ideker at UCSD, one by Chuang et al. [17] and one by Lee et al. [44], offer to substitute the use of expression levels of individual genes with an aggregate function of the expression levels of a set of genes. In their algorithm, *pinnacleZ*, they offer to constraint such sets using network topology [17], or known pathways [44] and provide a method for greedily searching for such subnetwork markers within the network. They use cancer data, differentiating between tumors that eventually turned metastatic, and those that did not, and show improvement in performance of classification of such tumors using the subnetwork markers.

The authors offer a statistical analysis for determining the significance of the subnetworks they discover, and show that individual genes that are known to take function in cancer processes (or *Hallmark* genes) are included in part of significant subnetwork markers, although they are not significantly differentially-expressed as individual markers. Finally, they show that network markers are more reproducible across datasets, and offer new putative genes discovered in significant subnetworks as related to cancer.

The basic use of a subnetwork as a marker simply takes the normalized ($\mu = 0$ and

$\sigma = 1$) expression levels of all the genes in the subnetwork, and averages it. Let z_{ij} be the standardized expression level of the i'th gene in the j'th sample, where each gene is normalized to have mean 0 and standard deviation 1 across the samples. Let M be a set of gene indices representing a subnetwork of size n. they define the *activity score* of M for the j'th sample to be

$$a_j = \frac{\sum_{i \in M} z_{ij}}{\sqrt{n}}$$

The square root in the denominator is compensating for the fact that we use normalized expression values rather than the actual expression values.

Chuang et al. search for subnetworks whose activity scores are highly correlated with the labels (that is, have high activity score throughout the samples of one class, and low activity score throughout those of the other class). They use either *Studendt's t-test* or the *mutual information* score for measuring this correlation, and call this correlation the *discriminative potential* of the subnetwork. In order to find such subnetworks, they build them by adding one gene at a time. Starting from an initial *seed* of a single gene, they look at all its neighbors, and add the one that adds the most to the discriminative potential. They continue doing so until a stop criterion is met, and repeat the process multiple times starting from different seeds. This way, for each seed they end up with a subnetwork with maximal discriminative potential. Next, they score the significance of each subnetwork, using three different tests and only those networks that passed all three significance tests are kept for later use.

Testing their method on the Wang [75] and van de Vijver [71] breast cancer datasets they select a set of features using one dataset, and report AUC scores using cross validation on the other dataset. Selecting the top discriminative single-gene markers on the Wang dataset by the method of [75] and testing on the van de Vijver dataset, they reach AUC of 0.66. For their method, selecting subnetwork as markers, they report AUC of 0.72. For the reciprocal test (selecting on van de Vijver using the method of [71] and testing on Wang) they improve from 0.59 to 0.62. They also increase the fraction of overlapping genes selected on the two datasets from 7% in individual genes, to 13% in genes that are members of selected subnetwork markers.

In a similar manner, Lee et al. start by obtaining pathways known to be related to the study from the MSigDB C2 curated gene sets [69], sort their gene members by their *t*-test

score, and greedily add the top ranked genes, one after another until the discriminative potential of the subset no longer increases. They term their resulting sets as ***CORGs*** - ***CO****nditionally* ***R****esponsive* ***G****enes*.

The authors report a significant improvement in AUC scores of pathway markers over individual gene markers in 6 of 7 datasets tested. For example, in one lung cancer dataset [9] they improve from 0.59 to 0.69, and in another [10], they improve from 0.5 to 0.52. Their improvement in the breast cancer datasets mentioned is comparable to the one reported by Chuang et al.

Both processes are actually feature extraction methods, and the final subnetworks can be used as features for various machine learning tasks. Given a gene expression profile, one obtains a set of feature values corresponding to the extracted subnetworks activity scores, and use the subnetworks as features instead of using individual genes. For every sample we now have a subnetwork expression profile instead of a gene expression profile.

The two methods share a similar algorithmic approach. No model is assumed (apart from normal distribution assumption for expression data and activity scores, as implied from the use of *t*-test and the values standardization), and the methods mainly differ by the choice of network type: large scale PPI network in Chuang's work and known pathways in Lee's work. Both methods fit only labeled data and can be viewed as feature extraction preprocessing before actual use for tasks such as clustering or classification.

3.6.2 Direct Optimization using a Network Loss Function

In an optimization framework, it is possible to formulate a loss function that tends to grow as results deviate from the selected network integration assumption. Hwang et al. [32] and Tian et al. [70] take this approach, using a global loss function for gene expression profile classification.

For every gene, they group together the subset of samples where this gene is highly expressed, and another subset of samples where this gene is lowly expressed. Using a *hypergraph* terminology, they call such a subset a *hyperedge*.

Their target is to assign every sample u with a label $-1 \leq h(u) \leq 1$ corresponding to the range between a negative class (-1) and a positive class (1), and assign every

hyperedge e with a weight $0 \leq w(e) \leq 1$ corresponding to the importance of its gene to the classification task. Labels and weights are assigned so to explicitly minimize the weighted sum of three functions:

Misclassification error. Mean square error of h from the true labeling y, i.e.

$$\sigma_i(h(u_i) - y(u_i))^2$$

Label disagreement among highly similar samples. Mean square error between pairwise samples that are grouped together in the same set (weighted by the set's weight), i.e.

$$\frac{1}{2}\sum_{e\in E}\frac{w(e)}{d(e)}\sum_{u,v\in e}(\frac{h(u)}{\sqrt{d(u)}} - \frac{h(v)}{\sqrt{d(v)}})^2$$

where $d(e)$ is the degree of hyperedge e in the graph.

Weight disagreement among adjacent genes in the network. The third term assumes that given an interaction network, neighbor genes should have a similar weight. Its value directly uses the network:

$$\frac{1}{2}\sum_{i=1}^{|E|}\delta_{i,j}\left(\frac{w(e_i)}{\sqrt{\sigma(e_i)}} - \frac{w(e_j)}{\sqrt{\sigma(e_i)}}\right)^2$$

where $\delta_{i,j}$ is 1 if nodes e_i and e_j are close in the network and 0 otherwise, and $\sigma(e_i)$ is the degree of node e_i in the network.

The objective function is to be minimized subject to the following constraints that constrain the genes' weights and maintain the hypergraph's structure:

$$\begin{aligned} w(e) &\geq 0 && \forall e \in E \\ \sum_{v\in e} w(e) &= d(v) && \forall v \in V \end{aligned}$$

The non-linear optimization is solved using an iterative algorithm named *HyperGene*. It alternates between finding an optimal weight function w for a given current labeling f and finding an optimal labeling for the last resulted weight function.

3.6.2.1 Classification and feature selection using HyperGene

By assigning an initial value of 0 to an unlabeled expression profile, the algorithm can classify it by assigning it with a number close to -1 or close to 1. In addition, the algorithm also automatically selects features that are important for the classification task, by assigning them with weights close to 1 and discards non-important features by assigning them with weights close to 0.

In order to impose the network integration assumption that close genes in the network should have similar contribution to the classification process, and thus similar weights, the authors define two genes to be close if they are at most two genes away on the network. They test their algorithm on three breast cancer prognosis datasets, one of van't Veer et al [72], one of van de Vijver et al [71], and one of [75], showing a better performance than standard SVM methods, and than a label propagation algorithm without network integration. Specifically, using initial 231 top correlated genes with labels from the breast cancer dataset of van 't Veer [72], they report AUC score of 0.893 using HyperGene vs. 0.845 using SVM classification. For an independent set of 325 cancer genes from [63] on the van de Vijver breast cancer dataset [71] they report AUC score of 0.699 using HyperGene vs. 0.679 using SVM.

In addition, they compare their gene weights to gene weights based on correlation between the expression levels of a gene and the class label, showing that known breast cancer related genes are ranked higher by their method than by mere correlation with labels. For example, the BRCA1 gene is known to play a crucial role in DNA damage repair, and to significantly increase the probability of a carrier of a mutated gene to develop breast cancer. BRCA1 is ranked 629 in its correlation with the samples' labels vector, and 14 in its weight as assigned by HyperGene.

In a more recent work [70], Tian and Hwang expand their work to other types of data, and also offer an alternative network integration assumption, with an algorithm they name *HyperPrior*. Instead of directly requiring pairwise similarity between two nodes in the integrated network, they use a neighborhood constraint, requiring the weight of a node to be close to the average weight of its neighbors:

$$\frac{1}{2}\sum_{i=1}^{|E|}\left(w(e_i)-\sum_{j=1}^{|E|}\frac{\delta_{i,j}}{\sigma(e_i)}w(e_j)\right)^2$$

In addition, instead of using a binary $\delta_{i,j}$ as an indicator whether two nodes i and j are close or not, they offer a continuous alternative, where $\delta_{i,j}$ is inversely proportional to the distance between i and j in the network.

The authors compared their results to SVM as well as to the algorithm proposed by Rapaport et al. [59] (see Section 3.6.4) and a net propagation algorithm without network integration on two breast cancer gene expression datasets of van 't Veer et al. [72] and van de Vijver et al. [71]. On the van de Vijver dataset, Hyperprior achieved AUC average of 0.692 compared to 0.671 with linear SVM and 0.665 using the spectral approach algorithm of Rapaport et al. On the van 't Veer dataset, Hyperprior achieved AUC average of 0.881 comparing to 0.857 AUC with linear SVM and 0.869 with the algorithm of Rapaport et al.

3.6.3 Network Guided SVM Regularization

Regularization refers to restricting the effective range of parameters. It is used in order to solve ill-posed problems or fight overfitting. Zhu et al. [83] offer regularization of SVM that is based on the network topology.

The soft-margin SVM (see Section 2.1.2) objective function can be re-formulated as:

$$\min_{\mathbf{w},w_0}\left\{\sum_{i=1}^{n}[1-y_i(\mathbf{x_i}^T\mathbf{w}+w_0)]_+ + \lambda\|\mathbf{w}\|_2^2\right\} \tag{3.1}$$

Where the subscript "+" denotes the positive part, i.e. $z_+ = \max(z,0)$, and $\|\mathbf{w}\|_2^2 = \sum_{j=1}^{p}(w_j)^2$. In this formulation, λ is the regularization parameter, and the regularization term $\lambda\|\mathbf{w}\|_2^2$ forces the average of weights' absolute value to be low, that is, close to 0.

Zhu et al. [83] take advantage of this formulation, to introduce an alternative regularization term that takes the network into account. Let E be the set of edges in the network, and w_v be the component in the weight vector corresponding the node $v \in V$. Let us define a normalization factor $s_v > 0$ associated with every node v. The regularization term they propose (replacing $\lambda\|\mathbf{w}\|_2^2$) is:

$$\lambda \sum_{(u,v)\in E} \max\left\{\frac{|w_u|}{s_u}, \frac{|w_v|}{s_v}\right\}$$

The basic idea is decreasing (or eliminating) the weights for neighboring genes. The objective function pulls one gene toward zero if its neighbor's weight is zero. According to the authors, this formulation satisfies our assumption that neighboring genes tend to contribute (or not contribute) to the same biological process at similar conditions. One important property of this formulation is that not only it reduces the total weights, but it does so by zeroing many non-useful features' weights, and this way performs feature selection, by taking into account only the non-zero weight features.

The authors tested the formulation on case-control *Parkinson's Disease* gene expression data, using a network borrowed from [47], which combines 33 KEGG regulatory pathways that are related to neuronal activity, and used the degree, or the square root degree, of a node as the normalization factor s_v. They obtained classification performance comparable to the standard SVM, with more known disease related genes selected by their network-based SVM than by the standard SVM.

3.6.4 A Spectral Approach

Rapaport et al. [59] present a linear transformation of original expression profiles that essentially smoothes the expression values over the network. That is, if two genes are near each other in the network, their expression values will be corrected so that they will become more similar. They consider this correction as a noise reduction procedure over the original expression profiles.

The method takes advantage of popular noise reduction techniques used in digital signal processing based on *Fourier transform*. The idea is to take the signal, and represent it in another space, where each component corresponds to a different frequency. The weight of each component is its coordinate's value in the transformed vector. Now, assuming most of the signal data are in the low frequencies and that high frequencies are mostly noise, we can set the high frequencies' weights in the transformed vector to zero, and then use the inverse transformation to get back the original signal, this time without the noise.

The authors argue that the method can be used with any large scale network, and test it on a metabolic network comprised of KEGG pathways. As they use the pathways from the well studied *Saccharomyces cerevisiae*, their network is quite large and relatively complete. There are no model assumptions, and the method is actually a preprocessing step that can be applied to data regardless of the subsequent goals, so it can be used on both labeled and unlabeled data.

After applying the transformation to gene expression data, the authors cluster the genes according to their transformed expression values and compare the coherence of the resulting clusters to that of clusters obtained without the transformation. They define a loss function that measures the ratio between the sum of pairwise distances of genes that belong to the same KEGG class, and the total sum of pairwise distances, and show that the loss dropped significantly from 0.444 to 0.43. In addition, they apply supervised binary classification of the transformed data, and obtain slightly lower results for the transformed data vs. the original data.

3.6.5 Disease Genes Discovery by Network Analysis of Differential Expression

As mentioned before, one of our analysis goals is to find candidate genes that may serve as strong leads for downstream research. In a differential expression context, where we have labeled data, Nitsch et al. [55] offer to score a gene using its neighbors' scores, under the assumption that strong leads tend to be surrounded by differentially expressed neighbors (sometimes regardless of their own differential expression level).

The authors define a soft neighborhood of a source gene where each gene is assigned with a weight inversely proportional to its distance from the source gene. Given a differential expression score for every gene (e.g. based on *t*-test), the final differential expression score of the source gene is a weighted average of the scores of all genes in its neighborhood. They set the actual weights based on the *Laplacian exponential diffusion kernel* [42]. The kernel allows for computation of a global distance function between every two nodes that takes into account all the paths and the overall accessibility between the two genes.

The Laplacian exponential diffusion kernel defines a similarity matrix K between

pairs of nodes, that is equivalent to an infinite simulation of a lazy random walk on the graph, starting from a node v. The similarity between v and u, found in the matrix at K_{uv} is the probability to end the random walk at u following an infinite number of steps. At each step we stay at the the current node with probability β and if we do not stay at the current node, we have an equal probability to take a step to each of the nodes adjacent to the current node.

For a given neighborhood of a source gene v, the authors set r_u to be the rank of the distance between nodes v and u. The weight of node u required for the calculation of the differential expression score of v is set to be $e^{-\beta \cdot r_u}$, where β is a parameter that defines the degree of diffusion. The differential expression score of v using its neighborhood $N(v)$ is then:

$$s(v) = \sum_{u \in N(v)} e^{-\beta \cdot r_u}$$

Nitsch et al. aim at finding what they term *"disease causing genes"*, and do not look at subsequent learning tasks such as classification or clustering. They present four case studies, each of a different disease. For every disease they pick a single gene from literature, and report the ranking and significance of that gene using their method on a dataset related to the disease. For example, for Cystic Fibrosis they choose CFTR as their disease gene. Using their method, it was ranked 7 out of 110 candidates. The method is only applicable on labeled data and the authors do not assume any statistical model. They use STRING [66] as their network resource.

3.6.6 Statistical Hypothesis Testing Frameworks

From a statistical hypothesis testing point of view, one formulates a *null hypothesis* (implying an opposite *alternative hypothesis*), and tests for the probability that the null hypothesis is false using standard statistical tools. If it is found to be false with high probability, the alternative hypothesis is accepted. For example, a statistical test can be conducted to see whether a gene is differentially expressed or not. In the standard setting, the result of such a test is independent of the result of similar tests for other genes although it is known that such dependencies do exist.

Wei and Pan [77] take a statistical approach built on a *mixture model*. The idea of their mixture model is to provide a probability function that is actually a combination of a number of basic probability functions that corresponds to a two-step sampling process: first, pick among the different populations according to some distribution, then pick (or generate) a sample from the chosen population according to a population-specific distribution.

Under a mixture model assumption, it is possible to estimate the prior probabilities and parameters for each of the populations given a set of samples. This may be done using methods such as Expectation-Maximization (EM). Applying the standard mixture model to differential expression of genes, we have two populations of genes - differentially expressed (DE) and equally expressed (EE), each with its own probability density function, $f_{\triangle}$ and $f_{=}$, respectively. A gene has the prior probability $\pi_{\triangle}$ to belong to the differentially expressed population and probability $\pi_{=} = 1 - \pi_{\triangle}$ to belong to the equally expressed population. The *marginal distribution function*, or the probability that a gene i will take the expression level z_i is:

$$f(z_i) = \pi_{\triangle} f_{\triangle}(z_i) + (1 - \pi_{\triangle}) f_{=}(z_i)$$

Here, all genes have the same prior probability to be differentially expressed. One option for the actual function $f_{=}$ can be based on the hypothesis that all expression values of the equally expressed genes come from a single population, and for $f_{\triangle}$, it can be based on the hypothesis that expression values of the differentially expressed genes come from two populations - either from the case population or the control population. Note that the latter is also a mixture model (within the DE genes).

