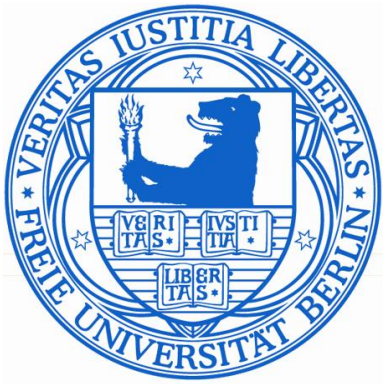
# A new approach for biomarker detection using fusion networks

**Master thesis**
**Bioinformatics**




Mona Rams

June 15, 2016

Department Mathematik und Informatik
Freie Universität Berlin

## Statutory Declaration

I declare that I completed this work on my own and that information which has been directly or indirectly taken from other sources has been noted as such. Neither this, nor a similar work, has been published or presented to an examination committee.

| | |
|---|---|
| Place, date | Mona Rams |

# Contents

# 1. Abstract

In this thesis we propose a new approach for biomarker detection using single source and fusion networks. Our algorithm detects metastable regions with similar and high weights in networks.

Standard methods for biomarker detection analyse the differentially expression, or other biological measurement, of genes, without taking any further biological knowledge into account. Network approaches include further insight into the analysis, nevertheless most of them are limited to the examination of one data type.

With our fusion network algorithm we introduce a new and promising approach. We analyse breast cancer gene expression and methylation data using our fusion network approach. The proposed method detects known and novel biomarkers, which are highly supported by breast cancer literature, biological pathways, and classification power.

# 2. Introduction

Biomarker[1] discovery for complex diseases derived from molecular data is a widely used strategy to gain deeper insights into disease pathogenesis. Their presence as primary endpoints in clinical studies is widely accepted. In a complex disease such as cancer, multiple biomarkers are relevant, resulting in a disease module. Module based biomarkers achieve greater predictive power and reproducibility compared to single gene biomarkers [49]. In this thesis, we propose a new algorithm to identify disease relevant modules in a fusion network (network with multiple integrated data measurements).

Commonly applied approaches for the selection of relevant disease biomarkers are extensive feature selection methods, such as support vector machine [16], elastic net [90], and random forest [11]. These methods are applied to classify individual observations into distinct classes, which results in a model with rankings of features. The most important features are considered as biomarkers. We introduce these feature selection methods in §4 and apply them on real world biological data.

It has been observed that biomarkers from different studies often have only few overlap [26]. A reason for this is that often many features have similar discriminatory power, each method selects from with a different approach. Furthermore, these methods select on mathematical background only and hence do not give a (biological) interpretation for the weighting of features. Hence, the crucial step of disease relevant biomarker discovery is, to obtain biomarkers with a reasonable biological argumentation.

Studies applying these standard methods investigate experimental measurements only. They assume that the significant differential expression of a gene, or another biological measurement, between two conditions is sufficient enough to classify it to be pathogenic [18, 62, 87]. This assumption neglects the fact that impacts of individually altered proteins can often be balanced, whereas alterations of interacting proteins often lead to major consequences [7, 12]. Hence, in this thesis we suggest taking a network approach which includes biological knowledge into the analysis (§4).

Clustering approaches for the detection of interesting regions in a network often consider the topology of the network only [33, 43, 53, 89]. These approaches do not include the analysis of experimental measurements for the observed disease. Our network algorithm is a continuous time random walk based approach, to identify regions (metastable sets) of interacting, high weight nodes in a network. The weight of a node (gene/ protein) corresponds to its change in disease, compared to a control group. Hence, our algorithm incorporates network connectivity and experimental measurements. We test our algorithm in simulation studies, and apply it on real world biological data.

[1]The term biomarker refers to a medical response (substance, structure, or process) that can be measured objectively, accurately, and reproducibly in the body or its products, representing an indicator of a medical state in a biological system [15, 55, 80].

Approaches for biomarker detection in biological networks that analyse more than only the network topology, often evaluate topological features of weighted biological networks without additional data, or by a combination of network and one data type [45, 83, 85]. These approaches do not make use of data fusion. With the term *fusion*, we refer to an approach that integrates more than one data type of experimental measurements. Results obtained from any single omic approach, however, should be interpreted cautiously [27], because single source analysis is not sufficient for an understanding of complex biological processes, and because omics data often contains lots of noise [35]. Firstly, real life biological systems contain more than one component, and should thus be analysed based on multiple data types. Secondly, if one of several measurements of the same sample is corrupted by noise, then integrating them can help to detect and discard the noise.

In recent years, the availability of high throughput datasets has increased considerably, quantifying thousands of cellular components. This is a great advantage for the field of systems biology, as it allows for a comprehensive analysis of molecular and clinical datasets. Methods integrating omics data, thus making use of the available diversity of data, are required to improve predictive capabilities of computational models [10].

Xia et al. [84] propose a fusion network analysis algorithm for the identification of candidate genes in a biological network. They define a selection of 'seed proteins' around which a network is built as the first main task. This method is often applied in big data analysis [40, 71]. Indeed, it makes data analysis easier because it reduces the problem, but it adds a strong bias to the problem [20]. An algorithm for network based biomarker detection, going beyond one dimensional approaches and not making use of initial data restriction, is a progress compared to these methods. We therefore adjust our single source network algorithm to a fusion network algorithm presented in §5, which we test in simulation studies and apply on real world biological data. In §6, we identify and compare breast cancer gene expression and methylation biomarkers from four different approaches:

- Machine learning based biomarker detection for single source data
- Biomarker detection with our single data network approach
- Machine learning based biomarker detection for fusion data
- Biomarker detection with our fusion network approach

Breast cancer is one of the most common cancers and one of the leading cause of cancer related death among women worldwide [67]. The Cancer Genome Atlas (TCGA) [46] data repository provides a comprehensive characterisation of breast cancer including the genomic and epigenetic level.

Our fusion network approach reveals biomarkers which are highly supported by breast cancer literature, biological pathways, and classification power.

# 3. Machine learning methods for biomarker detection

Machine learning based feature selection is a widely applied approach in biomarker detection [44, 56, 69].

Multiple methods are relevant in this context. In biological analysis data is usually of the format *number of features >> number of samples*. Features can be genes, proteins, or other biological entities. We describe three of the most common methods which are suited for such data sets: Support vector machine (SVM), random forest, and elastic net.

A detailed explanation of all the three methods would go beyond the scope of this thesis. We explain the SVM method in detail, because it is nowadays more often applied than the other methods (on Google Scholar, citations since 2012 using SVM are three times more frequent than citations using the other two methods) and give a brief overview of elastic net and random forest.

## 3.1. Methods

Supervised machine learning approaches are classification (discrete/ categorical output variable), or regression (continuous output variable) problems. In classification approaches, the aim is to identify, which class a new observation belongs to, on the basis of a set of training data observations, with known class label. The result is a model consisting of scored features. The features with the highest scores are selected in a machine learning based feature selection.

The input to a classification is a matrix of the format $samples \times features$, and a class label $y_i$ for each sample. A vector $\vec{x_i} \in \mathbb{R}^m$, $m >> 1$ the number of features, of all feature values for one sample is one data point. Each method observes this data matrix to detect the feature relevance for the classification of each point.

For the application of the methods, we use the R package CARET with functions SVMRADIAL, RF, and GLMNET and 10-fold cross validation with 5 repeats. The TRAIN function in CARET includes optimisation of method parameters.

### 3.1.1 Support vector machines

SVM is a binary linear classifier that can deal with linear and non-linear separation problems.

It separates points of different classes by means of a separating hyperplane. The hyperplane is obtained by a maximisation of the margins, which are the distances of the hyperplane to its closest training data points. These data points are called support vectors.

The hyperplane can be parametrised by a vector $\vec{\omega}$ and a scalar $b$. All points $\vec{x}$ on the hyperplane satisfy $\vec{\omega}^T\vec{x} = -b$, where $b$ is the intercept term and $\vec{\omega}$, the normal vector to the hyperplane.

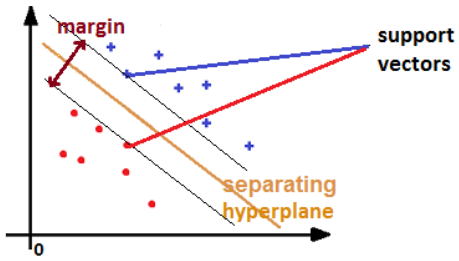



Figure 3.1.: Main terms in SVM notation

### SVM for linearly separable data

For a set of training data points $\vec{x}_i$ and corresponding class label $y_i \in \{-1, 1\}$, we want to classify the points using

$$f(\vec{x}) = \text{sign}(\vec{\omega}^T\vec{x} + b) \ , \tag{3.1}$$

that returns class labels.

As depicted above, we want to minimize the margin of the hyperplane to the support vectors. The functional margin for a training example $(\vec{x}_i, y_i)$ is $y_i(\vec{\omega}^T\vec{x}_i + b)$.

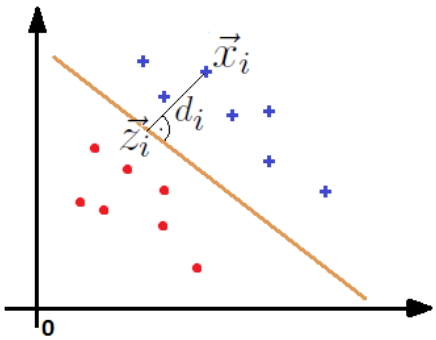



Figure 3.2.: Smallest distance $d_i$ of $x_i$ to hyperplane

The minimal distance of each point $x_i$ of the data set to the hyperplane, is the distance between $x_i$ and a point $z_i$ on the hyperplane closest to $x_i$. $\vec{\omega}/|\vec{\omega}|$ is the unit length vector, pointing in the same direction as $\vec{\omega}$ (which is orthogonal to the hyperplane). Therefore

$z_i = \vec{x_i} - y_i d_i \vec{\omega}/|\vec{\omega}|$, where $d_i$ is the distance between $x_i$ and $z_i$. We know that $z_i$ lies on the hyperplane, so it holds $\vec{\omega}^T \vec{z_i} + b = \vec{\omega}^T \left(\vec{x} - y_i d_i \vec{\omega}/|\vec{\omega}|\right) + b = 0$, and solving for $d$ yields

$$d_i = \frac{\vec{\omega}^T \vec{x_i} + b}{y_i |\vec{\omega}|} \ . \tag{3.2}$$

Now we want to find the hyperplane that maximises the functional margin of the data set. We add the constraint

$$y_i(\vec{\omega}^T \vec{x}_i + b) \geq 1 \ , \tag{3.3}$$

so that the problem is of a form that can be solved efficiently.

Because $\vec{x_i}$ is a support vector, $\vec{\omega}^T \vec{x_i} = 1$. Further, $y_i \in \{-1, 1\}$. It follows that the margin of the hyperplane (which is equal to twice the minimum distance of the support vectors to the hyperplane) is $2/|\vec{\omega}|$, which shall be maximised. This is equal to minimising $|\vec{\omega}|/2$. Hence, the optimisation problem can be denoted as:

$$\min(\frac{1}{2}|\vec{\omega}|) \ ,$$

s.t.

$$y_i(\vec{\omega}^T \vec{x}_i + b) \geq 1, \ i = 1...m \ . \tag{3.4}$$

### SVM for non-linearly separable data

Data is usually not linearly separable. We therefore allow for misclassification. For this purpose, a slack variable $\xi_i$ for each training point $\vec{x_i}$ is introduced. Via inclusion of the slack variable into our optimisation problem, we penalise misclassification. It enables a trade-off between misclassification and overfitting.

The minimisation term in (3.4) is adjusted to:

$$\min(\frac{1}{2}|\vec{\omega}| + C \sum_{i=1}^{m} \xi_i) \ ,$$

s.t.

$$y_i(\vec{\omega}^T \vec{x}_i + b) \geq 1 - \xi_i, \ i = 1...m, \ \xi_i \geq 0 \ , \tag{3.5}$$

where $C$ is a regularisation constant, which allows to control overfitting. The larger the $C$, the more the error is penalized.

### SVM multiclass separation

Several options for multiclass separation with SVM exist. A standard approach is to build as many SVMs as classes, and each SVM trains for one class versus the other

classes. This is often called one-versus-all (OVA) classification. The class is assigned according to the SVM with the highest function value, as defined in (3.1).

In the one-versus-one approach, SVMs for every pair of classes are trained. Each SVM classifies a data point. The point is assigned the class label, that results most frequently.

#### Support Vector Machines with Radial Basis Function Kernel

Often, non-linear separation yields better results than a linear separation. With the kernel function, the data is implicitly mapped into a high dimensional feature space in which linear separation can be applied. In the *implicit* mapping, the kernel function, which encodes the dot product in the high dimensional space, is simply evaluated on all pairs of data points (in low dimension).

The radial basis function kernel for two vectors $\vec{x}$ and $\vec{\omega}$ is given by

$$K(\vec{x}, \vec{\omega}) = exp(-\frac{||\vec{x} - \vec{\omega}||^2}{2\sigma^2}) \ . \tag{3.6}$$

The kernel takes its maximum value at the support vector, decaying uniformly around the support vector with a spread $\sigma$ to all directions.

#### Computational complexity

The computational complexity of SVMs on data sets with number of features >> number of samples, which is usually the case in biology, is $\mathcal{O}(m_{features} n_{samples})$ [13].

### 3.1.2 Elastic net

Ridge regression and lasso are highly applied methods in the context of extensive feature selection. However, they have certain limitations. Ridge penalty shrinks the coefficients of correlated predictors towards each other, while the lasso tends to pick one of them and discard the others. Ridge regression however, does not give sparse solutions.

The Elastic net regularisation is a mixture of the $\ell_1$ (lasso) and $\ell_2$ (ridge regression) penalties. For $m$ observation pairs $(\vec{x_i}, y_i)$, $y \in \mathbb{R}$, the elastic net solves the following problem:

$$\min_{\beta_0, \vec{\beta}} \frac{1}{2m} \sum_{i=1}^{m} (y_i - \beta_0 - \vec{x}^T \vec{\beta}) + \lambda \left[ (1-\alpha) \frac{1}{2} ||\vec{\beta}||_2^2 + \alpha ||\vec{\beta}||_1 \right] \ . \tag{3.7}$$

The tuning parameter $\lambda \geq 0$ weights the regularization term, hence controls the overall strength of the penalty. With increasing $\alpha$ from 0 to 1, for a fixed $\lambda$, the sparsity of the solution to equation (3.7) increases monotonically from 0 to the sparsity of the lasso solution [24].

For a classification problem where $y \notin \mathbb{R}$, a data point is assigned a class label via thresholds for the result of the model with the fitted coefficients applied to a new node.

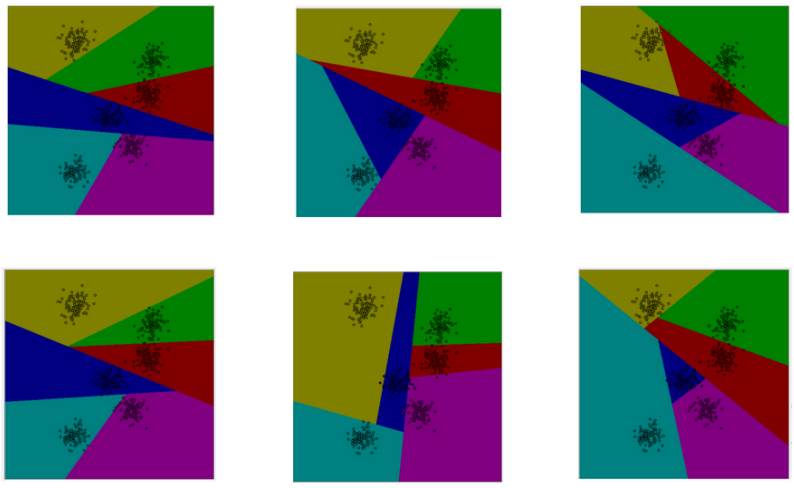


Figure 3.3.: Decision boundaries of six different decision trees [1]. The boundaries are very different in each tree, however, the ensemble of them yields good results.

#### Computational complexity

The computational complexity of elastic net with number of features >> number of samples is $\mathcal{O}(m^3_{features})$ [13].

### 3.1.3 Random forest

The random forest algorithm constructs a collection of decision trees built by several bootstrap samples from the training data. The idea is that decision trees, trained with random bootstrap samples of the data, can yield different decision boundaries, none of which is particularly good on its own (Figure 3.3). The ensemble of them however yields good results.

Each node in a tree examines the features values of the data points. The threshold for a horizontal or vertical features split is derived by testing for the features that optimally splits the data. However, in each level of the tree, only a selection of features are randomly chosen from all features as split candidates. Without this restriction different trees would use the same strong features (if there are strong features) in the beginning, which would lead to highly correlated trees.

The random forest decides on a class by evaluating all trees and choosing the class that was predicted the most.

#### Computational complexity

Computational complexity for building a complete unpruned decision tree is $\mathcal{O}(m_{features} n_{samples} log(n_{samples}))$. Building a random forest with *ntree* trees, the com-

[1] Source: https://sites.google.com/a/iupr.com/advanced-machine-learning/home/lectures/lecture-06—random-forests

plexity is $\mathcal{O}(ntree * mtry * n_{samples} log(n_{samples}))$, where $mtry$ is the number of variables sampled at each node.

## 3.2. Downsides of ML based biomarker detection

Machine learning approaches are often regarded as black boxes. The methods select on mathematical background only and hence do not give a (biological) interpretation for the weighting of features.

Further, for data sets with far more features, than samples, also referred to as $p>>n$ problem, the methods are likely to give poor performances. This phenomenon was first demonstrated for discrete classification by Hughes [34]. A reason is that different features have similar discriminatory power. Biological studies, however, are usually of exactly this kind: number of measurements per patient >> number of patients. Therefore, methods, that are better suited for this kind of data, are introduced in this chapter. Nevertheless, we find these arrangements do not solve the problem entirely. In addition, the computational time of the described methods is comparably large.

Another problem when applying ML methods for biomarker detection is that one has to be careful with conclusions based on classifier training: Results are dependent on the applied method. Several methods should be applied and results must be regarded carefully. The application in §6.2 exemplifies these issues.

Consequently it is of advantage to include further (biological) knowledge into the analysis, to obtain relevant biomarkers. We demonstrate this with a network approach in §4 and §5.

# 4. A new network approach for biomarker detection

Standard biomarker detection approaches, like machine learning, or correlation based approaches, investigate experimental measurements only. Studies applying these standard approaches are hence based on the assumption that the significant differential expression of a gene, or another biological measurement, between two conditions is sufficient enough to classify the gene to be pathogenic [18, 62, 87]. This assumption neglects the fact that impacts of individually altered genes can often be balanced, whereas numerous alterations of interacting proteins often lead to major consequences [7, 12].

Furthermore, the results in §3 motivate for an approach, that incorporates additional biological knowledge, because results are determined on a mathematical basis only, which is why we have to question their relevance in the investigated medical state. Networks are an option to incorporate additional biological knowledge: By a combination of biological network and measurements, the network approach can identify areas of connected entities with significant measurements, where the connection represents some biological process. We therefore choose to develop a new network approach. Remember that biomarkers are objectively measurable indicators of a medical state in a biological system. Nodes in the network should carry weights, representing objectively measurable data, or information derived from it. We assume that the node weights represent a measure of change between case and control.

We introduce, analyse, and apply a biased continuous time random walk based algorithm for the identification of metastable sets in a network.

## 4.1. The algorithm

With our approach we aim to analyse biological interaction networks, where edges are undirected and node weights are derived from biological data. The aim of our algorithm can be formulated as: For an undirected network with weighted nodes, the problem is to find connected nodes with similar and large weights. We denote such sets as modules.

In a biological context, a proportion of biological entities from the observed set, should be classified as biomakers. Hence, in our algorithm, we assume that the total number of nodes in modules is small compared with the size of the network.

## Motivation

A common approach to detect connected substructures with high node weights in a network is to construct a random walk, a process that a walker randomly takes from one vertex to one of its adjacent nodes with a probability proportional to the weight of the edge connecting them. Komurov et al. [40] introduce such an algorithm, *NetWalk*, based on biased discrete random walks, which is more likely to jump to adjacent nodes with higher values. In contrast, Sarich et al. [65] could show that *continuous* random walks are better suited for this purpose, because of their ability to detect modules in a network.

We assume that few regions in the network are relevant, which results in modules of comparably few nodes and a set of nodes which are not in modules.

We present an algorithm, that combines the above mentioned ideas by Komurov et al. and Sarich et al., to identify candidate genes based on biased continuous random walks.

## Background

### Biased random walks

Let $S = \{1, 2, ..., n\}$ denote the set of nodes and let $A = A_{ij} \in \{0, 1\}^{n \times n}$ be the adjacency matrix with zero diagonal. The node weights are given by $w \in \mathbb{R}^n_{\geq 0}$. The topological and the biased discrete time random walk transition matrices are respectively represented by

$$P_{T,ij} = \begin{cases} \frac{1}{|N_i|}, & A_{ij} = 1 \\ 0, & A_{ij} = 0 \end{cases}, \tag{4.1}$$

and

$$P_{W,ij} = \begin{cases} \frac{w_j}{\sum_{k \in N_i} w_k}, & A_{ij} = 1 \\ 0, & A_{ij} = 0 \end{cases}, \tag{4.2}$$

where $N_i = \{k | A_{ik} = 1\}$ is the set of neighbours to the node $i$. Assuming reversibility and irreducibility for the Markov chains, the values of the respective stationary distributions $\mu_W, \mu_T$ give the relative amounts of time, the random walk spent in each node in the long run. Komurov et al. [40] propose to detect the relevant structures as connected components of the set $\{\mu_W > \mu_T\}$.

### Continuous time random walk

The ratio $\mu_W/\mu_T$ is highly correlated with the weight vector $w$ (see equation (4.2)). In particular, if weights form two concentrated and well-separated groups, *NetWalk* finds exactly the high-valued nodes. However, if the high weights are not clearly separated from the low ones, the method struggles or even fails to find the desired parts. In preliminary studies we could show that a perturbation of node weights can easily corrupt the results by this algorithm.

Focusing on this issue, the impact of the network topology on the average time spent by the biased random walk in the single nodes is of high interest. We therefore switch to a continuous time random walk: The continuous time random walk allows to have different holding times for each node. Thus, the random walk can be adjusted, such that it remains longer in nodes with high weights, than in small node weights (see equations (4.3) and (4.4)).

An example of a topological continuous random walk with equal jumping probabilities for each node $i$ to all its neighbours $N_i$ and unit holding times, is:

$$L_{T,ij} = \begin{cases} \frac{A_{ij}}{|N_i|}, & i \neq j \\ -1, & i = j \end{cases} . \tag{4.3}$$

In contrast, the biased random walk has jump rates proportional to the node weights of the destination of the jump process:

$$L_{W,ij} = \begin{cases} A_{ij} w_j & i \neq j \\ -\sum_{k \in N_i} w_k, & i = j \end{cases} . \tag{4.4}$$

In the continuous time random walk, $\mu_T = \mathbf{1}/n$, where $\mathbf{1} = (1, ..., 1)^T$ and $\mu_W = {}^{w}/_{\sum_{k \in N_i} w_k}$. The ratio ${}^{\mu_W}/_{\mu_T}$ is now representing the relative node weights ${}^{w}/_{\bar{w}}$, with $\bar{w} = {}^{1}/_{n} \sum_{k \in N_i} w_k$, the mean node weight. Note that holding times of the random walk in each node, however, are now decisive control parameters in tuning the random walk such that it reveals structures of interest.

### Metastability of random walks

If there are at least two sets of nodes with high weights, without any direct connecting edge between the individual sets, meaning that transitions between the sets are rare events on the natural time scale of the random walk, then these sets are called *metastable*. Processes with this behaviour are called *metastable*. More precisely, a disjoint collection of sets $M_i, ..., M_m$ is called *metastable*, if the average time the random walk takes to enter one of the $M_i$ from outside $\bigcup_j M_j$, is much smaller than the time required to switch to some $M_j, j \neq i$, when currently being in $M_i$. $T = (\bigcup_j M_j)^c$, the complement of the metastable sets, is called *transition region*. By forcing the random walk to spend more time in regions with high node weights, it becomes metastable.

The mean first hitting time $\tau_B(x)$ of a metastable process is relevant for the detection of module *cores*, the central module nodes. It is defined as the expected first entry time of the process into the set $B$, conditional on starting in node $x$: $\tau_B(x) = E[T_B(x)]$.

### Identification of metastable structures

To find the metastable sets, recently developed theory on coarse graining of random walks, where metastable sets do not form a full partition [65, 66], i.e. $T \neq \emptyset$ , is applied. For this, committors are needed. The committor function $q_i : S \to [0; 1]$ is defined as the probability that a process starting in node $x$ visits the set $M_i$ first, rather than any

other set $M_j$, $j \neq i$. Metastable sets should have the property that reducing any $M_i$ to any of its nodes $x$ will result in the discarded nodes $M_i \setminus \{x\}$ having commitor values $q_i$ close to one. The metastable sets are detected based on the commitors (as explained below, in the steps of our algorithm).

## Concept

We apply the above mentioned theory to find modules of nodes with *high* and *similar* weights. For this purpose, a weight similarity measure $d(w_i, w_j)$ is introduced. We will use $d$ to construct a continuous time random walk, such that modules of interest form metastable sets, and then locate these sets.

### Node weight similarity

We can set the similarity measure $d$ to $d(w_i, w_j) = |w_i - w_j|$ if both node weights follow similar distributions and are equally relevant in the context of the analysis. We can define $d$ differently, if this is not the case. For the inclusion of node weight similarity, the *similarity matrix* $S \in \mathbb{R}^{n \times n}$ is defined as:

$$S_{ij} = \begin{cases} exp(-d(w_i, w_j) * p), & A_{ij} = 1 \\ 0, & otherwise \end{cases} , \tag{4.5}$$

where $p$ is a free similarity parameter.

The jump matrix $P_S$ is defined by $P_{S,ij} = S_{ij} / \sum_k S_{ik}$. The rate matrix for the *continuous time random walk with similarity* is defined as:

$$L_S = D_w^{-1}(P_s - I) , \tag{4.6}$$

where $D_w$ denotes the diagonal matrix with diagonal $w$. The Markov Jump process regulated by $L_S$ has expected holding times equal to one and jumping probabilities encoded in $P_s$.

### Steps

1. Initialization, step 1 (first two module cores): take a random $i^* \in S$, then set module core $c1 = argmax_{j \in S} \tau\{i^*\}(j)$. Set $c2 = argmax_{j \in S} \tau\{c1\}(j)$ and $C = \{c1, c2\}$.

2. Initialization, step 2 (first two clusters): compute the committor functions $q\{c1\}$ and $q\{c2\}$ for the two target sets $\{c1\}$ and $\{c2\}$, and identify the clusters $M_i$, $i = 1, 2$, as the indices of positive outliers in the data set $\big(q\{c_i\}(j)\big)_{j \in S}$.

