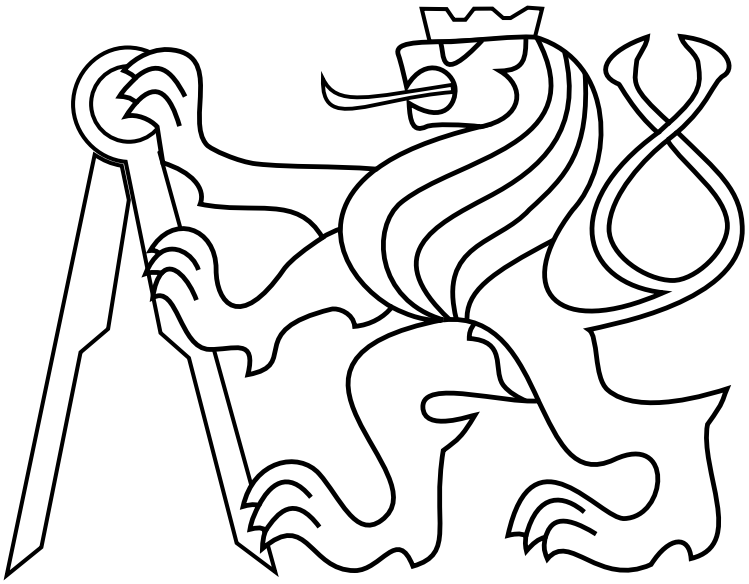
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF ELECTRICAL ENGINEERING
Department of Computer Science and Engineering

BACHELOR THESIS



Filip Masri

Feature network regularized SVM omics classifier

Supervisor: doc. Ing. Jiří Kléma, Ph.D.
Prague, 2015

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science and Engineering

BACHELOR PROJECT ASSIGNMENT

Student: **Filip Masri**

Study programme: Software Engineering and Management
Specialisation: Software Engineering

Title of Bachelor Project: **Feature network regularized SVM omics classifier**

Guidelines:

1. Get familiar with SVM (support vector machines). Focus on the aspects of their regularization.
2. Learn about the nature of omics (genomics, proteomics or metabolomics) data integrating heterogeneous features. Assume that prior knowledge on their interactions is available, however, this knowledge does not have to be complete nor correct.
3. Study and implement the existing method ad Lavi [1], the method regularizes SVM mRNA classifier with the protein-protein interaction network. Generalize for omics data, try to employ sparsity regularition term as well.
4. Maximize the efficiency of the proposed method (IBM CPLEX optimizer).
5. Test the proposed method on a set of omics datasets provided by your supervisor. Compare with the classical SVM and SVM ad Lavi. The main criteria shall be classification accuracy and interpretability of the resulting model.

Bibliography/Sources:

[1] O. Lavi, G. Dror, and R. Shamir. Network-induced classification kernels for gene expression profile analysis. Journal of Computational Biology, 19(6):694{709, 2012.
[2] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. Advances in neural information processing systems, 16(1):49-56, 2004.
[3] M. Andel, J. Klema, Z. Krejcik: Network-Constrained Forest for Regularized Omics Data Classification. In Proceedings of The IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 410-417, 2014.
[4] A. D. Baxevanis, B. F. Francis Ouellette: Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, 3rd Edition, Wiley, 2004.

Bachelor Project Supervisor: doc.Ing. Jiří Kléma, Ph.D.

Valid until the end of the summer semester of academic year 2015/2016

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 22.5.2015 .....

Abstract

The term omics refers to genomics, proteomics or metalobomics that generate large amounts of measurements complemented by complex knowledge of functioning of genes, proteins and other molecular entities including their mutual interactions. There are many classification algorithms which can be used to analyze omics data. However, not many algorithms provide required qualities such as accuracy, stability, interpretability and efficiency. SVM algorithm shows accuracy, stability and robustness to cope with the frequent characteristic of omics data, a small number of samples compared to the number of features. In this thesis, several ways of regularizing the SVM algorithm are considered. They are expected to improve its interpretability and accuracy. The traditional SVM input gets supplemented by the prior knowledge of feature interactions expected to mirror the true interactions inside organisms. The classifier obtains penalty for assigning different weights to directly interacting features. Furthermore, SVM is regularized by the sparsity term. The final regularized form is called Sparse network-constrained L2-norm SVM. The presented results demonstrate its great improvement in terms of classifier interpretability while only moderately decreasing its accuracy.

Termín ómika se vztahuje ke genomice, proteomice či k metabolomice, což jsou obory, které generují velké množství dat doplněných komplexními znalostmi funkcí genů, proteinů a jiných molekulárních entit, včetně jejich vzájemných interakcí. Pro analyzování ómických dat může být použito více klasifikačních algoritmů, nicméně jen několik z těchto algoritmů je dostatečně kvalitních v ohledu přesnosti, stability, interpretability a efektivnosti. SVM algoritmus se s dostatečnou mírou přesnosti, stability a odolnosti dokáže vyrovnat s často zmiňovanou charakteristikou ómických dat, která se vyznačuje malých počtem vzorků v porovnání s počtem příznaků. V této práci je zvažováno několik způsobů regularizace SVM algoritmu, od kterých se očekává navýšení přesnosti a interpretability. Vstup do SVM algoritmu je obohacen apriorními znalostmi interakcí příznaků, s předpokladem, že tyto interakce

reflektují skutečné interakce unvitř organismů. Klasifikátor následně aplikuje odlišná ohodnocení pro přímo interagující příznaky. SVM je navíc regularizováno výrazem, který upravuje řídkost modelu. Výsledná regularizovaná forma se nazývá Sparse network-constrained L2-norm SVM. Prezentované výsledky takového klasifikátoru prokazují ohromné zlepšení intepretability klasifikátoru za přiměřeného poklesu přesnosti.

Contents

Chapter 1

Introduction

There are different domains where one would like to predict the evolution of some phenomenon. Example of such a domain are living organisms whose evolution is predefined by DNA (deoxyrubonucleic acid). It would be pleasing if a predisposition of a human being to some disease could be predicted according to its DNA. However, organisms are controlled by complex internal processes and thus the prediction of some phenomenon should be based both on the DNA and a simulation of the internal processes. This problem can be modeled by exploiting machine learning methods.

First, we need to understand the nature of the input data/samples to the machine learning method in order to select an appropriate one. In this thesis were processed gene expression (GE) and micro ribonucleic acid (miRNA) expression files containing measured values of GE/miRNA of different patients. GE quantifies how much gene takes part in creating products, mainly proteins. MiRNA expression quantifies how much miRNA molecules functions in ribonucleic acid (RNA) silencing and post-transcriptional regulation of GE. GE and bioinformatics is further explained in [11]. Patients in collected data sets are marked as positive/negative with regard to having myelodysplastic syndrome (MDS)

Second, we need to select an appropriate machine learning method/algorithm. The process of predicting a phenomenon by an algorithm has two phases. At the beginning, the algorithm learns the decision rules how to distinguish the input samples. Then, the algorithm assigns value of the assessed phenomenon (positive/negative) to a random sample according to the decision rules. This process is in machine learning called classification.

One of the methods is Support Vector Machines (SVM) [25]. The reasons

for using SVM are that it optimally separates the input samples in multidimensional space and also it deals with the $n << p$ problem when there are much more features p compared to the number of samples n.

The aim of this thesis is to assess the precision of classification models using classical SVM algorithm and its regularized versions. There are two different regularizations considered. First regularization includes interactions between mRNA's and miRNA's which tries to simulate inner processes in an organism. Second regularization selects only those features which seem to be important when deciding about some phenomenon. The final goal of this thesis is to verify whether the more complex and time-consuming regularized forms of SVM may be an improvement in the terms of the model accuracy.

Results will be compared to other experiments [8][18] which also worked with MDS data.

Chapter 2

Models used for classification

This chapter introduces mathematical background of created models and reasons for using them. First, classical SVM is presented. Then, network-constrained L2-norm is introduced which regularizes the classical SVM model. Afterwards, sparsity regularization term is added to the model in order to minimize the number of features in classification. Finally, negative regulatory assumptions are considered.

2.1 Classical SVM

Let us assume a binary classification problem. The input feature space is separable by an infinite number of hyperplanes, example of such hyperplanes is shown in Figure 2.1. The advantage of SVM method is that it looks for the optimal hyperplane which separates the two sample classes. It tries to find a hyperplane which is at maximum distance from both positive and negative samples [25].

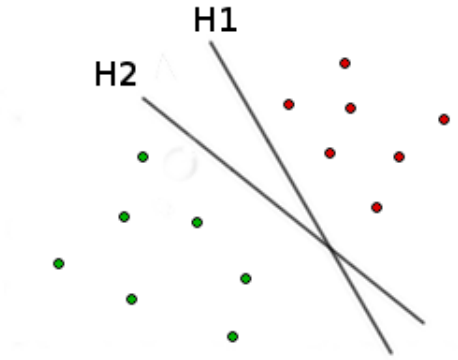


Figure 2.1: Possible hyperplanes to separate the points

SVM maximizes the margin, between the samples and the separating hyperplane. That is a feasible quadratic programming problem. Input to SVM method is a set of training pairs $x_i \in R^M, y_i \in \{-1, 1\}$. Output is a weight vector $\mathbf{w}$, one weight w_j for each feature, whose linear combination predicts value y_i'. The idea of SVM is that only important sample points should decide about the optimization of the solution.

2.1.1 Primal problem

Let me describe the problem in Figure 2.2.

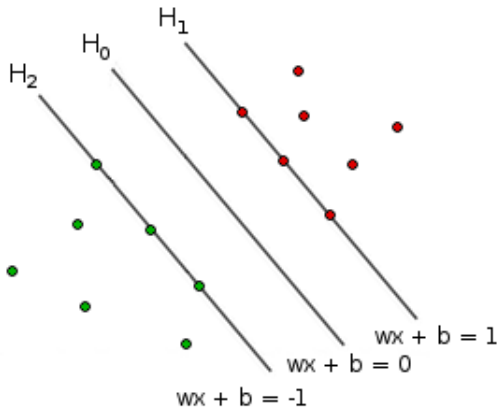


Figure 2.2: Defining support vectors and hyperplanes

General definition of points lying on a hyperplane is $\mathbf{w} \cdot \mathbf{x} + b = 0$. This hyperplane separates the positive and negative samples. In order to create the margin between the two classes we define one class of samples, for example the positive one, as half-plane $\mathbf{w} \cdot \mathbf{x} + b \geq 1$. Negative class of samples is defined as $\mathbf{w} \cdot \mathbf{x} + b \leq -1$. Samples belonging to hyperplanes $\mathbf{w} \cdot \mathbf{x} + b = \pm 1$ are called support vectors.

Also the width of the margin needs to be determined. It is computed as twice the distance of a point $X[x_0, y_0]$ of one support vector hyperplane to the separating hyperplane $p : ax + by + c = 0$. The equation for calculating the distance of a point from a line is:

$$dist(X, p) = \frac{\mid ax_0 + by_0 + c \mid}{\sqrt{a^2 + b^2}}. \tag{2.1}$$

Then the distance of hyperplanes H1 a H0 is $dist(H1, H0) = \frac{|\mathbf{wx}+b|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$. The total width of the margin is apparently $\frac{2}{\|\mathbf{w}\|}$.

The way to maximize this fraction is to minimize the denominator $\|\mathbf{w}\|$ under condition that there are no points between hyperplanes H1 and H2. This condition is ensured by defining constraints:

1. $\forall i \mid y_i = 1 : \mathbf{wx}_+ + b \geq 1$
2. $\forall i \mid y_i = -1 : \mathbf{wx}_- + b \leq 1$

Then the constraint $y_i (\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1$ is the same for both positive and negative samples. The minimization of $\|\mathbf{w}\|$ can be rewritten as minimization of $\frac{1}{2}\|\mathbf{w}\|^2$ because $\|\mathbf{w}\|^2$ reaches its extremes in the same values as $\|\mathbf{w}\|$ and $\frac{1}{2}$ is used because it is mathematically more convenient.