Wei and Pan propose that every gene will have its own prior probability that it is differentially expressed. That is, instead of one $\pi_{\triangle}$ for all genes we now have for each gene $i = 1, \ldots, p$ its own parameter $\pi_{i,\triangle}$. However these prior probabilities are not independent, and Wei and Pan assume a simplifying *Markovian assumption*: the prior probability that a gene is differentially expressed given the prior probabilities of all other genes, depends only on the prior probabilities of its neighbor genes.

The network integration assumption they use is then that neighbor genes are more likely to be in the same state - either both DE or both EE. They assume a full statistical

model for both the expression data (coming from a mixture of normal distributions), for the prior probability of genes' states, and for the relation between these prior probabilities (the Markovian assumption). They fit the model parameters using Bayesian methods, and based on the fitted parameters they can ultimately get the probability of a gene being differentially expressed given the data, and used various thresholds on this value, for their final decision.

The authors used *ConfidentNet* [46], a probabilistic functional *Saccharomyces Cerevisiae* network, assembled by Lee et al. from various sources. Although they used differential expression examples throughout they work, they eventually applied the data to a different gene expression analysis task, transcription factor (TF) target prediction. They focused on a single transcription factor and conducted a *ChIP-chip* experiment - an experiment that uses microarray technology in order to quantify the occupancy of genome-wide promoter regions by the selected TF. In the standard setting, the quantities obtained from the ChIP-chip experiment are assigned with a statistical significance level based on their assumed underlying distribution.

Wei and Pan collected from literature a set of genes that are known to be regulated by the chosen TF and another set of genes that are unlikely to be regulated by it. Based on the ChIP-chip results, they used their method to predict the set of genes regulated by the chosen transcription factor, and compared their set to the control sets they collected, measuring sensitivity and specificity of their prediction, and producing a ROC curve. When the specificity ranged from $0.4 - 0.9$ their spatial normal mixture model gave a higher sensitivity than that of the standard normal mixture model.

Another statistical modeling approach is offered by Wei and Li [78]. Instead of using prior probabilities for each gene to be either differentially expressed or equally expressed, they assume that each gene has a true unknown binary *state*, either differentially expressed (DE) or equally expressed (EE). They use a standard statistical hypothesis testing framework for testing a gene for being differentially expressed, and integrate the network by assuming that genes that are neighbors in some pathway are more likely to be in the same state.

As for the expression data, they too assume that every expression of an EE gene is sampled either from a single distribution that is valid for all the samples, and every DE

gene expression is sampled from one of two distributions - one for the case group and another for the control group.

The network effect in the model makes it more likely that neighbor genes share the same state, by using a *Random Markov Field* model: For a given vector of states $\mathbf{s}$, let $n_{\triangle}$ and $n_{=}$ be, respectively, the numbers of differentially expressed and equally expressed genes. The probability for getting this vector is a function of $n_{\triangle}$ and $n_{=}$:

$$p(\mathbf{s}) \propto \exp(\gamma_{\triangle} n_{\triangle} + \gamma_{=} n_{=} + \beta n_{(\triangle,=)})$$

where $n_{(\triangle,=)}$ is the number of edges in the network whose endpoint nodes are in different states, and $\gamma_{\triangle}$, $\gamma_{=}$ and $\beta > 0$ are the model's parameters.

Wei and Li use an iterative method for selecting the most likely vector of states $\mathbf{s}$ in the following way:

1. Obtain an initial estimation $\hat{\mathbf{s}}$ of $\mathbf{s}$ using *t*-test
2. Estimate the set of parameters θ of the expression data distribution by finding $\hat{\theta}$ that maximize the likelihood of the expression data given the state vector $\hat{\mathbf{s}}$.
3. Estimate the parameters $\gamma_{\triangle}$, $\gamma_{=}$ and β that maximize the likelihood of the estimated state vector given the network.
4. Obtain a new state vector by updating the gene states one gene after another, such that the new states maximize the joint likelihood of this gene's expression levels given its state and the states of other genes in the network.
5. Go to step 2 until a stop criterion is met.

The authors used a network comprised of 33 human regulatory pathways from the KEGG database, that contain both regulatory and PPI information. They applied their method on two breast cancer prognosis datasets (Wang [75] and Müller [52]) assuming a Gamma-Gamma model for the expression levels distribution [53, 39]. They show that at the same significance level, their method is able to find 103 differential expressed genes, while a standard Gamma-Gamma model without network integration finds 82 genes, and a simple *t*-Test filter results with only 4 significant genes.

Both works initially aim at improved modeling of labeled gene expression data by the integration of network data, under the assumption that the differential expression level of two neighbor genes tend to be similar. Both algorithms do not try to solve a specific classification or clustering task, and their output may be used as feature selection for subsequent tasks. Both papers and models are conceptually similar, with a slight difference at the basic assumptions - Wei and Li treat the differential expression trait as discrete (either differentially expressed or not), while Wei and Pan assign a probability to it. Minor differences are also found in the choice of distribution function for the expression data - Normal vs. Gamma distribution. Finally, the algorithms used for parameter estimation are different.

3.7 Algorithms Taxonomy

We chose to summarize the properties of the algorithms discussed above along the axes that define the basic integration questions we asked earlier:

1. What is the analysis goal or task?
2. What type of prior knowledge (network) is used?
3. Does the method assume a statistical model or not?
4. Does the algorithm require labeled data (i.e. is it supervised) or not?
5. What is assumed regarding the influence of the network on the expression?

Table 3.7 provides a summary and allows comparison of the methods.

	Task					Network				
	Noise reduction	Feature selection	Feature extraction	Classification	Clustering	PPI / STRING / Functional	Regulatory Pathways	Statistical Mode	Requires labeled data	Integration assumption
Chuang			Y	Y		Y			Y	1, 2
Lee			Y	Y			Y		Y	1, 2
Rapaport	Y		Y	Y	Y	Y				1
Wei and Li		Y		Y		Y	Y	Y	Y	3
Efroni		Y	Y	Y	Y		Y	Y		4
Wei and Pan		Y				Y	Y	Y		3
Zhu		Y		Y					Y	3
Nitsch		Y				Y			Y	5
Hwang		Y		Y		Y				6
Tian		Y		Y		Y				6

Table 3.1: **Taxonomy of Network and Expression Data Integration Methods**. The table summarizes the different aspects of network and expression data integration in the different methods. **1.** Close genes co-express. **2.** Expression of subnetworks is more robust. **3.** Close genes co-differentially-express. **4.** Expression reflects materialization of regulatory interactions. **5.** Causative genes are surrounded by differentially expressed neighbors. **6.** Close genes contribute similarly to classification model.

4

Results on Network Impact

In this chapter we will present results on the relations between network data and expression data. In Section 4.1 we carry out an analysis showing support for some of the integration assumptions discussed in Chapter 3. In Section 4.2 we test for the actual impact of network integration on two of the methods reviewed in Section 3.6. Both results lead us to the choice of our working assumptions.

4.1 Informativeness of Integrated Network Data

In this work, we chose to focus on network data, and particularly on data that represent relationship between pairs of genes or proteins. STRING [36], mentioned earlier, is such a network, combining curated data to provide both physical and functional interactions between proteins. We first tested whether the level of co-expression of gene pairs whose corresponding proteins are close in the STRING network is higher than that of a random pair of genes.

Let $\mathbf{x}$ and $\mathbf{y}$ be two vectors of expression measurements of two genes over a given set of samples. To measure the level of co-expression of two genes we used the sample correlation coefficient based on the Pearson correlation. Let x_i and y_i be expression measurements of the two genes on the i'th sample. The sample correlation coefficient r_{xy} is defined as:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of x and y, and s_x and s_y are the sample standard deviations of x and y. the Pearson correlation is 1 for positive linear dependency

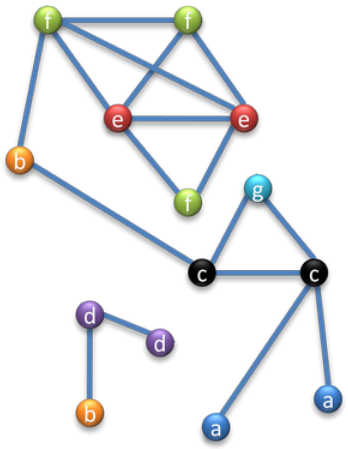


Figure 4.1: **Toy example of gene pair populations.** Among the adjacent genes in the network are: pair **c**, pair **d** and pair **e**. Pair **a** is 2 nodes away, pair **b** is not connected at all and thus considered as 7+ nodes away. Pair **c** is considered more connected than pair **d**, as it is also a member of a 3-clique formed by node **g**. This pair is described as adjacent member in one 3-clique. Pair **e** has an even higher connectivity, and is described as adjacent member in three 3-cliques, using the three 3-cliques formed by three nodes marked **f**.

between the vectors and -1 for negative linear dependency. The further away the vectors are from linear dependency, the closer the correlation is to 0. Although in some cases dependency between expression vectors of two genes can be non-linear, we still expect high correlation in many biological cases, such as between proteins levels of members of the same complex, or those that share similar regulation factors. Even in the cases of other monotonic non-linear dependency, such dependency can be still visible in the Pearson correlation score, albeit with lower values.

In order to account for both negative and positive correlation we used the absolute value of the Pearson correlation, comparing distributions of correlation values sampled from various gene-pairs populations. As a baseline we sampled random pairs from the population of all gene pairs that are neighbors in the network, comparing the distribution of correlations to non-neighbors (distant) pairs. We also looked at more specific populations of pairs according to their distance - pairs that are 3, 4, 5, 6 nodes away, and pairs that are 7 or more nodes away, including nodes that have no path between them.

Distance is a highly local measure, and does not take into account the existence of multiple paths between nodes. We were also interested in a connectivity measure that

Pair population (# of pairs sampled)	Mean correlation	Significance
Adjacent-baseline (15340)	0.12302	N\A
Distant (19978)	0.11078	7.24×10^{-31}
2-nodes away (654)	0.11746	6.99×10^{-2}
3-nodes away (3171)	0.11527	1.33×10^{-5}
4-nodes away (7733)	0.11167	7.24×10^{-18}
5-nodes away (4453)	0.11115	1.68×10^{-14}
6-nodes away (1207)	0.11263	9.56×10^{-5}
7+ nodes away (2755)	0.10857	9.65×10^{-15}
Adjacent members in 1 3-clique (5625)	0.1303	1.53×10^{-5}
Adjacent members in 2 3-cliques (2683)	0.13555	1.84×10^{-7}
Adjacent members in 3 3-cliques (1485)	0.14518	3.13×10^{-11}
Adjacent members in 5 3-cliques (622)	0.15377	8.83×10^{-9}

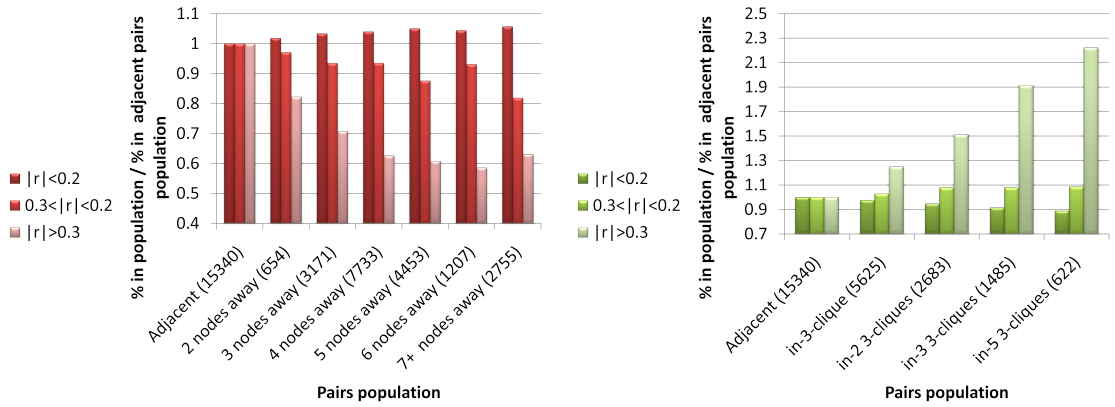
Table 4.1: **Mean correlation among different gene pairs populations.** The right column measures the probability that the samples of the population and of the baseline came from the same distribution.

will take into account multiple connections, under the assumptions that due to the noisy nature of large scale networks, multiple paths strengthen the confidence in the relation between two nodes in the network. To this end, we looked at few additional gene-pairs populations - adjacent genes to those that are also members in one or more 3-cliques. We gathered those that are members in 2, 3, 4 and 5 3-cliques. A toy example illustrating the different gene-pair populations samples can be seen in Figure 4.1.

4.1.1 Distribution of Correlation Between Pairs

We measured the gene expression correlation on the Wang [75] breast cancer dataset, looking at the different gene-pair populations sampled from the STRING [66] network. The STRING human network used has 6243 genes (nodes) and 19102 interactions (edges). Overall, the mean correlation of adjacent genes is slightly higher than that of distant (non-adjacent) genes with $\bar{r} = 0.1107$ for distant genes, and $\bar{r} = 0.12302$ for adjacent genes, but this difference has high significance of p-value$< 7.24 \times 10^{-31}$. As seen in Table 4.1, the correlation is stronger among genes that are not only adjacent, but also members of 3-cliques, and the number of 3-cliques is a good indicator of the strength of correlation.

To evaluate the distribution of correlation for each pair category, we plotted their distributions. Figure 4.2 summarizes the results. Looking at the high correlation range, we can see that the percentage of adjacent genes that exhibit high correlation values, is



(a) (b)

Figure 4.2: **Dependence of gene pair expression correlation on their physical closeness.** Percentage of pairs exhibiting certain levels of correlation in selected pair populations compared to the percentage of pairs with the same correlation levels among adjacent pairs. **(a)** The graph shows three levels of correlations (in their absolute values) in a number of gene-pair populations. For each level and selected population, the bar shows the ratio between the percentage of gene pairs exhibiting the specified correlation level in the selected population, and the percentage of gene pairs exhibiting that absolute correlation value in the adjacent pairs population. At the low correlation level ($|r| < 0.2$) there are relatively more distant pairs, and at the high range ($|r| > 0.3$) there are relatively more adjacent pairs. For visualization, the adjacent pairs population is included as well. In parentheses - the number of pairs sampled in each population. **(b)** The same presentation for populations of gene-pairs that are also members of k 3-cliques, for different k's. Here, the opposite picture is obtained. For the low correlation range ($|r| < 0.2$), there are relatively more adjacent pairs than 3-cliques pairs. For the high correlation range ($|r| > 0.3$), the greater the number of 3-cliques the pair shares, the higher the percentage of highly correlated pairs.

higher than that of distant genes, and this percentage decreases with gene distance (Figure 4.2(a)). The opposite is true for the low correlation range. Adjacent genes that are also highly connected, as measured by their membership in multiple 3-cliques, show even higher percentage in the high correlation range (Figure 4.2(b)).

4.2 Does the Network Make a Difference?

Although networks are indeed informative with regard to co-expression, it is not clear whether their use in the different methods covered above in Section 3.6 is actually responsible for the improvement in results. In some cases, it is hard to compare results of methods that do use network and those that do not. One example for these difficulties can be seen in [78]: Searching for differentially expressed genes, at the same significance level, Wei and Li find more genes with network data integration than without it. However, computing the significance level for different methods might not be comparable, and one can question the assumption that a longer list is better at all.

One more example can be seen in the work of Chuang et al. [17] (see Section 3.6.1). There, the authors extract subnetworks as features, such that each feature is actually constructed of a number of genes. Performance of feature-selection methods should be compared when all methods use a similar number of features. The question here is whether to compare the performance based on the number of features (subnetworks), or based on the number of genes constructing these subnetworks, which is an order of magnitude larger.

Last, many of the algorithms include improvement potential that is not only due to the network integration. Chuang et al. introduce a greedy search that is guided (or constrained) by the network topology. Is the improvement they show due to the greedy search, the network, or both? To answer this question, one needs to eliminate the network influence on the algorithm. One way to do it is to apply the algorithm on random networks, and check whether the improvement of the true network is significant or not. Chuang et al. tested whether their resulting features are significant by comparing the score of each feature (subnetwork) seeded at node v to scores of features obtained by using three different randomization procedures. At each procedure, they built a set of features that served as the background set for estimating the significance of their real features:

1. Subnetworks that were extended randomly (i.e. not in a greedy manner) starting from node v

2. Subnetworks that were extended randomly (i.e. not in a greedy manner) starting from random nodes.

3. Subnetworks that were extended greedily, according to a a random permutation

labeling of the samples.

All of their significance tests were based on the true network, and they report improvement due to the network use.

We compared the performance of Chuang's feature selection method with the STRING [36] network to that of their method with the STRING network randomly permuted. Permutation was done by renaming the network's nodes, so that the topology of the network is preserved, but any correlation between expression and network is removed. We report the average of 50 different permutations, vs. 50 runs of the original algorithm, using the same number of features (200). There are two elements of randomness in the algorithm that required us to run the original algorithm multiple times for comparison. The first includes the significance tests: Although the algorithm is deterministic, and will always find the same subnetwork starting from a specific seed, the calculation of significance level of the resulting subnetwork is based on sampling and hence may be different every time. The second source of randomness is due to the different folds used to measure the classification performance.