3. Loop, step 1 (find the next module core): set the new, $k$-th core as the one "furthest" from all clusters, i.e. $c_k = argmax_{j \in S} min_{i=1,\ldots,|C|} \tau_{Mi}(j)$ and update the core sets $C \to C \cup \{c_k\}$

4. Loop, step 2 (assign cluster): compute the commitor functions $q_{M1}, ..., q_{Mk-1}$ and $q_{c_k}$ for the $k$ target sets $M_1, ...M_{k-1}$ and $\{c_k\}$, then identify the cluster $M_k$ as the indices of positive outliers in the data set $\big(q\{c_k\}(j)\big)_{j\in S}$

### Parameter $p$

The parameter $p$ in equation (4.5) has the following influence on the results: With increasing $p$, the similarity of weights of neighbour given more importance. If it is too large, the weights become insignificant.

### Parameter $\#m$

The application of the algorithm requires a specification of the number of modules that should be detected, which we denote as $\#m$.

Spectral clustering algorithms [76] use the spectral gaps of the random walk matrix as an indicator of the number of relevant clusters. Based on the theory of these algorithms $\#m$ is chosen as the number of eigenvalues for which a spectral gap appears. Thus, $\#m$ is determined argumentatively. In each run, a visualisation of the spectrum of $L_S$ (equation (4.6)) is provided (Figure 4.1 shows an example).

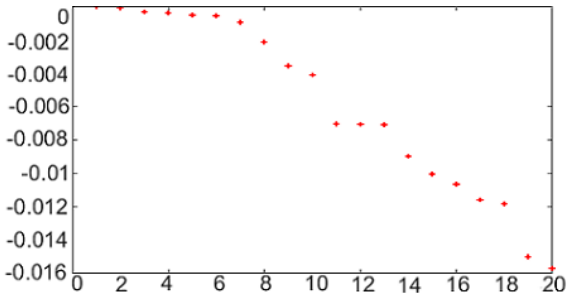



Figure 4.1.: Spectrum of the first 20 eigenvalues of $L_S$ for an example network. Significant spectral gaps show up for eigenvalues 10, 13, and 18. These values can be used as candidates for $\#m$.

### Implementation

The algorithm is implemented in MATLAB. The code is attached in the appendix.

### Numerical costs

Since both commitors and expected hitting times are characterized by a system of linear equations, both of these objects can be computed essentially in $\mathcal{O}(n)$ time by utilizing iterative solvers, given the defining matrices are sparse. Our random walks are reversible, which is equivalent to the self-adjointness of the rate matrix with respect to a specific inner product (this requires the knowledge or computation of the stationary distribution of the process). This enables the usage of specific solvers, which has advantages in accuracy and runtime.

## 4.2. Postprocessing

We apply our algorithm to artificial and biological data (see §4.3 and 6.3). Three artefacts occur in the results:

1. Some modules have rather small node weights.
2. Some modules are adjacent, hence should be jointed.
3. Some modules are very small.

However, it can be understood why these artefacts occur, and we develop a postprocessing, that detects and removes them.

### Modules with small node weights

If a set of connected nodes is very similar in terms of weight, it can be attractive for the algorithm, although its nodes have small weights. These modules contradict with the aim to identify modules of high weight, but they can easily be identified and, hence, do not create problems. We define a sufficiently high weighted module, such that the mean weight of the module is larger or equal to the 90th percentile of all node weights. The 90th percentile is often applied as a threshold [29, 47, 2]. In the postprocessing all modules that do not fulfil this criterion are excluded.

### Neighbouring modules

Based on the values of the commitor function of each identified module core, each module is extended with further nodes. It can happen that two *cores* are extended, such that the resulting modules are adjacent. These modules can be jointed.

For each pair of modules, all the shortest paths between all nodes of the one module, to all nodes of the other module, are computed. If any of these shortest paths is 1, the modules are jointed.

### Modules of small size

If a core node has a high weight, but all its neighbours have small weights, the module is not extended and remains a single node module.

Our working hypothesis is that the impact of single gene changes, is comparatively small. We are therefore interested in bigger modules. The minimum number of genes in a module considered in studies for network motif detection is usually three [5, 6, 81]. This values is adopted for our approach: If modules of size $\leq 2$ are still present after the foregone postprocessing step, these modules are excluded.

## 4.3. Simulation studies

We test our algorithm using artificial networks. This enables us to assess whether the algorithm detects the interesting modules in a network. This is not possible with a real data network, because we do not know where the interesting modules in the network are.

We create artificial networks of 5000 nodes, with predefined modules of different kind and compare the detected modules with the modules that have been inserted. We decide for a network size of 5000 nodes, because this represents well a dimension of a medium size biological network. The artificial network has seven topological and hairball modules. Details on the modules are given in the following sections.

Real world networks are usually almost scale free [3]. Artificial networks are therefore created with the Barabási–Albert model [4]. The algorithm starts with a network of $m_0$ nodes. The network is evolved by adding one node at a time. Each new node is connected to m$<m_0$ nodes. The probability for a link from the new node to an existing node $i$ is: $P(k_i) = {}^{k_i}/_{\sum_j k_j}$, where $k_i$ is the degree of node $i$ and $j$ the indices of nodes already in the network. Each node has at least one link. Accordingly, nodes with comparably many links, are more likely to be chosen as neighbour nodes, while nodes with few links are unlikely to attract new nodes in this setting.

The node weight distribution of nodes, that are not in modules, are designed similar to the biological data from §6.3. The weight distribution follows a gamma distribution with shape 1.7 and scale 0.07 (Figure 4.2).

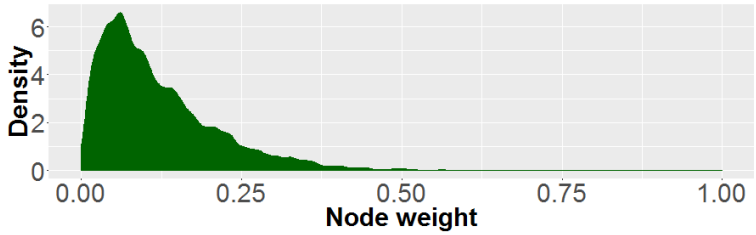



Figure 4.2.: Distribution of node weights in the artificial network. Few nodes that are not in modules have weights larger than 0.3. The maximum node weight of nodes that are not in modules is approximately 0.7.

*Predefined modules* of 30 nodes each are added. We choose each cluster to be of size 30, because in our experiments with biological data we observe that modules are often of similar size. A module seed node is chosen randomly and its weight is set high. The module is extended by randomly choosing neighbours of the module node and adjusting their weights. The weights for the *predefined module* nodes are chosen from a predefined normal distribution. We perform two simulation studies and evaluate their results:

1. For the clustering of networks with *predefined modules* with **similar weight distributions**, we **vary algorithm parameters** $\#m$ and $p$ (see §4.3.1).
2. For **fixed parameters** derived from the first study, we cluster networks with *predefined modules* with **varying weight distributions** (see §4.3.2).

When studying the performance of the algorithm for artificial networks, a score for the evaluation, based on the agreement of *predefined* and detected modules, is required. It is difficult to compare modules in a network with a large transition region - as in our approach - because the agreement of module IDs is always high: Assume a module $m_1$ in partition $P1$ and a partition $P2$. In partition $P2$ the majority of nodes from $m1$ are assigned to the transition region. Since most of the nodes from $m_1$ are in one module in $P2$ (the transition region), this module is evaluated as highly agreeing, in terms of similar module separation.

We therefore choose to evaluate modules by the percentage of detected nodes from the *predefined modules*, denoted as *PDN*. The percentage overcomes the above mentioned problem. However, it does not include a validation of module separation, nor an assessment of nodes that are additionally identified as modules nodes.

It is not known if any modules of interest occur by random. This can happen, because the background includes high weights. If the respective nodes are connected, they should be detected as a module. Therefore, the occurrence of new module nodes should not downgrade the assessment of results, which is not the case with the described percentage evaluation.

### 4.3.1 Varying algorithm parameters $\#m$ and $p$

The application of the algorithm requires a specification of the two parameters $p$ and $\#m$. The evaluation of results for artificial networks allows for the study of the influence of parameters $p$, which influences the similarity criterion, and $\#m$, the desired number of modules, and postprocessing.

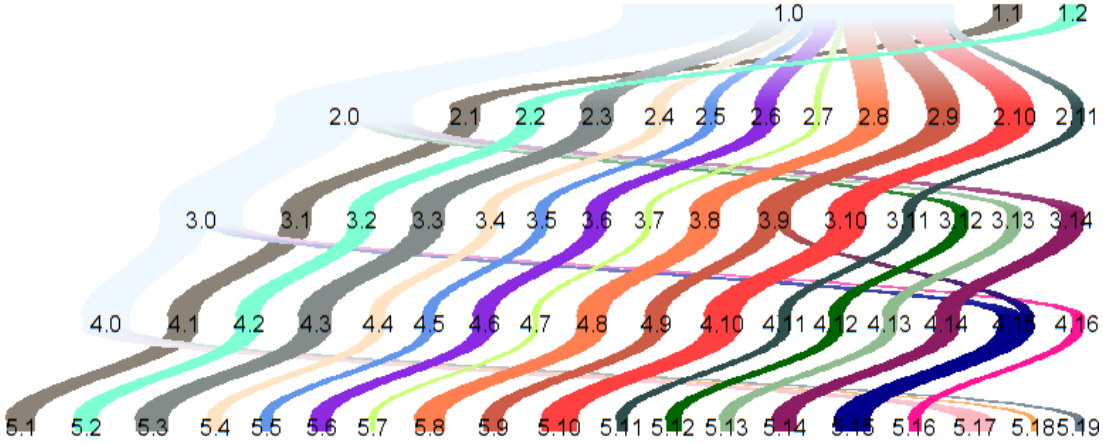
In this section, the modules are created such that their node weights follow a normal distribution with mean 0.5 and variance 0.05, because we often observer similar distributions in modules from biological data. The maximum of the node weight distribution of nodes not in modules is approximately 0.7 and only few nodes have similar weight. Hence, 0.5 is a comparably high value, which our algorithm is designed to find.

#### Parameter $\#m$

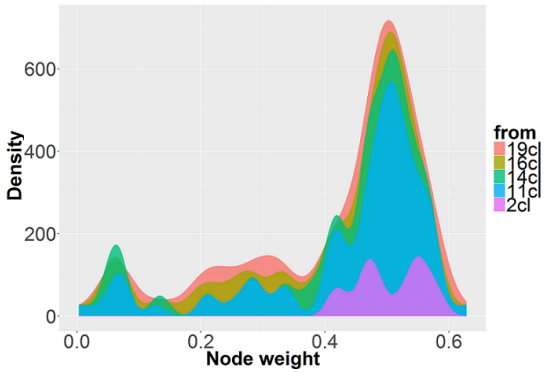
In this section, the values of $\#m$ are determined via the spectral gaps of the spectrum of the generator matrix $L_S$ (equation (4.6)).

In our studies we find, that in almost all cases, for increasing $\#m$, the PDN as well as the number of module nodes in general increases. Generally, for increasing $\#m$, we observe:

- Some modules split into several modules, or parts of them combine with other modules.
- Nodes from the transition region form new modules, or they are integrated into modules that are found for smaller values of $\#m$.
- The majority of nodes that are in a module for smaller values of $\#m$, appear in modules for larger values of $\#m$.




(a) Development of the modules for five different values of $\#m$. Each module state has a prefix (1., 2., 3., 4., 5.*cluster number*), representing the clustering iteration. For larger values of $\#m$, the number of nodes in modules increases, thus some nodes which are in the transition region in a module for a smaller value of $\#m$, are in a module for a larger value of $\#m$. These nodes are denoted as the *.0* module.




(b) Density of weights of nodes in all detected module per value of $\#m$. For higher values of $\#m$, more high node weights and some small weight nodes are in the resulting module set.

Figure 4.3.: Clustering of one artificial network for $p = 10$ and $\#m = 2, 11, 14, 16, 19$

Figure 4.3a shows the development of modules for five different values of $\#m$, determined from spectral gaps, for one artificial network. For larger values of $\#m$, the number of nodes in modules increases. The initial modules are almost always adopted for increasing $\#m$. More modules of similar size are returned for larger values of $\#m$. The module 3.9 splits into two modules, one of which includes additional nodes.

Figure 4.3b reveals, that for this network and algorithm setting, more high weight nodes are identified for larger values of $\#m$. A small amount of low weight nodes are also added, however. We observe this behaviour for the majority of the tested networks.

Figure 4.4 shows the PDN for $p = 10$ and $\#m = 1, ..., 20$ for five different networks. In general for increasing $\#m$, more *predefined module* nodes are detected. The curve flattens for large values of $\#m$.

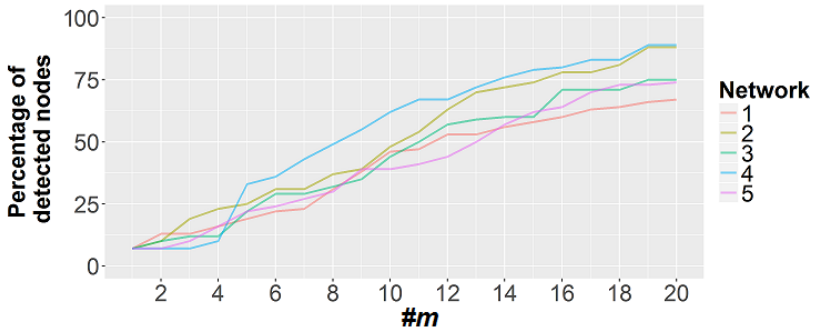



Figure 4.4.: PDN for five artificial networks for $\#m = 1, ..., 20$, for $p = 10$. The PDN increases for increasing values of $\#m$ (more drastically for smaller values).

### Parameter $p$

Results for $p < 10$ include significantly more small weight nodes, compared to results for larger $p$. Hence, $p = 10$ is chosen as the minimum $p$. For $p > 25$, $L_S$ (equation (4.6)) becomes numerically unstable. Reducing the tolerance does not remove all conflicts. Therefore, $p = 10$ and $p = 25$ are tested in this simulation study.

In Figure 4.5a we can see, that the number of nodes from clusterings with different values of $p$ and the same $\#m$ is similar for $\#m \geq 10$: They differ by no more than 5%. For 18 of 20 values of $\#m$, the mean PDN is similar or higher for $p = 25$ than for $p = 10$.

### Postprocessing

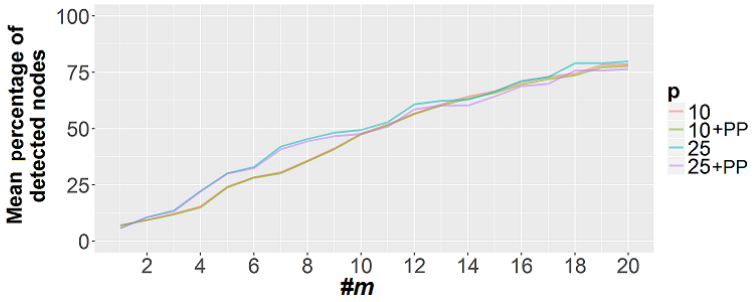
Figure 4.5a reveals that the postprocessing removes less than 10% of the module nodes belonging to *predefined modules* for any values of $\#m$.

However, the number of detected modules differs significantly: For initially $2, 11, 14, 16, 19$ modules in the clustering illustrated in Figure 4.3, the postprocessing returns the same 3 modules in all cases. We observe this development for three of the five tested networks, and for one network highly similar modules are returned after the postprocessing for varying numbers of $\#m$. This indicates that if the postprocessing is included, the initially fixed parameters $p$ and $\#m$ no longer change the results significantly.

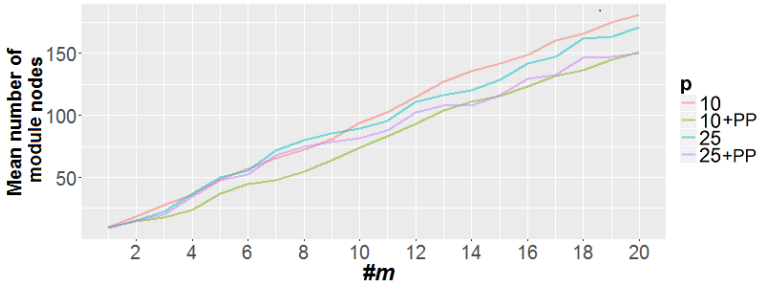
### Number of nodes in modules

It must be noted that it more module nodes do not necessary indicate an improved result. In sparse approaches, the aim is to identify few, mainly relevant entities. In biomarker detection, this leads to biomarkers with high interpretability. Further, medical testing for few biomarkers is simpler and cheaper.

Figure 4.5b shows that for larger values of $\#m$, the number of module nodes increases almost linearly. We have observed that the curve of the PDN flattens for large values of $\#m$ (Figure 4.4). Bearing in mind the idea of sparseness, a compromise regarding $\#m$




(a) The PDN changes minimally after postprocessing. Only for $p = 25$ and values of $\#m \geq 10$ the PDN changes by more than 3%.




(b) Total number of module nodes: Significant changes after postprocessing. In the postprocessing for $p = 10$, more nodes are removed in the postprocessing than in the postprocessing for $p = 25$

Figure 4.5.: Mean results for five different networks for $\#m = 1, ...20$, for $p = 10, 25$ before and after postprocessing (PP)

might be ideal: For larger values of $\#m$, more module nodes of *predefined modules* are detected. However, for higher values of $\#m$, the number of module nodes which do not belong to any of the *predefined modules* increases even more.

### 4.3.2 Varying weight distributions of predefined modules

In our first simulation study, evaluations are performed on networks with modules having similar node weight distribution. It is of interest how the results change when these distributions are altered.

The distribution of weights of nodes that are not in modules is left unchanged. The parameter $\mu$ of the normal distribution of the *predefined modules* is altered between $\mu = 0.1, ..., 1$. The variance is fixed at $\sigma^2 = 0.1$. From results in §4.3, it follows that $p = 25$ is slightly better than $p = 10$ and that $\#m$ should be chosen relatively large. Results in this section are therefore computed for $p = 25$ and $\#m = 16$. For each $\mu$, five networks are computed and clustered with our algorithm.

Figure 4.6 shows mean and standard deviation of the PDN. For $\mu \geq 0.5$, the PDN are satisfying. For $\mu \geq 0.6$, the PDN remain almost unchanged. The PDN being small for small values of $\mu$ does not mean that the algorithm preforms poor in that case, because the algorithm is designed to detect modules with high weights. The background

distribution has its maximum at roughly 0.7. If similar high weight nodes are adjacent by chance, they form more interesting modules for our purpose, compared to the *predefined modules*.

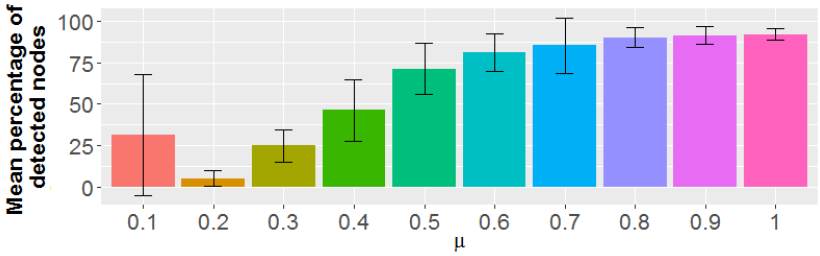
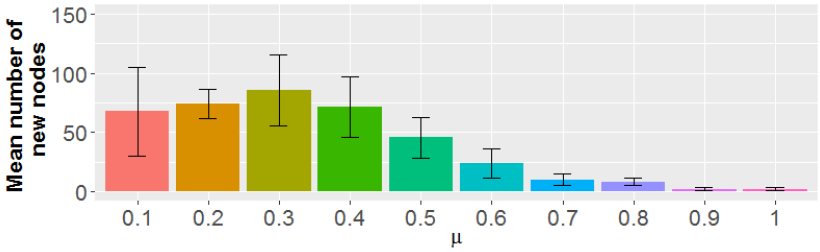
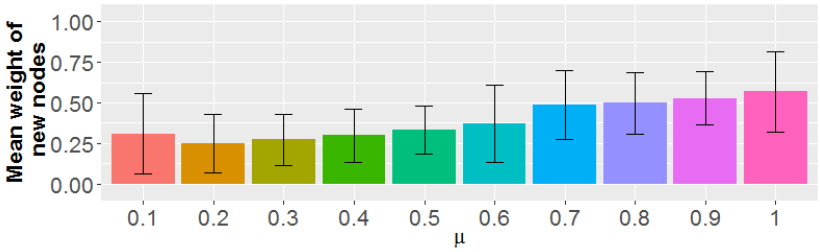



Figure 4.6.: PDN (respective mean and standard deviation) of clustering results with weights in *predefined modules* $\sim N(\mu, 0.05)$, $\mu = 0.1, ..., 1$. For $\mu \geq 0.6$, the PND remains almost unchanged.

Figures 4.7a and 4.7b show the mean number, respectively weight, of nodes that are assigned to a module, but are not in the *predefined modules*. For increasing $\mu \geq 0.4$, the number of these nodes decreases. We can assume that with increasing $\mu$, more significant modules form in the network. Further, for small values of $\mu$, nodes in *predefined modules* have small weights compared to the other nodes, which is why few of the *predefined module* nodes are detected.

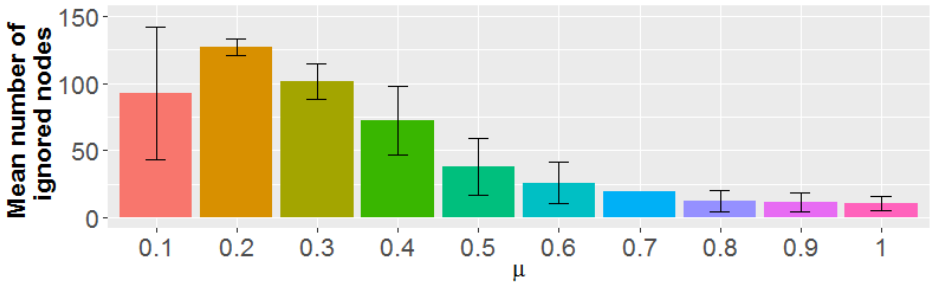



(a) Number of returned module nodes not in *predefined modules*

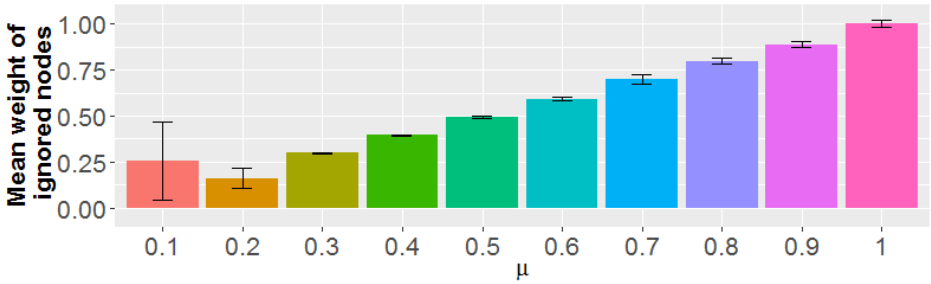



(b) Mean weight of returned module nodes not in *predefined modules*

Figure 4.7.: Evaluation (respective mean and standard deviation) of module nodes from our algorithm that are not in *predefined modules* with weights in *predefined modules* $\sim N(\mu, 0.05)$, $\mu = 0.1, ..., 1$. For increasing $\mu \geq 0.4$, the number of these nodes decreases.




(a) Mean number of nodes from *predefined modules* that are not in detected modules from our algorithm. The number of these nodes decreases for increasing $\mu \geq 0.2$.



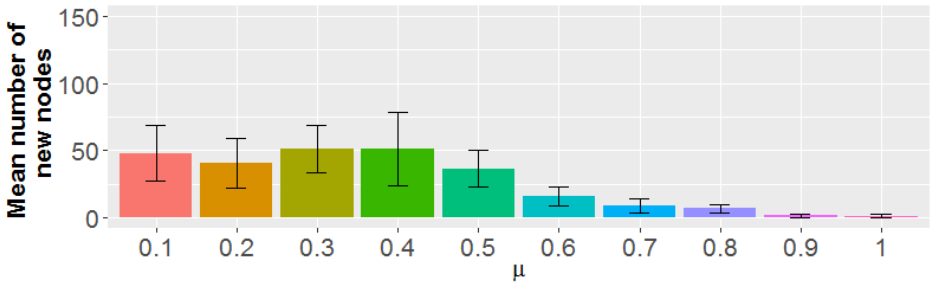

(b) Mean weight of nodes from *predefined modules* that are not in detected modules from our algorithm. The mean weight of the ignored nodes increases for increasing $\mu \geq 0.2$.

Figure 4.8.: Evaluation (respective mean and standard deviation) of nodes from *predefined modules* that are not in modules of network clustering with weights in *predefined modules* $\sim N(\mu, 0.05)$, $\mu = 0.1, ..., 1$

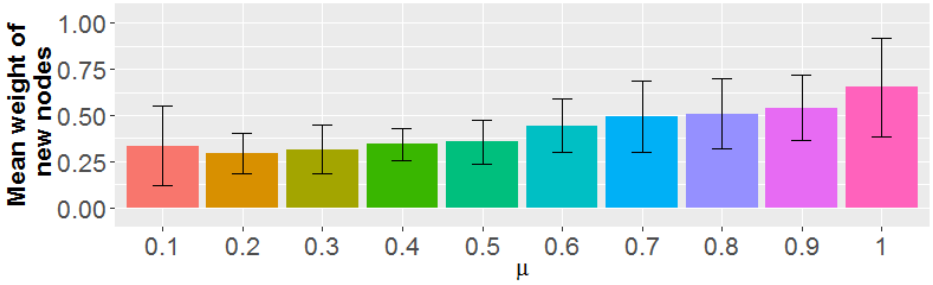
Figures 4.8a and 4.8b show the mean number, respectively weight, of nodes from *predefined modules* not in detected modules by our algorithm. The number of these nodes decreases for increasing $\mu \geq 0.2$. It can happen that the weights of two adjacent nodes from *predefined modules* are at the opposite ends of the module weight distribution. Hence, the similarity for these adjacent nodes would no longer be satisfied and the node is excluded from the module. The mean weight of the ignored nodes increases, which makes sense, since the weight of nodes in *predefined modules* increases.

Changes due to postprocessing are mainly visible for the module nodes not in *predefined modules* (Figure 4.9, compare Figure 4.7). The number of these nodes decreases significantly and their mean weight increases. This proves for a successful postprocessing.