The regularized term is:

$$\min_{w_0, w_1 \ldots w_m} \left\{ \frac{1}{2} \sum_{i=1}^{M} w_i^2 \right\}. \tag{2.2}$$

$$\text{s.t. : } \left(\mathbf{w}^T \mathbf{x_i} + w_0 \right) y_i \geq 1, \forall i = 1, \ldots n$$

2.1.2 Dual problem

The problem is to find local extremes of function constrained by other functions. These local extremes can be found by using Lagrange multipliers. Slack variables are defined in order to rise values of points which are support vectors and decrease the values that are not support vectors. Desired point P is a point where:

1. $\nabla f(P) = \nabla \alpha g(P)$ (point where gradient of optimized function is parallel to the gradient of constraint function),
2. $g(P) = 0$ (Constraint condition. Support vector is tangent)

The joint of previous equations leads to $L(x, \alpha) = f(x) - \alpha g(x)$. Then:

1. Partial derivative with respect to $\mathbf{w}$ satisfies parallelism of normals,
2. Partial derivative with respect to b satisfies condition that $g(x)$ is tangent to $f(x)$.

Generally:

$$L(x, a) = f(x) + \sum \alpha_i g_i(\mathbf{x}),$$

$$\min_{\mathbf{W},b} L = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i \left(\mathbf{w} \cdot \mathbf{x_i} + b\right) - 1\right),$$

$$\min_{\mathbf{w},b} L = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i y_i \left(\mathbf{w} \cdot \mathbf{x_i} + b\right) + \sum_{i=1}^{N} \alpha_i.$$

Partial derivatives:

$$\frac{\partial L(\mathbf{w},b,\alpha)}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0,$$

$$\frac{\partial L(\mathbf{w},b,\alpha)}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x_i} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x_i}\ .$$

Instead of minimizing over w, b under conditions of α_i it can be maximized over α under conditions gained from partial derivatives with respect to $\mathbf{w}$ and b:

Solution must be in compliance with relations:

1. $\sum_{i=1}^{N} \alpha_i y_i = 0,$

2. $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x_i}$.

By substituting into original equation we eliminate the dependence on $\mathbf{w}$ and b:

$$L_D = \frac{1}{2}\left(\sum_{i=1}^{N}\alpha_i y_i \mathbf{x_i}\right)\left(\sum_{i=1}^{N}\alpha_i y_i \mathbf{x_i}\right) - \sum_{i=1}^{N}\left[\alpha_i y_i \mathbf{x_i}\left(\sum_{i=1}^{N}\alpha_i y_i \mathbf{x_i}\right) + \sum_{i=1}^{N}\alpha_i y_i\right] + \sum_{i=1}^{N}\alpha_i$$

$$\begin{aligned} L_D &= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j {\mathbf{x_i}}^T \mathbf{x_j}, \\ s.t. &\sum_{i=1}^{N}\alpha_i y_i = 0 \wedge \alpha_i \geq 0 \end{aligned} \tag{2.3}$$

Dual form enables to compute only dot products ${\mathbf{x_i}}^T\mathbf{x_j}$. With obtained slack variables α_i can be derived the weights vector $\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x_i}$. Bias b will be computed from all the support vectors [16]:

$$b = \frac{1}{N}\sum\left(y_i - \mathbf{w}\mathbf{x_i}\right) \tag{2.4}$$

New samples can be classified by the rule $f(x) = \mathbf{w}\mathbf{x} + b$.

2.2 Network-constrained L2-norm

The approach of regularizing SVM by network constraint L2-norm is e.g., in Lavi, 2012 [20]. The criterion minimized here is the classical SVM criterion with an additional network-regularization term:

$$\underset{w_0, w_1 \ldots w_m}{min}\left\{\frac{1}{2}\sum_{i=1}^{M} w_i^2 + \frac{1}{2}\beta\sum_{ij\in\tau} A_{ij}\left(w_i - w_j\right)^2\right\} \tag{2.5}$$

$$\text{s.t. : } \left(\mathbf{w}^T\mathbf{x_i} + w_0\right) y_i \geq 1, \forall i = 1, \ldots n$$

where $\mathbf{x_i} \in \mathbb{R}^{\mathbb{M}}$ is a data sample, $A \in B^{M \times M}$ is a network adjacency matrix, β is a network strength parameter, and $\mathbf{w}$ and b are coefficients of decision function which are to be optimized. The criterion can be rewritten in a vector-matrix form:

$$\underset{\mathbf{w},w_0}{min} \left\{ \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{1}{2}\beta\mathbf{w}^T \left(\tilde{A} - A\right)\mathbf{w} \right\} \tag{2.6}$$

where $\tilde{A}$ is diagonal matrix of A, with $\tilde{A_{i,j}}$ being the degree of node i. Henceforth, the matrix $B = \tilde{A} - A$ is known as the Laplacian matrix of the network graph. Similarly as to the standard SVM, it is not necessary to compute all of $M + 1$ weight coefficients. The problem is possible to convert into a dual form, where N only slack variables α_i, referring to support vectors, i.e. Lagrangian multipliers are to be optimized. The primal Lagrangian of (2.5) is:

$$\begin{aligned} L &= \mathbf{w}^T\mathbf{w} + \beta\mathbf{w}^T B\mathbf{w} - \sum_{i=1}^{N}\alpha_i((\mathbf{x_i}^T\mathbf{w} + w_0)y_i - 1), \\ L &= \mathbf{w}^T(I + \beta B)\mathbf{w} - \sum_{i=1}^{N}\alpha_i((\mathbf{x_i}^T\mathbf{w} + w_0)y_i - 1) \end{aligned} \tag{2.7}$$

The conventional L2-norm, originally $\|\mathbf{w}\|_2 = \sqrt{\mathbf{w}^T\mathbf{w}}$ is here simply enriched by the Laplacian of the network graph. The corresponding dual form is following:

$$L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T(I + \beta B)^{-1}\mathbf{x_j} \tag{2.8}$$

which is the same as in the case of standard SVM besides the kernel $(I + \beta B)$.

2.3 Sparse Network-constrained L2 norm

Network constrained L2-norm in general increases models stability and interpret-ability, encouraging the features sharing the same interaction to

have similar weights w in the discriminative function. Henceforth, if one of the features obtains high weight, the other similar features are expected to have similar weight, which increases the reliability of the model. However, under this criterion the irrelevant features still remain in the model. The problem of model size and irrelevant features in general is addressed by L1-norm, [27]. L1-norm, $\sum_{i=1}^{M} |w_i|$, shrinks irrelevant features to zero, implicitly performing feature selection. Adding L1-norm to (2.5) might enable selecting only these sets of interaction which are model relevant and interpretable. Henceforth, according to the specificity of persisting set of interactions towards particular locus or regulator may lead to another investigations.

$$\begin{aligned} \underset{w_0, w_1 \ldots w_m}{min} \left\{ \sum_{i=1}^{M} |w_i| + \frac{1}{2} \sum_{i=1}^{M} w_i^2 + \frac{1}{2} \beta \sum_{ij \in \tau} A_{ij} \left(w_i - w_j \right)^2 \right\}, \\ s.t. : s.t. : \left(\mathbf{w}^T \mathbf{x_i} + w_0 \right) y_i \geq 1, \forall i = 1, \ldots N \end{aligned} \tag{2.9}$$

Unfortunately, this criterion is not differentiable, thus impossible to convert into a dual form. The solution is consequently much more computationally demanding.

2.4 Sparse L2 norm

Effect of Sparsity term was explained in Section 2.3. However, in order to check whether Sparse Network-constrained L2 norm chooses a set of more relevant features due to the interaction constraints it has to be compared to the Classic Sparse SVM. After performing comparison it will be visible how much the interaction network improves the selection of relevant features.

$$\begin{aligned} \underset{w_0, w_1 \ldots w_m}{min} \left\{ \sum_{i=1}^{M} |w_i| + \frac{1}{2} \sum_{i=1}^{M} w_i^2 \right\}, \\ s.t. : s.t. : \left(\mathbf{w}^T \mathbf{x_i} + w_0 \right) y_i \geq 1, \forall i = 1, \ldots N \end{aligned} \tag{2.10}$$

2.5 Network-constrained L2-norm with Negative Regulatory Assumption

The criteria (2.5),(2.6) of Lavi, 2012 [20] consider protein-protein interactions (PPI). His assumption was, genes with interacting proteins contribute similarly to the phenotype in the positive sense. Which means, if i-th gene's coefficient w_i has highly positive or highly negative value, respectively, its neighbourgh is expected to have also highly positive or highly negative w_j, respectively. However, this does not reflect possible inhibitive regulatory relationship when a gene is down regulated by a miRNA or transcription factor (TF). In this case, I might expect that if an i-th miRNA has highly positive contribution in terms of high value of corresponding coefficient wi , its j-th target being inhibited would have highly negative wj ; or vice versa. Fortunately, an inhibitive regulation is simply possible to model by re- placing - by + in (2.5). Henceforth, a new criterion reflecting inhibitive interactions looks as follows:

$$\begin{gathered} \underset{w_0,w_1\ldots w_m}{min} \left\{ \frac{1}{2}\sum_{i=1}^{M} w_i^2 + \frac{1}{2}\beta \sum_{ij\in\tau} A_{ij}\left(w_i + w_j\right)^2 \right\}, \\ s.t. : \left(\mathbf{w}^T\mathbf{x_i} + w_0\right) y_i \geq 1, \forall i = 1, \ldots n \end{gathered} \tag{2.11}$$

Expressing the criterion in matrix form is exactly same as in (2.6), only this time the graph adjacency matrix A has -1 values where an inhibitive relationship appears. This criterion is also possible to express in a dual form and effectively solve.

2.6 Network-norm-SVM

However [20] uses gene networks to regularize SVM criterion, the L2 norm persists in the criterion. There is a quest why not to use *only* the network-regularization term, similarly as sparse SVM uses only L1 norm. In this case the primal problem would look as follows:

$$\begin{gathered} \min_{w_0,w_1,\ldots,w_M} \left\{ \frac{1}{2}\beta \sum_{ij\in\mathcal{I}} A_{ij}(w_i - w_j)^2 \right\}, \\ s.t. : (\mathbf{w}^T\mathbf{x}_i + w_0) y_i \geq 1, \forall i = 1, \ldots, N, \end{gathered} \tag{2.12}$$

while the dual problem looks as:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T (\beta \mathbf{L})^{-1} \mathbf{x}_j, \tag{2.13}$$

This criterion is strictly convex since the graph Laplacian matrix $\mathbf{L}$ is symmetric positive semidefinite. Therefore the problem has only one optimal solution and is ready to be used within robust machine learning experiments.

Chapter 3

Data

This chapter mentions the analyzed data sets. Four types of data sets were used for testing classical SVM. First two of them, described in Section 3.2.1 and 3.2.2, were used to verify the code. The third, described in 3.3.1, contains the biological data which are the reason of this assessment. The last one, mentioned in Section 3.3.2, contains artificial biological data.

3.1 General form of a data set

Each data set includes the metadata of columns. These are the names of the measured features. First column of data sets contains indexing of the samples. Last column contains the class of the sample. Sample is positive if there is number 1 in the last column of its row, otherwise it is negative. 3.1.

	Feature_1	Feature_2	...	Feature_N	Class
0	12.33	100.12	...	54.34	0
1	14.01	100.32	...	56.77	1
2	12.2	102.23	...	55.23	1
...	...	...	...	45.65	...
N	15.76	102.65	...	65.23	1

Figure 3.1: Example of gene expression file

3.2 Data for code verification

This section describes the data used for code verification.