The test was conducted on two breast cancer datasets [75], [71] using two classification algorithms. As a performance measure we used the AUC score. As seen in Figure 4.3, there is no significant difference between the results using a real network, or a permuted one.

We also conducted a single run comparison on eight more datasets, using two different networks - STRING [36] and a PPI network from the IntAct database [40, 5]. The results were similar (Figure 4.4).

We suspect then, that the improvement shown by Chuang et al. is due to the greedy search the algorithm includes rather than due to the real network topology. The network topology merely limits the subset of nodes (genes) that are reachable from every node. If this subset is large enough, the greedy algorithm will find a combination of genes within this subset that will improve the classification results, regardless of the actual content of this subset.

A similar test on the HyperGene algorithm [32] (see Section 3.6.2), resulted with a significant performance decrease when substituting the network with a permuted one. On

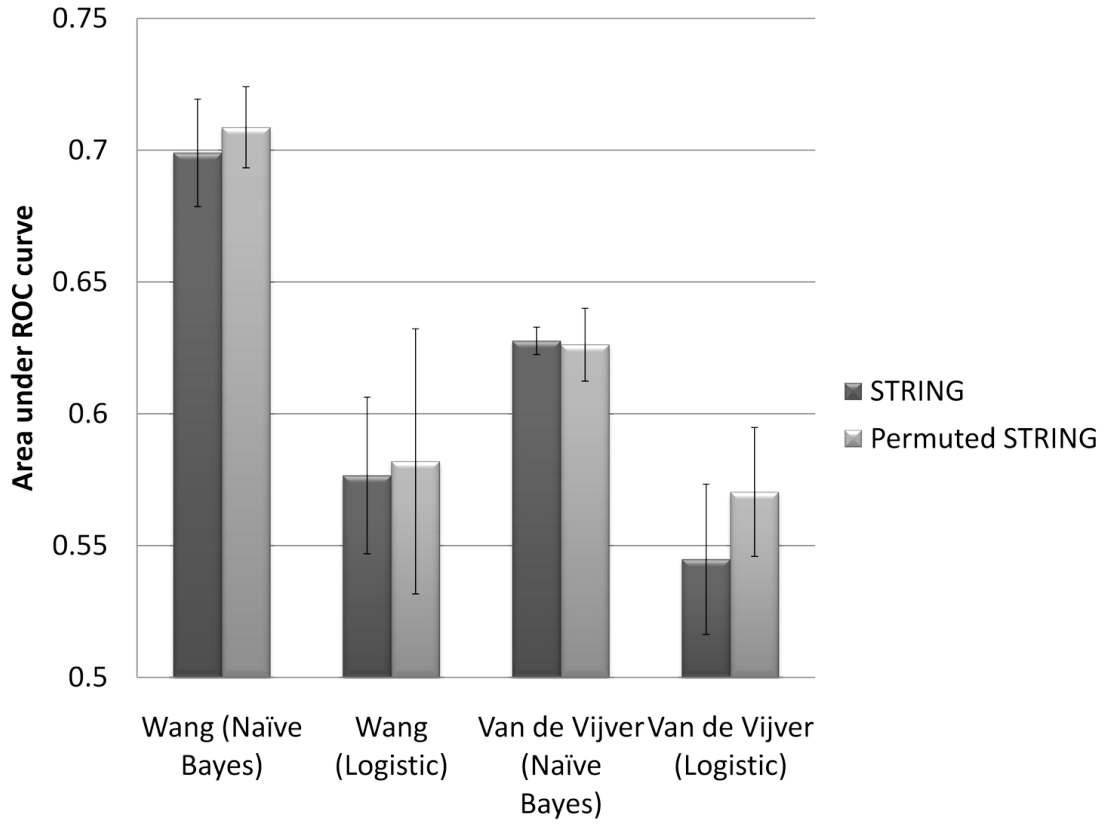


Figure 4.3: **Algorithm performance is indifferent to the underlying network.** The figure presents the classification performance based on features selected by the PinnacleZ algorithm - AUC average and standard deviation of 50 runs of the PinnacleZ algorithm from Chuang et al [17] using the STRING network [36] and of 50 different permutations of the network. Four results are shown from two different classification algorithms and two different datasets.

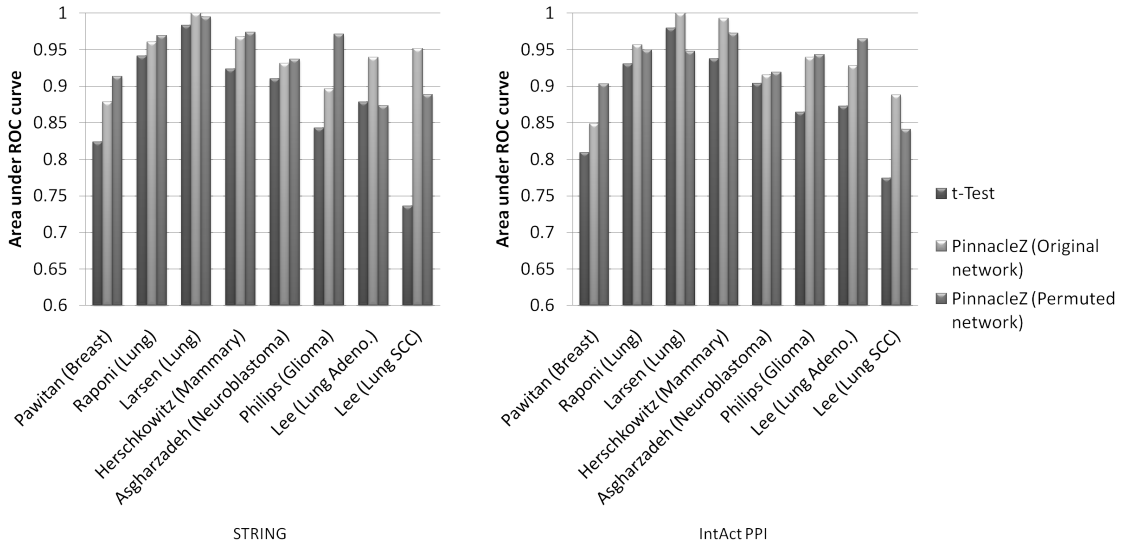


Figure 4.4: **Performance of PinnacleZ algorithm on the original and permuted networks.** The figures present the classification performance of a Naive Bayes classifier, based on top 200 features selected by t-Test and the PinnacleZ algorithm with the original network and with a randomized network. The test was repeated using two different networks (STRING [36] and human PPI network [40, 5]) on eight different datasets ([57, 60, 43, 30, 6, 45, 58])

the van't Veer dataset [72], the HyperGene algorithm with 50 random networks resulted with AUC scores ranging from 0.7024 to 0.8095. 5 fold CV score using a real network resulted with an average AUC score of 0.845 for SVM and 0.893 for the HyperGene algorithm. Figure 4.5 shows the results. In this case it seems that the topology of the network does play a role in the improvement achieved. In Section 6.2.3 we show similar results when conducting the same test on our new method presented in Chapter 5.

4.3 Derivation of Working Assumption

Encouraged by the validation of informativeness of relations between gene pairs to expression values, we put together our working assumption for integrating network data into the task of gene expression profiles classification.

First, since our problem is a supervised learning problem, we require that two genes that have similar expression profiles also contribute similarly to the classification model. Following the observation that highly connected genes are more likely to share similar expression profiles, we concluded, based on the improvement observed in the HyperGene

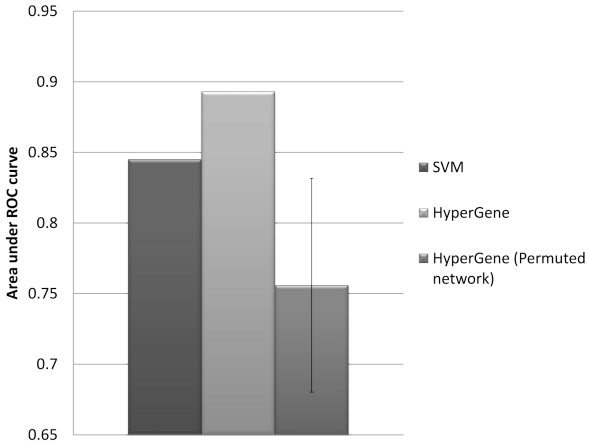


Figure 4.5: **Performance of HyperGene algorithm on the original and permuted networks.** The figures present the classification performance of SVM classification, the HyperGene algorithm with the original network supplied by [32] and the average and standard deviation of 50 runs of the HyperGene algorithm with a randomized network.

algorithm, that neighboring genes should exhibit similar contribution to the classification model.

Second, as shown by the 3-clique membership influence, the relation between two genes should not be limited to their pairwise adjacency in the network. It should also take into account connectivity as measured by alternative (short) paths between the genes, rather than merely by their shortest-path distance. In chapter 5 we describe how we turn these findings into a practical algorithm for supervised learning with prior knowledge based on network data.

5

A New Network-Based Kernel and Transformation

In Section 2.1.2 we introduced SVM classification. SVM explicitly minimizes the norm of the features' weight vector. Later on, in Section 3.6.3 we showed one method by Zhu et al. [83] that regularizes SVM by optimizing the weight vector so that assigning a low weight to one feature will push the algorithm to assign low weights also to its neighbors. Instead of minimizing the norm of the features' weight vector, the authors construct a different weight vector, with one weight per edge in the network. The weight corresponding to an edge is assigned to be the maximum among the normalized weights its two endpoint features. This way, in an indirect manner, if the algorithm wishes to assign a small or zero weight to one feature, the objective function will only benefit from this if the feature's neighbor's weight will also be small or zero.

Hwang et al. [32] (see Section 3.6.2) explicitly minimize the mean square pairwise difference of all pairs of neighbor genes' weights in a non-SVM framework. We propose here to explicitly add the mean square pairwise difference to the SVM objective function, and directly minimize it while maximizing the margin. We would like to regularize the weights so that adjacent nodes will be assigned with similar weights, and offer a tradeoff between the maximal margin, and this regularization. In fact, our formulation is not restricted to the use of a graph as a binary source of dependencies between features and can use any non-negative gene-gene similarity matrix. The method, as will be shown hereafter, reduces to a linear transformation of the original data that allows us to use current SVM formulation, algorithms and implementations for building and testing our classification model.

5.1 SVM Regularization via Feature Similarity

Recall SVM formulation for the linearly separable case:

$$\min_{\mathbf{w},w_0}\left\{\frac{1}{2}\|\mathbf{w}\|^2\right\}$$

subject to

$$(\mathbf{w}^T\mathbf{x}_i + w_0)\cdot y_i \geq 1 \qquad \text{for any } i = 1,\ldots,n$$

let A be a symmetric $p \times p$ matrix with non-negative values, where $A_{i,j} = A_{j,i}$ stands for the similarity level between features i and j. In order for the weights to be closer for features that are more similar, we wish to minimize the following mean square pairwise difference expression:

$$\frac{1}{2}\sum_{j>k} A_{j,k}(w_j - w_k)^2$$

We add this expression to the objective function, introducing a non-negative tradeoff parameter $\beta \geq 0$:

$$\min_{w_0,\ldots,w_p}\left\{\frac{1}{2}\sum_{i=1}^{p} w_i^2 + \frac{1}{2}\beta\sum_{j=1}^{p}\sum_{k=j+1}^{p} A_{j,k}(w_j - w_k)^2\right\} \tag{5.1}$$

Let $\tilde{A}$ be a $p\times p$ diagonal matrix with the sum of row j of A at $A_{j,j}$, $\tilde{A}_{j,k} = \delta_{j,k}\sum_l A_{j,l}$. Here $\delta_{j,k}$ is the Kronecker delta with $\delta_{j,k} = 1$ if $j = k$ and $\delta_{j,k} = 0$ otherwise. The matrix notation of 5.1 is:

$$\min_{\mathbf{w},w_0}\left\{\frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}\beta\mathbf{w}^T(\tilde{A} - A)\mathbf{w}\right\} \tag{5.2}$$

For simplicity, we denote $B = \tilde{A} - A$. Note that for a simple adjacency matrix based on a graph, where $A_{i,j} = 1$ if i and j are adjacent and $A_{i,j} = 0$ if they are not, $\tilde{A}$ is a diagonal matrix with $\tilde{A}_{i,i}$ being the degree of node i, and B is known as the *Laplacian Matrix* of the graph. The newly added term captures the assumption that close genes are

more probable to co-express and thus to similarly contribute to the learned classification model.

Although constructed of many local relations between adjacent features only, the term also gives room to a more global view of connectivity, as highly connected features, such as those described in Section 4.3 have multiple short paths between them and in turn have more local relations in the given summation, increasing their overall weight in the objective function.

The solution to the original SVM quadratic programming problem is obtained by transforming the optimization problem to the dual form. In order to get to the dual form, we will introduce Lagrange multipliers $\alpha_1, \ldots, \alpha_n$, $\alpha_i \geq 0$, one for each constraint (corresponding to a single sample point).

The primal Lagrangian is:

$$\begin{aligned} L_P &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}\beta\mathbf{w}^T B\mathbf{w} - \sum_{i=1}^{n}\alpha_i((\mathbf{x}_i^T\mathbf{w} + w_0)y_i - 1) \\ L_P &= \frac{1}{2}\mathbf{w}^T(I + \beta B)\mathbf{w} - \sum_{i=1}^{n}\alpha_i((\mathbf{x}_i^T\mathbf{w} + w_0)y_i - 1) \end{aligned} \tag{5.3}$$

In order to reach the same solution of 5.2, we need to find a saddlepoint of L_P where it is maximized with respect to $\mathbf{w}, w_0$ and minimized with respect to the Lagrange multipliers $\alpha_1, \ldots, \alpha_n$. We differentiate L_P with respect to w_0 to get:

$$\frac{\partial L_P}{\partial w_0} = \sum_{i=1}^{n}\alpha_i y_i$$

and set it equal to 0 to get:

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{5.4}$$

We differentiate L_P with respect to $\mathbf{w}$ to get:

$$\frac{\partial L_P}{\partial \mathbf{w}} = (I + \beta B)\mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i$$

and also set it equal to 0 to get:

$$\mathbf{w} = (I + \beta B)^{-1} \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{5.5}$$

By substituting $\mathbf{w}$ into 5.3 we get the dual from of the Lagrangian:

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T (I + \beta B)^{-1} \mathbf{x}_j \tag{5.6}$$

Consider the matrix B. From the equivalence of 5.1 and 5.2 it follows that $\mathbf{w}^T B \mathbf{w} = \sum_{j>k} A_{jk}(w_j - w_k)^2$ Since $Ajk \geq 0$ for all j, k it follows that $\mathbf{w}^T B \mathbf{w} \geq 0$ for all $\mathbf{w} \neq \mathbf{0}$ and hence B is positive semidefinite. Since $\beta \geq 0$, $I + \beta B$ is positive definite, and therefore it is invertible, and the inverse matrix $(I + \beta B)^{-1}$ is also positive definite. Since $I + \beta B$ is symmetric, $(I + \beta B)^{-1}$ is also symmetric. Being both positive definite and symmetric, $(I + \beta B)^{-1}$ can be decomposed using *Cholesky decomposition* [26]: there exists an lower-triangular matrix L such that $(I + \beta B)^{-1} = LL^T$. Plugging LL^T into 5.6 yields:

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T L)(L^T \mathbf{x}_j) \tag{5.7}$$

It is possible, then, to transform the sample vectors x_i using L as a linear transformation to obtain a set of transformed samples where $\mathbf{x}_i^{new} = \mathbf{x}_i L$. Now we can run the SVM optimization procedure in order to learn a model of the transformed samples. In order to classify a new unseen sample, we should first transform it in the same manner, using L and then use the model to classify it.

5.2 Transformation Analysis

The matrix $Q = (I + \beta B)^{-1}$ defines a kernel replacing the default L_2 norm with a new norm that takes into account the graph structure and edge weights according to the tradeoff parameter β. The matrix Q, which we analytically derived from our regularized SVM formulation, was first introduced in [25] in the scope of electronic engineering and chemistry. It has been investigated independently by [49] and in a series of papers by

Chebotarev et al. including [15]. It is a *doubly-stochastic* matrix - the sum of every row and every column is 1. Articles [14, 15] provide two probabilistic interpretations for its entries.

For sake of simplicity, we will consider the case where $\beta = 1$ and restrict A (the initial similarity matrix) to be an adjacency matrix representing a simple graph. The first interpretation relates to spanning forests of the underlying graph. A *forest* is a graph without cycles. A *spanning forest* of a graph G is a subgraph of G that contains all of its nodes, and some of its edges, and is a forest. A *spanning rooted forest* of G is a spanning forest of G where at each connected component of the forest one node is marked as its *root*. Note that each connected component of a spanning forest is a tree. Let $F_{i,j}$ be the number of spanning rooted forests of G where j is in a tree rooted at i. Let F be the total number of spanning rooted forests of G, then $Q_{i,j} = \frac{F_{i,j}}{F}$.