We could also investigate, how far the PDN alters, when $\sigma^2$ of the normal distribution of the *predefined modules* is altered. However, we can see that $\sigma^2 = 0.1$ produces satisfying results for a sufficiently large value of $\mu$. Increasing $\sigma^2$ in the node weight distribution contradicts with the objective of finding modules of similar weight. We are therefore satisfied with the previous observations within the scope of this thesis.




(a) Number of module nodes not in *predefined modules*




(b) Mean weight of module nodes not in *predefined modules*

Figure 4.9.: Evaluation (respective mean and standard deviation) of module nodes from our algorithm not in *predefined modules* of network clustering after post-processing with weights in *predefined modules* $\sim N(\mu, 0.05)$, $\mu = 0.1, ..., 1$. The number of these nodes decreases significantly and their mean weight increases. Compare Figure 4.7.

# 5. A new fusion network approach for biomarker detection

Due to the progress of high throughput technologies in recent years, an abundance of omics data from a variety of biological data types is present today. This data has enabled the creation of a multitude of databases that significantly facilitate biomedical research. Integration of these sources allows a comprehensive analysis of molecular and clinical datasets. Thus, it can improve molecular based insights to diseases.

As previously discussed, applying networks into a biological study improves results in another aspect. Approaches for biomarker detection using PPI networks, often evaluate topological features of weighted PPI networks without additional data, where weights represent e.g. confidence of interactions. Other approaches evaluate topological features of weighted PPI networks, where weights are derived by the integration of one data source [45, 83, 85]. These approaches do not make use of data fusion. Thereby, specific results might be over interpreted or missed [27]. Equally important is that fusion approaches enable us to capture more dimensions of complex biological processes.

An example for a network analysis algorithm performing data fusion for identification of candidate genes in a PPI network is given by Xia et al. [84]. They define a selection of 'seed proteins', around which a network is built, as the first main task. This is a method often applied, when it comes to big data analysis [40], as for example in [71]. Indeed, simplifies the analysis, but it adds a strong bias to the problem [20] and it contradicts with the idea of capturing biological processes more realistically.

An algorithm for network based biomarker detection, going beyond simple one dimensional approaches and not making use of initial data restriction, is a progress compared to these methods. In this chapter, we apply the ideas of our algorithm from §4 and adjust it to a fusion network algorithm.

## 5.1. Algorithm adjustments towards a fusion approach

The algorithm is adjusted such that it detects structures based on multi-dimensional node weight similarity. From a weight similarity measure, a continuous time random walk is constructed such that the modules of interest form metastable sets of the walker.

While the holding times in the single source approach are computed on the basis of each node weight, in the fusion approach holding times have to be based on a vector of weights. Further, for the computation of the distance of nodes, a distance measure for vectors is required.

To get an idea of a good choice of holding times, as well as for the distance measure, several settings are evaluated on artificial networks in §5.3. When each data (weight dimension) follows similar distribution, standard distance measures, such as Manhattan or Euclidean distance, can be used. For a reasonable choice, the data has to be observed.

The general aim of the algorithm is to find modules of connected nodes with high weights. In the one dimensional case modules could be assessed immediately by examining whether the nodes are connected and by observing the distribution of node weights. In the fusion network approach, however, the assessment of modules becomes more complex. In an analysis we find that the algorithm returns different kinds of modules. In one kind, all weights are high, in the other, only one weight is high. Both modules are interesting. Modules with one high node weight are also interesting, as they are a region of connected nodes with a significant characteristic in one dimension.

Nevertheless, in our fusion approach we aim to find nodes, which show interesting characteristics in all dimensions. We therefore design the random walker to leave nodes with small weights in any dimension quickly. Hence, for modules with one high weight dimension, the other weight dimensions must be slightly increased, for the module to be detected. We decide for $exp(min(nodeweight))$, for the holding times.

## 5.2. Postprocessing

We apply our algorithm to artificial and biological data (see §5.3 and §6.5). We observe similar artefacts as in the single source approach (compare §4.2). The postprocessing needs to be adjusted for fusion data, because each module has two weight distributions that need to be evaluated. In the single source approach, we correct for:

1. Modules with small node weights;
2. Modules that are adjacent;
3. Modules of size $\leq 2$.

Similar to the single source postproceesing, in the fusion postprocessing, we correcting for modules with small weight, such that modules are removed if the mean of the weight distribution of *each dimension*, is smaller than the 90th quantile of the overall network distribution.

Modules are jointed if the shortest path for any node pair from the respective modules is at maximum 1. However, modules are only combined, if their node weight distributions are similar, according to the following criterion: A two sided Kolmogorow-Smirnow (KS) test between the weights of nodes of the respective modules, for each weight dimensions, is performed in R with KS.TEST. The KS test determines if two samples differ significantly. We choose the KS test, because it makes no assumption about the distribution of data. If the p-value is larger than 0.05 for all weight dimensions, the modules are combined. The threshold is chosen comparably small, because we define it to be sufficient for two modules to be combined, if distributions are relatively similar.

Modules of size one are combined with another module if the mean weight of the singleton module and the other module differ no more than the variance of the background distribution from each other.

Similar to the postprocessing for single source networks, remaining modules of size two or one are removed.

## 5.3. Simulation studies

In this section we analyse how to adjust the algorithm for a fusion approach, such that it return modules of requested kind. To enable a reasonable evaluation of results, we analyse artificial networks. This enables us to tell, whether the algorithm detects the interesting modules in a network.

We create artificial networks, with predefined modules of different kind and compare the detected modules with the modules that have been inserted. Real world networks are usually almost scale free. Artificial networks are therefore created with the Barabási–Albert model [4], for details see §4.3. Each node is assigned to weights. The node weight distributions of those nodes that are not in modules, are designed similar to §4.3. Both distributions follow a gamma distribution with shape 1.7 and scale 0.07 (Figure 5.1).

Clusterings are evaluated by the PND, the percentage of detected nodes the *predefined modules* defined in §4.3. We perform two simulation studies on the artificial networks and evaluate their results:

1. For the clustering of networks with *predefined modules* with **similar weight distributions**, we **vary algorithm setting and parameters** (see §5.3.1).
2. For a **fixed algorithm setting and parameters** derived from the first study, we cluster networks with *predefined modules* with **varying weight distributions** (see §5.3.2).

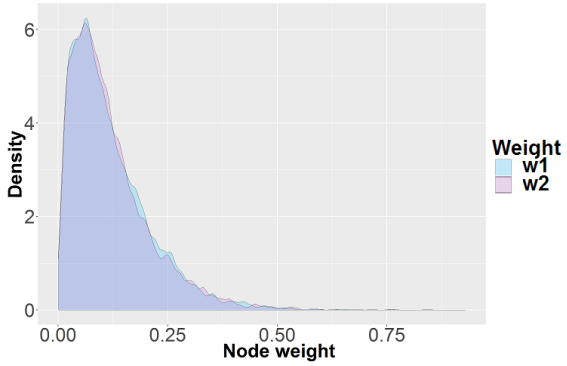



Figure 5.1.: Distribution of node weights ($w1$,$w2$) in the artificial network for nodes not in predefined modules. The maximum of node weights is approximately 0.7, but only few nodes have such high weights.

### 5.3.1 Varying algorithm setting and parameters

In this section, we analyse artificial for which the node weights in the seven modules follow a normal distribution with mean 0.5 and variance 0.05. Details on how these modules are derived are given in the introduction of §4.3. For two of seven modules, only one weight is high. The other modules are high in both weights.

Firstly, the algorithmic settings are evaluated. We evaluate combinations of different choices for:

1. Distance $d$ between two nodes: $d(node_i, node_j)$
2. Exponent $x$ in holding times: $exp(min(nodeweight)^x)$

Table 5.1.: PDN for different algorithm settings (node distance $d$ and exponent $x$ in holding times $x$) and parameters (similarity parameter$p$ and number of modules $\#m$), for module node weights $\sim N(0.5, 0.05)$; PDN$>$ 0.7 marked blue.

| $d$ | $x$ | $p$ | $\#m$ | $PDN$ |
|---|---|---|---|---|
| Euclidean | 1 | 10 | 15 | 0.61 |
| | | | 19 | **0.83** |
| | | 25 | 14 | 0.59 |
| | | | 18 | **0.83** |
| | | | 19 | **0.9** |
| | | 50 | 17 | **0.81** |
| | | | 18 | **0.83** |
| | | | 19 | **0.83** |
| Euclidean | 2 | 10 | 8 | 0.4 |
| | | | 15 | 0.61 |
| | | | 16 | 0.64 |
| | | | 19 | **0.83** |
| | | 25 | 14 | 0.59 |
| | | | 18 | **0.72** |
| | | | 19 | **0.9** |
| Manhattan | 1 | 10 | 8 | 0.24 |
| | | | 12 | 0.54 |
| | | | 14 | 0.64 |
| | | | 16 | **0.73** |
| | | | 19 | **0.88** |
| | | 25 | 18 | **0.72** |
| Manhattan | 2 | 10 | 8 | 0.24 |
| | | | 13 | 0.61 |
| | | | 15 | 0.66 |
| | | | 18 | 0.69 |
| | | 25 | 19 | **0.9** |
| | | 50 | 20 | **0.83** |

Table 5.1 shows evaluations for different algorithmic settings, for different $p$ and $\#m$, for a network with 5000 nodes and seven modules. The modules, that have been inserted artificially, are further referred to as *predefined modules*. Values for $\#m$ are determined from the spectrum of $L_S$ (equation (4.6)).

We can use simple distance measures, like Euclidean or Manhattan distance, for $d$, since both the node weights come from similar distributions. A difference between Euclidean and Manhattan distance is, that Manhattan distance sums up each dimension's difference, whereas Euclidean distance averages the distances.

Results for $p < 10$ include significantly more small weight nodes, compared to results for larger $p$. Hence, $p = 10$ is chosen as the minimum $p$. For $p$ significantly larger than 50, $L_S$ (equation (4.6)) becomes numerically unstable. The maximum $p$ is therefore set to 50. $p = 25$ is tested as an intermediate $p$. In Table 5.1 PDN are marked blue, if they are 70 or higher. For two settings for $p = 50$ $L_S$ (equation (4.6)) becomes numerically unstable.

For a sufficiently large $\#m$, all but one setting yield PDN of $> 70\%$. This is an important finding. Results in §4.3 indicate a tendency of more nodes in the resulting modules for increasing $\#m$. These nodes include nodes from *predefined modules*. The same holds for the fusion network approach for both distance measures.

The number of modules, however, is more than twice as high as 7 (which is the number of *predefined modules*) for results with PDN $> 70\%$. This is enhanced in the postprocessing. We fix the difference measure to the Euclidean distance. $x$ is set to 1 since results are amongst the best and interpretation is easier than for $x = 2$. Further, we fix $p = 25$, because it yields best results in this setting.

How to find the best $\#m$ remains an open question. The PDN is usually better for larger $\#m$ up to a certain value of $\#m$, because the modules for larger values of $\#m$ include the nodes of modules for smaller values of $\#m$. Often, the best result is achieved at the biggest spectral gap. Figure 5.2 shows results for clusterings of 10 different artificial networks, that are computed with the same settings ($p = 25$, Euclidean distance). $\#m$ is set to the following high values:

- The number of the eigenvalue, when sorted, for the biggest spectral gap of $L_S$ (4.6) (which we observer is always $< 20$).
- 19, because this yielded the best result in Table 5.1 for the selected setting.
- 20, to see if results get considerably better for large values.

In most of the cases, the results for the three values of $\#m$ do not vary significantly. We conclude that setting $\#m$ to a large number for a big spectral gap is recommendable.

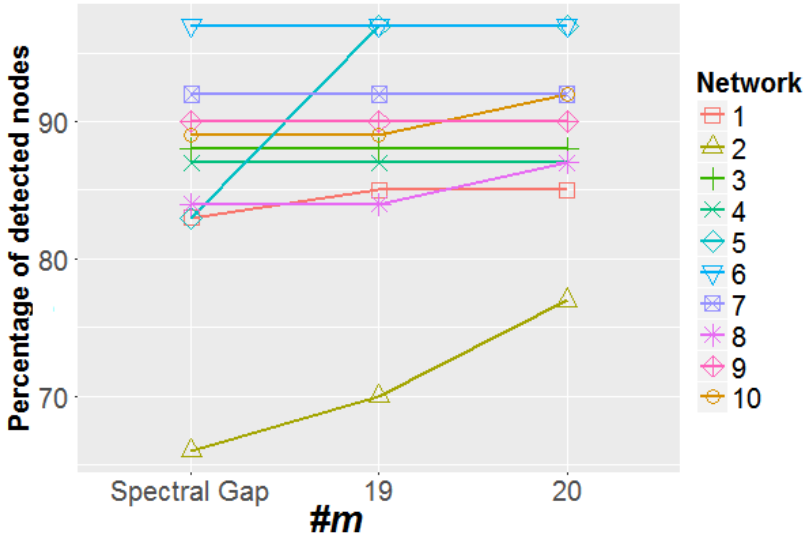


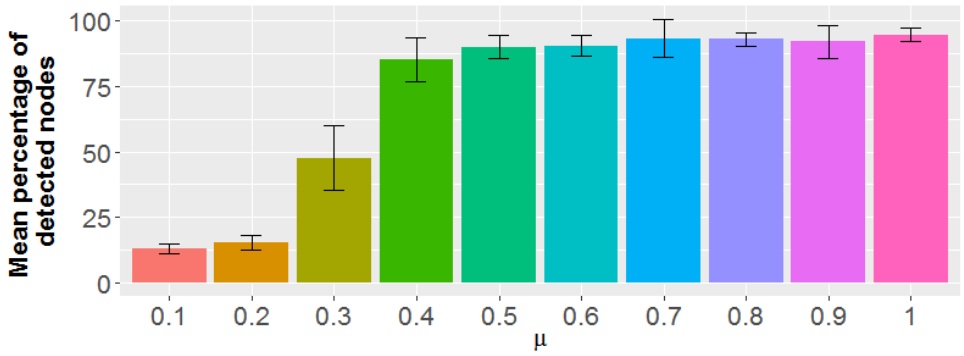

Figure 5.2.: PDN for 10 artificial networks for different values of $\#m$; $d$ is set to Euclidean distance, exponent $x$ in holding times is 1, and $p = 25$. In most of the cases, the results for the three values of $\#m$ do not vary significantly.

### 5.3.2 Varying weight distributions of predefined modules

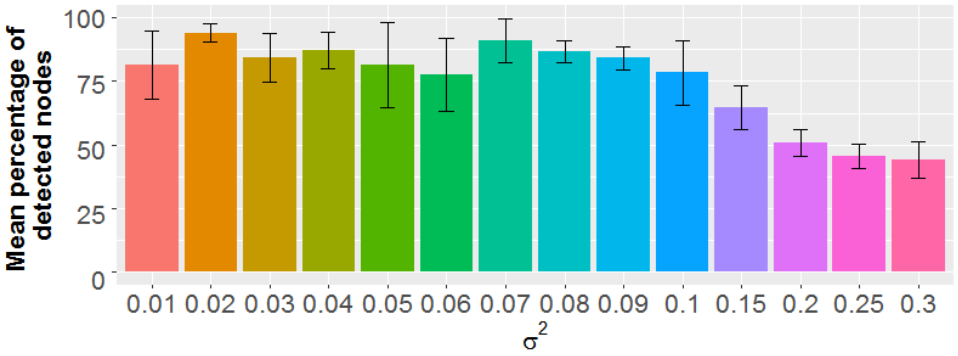
In §5.3.1 the optimal parameters for the algorithm to analyse networks with similar node weight distributions are determined. In this section we evaluate clustering results for networks with modules that have different weight distributions. The first simulation study reveals, that results for $\#m = 19$ are consistently good: The PDN is on average 0.897 with variance across different networks 0.006. This is a satisfying result. We therefore fix $\#m = 19$ for the following analysis in this section.

The background distribution is unchanged (see §5.3.1). In a first analysis, the parameter $\mu$ of the normal distribution of the module node weights is altered between $\mu = 0.1, ..., 1$. The variance is fixed at $\sigma^2 = 0.1$. For each $\mu$, five networks are computed, over which we evaluate. Figure 5.3a shows the mean and standard deviation of the PDN. For $\mu \geq 0.4$, the PDNs are satisfying and do not change considerable any longer.

The same test setting is performed for networks for module node weight distributions with $\mu = 0.5$ and $\sigma^2 = 0.01, ..., 0.1, 0.15, 0.2, 0.25, 0.3$. Results are shown in Figure 5.3b. For modules with node weight distributions with $\sigma \leq 0.1$ the PDN is similar and high, compared to higher values of $\sigma^2$. That the PDN is small for higher values of $\sigma^2$ can be exploited by the fact that the algorithm detects modules with *similar* weights. It has to be assumed that this is no longer given, for network modules with a wide node weight distribution.




(a) $\mu = 0.1, ..., 1$. For $\mu \geq 0.4$, the PDNs are satisfying and do not change considerable any longer




(b) $\sigma^2 = 0.01, ..., 0.1, 0.15, 0.2, 0.25, 0.3$. For modules with node weight distributions with $\sigma \leq 0.1$ the PDN is similar and high, compared to higher values of $\sigma^2$.

Figure 5.3.: Mean and standard deviation of the PDN for node weight distribution from normal distributions $N(\mu, \sigma^2)$ with varying values for $\mu$ (a) or $\sigma^2$ (b)

### 5.3.3 Influence of postprocessing

The PDN are nearly equal before and after postprocessing (Figure 5.4), except for small $\mu$ for which the PDN decreases (compare Figure 5.3a). Indeed the *predefined modules* are not of interest due to their small mean weight and therefore the exclusion of these nodes from detected modules is desirable.

Further, the number of module nodes returned by the algorithm that are not in *predefined modules* decreases (Figure 5.5). As already mentioned, we do not know if any interesting structures appear by chance when the networks are created. Figure 5.6 reveals that the weights in these nodes (precisely, the maximum of the weight vector of each new node) are high in general and that their mean weight is increased after postprocessing.

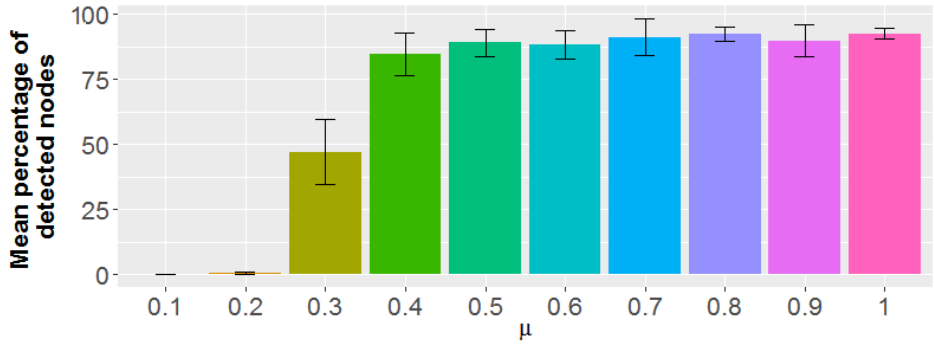
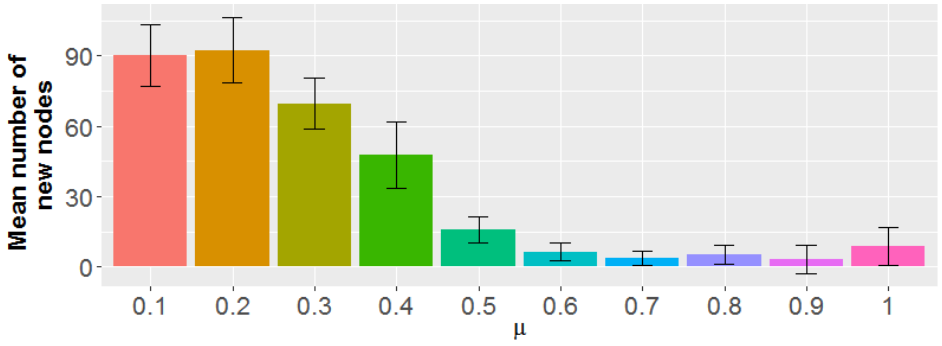
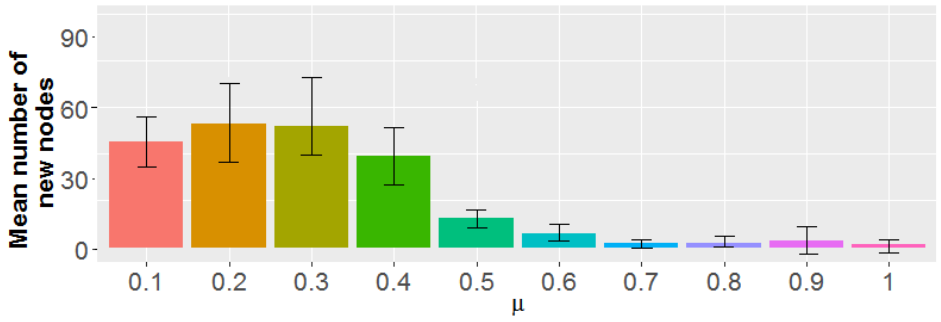



Figure 5.4.: Mean and standard deviation of the PDN after postprocessing, for weights of *predefined modules* $\sim N(\mu, 0.05)$, $\mu = 0.1, ..., 1$

For networks with fixed $\mu$ and altered $\sigma^2$ (Figure 5.3b), the postprocessing does not changes results much. The postprocessing of networks with fixed $\sigma^2$ and altered $\mu$ (Figure 5.3a), results in larger differences. We therefore only observe the influence of postprocessing on networks with fixed $\sigma^2$ and altered $\mu$.

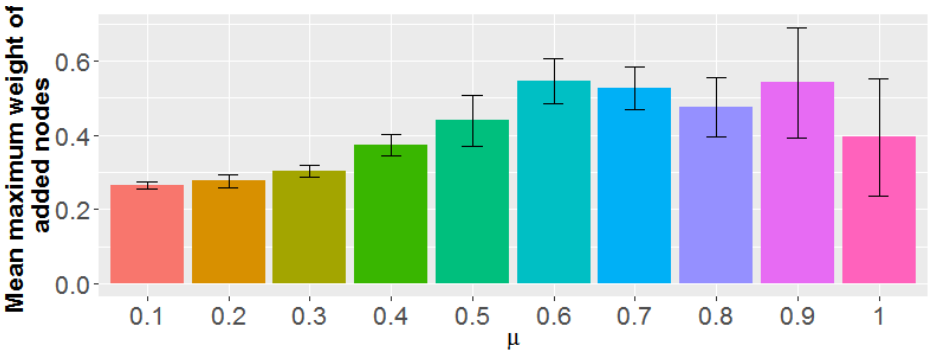



(a)

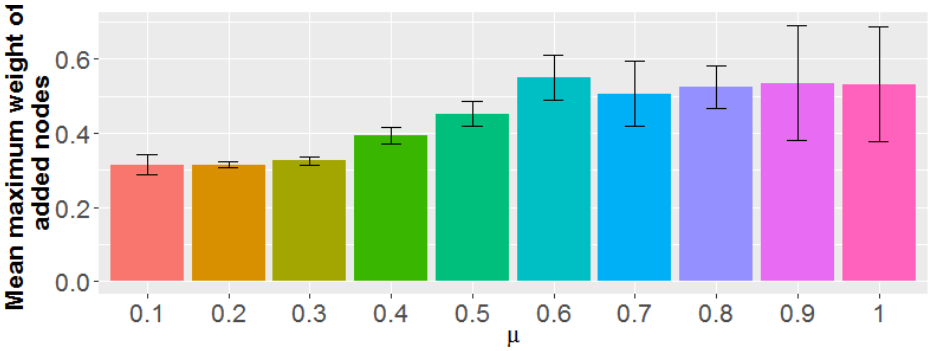



(b)

Figure 5.5.: Mean and standard deviation of the number of returned module nodes not in *predefined modules* before (a) and after (b) postprocessing, for weights of *predefined modules* $\sim N(0.5, \sigma^2)$, $\sigma^2 = 0.01, ..., 0.1, 0.15, 0.2, 0.25, 0.3$. The number of module nodes returned by the algorithm that are not in *predefined modules* decreases.




(a) Before postprocessing




(b) After postprocessing

Figure 5.6.: Mean and standard deviation of the maximum of the weight vector of each node in module nodes not in *predefined modules* before (a) and after (b) postprocessing, for weights of *predefined modules* $\sim N(0.5, \sigma^2)$, $\sigma^2 = 0.01, ..., 0.1, 0.15, 0.2, 0.25, 0.3$. The weights are high in general and that their mean weight is increased after postprocessing.

### 5.3.4 Simulation studies on a biological network

So far we have observed that the algorithm performs well on artificial networks. It remains an open question whether the algorithm detects all the interesting modules in a biological network. Answering this question is very difficult, if not impossible. However, we can easily investigate a related question: Whether a set of connected genes in a biological network, whose node weights are set high, is be detected as a module.

For this purpose, we insert a module into a biological fusion network, by changing the node weights of connected nodes such that they build a module, as defined by our algorithm hypotheses. We adjust nodes in a biological protein protein interaction network with weights that describe the change of each protein (related gene) in breast cancer, compared to healthy tissue for two different omics data collections (details in §6.5). A set of genes that is connected, but has not been detected as a module before, is selected (Figure 5.7).

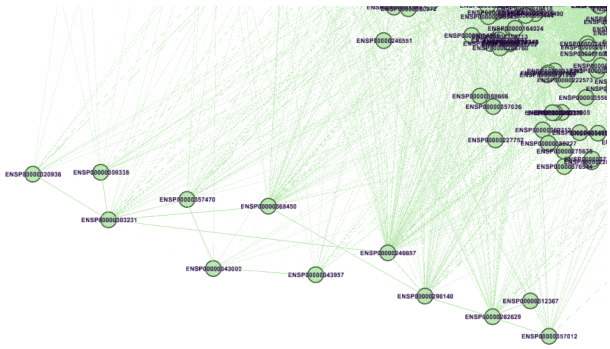
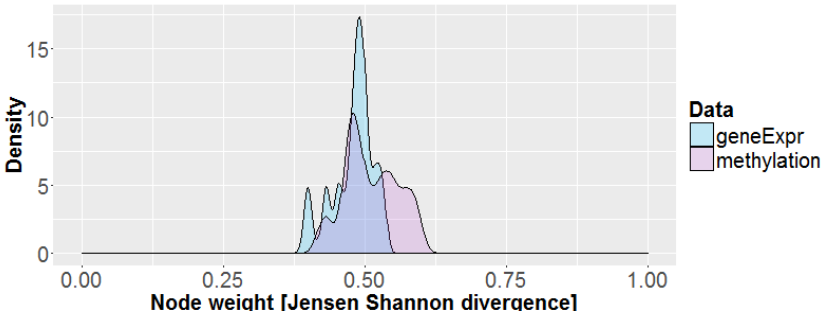


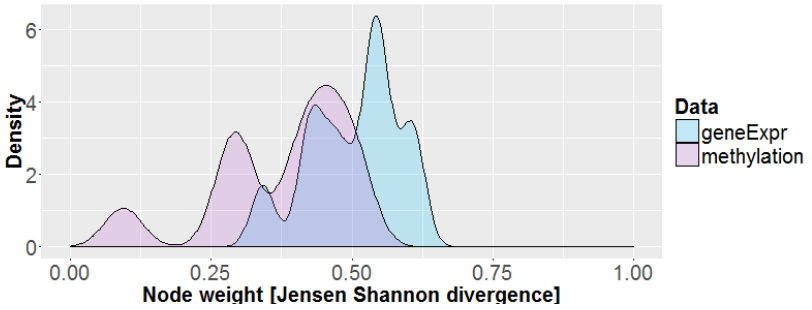
Figure 5.7.: Visualisation of the connected genes in a biological network for which node weights are altered, which thereby fits the characteristics of a module that our algorithm should detect.