3.2.1 Linearly separable 2D points

First were used the datasets with linearly separable 2D points. First task contains classes separable by line $y = x$, that means bias equals zero. The second task is separable by $x = 2$.

3.2.2 Iris data

Then was used the Iris dataset from UCI machine learning repository [15]. The dataset contains three classes. Nevertheless two of them are not separable so only two clusters are considered to be found. I set Iris virginica and Iris versicolor label to 0 and Iris setosa to 1.

3.3 Omics data

Omics data refer to the fields of biology ending with 'omics'. Such fields are genomics, proteomics and metabolomics [24]. Genomics explores the genomes of organisms. Proteomics studies the structure and functions of proteins. Metabolomics refers to the biological compounds that help regulating biological processes. The following datasets are examples of omics data.

3.3.1 Myelodysplastic syndrome data

The third group of data sets was provided by Institute of Hematology and Blood Transfusion in Prague. They are related to myelodysplastic syndrome (MDS) and include mRNA and miRNA values.

SVM algorithm was tested on three combinations of files. First, the mRNA data sets containing 16666 features of GE were tested. Then the miRNA data sets containing 1146 features were tested. Finally, those two data sets were merged together and tested.

There are ten different tasks in the MDS data sets containing either samples of mRNA/miRNA GE taken from bone marrow (BM) or peripheral

blood (PB) before treatment (BT) or during treatment (DT). The tasks with their ratio of positive vs. negative samples are:

1. BMBT_DT5q - 11:5
2. BMH_ABT5q - 10:11
3. BMH_ABTnon-5q - 10:6
4. BMH_ADT5q - 10:5
5. BMnon-5q_5q_BT - 6:11
6. PBBT_DT5q - 9:13
7. PBH_ABT5q - 10:9
8. PBH_ABTnon-5q - 10:4
9. PBH_ADT5q - 10:13
10. PBnon_-5q_5qBT - 4:9

These tasks are described in detail is in [8].

Interactions files

There are two types of feature interactions that have to be added to regularized models. First are protein-protein interactions. Second are mirna-target interactions. These known interactions were gathered from [26][14][17][13] Moreover, different approaches are used when adding interactions to the primal and dual form.

Protein-Protein interactions file

This file contains interactions among any two proteins in the organism. Each row of the file consists of two names of the features which are separated by a tab. There are 80443 known interactions among proteins.

Mirna-Protein interactions file

This file contains interactions among miRNAs and proteins. Each row of the file consists of the name of the miRNA feature and the name of the mRNA feature which are separated by a tab. This interactions can be translated as: "Selected miRNA inhibits the creation of specific mRNA". There are 108528 known mirna-target interactions.

Adding interactions to a Dual form

Interactions are added to the dual form in a form of a matrix. First, the creations of adjacency and degree matrix of a graph were needed in order to create its Laplace matrix $L{=}\tilde{A} - A$. The Laplace matrix can be used in vector-matrix primal form of the SVM. However, $(I + \beta B)^{-1}$ is needed in the dual form. This leads to addition of the Laplace matrix and the identity matrix. The addition ensures that matrix will be invertible because of the non-zero numbers on the diagonal line. Example of a matrix to invert is below. $\{a_{11}, a_{22}...a_{nn}\}$ represent degree of node n. (sum of column/row n from adjacency matrix)

$$\left(\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array} \right. + \left(\begin{array}{ccccc} a_{11} & 0 & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 & 0 \\ 0 & 0 & a_{33} & 0 & 0 \\ 0 & 0 & 0 & a_{44} & 0 \\ 0 & 0 & 0 & 0 & a_{nn} \end{array} \right. - \left. \left. \begin{array}{ccccc} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{array} \right) \right)$$

This kind of matrix is difficult to create when running the python program for solving model. It takes more than 5 hours. Because of this the matrices were generated in MATLAB [21].

There are three matrices which can be used, one containing both protein-protein interactions (further named as P-P), second containing miRNA-protein interactions (further named as miRNA-P) and last containing both P-P and miRNA-P interactions (further named as miRNA-P,P-P). The sizes of those files vary from 1.4GB to 2.7GB.

Adding interactions to a Network-norm-SVM Dual form

This form computes with the inverse of Laplace graph $(\beta B)^{-1}$ containing -1 on indexes of interacting nodes. However this matrix is singular and thus doesn't have an inverse matrix.

Adding interactions to a Primal form

In the primal form it is only need to know whether the interaction between features exists ($A_{ij}\left(w_i - w_j\right)^2$). If the interaction among w_i and w_j exists in the interactions file then A_{ij} is set to 1, otherwise it is set to 0.

3.3.2 Artificial Data

Artificial datasets refer to bacteria Escherichia coli and were generated by a framework GeneNetWeaver [7], providing methods for both in silico benchmark generation and performance profiling of network inference algorithms. This dataset enables training/testing models on sets containing more samples (around 100). The idea of using artificial data is supported because it could later give better insight into the process of selecting/updating weights of the features. Moreover the features in datasets could be changed in order to asses the behavior of the algorithms. Also experiments on this dataset consume less time because they have only 1565 features.

Interactions file

The interactions file contains relations between any two features. The relation is either amplification (+), more of the second feature is produced due to the first feature, or inhibition (-), less of the second feature is produced due to the first feature.

Chapter 4

Platform

In this chapter will be mentioned tools used for assessing individual tasks. By tools are meant computation resources, programming language, mathematical modeling languages, libraries for matrix computing, optimizer and tools for analyzing results.

4.1 Metacentrum

MetaVO is an organization which lets students and academics use its computational resources. The computations were moved to Metacentrum because the tasks take up to 260GB of memory and up to 40 hours. Furthermore, it enables the parallelization of computing tasks.

4.2 IBM ILOG CPLEX Optimizer

Searching for optimal boundary which separates the samples requires solving multiple equations. Therefore, an optimizer is required. One of high-performance solvers is IBM ILOG CPLEX Optimizer (CPLEX) [1]. CPLEX is a programming solver for linear programming, mixed integer programming and quadratic programming, offering multiple ways how to solve models. One possibility is to use its IBM ILOG Optimization Studio client. Second is using CPLEX application programming interface (API) in high-level programming languages.

4.2.1 Cplex algorithms

CPLEX offers different algorithms for solving created models. The possibilities for choosing optimizer are:

1. Default Setting, IloCplex.Algorithm.Auto,
2. Primal Simplex, IloCplex.Algorithm.Primal,
3. Dual Simplex, IloCplex.Algorithm.Dual,
4. Network Simplex, IloCplex.Algorithm.Network,
5. Barrier Optimizer, IloCplex.Algorithm.Barrier,
6. Sifting Optimizer, IloCplex.Algorithm.Sifting,
7. Concurrent Dual, Barrier, and Primal, IloCplex.Algorithm.Concurrent.

4.2.2 Default setting

Default setting lets CPLEX determine which optimizer to choose. On a serial computer and on a parallel computer where only one thread can be used is chosen the dual simplex algorithm. Unless you have an advanced basis which is ascertained to be primal feasible, then primal simplex will be used. On a computer with parallel thread a concurrent optimizer will be called.

4.2.3 Dual simplex

A mathematical programming problem can be written in a primal or dual form. An optimal solution of the dual model has a direct relationship to primal model. CPLEX Dual Simplex Optimizer uses this relationship and still reports the solution in terms of primal model. The dual simplex is usually faster than the primal simplex optimization.

4.2.4 Primal simplex

Primal simplex is no longer the first choice after the dual simplex optimizer was developed. However sometimes it could work better when the

number of the variables significantly exceeds the number of constraints. Using primal simplex is considered when having problem with the dual simplex optimizer.

4.2.5 Barrier

Barrier optimizer is used on large, sparse problems. It is quick but because of the speed it sometimes doesn't reach the optimal solution. Instead of optimal solution it returns an unbounded solution or numBest solution. Unbounded means that if there is a feasible solution with objective value x*, there exists a feasible solution with objective value x*-1 for a minimization problem, or x*+1 for a maximization problem, described in [5]. If it is numBest then solution is available, but not proved optimal, due to numerical difficulties during optimization.

4.2.6 Sifting optimizer

Sifting optimizer is efficient for problems problems with high ratio between the number of columns and the number of rows. It could be used on models where most of variables are expected to have values on the bottom of the bound limit. Sifting optimizer cannot used for solving quadratic problems.

4.2.7 Concurrent

The concurrent optimizer launches distinct optimizers on multiple threads.

4.2.8 Optimizer used in this assessment

It was decided to use the dual and primal simplex optimizers in this thesis. Dual simplex will be used in most situations. However, the primal simplex optimizer will be used if the dual simplex optimizer runs into trouble.

4.3 Programming language

As mentioned in Section 4.2, CPLEX offers possibility to use its API in high-level programming languages. This possibility was an obvious choice

because of the need to launch solving models remotely on Metacentrum. Also this choice enables better preprocessing of data sets and using other data mining tools. CPLEX offers API in several programming languages. Such languages are Java, C++, C#, Python and maybe other. Java was chosen at first as there was previous experience with Java before. However, it turned to be inefficient with memory when adding quadratic expression objects to classification models and when working with big matrices. Second choice was Python. Using Python API to CPLEX is not easy to use. However, modeling language modules, mentioned in Section 4.4, simplify using CPLEX API. Combination with numeric module Numpy [23], Sci-kit learn tool for data manipulation [22], and Pandas for reading csv files [2], seems to be much more suitable for this assessment.

4.4 Modeling language

Using CPLEX Python API is problematic. The documentation is minimal and thus it is complicated to create quadratic problem models. Fortunately, there are several python modules with modeling languages which can be used. First, PICOS [3] modeling language was tried. It simplified modeling problems in program but it ran into trouble when summing quadratic expressions. PICOS was too slow and it took hours to create a model. Then was found PyCPX [4] which is both simple and efficient when working with quadratic expressions. With PyCPX can be easily created both linear and quadratic programming models and the models can be passed to the CPLEX optimizer.

4.5 MKL - Math Kernel Library

Models need to multiply big matrices when computing dual form SVM. This would take lots of time when using Numpy module build on linear algebra package LAPACK [5], which uses basic linear algebra subprograms (optimized subroutines like copying vectors etc...) BLAS [12]. Fortunately, there exist several libraries with optimized math operations. One of them is MKL [6]. Metacentrum provides a MKL module so Numpy module linked to MKL had to be build. Additionally Pandas module module had to be built upon boosted Numpy module.

4.6 Graphviz - Graph Visualization Software

The analysis whether selected weights (features) interact together had to be performed in order to determine how much interpretable are sparse models. This could be displayed thanks to a python interface to Graph Visualization Software. A python script was implemented which connects together the selected features which interact together according to the interaction files. Such visualization of results of a specific model can be seen in Figure 6.1.

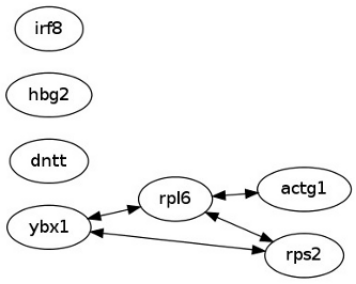


Figure 4.1: Interactions among selected features (mRNA/miRNA) from result of a Sparse Netwok-Constrained L2-norm

Chapter 5

Execution of tasks on Metacentrum

This chapter describes the whole process of executing programs on Metacentrum.

5.1 Registering and Accessing Metacentrum

Metacentrum requires registration before it authorizes anyone to use its resources. It also requires users to define their purpose of using Metacentrum and the academic institution they are part of. After obtaining authorization, users access the servers by SSH client [10]. Command to access the front-end node named skirit is:

```
ssh username@skirit.ics.muni.cz
```

Front-end node is a place where user stores scripts and programs he wants to execute on Metacentrum. Another type of a node is a computation node. Computation nodes are assigned to user's tasks. Last type of a node is a storage node where user redirects output of programs.