The second interpretation relates to random walks on G with p nodes. Define a *random walk with a random number of steps* as the following process: Starting at node i, at each step, one performs a Bernoulli experiment with success probability $q = \frac{1}{p}$. Upon success, the random walk stops. Upon failure, the walker moves to an adjacent node with equal probability for all neighbors, and the process is repeated. $Q_{i,j}$ is the probability to stop at node j following a random walk as defined above, that started at node i.

Both interpretations give insight as for the entries of Q. $Q_{i,j}$ is a connectivity or proximity measure between two vertices that takes into account both their distance and their surrounding neighborhood. The closer the vertices are, the greater the probability that they will fall in the same tree for a randomly selected forest. For the second interpretation, the random walker is more likely to stop at nodes that are closer to the starting node.

In the same manner, nodes with many paths between them have more trees that can span them, and thus are more probable to reside in the same tree for a randomly selected forest. For the same reason, the random walker has greater chance to reach (and eventually stop) at a node that is connected to the starting node by multiple paths.

5.2.1 Cholesky decomposition of Q

As described, it is possible to decompose Q into a product of a lower triangular matrix L and its transpose.

Let L be a $p \times p$ lower triangular matrix, where $Q = LL^T$. Since we use L as a linear transformation on the original data samples, every column of L stands for a set of coefficients in a linear combination of all values of a given data sample. The transformation constructs a new set of *meta-features*, such that the value of every meta-feature is obtained as a weighted average of values of all original features. We would like to explore the derivation of the set of weights from the underlying graph.

Let us look at the basics of the Cholesky decomposition:

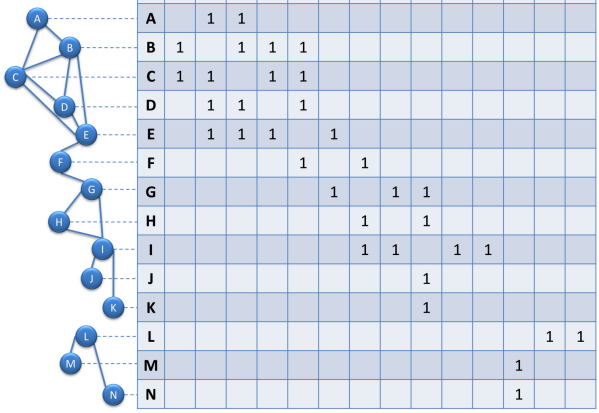
$$\begin{pmatrix} Q_{1,1} & Q_{1,2} & \dots & Q_{1,p} \\ Q_{1,2} & Q_{2,2} & & \\ \vdots & & \ddots & \\ Q_{1,p} & & & Q_{p,p} \end{pmatrix} = \begin{pmatrix} L_{1,1} & 0 & \dots & 0 \\ L_{2,1} & L_{2,2} & 0 & \vdots \\ \vdots & & \ddots & 0 \\ L_{p,1} & \dots & & L_{p,p} \end{pmatrix} \begin{pmatrix} L_{1,1} & L_{2,1} & \dots & L_{p,1} \\ 0 & L_{2,2} & & \vdots \\ \vdots & 0 & \ddots & \\ 0 & \dots & 0 & L_{p,p} \end{pmatrix}$$

Since L is lower triangular, we can see that the first meta-feature, described by the first column, is a combination of all features. Then, we discard the first feature, so that the next column is a combination of all features but the one discarded in the previous column. We will term the features corresponding to $Q_{i,i}$ as *pivot* features, so that every column has a pivot that is discarded in the next row. See Figure 5.1 for a toy net example that illustrates the derivation of L, and demonstrates its final structure and some of its properties.

5.2.1.1 Dominance of Pivot Feature

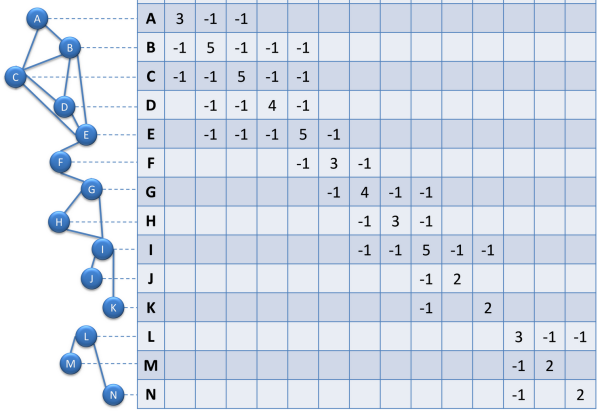
The values on the diagonal of Q hold some interesting properties due to [50] and [49]:

- $Q_{i,j} \leq \frac{Q_{i,i}}{2}$ for every node i and j. Since the matrix is doubly stochastic it is also true that $Q_{i,j} \leq \frac{1}{2}$ for all $i \neq j$.



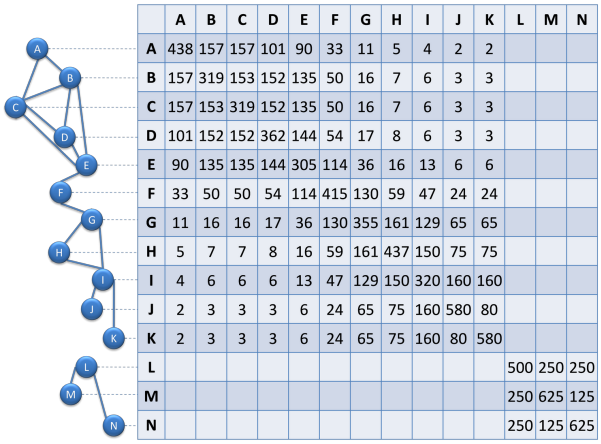
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A		1	1											
B	1		1	1	1									
C	1	1		1	1									
D		1	1		1									
E		1	1	1		1								
F					1		1							
G						1		1	1					
H							1		1					
I							1	1		1	1			
J									1					
K									1					
L													1	1
M												1		
N												1		

(a) **Step 1**: A toy net example with its corresponding adjacency matrix. $A_{i,j} = 1 \iff (i,j) \in E(G)$. Empty entries denote the value 0.



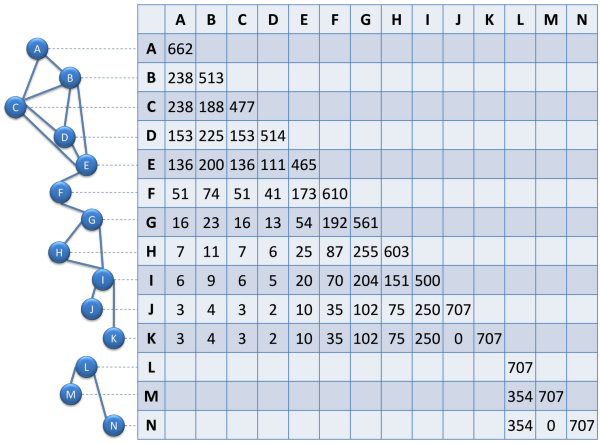
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A	3	-1	-1											
B	-1	5	-1	-1	-1									
C	-1	-1	5	-1	-1									
D		-1	-1	4	-1									
E		-1	-1	-1	5	-1								
F					-1	3	-1							
G						-1	4	-1	-1					
H							-1	3	-1					
I							-1	-1	5	-1	-1			
J									-1	2				
K									-1		2			
L												3	-1	-1
M												-1	2	
N												-1		2

(b) **Step 2**: The matrix $I + \beta B$, ($\beta = 1$) of the same graph, where $B = \tilde{A} - A$ is the graph Laplacian ($\tilde{A}$ is a diagonal matrix, $\tilde{A}_{i,i} = d_i$ is the degree of node i).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A	438	157	157	101	90	33	11	5	4	2	2			
B	157	319	153	152	135	50	16	7	6	3	3			
C	157	153	319	152	135	50	16	7	6	3	3			
D	101	152	152	362	144	54	17	8	6	3	3			
E	90	135	135	144	305	114	36	16	13	6	6			
F	33	50	50	54	114	415	130	59	47	24	24			
G	11	16	16	17	36	130	355	161	129	65	65			
H	5	7	7	8	16	59	161	437	150	75	75			
I	4	6	6	6	13	47	129	150	320	160	160			
J	2	3	3	3	6	24	65	75	160	580	80			
K	2	3	3	3	6	24	65	75	160	80	580			
L												500	250	250
M												250	625	125
N												250	125	625

(c) **Step 3**: Q, the inverse of the matrix from step 2, namely $Q = (I + \beta B)^{-1}$. The matrix is symmetric, and doubly stochastic - every row and every column sum to 1. One can see the dominant values on the diagonal, with decreasing values at each column as the nodes are further away from the node associated with the value on the diagonal. The value associated with two nodes that reside in two different connected components is 0. Values are multiplied by 10^3 for clarity of presentation.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
A	662													
B	238	513												
C	238	188	477											
D	153	225	153	514										
E	136	200	136	111	465									
F	51	74	51	41	173	610								
G	16	23	16	13	54	192	561							
H	7	11	7	6	25	87	255	603						
I	6	9	6	5	20	70	204	151	500					
J	3	4	3	2	10	35	102	75	250	707				
K	3	4	3	2	10	35	102	75	250	0	707			
L												707		
M												354	707	
N												354	0	707

(d) **Step 4**: The transformation matrix L, obtained by Cholesky decomposition of Q, $Q = LL^T$. It is not doubly stochastic, but keeps the dominance property of the values on the diagonal (pivot nodes). The columns are used to build linear combinations of feature values. The first column represents a meta-feature of the first pivot, and is a linear combination of all nodes (in the connected component). In the next column, the previous pivot is discarded, and does not play a role in the meta-feature of the second pivot. Here too, values are multiplied by 10^3.

Figure 5.1: Illustration of features similarity transformation via a toy net example

- $Q_{i,i} \geq \frac{1}{1+d_i}$ where d_i is the degree of node i.

These two properties show the dominance of the pivot feature for a given row in Q when used as a kernel matrix: That the dominant part for every row is still the original feature associated with this row, and if the pivot feature has a small number of neighbors it is bound to be more dominant.

We conjecture that the transformation matrix L holds similar dominance properties of the pivot feature, although L is not doubly stochastic. We observed the properties in all the examples that we tested, but could not prove them.

5.3 Dimension Reduction

In the above setting, for a given sample, p original feature values are transformed into p meta-feature values, each of which is a weighted average of other features' values. If we generate the meta-features according to the order of the rows of L, the i'th meta-feature is a linear combination of exactly $p - i + 1$ features (due to the triangular form of L). Every meta-feature is associated with a single original feature, the pivot feature.

In a scenario of gene expression data, where the dimension of p (the number of genes) is much higher than n (the number of samples), many features are redundant or non-informative, and can harm the analysis, as they introduce noise to the system. We expect that relying on all p original genes would retain high level of noise, and thus we need to reduce the number of genes we base the analysis upon, and adapt the transformation accordingly.

Recall that dimensionality reduction can be obtained by either feature selection or feature extraction, and that two of the advantages of a low number of features is interpretability and cost. For practical purposes, we focus in this work on feature selection methods that also contribute to those two aspects. For example, in order to produce a diagnostic chip tailored for a specific disease, it is desirable to reduce the number of measured genes so that every measurement will be cheaper and more accurate. At the same time, we would like the resulting set of features to be interpretable by experts and allow for downstream research, which is bound to start from a small number of genes.

Adapting our method to the reduced dimension can be done in one of two approaches. In *early selection*, we first select the subset of features to work with and then adapt the method to use only a subset of the features. In *embedded selection*, we embed selection of the features within the method. A third approach, *late selection*, would be to first transform the data using our transformation matrix L, and then select meta-features by using any feature selection method on the transformed data.

Each approach has its pros and cons: in early selection and late selection, we have the advantage that any off-the-shelf method for feature selection can be used. The main disadvantages of early selection are the requirement to adapt the method to work with a small number of features, and the fact that the prior knowledge from the network data does not influence the selection procedure. Embedded selection requires the development of a tailor made method, and does not enable us to use proven and well established feature selection methods and choose the most suitable among them. Late selection results with selection of meta-features, but essentially does not reduce the number of original features we rely on. This is in contrast to our goal at the scenario of gene expression data mentioned above, and thus late selection will not be discussed here.

5.3.1 Early Selection

In early selection, we first select a subset of the features of size m, based on the feature values alone. We shall call that subset the *initial feature set*. A naive approach would be to replace the graph G with a subgraph of G that contains the selected features only and create the transformation matrix L based on the subgraph alone. Although it requires no modifications in the algorithm, we lose almost all the graph data. Only edges that connect two of the selected features will remain, and the global topology of the graph will not be encoded into the transformation matrix. Therefore, this naive approach is not acceptable.

Another option is to build the transformation matrix L based on the whole graph, and adapt it to the selected features by keeping a submatrix that is related to the selected features. Three simple available submatrices define three variants of our early selection:

Row subset Keeping all p columns, but only the m rows that represent the selected features.

Column subset Keeping all p rows, but only the m columns that represent the selected features.

Principal submatrix Keeping only the m columns and m rows that represent the selected features.

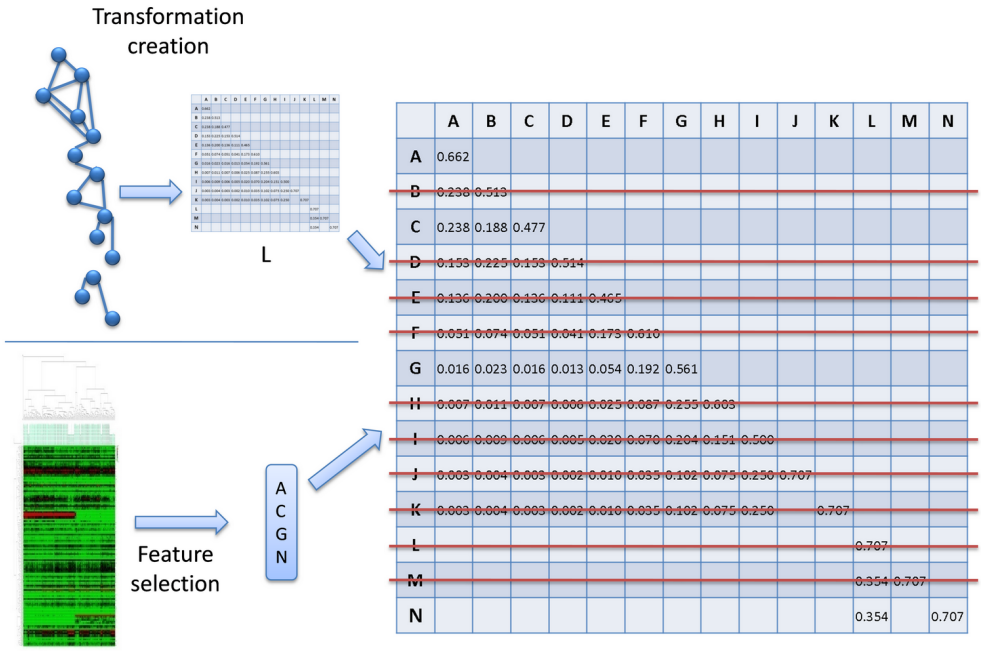
Recall that column i in L stands for a the coefficients of a linear combination that will construct a meta-feature associated with the pivot feature i. The first variant then, will result with m meta-features, each of which is based on p original features. The second variant will result with p meta-features, based on m original features only, and the third variant will result with m meta-features based on m original features. All three variants are illustrated in Figure 5.2. The column subset variant can be used for reducing the number of measured features as the p meta-features actually rely only on the m features of the initial feature set. However, it does not improve the complexity of the learning algorithm, as the number of features is still p. On the contrary, the row subset variant reduces the number of input features for the learning algorithm, but relies on all p original features. Any difficulty associated with the large number of features that might influence the classifier's performance is expected to propagate to the algorithm when the row subset variant is used.

The principal submatrix variant (selecting both row and column subset) is cheap both in term of computational power and number of original features measured. However, if the selected features are sparse over the network, the network will not come in hand since every meta-feature will be highly similar to its pivot-feature and the weights of all other features will be very small.

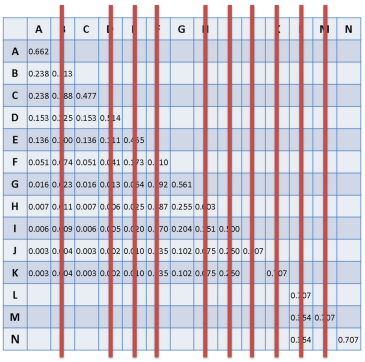
5.3.2 Embedded Selection

A generalization of SVM-RFE (see Section 2.1.4.6) is proposed in [29], dealing with kernels other than the linear kernel. Using the SVM dual form, the authors offer to look at the difference in the objective function resulting from removing the k'th feature. The original objective function is:

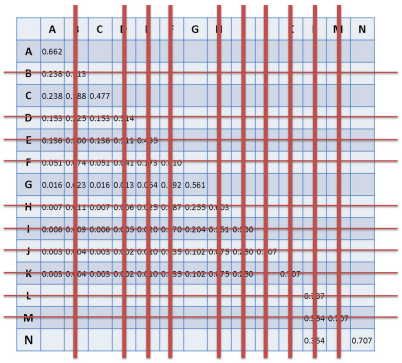
$$J = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T Q \mathbf{x}_j$$



(a) **Row subset variant.** L is created based on the whole network (top left corner of the figure), preserving all data encapsulated in the network. In parallel, feature selection is applied to expression data alone selecting m features (A, C, G, N in the example, bottom left corner of the figure). Then, columns and rows of L corresponding the features are selected. The example illustrates the column subset variant, where all p (14) columns are kept, yielding p (14) meta-features constructed of m (4) original features. Column C, for example, produces a new meta-feature, associated with the original pivot feature C. It is a combination of original features C and G, with weights 0.477 and 0.016, respectively.