The module weights for the two weight distributions of the selected nodes are manipulated twice: Firstly, similar high means for each weight type and small variance. Secondly, differing means and higher variance (see Figure 5.8). The weights in the complete network are distributed between approximately 0 and 0.6.

The set of nodes is entirely detected by the algorithm as one module for $p$=25 and several $\#m$.




(a) Very small difference between mean and small variance




(b) Slightly higher difference between means and higher variance

Figure 5.8.: Artificial distribution of increased node weights (Jensen–Shannon divergence for *gene expression* and *methylation*) of connected nodes in a biological network

# 6. Application to real world biological data

We have discussed three approaches for biomarker detection from biological data: Machine learning based biomarker detection, a network approach for biomarker detection, and a fusion network approach for biomarker detection. In this chapter, we evaluate these approaches on biological data.

We evaluate two different fusion approaches: the machine learning approach from §3 and our network from §5. The fusion data approach with machine learning methods is based on a simple vector joint, meaning that for each sample a vector of features from all data types is analysed. In the network case, the data fusion implies vectorial node weights, meaning that each feature (biological entity) is assigned several weights.

Note that several steps for preprocessing, transformation and analysis are necessary. For this purpose, we wrote R[1]-scripts of roughly 3500 lines.

## 6.1. Data

We analyse TCGA [46] level 3[2] breast cancer data from 31 healthy tissue and 199 tumour tissue samples. More precisely, we analyse level 3 methylation (Illumina Infinium 450k), as well as gene expression (RNAseq[3]) data. Hence, we analyse the DNA and mRNA level (see Figure 6.1). We choose these data types, to analyse data from inside and outside of the nucleosome, which enables an analysis of cellular processes at different regions of the cell. Furthermore, it allows for an investigation of their relation.

The application of biological data requires a preceding data analysis and preprocessing. Level 3 data is already normalised [63, 72, 14]. Therefore, a detailed normalisation is not performed. Exploration of this topic goes beyond the scope of this thesis. However, normalisation is an important step when biological data is analysed, and we can skip it only because TCGA's level 3 data is already normalised.

[1] Software environment, see [36]

[2] Level 1: Low level data for single sample, not normalized, level 2: Normalized single sample data, level 3: Aggregate of processed data from single sample

[3] For details see [78]

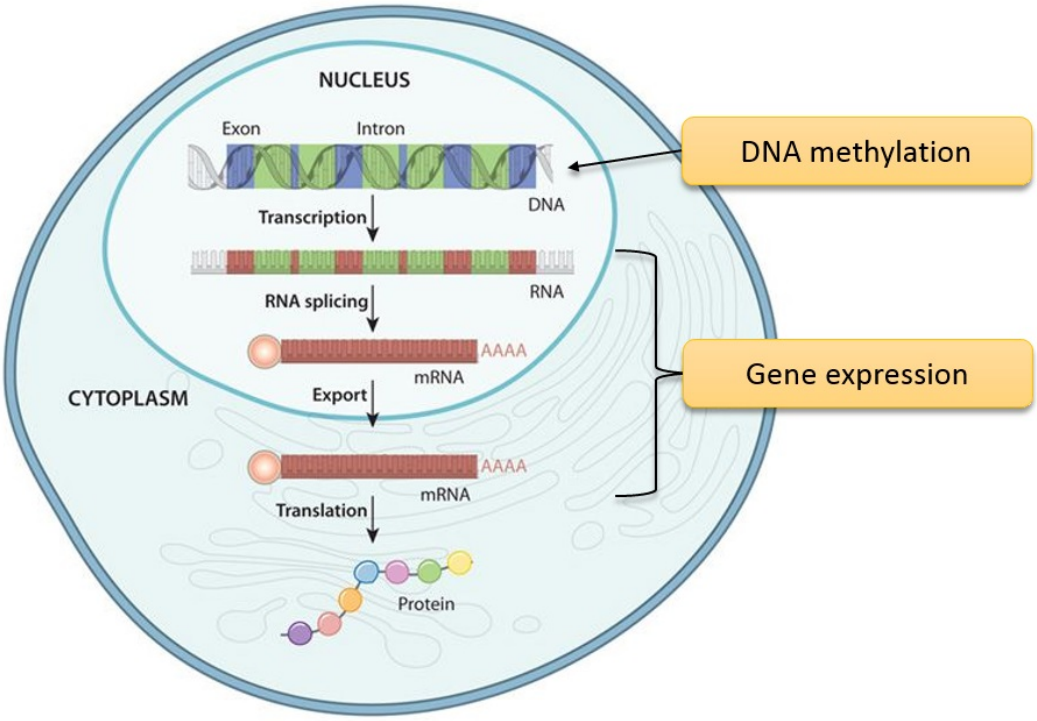



Figure 6.1.: Central dogma of molecular biology [4]. The DNA is transcribed to RNA, which is henceforth translated to proteins. In this work we analyse DNA methylation (DNA level) and gene expression (mRNA level) data.

### 6.1.1 Gene expression data

TCGA level 3 RNAseq data contains RPKM (Reads Per Kilobase per Million mapped reads) normalised read counts for 20502 genes.

RPKM normalises for total read length and the number of sequenced reads. It is defined as

$$RPKM = \frac{10^9 * C}{N * L} , \tag{6.1}$$

where $C$ is the number of reads mapped to a gene, $N$ the total number of mapped reads in the experiment and $L$ the exon length in base-pairs for a gene.

Figure 6.2 shows the distribution of RPKM values in our data.

[4]Source: www.nature.com

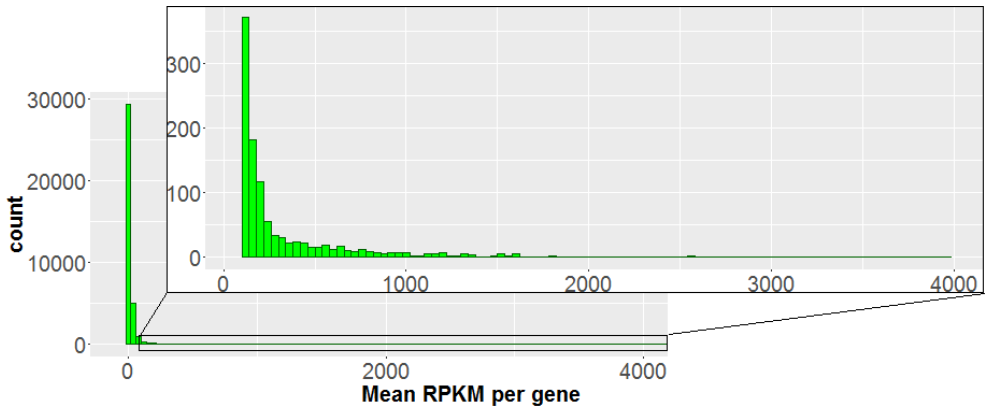



Figure 6.2.: Histogram of mean RPKM per gene in level 3 breast cancer data from TCGA. The majority of PRKM values is smaller than 100. The maximum RPKM value is approximately 4000.

### Handling of zero entries

In RNAseq level 3 gene expression data, approximately one percent of RPKM values are exactly zero, which is striking, because it means that a gene is not expressed at all. 10% of RPKM values are smaller than 0.1, but different from zero, which suggests that values being exactly zero are actually of such kind, but arise from normalisation or erroneous measurement. To analyse if this is the case, we analyse the distribution of genes with at least 10 zero entries, excluding the values that are exactly zero. It shows that for approximately 50% of these genes, the non zero RPKM values are significantly larger than 0.1.

We cannot allow for large numbers of dubious measurements, because we are interested in identifying typical patterns for a specific disease (stage). Including dubious measurements can have negative effects on our results if they are incorrect. Therefore we exclude genes, for which we do not have enough values different from zero. Figure 6.3 shows the amount of genes having a certain number of zero entries. The maximum threshold is chosen to be 31, because the control data set has 31 samples. We can see

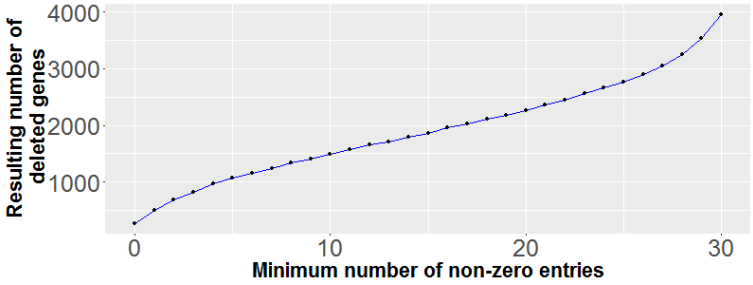



Figure 6.3.: Relation of removed values for allowed RPKM measurements different from zero. For allowed values between 5 and 25 the number of removed PRKM measurements increases almost linearly.

that there is no value that results in a considerably smaller number of deletions than any higher value. For thresholds larger than approximately 20, we would delete more than 10% of genes. Therefore, we remove genes with up to 20 zero entries per gene. This results in a deletion of data for 2266 genes.

### Handling of outliers

We observe that RPKM values for some genes are aggregated around a certain value, but a few measurements lie very far apart from the rest (Figure 6.4). In the later study we analyse the distribution of measurements per gene. Outliers lead to an increase of the range of the distribution, which makes small changes less visible.

We decide to replace these outliers with random values, that are still higher than normal, but not as dramatically divergent from mean. Values are replaced randomly to not to perturb the distribution of measurements all too much: If we would set all these outliers to a fixed value, a second peak in the distribution of measurements arises if there is more than one measurement to correct for.

A measurement is considered an outlier, if, by exclusion of all gene's RPKM values larger than the 99th percentile of all RPKM values of that gene, the mean of all measurements is altered by more than 25%. We define this threshold based on observations of various RPKM distributions, as it detects striking outliers well. These values are replaced with random values, from the uniform distribution $\mathcal{U}(mean(RPKM) + |mean(RPKM) + 99th\,percentile|/2,\ 99th\,percentile)$, which are high compared to all measurements for the gene. This is done iteratively until the condition is fulfilled for all genes. An example for the result of the outlier replacement is given in Figure 6.4.

Measurements for 1772 genes from the case and for 597 genes from the control group are modified by the above mentioned procedure.

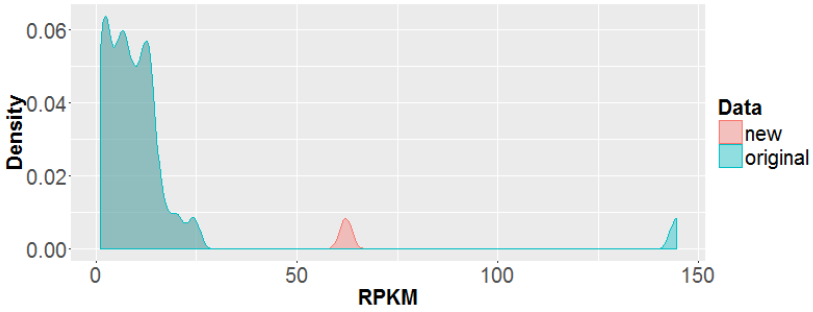



Figure 6.4.: Distribution of RPKM values for gene ASPN in control data before (blue) and after (red) outlier replacement. Before the outlier is replaced, it appears as a striking very large value. After the data processing a new peak appears at $\sim 60$. The distribution of all PRKM values remains almost unchanged.

### 6.1.2 Methylation data

The Illumina Infinium HumanMethylation450 BeadChip allows genome-wide profiling of methylation for over 450000 methylation sites [37].

For each methylation site a beta value is provided. The beta value is the ratio of the methylated probe intensity and the overall intensity. The beta value for a methylation site $i$ is defined as:

$$\beta_i = \frac{max(y_{i,meth}, 0)}{max(y_{i,unmeth}, 0) + max(y_{i,meth}, 0) + \alpha} , \tag{6.2}$$

where $y_{i,unmeth}$ and $y_{i,meth}$ are the intensities measured for the $i^{th}$ unmethylated, respectively methylated probes. A constant bias $\alpha$ is added to regularise $\beta$ when the intensities for both probes are low. Background subtraction might result in negative values, which is why the maximum term is chosen. This results in a beta-value between 0 and 1 [22].

Figure 6.5 shows the distribution of beta values in the TCGA breast cancer data.

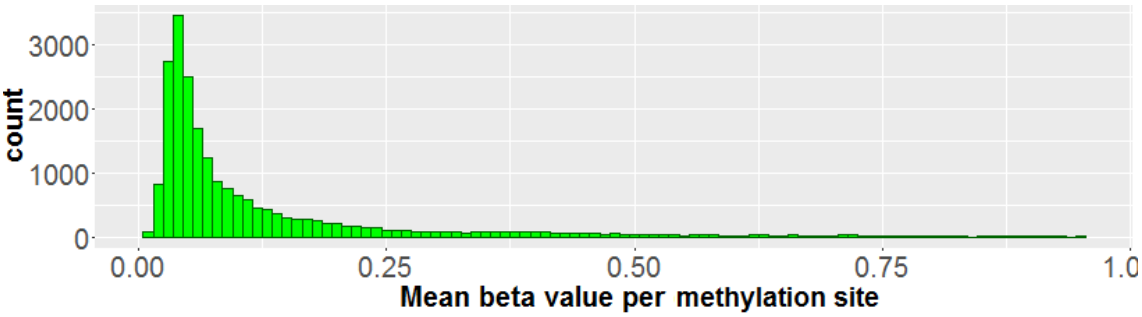



Figure 6.5.: Histogram of mean beta value per methylation site in level 3 breast cancer data from TCGA. The majority of beta values lies in a range of 0.1 and 0.25. The maximum beta value is close to 1.

#### Integration of Methylation Data

In the used TCGA's level 3 methylation data a mapping between 365925 of the 485577 methylation sites to 21232 genes is included. Accordingly numerous distinct methylation sites are assigned to one gene. Further, 33590 methylation sites are assigned to at least two genes.

For the integration of methylation data in our analysis, it is required to assign one value to each gene (per data source). Therefore, methylation data has to be processed, before we can include it in the analysis.

How to process methylation data reasonably, such that one value for each gene remains, is still unclear. The question of the consequences of methylation comes along with this problem. Unfortunately, there is not yet a definite answer. The consensus in literature is to assume a reduction of the gene expression when the methylation appears in a promoter region and an increase of gene expression for a methylation in the gene-body

[60, 77, 9]. However, it has been demonstrated that this relationship does not always hold [77].

While the consequences of methylation are not yet clear, approaches for the transformation of methylation data already exist. There are currently two main approaches.

One approach is to restrict the analysis to the methylation of the promoter region of a gene [73, 42]. An argument for this approach is the high influence of the promoter in the expression of the gene in general. The information on methylation of the non promoter regions is neglected in this approach.

Another approach is to consider the mean methylation beta value of all methylation sites that are associated with the gene [50, 82]. This approach contradicts the assumption of the opposite effects of methylation in the promoter region and the gene-body. Further, this approach might be an oversimplification. If a small amount of methylation sites of a gene are highly methylated and the others are not, this information is lost by considering the mean.

To decide for one transformation, we evaluate the results of both approaches as depicted in the following.

**Methylation data transformation.** Methylation data is transformed in two different ways:

- The transformation in which the mean of the methylation sites that are associated with the gene is selected as the gene's methylation value, is referred to as the *all sites* transformation in the following
- The approach to associate the mean of the methylation sites that have a promoter association with the gene, is referred to as the *promoter* transformation.

Promoter associations for all methylation sites are obtained by the R-Package IlluminaHumanMethylation450k.db. For 10612 of the 21232 genes, associations for at least one of their methylation sites as a promoter region, exist.

To allow for comparison of the different methylation data transformations, methylation data for genes which have at least one promoter associated methylation site are selected in the *promoter* and in the *all sites* transformation. This is performed for case and control data. In the *promoter* transformation, we keep only those methylation sites that have a promoter association.

Next, for each sample, for each remaining gene, the mean of the beta values of the remaining methylation sites associated with that gene is stored.

**Analysis of methylation data transformations.** To evaluate which transformation should be applied, classifiers are trained with the three methods described in §3 and their performances are compared with the F1 score (see below). The analysis of results for three different methods allows to better detect if the performance is based on the data or on the method.

This approach requires data of the following format: for each sample, data for the features (genes) and a response based on which the classifier partitions the data. The

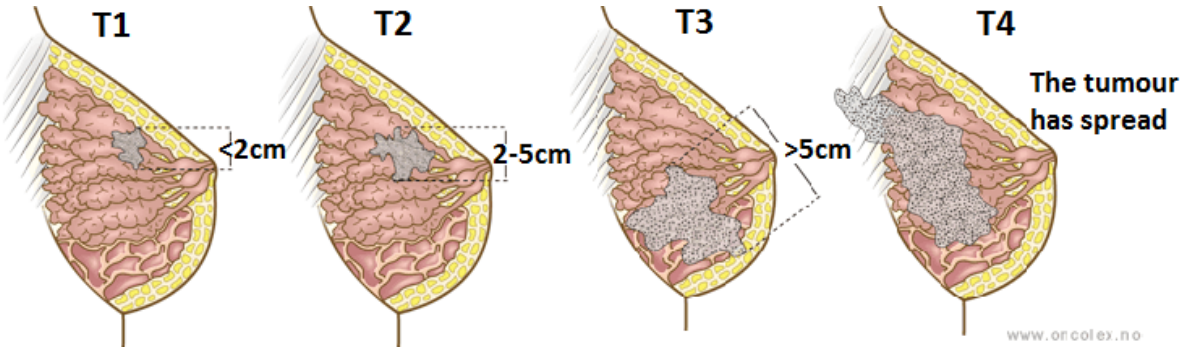



Figure 6.6.: Illustration of T stages 1 to 4 of breast cancer[5]. The grey coloured area represents the tumour. Classes T1, T2, and T3 are characterised by the increasing size of the tumour. A tumour of class T4 has spread into the surrounding tissue.

tumour stage (column *ajcc_tumour_pathologic_pt* in TCGA's clinical data) is chosen as the response.

The tumour stage describes the extent of the primary tumour or the infestation of additional tissue. Thus, it describes a characteristic of the tumour at the time of data acquisition. The information *days to death* for example, which is also available in TCGA's clinical data, does not fulfil this requirement: It is influenced by initial disease stage, therapy treatment, psychological issues, and other, possibly non-disease, related issues. In Figure 6.6, the T stages of breast cancer are illustrated.

For the present data, this results in 8 different classes: T1 (8 cases), T1a (1 case), T1b (3 cases), T1c (34 cases), T2 (118 cases), T3 (29 cases), T4 (2 cases), T4b (2 cases), TX (2 cases). *T* stands for *primary tumour*. The number describes the extent of the primary tumour or the infestation of additional tissue, X means unclear. The lower case letters are subcategories of the number categories. The subcategories are neglected, so as to obtain classes of bigger size.

Classes T4 and TX have few representatives only, which is why we decide to exclude them from this study. This results in three classes: T1, T2, and T3. Class T3 has the minimum number of cases: 29. Control data is included with class label T0.

**Pipeline of methylation data analysis.** We compute data sets that have been transformed by either of the two transformation methods described above. For these data sets, for each gene, the Jensen–Shannon divergence (equation (6.5)) between case and control data is computed. Only the genes with the 100, respectively 1000 highest Jensen–Shannon divergences are kept.

We train classifiers with the remaining data from the *all sites* and *promoter* transformations. 10-fold cross validation with 5 repeats is applied, which allows to not to split the data into training and testing data, which we would like to circumvent, since the data set has few samples per class. The results in this step are strongly influenced by

[5]Source: `http://oncolex.org/Breast-cancer/Background/Staging`

the method applied for training the classifier, which means we are not only evaluating the transformation, but also the method that is used to train the classifier. To slightly reduce this problem, we evaluate the transformed data with three standard methods: SVM, random forest, and elastic net, so that we can better detect if the performance is based on the data or on the method. Details on the applied methods are given in §3. The classifiers are evaluated based on their classification performance on sampled test data (F1 score, see below). Subsequently, the classification results for the different classifiers are evaluated. Figure 6.7 shows a visualisation of the steps that are performed for each transformation.

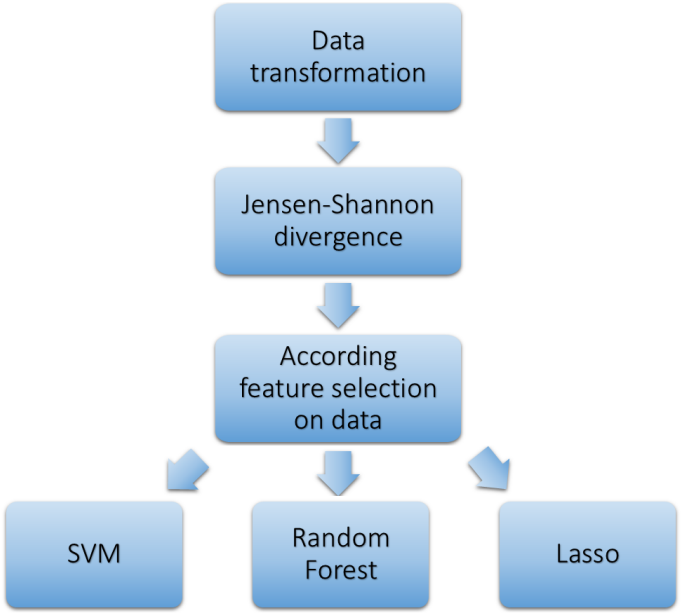



Figure 6.7.: Pipeline of methylation data analysis for one transformation: Once the data is transformed, the JSD for each gene between case and control data is computed and the genes with the highest JSDs are selected. Subsequently, classifiers are trained and tested with the measurements of the remaining genes.

**Classifier performance.** For the training we select the same number of samples for each class from all samples, to not to bias the resulting classifier. We randomly select 29 samples from all samples. The testing data set has approximately half of the size of the training data set: 14 samples from each tumour stage, which we randomly select from all samples.

Classifiers are trained with R package CARET. The TRAIN function in CARET is applied to train classifiers. It includes optimisation of method parameters. To assess the performance of each model, the F1 score is computed for 100 classifications with randomly selected samples. The F1 score is defined as:

$$F1 = \frac{2 * PPV * TPR}{PPV + TPR} , \tag{6.3}$$

where PPV is the positive predictive value (precision) and TPR the true positive rate (recall). Hence, the F1 score is the harmonic mean of precision and recall. We choose the F1 score over standard accuracy, because standard accuracy can be misleading. Sometimes it may be desirable to select a model with a lower standard accuracy because it has a greater predictive power on the problem. The F1 score incorporates this.

Results are shown in Table 6.1. The *promoter* transformation performs best for all classifiers and will therefore be chosen as the transformation method for the following analysis.

Table 6.1.: Mean F1 score over 100 testing runs of classifiers for methylation data transformations *promoter* and *all sites*. Each classifier observes data for genes with the top 100/1000 Jensen–Shannon divergences (JSD). Transformation *promoter* outperforms *all sites* transformation in all settings.

| | Transformation | F1 score, SVM classifier | F1 score, Random forest classifier | F1 score, Elastic net classifier |
|---|---|---|---|---|
| JSD top 100 features | *promoter* | **0.805** | **0.807** | **0.812** |
| | *all sites* | 0.770 | 0.804 | 0.795 |
| JSD top 1000 features | *promoter* | **0.830** | **0.839** | **0.831** |
| | *all sites* | 0.794 | 0.793 | 0.800 |

## 6.2. Machine learning based biomarker detection for single source data

For the machine learning approach a matrix of type $samples \times features$ and a class label for each sample are required. We evaluate the preprocessed data described in §6.1. The class label is either *healthy* or *tumour*.

The F1 score (see above) of the classifiers are shown in Table 6.2. All F1 scores are equal or close to 1. This is not surprising, and has to do with the so called *curse of dimensionality*: the complexity of a subspace increases with the number of dimensions. In order to obtain a statistically reliable result, the amount of data needed to support the result often grows exponentially with the dimensionality. We do not have such a large number of samples and are hence overfitting.

We performed similar experiments with the three tumour classes from §6.1.2 and control group. Results are contained in the appendix. In that case, except for the classifier trained with gene expression data and elastic net, the results are all similar and slightly better than 80%.

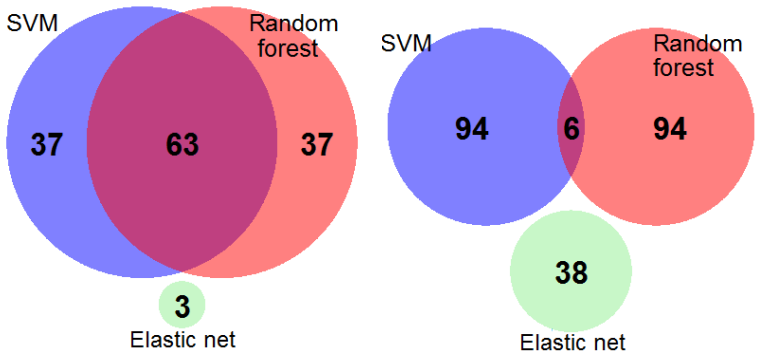
Usually, one is interested in identifying a small set of biomarkers. In biomarker detection, this leads to biomarkers with high interpretability. Further, medical testing for

Table 6.2.: Mean F1 score over 100 testing runs. Two classifier are trained with the complete gene expression, respectively methylation data. All F1 scores are equal or close to 1.

| Data type | F1 score, SVM classifier | F1 score, Random forest classifier | F1 score, Elastic net classifier |
|---|---|---|---|
| Gene expression data | 1 | 0.989 | 1 |
| Methylation data | 1 | 0.989 | 1 |

few biomarkers is simpler and cheaper. Therefore the features with the highest classifier relevance are focussed in the result. They are mainly driving the classifier's results. The 100 main features vary strongly between classifiers (Figure 6.8). Only the top features for methylation data SVM and random forest classifiers show a high overlap. The elastic net selects only 3 features from the methylation data.