5.2 Uploading data to Metacentrum and initializing environment

Five activities have to be performed in order to properly run a program on Metacentrum. Such activities are depicted in Figure 5.1.

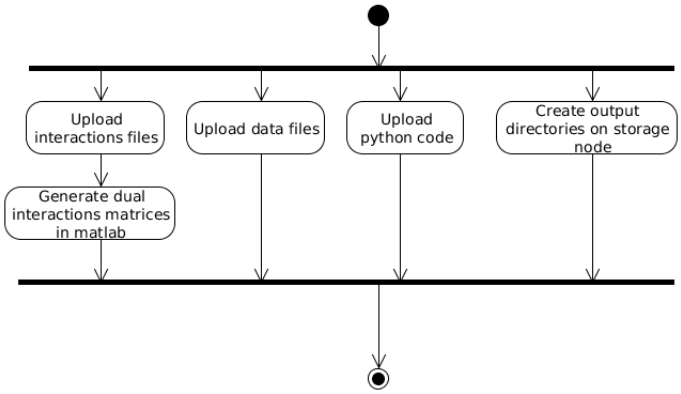


Figure 5.1: Activity diagram of initializing environment on Metacentrum

Important files and inputs to models, are uploaded to front-end nodes like skirit. Skirit is not the only front-end node. There are more of them, for example node named tarkil.

User has a home directory on these front-end nodes. Linux command to access this directory after logging to Metacentrum, mentioned in Section 5.1, is:

```
cd /storage/praha1/home/username
```

"praha1" defines which server to use.

Initializing environment also includes generating dual interaction matrices. These matrices could not be uploaded because of their size (around 2GB). Thus MATLAB scripts were created in order to generate interaction matrices. Scripts are executed by using Metacentrum launcher, mentioned in Section 5.4.

5.3 Initializing computation nodes and executing program

Execution of the program is not performed on the front-end node where the program is stored but on the computation node. The computation node is assigned to the task by Metacentrum launcher. This process is explained in the Section 5.4.

However, the computation node's default state does not contain CPLEX, Python modules like Numpy, Scikit, Pandas. These modules have to be added to the computation node before executing python programs for creating models. This is done by a bash script containing commands initiating the node's modules and then launching the python program.

First, the correct version of Python programming language must be added to the environment. Then must be added Python modules used in the program which are Pandas, Numpy and Sci-kit. Afterwards, the CPLEX module and also the PyCPX modeling language has to be added in order to solve the models. However PyCPX is not among the default Metacentrum modules so it had to be added to the front-end node and it has to be explicitly added to a PYTHONPATH every time the program is launched.

Structure of such script is shown in Figure 5.2.

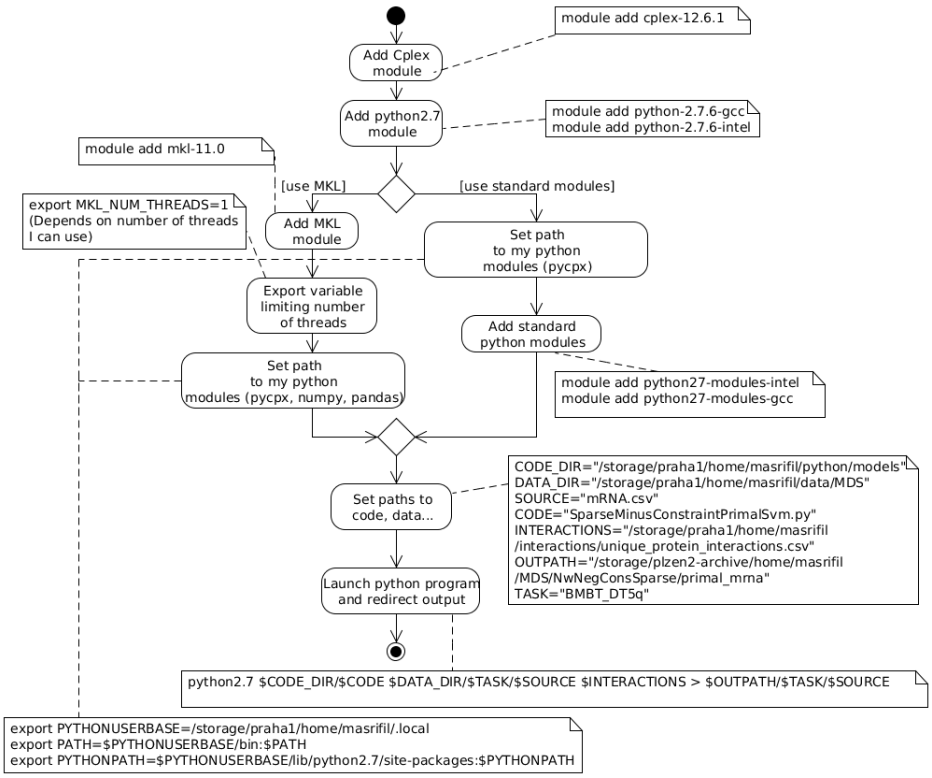


Figure 5.2: Structure of bash script to launch python program

Experiment's scripts are in appendix A, directory "bashScripts".

5.4 Adding tasks to Metacentrum launcher

Once the task's script is prepared with all the configuration, it has to be added to Metacentrum's launcher. The launcher takes the task, evaluates it with priority and passes it to a queue for some computational resource. The priority is derived from resources the program is going to process. By launching command is specified the amount of maximum time the task can run, amount of memory it can consume during computing and the amount of processors needed. Here are presented examples of launching commands:

1. `qsub -l walltime=50h -l nodes=1:ppn=2:mem=80gb launcher.sh`
2. `qsub -l walltime=30h -l nodes=1:ppn=1:xeon -l mem=80gb launcher.sh`

On these two examples will be described launching a program on Metacentrum. The command for launching a script is named "qsub". This command can be supplemented by additional parameters. A parameter is announced by "-l". Afterwards can be set individual parameters.

- "walltime" - maximum run-time of a task
- "nodes" - number of machines my task is going to run on
- "ppn" - number of processors on each machine
- "mem" - memory a program can use

Additionally, Math Kernel Library is needed for some tasks. However MKL can be used only on machines with Intel processors. A request for a machine with for example Xeon processor can be performed by adding ":xeon" to the parameters. If a Numpy module build upon MKL would be used on a non-Intel processor, the program would crash because the processor would not recognize incoming instructions.

5.5 Retrieveing results

The results of the python program are redirected to a file which is created on a storage node. Storage nodes "plzen2-storage" and "jihlava1-cerit" were used as the destination nodes. This directory can be accessed after logging to Metacentrum by command:

```
cd /storage/jihlva1-cerit/home/username
```

Experiment outputs are located in apendix A, directory "modelOutputs".

Chapter 6

Classifier implementation

This chapter presents the structure of the programmed models.

6.1 Structure of programmed models

There are different ways of regularizing SVM models. Each of these models has two sets of functions and attributes. First set is common among all models. Second set is unique for specific model. This separation leads to inheritance design among model classes.

Ten different classes are needed in order to design the whole problem which can be seen in Figure 6.1.

Class Svm is the root class responsible for loading data set files into program and dividing samples into folds according to stratified x-validation, described in Section 7.1.

PrimalSvm and DualSvm represent models mentioned in Section 2.1. Primal and dual forms are separated into individual models because their weight and bias variables are represented differently.

ConstrainedPrimalSvm and ConstrainedDualSvm are classes responsible for loading interaction files into the program. SparsePrimalSvm represents model in Section 2.4. MinusConstrainedPrimalSvm adds the interaction constraints according to the term $(w_i - w_j)^2$, mentioned in Section 2.2. PlusConstrainedPrimalForm differs only in the interactions term $((w_i + w_j)^2)$, described in Section 2.5.

SparseMinusConstrainedPrimalSvm and SparsePlusConstrainedPrimalSvm extend its parents by adding the sparsity term. SparseMinusConstrainedPri-

malSvm model is explained in Section 2.3.

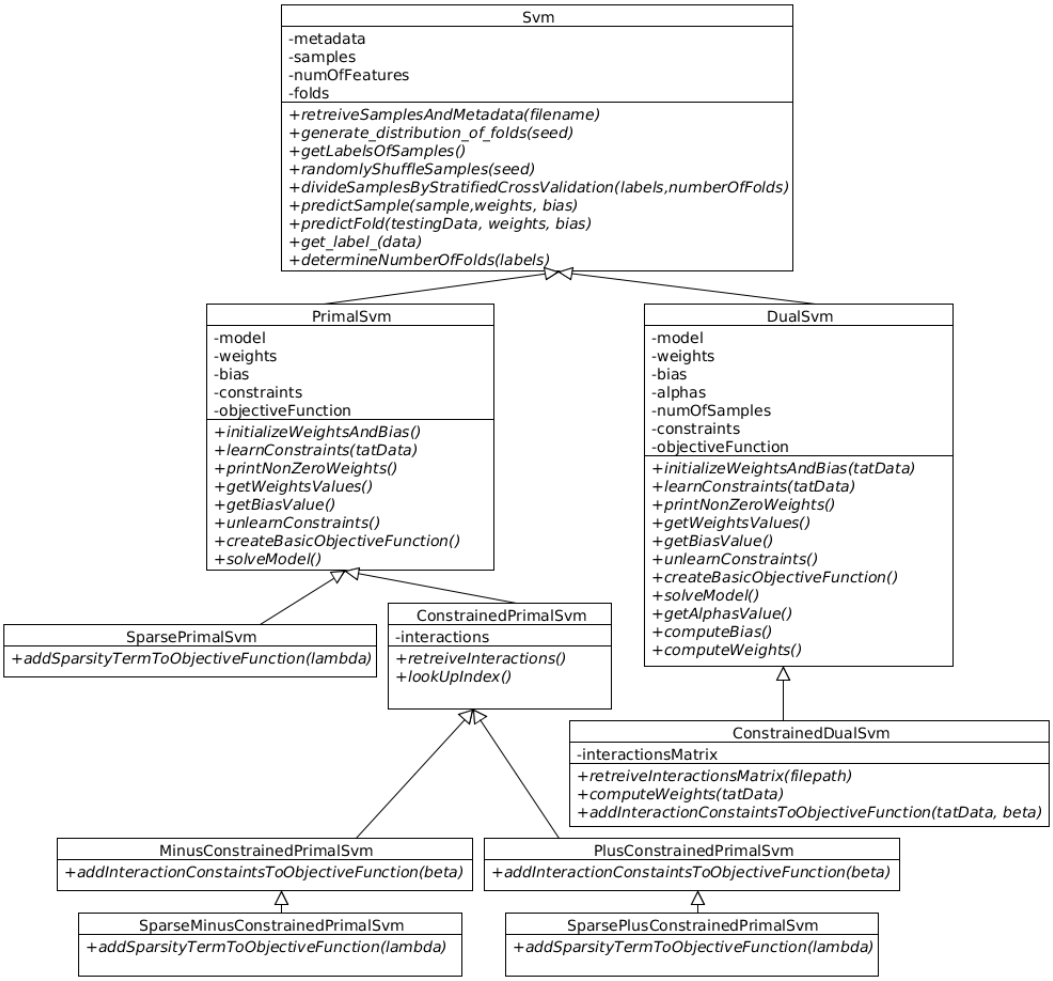


Figure 6.1: Class diagram of SVM models

Implementations of all classes are in appendix A, directory "pythonScripts".

Chapter 7

Experimental workflow

This chapter explains the methods used for training and testing the models.

7.1 Cross-validation

Cross-validation is nowadays a standard used to examine accuracy of created models. Cross-validation creates different distributions of training/testing samples in order to train/test models on different sets of samples. This way can be determined the stability of models [19]. Furthermore, 5-times repeated cross-validation is used to allow more combinations of samples in training/testing set which leads to more stable accuracy of created models and independence on particular combinations of training/testing sets.