(b) **Column subset variant.** Each of the 4 meta-features is associated with one selected pivot features and is a linear combination of all features.



(c) **Principal matrix variant.** We construct only 4 meta-features, based on the 4 selected features.

Figure 5.2: Illustration of early feature selection variants.

and the difference between the objective function and the one resulting from removing the k'th feature is:

$$\begin{aligned}\Delta J(k) &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T Q \mathbf{x}_j + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T Q_{(-k)} \mathbf{x}_j \\ &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T Q_{(+k)} \mathbf{x}_j\end{aligned}$$

where the $\cdot_{(-k)}$ notation denotes the removal of the k'th component, and $\cdot_{(+k)}$ notation denotes the restriction of term (vector or matrix) to the k'th component only. Note that $Q_{(+k)}$ is a symmetric $p \times p$ matrix with all zeros, except in the k'th row and the k'th column (due to symmetry), with the k'th row (column) being the vector of coefficients corresponding the k'th feature computed from the graph as described in Section 5.2.

6

Experimental Results

In this chapter we present experimental results of applying our algorithm on a number of cancer datasets taken from literature.

6.1 Data

We collected nine labeled gene expression datasets, used for binary classification. The datasets are listed in Table 6.1. All datasets relate to cancer prognosis, aiming at differentiating tumors of patients with good prognosis from those with poor prognosis, as reflected by survival time, or metastasis free progression of the disease. Six of the datasets are taken from breast cancer tissues, and three from lung cancer. We aimed at testing for improvement of our method over baseline classification performance, and detecting whether the improvement indeed originates from prior knowledge, and is not an artifact of the method, regardless of the integrated knowledge.

As a reference network, we used STRING [36] to be our prior knowledge source for obtaining similarities between genes. The network we used includes 6243 genes (nodes) and 19102 interactions (edges).

6.2 Testing for Improvement

Since our method is built on top of SVM, we realized that it would be best to take SVM as our baseline method for comparison. Recall (Equation 5.2) that we added an additional term to the SVM objective function, together with a tradeoff parameter $\beta \geq 0$. The larger β is, the more stress is put onto the prior knowledge, as reflected by proximity

Dataset	Author	Reference	Details	n
GSE5123	Larsen	[43]	Lung Cancer	51
GSE4573	Raponi	[60]	Lung Cancer	130
vant' Veer	vant' Veer	[72]	Breast Cancer	117
E-TABM158	Chin	[16]	Breast Cancer	118
GSE2034	Wang	[75]	Breast Cancer	286
GSE3141	Nevins	[11]	Lung Cancer	111
VanDeVijver	van de Vijver	[71]	Breast Cancer	295
GSE4922; GSE1456	Ivshina	[34]	Breast Cancer	99
Pawitan	Pawitan	[57]	Breast Cancer	159

Table 6.1: The datasets used in the experimental results. "Author" refers to the first author of the study.

between features. Note that with $\beta = 0$, our formulation is equivalent to the standard L_2-SVM.

As a performance measure, we used the AUC criterion, using 5-fold cross validation. Because of the large number of features, we also used early feature selection as described in 5.3.1.

For each dataset, we first selected the 2000 genes with highest variance among the samples, regardless of their labeling. We then chose G to be the subnetwork of our reference network, restricted to the proteins that are products of the selected 2000 genes. This was done due to computational limitations, keeping in mind that the reduced network should still be large enough to preserve enough of the original network's topology, and the genes' relevance to the expression data. Using the transformation derived from Equations 5.7 and 5.6 we obtained the transformation matrix L from G.

We used 5-fold cross validation to get an average AUC score for each tested value of the parameter β: for each fold we selected one fifth of the data to serve as a test set, and the remaining four fifths were used to train the classifier. For every training set, we first selected 250 genes that best discriminate the two classes within the training set by applying Student's *t*-Test on every gene and keeping those 250 genes that obtained the highest t-Test score.

We used the selected genes to restrict our transformation matrix L, as described in Section 5.3.1 to get the submatrix L' using all three variants, and transformed the original training data using L'. We then trained a Linear SVM with the transformed training set to

get a SVM classifier. In the same manner, we transformed the original test data using L', and measured the AUC using the classifier obtained in the training phase.

After repeating the training and testing procedure for all five folds, we averaged the AUC score. We repeated the whole folding procedure 20 times and obtained an average of all AUC scores from all repeats as our final performance measure for the method on a given dataset. We further tested whether the improvement is indeed statistically significant based on those 20 repeats.

For every dataset, we repeated the measurements for β values of 0 (equivalent to standard SVM), 0.05, 0.1, 0.5, 1, 2, 5 and 10. For every value of β we used exactly the same folds in order to decrease the noise in the AUC results that is due to the random choice of folds. We tested this way all three early selection variants as described in Section 5.3.1.

Overall, the row subset variant was the most successful early selection variant, showing improvement in six of the nine datasets. The principal submatrix variant did not show any change in performance, and the column subset variant showed a monotonic decrease in performance for increasing values of β or did not show any change in performance.

Final results are shown in Figure 6.1 for the row subset variant and in Figure 6.2 for the other two variants. Figure 6.3 shows a comparison of the improvement using the three variants.

We also tested for the influence of the number of features on the performance of the algorithm. As discussed in Section 5.3, feature selection is almost obligatory in the context of gene expression profiles. However, if we want our method to benefit from the network, we can not select a small number of features that will significantly decrease the network's connectivity. For example, on the Wang dataset [75], selecting the 50 top genes using *t*-test results with only four genes whose nodes are not isolated. Figure 6.4 compares the performance of algorithm for different number of genes selected.

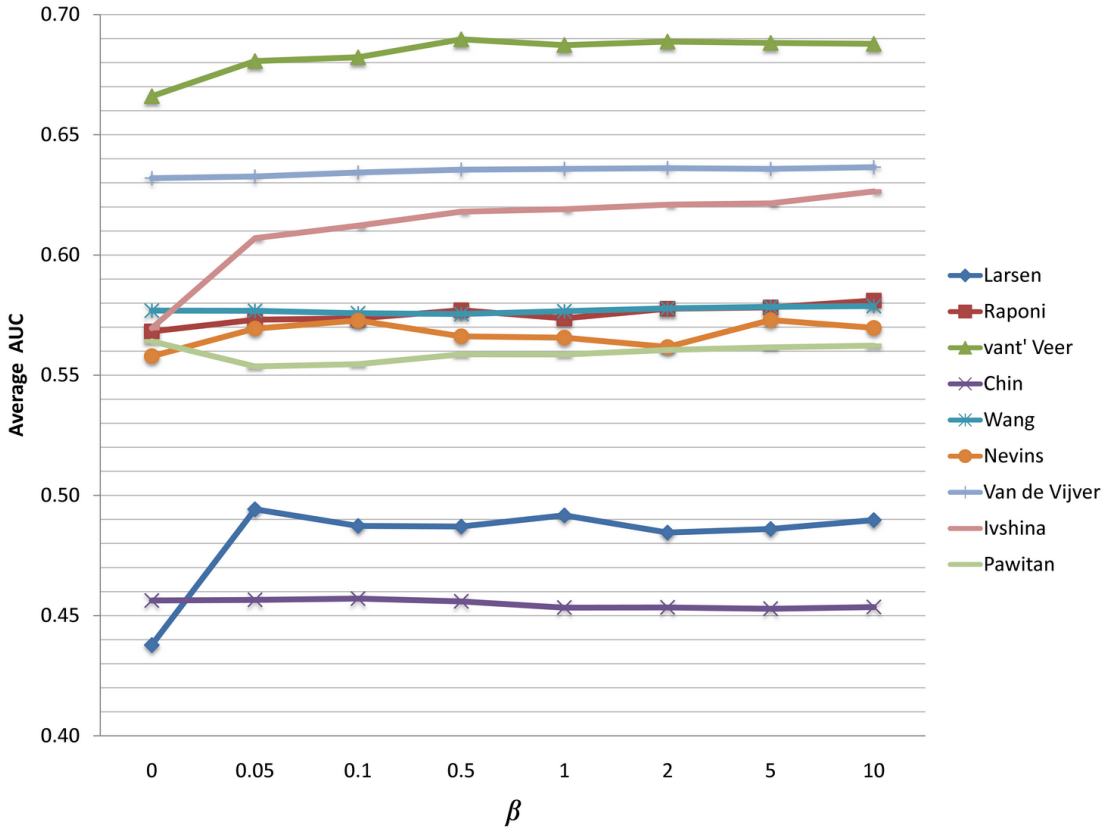
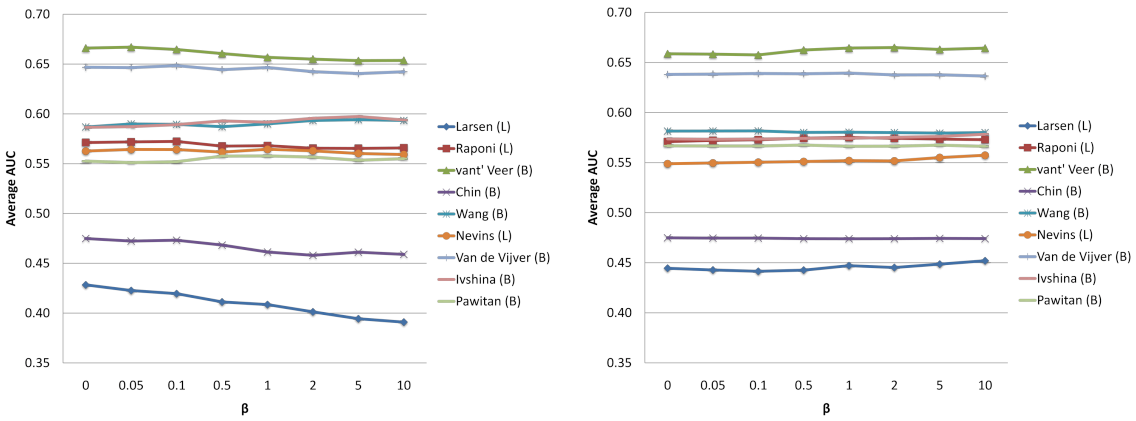


Figure 6.1: **Classification performance comparison for the row subset variant**. The figure displays average Area Under ROC curve measurements for SVM classification of different binary datasets, using our network kernel with different values of β, reflecting different weights given to the prior knowledge as obtained from the STRING network. $\beta = 0$ serves as a baseline and is equivalent to standard SVM.



(a) (b)

Figure 6.2: **Classification performance comparison for column subset variant and principal submatrix variant**. The figure displays average area under ROC curve measurements for the column subset early selection variant (a) and the principal submatrix early selection variant (b).

6.2.1 Improvement Significance

There are two main sources of randomness in the above results. The first is due to the random data stratification, which leads to different choices of actual folds. The second is due to the method itself, and specifically due to creation of a submatrix of L, based on features selected in the early selection phase, and the order of the adjacency matrix representing the graph. Each row (and column) in the adjacency matrix corresponds to a single node in the graph. Later on, when meta-features are created due to the resulted transformation matrix, they are influenced by the ordering of these rows (and columns).

While mathematically, the objective function is indifferent to this ordering, the early selection of features leads to an approximation of the original transformation matrix, (and as a consequence also only approximates the original objective function), an approximation which is a function of the selected features and the order of nodes in the adjacency matrix.

We looked for statistically significant improvement throughout the different choice of folds, and different ordering of the nodes. To this end, we conducted a pairwise t-Test to compare the mean AUC score of the baseline SVM result with that of other values of β. Out of the nine dataset, six (Larsen [43], Raponi [60], van t' Veer [72], Nevins [11],

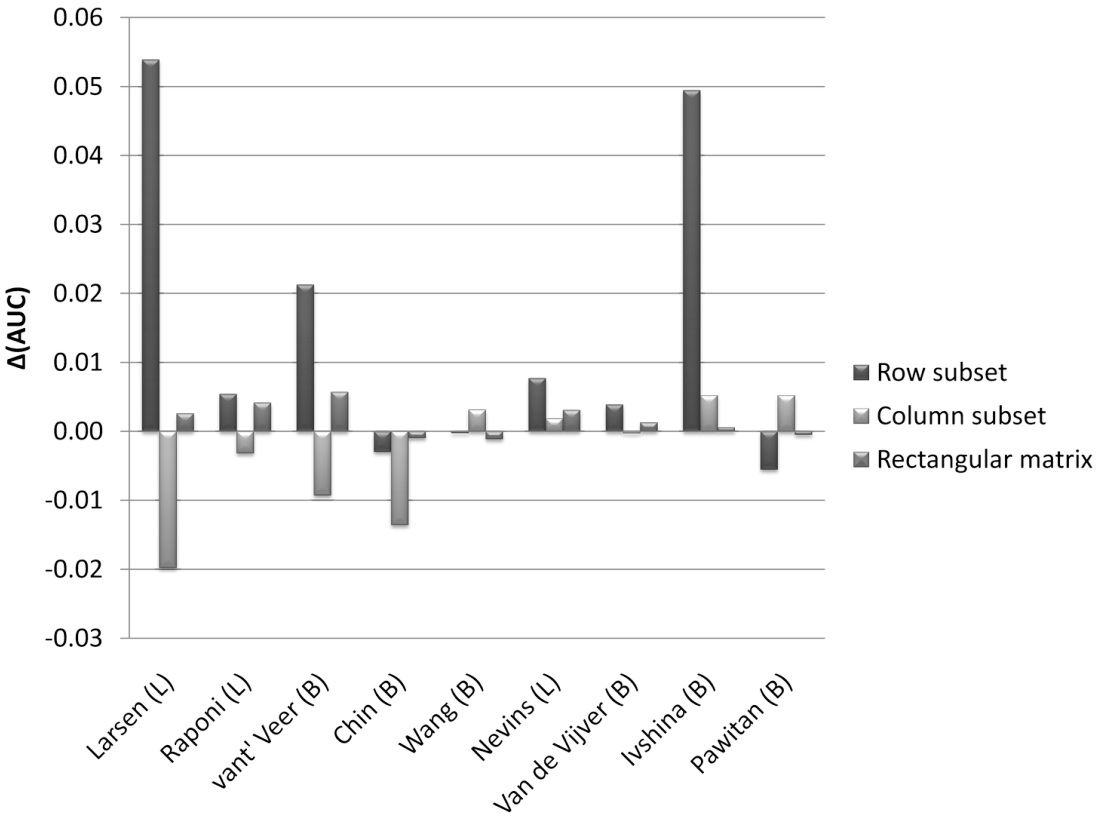
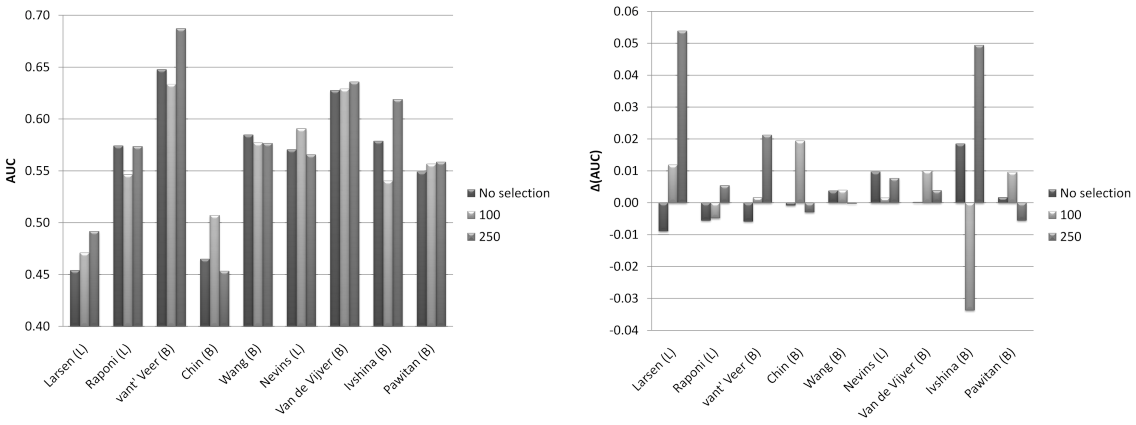


Figure 6.3: **Difference in AUC scores for the three early selection variants.** The difference in average AUC scores between the baseline SVM ($\beta = 0$) and SVM using our network kernel ($\beta = 1$) are shown for the nine datasets.