A biological interpretation of the result is questionable. First of all, it is not clear which features to select. We could select the top 100 (10,20,50,1000,...) features, or define a threshold for the feature importance, for features to be included in the final result. Either way, we would have to come up with a number that can hardly be argued for. This does not hold for the elastic net result, because it includes a feature selection (embedded approach). Further, we would have to investigate the function of each gene individually. Hence, it is difficult to draw conclusions in terms of biomarker selection at this point.




(a) Methylation data (b) Gene expression data

Figure 6.8.: Venn diagram of the 100 most important features of classifiers trained with three different methods for methylation data (a) and gene expression data (b). Only the top features for methylation data SVM and random forest classifiers show a high overlap. The elastic net selects only 3 features from the methylation data.

## 6.3. Biomarker detection with the single data network approach

In the network approach we analyse data that is combined with a network to obtain highly interpretable results. We analyse a protein protein interaction (PPI) network. In the PPI network nodes correspond to proteins and edges represent pairwise interactions between proteins. The reason for the incorporation of a network is that impacts of individually altered proteins can often be balanced, whereas alterations of interacting proteins often lead to major consequences [7, 12]. Hence, our results are not only better interpretable than results from standard approaches (as for example the machine learning approach which is applied in §6.2), but also more meaningful.

We combine the STRING PPI network [70], version 10, with our data. Combining data and network in this case means that each node is assigned a weight, which is derived from the data. STRING (*Search Tool for the Retrieval of Interacting Genes/Proteins*) is a database of predicted PPIs. Interactions are derived from three kinds of sources: High-throughput experiments, mining of databases and literature, and prediction from genomic context analysis. The human STRING PPI network consists of 8548002 interactions for 19247 proteins.

We apply our algorithm (§4) to detect biomarker genes. More precisely, we search for proteins, that are altered in the case group, compared to the control group (specifically the expression of one of its coding genes, or the methylation of promoter regions of the genes), which are interacting with other affected proteins.

### 6.3.1 Preprocessing for network application

Before we can start the network analysis, we need to preprocess data further, such that it is mappable to a network:

- The gene IDs in network and data are distinct, so either of the IDs has to be converted.
- One value per gene and data type is required, describing the difference of measurements between case and control.

#### ID conversion

Nodes in the STRING PPI network are labelled with ensembl protein IDs. TCGA data, however, contains data for genes labelled by gene symbol. To allow for a combination of network and data, TCGA's gene symbol IDs are converted to ensembl protein IDs. Mappings of the gene symbols to ensemble proteins are derived from R-package MYGENE as well as databases bioDBnet [52] and g:Profiler [61] .

Use of package MYGENE yielded 81927 ensembl protein IDs for 16612 of the 22290 gene symbols form TCGA data. From databases bioDBnet and g:Profiler 94992 conversions for 17761 gene symbols are obtained. Altogether, 96433 ensembl protein conversions for 17922 gene symbols are obtained, whereof 94669 ensembl protein IDs are distinct.

Precisely, for the RNAseq data for 17605 of 18236 gene symbols, a conversion to ensembl protein was obtained. For methylation data a conversion for 9493 of 10679 gene symbols to 58379 ensembl gene IDs was found.

There are two reasons for the multiple assignment: Firstly, one protein can consist of several polypeptides, each of which is coded by one gene. Secondly each gene can code for multiple polypeptides, due to alternative splicing.

The conversion step has a high influence, since on the one hand we are loosing data for numerous genes for the subsequent analysis and on the other hand get numerous multi alignments. Nonetheless it is a required step. We should keep in mind that if we were to improve in this step, the results would change most likely.

### Difference measure

In the final network analysis, it is required to assign one value to each protein for each omics data type. Up to this point, case and control data are still separate. To compare between all case and all control samples with a network approach, we need to summarise.

The value assigned to each node should inform about the change of the measurements of a gene between case and control group. In this section, the results of two difference measures: Fold change and Jensen–Shannon divergence (further JSD), are analysed.

**Fold change.** Applying log fold change, defined as

$$\left| \log_2 \left( \frac{mean(measurement_{cotrol})}{mean(measurement_{case})} \right) \right| , \tag{6.4}$$

which I will refer to as fold change in the following, is a standard method for data group comparison. The fold change provides information about the shift of the mean of two distributions, but not about an altered type of these. Hence, the fold change involves a normality or uniform distribution assumption of the measurements for each gene, because it only regards the mean of all measurements. Furthermore, if the values follow e.g. a normal distribution with the same mean, but different standard deviation, the fold change will not detect this difference.

**Jenson-Shannon divergence.** The Jenson-Shannon divergence, defined for two probability distributions $P$ and $Q$, by

$$JSD(P \parallel Q) = \frac{1}{2} D(P \parallel M) + \frac{1}{2} D(Q \parallel M) , \tag{6.5}$$

where $M = \frac{1}{2}(P + Q)$ and $D(P \parallel Q) = \sum_{x \in X} P(x) \cdot \log \frac{P(x)}{Q(x)}$ is the Kullback–Leibler divergence, measures the distance between two distributions $P$ and $Q$ and does not include an assumption on the type of distribution. Another advantage of the JSD is that it is bounded by 0 and 1 and can therefore be interpreted easily.

In the Jenson-Shannon divergence analysis between case and control distribution, it must be noted that the number of samples per class differs: 30 samples for control and


(a) Gene expression data (b) Methylation data

Figure 6.9.: Scatter plots of fold change and JSD for gene expression data (a) and methylation data (b). For both data types, the plot tends towards a linear trend. Two things are striking: Firstly, the range of the fold change is smaller for methylation data, than it is for gene expression data ($\sim$[0,2], compared to $\sim$[0,8]). Secondly, in both plots points accumulate along the x-axis.

200 samples for case (minus zero measurements for gene expression data, see §6.1.1). This might have a critical influence on the resulting distributions. We are restricted to the number of samples provided, but we should keep this in mind.

**Comparison of fold change and Jenson-Shannon divergence.** In order to decide for one strategy, we look at the results for fold change and JSD, applied on the methylation and gene expression data. Concerns regarding the fold change have been indicated above. The following analysis will reveal if they are justified for the selected data.

The scatter plot (Figure 6.9) of fold change and JSD, gives a first overview of their mutual behaviour. For both data types, the plot tends towards a linear trend. Two things are striking: Firstly, the range of the fold change is smaller for methylation data, than it is for gene expression data ($\sim$[0,2], compared to $\sim$[0,8]). Secondly, in both plots points accumulate along the x-axis.

The different range of the fold change values is caused by the range of the data. For gene expression data, the values per gene spread more than for methylation data.

Figure 6.10 shows the case and control RPKM-distribution of a gene with very small fold change and medium JSD. The assignment of a small fold change is misleading, because the range of the RPKM values in the case group differs from the range of the RPKM values in the control group ($\sim$[3,6], compared to $\sim$[2,14]). This observation signifies a difference between the two groups, and the fold change does not detect this difference. We should remember at this point that the group sizes are different, but this does not necessarily mean that the range of the distribution should be decreased as much. For gene SIRT5 not a single sample in the control group has an RPKM values

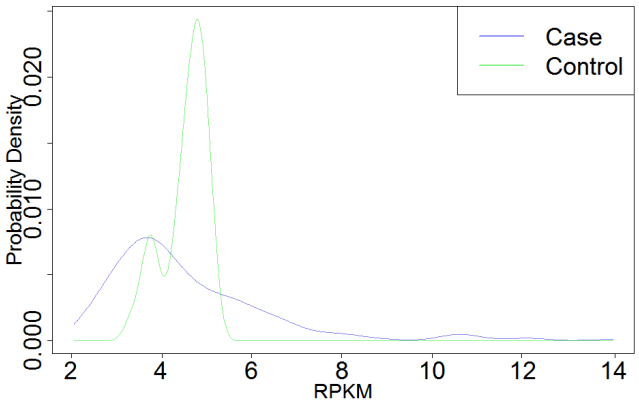



Figure 6.10.: Distribution of RPKM values for gene SIRT5 (Fold change = 0.03; JSD = 0.24). The range of the RPKM values in the case group differs from the range of the RPKM values in the control group. The fold change does not capture the difference of the distributions.

greater than 6. Numerous other genes show this behaviour. Based on these observations, the JSD is chosen as difference measure.

With the decision to use a difference measure as node weight, we loose the information of the actual expression, respectively methylation intensity. In this analysis we content ourselves with this, nevertheless it could be interesting to include both, change and respective intensity, in the disease samples in a later analysis. For the comparison of approaches with and without a network, we have to remember that the approach without the network has all the gene measurements as input.

### 6.3.2 Results

For the analysis of the data with our network approach, we have to decide on a range of parameters to observe. Based on observation in §4.3, the minimum of $p$, which is the parameter that influences the similarity criterion, is chosen to be 10. For $p$ significantly larger than 50, $L_S$ (equation (4.6)) becomes numerically unstable. The maximum $p$ is therefore set to 50. $p = 25$ is tested as an intermediate $p$. Values for $\#m$, the desired number of modules, are determined from the spectrum of $L_S$ (equation (4.6)).

We evaluate the genes in the module sets with two different approaches: In the mathematical evaluation, we regard each module as a set of connected biomarkers. The modules should consist of nodes with a high weight (JSD), thus with a high difference between case and control data. We apply the methods explained in §3.1. The input matrix consists of measurements for the module genes only. F1 score (6.3) is applied for performance assessment. As discussed in §3.2 the machine learning method has a great influence on the result. For this, we train classifiers with data for the module genes (this is afeature selection). Three different training methods are applied to evaluate these feature selections: SVM, random forest, and elastic net, so that we can better detect if the performance is based on the data or on the method.

In the biological evaluation, KEGG [39] pathway analysis of the module genes using DAVID [19] is performed, to analyse if the results are relevant in the context of breast cancer. We choose KEGG database, because it returned the largest number of pathways in our analysis, comparing to other pathway databases in DAVID.

### Scoring of modules

To compare between module sets for varying parameters $p$ and $\#m$, the F1 score is considered. The reason that we analyse multiple values of $\#m$ is, that in almost all cases more than one spectral gap shows up. However, the weight distribution of each module is another important characteristic, which we evaluate with the JSD of the weight distribution of all network nodes, compared to module set distribution. To allow for comparison between results we score each module with a combination of the aforementioned F1 score and JSD: $JSD * F1_{max}$, where $F1_{max}$ is the maximum F1 score of the three classifiers trained with SVM, random forest, and elastic net.

The consequence of choosing module sets with high scores $JSD * F1_{max}$, is that in most cases small set of genes have maximum score compared to other module sets. The reason is that the algorithm usually returns modules with higher JSD for smaller values of $\#m$, because it detects the most significant nodes in the first modules. If we were interested in finding as many interesting genes as possible, we could use a score that includes the number of genes detected.

### (1.) Gene expression data analysis

After postprocessing of the modules in the gene expression network we observe: all module genes for $p = 10$ are in modules for $p = 25$ (Figure 6.11). For distinct $\#m$, but for same $p$, modules for larger values of $\#m$ always contain all nodes of modules for smaller values of $\#m$. To simplify, in that case, only the genes for larger values of $\#m$ are shown in Figure 6.11.

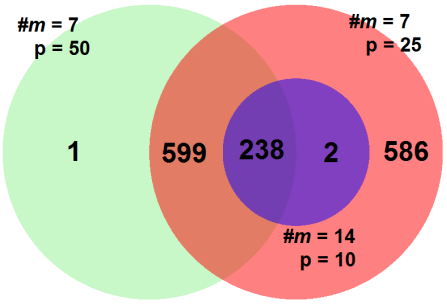



Figure 6.11.: Venn diagram of the genes belonging to the largest module sets (respective $\#m$ in visualisation) obtained from gene expression data after postprocessing for for $p = 10, 25, 50$. All module genes for $p = 10$ are in modules for $p = 25$. All except one gene for $p = 50$ are in module genes for $p = 25$.

Table 6.3.: Mean F1 score over 100 testing runs. Each classifier is trained with data for the module genes returned by our algorithm for different values of $p$ and $\#m$ for gene expression data before and after (*)postprocessing. After postprocessing the score for the module sets with fewest genes is the highest for each $p$ iteration. The $F1_{max}$ is close to 1 in all cases. The number left and right to the "$\rightarrow$" in the $\#m$ of the preprocessing modules indicated which module set is altered and how many modules remain.

| Data | Parameters | | Results | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $p$ | #m | Genes | JSD | F1 score, SVM classifier | F1 score, Random forest classifier | F1 score, Elastic net classifier | Score: $JSD * F1_{max}$ |
| Gene expression | 10 | 6 | 160 | **0.562** | **0.996** | 0.992 | **0.996** | **0.560** |
| | | 8 | 165 | 0.523 | **0.996** | 0.983 | **0.996** | 0.521 |
| | | 14 | 283 | 0.448 | **0.996** | **0.995** | 0.992 | 0.446 |
| | 25 | 7 | 1432 | **0.497** | **1** | 0.992 | **0.996** | **0.497** |
| | | 9 | 1438 | 0.492 | **1** | **0.996** | **0.996** | 0.492 |
| | | 11 | 1447 | 0.484 | **1** | **0.996** | **0.996** | 0.484 |
| | | 18 | 1460 | 0.472 | **1** | **0.996** | **0.996** | 0.472 |
| | 50 | 7 | 1593 | **0.399** | **0.996** | 0.992 | **0.996** | **0.397** |
| | | 9 | 1596 | 0.395 | **0.996** | **0.996** | **0.996** | 0.393 |
| | | 11 | 1607 | 0.383 | **0.996** | **0.996** | **0.996** | 0.381 |
| | | 18 | 2808 | 0.28 | **0.996** | 0.992 | **0.996** | 0.279 |
| | 10* | 6 $\rightarrow$ 1 | 152 | **0.608** | **0.996** | 0.992 | **0.996** | **0.606** |
| | | 8 $\rightarrow$ 1 | 150 | **0.608** | **0.996** | 0.992 | **0.996** | **0.606** |
| | | 14 $\rightarrow$ 1 | 240 | 0.599 | **0.996** | **0.995** | 0.992 | 0.597 |
| | 25* | *all* $\rightarrow$ 2 | 1425 | **0.507** | **1** | **0.996** | **0.996** | **0.507** |
| | 50* | *all* $\rightarrow$ 3 | 838 | **0.555** | **0.996** | **0.996** | **0.996** | **0.553** |

**Classifiers.** Table 6.3 shows details on the modules and their performance as feature selections for all parameters $p$ and $\#m$ evaluated. Column *Genes* specifies the number of genes in all modules for the respective setting. Best values per $p$ are marked bold.

After the postprocessing explained in §4.2, the score for the module sets with fewest genes is the highest for each $p$ iteration. The $F1_{max}$ is close to 1 in all cases, so that the JSD has an high influence on the difference of scores. The score decreases for increasing $p$ before postprocessing.

Modules for $p = 10$ and $\#m = 6$, respectively $\#m = 8$ after postprocessing have the highest score. The 150 genes in modules after postprocessing for $p = 10$ and $\#m = 8$ are included in the genes for $p = 10$ and $\#m = 6$. Figure 6.12 shows a visualisation of the module for $p = 10^*$ and $\#m = 6$.

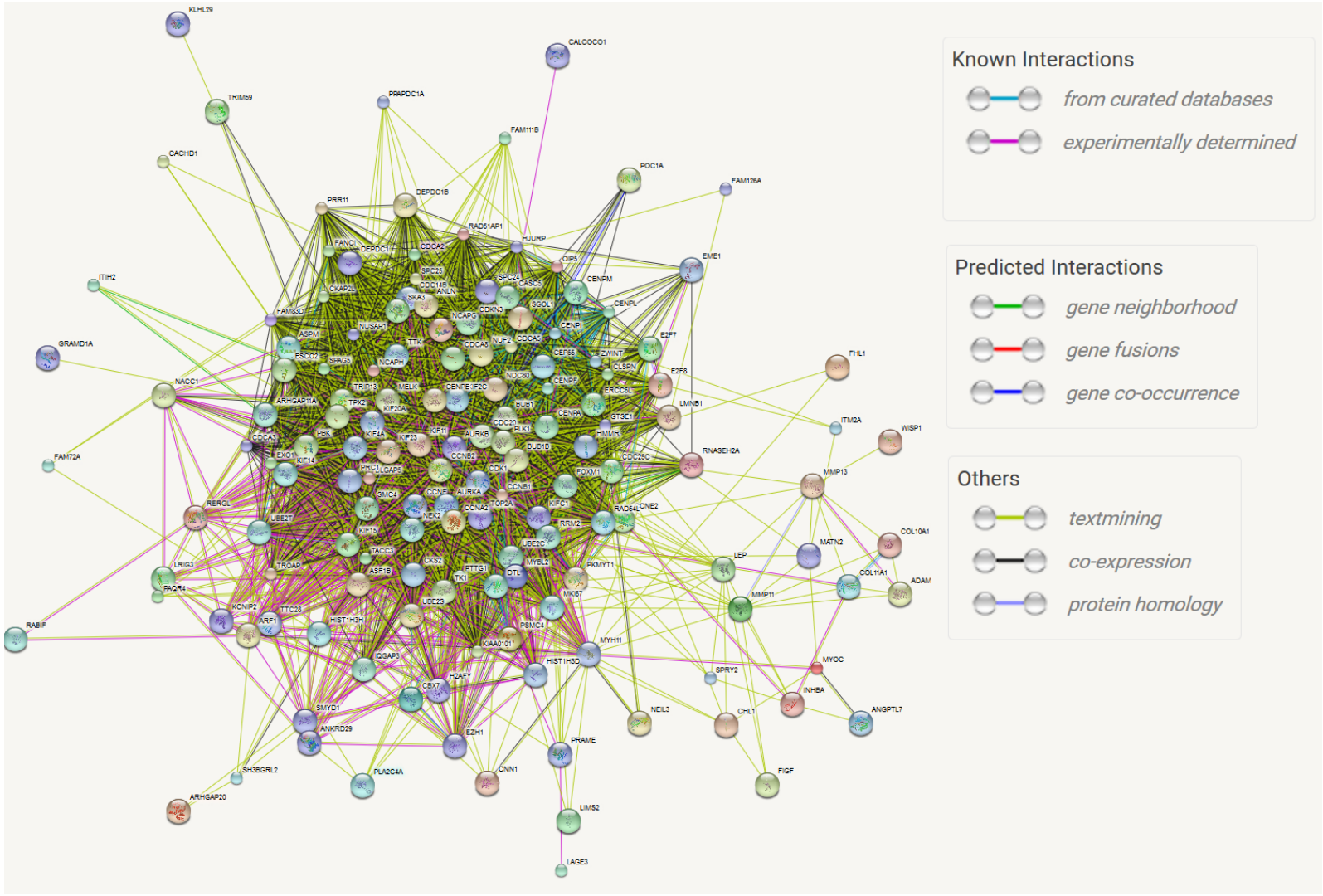



Figure 6.12.: The best scoring module from the gene expression data network for $p = 10$ and $\#m = 6$ after postprocessing: 152 genes connected with different edge types.

**Biological analysis of best performing result.** Only the modules with the best score are evaluated ($p = 10$ and $\#m = 6$ and $p = 10$ and $\#m = 8$; after postprocessing). A complete list of genes in the larger module set ($p = 10^*/\#m = 6$) is listen in the appendix. The pathways analysis results are identical for both modules. P values for all pathways are significant:

| Module | KEGG pathway | P value | Genes involved |
|---|---|---|---|
| 1 | Cell cycle | $1.38\hat{}10^{-12}$ | 14 |
| | Oocyte meiosis | $1.56\hat{}10^{-10}$ | 12 |
| | Progesterone-mediated oocyte maturation | $2.11\hat{}10^{-6}$ | 8 |
| | p53 signaling pathway | $1.15\hat{}10^{-4}$ | 6 |

**Cell cycle pathway.** Module genes involved in the *Cell cycle* pathway are cell division (CD) genes, cyclin (CCN) genes, protein kinases, mitotic checkpoint complex (BUB) genes, and an oncogene: PTTG [74], which is an anaphase promoting complex substrate.

**Oocyte meiosis pathway.** 10 genes of the *Cell cycle* pathway, reappear in the *Oocyte meiosis* pathway. The two additional genes are AURKA and SGOL1.

AURKA has been mentioned in the context of breast cancer [68, 88, 75].

**Progesterone-mediated oocyte maturation pathway.** Genes in involved in pathway *Progesterone-mediated oocyte maturation* are within the genes involved in *Cell cycle* and *Oocyte meiosis* pathways.

**p53 signalling pathway.** p53 is a well know tumour suppressor gene [28]. Genes involved in this pathway, that are not within the previously mentioned pathways, are: GTSE1 and RRM2. An importance of RRM2 for breast cancer is indicated by Closer et al. [17].

**Remark on pathway results.** All pathways are reasonable in the context of breast cancer, as they refer processes in women, or processes relevant in cancer.

Note that the module genes assigned to the pathways are intersecting - in fact this is not surprising because of our underlying network. All genes are in one module. Hence they are connected in the network and the pathway analysis confirms this.

Only 18 of the 150 module genes are involved in the pathways. This, however, does not mean that the results are weak. This highlights one of the strength of our approach, as our methods does not require a pathway annotation. Firstly, KEGG pathways cover only a fraction of proteins in a whole proteome. When applying networks as the STRING network, more information than pathways are included. The results benefit from that. The genes not involved in pathways are interacting with the pathway genes is some way, which is interesting. The genes not involved in KEGG pathways are interacting with the KEGG pathway genes, and this is indeed interesting.

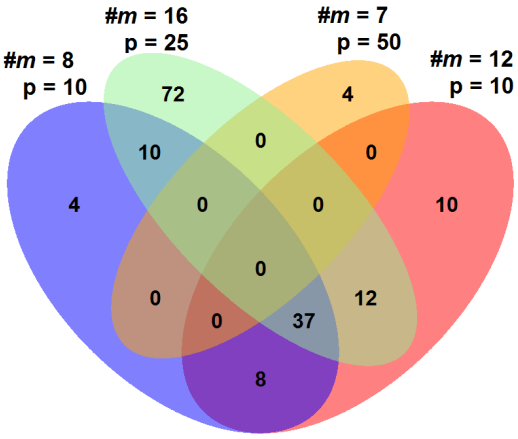



Figure 6.13.: Venn diagram of the genes of the largest module sets obtained from methylation data after postprocessing for $p = 10, 25, 50$ (for $p = 10$ the largest module set does not contain genes of all smaller module sets, therefore overlaps of genes in module sets for $m = 8, 10$ are visualised)

### (2.) Methylation data analysis

Figure 6.13 shows the overlap of module genes for different parameters after the postprocessing. For distinct $\#m$, but for same $p$, modules for higher $\#m$ often contain all nodes of modules for smaller $\#m$. This appears for all $p$, except for $p = 10$. To simplify, in this case, only the genes for higher $\#m$ are shown in Figure 6.13.

For $p = 50$, the algorithm returns modules that contain all network genes. After the postprocessing, the genes of the remaining module for $p = 50$, are not contained in any other module. The remaining module contains only 4 genes.

The modules for $p = 25$ contain genes of both distinct module sets for $p = 10$ and some additional nodes.

The number of genes in all modules per $\#m$ after postprocessing is significantly smaller than for gene expression data (the maximum number of genes in methylation data is 131, whereas it is 1425 is gene expression data).

**Classifiers.** The results of the mathematical anylysis of modueles is shown in Table 6.4. The Table shows details on the modules and their performance as feature selections for all parameters $p$ and $\#m$ evaluated. Column *Genes* specifies the number of genes in all modules for the respective setting. Best values per $p$ are marked bold.

After the postprocessing explained in §4.2, the score for the modules with fewest genes for each value of $p$ is the highest. The $F1_{max}$ is close to 1 in all cases, so that the JSD has an high influence on the difference of scores.

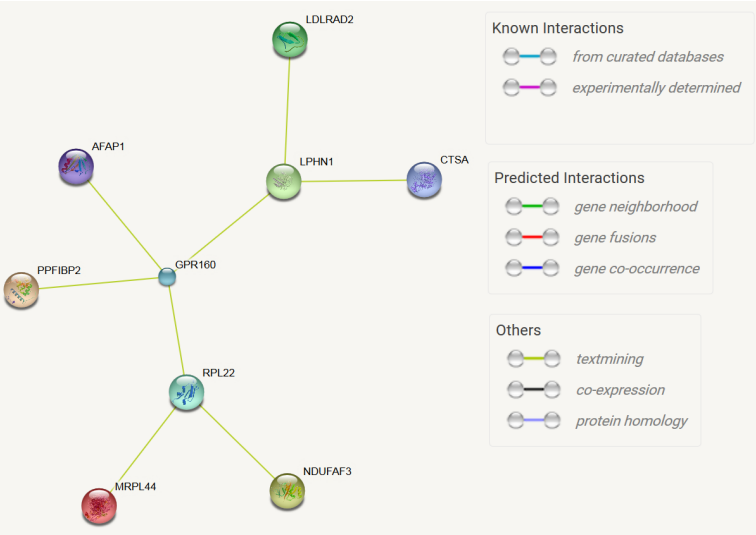
The JSD for the module sets from $p = 50$ is zero, because it contains all networks

nodes, hence their is no difference between the weight distributions of all network nodes and module nodes.

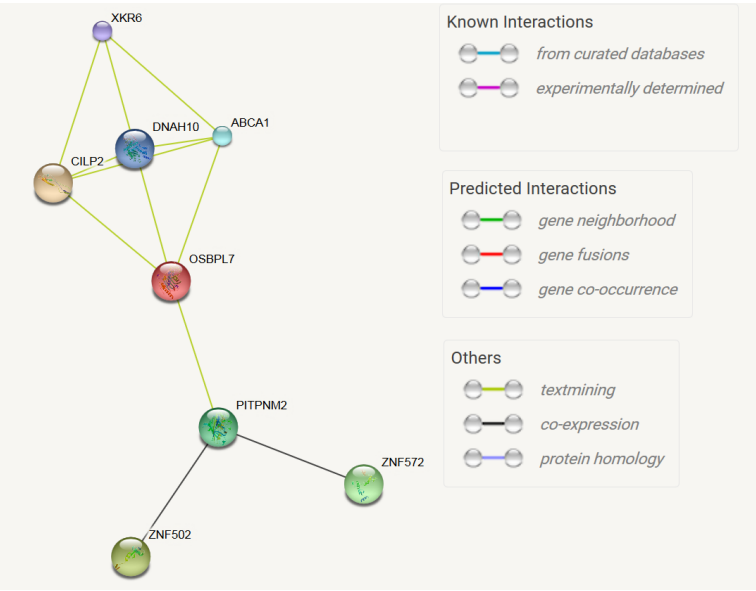
$p = 25$ and $\#m = 7$ has the highest score, after postprocessing. Figure 6.14 shows a visualisation of the modules for $p = 25$ and $\#m = 7$.