7.1.1 Stratified Cross-validation

Stratified cross-validation is used to preserve the ratio between positive and negative samples in training/testing set compared to the source data set. This approach should increase stability of created models.

7.1.2 Determining number of folds

According to experiments in [19], sufficient number of folds is 10 in order to obtain optimal bias. However, the distribution of positive and negative samples in test/train set has to be preserved when combined with stratified

cross-validation. Thus the number of folds has to be decreased to the size of the minor class set in data set because in some data sets are less then 10 positive/negative samples. That means the number of folds is determined as the minimum of the minor size of the class in data set and 10.

7.1.3 Creating model workflow

Creating new model every time in 5-times repeated stratified cross-validation would take enormous amount of time and memory. Therefore, the process of creating models had to be optimized. Considering that the only variable part in models is the constraints referring to the training samples, I can create only one model and modify those constraints [9]. This makes the models executable on Metacentrum because it does not waste memory when compared to creating new model in each iteration. The constraints which have to be modified are:

$$s.t. : \left(\mathbf{w}^T \mathbf{x_i} + w_0\right) y_i \geq 1, \forall i = 1, \ldots n \tag{7.1}$$

The workflow of creating models is depicted in Figure 7.1. First, the non-varying part of the model is created (nodes "Interaction knowledge " and "Initialize knowledge"). This includes initializing the quadratic term ($\frac{1}{2}\sum_{i=1}^{M} w_i^2$), the interactions term ($\frac{1}{2}\beta \sum_{ij \in \tau} A_{ij} \left(w_i - w_j\right)^2$) and the sparsity term ($\sum_{i=1}^{M} |w_i|$). Then are added the varying parts (nodes "Add data constraints" and "Train fold"), presented in Equation 7.1. The constraints are dependent on the samples in the train fold and thus are changed in every iteration. Now the model is complete and passed to the solver (node "Solve QP problem").

Afterwards, the computed weights and bias are passed to the testing part of the process (nodes "Make linear predictive model", "Predict" and "Test fold") where are predicted the classes of the samples from the test fold.

Finally, the constraints are removed from the current model (node "Remove data constraints") in order to reuse the existing model on the next train/test fold.

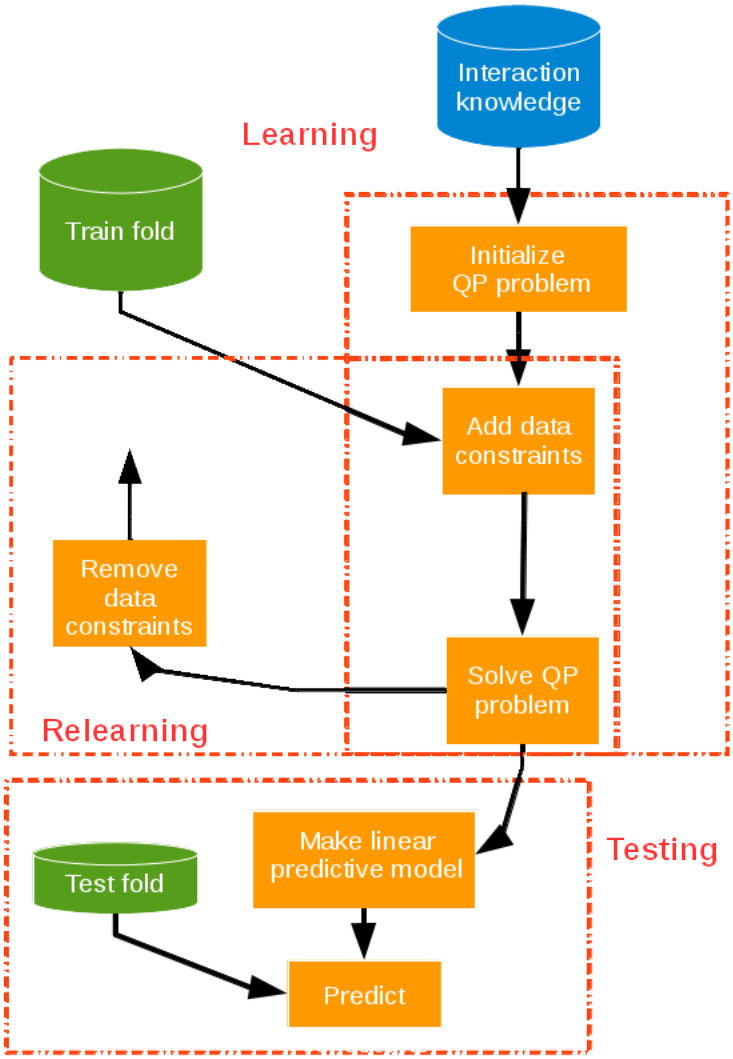


Figure 7.1: Workflow of creating models

Chapter 8

Results

This chapter presents the results of constructed models explained in Chapter 2.

8.1 Classic SVM

Following sections present the results of using Classic SVM written in Primal and Dual form on three types of data inputs. The data inputs are:

1. only measured mRNA features (mRNA)
2. only measured miRNA features (miRNA)
3. measured both mRNA/miRNA (merged)

8.1.1 Classic Dual SVM

The Table 8.1 contains accuracies and time consumptions (T) of models for different tasks and data inputs. The average memory consumption on all data inputs was 150MB.

Input → **Task ↓**	**mRNA** Accuracy(%)	T h:m:s	**miRNA** Accuracy(%)	T h:m:s	**merged** Accuracy(%)	T h:m:s
BMBT_DT5q	69.66	0:0:9	87	0:0:9	84.66	0:0:8
BMH_ABT5q	100	0:0:26	93.66	0:0:8	100	0:0:10
BMH_ABTnon-5q	95	0:0:24	100	0:0:6	100	0:0:10
BMH_ADT5q	100	0:0:23	86.66	0:0:8	86.66	0:0:10
BMnon-5q_5qBT	87.22	0:0:21	100	0:0:6	100	0:0:8
PBBT_DT5q	83.33	0:0:27	84.07	0:0:10	86.3	0:0:12
PBH_ABT5q	100	0:0:23	90.74	0:0:9	100	0:0:12
PBH_ABTnon-5q	100	0:0:49	100	0:0:6	100	0:0:9
PBH_ADT5q	98.33	0:0:49	88	0:0:9	89.66	0:0:12
PBnon-5q_5qBT	100	0:0:21	100	0:0:6	100	0:0:7
Ø	93.35	-	93.01	-	94.73	-

Table 8.1: Results of using Classic Dual SVM

8.1.2 Classic Primal SVM

The Table 8.3 contains accuracies and time consumptions (T) of models for different tasks and data inputs. The average memory consumptions are:

Data input	**Memory**
mRNA	1GB
miRNA	10MB
merged	1GB

Table 8.2: Average memory consumption of models for different data inputs

Input → **Task ↓**	**mRNA** Sucess(%)	T h:m:s	**miRNA** Sucess(%)	T h:m:s	**merged** Sucess(%)	T h:m:s
BMBT_DT5q	69.66	0:06:24	89.66	0:0:7	84.66	0:03:52
BMH_ABT5q	100	0:08:37	93.66	0:0:7	100	0:07:51
BMH_ABTnon-5q	93.33	0:02:49	100	0:0:6	100	0:04:07
BMH_ADT5q	100	0:02:20	86.66	0:0:4	86.66	0:03:25
BMnon-5q_5qBT	87.22	0:05:28	100	0:0:43	100	0:05:45
PBBT_DT5q	83.33	0:08:39	86.66	0:0:39	86.3	0:09:14
PBH_ABT5q	100	0:08:52	91.85	0:0:6	100	0:10:33
PBH_ABTnon-5q	100	0:01:28	100	0:0:6	100	0:02:08
PBH_ADT5q	98.33	0:09:15	87	0:0:9	89.66	0:11:21
PBnon-5q_5qBT	100	0:01:22	100	0:0:6	100	0:02:32
Ø	93.19	-	93.55	-	94.73	-

Table 8.3: Results of using Classic Primal SVM

8.1.3 Comparison of Primal and Dual results

Primal and Dual forms give similar results. The accuracy of created models is quite high as expected. The best classification of both methods was on merged data set. Methods are expected to compute similar results. However each method uses different variables to find optimal solution so weights of the separating can slightly differ.

8.2 SVM with Network-constrained L2-norm

Following sections present the results of using SVM with Network-constrained L2-norm written in Primal and Dual form on three data inputs. The inputs are:

1. only measured mRNA features constrained by protein-protein interactions (P-P)
2. measured both mRNA/miRNA (merged) features constrained by mirna-target interactions (miRNA-target)
3. measured both mRNA/miRNA (merged) features constrained by both protein-protein and mirna-target interactions (miRNA-target, P-P)

8.2.1 Dual SVM with Network-constrained L2-norm

The Table 8.5 contains accuracies and time consumptions (T) of models for different tasks and data inputs. The average memory consumptions are:

Data input	**Memory**
P-P	2GB
miRNA-target	2.5GB
miRNA-target, P-P	2.5GB

Table 8.4: Average memory consumption of models for different data inputs

Input →	**P-P**	T	**miRNA-target**	T	**miRNA-target, P-P**	T
Task ↓	Success(%)	h:m:s	Success(%)	h:m:s	Success(%)	h:m:s
BMBT_DT5q	64.66	0:21:45	92	0:35:38	92	0:26:22
BMH_ABT5q	100	1:42:55	100	2:23:24	100	1:25:57
BMH_ABTnon-5q	93.33	0:30:20	94.44	0:26:18	93.33	0:37:38
BMH_ADT5q	98.66	0:19:15	86.66	0:20:57	86.66	0:20:13
BMnon-5q_5qBT	81.66	0:32:31	100	0:31:15	100	0:42:16
PBBT_DT5q	83.33	1:57:29	92.22	1:51:03	92.96	2:15:06
PBH_ABT5q	100	0:59:47	100	1:12:18	100	1:46:19
PBH_ABTnon-5q	100	0:12:17	100	0:17:14	100	0:15:31
PBH_ADT5q	98.33	1:55:28	89.66	2:07:37	88.66	0:15:46
PBnon-5q_5qBT	100	0:09:39	100	0:10:49	100	0:10:45
Ø	92	-	95.5	-	95.36	-

Table 8.5: Results of using Dual SVM with Network-constraint L2-norm

8.2.2 Primal SVM with Network-constrained L2-norm

The Table 8.7 contains accuracies and time consumptions (T) of models for different tasks and data inputs. The average memory consumptions are:

Data input	Memory
P-P	37GB
miRNA-target	65GB
miRNA-target, P-P	192GB

Table 8.6: Average memory consumption of models for different data inputs

Input →	P-P	T	miRNA-target	T	miRNA-target, P-P	T
Task ↓	Success(%)	h:m:s	Success(%)	h:m:s	Success(%)	h:m:s
BMBT_DT5q	64.66	0:50:16	92	0:38:12	92	5:08:05
BMH_ABT5q	100	2:07:25	100	2:43:14	100	4:54:35
BMH_ABTnon-5q	93.33	1:23:24	93.33	1:19:07	93.33	3:43:23
BMH_ADT5q	98.66	2:44:43	86.66	1:02:23	86.66	6:15:12
BMnon-5q_5qBT	81.66	1:16:04	100	1:09:18	100	7:42:02
PBBT_DT5q	83.33	5:02:00	92.22	1:31:06	92.22	5:10:59
PBH_ABT5q	100	3:59:19	100	1:33:26	100	4:42:13
PBH_ABTnon-5q	100	0:38:40	100	1:05:12	100	5:29:14
PBH_ADT5q	98.66	1:57:53	89.66	2:10:36	88.66	10:22:48
PBnon-5q_5qBT	100	2:17:21	100	1:05:50	100	2:07:58
Ø	92.03	-	95.39	-	95.29	-

Table 8.7: Results of using Primal SVM with Network-constraint L2-norm

8.2.3 Comparison of Primal and Dual results

Primal and Dual forms give similar results. The accuracy of created models increased when compared to the Classic SVM in Section 8.1. Thus integrating interactions prior knowledge in decision models is reasonable.