(a) (b)

Figure 6.4: **The effect of the different number of features**. The figure displays average area under ROC curve measurements using no feature selection (with top 2000 genes according to their variance), and for the row subset early selection variant using 100 and 250 features. Since the baseline for different number of features might be different, we present both absolute AUC values with $\beta = 1$ (a) and difference between standard SVM ($\beta = 0$) and our algorithm ($\beta = 1$) (b)

Van de Vijver [71] and Ivshina [34]) showed an improvement with all values of β, and two (Chin [16] and Wang [75]) showed a mixed result, with an increased performance for some values of β and decreased for others. A single dataset (Pawitan [57]) showed a performance decrease for all values of β. However, none of the results for the mixed or decreased performance datasets were significant (FDR$<$0.05) while the improvement in two of the datasets (Ivshina [34] and Larsen [43]) that showed better results was found to be significant. All significance tests were corrected for multiple testing, accounting for the multiple datasets, and multiple values of β. We will discuss the selective improvement of the algorithm and the fact that it did not significantly worsen the results in any of the datasets later on in Section 7.4.

Based on these analyses, we chose to focus on the row subset early selection variant with 250 features and will use it from now on.

6.2.2 Choice of β

Larger values of β smooth the distribution of weights of original features composing a single meta-feature. As $\beta \to \infty$, a meta-feature turns into a simple average of all features in its connected component. We observed that in cases that the algorithm indeed improve the results, it did so as soon as a positive β was introduced, with mild changes as β was further increased from $\beta = 0.05$ to $\beta = 10$. We therefore used $\beta = 1$ as the default value.

6.2.3 Network Randomization

In order to test whether the improvement is due to the network data, and not due to other factors pertaining to the algorithm, we further tested the algorithm performance with randomized networks. Randomization was done as described in Section 4.1 where we tested for the network influence on two existing algorithms.

We generated 50 different random networks as described in Section 6.2 and ran the algorithm using each network on three of the datasets that the algorithm showed improvement on, van 't Veer, van de Vijver and Ivshina. For every random network we ran the algorithm 20 times, each time using 5-fold CV. We measured the average AUC score for each of the runs, comparing it to the average AUC result obtained with the original STRING network. Figure 6.5 summarizes the results. On all three datasets the AUC score with the real network ranked above all AUC scores achieved with random networks, a result that is similar to the one we achieved with the same test on the HyperGene algorithm (see Section 3.6.2). The scores using the real network on van 't Veer and Ivshina datasets were 0.687 and 0.619 respectively, and the average scores of random networks were 0.652 ± 0.0128 and 0.564 ± 0.022. For comparison, on the Wang [75] dataset, the real network is ranked 35 among the 50 randomized networks.

6.3 Comparison to Other Methods

In Section 3.6 we reviewed several methods that integrate network data into gene expression analysis, and briefly summarized the results of each method. In order to com-

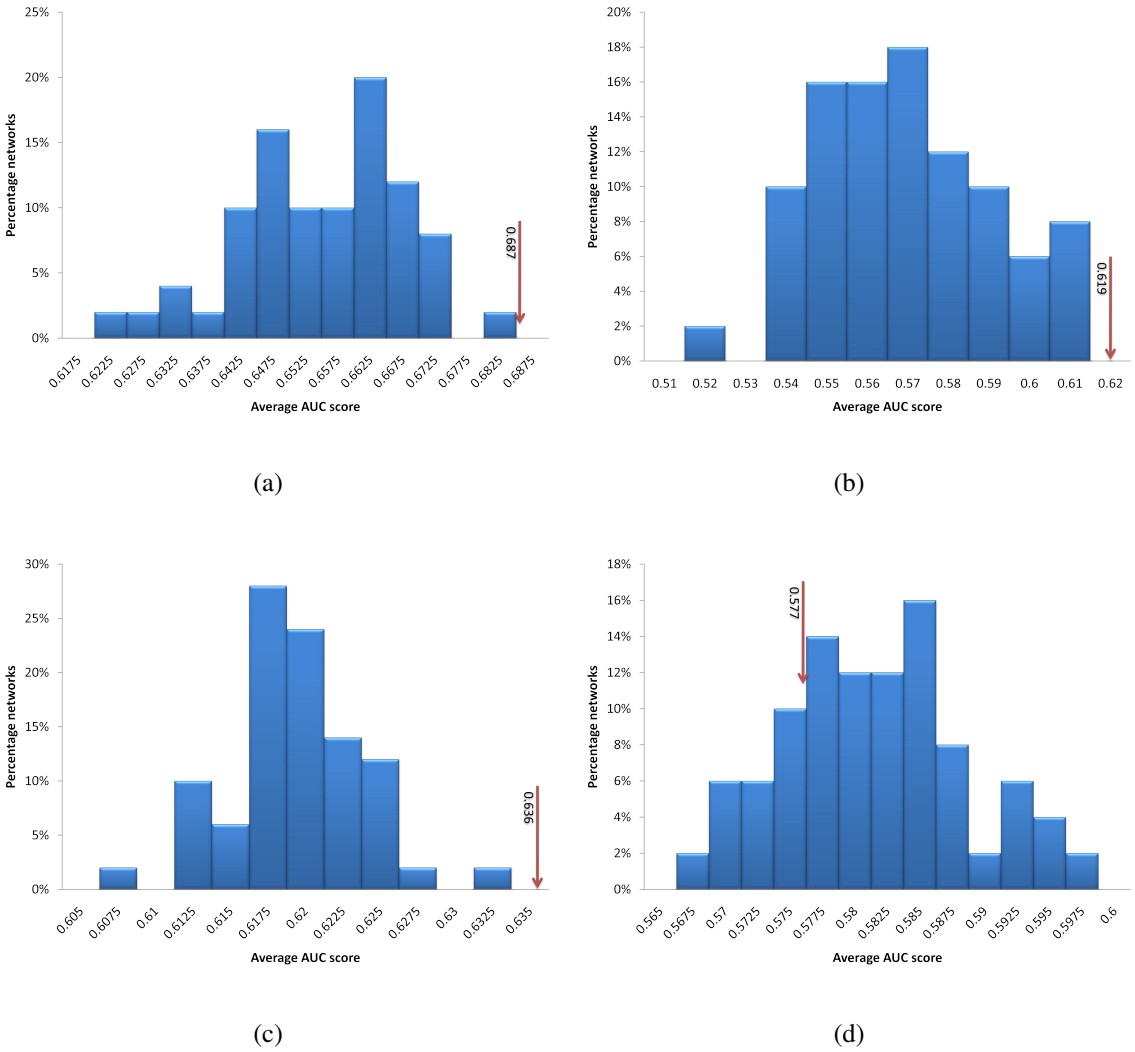


Figure 6.5: **Distribution of AUC scores in random networks vs. real network.** Results for the van 't Veer **(a)**, Ivshina **(b)**, van de Vijver **(c)** and Wang **(d)** datasets. Each plot shows a histogram of AUC scores obtained by running the algorithm with 50 different randomized STRING networks. The red arrow denotes the average AUC obtained by running the algorithm with the real network. (a), (b) and (c) show datasets that the method exhibited improvement on. In these cases, the real network is ranked above all randomized network runs. In (d), on a dataset where the method did not show improvement, the real network is ranked 35 among the 50 randomized networks.

pare our results with those of previous methods, we focus on those methods that use the network for classification and use a standard performance measure for the task. Different methods report different performance measures, on various data, and utilize different methods of preprocessing the data.

For example, Hwang et al. [32] test the performance of the HyperGene algorithm (see Section 3.6.2) when starting from a small number of genes, either by measuring the feature correlation with the labels, or by using predefined sets from the literature. Chuang et al. [17], emphasizing the robustness of their feature extraction method, report reciprocal performance of their algorithm: They construct a set of features based on one dataset, and test its performance on another. In addition, they use performance measures other than AUC. We noticed large differences in the performance reported by authors of the different methods, even when using the same performance measure on the same dataset using the same classification algorithm. We attribute the gaps mainly to the preprocessing step and further discuss this point in Section 7.5.

Among the classification methods reviewed, HyperGene achieved the best results on a number of datasets, and as shown in Section 4.2 the network indeed plays a part in its improvement over other methods. We compared the performance of our algorithm with that of HyperGene using HyperGene's MATLAB® implementation kindly provided to us by the authors. For comparison, we used four breast cancer datasets - van 't Veer [72], van de Vijver [71], Ivshina [34] and Wang [75]. In order to compare our method to HyperGene, we used the same data, network and preprocessing steps, and implemented our method in MATLAB so we could also compare running times. On one dataset (van 't Veer) we used the data and network preprocessed by Hwang et al., and for the three other datasets we used the STRING network and preprocessed the data ourselves. Preprocessing included standardization of the data (with mean 0 and standard deviation 1 for every gene) for the Wang and Ivshina datasets attaining, to conform the requirements of the HyperGene algorithm, and filtering the data, keeping $2,000$ most variable genes.

Due to the large computational requirements of using quadratic programming in HyperGene, its authors limited the size of the network: They first selected a sub-network of $1,000$ nodes that correspond to the genes most correlated with the labels, and then selected a subset of these genes to reduce the feature dimension of the data used for learning. This subnetwork may lose a lot of the connectivity of the original network, and in order

to preserve more of the topology of the original network outside of these $1,000$ nodes, the authors added edges among these $1,000$ nodes, connecting two nodes by an edge if there exists a path of length 1 or 2 between them in the original network. For the van 't Veer comparison we used this network, and for the other three datasets, we used a STRING sub-network of size $2,000$ based on the $2,000$ genes with the highest variance without the added edges.

Throughout our experiments described in Section 6.2 we first filtered the data by selecting the top $2,000$ genes with highest variance. Such a filter is unsupervised as it does not rely on the samples' labels. Hwang et. al correlation-based filter does rely on the labels, and thus calculating an average AUC based on cross validation would be biased, and their method of choice here was testing the AUC score on an independent test set based on the original paper by van 't Veer et al. Once again, for sake of comparison on the van 't Veer dataset, we also followed this choice here. On the three other datasets we used 5-fold cross validation selecting the top correlated features separately for each fold based on the training data alone.

Hwang et. al reported their results for 500 top correlated genes, while we chose 250 as the feature set size during our experiments. We compared the algorithms with sets of different sizes ranging from 25 to 500. We also compared the two methods with their baselines - SVM for the network kernel, and network propagation classification [81] (the term network in this algorithm refers to an internal representation of the samples and features within the algorithm, and not to the PPI network integration). Table 6.2 and Figure 6.6 summarize the comparison results. In some cases, the quadratic programming solver failed during runs of the HyperGene algorithms before optimization process was finished. The HyperGene score in these cases could be low due to the incomplete optimization.

Our net-based kernel SVM ranks first in 7 of the 20 cases tested, the net propagation algorithm ranks first in 7 others, SVM ranks first in 4 cases, and HyperGene ranks first in two cases. HyperGene ranks last in 15 of the 20 cases. While the gaps between the best and second best scores are sometimes very small, the gap between the best and worst scores is often quite large.

Dataset	# features	Classifier			
		NetProp	**HyperGene**	**SVM**	**SVM+Net Kernel**
Ivshina	25	0.6289	*0.4893*	0.6336	**0.6665**
	50	**0.6756**	*0.5352*	0.6725	0.6327
	100	0.6238	*0.6015* *	**0.6366**	0.6103
	250	0.6034	**0.6114** *	0.6098	*0.5904*
	500	0.6184	*0.5606* *	0.6084	**0.6266**
Wang	25	0.6466	*0.6456*	**0.6592**	0.6562
	50	0.6398	*0.6123*	0.6713	**0.6732**
	100	0.6609	*0.5958*	0.6886	**0.6918**
	250	0.6782	*0.6007*	**0.6937**	0.6861
	500	**0.6792**	*0.5805* *	0.6584	0.6623
van de Vivjer	25	0.7225	0.7224	*0.7186*	**0.7257**
	50	**0.7268**	0.7196	0.7114	*0.6788*
	100	**0.7316**	*0.6977*	0.7262	0.7208
	250	0.743	*0.6757*	0.755	**0.7564**
	500	0.7456	*0.7023*	0.7508	**0.7578**
van 't Veer	25	**0.8452**	0.7738	*0.7381*	0.7857
	50	**0.8333**	*0.7976*	0.8095	0.8214
	100	**0.8452**	*0.7143*	0.8214	0.8333
	250	0.8214	*0.8095*	**0.8333**	0.8214
	500	*0.8214*	**0.869**	0.8333	0.8333

Table 6.2: **Performance Comparison.** Area under ROC curve results of four algorithms with four breast cancer datasets. The table shows the AUC average of 5-fold cross validation for Ivshina, Wang and van de Vijver datasets, and an AUC for a single run on the original training and test set from van 't Veer based on data compiled by Hwang et al. The table compares HyperGene and SVM with our network-based kernel, with their corresponding baseline algorithm (algorithms that do not use PPI network data) - net propagation as HyperGene's baseline and linear kernel SVM as the baseline to our kernel. Numbers in bold (italics) indicate the highest (lowest) score among the four algorithms in each row. The * mark denotes incomplete runs of the HyperGene algorithm aborted by the quadratic programming solver.

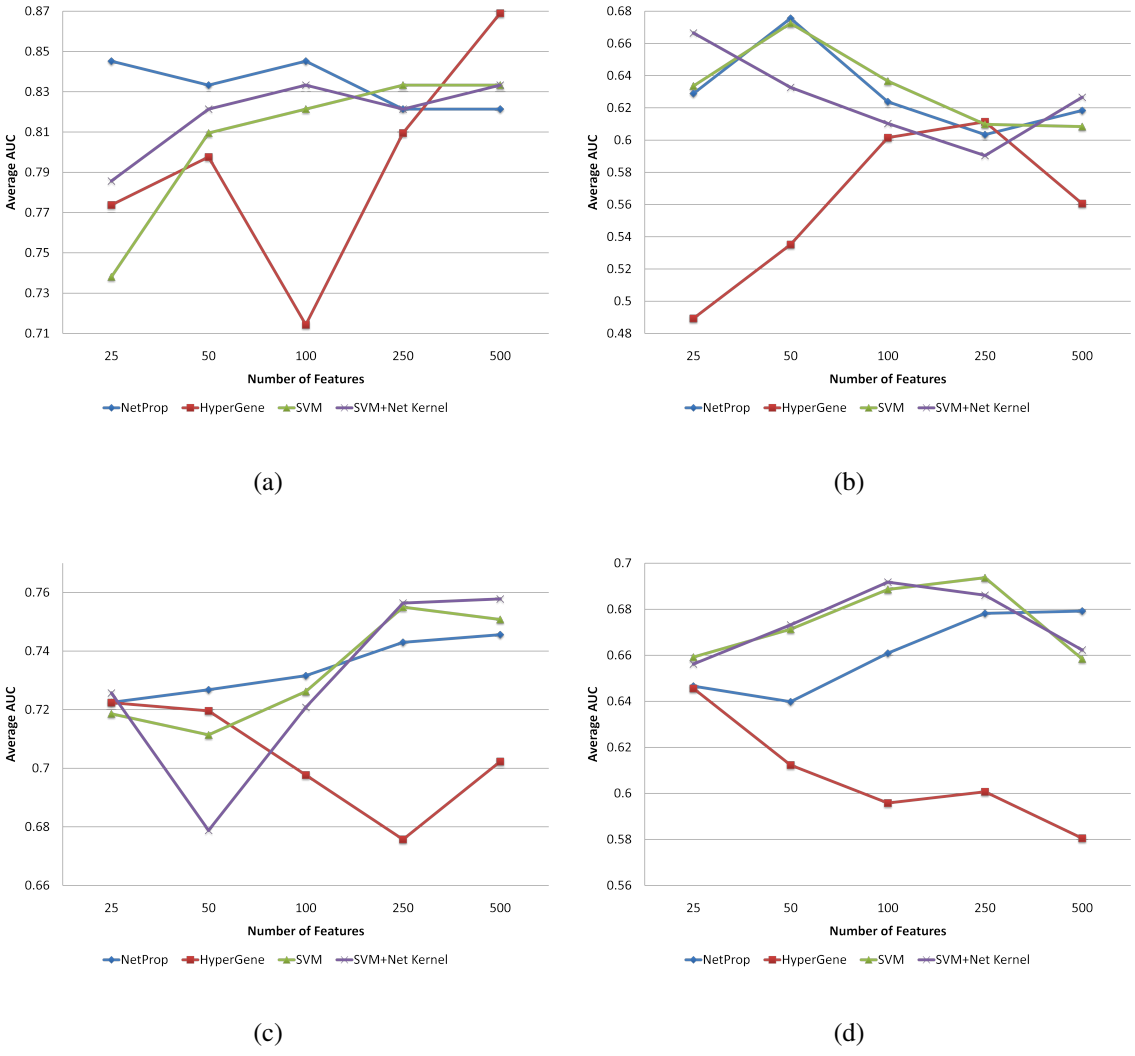
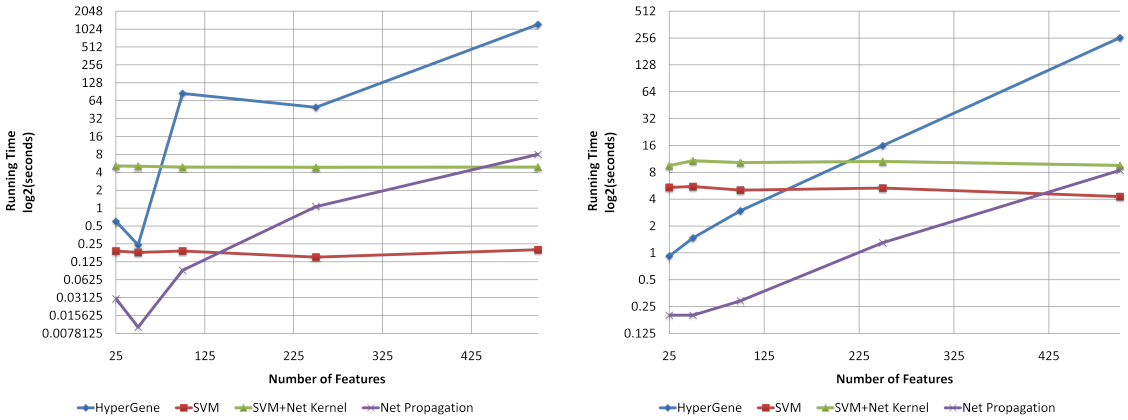


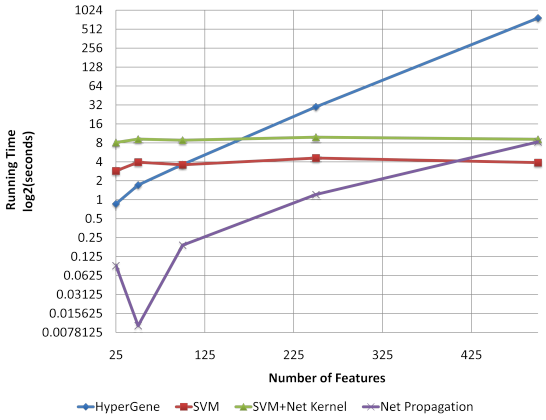
Figure 6.6: **Performance Comparison.** The figure presents the data in Table 6.2 comparing four different algorithms on four breast cancer datasets - **(a)** van 't Veer. **(b)** Ivshina. **(c)** van de Vijver. **(d)** Wang. See Table 6.2 for full details.