Table 6.4.: Mean F1 score over 100 testing runs. Each classifier is trained with data for the module genes returned by our algorithm for different values of $p$ and $\#m$ for methylation data before and after (*)postprocessing. After postprocessing the score for the module sets with fewest genes is the highest for each $p$ iteration. The $F1_{max}$ is close to 1 in all cases. The number left and right to the "→" in the $\#m$ of the preprocessing modules indicated which module set is altered and how many modules remain.

| Data | Parameters | | Results | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $p$ | #m | Genes | JSD | F1 score, SVM classifier | F1 score, Random forest classifier | F1 score, Elastic net classifier | Score: $JSD * F1_{max}$ |
| Methylation | 10 | 3 | 5 | 0.506 | 0.988 | 0.982 | 0.963 | 0.500 |
| | | 4 | 6 | 0.327 | 0.982 | 0.984 | 0.959 | 0.322 |
| | | 8 | 62 | **0.559** | **1** | **1** | **1** | **0.559** |
| | | 12 | 80 | 0.409 | **1** | **1** | 0.998 | 0.409 |
| | 25 | 2 | 3 | **0.566** | 0.98 | 0.972 | 0.934 | 0.555 |
| | | 4 | 5 | 0.298 | 0.985 | 0.979 | 0.932 | 0.294 |
| | | 5 | 7 | 0.205 | 0.961 | 0.983 | 0.921 | 0.202 |
| | | 7 | 24 | 0.414 | 0.998 | **1** | 0.998 | 0.414 |
| | | 12 | 89 | 0.547 | **1** | **1** | 0.998 | 0.547 |
| | | 16 | 138 | **0.566** | **1** | **1** | **1** | **0.566** |
| | 50 | 7 | 9220 | **0** | **1** | **1** | **0.974** | **0** |
| | | 10 | 9220 | **0** | **1** | **1** | **0.974** | **0** |
| | | 12 | 9220 | **0** | **1** | **1** | **0.974** | **0** |
| | | 16 | 9220 | **0** | **1** | **1** | **0.974** | **0** |
| | 10* | 3,4 → 1 | 3 | **0.636** | 0.976 | 0.979 | 0.967 | **0.623** |
| | | 8 → 2 | 59 | 0.612 | **1** | **1** | **1** | 0.612 |
| | | 12 → 2 | 67 | 0.589 | **1** | **1** | 0.998 | 0.589 |
| | 25* | 7 → 2 | 17 | **0.633** | **1** | **1** | **1** | **0.633** |
| | | 12 → 2 | 82 | 0.632 | **1** | **1** | 0.998 | 0.632 |
| | | 16 → 2 | 131 | 0.626 | **1** | **1** | **1** | 0.626 |
| | 50* | *all* → 1 | 4 | **0.558** | **0.977** | **0.98** | **0.903** | **0.547** |




(a) Module 1, 9 genes




(b) Module 2, 8 genes

Figure 6.14.: The best scoring module set from the methylation data network for $p = 25$ and $\#m = 7$ after postprocessing: genes are connected with different edge types.

**Biological analysis of best performing result.** Only the module set with the best score is evaluated ($p = 25$ and $\#m = 7$ after postprocessing). A complete list of genes in the modules is listed in the appendix. For neither of the genes of the module sets, sufficiently enriched pathways are found. This highlights one of the strength of our approach, as our methods does not require a pathway annotation.

**Module 1.** Module 1 is comprised of nine genes. For five of the nine genes, we could find publications suggesting an impact of the gene on breast cancer.

Six of the nine genes in module 1 are coding proteins, that have a function connected to the cell membrane:

- Actin is part of the cytoskeleton;
- Receptors are transmembrane proteins;
- NADH Dehydrogenase translocates protons from the matrix into the intermembrane space of the mitochondrion.

NDUFAF3 hence, has a function in both, the cell membrane and the mitochondrion. MRPL44 and RPL22 also have a function connected to the mitochondrion.

Summarising, eight of the nine proteins in module 1 can be assigned functions connected to the cell membrane and/or mitochondrion.

The relation of altered functions of the mitochondrion and cancer is well known: Warburg first proposed that cancer originated from irreversible injury to mitochondrial respiration [79].

| Gene | Full name | BRCA impact |
|---|---|---|
| AFAP1 | Actin Filament Associated Protein 1 | [21] |
| GPR160 | G Protein-Coupled Receptor 160 | [23] |
| NDUFAF3 | NADH Dehydrogenase (Ubiquinone) Complex I, Assembly Factor 3 | [48] |
| PPFIBP2 | TPRF Interacting Protein, Binding Protein 2 (Liprin Beta 2) | [64] |
| LPHN1 | Adhesion G Protein-Coupled Receptor L1 | [54] |
| LDLRAD2 | Low Density Lipoprotein Receptor Class A Domain Containing 2 | - |
| MRPL44 | Mitochondrial Ribosomal Protein L44 | - |
| RPL22 | Ribosomal Protein L22 | - |
| CTSA | Cathepsin A | - |

**Module 2.** Module 2 is comprised of eight genes. For four of them, we could find publications, suggesting an impact of the gene on breast cancer.

We cannot find a reasonable biological interpretation for module 2.

| Gene | Full name | BRCA impact |
|---|---|---|
| OSBPL7 | Oxysterol Binding Protein-Like 7 | [1] |
| PITPNM2 | Phosphatidylinositol Transfer Protein, Membrane-Associated 2 | [41] |
| ABCA1 | ATP-Binding Cassette, Sub-Family A (ABC1), Member 1 | [59] |
| CILP2 | Cartilage Intermediate Layer Protein 2 | [38] |
| ZNF572 | Zinc Finger Protein 572 | - |
| ZNF502 | Zinc Finger Protein 502 | - |
| XKR6 | XK, Kell Blood Group Complex Subunit-Related Family, Member 6 | - |
| DNAH10 | Dynein, Axonemal, Heavy Chain 10 | - |

## 6.4. Machine learning based biomarker detection for fusion data

Data can be integrated by attaching one data set to the other, which results in one long vector. For the data used in this work, this results in a matrix of 28848 features (18236 gene expression and 10612 methylation features, for 30 control + 200 case samples). This data can be analysed with methods depicted in §3, because the result matrix, only of higher dimension than in §6.3.

The performance results of the classifiers trained with single, respectively fusion data sets, with three different methods each (details on methods in §3), are shown in Table 6.5. We evaluate three different methods, so that we can better detect if the performance is based on the data or on the method. The class variable is either *healthy* or *tumour*. The performance of the classifiers trained with fusion data are better or equally good, as the single data trained classifiers.

Figure 6.15 visualises that classifiers trained with SVM and random forest tend to pick features from gene expression data, whereas for elastic net the two data sources are nearly balanced. The overlap of the 100 most important features is at maximum 10%

Table 6.5.: Mean F1 score over 100 testing runs. Each classifier is trained with data for the module genes returned by our algorithm for different values of $p$ and $\#m$ for different training data. For random forest classifiers, the fusion data classifier performs slightly better, the other classifiers perform equally perfect.

| Data type | F1 score, SVM classifier | F1 score, Random forest classifier | F1 score, Elastic net classifier |
|---|---|---|---|
| Gene expression data | 1 | 0.989 | 1 |
| Methylation data | 1 | 0.989 | 1 |
| Fusion data | 1 | 1 | 1 |

between classifiers (SVM & elastic net 10%, random forest & elastic net 2%, SVM & random forest 5%).

It is difficult to draw conclusions in terms of biomarker selection at this point. The argumentation is the same as in §6.2, for example it is not clear which features to select (this does not hold for the elastic net result, because it includes a feature selection), each feature has to be regarded separately. Moreover, the curse of dimensionality is another problem.

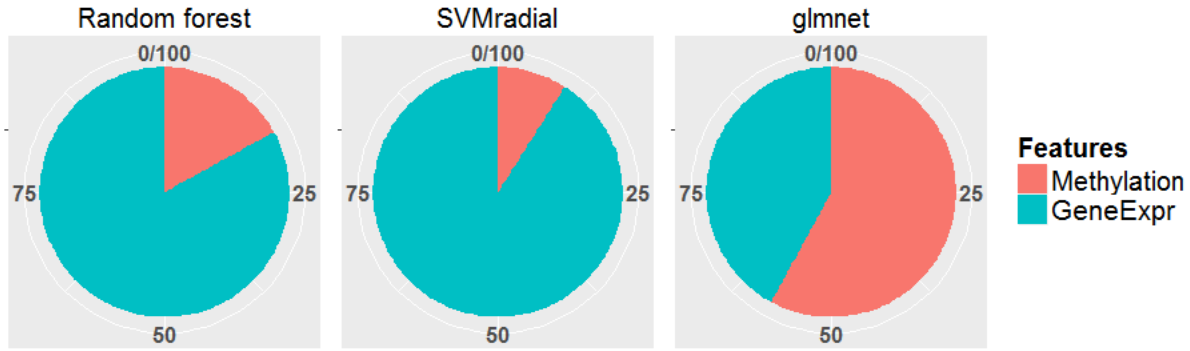



Figure 6.15.: Pie charts of 100 most important features for classifiers for fusion data; With methods random forest (l), SVM with RBF kernel (c), elastic net (r). Classifiers trained with SVM and random forest tend to pick features from gene expression data, whereas for elastic net the two data sources are nearly balanced.

## 6.5. Biomarker detection with the fusion network approach

We use the processed data and network described in §6.3, for the application of our fusion network algorithm (§5) by assigning each node a vectorial weight. Only those genes are maintained for which gene expression and methylation data are available. This results in a network of 9060 nodes and 5544788 edges.

In our algorithm, we choose Euclidean distance for the similarity measure $d$ between two nodes. We can use Euclidean distance, because node weights are both similarly distributed and minimum (0.001 and 0.003) and maximum (0.63 and 0.66) of both JSD distributions are similar. Holding times of the random walker are set to $exp(min(weight_i, wight_j))$, as describes in 5.3.

### 6.5.1 Results

Based on observation in §5.3, the minimum of $p$, which is the parameter that influences the similarity criterion, is chosen to be 10, because the results for smaller values of $p$ contain a lot of smaller weight nodes. For $p$ significantly larger than 50, $L_S$ (equation (4.6)) becomes numerically unstable. The maximum $p$ is therefore set to 50. $p = 25$ is tested as an intermediate $p$. Values for $\#m$ are determined from the spectrum of $L_S$ (equation (4.6)).

For a fixed value of $p$, modules detected for higher $\#m$ often contain all nodes belonging to modules detected for smaller $\#m$. This appears for all $p$, except for $p = 10$. To simplify considerations, in that case only the genes belonging to modules detected with higher $\#m$ are observed in Figure 6.16. All module sets contain the same 67 genes.

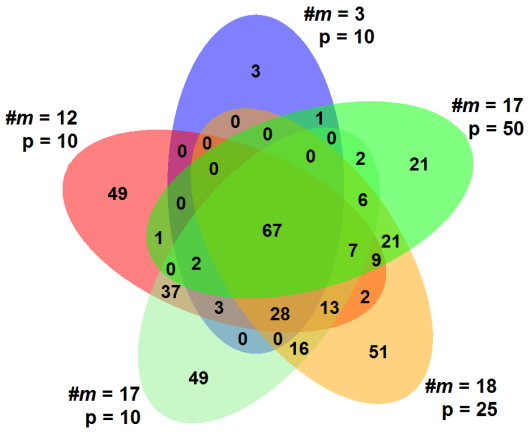
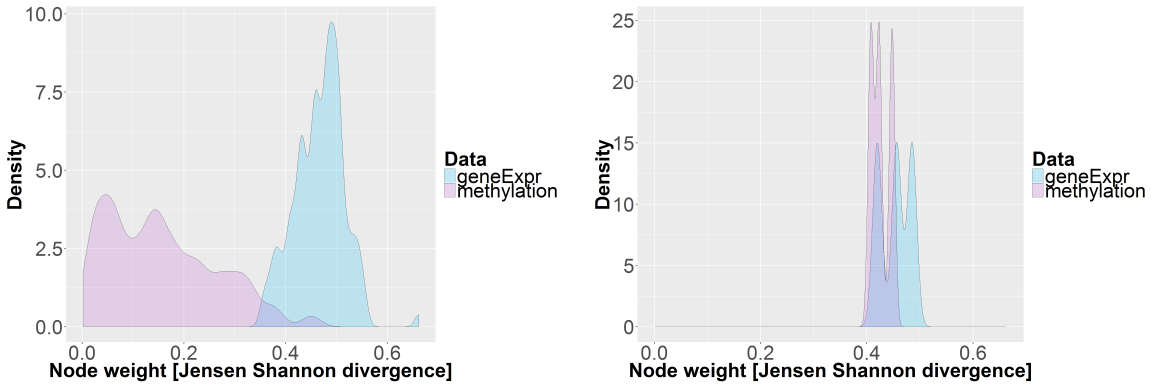



Figure 6.16.: Venn diagram of genes of the largest module sets for fusion data modules after postprocessing for $p = 10, 25, 50$ (for $p = 10$ the largest module set does not contain genes of all smaller module sets, therefore overlaps of genes in module sets for $m = 3, 10, 12$ are visualised)




(a) Module 1 (b) Module 2

Figure 6.17.: Node weight distribution for highest scoring module set in fusion data. In Module 1 the measurement for gene expression are high for all genes. In Module 2, both data measurements are high.

**Classifiers.** Table 6.6 shows details on the modules and the performance of classifiers trained with module genes returned by our algorithm, for all parameter sets evaluated. Column *Genes* specifies the number of genes in all modules for the respective setting. Best values per $p$ are marked bold.

Each result is assigned a score: $JSD * F1_{max}$, where JSD is the Jensen–Shannon divergence between the higher module weight distribution and the distribution of all network weights; $F1_{max}$ is the maximum F1 score of the three classifiers using methods SVM with RBF kernel, random forest, and elastic net.

After the postprocessing explained in §5.2, the score is the highest for each $p$ iteration when the total amount of genes in modules is the smallest. The $F1_{max}$ is close to 1 in all cases, so that the JSD has an high influence on the difference of scores. The score $JSD * F1_{max}$ is the highest for genes in modules for $p = 50^*$ and $\#m = 12$ ($^*$ := after postprocessing). For this setting two modules remain after postprocessing: Module 2 (3 genes) is high in both, gene expression and methylation JSDs. Module 1 (126 genes) is high in gene expression JSDs, and methylation JSDs spread from approximately 0 to 0.5 (Figure 6.17). Figures 6.18 and 6.19 show a visualisation of the modules for $p = 50^*$ and $\#m = 12$.

Table 6.6.: Mean F1 score over 100 testing runs. Each classifier is trained with data for the module genes returned by our algorithm for different values of $p$ and $\#m$ for fusion data before and after (*)postprocessing The number left and right to the "→" in the $\#m$ of the preprocessing modules indicated which module set is altered and how many modules remain.

| Data | Parameters | | Results | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $p$ | #m | Genes | JSD | F1 score, SVM classifier | F1 score, Random forest classifier | F1 score, Elastic net classifier | Score: $JSD * F1_{max}$ |
| Fusion | 10 | 3 | 110 | **0.481** | 0.996 | 0.974 | 0.995 | **0.479** |
| | | 12 | 400 | 0.116 | **1** | **0.999** | 0.997 | 0.116 |
| | | 17 | 386 | 0.122 | **1** | 0.996 | **1** | 0.122 |
| | 25 | 8 | 160 | **0.363** | **1** | 0.979 | **1** | **0.363** |
| | | 12 | 196 | 0.263 | **1** | 0.983 | **1** | 0.263 |
| | | 18 | 267 | 0.271 | **1** | **0.999** | 0.997 | 0.271 |
| | 50 | 5 | 7 | 0.308 | 0.959 | 0.985 | 0.964 | 0.303 |
| | | 7 | 9 | 0.145 | 0.963 | 0.977 | 0.961 | 0.142 |
| | | 12 | 144 | **0.463** | **1** | **0.995** | **1** | **0.463** |
| | | 14 | 149 | 0.437 | **1** | 0.993 | **1** | 0.437 |
| | | 17 | 165 | 0.361 | **1** | **0.995** | **1** | 0.361 |
| | 10* | 3 → 1 | 104 | **0.56** | 0.996 | 0.976 | 0.995 | **0.558** |
| | | 12 → 3 | 218 | 0.29 | **1** | 0.995 | 0.997 | 0.29 |
| | | 17 → 6 | 230 | 0.271 | **1** | **0.999** | **1** | 0.271 |
| | 25* | 8 → 2 | 149 | **0.418** | **1** | 0.982 | **1** | **0.418** |
| | | 12,18 → 7 | 220 | 0.391 | **1** | **0.996** | 0.997 | 0.391 |
| | 50* | 12 → 2 | 129 | **0.598** | **1** | **0.997** | **1** | **0.598** |
| | | 14,17 → 3 | 137 | 0.498 | **1** | 0.996 | **1** | 0.498 |

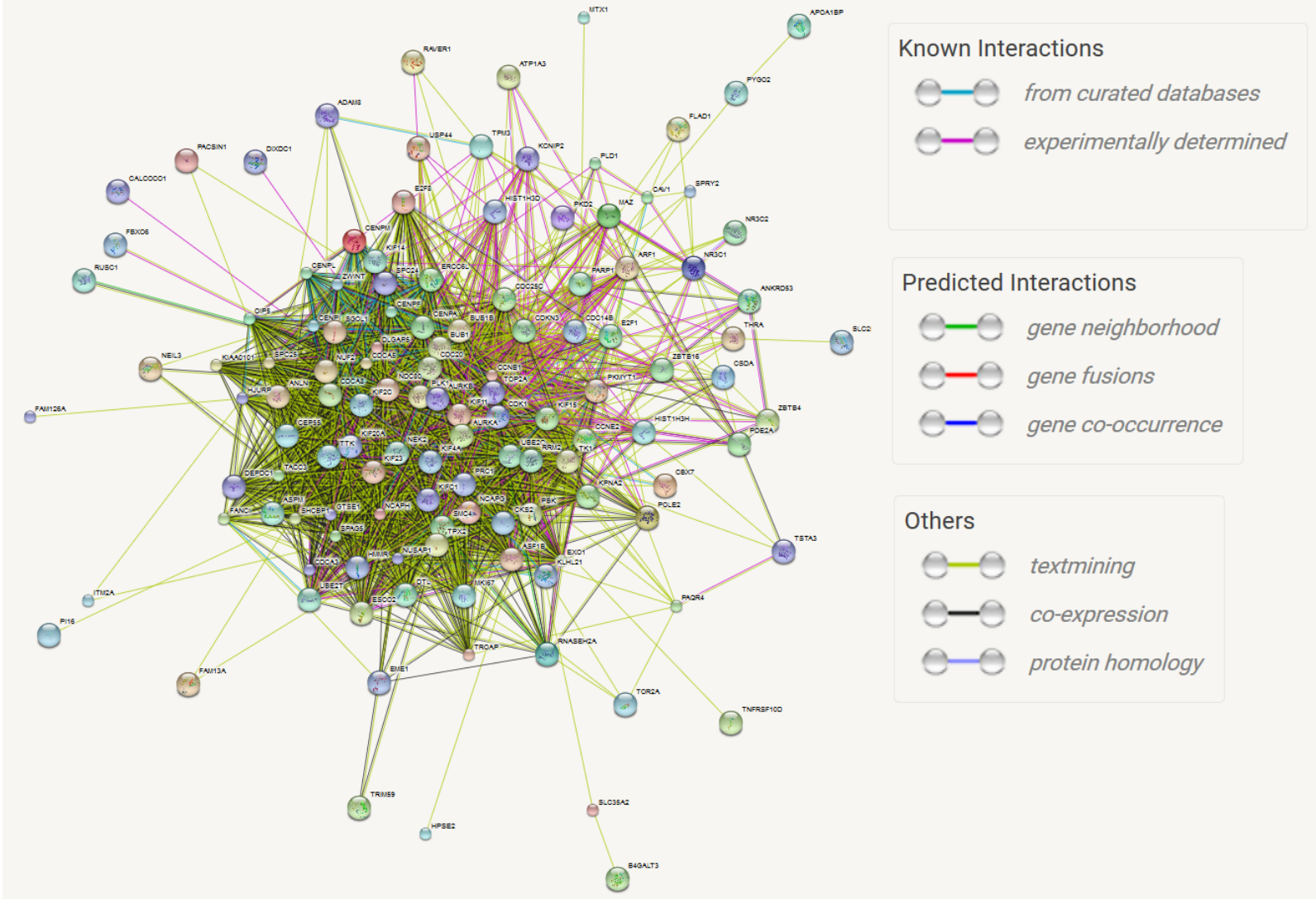



Figure 6.18.: The best scoring module set (module 1) from the gene expression data network for $p = 50$ and $\#m = 12$ after postprocessing: 126 genes connected with different edge types.

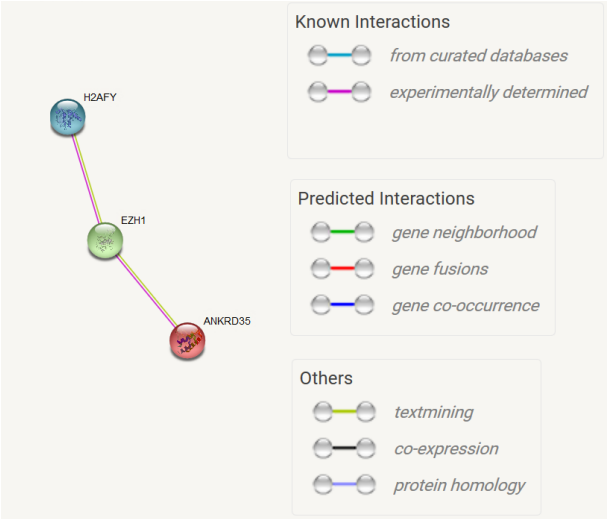



Figure 6.19.: The best scoring module set (module 2) from the fusion data network for $p = 50^*$ and $\#m = 12$ after postprocessing: 3 genes connected with different edge types.

**Biological analysis of best performing result.** To evaluate the biological meaning of module genes, KEGG [39] pathway analysis using DAVID [19] is performed. Only the module set with the best score is evaluated ($p = 25$ and $\#m = 12$ after postprocessing). A complete list of genes in the modules is listed in the appendix.

| Module | KEGG pathway | P value | Genes involved |
|---|---|---|---|
| 1 | Cell cycle | $2.86\hat{}10^{-9}$ | 12 |
| | Oocyte meiosis | $1.83\hat{}10^{-7}$ | 10 |
| | Progesterone-mediated oocyte maturation | $6.40\hat{}10^{-4}$ | 6 |
| | p53 signaling pathway | $2.23\hat{}10^{-3}$ | 5 |
| | Base excision repair | $3.34\hat{}10^{-2}$ | 3 |
| 2 | - | - | 3 |

Four of the five pathways occur for gene expression data as well (see §6.3.1). The following analysis therefore includes considerations from the biological analysis of modules in the gene expression network in §6.3.1.

**Cell cycle pathway.** Three genes from *Cell cycle* pathways that are enriched in modules for gene expression data, are not included in these *Cell cycle* pathway genes: two cyclin genes and PTTG1. However, one new gene appears: E2F1. Interestingly, E2F1 often occurs in the context of breast cancer [25, 8, 30].

**Oocyte meiosis pathway.** For genes in the fusion data modules that show an enrichment in the *Oocyte meiosis* no additional genes occur, compared to modules for gene expression data which are enriched in *Oocyte meiosis* pathway. One cyclin gene and PTTG1 are not within the module.

**Progesterone-mediated oocyte maturation pathway.** For genes in the fusion data modules that show an enrichment in the *Progesterone-mediated oocyte maturation* pathway no additional genes occur, compared to modules for gene expression data which are enriched in *Progesterone-mediated oocyte maturation* pathway. Two cyclin genes less are within the module.

**p53 signalling pathway.** For genes in the fusion data modules that shows an enrichment in the *p53 signalling* no additional genes occur, compared to modules for gene expression data which are enriched in *p53 signalling* pathway. One cyclin gene less is within the module.

**Base excision repair.** Neither of the pathways for gene expression module genes, nor any of the previously mentioned fusion pathway genes, contains genes that are associated with *Base excision repair* pathway. The three genes associated with the *Base excision repair* pathway are NEIL3, a DNA glycosylase, PARP1, a poly(ADP-ribosyl)transferase, and POLE2, a polymerase. PARP1 is associated with breast cancer [57, 58, 31].

**Remark on pathway results.** All pathways are reasonable in the context of breast cancer, as they refer processes in women, or processes relevant in cancer.

For module 1, only 19 of the 126 module genes are involved in the pathways. This, however, does not mean that the results are weak. This highlights one of the strength of our approach, as our methods does not require a pathway annotation. When applying networks as the STRING network, more information than pathways are included. The results benefit from that. The genes not involved in pathways are interacting with the pathway genes is some way, which is interesting.

**Module 2.** Module 2 is composed of genes ANKRD35, EZH1, and H2AFY.

Yoo and Hennighausen [86] describe an impact of EZH1, EZH2 and H2A on breast cancer, with relations to EZH1 and histone H2A:

'PRC1 and PCR2 are associated with chromatin condensation. PRC1 catalyses the monoubiquitylation of histone H2A and PRC2 contributes to the methylation of H3K27. PRC2 is composed of several proteins, including [...] and either EZH1 or EZH2. [...] Experimental evidence from several systems suggests that H3K27 methylation is mainly achieved by EZH2 and to a lesser extent by EZH1. [...] Excessive EZH2 concentrations have been reported as a marker of aggressive breast cancer.'

EZH2 and EZH1 are neighbours in the network. ANKRD35 is a neighbour of EZH1. We could not find evidence for a relation of ANKRD35 to breast cancer. However, other ANKRD genes are mentioned as biomarkers for (breast) cancer [51, 32].

### Remark

We observe that node weights in most modules are either high in methylation, or in gene expression JSD, as for example in our selected module 1 (Figure 6.17a). This is driven by the data, as there are few nodes that are high in both JSDs, as we can see in Figure 6.20. Accordingly the results are satisfying.

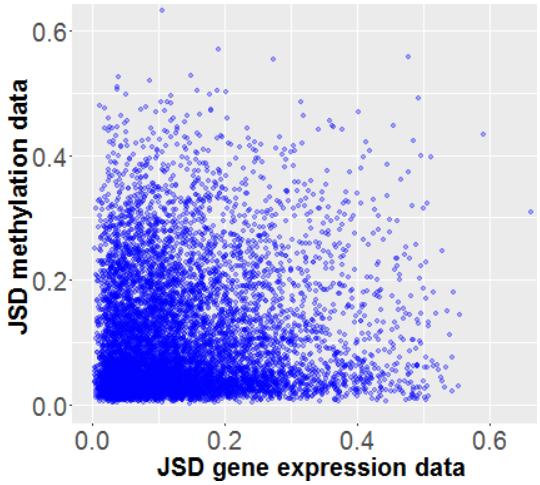



Figure 6.20.: Scatter plot of JSD for methylation data and JSD for gene expression data. Only few genes have a high JSD in methylation and gene expression. No linear trend can be seen.