8.3 SVM with Network-constrained L2-norm with Negative Regulatory Assumption

Following sections present the results of using SVM with Network-constrained L2-norm with Negative Regulatory Assumption written in Primal and Dual form on three data inputs. The inputs are:

1. only measured mRNA features constrained by protein-protein interactions (P-P)
2. measured both mRNA/miRNA (merged) features constrained by mirna-target interactions (miRNA-target)
3. measured both mRNA/miRNA (merged) features constrained by both protein-protein and mirna-target interactions (miRNA-target, P-P)

8.3.1 Dual SVM with Network-constrained L2-norm with Negative Regulatory Assumption

The Table 8.9 contains accuracies and time consumptions (T) of models for different tasks and data inputs. The average memory consumptions are:

Data input	**Memory**
P-P	2GB
miRNA-target	2GB
miRNA-target, P-P	2.5GB

Table 8.8: Average memory consumption of models for different data inputs

Input →	**P-P**	T	**miRNA-target**	T	**miRNA-target,P-P**	T
Task ↓	Success(%)	h:m:s	Success(%)	h:m:s	Success(%)	h:m:s
BMBT_DT5q	62	0:58:27	92	0:21:08	92	0:22:01
BMH_ABT5q	100	1:23:52	100	1:38:56	100	1:43:43
BMH_ABTnon-5q	92.22	0:24:45	94.44	0:31:38	93.33	0:30:52
BMH_ADT5q	98.66	0:16:49	86.66	0:22:25	86.66	0:21:55
BMnon-5q_5qBT	81.66	0:38:55	100	0:40:44	100	0:54:02
PBBT_DT5q	83.33	1:22:06	92.22	2:23:40	92.96	2:41:13
PBH_ABT5q	100	1:09:21	100	1:00:19	100	1:46:53
PBH_ABTnon-5q	100	0:11:15	100	0:13:06	100	0:18:48
PBH_ADT5q	97	1:40:23	89.66	1:53:05	88.66	2:21:35
PBnon-5q_5qBT	100	0:09:39	100	0:12:57	100	0:24:42
Ø	91.49	-	95.5	-	95.36	-

Table 8.9: Results of using Dual SVM with Network-constraint L2-norm with Negative Regulatory Assumption

8.3.2 Primal SVM with Network-constrained L2-norm with Negative Regulatory Assumption

The Table 8.11 contains accuracies and time consumptions (T) of models for different tasks and data inputs. The average memory consumptions are:

Data input	**Memory**
P-P	37GB
miRNA-target	65GB
miRNA-target, P-P	192GB

Table 8.10: Average memory consumption of models for different data inputs

Input → Task ↓	P-P	T	miRNA-target	T	miRNA-target, P-P	T
	Success(%)	h:m:s	Success(%)	h:m:s	Success(%)	h:m:s
BMBT_DT5q	62	0:46:28	92	1:31:18	92	2:39:18
BMH_ABT5q	100	1:27:45	100	2:12:44	100	5:18:42
BMH_ABTnon-5q	93.33	0:49:41	93.33	0:43:34	93.33	3:15:17
BMH_ADT5q	98.66	0:44:25	86.66	0:41:20	86.66	2:04:47
BMnon-5q_5qBT	78.33	0:57:25	100	0:43:40	100	2:46:04
PBBT_DT5q	83.33	1:18:43	92.22	0:56:59	92.96	8:35:00
PBH_ABT5q	100	1:13:29	100	1:06:43	100	8:52:23
PBH_ABTnon-5q	100	0:38:40	100	0:38:27	100	5:36:26
PBH_ADT5q	98	1:20:53	89.66	1:08:19	88.66	9:26:40
PBnon-5q_5qBT	100	0:36:27	100	0:36:12	100	4:54:36
Ø	91.37	-	95.39	-	95.36	-

Table 8.11: Results of using Primal SVM with Network-constraint L2-norm with Negative Regulatory Assumption

Results of the primal and dual form are very similar as is the time consumption. The computation of the dual form was expected to be faster. The reason of the reduced speed is the demanding matrix multiplications in the dual form.

8.3.3 Comparison of Primal and Dual results

Primal and Dual forms give similar results. The accuracies are very similar to the results of SVM with Network-Constrained L2-norm in Section 8.2. The negative regulatory assumption did not much effect the weights in the decision models.

8.4 Sparse L2 norm

Following section presents the results of using Sparse L2 norm on two data inputs. The inputs are:

1. only measured mRNA features (mRNA)
2. measured both mRNA/miRNA (merged)

The Table 8.13 contains accuracies and time consumptions (T) of models for different tasks and data inputs. The average memory consumptions are:

Data input	Memory
mRNA	6GB
merged	5GB

Table 8.12: Average memory consumption of models for different data inputs

Input → **Task ↓**	**mRNA** Sucess(%)	T h:m:s	**merged** Sucess(%)	T h:m:s
BMBT_DT5q	79.66	1:28:44	84.66	1:48:02
BMH_ABT5q	100	1:14:15	87.33	3:06:26
BMH_ABTnon-5q	85	0:45:31	91.11	1:03:17
BMH_ADT5q	100	0:36:43	81.33	0:51:33
BMnon-5q_5qBT	83.33	0:53:05	93.88	0:52:13
PBBT_DT5q	70	7:13:21	78.52	2:33:49
PBH_ABT5q	94.44	5:04:35	88.52	2:36:40
PBH_ABTnon-5q	100	1:40:58	100	0:45:53
PBH_ADT5q	94	5:57:55	81.66	2:28:36
PBnon-5q_5qBT	85.83	0:48:03	100	0:36:22
Ø	82.22	-	88.7	-

Table 8.13: Results of using classic Sparse Primal SVM

These results are going to be compared later in the thesis with other sparse models from section 8.5

8.5 Sparse Network-constrained L2 norm

Following sections present the results of using primal form of Sparse SVM with Network-constrained L2-norm and Sparse SVM with Network-constrained L2-norm with Negative Regulatory assumption on three data inputs. The inputs are:

1. only measured mRNA features constrained by protein-protein interactions (P-P)
2. measured both mRNA/miRNA (merged) features constrained by mirna-target interactions (miRNA-target)
3. measured both mRNA/miRNA (merged) features constrained by both protein-protein and mirna-target interactions (miRNA-target, P-P)

The memory consumptions of both sparse forms were:

Data input	**Memory**
P-P	67GB
miRNA-target	104GB
miRNA-target, P-P	258GB

Table 8.14: Average memory consumption of models for different data inputs

8.5.1 Primal SVM with Network-constrained L2-norm

The Table 8.15 contains accuracies and time consumptions (T) of models for different tasks and data inputs.

Input →	**P-P**	T	**miRNA-target**	T	**miRNA-target,P-P**	T
Task ↓	Sucess(%)	h:m:s	Sucess(%)	h:m:s	Sucess(%)	h:m:s
BMBT_DT5q	79.66	10:54:06	82.33	7:08:17	82.33	15:27:04
BMH_ABT5q	100	30:33:32	87.33	14:23:30	100	28:33:27
BMH_ABTnon-5q	85	15:03:40	91.11	8:00:34	91.11	30:23:33
BMH_ADT5q	98.66	10:53:16	81.33	5:44:47	81.33	21:43:15
BMnon-5q_5qBT	82.77	27:23:35	93.88	3:48:10	93.88	12:29:30
PBBT_DT5q	70	12:77:11	77.77	6:56:28	77.77	36:01:41
PBH_ABT5q	94.44	22:21:49	88.52	13:49:20	88.52	35:31:01
PBH_ABTnon-5q	100	8:22:33	100	2:31:51	100	12:09:37
PBH_ADT5q	94	15:53:50	81.66	13:43:52	81.66	27:48:35
PBnon-5q_5qBT	85.83	10:42:51	100	1:55:24	100	14:27:19
Ø	89	-	88.39	-	89.66	-

Table 8.15: Results of using Primal Sparse Network-constrained SVM

8.5.2 Primal SVM with Network-constrained L2-norm with Negative Regulatory Assumption

The Table 8.16 contains accuracies and time consumptions (T) of models for different tasks and data inputs.

Input →	**P-P**	T	**miRNA-target**	T	**miRNA-target,P-P**	T
Task ↓	Sucess(%)	h:m:s	Sucess(%)	h:m:s	Sucess(%)	h:m:s
BMBT_DT5q	79.66	7:52:19	82.33	6:22:46	82.33	14:08:10
BMH_ABT5q	100	17:23:39	87.33	14:16:10	87.33	31:20:55
BMH_ABTnon-5q	85	11:06:33	91.11	7:53:14	91.11	15:25:28
BMH_ADT5q	98.66	8:15:57	81.33	9:23:19	81.33	22:38:46
BMnon-5q_5qBT	82.77	18:11:37	93.88	6:10:44	93.88	27:59:39
PBBT_DT5q	70	27:04:22	77.77	11:23:15	77.77	8:35:0
PBH_ABT5q	94.44	15:03:07	88.52	11:16:50	88.52	21:41:05
PBH_ABTnon-5q	100	6:55:10	100	7:20:29	100	9:51:08
PBH_ADT5q	94	21:24:48	81.66	16:05:39	81.66	32:09:48
PBnon-5q_5qBT	85.83	8:05:05	100	3:59:48	100	9:23:02
Ø	89.01	-	88.39	-	88.39	-

Table 8.16: Results of using Primal Sparse Network-constrained SVM with Negative Regulatory Assumption

8.5.3 Comparison of Primal Sparse Network-constrained SVM and Primal Sparse Network-constrained SVM with Negative Regulatory Assumption results

Both models give very similar results. The accuracies decreased when compared to the models without the sparsity term mentioned in Sections 8.2, 8.3. However the accuracies increased when compared to the Sparse L2-norm without the interactions term in Section 8.4. This confirmed the statement about integrating interactions in the decision models.

8.6 Comparing Sparse models

This section compares Sparse SVM L2 norm, Sparse SVM with Network-constrained L2-norm and Sparse SVM with Network-constrained L2-norm with Negative Regulatory Assumption from the view of number of extracted weights and number of interactions. A variable defining density of interactions between selected features in models is presented in order to compare those models:

$$Density(task) = \frac{Average\ number\ of\ interactions\ in\ a\ model}{Average\ number\ of\ features\ in\ a\ model} \tag{8.1}$$

The Table 8.17 represents average number of features, average densities and average accuracy of chosen forms across all computed tasks.

Forms	Classic		$(w_i - w_j)^2$			$(w_i + w_j)^2$		
Datasets + interactions	mRNA	merged	mRNA + PPI	merged + miRNA-T	merged + PPI +miRNA-T	mRNA + PPI	merged + miRNA-T	merged + PPI +miRNA-T
Accuracy(%)	82.22	88.7	89	88.39	89.66	89.01	88.39	88.39
Features(Ø)	6.34	6.15	8.014	22.82	32.97	11.19	23.95	27.35
Density(Ø)	0.01	0.002	0.14	0.66	0.8	0.16	0.81	0.67

Table 8.17: Compared sparse models.

It can be seen that the accuracy of evaluating samples with mRNA features constrained by interactions increased by 6.78% due to the network constraints. The accuracy of evaluating samples with merged features did

not change much but the density of the network of selected features increased. On the samples with mRNA features density increased by 0.14 interactions/feature. On the samples with merged features density increased by up to 0.8 interactions/feature.