We ran the four algorithms on a 2-quad core intel Xeon 5160 at 2.33 Ghz with 16GB memory running 64 bit Linux using MathWorks MATLAB ver. 7.2. The process only utilized a single core. Figure 6.7 shows a comparison of running times. Times include preprocessing and training on a single fold times. Both HyperGene and our net-based kernel SVM require some preprocessing of the data using matrix operations. However, while following this preprocessing we simply run plain linear SVM, HyperGene runs an iterative process solving a number of quadratic programming problems. For example, using 500 features on the Ivshina dataset, with a network of $2,000$ nodes and $9,914$ edges, our net-based kernel SVM takes less than 5 seconds to run a single fold, which is about 250 times faster than HyperGene. Note that our algorithm and SVM take roughly constant time, while HyperGene and NetProp show running times growing exponentially with the feature set size.



(a)

(b)



(c)

Figure 6.7: **Running Time Comparison.** The figure shows running times of the four algorithms on different datasets with different number of features. For c clear presentation, time is displayed in log scale (seconds). **(a)** Ivshina. **(b)** van de Vijver. **(c)** Wang.

7

Conclusions and Future Work

We started this work investigating a number of extant methods used for integrating biological knowledge encapsulated in biological networks into gene expression data analysis. Following a systematic categorization of the different methods we ended up developing a novel method for integrating knowledge regarding pairwise relations between features into the process of learning a classifier based on these features. We applied the method to gene expression based classification with large-scale protein interaction network as our prior-knowledge resource under the assumption that close and highly connected proteins in the network should have similar contribution of their corresponding genes to the classification model.

In Sections 7.1 and 7.2 we will go over the advantages and limitations of our proposed method. In Section 7.3 we will discuss the influence of the network in the scope of feature selection, and in Section 7.4 we will try to point out factors that make the use of PPI network valuable in the scope of gene expression classifications. We will conclude by describing promising directions for future work.

7.1 The Advantages of the Method

A major advantage of the method is its simplicity, due to the new kernel and transformation introduced. Given a graph (or any non-negative similarity matrix), obtaining the kernel matrix Q is straightforward. The presented decomposition of the kernel matrix allows for reducing the original problem to the standard SVM problem by first transforming the data using the transformation matrix L. After the transformation any current SVM machinery can be used - solvers, implementations etc. As seen in Section 6.3 the method

does not involve any search procedure over the network, and is thus fast, and scales well with the network size and the number of features.

Although SVM is a supervised classification algorithm, and the kernel and transformation were derived based on labeled data, eventually the kernel and the transformation are not label-dependent. In fact, they depend only on the network itself, while the data and labels information is moved to the constraints. Interpretation of the transformation matrix is quite straightforward. It is very clear how a meta-feature is constructed from its neighbor features, and the network topology is directly reflected in the weights of original features in the meta-feature. Since the matrices are independent of the data, the transformation and kernel can be applied to unsupervised methods such as clustering or unsupervised feature selection and extraction as well. In particular, methods that use $\|\mathbf{w}\|_2^2$ regularization, such as Ridge regression [31] can justifiably use our regularization term or kernel.

7.2 Limitations of the Method

One of the goals of applying the data to gene expression data using protein interaction networks, was to reduce the inherent noise in gene expression data. However, by doing so, we introduce a new source of noise to the system, in the form of the network data itself. PPI networks are known to have both false positives - i.e. include false edges, and false negatives - i.e. miss some true edges. In addition, they are constructed by and large by in-vitro experiments that can not mimic the exact in-vivo conditions. Also, semantically, they are constructed of pairwise relations, and it is hard to interpret paths and topology of the whole network, as different conditions may yield different sets of edges, and the PPI networks exhibit all the edges together. All of these limitations are also true for of other algorithms that use PPI network data.

In addition, one may doubt the assumption that close genes should contribute similarly to the classification model. In the extreme case the opposite may be true, e.g. when one protein suppresses the function of another protein. Also, the leap from mRNA levels in gene expression to protein interactions in the network level is not trivial, and as we have shown in Section 4.1.1 the signal is not very strong.

Although the transformation matrix directly reflects the network topology, it has some limitations. The first is due to the global nature of the transformation. A single meta-feature can be a weighted average of all features in a single connected component. That means that even the most distant features contribute to the meta-feature's value, albeit in very small weights. One can argue that this fact is not supported by biology. Also, recall that the transformation matrix is a triangular matrix, which poses two problems in interpreting it. The first is that the meta-feature corresponding to the first column includes all original features in a given connected component, and as we advance in the columns of the matrix, meta-features corresponding to the columns include less and less features. Such a set contains highly overlapping meta-features (in terms of original features they contain) with a large variability in size. These meta-features can not directly correspond to biological pathways or complexes, which are disjoint by and large. This makes every feature by itself hard to interpret biologically. In addition, every permutation of the nodes in the initial adjacency matrix will produce a different transformation matrix, which makes such an interpretation even harder.

The transformation also does not reduce the dimension of the data, neither by selecting a subset of the original features, nor by extracting a small number of new meta-features. The number of meta-features is identical to the number of original features.

Finally, derivation of the method is tightly coupled with SVM and thus the obtained transformation that is based on the network is meaningful in the scope of SVM. However, it is not clear whether it is possible to use it as a general transformation that smoothes data according to the network topology in other frameworks. Although the obtained kernel is applicable in other kernel methods, its exact meaning and interpretation may not be transferrable as is from the SVM formulation, and thus such an application may not be justified.

7.3 Influence of the Network on Feature Selection

At the initial steps of this research, we asked ourselves a major question regarding networks and feature selection in the scope of gene expression profiles: Will the network push the feature selection algorithm towards selecting neighbor genes or distant genes? The assumptions underlying the question were:

- Close genes tend to operate together.
- A similar biological phenotype can be a result of a few different biological processes.

Even the first assumption alone can lead us to opposite conclusions: On the one hand, if indeed close genes tend to work together, maybe a single gene is enough for representing a specific neighborhood of genes, as its neighbors are redundant after selecting it. On the other hand, due to the noisy data, it might be justified to select more than one gene from a given neighborhood and increase the statistical robustness this way.

The second assumption suggests that we might want to force the algorithm to select genes from distant neighborhoods. Assume a disease with multiple causes, where the major cause is responsible for the vast majority of cases. Although we could have identified the other cases by looking at genes related to their (rare) cause, the genes related to the major cause would obscure the effect of the rest of the genes, at least when using a simple ranking algorithm based on differential expression based on *t*-test or mere correlations with labels.

One can try direct graph-based approaches for optimizing some density function related to the genes over the network, explicitly requiring to select sparse genes, or sparse sub-networks. Eventually, our method avoids the question by utilizing the assumption as is, without explicitly directing the algorithm towards selecting neither close nor distant genes. The decision is left for the algorithm itself, with the assumption integrated into its objective function.

7.4 When is the Network Informative?

In section 4.2 we compared two cases of algorithms that use network data for classification purposes. In the algorithm of Chuang et al. [17] (pinnacleZ, see Section 3.6.1) the network did not seem to improve the classification results, while in the algorithm of Hwang et al. [32] (HyperGene, see Section 3.6.2) it did. We would like to note a number of differences that may be the cause for the difference.

Network size While pinnacleZ uses the whole network, HyperGene, due to complexity

and performance issues, uses subnetworks of size $300 - 1000$ of genes selected in advance. These sets can be selected based on ranking of gene absolute correlation with the labels. Alternatively, they can be chosen manually by experts based on the specific disease under study.

Pairwise relation The subnetwork used by HyperGene is highy disconnected. To overcome this, Hwang et al. treated every pair of genes that are 1 or 2 nodes away in the original full network as adjacent. In contrast, the standard adjacency definition was used in pinnacleZ by Chuang et al.

Integration assumption HyperGene uses only pairwise relation among genes, minimizing the total difference between weights assigned to neighbor genes. In contrast, pinnacleZ greedily maximizes the correlation of a whole subnetwork to be used as a meta-feature with the labels as an indication for high discriminative potential.

Looking at our algorithm, it exhibited improvement in performance only in some datasets we used, while leaving the performance comparable in the rest. The performance improvement did not require high values of β - in cases the network indeed helped, it did so even when the network component weight was modest. We think that in some cases, and for some datasets, the relevant sections in the network are richer (i.e. have less false negative edges), while in other cases they are still very incomplete. The reason for this may be a bias towards more researched areas (this is especially true for edges based on co-citations). However, in datasets that did not exhibit improvement the network did not worsen the results, as missing edges simply leave the original data intact.

7.5 Performance Improvement

We showed that the method can improve results over a baseline classifier that does not use network prior information. However, the work would be incomplete without referring to the fact that a large variability in the measured performance is found in different works, even using the same dataset. This variability can be attributed to preprocessing of the data; the difference such preprocessing can make is sometimes much higher than the improvement obtained by incorporating prior knowledge. Specifically, we noticed this difference when looking at SVM results reported by Hwang et al. compared to those we

obtained on the same dataset. Using the same preprocessing canceled this difference and allowed us to perform a fair comparison between the methods. Nevertheless, it seems that a lot of research should still take place regarding the preprocessing steps of gene expression data - normalization, noise reduction and dimension reduction before going on to apply any specific classification algorithm or integrating more data such as network data into the process.

7.6 Possible Extensions and Applications to Other Areas

We think that integration of biological networks into analysis of expression data will continue to be an important research topic for improving the analysis. Even if more accurate gene expression data are provided by next generation sequencing (NGS) methods, the gap between the feature and sample dimensions will not be closed soon. However, gene expression analysis can greatly benefit from more refined networks, such as condition-dependant PPI networks, networks containing more accurate information regarding the in-vivo behavior of proteins, and large scale regulatory networks, when these become available.

Our method uses the network but does not perform feature selection. Other methods have used the network to detect disease causing genes [38, 73] but have not gone ahead to use these genes as classification biomarkers. We believe that the classification performance may benefit from integrating the network during a preliminary feature selection step as described in Section 5.3.2.

One problem that we raised has to do with the difficulty to interpret the transformation obtained from the original network and the possible redundancy of the resulting meta-features. It is possible to modify the Cholesky decomposition to account for a-priory dependencies among the meta-features. According to this modification [1], during the Cholesky decomposition, we eliminate a row and a column associated with a pivot-feature if the value of the pivot feature is smaller than some threshold. The resulting rows are closer to be linearly independent, and the overlap among the meta-features is reduced.

It is also possible to integrate networks of more than one type to create a combined, possibly weighted kernel. Such networks can encapsulate different types of prior knowl-

edge. In the scope of genomic data, we can integrate both protein interaction networks, metabolic networks, regulatory networks and functional similarity networks that are built using different annotations such as GO [7].

The kernel matrix Q can be used to partition the graph in a soft manner. For the problem of clustering a graph merely according to its topology, it is possible to apply to Q clustering algorithms that operate on similarity matrices. This way, the clustering algorithms can benefit from the similarity measure defined by Q.

In addition, there are many kernel methods that can use Q as a network-based kernel. Although the derivation of Q started from a supervised framework, obtaining Q from a given graph does not require labeled data. For this reason, Q can be used for both supervised and unsupervised learning.

Network data are being used in machine learning tasks in many other fields where our algorithm may be applied. Often, network data are used to link between samples, while our method links between features. In some cases the assumption that close features on the network exhibit similar behavior is more straightforward than in the case of PPI networks and gene expression. We propose a number of such fields that can utilize our algorithm:

1. **Brain imaging - physical distance network.** In brain imaging, a 3-dimensional matrix of datapoints is created from brain scanning in a number of conditions. The matrix represents spatial data such that each datapoint (named *voxel*) corresponds to a specific location in the brain and gives the level of activity of that location under the given condition. In one setting, one wishes to find regions in the brain that are responsible for specific functions. It is possible to create a network of the voxels such that adjacent voxels in the space would have an edge connecting them in the network. This would form some kind of a grid, where the assumptions that close voxels function together is applicable.

2. **Social networks - the trivia team example.** Suppose one wishes to form a trivia team out of a set of candidates. One way to do it is to create questionnaires, and select people according to their answers to the questionnaire. Here again, it is likely that friends are similar in some way, and share the same fields of interest and knowledge. To this end, one can represent binary friendship relations by a network,

and integrate the friendship network might help us to select a better team.

3. **Document classification - using words network.** Using a standard "bag of words" setting, where features correspond to word occurrences, one can use an ontology network where the nodes are words e.g. WordNet [23], and edges represent semantic or syntactic relations between the words (e.g. synonyms), under the assumption that similar words are linked in the network.

Finally, the World Wide Web (WWW) is also a good candidate network that might be suitable for the similarity assumption between a page and the pages it links to. However, in this case the application of our method is not straightforward and must be developed, both because the WWW forms a directed graph and because the pages in our formulations should serve as features for learning on other types of samples. One example could be the classification of a visitor based on the set of pages she visited, under the assumption that linked pages are more likely to suggest the same class for the user. Classes can differentiate between different user interests, age group, potential for repeating visits etc.

Bibliography

[1] S. Abe. *Support Vector Machines for Pattern Classification*. Springer Publishing Company, Incorporated, 2010.

[2] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Garland Science, 5 edition, November 2007.

[3] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States of America*, 96(12):6745–6750, 1999.

[4] E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theor. Comput. Sci.*, 209(1-2):237–260, 1998.

[5] B. Aranda, P. Achuthan, Y. Alam-Faruque, I. Armean, A. Bridge, C. Derow, M. Feuermann, A. T. Ghanbarian, S. Kerrien, J. Khadake, J. Kerssemakers, C. Leroy, M. Menden, M. Michaut, L. Montecchi-Palazzi, S. N. Neuhauser, S. Orchard, V. Perreau, B. Roechert, K. van Eijk, and H. Hermjakob. The intact molecular interaction database in 2010. *Nucl. Acids Res.*, page gkp878, 2009.

[6] S. Asgharzadeh, R. Pique-Regi, R. Sposto, H. Wang, Y. Yang, H. Shimada, K. Matthay, J. Buckley, A. Ortega, and R. C. Seeger. Prognostic Significance of Gene Expression Profiles of Metastatic Neuroblastomas Lacking MYCN Gene Amplification. *J. Natl. Cancer Inst.*, 98(17):1193–1203, 2006.

[7] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. *Nat Genet*, 25(1):25–29, May 2000.

[8] P. Baldi and G. W. Hatfield. *DNA Microarrays and Gene Expression*. Cambridge University Press, Cambridge, UK, 2002.

[9] D. Beer, S. Kardia, C. Huang, T. Giordano, A. Levin, D. Misek, L. Lin, G. Chen, T. Gharib, and D. Thomas. Gene-expression profiles predict survival of patients with lung adenocarcinoma. *Nat Med*, 8(8):816–824, 2002.

[10] A. Bhattacharjee, W. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, and M. Gillette. Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses. *Proc Natl Acad Sci USA*, 98(24):13790–13795, 2001.

[11] A. H. Bild, G. Yao, J. T. Chang, Q. Wang, A. Potti, D. Chasse, M.-B. B. Joshi, D. Harpole, J. M. Lancaster, A. Berchuck, J. A. Olson, J. R. Marks, H. K. Dressman, M. West, and J. R. Nevins. Oncogenic pathway signatures in human cancers as a guide to targeted therapies. *Nature*, 439(7074):353–357, January 2006.