# 7. Discussion

In this work we present a promising approach for biomarker detection using fusion networks.

A clear advantage of network approaches over standard ones, apart from the inclusion of a more realistic hypothesis of cell behaviour, incorporating that impacts of individually altered proteins can often be balanced, whereas alterations of interacting proteins often lead to major consequences [7, 12], is the interpretability of results. The network approach returns modules of interacting proteins. They can often be grouped to similar functions, or pathways. In the machine learning approach, each gene has to be regarded separately.

For methods that are not embedded, a number or rule has to be decided on, to choose biomarkers from all features. We have to fix such a rule in our network approach as well, but the score we apply is based on reasonable arguments, such as difference between weight distributions of resulting genes and all genes and classifier performance from according feature selections. The performance of the models from machine learning is equal or similar to perfect accuracy. We have comparably few samples for a large number of features, which brings along the curse of dimensionality and leads to overfitting. This results in good accuracy most probably, but we have to assume that we are concerned with overfitting.

To analyse the relevance of network modules from our algorithm, they are considered as feature selections. Three different training methods are applied to evaluate these feature selections: SVM, random forest, and elastic net. The feature selection derived from the network approaches reach very good results, which is interesting because they contain only a small number of features: In one module set in the methylation data results, the F1 score is 0.98 for five genes only (0.976 for the remaining 3 genes after postprocessing).

When comparing the single source network with the fusion network approach, the results of the fusion approach outperform the single source approach in terms of biological relevance and they are similarly good as one of the single source analysis results regarding the classification power.

In the network approach, the percentage of F1 scores of all experiments being exactly 1 is the highest in methylation data results (51%), followed by fusion data results (44%), and gene expression data results (10%). The biological analysis of the best scoring module set for methylation data does not reveal as extensive nor convincing results, as the best scoring fusion network module set. 41 genes in the fusion module sets, are not in the best scoring modules of the single source network approach (Figure 7.1).

The biological analysis of fusion module sets reveals one additional pathway, compared to single source module sets. Furthermore, it reveals one module of genes that are not

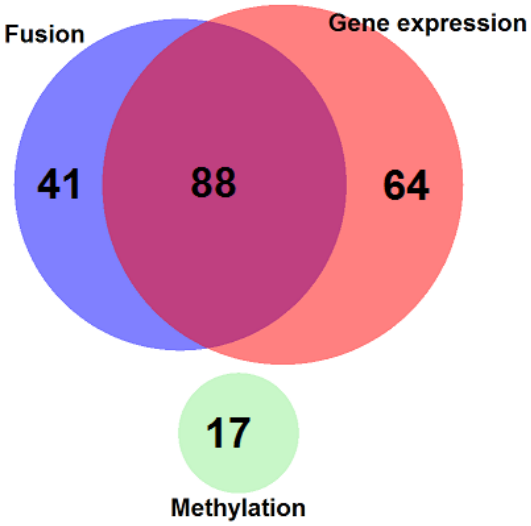



Figure 7.1.: Venn diagram of genes in best scoring module sets for all network approaches. 88 genes of modules in fusion and gene expression networks are overlapping. The gene in modules from the methylation network do not overlap with any of the other module genes.

in the best scoring modules of the single source network approach for which we find publications that indicate a role of the genes in breast cancer.

Hence, the biomarkers we detect are highly supported by breast cancer literature, biological pathways, and classification power.

# 8. Conclusion and outlook

In this thesis we propose a new network analysis approach for biomarker detection which can be applied to single source, e.g. gene expression and methylation data, as well as fusion networks. Our algorithm is a powerful approach to detect metastable structures in a fusion network with scalar or vectorial node weights.

We test our algorithm on simulated and breast cancer data. The simulation study confirms that the proposed algorithm is able to identify relevant areas in the network (single source and fusion).

We apply our algorithm (single source and fusion) and standard machine learning methods to real world biological data. The underlying network is the STRING PPI network and the data is methylation and gene expression breast cancer data from TCGA. We have discussed that deriving meaningful biomarkers using machine learning methods is questionable, because firstly, it does not incorporate biological processes. Secondly, from results of methods that are not embedded a selection of biomarkers is difficult and not based on a biological argumentation. Further, the data contains far more features than samples, which can lead to overfitting. We can see in our results, that they depend on the applied method and the overlap between methods is often small.

In contrast, the biomarkers our approach returns do not show these problems, which makes them more biologically relevant. Researchers can use these biomarkers as suggestions for e.g. new drug targets or diagnosis.

Some issues are worth further exploration. We have restricted our analysis to a static PPI network. However, PPI networks do not cover all processes in the cell. It would be interesting to study more complex networks. Further, we have restricted node weights to carry information of change. We could additionally include measurement intensity. An idea is to alter edges depending on measurements. Furthermore, it would be interesting to test various more parameters, respectively algorithm settings.

In summary, this thesis presents a new approach to extract relevant modules of fusion networks. The analyses demonstrates that it is a promising approach to identify biomarkers.

# Bibliography


[1] Aaroe, J. ; Lindahl, T. ; Dumeaux, V. ; Saebo, S. ; Tobin, D. ; Hagen, N. ; Skaane, P. ; Lonneborg, A. ; Sharma, P. ; Borresen-Dale, A.-L. : Gene expression profiling of peripheral blood cells for early detection of breast cancer. In: *Breast Cancer Res* 12 (2010), Nr. 1, S. R7

[2] Agarwala, A. ; Billheimer, J. ; Rodrigues, A. ; Risman, M. ; McCoy, M. ; Cuchel, M. ; Rader, D. : Hdl Phospholipid is Associated With Coronary Artery Disease (cad) in Individuals With Hdl-c Above the 90th Percentile Who Develop Cad. In: *Circulation* 128 (2013), Nr. 22 Supplement, S. A15126

[3] Albert, R. : Scale-free networks in cell biology. In: *Journal of cell science* 118 (2005), Nr. 21, S. 4947–4957

[4] Albert, R. ; Barabási, A.-L. : Statistical mechanics of complex networks. In: *Reviews of modern physics* 74 (2002), Nr. 1, S. 47

[5] Alon, U. : Network motifs: theory and experimental approaches. In: *Nature Reviews Genetics* 8 (2007), Nr. 6, S. 450–461

[6] Altay, G. ; Emmert-Streib, F. : Revealing differences in gene network inference algorithms on the network level by ensemble methods. In: *Bioinformatics* 26 (2010), Nr. 14, S. 1738–1744

[7] Barabási, A.-L. ; Gulbahce, N. ; Loscalzo, J. : Network medicine: a network-based approach to human disease. In: *Nature Reviews Genetics* 12 (2011), Nr. 1, S. 56–68

[8] Berteaux, N. ; Lottin, S. ; Monté, D. ; Pinte, S. ; Quatannens, B. ; Coll, J. ; Hondermarck, H. ; Curgy, J.-J. ; Dugimont, T. ; Adriaenssens, E. : H19 mRNA-like noncoding RNA promotes breast cancer cell proliferation through positive control by E2F1. In: *Journal of Biological Chemistry* 280 (2005), Nr. 33, S. 29625–29636

[9] Bird, A. : DNA methylation patterns and epigenetic memory. In: *Genes & development* 16 (2002), Nr. 1, S. 6–21

[10] Blazier, A. S. ; Papin, J. A.: Integration of expression data in genome-scale metabolic network reconstructions. In: *Front Physiol* 3 (2012), Nr. 299

[11] Breiman, L. : Random forests. In: *Machine learning* 45 (2001), Nr. 1, S. 5–32

[12] Calvano, S. E. ; Xiao, W. ; Richards, D. R. ; Felciano, R. M. ; Baker, H. V. ; Cho, R. J. ; Chen, R. O. ; Brownstein, B. H. ; Cobb, J. P. ; Tschoeke, S. K. u. a.: A network-based analysis of systemic inflammation in humans. In: *Nature* 437 (2005), Nr. 7061, S. 1032–1037

[13] Chapelle, O. : Training a support vector machine in the primal. In: *Neural computation* 19 (2007), Nr. 5, S. 1155–1178

[14] Chou, A. P. ; Chowdhury, R. ; Li, S. ; Chen, W. ; Kim, A. J. ; Piccioni, D. E. ; Selfridge, J. M. ; Mody, R. R. ; Chang, S. ; Lalezari, S. u. a.: Identification of retinol binding protein 1 promoter hypermethylation in isocitrate dehydrogenase 1 and 2 mutant gliomas. In: *Journal of the National Cancer Institute* (2012), S. djs357

[15] Colburn, W. ; DeGruttola, V. G. ; DeMets, D. L. ; Downing, G. J. ; Hoth, D. F. ; Oates, J. A. ; Peck, C. C. ; Schooley, R. T. ; Spilker, B. A. ; Woodcock, J. u. a.: Biomarkers and surrogate endpoints: Preferred definitions and conceptual framework. Biomarkers Definitions Working Group. In: *Clinical Pharmacol & Therapeutics* 69 (2001), S. 89–95

[16] Cortes, C. ; Vapnik, V. : Support-vector networks. In: *Machine learning* 20 (1995), Nr. 3, S. 273–297

[17] Coser, K. R. ; Chesnes, J. ; Hur, J. ; Ray, S. ; Isselbacher, K. J. ; Shioda, T. : Global analysis of ligand sensitivity of estrogen inducible and suppressible genes in MCF7/BUS breast cancer cells by DNA microarray. In: *Proceedings of the National Academy of Sciences* 100 (2003), Nr. 24, S. 13994–13999

[18] Dai, M. ; Wang, P. ; Boyd, A. D. ; Kostov, G. ; Athey, B. ; Jones, E. G. ; Bunney, W. E. ; Myers, R. M. ; Speed, T. P. ; Akil, H. u. a.: Evolving gene/transcript definitions significantly alter the interpretation of GeneChip data. In: *Nucleic acids research* 33 (2005), Nr. 20, S. e175–e175

[19] Dennis Jr, G. ; Sherman, B. T. ; Hosack, D. A. ; Yang, J. ; Gao, W. ; Lane, H. C. ; Lempicki, R. A. u. a.: DAVID: database for annotation, visualization, and integrated discovery. In: *Genome biol* 4 (2003), Nr. 5, S. P3

[20] Dezső, Z. ; Nikolsky, Y. ; Nikolskaya, T. ; Miller, J. ; Cherba, D. ; Webb, C. ; Bugrim, A. : Identifying disease-specific genes based on their topological significance in protein networks. In: *BMC systems biology* 3 (2009), Nr. 1, S. 1

[21] Dorfleutner, A. ; Stehlik, C. ; Zhang, J. ; Gallick, G. E. ; Flynn, D. C.: AFAP-110 is required for actin stress fiber formation and cell adhesion in MDA-MB-231 breast cancer cells. In: *Journal of cellular physiology* 213 (2007), Nr. 3, S. 740–749

[22] Du, P. ; Zhang, X. ; Huang, C.-C. ; Jafari, N. ; Kibbe, W. A. ; Hou, L. ; Lin, S. M.: Comparison of Beta-value and M-value methods for quantifying methylation levels by microarray analysis. In: *BMC bioinformatics* 11 (2010), Nr. 1, S. 587

[23] Dvorkin-Gheva, A. ; Hassell, J. A.: Identification of a novel luminal molecular subtype of breast cancer. In: *PloS one* 9 (2014), Nr. 7, S. e103514

[24] Friedman, J. ; Hastie, T. ; Tibshirani, R. : Regularization paths for generalized linear models via coordinate descent. In: *Journal of statistical software* 33 (2010), Nr. 1, S. 1

[25] Frietze, S. ; Lupien, M. ; Silver, P. A. ; Brown, M. : CARM1 regulates estrogen-stimulated breast cancer growth through up-regulation of E2F1. In: *Cancer research* 68 (2008), Nr. 1, S. 301–306

[26] Fröhlich, H. : Network based consensus gene signatures for biomarker discovery in breast cancer. In: *Plos one* 6 (2011), Nr. 10, S. e25364

[27] Ge, H. ; Walhout, A. J. ; Vidal, M. : Integrating omic information: a bridge between genomics and systems biology. In: *TRENDS in Genetics* 19 (2003), Nr. 10, S. 551–560

[28] Greenblatt, M. ; Bennett, W. ; Hollstein, M. ; Harris, C. : Mutations in the p53 tumor suppressor gene: clues to cancer etiology and molecular pathogenesis. In: *Cancer research* 54 (1994), Nr. 18, S. 4855–4878

[29] Group, H. S. C. R. u. a.: The hyperglycemia and adverse pregnancy outcome (HAPO) study. In: *International Journal of Gynecology & Obstetrics* 78 (2002), Nr. 1, S. 69–77

[30] Han, S. ; Park, K. ; Bae, B.-N. ; Kim, K. H. ; Kim, H.-J. ; Kim, Y.-D. ; Kim, H.-Y. : E2F1 expression is related with the poor survival of lymph node-positive breast cancer patients treated with fluorouracil, doxorubicin and cyclophosphamide. In: *Breast cancer research and treatment* 82 (2003), Nr. 1, S. 11–16

[31] Hastak, K. ; Alli, E. ; Ford, J. M.: Synergistic chemosensitivity of triple-negative breast cancer cell lines to poly (ADP-Ribose) polymerase inhibition, gemcitabine, and cisplatin. In: *Cancer research* 70 (2010), Nr. 20, S. 7970–7980

[32] He, Q. ; He, Q. ; Liu, X. ; Wei, Y. ; Shen, S. ; Hu, X. ; Li, Q. ; Peng, X. ; Wang, L. ; Yu, L. : Genome-wide prediction of cancer driver genes based on SNP and cancer SNV data. In: *American journal of cancer research* 4 (2014), Nr. 4, S. 394

[33] Huang, P. ; Cao, K. ; Zhao, H. : Screening of critical genes in lung adenocarcinoma via network analysis of gene expression profile. In: *Pathology & Oncology Research* 20 (2014), Nr. 4, S. 853–858

[34] Hughes, G. P.: On the mean accuracy of statistical pattern recognizers. In: *Information Theory, IEEE Transactions on* 14 (1968), Nr. 1, S. 55–63

[35] Ideker, T. ; Dutkowski, J. ; Hood, L. : Boosting signal-to-noise in complex biology: prior knowledge is power. In: *Cell* 144 (2011), Nr. 6, S. 860–863

[36] Ihaka, R. ; Gentleman, R. : R: a language for data analysis and graphics. In: *Journal of computational and graphical statistics* 5 (1996), Nr. 3, S. 299–314

[37] Illumina: *Infinium HumanMethylation450 BeadChip*. `https://www.illumina.com/content/dam/illumina-marketing/documents/products/datasheets/datasheet_humanmethylation450.pdf`

[38] Jiaqi, H. ; Morehouse, C. ; Streicher, K. ; Higgs, B. W. ; Gao, J. ; Baffa, R. ; Meggan, C. ; Boutrin, A. ; Brohawn, P. ; Zhu, W. u. a.: Altered expression of insulin receptor isoforms in breast cancer. In: *Cancer Research* 71 (2011), Nr. 8 Supplement, S. 327–327

[39] Kanehisa, M. ; Goto, S. : KEGG: kyoto encyclopedia of genes and genomes. In: *Nucleic acids research* 28 (2000), Nr. 1, S. 27–30

[40] Komurov, K. ; White, M. A. ; Ram, P. T.: Use of data-biased random walks on graphs for the retrieval of context-specific networks from genomic data. In: *PLoS Comput Biol* 6 (2010), Nr. 8, S. e1000889

[41] Krøigård, A. B. ; Larsen, M. J. ; Lænkholm, A.-V. ; Knoop, A. S. ; Jensen, J. D. ; Bak, M. ; Mollenhauer, J. ; Kruse, T. A. ; Thomassen, M. : Clonal expansion and linear genome evolution through breast cancer progression from pre-invasive stages to asynchronous metastasis. In: *Oncotarget* 6 (2015), Nr. 8, S. 5634

[42] Lee, B. B. ; Lee, E. J. ; Jung, E. H. ; Chun, H.-K. ; Chang, D. K. ; Song, S. Y. ; Park, J. ; Kim, D.-H. : Aberrant methylation of APC, MGMT, RASSF2A, and Wif-1 genes in plasma as a biomarker for early detection of colorectal cancer. In: *Clinical Cancer Research* 15 (2009), Nr. 19, S. 6185–6191

[43] Li, B.-Q. ; You, J. ; Chen, L. ; Zhang, J. ; Zhang, N. ; Li, H.-P. ; Huang, T. ; Kong, X.-Y. ; Cai, Y.-D. : Identification of lung-cancer-related genes with the shortest path approach in a protein-protein interaction network. In: *BioMed research international* 2013 (2013)

[44] Libbrecht, M. W. ; Noble, W. S.: Machine learning applications in genetics and genomics. In: *Nature Reviews Genetics* 16 (2015), Nr. 6, S. 321–332

[45] Ma, S. ; Jiang, T. ; Jiang, R. : Differential regulation enrichment analysis via the integration of transcriptional regulatory network and gene expression data. In: *Bioinformatics* 31 (2015), Nr. 4, S. 563–571

[46] McLendon, R. ; Friedman, A. ; Bigner, D. ; Van Meir, E. G. ; Brat, D. J. ; Mastrogianakis, G. M. ; Olson, J. J. ; Mikkelsen, T. ; Lehman, N. ; Aldape, K. u. a.: Comprehensive genomic characterization defines human glioblastoma genes and core pathways. In: *Nature* 455 (2008), Nr. 7216, S. 1061–1068

[47] Meijers, J. C. ; Tekelenburg, W. L. ; Bouma, B. N. ; Bertina, R. M. ; Rosendaal, F. R.: High levels of coagulation factor XI as a risk factor for venous thrombosis. In: *New England Journal of Medicine* 342 (2000), Nr. 10, S. 696–701

[48] Mitchell, N. E.: *Genome-wide DNA methylation changes during breast tumorigenesis*, University of Alabama at Birmingham, Diss., 2012

[49] Mitra, K. ; Carvunis, A.-R. ; Ramesh, S. K. ; Ideker, T. : Integrative approaches for finding modular structure in biological networks. In: *Nature Reviews Genetics* 14 (2013), Nr. 10, S. 719–732

[50] Model, F. ; Osborn, N. ; Ahlquist, D. ; Gruetzmann, R. ; Molnar, B. ; Sipos, F. ; Galamb, O. ; Pilarsky, C. ; Saeger, H.-D. ; Tulassay, Z. u. a.: Identification and validation of colorectal neoplasia–specific methylation markers for accurate classification of disease. In: *Molecular cancer research* 5 (2007), Nr. 2, S. 153–163

[51] Molloy, T. J. ; Roepman, P. ; Naume, B. ; Veer, L. J.: A prognostic gene expression profile that predicts circulating tumor cell presence in breast cancer patients. In: *PloS one* 7 (2012), Nr. 2, S. e32426

[52] Mudunuri, U. ; Che, A. ; Yi, M. ; Stephens, R. M.: bioDBnet: the biological database network. In: *Bioinformatics* 25 (2009), Nr. 4, S. 555–556

[53] Nitsch, D. ; Gonçalves, J. P. ; Ojeda, F. ; De Moor, B. ; Moreau, Y. : Candidate gene prioritization by network analysis of differential expression using machine learning approaches. In: *BMC bioinformatics* 11 (2010), Nr. 1, S. 1

[54] Obermayr, E. ; Sanchez-Cabo, F. ; Tea, M.-K. M. ; Singer, C. F. ; Krainer, M. ; Fischer, M. B. ; Sehouli, J. ; Reinthaller, A. ; Horvat, R. ; Heinze, G. u. a.: Assessment of a six gene panel for the molecular detection of circulating tumor cells in the blood of female cancer patients. In: *BMC cancer* 10 (2010), Nr. 1, S. 666

[55] Organization, W. H. u. a.: International Programme on Chemical Safety (IPCS)–Environmental Health Criteria 155: Biomarkers and risk assessment: concepts and principles. In: *International Journal of Aquatic Biology (2015)* 3 (1993), Nr. 6, S. 398–408

[56] Orrù, G. ; Pettersson-Yeo, W. ; Marquand, A. F. ; Sartori, G. ; Mechelli, A. : Using support vector machine to identify imaging biomarkers of neurological and psychiatric disease: a critical review. In: *Neuroscience & Biobehavioral Reviews* 36 (2012), Nr. 4, S. 1140–1152

[57] O'Shaughnessy, J. ; Osborne, C. ; Pippen, J. E. ; Yoffe, M. ; Patt, D. ; Rocha, C. ; Koo, I. C. ; Sherman, B. M. ; Bradley, C. : Iniparib plus chemotherapy in metastatic triple-negative breast cancer. In: *New England Journal of Medicine* 364 (2011), Nr. 3, S. 205–214

[58] Ossovskaya, V. ; Koo, I. C. ; Kaldjian, E. P. ; Alvares, C. ; Sherman, B. M.: Upregulation of poly (ADP-ribose) polymerase-1 (PARP1) in triple-negative breast cancer and other primary human tumor types. In: *Genes & cancer* 1 (2010), Nr. 8, S. 812–821

[59] Park, S. ; Shimizu, C. ; Shimoyama, T. ; Takeda, M. ; Ando, M. ; Kohno, T. ; Katsumata, N. ; Kang, Y.-K. ; Nishio, K. ; Fujiwara, Y. : Gene expression profiling of ATP-binding cassette (ABC) transporters as a predictor of the pathologic response to neoadjuvant chemotherapy in breast cancer patients. In: *Breast cancer research and treatment* 99 (2006), Nr. 1, S. 9–17

[60] Phillips, T. : The role of methylation in gene expression. In: *Nature Education* 1 (2008), Nr. 1, S. 116

[61] Reimand, J. ; Kull, M. ; Peterson, H. ; Hansen, J. ; Vilo, J. : g: Profiler a web-based toolset for functional profiling of gene lists from large-scale experiments. In: *Nucleic acids research* 35 (2007), Nr. suppl 2, S. W193–W200

[62] Roldo, C. ; Missiaglia, E. ; Hagan, J. P. ; Falconi, M. ; Capelli, P. ; Bersani, S. ; Calin, G. A. ; Volinia, S. ; Liu, C.-G. ; Scarpa, A. u. a.: MicroRNA expression abnormalities in pancreatic endocrine and acinar tumors are associated with distinctive pathologic features and clinical behavior. In: *Journal of Clinical Oncology* 24 (2006), Nr. 29, S. 4677–4684

[63] Saleem, M. ; Kamdar, M. R. ; Iqbal, A. ; Sampath, S. ; Deus, H. F. ; Ngomo, A.-C. N.: Big linked cancer data: Integrating linked tcga and pubmed. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 27 (2014), S. 34–41

[64] Sapkota, Y. ; Yasui, Y. ; Lai, R. ; Sridharan, M. ; Robson, P. J. ; Cass, C. E. ; Mackey, J. R. ; Damaraju, S. : Identification of a breast cancer susceptibility locus at 4q31. 22 using a genome-wide association study paradigm. In: *PloS one* 8 (2013), Nr. 5, S. e62550

[65] Sarich, M. : *Projected transfer operators: discretization of Markov processes in high-dimensional state spaces*, Berlin, Freie Universität Berlin, Diss., 2011, Diss., 2011

[66] Sarich, M. ; Djurdjevac, N. ; Bruckner, S. ; Conrad, T. O. ; Schütte, C. : Modularity revisited: A novel dynamics-based concept for decomposing complex networks. In: *Journal of Computational Dynamics* 1 (2014), Nr. 1, S. 191–212

[67] Siegel, R. L. ; Miller, K. D. ; Jemal, A. : Cancer statistics, 2016. In: *CA: A cancer journal for clinicians* (2015)

[68] Staff, S. ; Isola, J. ; Jumppanen, M. ; Tanner, M. : Aurora-A gene is frequently amplified in basal-like breast cancer. In: *Oncology reports* 23 (2010), Nr. 2, S. 307–312

[69] Swan, A. L. ; Mobasheri, A. ; Allaway, D. ; Liddell, S. ; Bacardit, J. : Application of machine learning to proteomics data: classification and biomarker identification in postgenomics biology. In: *Omics: a journal of integrative biology* 17 (2013), Nr. 12, S. 595–610

[70] Szklarczyk, D. ; Franceschini, A. ; Wyder, S. ; Forslund, K. ; Heller, D. ; Huerta-Cepas, J. ; Simonovic, M. ; Roth, A. ; Santos, A. ; Tsafou, K. P. u. a.: STRING v10: protein–protein interaction networks, integrated over the tree of life. In: *Nucleic acids research* (2014), S. gku1003

[71] Talwar, P. ; Silla, Y. ; Grover, S. ; Gupta, M. ; Agarwal, R. ; Kushwaha, S. ; Kukreti, R. : Genomic convergence and network analysis approach to identify candidate genes in Alzheimer's disease. In: *BMC genomics* 15 (2014), Nr. 1, S. 199

[72] TCGA: *TCGA RNAseq data*. `https://wiki.nci.nih.gov/display/TCGA/RNASeq`

[73] Teodoridis, J. M. ; Hall, J. ; Marsh, S. ; Kannall, H. D. ; Smyth, C. ; Curto, J. ; Siddiqui, N. ; Gabra, H. ; McLeod, H. L. ; Strathdee, G. u. a.: CpG island methylation of DNA damage response genes in advanced ovarian cancer. In: *Cancer research* 65 (2005), Nr. 19, S. 8961–8967

[74] Tfelt-Hansen, J. ; Kanuparthi, D. ; Chattopadhyay, N. : The emerging role of pituitary tumor transforming gene in tumorigenesis. In: *Clinical medicine & research* 4 (2006), Nr. 2, S. 130–137

[75] Vidarsdottir, L. ; Bodvarsdottir, S. K. ; Hilmarsdottir, H. ; Tryggvadottir, L. ; Eyfjord, J. E.: Breast cancer risk associated with AURKA 91T A polymorphism in relation to BRCA mutations. In: *Cancer letters* 250 (2007), Nr. 2, S. 206–212

[76] Von Luxburg, U. : A tutorial on spectral clustering. In: *Statistics and computing* 17 (2007), Nr. 4, S. 395–416

[77] Wagner, J. R. ; Busche, S. ; Ge, B. ; Kwan, T. ; Pastinen, T. ; Blanchette, M. : The relationship between DNA methylation, genetic and expression inter-individual variation in untransformed human fibroblasts. In: *Genome Biol* 15 (2014), Nr. 2, S. R37

[78] Wang, Z. ; Gerstein, M. ; Snyder, M. : RNA-Seq: a revolutionary tool for transcriptomics. In: *Nature Reviews Genetics* 10 (2009), Nr. 1, S. 57–63

[79] Warburg, O. u. a.: On the origin of cancer cells. In: *Science* 123 (1956), Nr. 3191, S. 309–314

[80] (WHO), W. H. O. u. a.: *International Programme on Chemical Safety. Biomarkers in Risk Assessment: Validity and Validation, 2001.* 2015

[81] Wong, E. ; Baur, B. ; Quader, S. ; Huang, C.-H. : Biological network motif detection: principles and practice. In: *Briefings in bioinformatics* (2011), S. bbr033

[82] Worsham, M. J. ; Chitale, D. ; Chen, K. M. ; Divine, G. : The methylome of ER-negative tumors is predominantly hypermethylated. In: *Cancer Research* 74 (2014), Nr. 19 Supplement, S. 1372–1372

[83] Wu, C. ; Zhu, J. ; Zhang, X. : Integrating gene expression and protein-protein interaction network to prioritize cancer-associated genes. In: *BMC bioinformatics* 13 (2012), Nr. 1, S. 182

[84] Xia, J. ; Benner, M. J. ; Hancock, R. E.: NetworkAnalyst-integrative approaches for protein–protein interaction network analysis and visual exploration. In: *Nucleic acids research* (2014), S. gku443

[85] Yang, R. ; Daigle, B. J. ; Petzold, L. R. ; Doyle, F. J.: Core module biomarker identification with network exploration for breast cancer metastasis. In: *BMC bioinformatics* 13 (2012), Nr. 1, S. 12

[86] Yoo, K. H. ; Hennighausen, L. : EZH2 methyltransferase and H3K27 methylation in breast cancer. In: *Int J Biol Sci* 8 (2012), Nr. 1, S. 59–65

[87] Zhang, C. ; Zhao, H. ; Li, J. ; Liu, H. ; Wang, F. ; Wei, Y. ; Su, J. ; Zhang, D. ; Liu, T. ; Zhang, Y. : The identification of specific methylation patterns across different cancers. In: *PloS one* 10 (2015), Nr. 3, S. e0120361

[88] Zheng, F. ; Yue, C. ; Li, G. ; He, B. ; Cheng, W. ; Wang, X. ; Yan, M. ; Long, Z. ; Qiu, W. ; Yuan, Z. u. a.: Nuclear AURKA acquires kinase-independent transactivating function to enhance breast cancer stem cell phenotype. In: *Nature communications* 7 (2016)

[89] Zhu, J. ; Qin, Y. ; Liu, T. ; Wang, J. ; Zheng, X. : Prioritization of candidate disease genes by topological similarity between disease and protein diffusion profiles. In: *BMC bioinformatics* 14 (2013), Nr. Suppl 5, S. S5

[90] Zou, H. ; Hastie, T. : Regularization and variable selection via the elastic net. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67 (2005), Nr. 2, S. 301–320

# A. Appendix

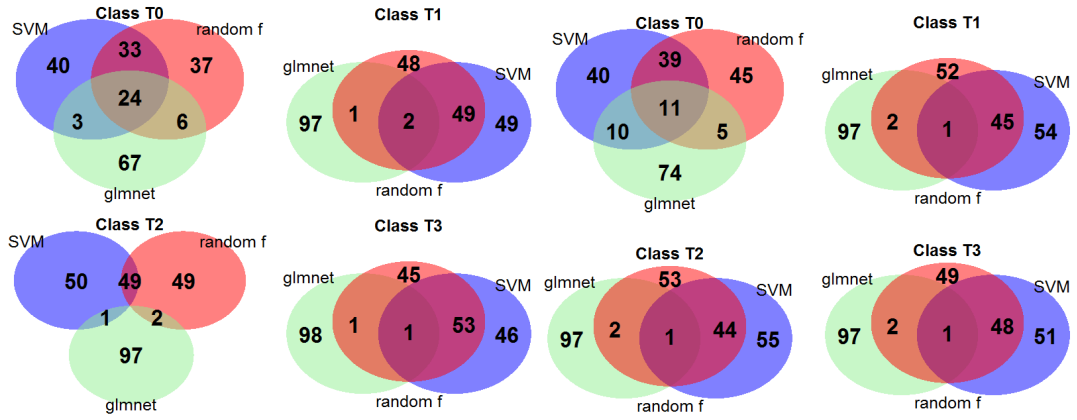
## Classifier training with biological data and four classes

This section contains classifier results, comparable to results in §6.2 and §6.4. In this section, we use tumour stages T1, T2, T3, and control label as response in the classification.