8.7 Updating the sparsity term in Sparse Primal SVM with Network-constrained L2-norm

The Sparse Primal SVM with Network-constrained L2-norm tries to maximally minimize the number of weights of the model. However, sometimes the number of features could be extremely low. Preserving more weights can be done by multiplying the sparsity term by λ. The following results were gained from models created on task BMBT-DT5q on dataset containing only mRNA features and constrained by protein-protein interactions.

$$\begin{aligned} \underset{w_0,w_1 \ldots w_m}{min} \left\{ \lambda \sum_{i=1}^{M} |w_i| + \frac{1}{2} \sum_{i=1}^{M} w_i^2 + \frac{1}{2} \beta \sum_{ij \in \tau} A_{ij} \left(w_i - w_j \right)^2 \right\}, \\ s.t. : s.t. : \left(\mathbf{w}^T \mathbf{x_i} + w_0 \right) y_i \geq 1, \forall i = 1, \ldots N \end{aligned} \tag{8.2}$$

The importance of the sparsity term decreased and thus the number of weights in created sparse models increased. In Table 8.18 are presented results of models with different values of λ. Each variable setting was tested on 5-times repeated stratified x-validation and thus multiple models with different number of features were created. The column "Min. features" represents the minimal number of features in a model (among all models belonging to the setting). The column "Max. features" represents the maximal number of features in a model. The column "Average features" represent average number of selected features among all the models. "Accuracy" represents average accuracy among all the models. The "B" task in Table 8.18 refers to "BMBT_DT5q" task and "P" refers to "PBBT_DT5q" task.

λ	Min. features		Max. features		Average features		Accuracy(%)	
	B	P	B	P	B	P	B	P
1	5	9	22	17	7.53	12.33	79.66	70
0.1	4	9	155	91	15.12	14.17	83.66	71.48
0.01	6	8	784	1180	71.56	40.31	72	72.96
0.001	28	-	3052	-	231.2	-	67	-
0.0001	359	362	7958	7649	1019.12	989.31	68.33	81.85
0.00001	3023	3114	9356	13303	4354.32	4138	68.33	86.66
No sparsity term	16653	16651	16664	16662	16660.32	16656	64.66	83.33

Table 8.18: Results of different λ settings.

From Table 8.18 can be seen that the average number of features changes similarly on both tasks with different λ settings. The task "BMBT_DT5q" obtains samples which are more difficult to separate. The sparsity term filters out the non-relevant features and thus the accuracy increases by 15%.

8.8 Updating the network strength parameter in Sparse Primal SVM with Network-constrained L2-norm

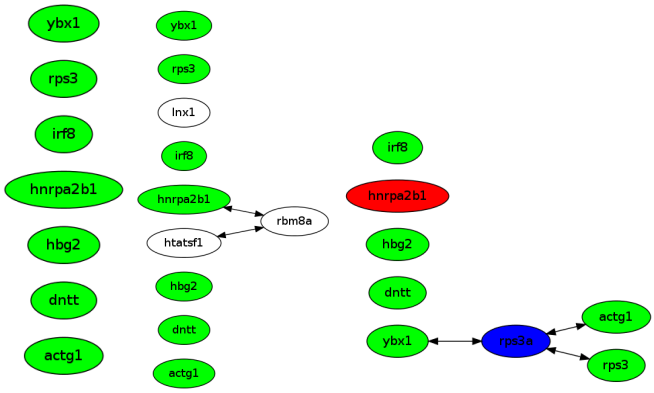
The number of selected features in sparse models does not depend only on the λ parameter. It can also be updated by the network strength parameter β. The following results were gained from models created on task BMBT-DT5q on dataset containing only mRNA features and constrained by protein-protein interactions.

β	Accuracy(%)	
	B	P
2	79.66	70
3	79.66	70
5	79.66	70
10	82.33	71.11
20	83.66	70.74
30	83.66	72.59
50	79.66	79.25
100	78.33	75.18

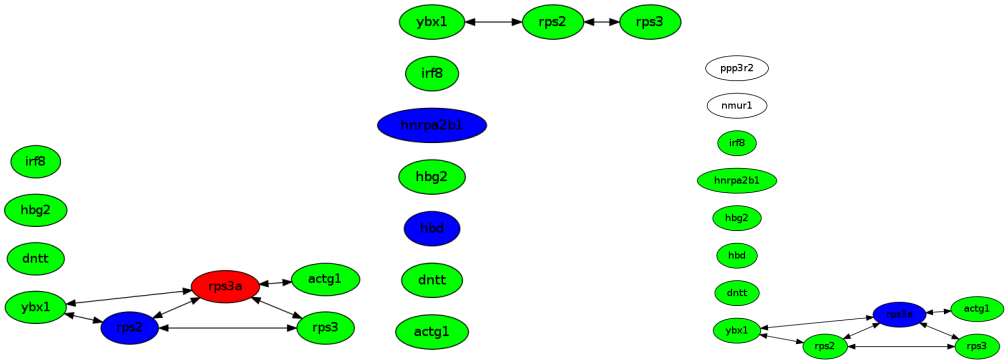
Table 8.19: Results of different β settings.

Following figures represent changes in the interaction network when increasing the strength parameter. Colors of the nodes represent:

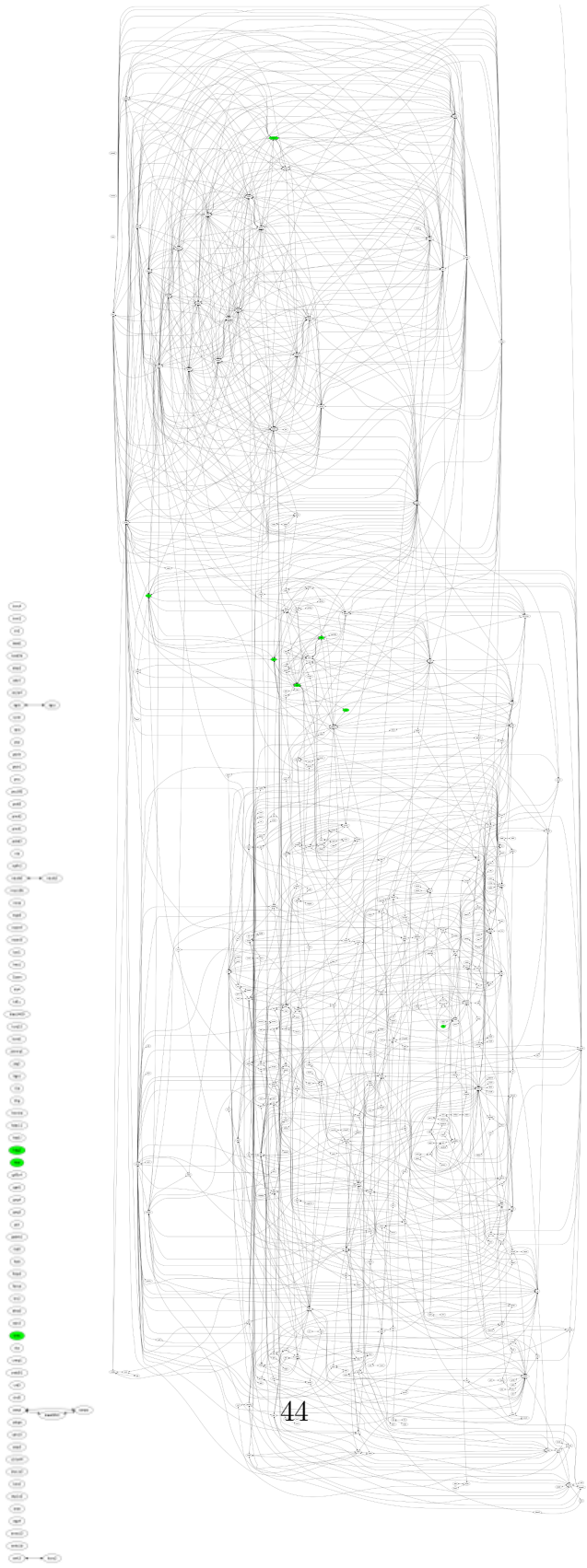
- Green - Feature appears in this models and also in the model with β one level higher and lower.
- Blue - Feature appears in this model and also in the model with β one level higher.
- Red - Feature appears in this model and also in the model with β one level lower.
- White - Feature appears only in this model.



(a) $\beta = 2$ (b) $\beta = 3$ (c) $\beta = 5$

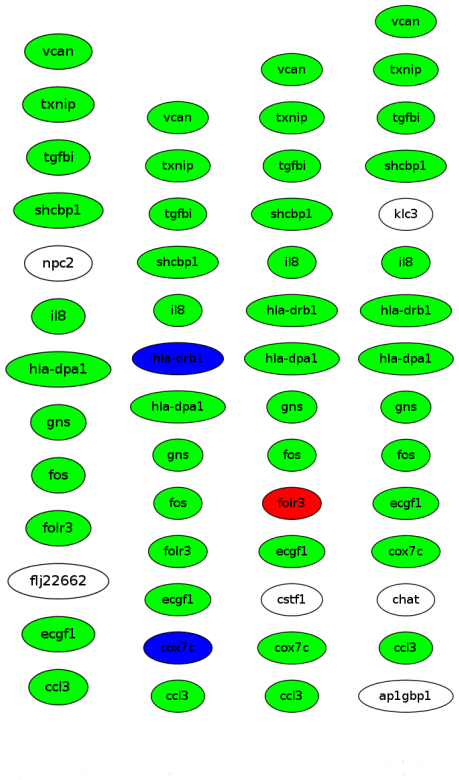


(d) $\beta = 10$ (e) $\beta = 20$ (f) $\beta = 30$

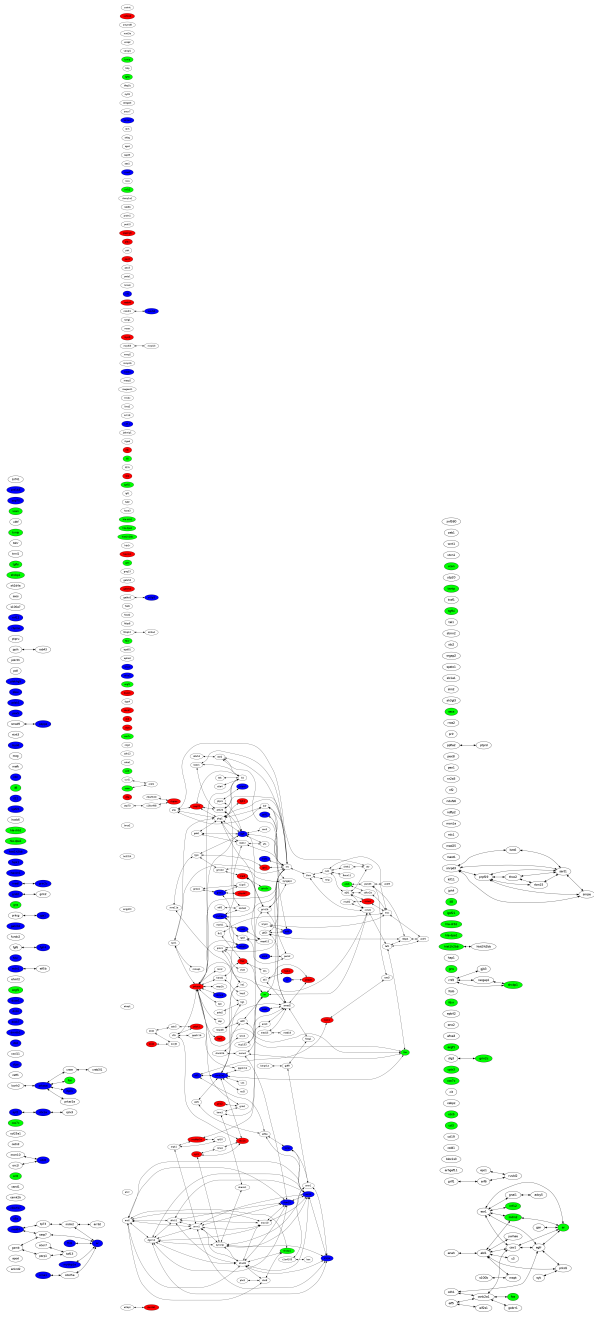


(g) $\beta = 50$

Figure 8.1: BMBT-DT5q task with mRNA features, protein-protein interactions



(a) $\beta = 2$ (b) $\beta = 3$ (c) $\beta = 5$ (d) $\beta = 10$



(e) $\beta = 20$ (f) $\beta = 30$ (g) $\beta = 50$

From the figures above is apparent that setting of the parameter β can much effect the density of the network. The following happens when setting sufficiently big strength of the network:

1. Weights of the interacting features increase due to the network-constrained term $\frac{1}{2}\beta \sum_{ij \in \tau} A_{ij} \left(w_i - w_j\right)^2$.