[12] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM.

[13] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[14] P. Chebotarev. Spanning forests and the golden ratio. *Discrete Appl. Math.*, 156(5):813–821, 2008.

[15] P. Chebotarev and E. Shamis. The matrix-forest theorem and measuring relations in small social groups. *CoRR*, abs/math/0602070, 2006.

[16] K. Chin, S. DeVries, J. Fridlyand, P. T. Spellman, R. Roydasgupta, W.-L. Kuo, A. Lapuk, R. M. Neve, Z. Qian, T. Ryder, F. Chen, H. Feiler, T. Tokuyasu, C. Kingsley, S. Dairkee, Z. Meng, K. Chew, D. Pinkel, A. Jain, B. M. Ljung, L. Esserman, D. G. Albertson, F. M. Waldman, and J. W. Gray. Genomic and transcriptional aberrations linked to breast cancer pathophysiologies. *Cancer Cell*, 10(6):529 – 541, 2006.

[17] H.-Y. Y. Chuang, E. Lee, Y.-T. T. Liu, D. Lee, and T. Ideker. Network-based classification of breast cancer metastasis. *Molecular systems biology*, 3, October 2007.

[18] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.

[19] S. Efroni, C. F. Schaefer, and K. H. Buetow. Identification of key processes underlying cancer phenotypes using biologic pathway analysis. *PLoS ONE*, 2(5):e425, 2007.

[20] L. Ein-Dor, I. Kela, G. Getz, D. Givol, and E. Domany. Outcome signature genes in breast cancer: is there a unique set? *Bioinformatics*, 21(2):171–178, 2005.

[21] L. Ein-Dor, O. Zuk, and E. Domany. Thousands of samples are needed to generate a robust gene list for predicting outcome in cancer. *PNAS*, 103(15):5923–5928, 2006.

[22] R. Elkon, R. Vesterman, N. Amit, I. Ulitsky, I. Zohar, M. Weisz, G. Mass, N. Orlev, G. Sternberg, R. Blekhman, J. Assa, Y. Shiloh, and R. Shamir. Spike - a database, visualization and analysis tool of cellular signaling pathways. *BMC Bioinformatics*, 9(1):110, 2008.

[23] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database (ISBN: 0-262-06197-X)*. MIT Press, first edition, 1998.

[24] D. Geman, C. d'Avignon, D. Q. Naiman, and R. L. Winslow. Classifying gene expression profiles from pairwise mrna comparisons. *Statistical Applications in Genetics and Molecular Biology*, 3(1):19, 2004.

[25] V. E. Golender, V. V. Drboglav, and A. B. Rosenblit. Graph potentials method and its application for chemical information processing. *Journal of Chemical Information and Computer Sciences*, 21(4):196–204, 1981.

[26] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[27] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, October 1999.

[28] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[29] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3):389–422, 2002.

[30] J. Herschkowitz, K. Simin, V. Weigman, I. Mikaelian, J. Usary, Z. Hu, K. Rasmussen, L. Jones, S. Assefnia, S. Chandrasekharan, M. Backlund, Y. Yin, A. Khramtsov, R. Bastein, J. Quackenbush, R. Glazer, P. Brown, J. Green, L. Kopelovich, P. Furth, J. Palazzo, O. Olopade, P. Bernard, G. Churchill, T. Van Dyke, and C. Perou. Identification of conserved gene expression features between murine mammary carcinoma models and human breast tumors. *Genome Biology*, 8(5):R76, 2007.

[31] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[32] T. Hwang, Z. Tian, R. Kuang, and J.-P. Kocher. Learning on weighted hypergraphs to integrate protein interactions and gene expressions for cancer outcome prediction. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 293–302, Washington, DC, USA, 2008. IEEE Computer Society.

[33] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences of the United States of America*, 98(8):4569–4574, 2001.

[34] A. V. Ivshina, J. George, O. Senko, B. Mow, T. C. Putti, J. Smeds, T. Lindahl, Y. Pawitan, P. Hall, H. Nordgren, J. E. Wong, E. T. Liu, J. Bergh, V. A. Kuznetsov, and L. D. Miller. Genetic Reclassification of Histologic Grade Delineates New Clinical Subtypes of Breast Cancer. *Cancer Res*, 66(21):10292–10301, 2006.

[35] R. Jansen, D. Greenbaum, and M. Gerstein. Relating Whole-Genome Expression Data with Protein-Protein Interactions. *Genome Research*, 12(1):37–46, 2002.

[36] L. J. Jensen, M. Kuhn, M. Stark, S. Chaffron, C. Creevey, J. Muller, T. Doerks, P. Julien, A. Roth, M. Simonovic, P. Bork, and C. von Mering. String 8–a global view on proteins and their functional interactions in 630 organisms. *Nucl. Acids Res.*, 37(1):D412–416, 2009.

[37] M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucl. Acids Res.*, 28(1):27–30, 2000.

[38] S. Karni, H. Soreq, and R. Sharan. A network-based method for predicting disease-causing genes. *Journal of Computational Biology*, 16(2):181–189, 2009.

[39] C. M. Kendziorski, M. A. Newton, H. Lan, and M. N. Gould. On parametric empirical bayes methods for comparing multiple groups using replicated gene expression profiles. *Stat Med*, 22(24):3899–3914, December 2003.

[40] S. Kerrien, Y. Alam-Faruque, B. Aranda, I. Bancarz, A. Bridge, C. Derow, E. Dimmer, M. Feuermann, A. Friedrichsen, R. P. Huntley, C. Kohler, J. Khadake, C. Leroy, A. Liban, C. Lieftink, L. Montecchi-Palazzi, S. E. Orchard, J. Risse, K. Robbe, B. Roechert, D. Thorneycroft, Y. Zhang, R. Apweiler, and H. Hermjakob. Intact - open source resource for molecular interaction data. *Nucleic Acids Research*, 35(Database-Issue):561–565, 2007.

[41] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1143. Morgan Kaufmann, 1995.

[42] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *In Proceedings of the ICML*, pages 315–322, 2002.

[43] J. E. Larsen, S. J. Pavey, L. H. Passmore, R. Bowman, B. E. Clarke, N. K. Hayward, and K. M. Fong. Expression profiling defines a recurrence signature in lung squamous cell carcinoma. *Carcinogenesis*, 28(3):760–766, 2007.

[44] E. Lee, H.-Y. Chuang, J.-W. Kim, T. Ideker, and D. Lee. Inferring pathway activity toward precise disease classification. *PLoS Comput Biol*, 4(11):e1000217, 11 2008.

[45] E.-S. Lee, D.-S. Son, S.-H. Kim, J. Lee, J. Jo, J. Han, H. Kim, H. J. Lee, H. Y. Choi, Y. Jung, M. Park, Y. S. Lim, K. Kim, Y. M. Shim, B. C. Kim, K. Lee, N. Huh, C. Ko, K. Park, J. W. Lee, Y. S. Choi, and J. Kim. Prediction of Recurrence-Free Survival in Postoperative Non Small Cell Lung Cancer Patients by Using an Integrated Model of Clinical Information and Gene Expression. *Clinical Cancer Research*, 14(22):7397–7404, 2008.

[46] I. Lee, S. V. Date, A. T. Adai, and E. M. Marcotte. A Probabilistic Functional Network of Yeast Genes. *Science*, 306(5701):1555–1558, 2004.

[47] C. Li and H. Li. Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics*, 24(9):1175–1182, 2008.

[48] G. J. McLachlan, K.-A. Do, and C. Ambroise. *Analyzing microarray gene expression data / Geoffrey J. McLachlan, Kim-Anh Do, Christopher Ambroise*. Wiley-Interscience, Hoboken, N.J. :, 2004.

[49] R. Merris. Doubly stochastic graph matrices. *Publ. Elektrotech. Fak. Univ. Beograd*, 1997.

[50] R. Merris. Doubly stochastic graph matrices ii. *Linear and Multilinear Algebra*, 45(2):275–285, 1998.

[51] T. Mitchell. *Machine Learning (Mcgraw-Hill International Edit)*. McGraw-Hill Education (ISE Editions), 1st edition, October 1997.

[52] A. Müller, B. Homey, H. Soto, N. Ge, D. Catron, M. E. Buchanan, T. Mcclanahan, E. Murphy, W. Yuan, S. N. Wagner, J. L. Barrera, A. Mohar, E. Verástegui, and A. Zlotnik. Involvement of chemokine receptors in breast cancer metastasis. *Nature*, 410:50–56, March 2001.

[53] M. A. Newton, C. M. Kendziorski, C. S. Richmond, and F. R. Blattner. On differential variability of expression ratios: Improving statistical inference about gene expression changes from microarray data. *Journal of Computational Biology*, 8:37–52, 2001.

[54] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 14, 2001.

[55] D. Nitsch, L.-C. Tranchevent, B. Thienpont, L. Thorrez, H. Van Esch, K. Devriendt, and Y. Moreau. Network analysis of differential expression for the identification of disease-causing genes. *PLoS ONE*, 4(5):e5526, 05 2009.

[56] S. Paik, G. Tang, S. Shak, C. Kim, J. Baker, W. Kim, M. Cronin, F. L. Baehner, D. Watson, J. Bryant, J. P. Costantino, C. E. Geyer, D. L. Wickerham, and N. Wolmark. Gene expression and benefit of chemotherapy in women with node-negative, estrogen receptor-positive breast cancer. *J Clin Oncol*, 24(23):3726–3734, August 2006.

[57] Y. Pawitan, J. Bjohle, L. Amler, A.-L. Borg, S. Egyhazi, P. Hall, X. Han, L. Holmberg, F. Huang, S. Klaar, E. Liu, L. Miller, H. Nordgren, A. Ploner, K. Sandelin, P. Shaw, J. Smeds, L. Skoog, S. Wedren, and J. Bergh. Gene expression profiling spares early breast cancer patients from adjuvant therapy: derived and validated in two population-based cohorts. *Breast Cancer Research*, 7(6):R953–R964, 2005.

[58] H. S. Phillips, S. Kharbanda, R. Chen, W. F. Forrest, R. H. Soriano, T. D. Wu, A. Misra, J. M. Nigro, H. Colman, and L. Soroceanu. Molecular subclasses of high-grade glioma predict prognosis, delineate a pattern of disease progression, and resemble stages in neurogenesis. *Cancer Cell*, 9(3):157–173, March 2006.

[59] F. Rapaport, A. Zinovyev, M. Dutreix, E. Barillot, and J. Vert. Classification of microarray data using gene networks. *BMC Bioinformatics*, 8(1):35, 2007.

[60] M. Raponi, Y. Zhang, J. Yu, G. Chen, G. Lee, J. M. Taylor, J. MacDonald, D. Thomas, C. Moskaluk, Y. Wang, and D. G. Beer. Gene Expression Signatures for Predicting Prognosis of Squamous Cell and Adenocarcinomas of the Lung. *Cancer Res*, 66(15):7466–7472, 2006.

[61] J. F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G. F. Berriz, F. D. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, N. Klitgord, C. Simon, M. Boxem, S. Milstein, J. Rosenberg, D. S. Goldberg, L. V. Zhang, S. L. Wong, G. Franklin, S. Li, J. S. Albala, J. Lim, C. Fraughton, E. Llamosas, S. Cevik, C. Bex, P. Lamesch, R. S. Sikorski, J. Vandenhaute, H. Y. Zoghbi, A. Smolyar, S. Bosak, R. Sequerra, L. Doucette-Stamm, M. E. Cusick, D. E. Hill, F. P. Roth, and M. Vidal. Towards a proteome-scale map of the human protein–protein interaction network. *Nature*, 437(7062):1173–1178, September 2005.

[62] S. C. Schuster. Next-generation sequencing transforms todayś biology. *Nat Methods*, 5(1):8–16, 2008.

[63] A. Seth, R. Kitching, G. Landberg, J. Xu, J. Zubovits, and A. Burger. Gene expression profiling of ductal carcinomas in situ and invasive breast tumors. *Anticancer Res*, 23:2043–2051, 2003.

[64] R. Shamir, A. Maron-Katz, A. Tanay, C. Linhart, I. Steinfeld, R. Sharan, Y. Shiloh, and R. Elkon. Expander - an integrative program suite for microarray data analysis. *BMC Bioinformatics*, 6(1):232, 2005.

[65] D. Singh, P. G. Febbo, K. Ross, D. G. Jackson, J. Manola, C. Ladd, P. Tamayo, A. A. Renshaw, A. V. D'Amico, J. P. Richie, E. S. Lander, M. Loda, P. W. Kantoff, T. R. Golub, and W. R. Sellers. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1(2):203 – 209, 2002.

[66] B. Snel, G. Lehmann, P. Bork, and M. A. Huynen. String: a web-server to retrieve and display the repeatedly occurring neighbourhood of a gene. *Nucl. Acids Res.*, 28(18):3442–3444, 2000.

[67] T. Sørlie, R. Tibshirani, J. Parker, T. Hastie, J. S. Marron, A. Nobel, S. Deng, H. Johnsen, R. Pesich, S. Geisler, J. Demeter, C. M. Perou, P. E. Lønning, P. O. Brown, A.-L. Børresen-Dale, and D. Botstein. Repeated observation of breast tumor subtypes in independent gene expression data sets. *Proceedings of the National Academy of Sciences of the United States of America*, 100(14):8418–8423, 2003.

[68] U. Stelzl, U. Worm, M. Lalowski, C. Haenig, F. H. Brembeck, H. Goehler, M. Stroedicke, M. Zenkner, A. Schoenherr, S. Koeppen, J. Timm, S. Mintzlaff, C. Abraham, N. Bock, S. Kietzmann, A. Goedde, E. Toks?z, A. Droege, S. Krobitsch, B. Korn, W. Birchmeier, H. Lehrach, and E. E. Wanker. A human protein-protein interaction network: A resource for annotating the proteome. *Cell*, 122(6):957 – 968, 2005.

[69] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 102(43):15545–15550, 2005.

[70] Z. Tian, T. Hwang, and R. Kuang. A hypergraph-based learning algorithm for classifying gene expression and arraycgh data with prior knowledge. *Bioinformatics*, pages btp467+, July 2009.

[71] M. J. van de Vijver, Y. D. He, L. J. van 't Veer, H. Dai, A. A. Hart, D. W. Voskuil, G. J. Schreiber, J. L. Peterse, C. Roberts, M. J. Marton, M. Parrish, D. Atsma, A. Witteveen, A. Glas, L. Delahaye, T. van der Velde, H. Bartelink, S. Rodenhuis, E. T. Rutgers, S. H. Friend, and R. Bernards. A Gene-Expression Signature as a Predictor of Survival in Breast Cancer. *N Engl J Med*, 347(25):1999–2009, 2002.

[72] L. J. van 't Veer, H. Dai, M. J. van de Vijver, Y. D. He, A. A. Hart, M. Mao, H. L. Peterse, K. van der Kooy, M. J. Marton, A. T. Witteveen, G. J. Schreiber, R. M. Kerkhoven, C. Roberts, P. S. Linsley, R. Bernards, and S. H. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415(6871):530–536, January 2002.

[73] O. Vanunu, O. Magger, E. Ruppin, T. Shlomi, and R. Sharan. Associating genes and protein complexes with disease via network propagation. *PLoS computational biology*, 6(1):e1000641+, January 2010.

[74] V. Vapnik. *The Nature of Statistical Learning Theory (Information Science and Statistics)*. Springer, 2nd edition, November 1999.

[75] Y. Wang, J. G. Klijn, Y. Zhang, A. M. Sieuwerts, M. P. Look, F. Yang, D. Talantov, M. Timmermans, M. E. Meijer-van Gelder, J. Yu, T. Jatkoe, E. M. Berns, D. Atkins, and J. A. Foekens. Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *Lancet*, 365(9460):671–679, 2005.

[76] A. R. Webb. *Statistical Pattern Recognition, 2nd Edition*. John Wiley & Sons, October 2002.

[77] P. Wei and W. Pan. Incorporating gene networks into statistical tests for genomic data via a spatially correlated mixture model. *Bioinformatics*, 24(3):404–411, 2008.

[78] Z. Wei and H. Li. A Markov random field model for network-based analysis of genomic data. *Bioinformatics*, 23(12):1537–1544, 2007.

[79] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. Use of the zero norm with linear models and kernel methods. *J. Mach. Learn. Res.*, 3:1439–1461, 2003.

[80] J. R. Yates, C. I. Ruse, and A. Nakorchevsky. Proteomics by mass spectrometry: Approaches, advances, and applications. *Annual Review of Biomedical Engineering*, 11(1):49–79, 2009.

[81] D. Zhou, J. Huang, and B. Sch?lkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems (NIPS) 19*, page 2006. MIT Press, 2006.

[82] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Neural Information Processing Systems*, page 16. MIT Press, 2003.

[83] Y. Zhu, X. Shen, and W. Pan. Network-based support vector machine for classification of microarray samples. *BMC Bioinformatics*, 10(Suppl 1):S21, 2009.