### 1.0.1 All features

Table A.1.: Overall mean F1 score over 100 testing runs for classifiers trained with single and fusion data with classes: Control, T1, T2, T3

| Data selection | Data type | F1 score, SVM classifier | F1 score, Random forest classifier | F1 score, Elastic net classifier |
|---|---|---|---|---|
| All features | Gene expression data | 0.828 | 0.839 | 0.677 |
| | Methylation data | 0.849 | 0.823 | 0.826 |
| | Fusion data | 0.849 | 0.805 | 0.790 |
| Top 100 JSD | Gene expression data | 0.683 | 0.818 | 0.627 |
| | Methylation data | 0.771 | 0.823 | 0.752 |
| | Fusion data | 0.737 | 0.836 | 0.705 |




(a) Methylation data (b) Gene expression data

Figure A.1.: Venn diagram of top 100 features separated by class for methylation data (a) and gene expression data (b) classifiers trained with random forest, SVM with RBF kernel, elastic net

### 1.0.2 Module features

Table A.2.: Mean F1 score over 100 runs for different training data with classes: Control, T1, T2, T3; Before, and after (*)postprocessing

| Data | $p$ | #m | Genes | JSD | SVM [SVMRADIAL] | Random forest [RF] | Elastic net [GLMNET] | $JSD * F1_{max}$ |
|---|---|---|---|---|---|---|---|---|
| Gene expression | 10 | 6 | 160 | **0.562** | **0.756** | **0.824** | **0.779** | **0.463** |
| | | 8 | 165 | 0.523 | 0.725 | 0.822 | 0.696 | 0.403 |
| | | 14 | 283 | 0.448 | 0.651 | 0.82 | 0.665 | 0.400 |
| | 25 | 7 | 1432 | 0.497 | **0.793** | 0.826 | 0.826 | 0.411 |
| | | 9 | 1438 | **0.492** | 0.787 | **0.83** | **0.837** | **0.412** |
| | | 11 | 1447 | 0.484 | 0.792 | **0.83** | 0.833 | 0.403 |
| | | 18 | 1460 | 0.472 | 0.792 | 0.822 | 0.84 | 0.396 |
| | 50 | 7 | 1593 | **0.399** | 0.799 | 0.821 | 0.709 | **0.328** |
| | | 9 | 1596 | 0.395 | 0.799 | 0.828 | 0.709 | 0.327 |
| | | 11 | 1607 | 0.383 | 0.802 | 0.835 | 0.773 | 0.319 |
| | | 18 | 2808 | 0.280 | **0.822** | **0.837** | **0.787** | 0.234 |
| | 10* | $6 \rightarrow 1$ | 152 | **0.599** | **0.779** | 0.829 | **0.766** | **0.497** |
| | 25* | $all \rightarrow 2$ | 1423 | **0.507** | **0.793** | **0.843** | **0.82** | **0.427** |
| | 50* | $all \rightarrow 3$ | 838 | **0.555** | **0.758** | **0.832** | **0.774** | **0.462** |

*Continued on next page*

**Table A.2 – continued from previous page**

| Data | $p$ | #m | Genes | JSD | SVM [SVMRADIAL] | Random forest [RF] | Elastic net [GLMNET] | $JSD * F1_{max}$ |
|---|---|---|---|---|---|---|---|---|
| Methylation | 10 | 3 | 6 | 0.263 | 0.674 | 0.789 | 0.569 | 0.208 |
| | | 7 | 49 | 0.521 | 0.759 | 0.8 | 0.713 | 0.417 |
| | | 8 | 65 | **0.543** | 0.755 | **0.817** | 0.732 | **0.444** |
| | | 18 | 161 | 0.509 | **0.794** | 0.812 | **0.809** | 0.413 |
| | 25 | 6 | 453 | **0.545** | 0.783 | 0.825 | 0.717 | **0.450** |
| | | 12 | 491 | 0.476 | 0.781 | 0.807 | 0.813 | 0.387 |
| | | 14 | 498 | 0.459 | 0.782 | 0.801 | **0.814** | 0.374 |
| | | 15 | 502 | 0.454 | 0.782 | 0.821 | **0.814** | 0.373 |
| | | 17 | 914 | 0.307 | **0.817** | **0.823** | 0.73 | 0.253 |
| | 50 | 12 | 846 | **0.490** | 0.796 | 0.83 | 0.732 | **0.407** |
| | | 15 | 2091 | 0.355 | **0.817** | **0.832** | **0.805** | 0.295 |
| | | 17 | 2097 | 0.350 | **0.817** | 0.821 | **0.805** | 0.287 |
| | 10* | 3,4 → 1 | 3 | **0.636** | 0.624 | 0.794 | 0.564 | **0.505** |
| | | 8 → 2 | 59 | 0.612 | 0.754 | 0.81 | 0.729 | 0.496 |
| | | 12 → 2 | 67 | 0.589 | **0.804** | **0.826** | **0.817** | 0.487 |
| | 25* | 7 → 2 | 17 | **0.633** | **0.776** | **0.8** | **0.703** | **0.506** |
| | 50* | *all* → 1 | 4 | **0.558** | **0.796** | 0.816 | **0.732** | **0.480** |
| Fusion | 10 | 3 | 110 | **0.481** | 0.781 | 0.809 | 0.771 | **0.389** |
| | | 12 | 400 | 0.116 | 0.793 | 0.815 | 0.794 | 0.095 |
| | | 17 | 386 | 0.122 | **0.798** | **0.821** | **0.804** | 0.100 |
| | 25 | 8 | 160 | **0.363** | 0.777 | 0.805 | **0.833** | **0.302** |
| | | 12 | 196 | 0.263 | 0.767 | 0.815 | 0.822 | 0.212 |
| | | 18 | 267 | 0.271 | **0.796** | **0.821** | 0.82 | 0.222 |
| | 50 | 5 | 7 | 0.308 | 0.685 | 0.816 | 0.598 | 0.251 |
| | | 7 | 9 | 0.145 | 0.683 | 0.796 | 0.575 | 0.115 |
| | | 12 | 144 | **0.463** | **0.784** | **0.839** | 0.777 | **0.388** |
| | | 14 | 149 | 0.437 | 0.778 | 0.824 | **0.811** | 0.367 |
| | | 17 | 165 | 0.361 | 0.772 | 0.835 | 0.806 | 0.301 |
| | 10* | 3 → 1 | 104 | **0.560** | 0.768 | 0.816 | 0.772 | **0.457** |
| | | 12 → 3 | 218 | 0.290 | **0.788** | **0.795** | 0.795 | 0.231 |
| | | 17 → 6 | 230 | 0.271 | 0.769 | **0.828** | 0.691 | 0.224 |
| | 25* | 8 → 2 | 149 | **0.418** | 0.77 | 0.802 | 0.825 | **0.397** |
| | | 12,18 → 7 | 220 | 0.391 | **0.774** | **0.825** | **0.834** | 0.326 |
| | 50* | 12 → 2 | 129 | **0.598** | **0.774** | **0.827** | 0.744 | **0.494** |
| | | 14,17 → 3 | 137 | 0.498 | **0.774** | 0.826 | **0.762** | 0.411 |

# Genes in best scoring modules

## 1.0.3 Genes in best scoring module for gene expression data

In this section we present all the genes in the modules detected by our algorithm for breast cancer data. Compare §6.3.1 and §6.5.1.

### Module 1

CKS2, GTSE1, HIST1H3D, HIST1H3H, KIF11, KIF4A, RABIF, RNASEH2A, UBE2T, ADAMTS5, ANGPTL7, CACHD1, FIGF, LEP, PLA2G4A, SH3BGRL2, ANLN, ARF1, ASF1B, ASPM, AURKA, AURKB, BUB1, BUB1B, CALCOCO1, CASC5, CBX7, CCNA2, CCNB1, CCNB2, CCNE2, CCNF, CDC14B, CDC20, CDC25C, CDCA3, CDCA5, CDCA8, CDK1, CDKN3, CENPA, CENPE, CENPF, CENPI, CENPL, CENPM, CEP55, CKAP2L, CLSPN, DEPDC1, DEPDC1B, DLGAP5, DTL, E2F7, E2F8, EME1, ERCC6L, ESCO2, EXO1, EZH1, FAM126A, FAM83D, FANCI, FOXM1, GRAMD1A, H2AFY, HJURP, HMMR, ITM2A, KCNIP2, KIAA0101, KIF14, KIF20A, KIF23, KIF2C, LAGE3, LMNB1, MELK, MKI67, MYBL2, NACC1, NCAPG, NCAPH, NDC80, NEIL3, NEK2, NUF2, NUSAP1, OIP5, PAQR4, PBK, PKMYT1, PLK1, PRC1, PRR11, PSMC4, PTTG1, RAD51AP1, RAD54L, RRM2, SGOL1, SKA3, SMC4, SPAG5, SPC24, SPRY2, TACC3, TK1, TOP2A, TPX2, TRIM59, TRIP13, TROAP, TTK, UBE2C, ZWINT, ANKRD29, ARHGAP20, CDCA2, CHL1, CNN1, COL10A1, COL11A1, FAM111B, FHL1, INHBA, IQGAP3, ITIH2, KLHL29, LIMS2, LRIG3, MATN2, MMP13, MYOC, SMYD1, TTC28, UBE2S, WISP1, FAM72A, POC1A, RERGL, ARHGAP11A, CDC2, FAM72D, KIF15, KIFC1, MMP11, MYH11, PPAPDC1A, PRAME, SPC25, BTBD14B

## 1.0.4 Genes in best scoring module for methylation data

### Module 1

MRPL44, AFAP1, CTSA, GPR160, LDLRAD2, NDUFAF3, PPFIBP2, RPL22, LPHN1

### Module 2

ZNF572, ABCA1, CILP2, OSBPL7, PITPNM2, XKR6, DNAH10, ZNF502

## 1.0.5 Genes in best scoring module for fusion data

### Module 1

CKS2, E2F1, GTSE1, HIST1H3D, HIST1H3H, KIF11, KIF4A, RNASEH2A, TNFRSF10D, UBE2T, ADAM8, ANKRD53, ANLN, APOA1BP, ARF1, ASF1B, ASPM, ATP1A3, AURKA, AURKB, B4GALT3, BUB1, BUB1B, CALCOCO1, CAV1, CBX7, CCNB1, CCNE2, CDC14B, CDC20, CDC25C, CDCA3, CDCA5, CDCA8, CDK1, CDKN3, CENPA, CENPF, CENPI, CENPL, CENPM, CEP55, DEPDC1,

DIXDC1, DLGAP5, DTL, E2F8, EME1, ERCC6L, ESCO2, EXO1, FAM126A, FAM13A, FANCI, FBXO6, FLAD1, HJURP, HMMR, HPSE2, ITM2A, KCNIP2, KIAA0101, KIF14, KIF20A, KIF23, KIF2C, KLHL21, KPNA2, MAZ, MKI67, MTX1, NCAPG, NCAPH, NDC80, NEIL3, NEK2, NR3C1, NR3C2, NUF2, NUSAP1, OIP5, PACSIN1, PAQR4, PARP1, PBK, PDE2A, PI16, PKD2, PKMYT1, PLD1, PLK1, POLE2, PRC1, PYGO2, RAVER1, RRM2, RUSC1, SGOL1, SHCBP1, SLC25A39, SLC35A2, SMC4, SPAG5, SPC24, SPRY2, TACC3, THRA, TK1, TOP2A, TOR2A, TPM3, TPX2, TRIM59, TROAP, TTK, UBE2C, USP44, ZBTB16, ZBTB4, ZWINT, CDC2, CSDA, KIF15, KIFC1, SPC25, TSTA3

**Module 2**

ANKRD35, EZH1, H2AFY

# Algorithm code

This section contains the code of the algorithm presented in §4 and §5.

### 1.0.6 Main function

```
%% initialize network and make generator
function [simmat] = mainFusion(patterns,ps)
%there must be a folder called like NetworkName in which there
    is a network called NetworkName _A and node weights called
    NetworkName _VECTCOLORS
for j=1:length(patterns)
    pattern = patterns{j}
    d = dir(pwd);
    isub = [d(:).isdir];
    nameFolds = {d(isub).name}';
    idx=~cellfun('isempty',regexp(nameFolds,pattern));
    NetworkNames = nameFolds(idx)
        for i=1:length(NetworkNames)
            NetworkName=NetworkNames{i}
            for p=ps
                disp([NetworkName,'; p =  ',num2str(p)]);
                theta = 2;
                sparseAdjacencyName = strcat(pwd, '\',
                    NetworkName, '\', NetworkName,'___','A.txt')
                    ;
                colorMatrixName      = strcat(pwd, '\',
                    NetworkName, '\', NetworkName,'___','
                    VECTCOLORS.txt');
```

```
18
19                  % load data
20                  % load JCDnetwork.mat
21                  sp = load(sparseAdjacencyName);
22                  colors0 = load(colorMatrixName);
23                  %
24                  colors = colors0;
25                  colors = abs(colors);
26
27                  % assemble adjacency matrix
28                  A = spconvert(sp);
29                  n = size(A,2);
30                  clear sp;
31
32
33                  % make the generator
34                  disp('Making time-continuous RW ...   ');
35                  [L,mu,simmat] = makeLnew(A, colors, p);
36
37                  % plot eigenvalues to decide how many clusters
                        it should be searched for
38                  fig = figure(2);
39                  num = 20;
40                  sL = sort(real(eigs(L,num,'lr')),'descend');
41                  plot(1:num,sL(1:num),'r+','LineWidth',2)
42                  title('Spectrum of L');
43                  filename = ['C:\Users\monarams\work\
                        MasterThesis\sp.' NetworkName '.' 'p'
                        num2str(p) '.png'];
44                  filename2 = [pwd '\' NetworkName '\'  'p'
                        num2str(p) '\Spectrum.' NetworkName '.' 'p'
                        num2str(p) '.png'];
45                  print(fig,'-dpng',filename)
46                  print(fig,'-dpng',filename2)
47
48                  %% do the clustering
49
50                  num_clusters_all = input('Give a number of
                        modules: '); %cl;
51
52                  for num_clusters=num_clusters_all
53                      % clustering
54                      disp(['Fuzzy clustering for number of
                            clusters =  ',num2str(num_clusters)]);
```

```
55                          [cores,clusters,q,m] = mht_cluster_plain(L,
                                num_clusters,theta);
56
57                          x=[];
58                          cluster_k=[];
59                          for k=1:num_clusters
60                              cluster_k = [cluster_k;find(clusters==k
                                    )];
61                              x = [x;repmat(k,length(find(clusters==k
                                    )),1)];
62                          end
63
64                          T = table(cluster_k, x);
65                          filename = [pwd '\' NetworkName '\' 'p'
                                num2str(p) '\' NetworkName '.' num2str(
                                num_clusters) 'p' num2str(p) '.txt'];
66                          writetable(T, filename);
67                          close all
68                      end
69                  end
70              end
71          end
72      end
```

### 1.0.7 Generator

```
1   function [L,L0,simmat] = makeLnew(A,colors,p)
2
3   type = 6;
4
5   n = size(A,2);
6   i = 1:n;
7
8   % initializations
9   d = diag(sum(A,2)); % degrees
10  color_type = 'vector';
11  switch color_type
12      case 'scalar'
13          colmat = abs(sparse(i,i,colors,n,n)*A - A*sparse(i,i,
                colors,n,n));
14          simmat = sparse(A.*exp(-p*colmat)); % similarity matrix
15          meancols = colors; % no averaging of colors needed
16      case 'vector'
17          % if wrong size, transpose
```

```
18          col_m = size(colors,2);
19          if col_m == length(A)
20              colors = colors';
21          end
22          colmat = squareform(pdist(colors,'euclidean'));
23          simmat = sparse(A.*exp(-p*colmat));
24          disp(['meancols = min(colors,[],2)'])
25          meancols = (min(colors,[],2));
26
27  end
28
29
30      % L0: times~exp(median(weights)), jumpprobs~similarities
31      % L: times~exp(weights), jumpprobs~similarities
32      Jp = sparse(i,i,1./(sum(simmat,2)),n,n) * simmat;
33      L0 = (Jp - speye(n,n))/exp(median(meancols));
34      hold_times = exp(meancols);
35      Dc = sparse(i,i,1./hold_times,n,n);
36      L = Dc*(-speye(n,n)+ Jp);
37
38
39  end % end function
```

## Clustering

```
1   function [cores,clusters,commitors,m] = mht_cluster_plain(L,
        num_cluster,theta)
2   % function [cores,clusters,commitors,m] = mht_cluster_plain(L,
        num_cluster,theta)
3   %
4   % Identifies clusters (metastable sets) of the Markov jump
        process given
5   % by the generator L based on mean first hitting times. The
        algorithm
6   % iterates the following steps:
7   % 1) Identify new core as the node where the minimal expected
        hitting time
8   % to the other clusters is maximal
9   % 2) Identify nodes which belong to the cluster defined by the
        new core as
10  % positive statistical outliers of the commitor function values
        .
11  %
12  % Input:  L:              n x n generator
```

```
13 %          num_cluster:   number of clusters to find
14 %          theta:         poitive scalar, multiplier of the
   standard
15 %                         deviation in the thresholding process
16 % Output: cores:          num_cluster x 1 array of core indices
17 %          clusters:      n x 1 array from {0,1,2,..,num_cluster
   }^n
18 %                         indicating the affiliation to the
   clusters
19 %          commitors:     (optional output) n x num_clusters
   array of
20 %                         commitor values to the cores
21 %          m:             n x num_cluster array of mfpt vectors
   to the
22 %                         clusters
23
24 % initialization
25 n = length(L);
26 clusters = zeros(n,1);
27
28 % random starting point
29 curr = ceil(n*rand());
30 % fprintf('Random starting node: %d\n\n',curr);
31
32 % find 1st core
33 cores = zeros(num_cluster,1);
34 m = mfpt(L,curr);
35 % choose node which takes the longest
36 % to get to curr
37 [~,curr] = max(m); % figure(20); plot(m,'.-');
38 clusters(curr) = 1;
39 cores(1) = curr;
40
41 % find 2nd core
42 m = mfpt(L,curr);
43
44 % choose node which takes the longest to get to curr
45 [~,curr] = max(m); % figure(21); plot(m,'.-');
46 clusters(curr) = 2;
47 cores(2) = curr;
48
49 % build the two clusters:
50 % construct right hand side for commitor computation (here the
   clusters are
```

```
% just the cores yet)
CCs = zeros(n,2);
for k=1:2
    CCs(cores(k),k) = 1;
end
% compute commitors
q = full(compute_committors_gen(L,CCs));
commitors{1} = q;
% set first 2 clusters as positive outliers
for k=1:2
    disp(['computing cluster ' num2str(k)])
    % get rid of big outliers (0 & 1), which are unambiguously
       identifyable
    statbase = find_statbase(q(:,k));
    % assign new cluster as nodes with positive statistical
       outlier
    % commitor values
    clusters(q(:,k) >= min(max(median(q(:,k)),10*eps) + ...
        theta*std(q(statbase,k)),1-10*eps)) = k;
    if std(q(statbase,k)) < 10*eps
        fprintf(['Standard deviation near machine precision, '
            ,...
            'results may be inaccurate\n']);
    end
end

% mean first hitting time vectors of the clusters
m = zeros(n,num_cluster-1);
% set the first one
rest = ~(clusters==1);
m(rest,1) = -L(rest,rest)\ones(nnz(rest),1);

% loop to find clusters number 3,...,num_cluster
for k=3:num_cluster
    disp(['computing cluster ' num2str(k)])
    % mean first hitting times to last cluster identified
    rest = ~(clusters == k-1);
    m(rest,k-1) = -L(rest,rest)\ones(nnz(rest),1);

    % find next core: where hitting time is maximal
    [~,curr] = max( min(m(:,1:k-1),[],2) );
    % assign core
    cores(k) = curr;
    clusters(curr) = k;
```

```
92
93      % assign nodes to this cluster
94      % construct right hand side for commitor computation
95      CCs = zeros(n,k);
96      for kk=1:k
97          % commitors of nodes
98          CCs(cores(kk),kk) = 1;
99      end
100     % compute commitors
101     q = full(compute_committors_gen(L,CCs));
102     commitors{k-1} = q;
103     % get rid of big outliers (0 & 1), which are unambiguously
104     % identifyable
105     statbase = find_statbase(q(:,k));
106     % assign new cluster as nodes with positive statistical
            outlier
107     % commitor values
108     clusters(q(:,k) >= min(max(median(q(:,k)),10*eps) + ...
109         theta*std(q(statbase,k)),1-10*eps)) = k;
110     if std(q(statbase,k)) < 10*eps
111         fprintf(['Standard deviation near machine precision, '
                ,...
112             'results may be inaccurate\n']);
113     end
114
115     % display
116     flag_disp = 0;
117     if flag_disp
118         for kk=1:k
119             cluster_k = find(clusters==kk)';
120             if length(cluster_k)<20,
121                 fprintf('Cluster %d:\n\n',kk);
122                 fprintf('    ');
123                 fprintf('%d ',cluster_k);
124                 fprintf('\n\n');
125             else
126                 fprintf(['Cluster %d too large to be displayed.
                        ',...
127                     'First 20 elements:\n\n'],kk);
128                 fprintf('    ');
129                 fprintf('%d ',cluster_k(1:20));
130                 fprintf('\n\n');
131             end
132         end
```

```
133     end
134 end
135
136 % make sure the cores are always in the clusters
137 for k=1:num_cluster
138     clusters(cores(k)) = k;
139 end
140
141 if nargout >= 4
142     rest = ~(clusters == num_cluster);
143     if any(clusters == num_cluster)
144         m(rest,num_cluster) = -L(rest,rest)\ones(nnz(rest),1);
145     else
146         m(:,num_cluster) = mfpt(L,cores(num_cluster));
147     end
148 end
149
150
151 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
152 function statbase = find_statbase(q)
153 % Function to compute the indices of q, in a consistent way,
    for which the
154 % statistical anaysis is carried out.
155
156 % get rid of big outliers (0 & 1), which are unambiguously
    identifyable
157 statbase = (q>0) & (q<1) & (q>2*median(q)-1);
158 % get rid of other big (negative) outliers (probably not
    necessary)
159 % statbase = logical(statbase.*( q > median(q) - std(q(statbase
    ) )));
```

**Compute committors**

```
1 function q = compute_committors_gen(L,C)
2 C = sparse(C);
3 Cs = sum(C,2);
4 d = -L*C;
5 Sk = L(Cs==0, Cs==0);
6 d = d(Cs==0,:);
7 f = Sk\d;
8 q = C;
9 q(Cs==0,:) = f;
```

**Find first two cores**

```
function m = mfpt(L,set)
    n=length(L(1,:));
    ind=1:n;
    sel = ones(n,1);
    sel(set)=0;
    ind=ind(sel==1);
    S = L(ind,ind);
    d = - ones(length(ind),1);
    f = S\d;
    m=zeros(n,1);
    m(ind) = f;
end
```