2. The sparsity term tries to minimize the number of selected features and keep the optimal separation of the sample space.

3. Thus different weights could be chosen in order to separate samples by constraints $y_i \left(\mathbf{w} \cdot \mathbf{x_i} + b\right) \geq 1$ when using different network force settings. (the weights of some important features grow too much because of the interactions and thus are omitted in the classification model because of minimizing the term $\frac{1}{2}\sum_{i=1}^{M} w_i^2$)

4. The most important features which separate the samples are preserved because their weights do not increase much by increasing the network strength parameter. This is visible on models with $\beta = 1$ and $\beta = 50$ where most of the features from the first model remain. The final model consists of those preserved features and their context which is added in order to separate the samples.

By updating the strength of the network can be manipulated both the number of interactions per feature in a decision model and the number of features. The accuracy of models of task "BMBT_DT5q" stays almost the same while increasing the density of the network. However the accuracy of models of "PBBT_DT5q" task increases when increasing the strength parameter, until some level. From the network-interaction graphs of task "PBBT_DT5q" It can be seen that the network gets denser by increasing the strength parameter. However the similarity among individual graphs is not that high (visible on $\beta = 20, 30, 40$ where are lots of white nodes).

8.9 Network-norm-SVM

The Network-norm-SVM could not be modeled in dual form because the Laplacian matrix is not singular and thus could not be inverted.

8.10 Artificial data

This section presents accuracies of models tested and trained on artificial data. Four types of classifiers were used on the artificial data. All of them are in the Primal form of SVM. First pair is Classic SVM and Network-constrained SVM. Both of them are trying to find weights for all the features in the data set. Second pair is Sparse Classic SVM, which is Classic SVM with the sparsity term, and Sparse Network-constrained SVM, which is SVM with both the sparsity and the network-constraint terms.

Form	Accuracy(%)	Features	Density
Classic SVM	73.8	1565	-
Network-constrained SVM	70.01	1565	-
Sparse Classic SVM	78.85	156.08	0.139
Sparse Network-constrained SVM	74.36	306.66	1.006

Table 8.20: Classification on artificial data.

The Sparse Classic SVM was the most successful when classifying the testing samples according to the Table 8.20. However there are not many interactions among the selected features the model is deciding upon.

The accuracy of Sparse Network-consrained SVM accuracy lowered by 4.49% when compared to the Sparse Classic SVM. However, the density of this model is one interaction per feature which is high when considering the doubled number of features in the decision model.

Chapter 9

Conclusion

This chapter summarizes the results of the regularizations of the SVM algorithm and their contribution in the terms of the required qualities which are accuracy, stability, interpret-ability and efficiency.

9.1 Accuracy

The accuracy can be influenced by both interaction term and the sparsity term. Regularizing SVM with the network interactions slightly improves the average accuracy on of Classic SVM on different data sets (93.8%). The sparsity term significantly decreases the accuracy (85.5). The combination of the Sparse and Network-constraint term is the golden mean with accuracy (89%).

9.2 Stability

GE files contain low number of samples thus it is difficult to reason about the stability of the models. However all the forms of SVM were trained/tested by 5-times repeated stratified x-validation. That means 5x(4-10)x10 models were created for each data set. 5 represents number of iterations, (4-10) represents number of folds which is at least 4 and maximally 10, 10 represents number of tasks. Considering this amount of models, each form of SVM achieved at least average accuracy 80%. Thus using regularized forms of SVM are considered to be stable.

9.3 Interpret-ability

The interpret-ability describes how much do the features in the decision models interact together and how comprehensible are the rules which decide about the class of the sample.

The interpret-ability of SVM with only the interaction term is low because the final decision model contains all the features, including the noise. Contrarily features in the decision models of SVM with only sparsity term are not interconnected at all. The average density of those decision models is 0.006 interactions per feature. The high interpret-ability is achieved by combining the interaction and the sparsity term. Then the final models contain in average 0.53 interactions per feature. The interpret-ability can be further enhanced by setting the strength of the interactions term β, which imposes higher importance on interacting features.

However the most important features which well separate the samples do not interact much with other features. Thus enhancing β only imposes to select more interacting context of features to those which are quite independent and important.

9.4 Efficiency

The biggest disadvantage of computing the Sparse network-constrained L2-norm SVN is its efficiency. The average time-consumption for running 5-times repeated stratified x-validation on a specific data set is 16 hours. The average time-consumption of Classic SVM is 5 minutes. The regularized form has also high memory-consumption. The memory needed for 5-times repeated stratified x-validation reaches up to 258GB. The Classic SVM needs maximally 1GB. The computation resources make it almost impossible to compute regularized forms on a PC.

9.5 Comparison to other experiments on omics data

Comparing regularized SVM to other highly interpretable models is very important. One of them is Network Constrained Forest (NFC) [8]. NCF

creates trees including the feature interactions and then classifies the samples. The average accuracy of NFC on MDS tasks is 78%.

The other is SVD-based aggregation (SVDba) [18]. Its aim in general, is to reduce the mRNA vectors and their respective targeting miRNAs into one aggregated feature. The accuracy on MDS data is 85.22%.

Sparse network-constrained L2-norm SVM outperforms the other algorithms in accuracy. (89%). However NFC and SVDba are more interpretable since the their classification models obtain concrete value thresholds for individual GE. By way of contrast, SVM models obtain only computed weights.

9.6 Future work

The approach explained in this thesis seems quite reasonable according to the presented results. However the Sparse network-constrained L2-norm SVN could be improved by properly setting the network strength parameter and the sparsity importance term parameter. Finding general relation between these two parameters could lead to more accurate and interpretable models.

Acknowledgements

Special thanks belongs to below mentioned institutions.

Metacentrum

Computational resources were provided by the MetaCentrum under the program LM2010005 and the CERIT-SC under the program Centre CERIT Scientific Cloud, part of the Operational Program Research and Development for Innovations, Reg. no. CZ.1.05/3.2.00/08.0144.

IBM Academic Initiative

Free access to IBM ILOG CPLEX Optimizer was provided under IBM Academic Initiative.

Bibliography

[1] IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer.

[2] Pandas. http://pandas.pydata.org/.

[3] Picos. http://picos.zib.de/.

[4] Pycpx. http://www.stat.washington.edu/~hoytak/code/pycpx/.

[5] *LAPACK: A portable linear algebra library for high-performance computers*, 1990. http://dx.doi.org/10.1109/superc.1990.129995.

[6] Intel Math Kernel Library, 2007. http://www.intel.com/cd/software/products/asmo-na/eng/307757.html.

[7] GeneNetWeaver: In silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*, 27(16):2263–2270, June 2011.

[8] M. Anděl, J. Kléma, and Z. Krejčík. Network-constrained forest for regularized classification of omics data. *Methods*, 2015.

[9] M. Anděl and F. Masri. Sparse omics-network regularization, to increase interpretability and performance, of svm-based predictive models. 2015.

[10] D. J. Barrett and R. E. Silverman. *SSH, The Secure Shell: The Definitive Guide*. 2001.

[11] A. D. Baxevanis and F. Ouellette. *Bioinformatics : a practical guide to the analysis of genes and proteins*. 1998.

[12] L. S. Blackford, J. J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley. An updated set of basic linear algebra subprograms (blas). *ACM Transactions on Mathematical Software*, 28:135–151, 2001.

[13] A Bossi and B Lehner. Tissue specificity and the human protein interaction network. *Mol Syst Biol*, 5, 2009.

[14] H. Dweep, C. Sticht, P. Pandey, and N. Gretz. mirwalk – database: Prediction of possible mirna binding sites by "walking" the genes of three genomes. *Journal of Biomedical Informatics*, 44(5):839 – 847, 2011.

[15] A. Frank and A. Asuncion. UCI machine learning repository, 2010. `http://archive.ics.uci.edu/ml`.

[16] T. M. Huang and V. Kecman. Bias term b in svms again. pages 441–448, 2004.

[17] T.S. Keshava Prasad, R. Goel, K. Kandasamy, S. Keerthikumar, S. Kumar, S. Mathivanan, D. Telikicherla, R. Raju, B. Shafreen, A. Venugopal, L. Balakrishnan, A. Marimuthu, S. Banerjee, D.S. Somanathan, A. Sebastian, S. Rani, S. Ray, C.J. Harrys Kishore, S. Kanth, M. Ahmed, M.K. Kashyap, R. Mohmood, Y.I. Ramachandra, V. Krishna, B.A. Rahiman, S. Mohan, P. Ranganathan, S. Ramabadran, R. Chaerkady, and A. Pandey. Human protein reference database - 2009 update. *Nucleic Acids Research*, 37(1):767–772, 2009.

[18] J. Kléma, J. Zahálka, M. Andel, and Z. Krejcık. Knowledge-based subtractive integration of mrna and mirna expression profiles to differentiate myelodysplastic syndrome. 2014.

[19] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. 14(2):1137–1145, 1995.

[20] O. Lavi, G. Dror, and R. Shamir. Network-induced classification kernels for gene expression profile analysis. *Journal of Computational Biology*, 19(6):694–709, 2012.

[21] MATLAB. *version 8.2.0 (R2013b)*. The MathWorks Inc., 2013.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[23] The scipy community. *NumPy*. `http://docs.scipy.org/doc/numpy/reference/`.

[24] M. Tyers and M. Mann. From genomics to proteomics. *Nature*, 422(6928):193–197, 2003.

[25] V. N. Vapnik. *Statistical learning theory*, volume 2. 1998.

[26] T. Vergoulis, I. S. Vlachos, P. Alexiou, G. Georgakilas, M. Maragkakis, M. Reczko, S. Gerangelos, N. Koziris, T. Dalamagas, and A. G. Hatzigeorgiou. TarBase 6.0: capturing the exponential growth of miRNA targets with experimental support. *Nucleic acids research*, 40(Database issue):D222–D229, January 2012.

[27] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. *Advances in neural information processing systems*, 16(1):49–56, 2004.

Appendices

Appendix A

CD

Attached CD contains datasets, interactions files, bash scripts, python scripts, MATLAB scripts, outputs of experiments and graphs of interactions of created models.

- /bashScripts - scripts for executing programs on computation nodes
  - /matlabProgramLaunchers
  - /pythonProgramLaunchers
- /datasets - files containing GE and miRNA expressions
  - /artificialData
  - /codeVerification
  - /MDS
- /interactions - files containing P-P/miRNA-target interactions
  - /artificialInteractions
  - /mdsInteractions -
- /matlabScripts - scripts for generating interaction matrices
- /modelOutputs - results of classification models
  - 2dPoints
  - codeVerification

  - MDS

- /networkGraphs - visualized interactions among features from classification models
- /pythonScripts - classifier implementation
- /thesis - source codes of the thesis and its PDF