# Inferring Regulatory Networks

Diplomarbeit in Bioinformatics by
Sebastian J. Schultheiß

*Advisor (Computer Science)*
Prof. Dr.-Ing. Oliver Kohlbacher
Eberhard Karls University of Tübingen
Wilhelm Schickard Institute for Computer Science
Division for Simulation of Biological Systems
Sand 14, 72076 Tübingen, Germany
oliver.kohlbacher@uni-tuebingen.de

*Advisor (Biology)*
Prof. Dr. Detlef Weigel
Max Planck Institute for Developmental Biology
Department VI – Molecular Biology
Spemannstr. 35, 72076 Tübingen, Germany
weigel@weigelworld.org

*Author*
Sebastian J. Schultheiß
Matrikel-Nr. 2343503
Kirschenweg 12, 72076 Tübingen, Germany
sebastian.schultheiss@tuebingen.mpg.de

Winter Semester 2006/2007
Tübingen, May 2007

**Abstract**

The goal of this thesis is to find transcription factor binding site motifs through the use of microarray expression data, protein-DNA binding data from chromatin immunoprecipitation on a microarray chip, and conservation of the genomic sequence among species.

The techniques described here were applied to data from *Arabidopsis thaliana*. In gene lists obtained using different techniques, over-represented sequence motifs were searched using Gibbs sampling algorithms in well-tested implementations. From these genes, at least some have to have their expression directly regulated by a transcription factor, and their promoter region will contain a binding site for it. This is a measurable property of the sequence that can be learned by a machine learning algorithm.

Using expression profiles of the genes as labels, a support vector machine with an integrated kernel learned to distinguish between regulated and unregulated genes on the basis of differences in the motifs, in terms of the motif's surrounding sequence, distribution, relative distance to the transcription start, pairwise distance, conservation in related *plantae*, and other sequence characteristics.

This information can be used to infer a network of genes responding to the change in expression of a transcription factor, if the logic of the experiments is taken into account. With the motif now known we can go on to analyze other genes that have this motif and may be part of the network, but were not contained in the original list because, for instance, their expression level was below the microarray sensitivity or they were not present on the chip. Network creation ultimately remains a manual process, because modeling the experimenter's rationale in designing these experiments is a difficult enough task on its own.

To test the performance of the selected approach, we try to predict an initial network of genes responding to heat shock, a response that is well characterized and the primary response genes' regulatory motif is known. The work flow proposed is able to find this motif and ranks it among the top three for the heat shock microarray data.

The approach was also trained on data obtained from loss- and gain of function experiments centered around the transcription factor WUSCHEL. The results on this original research data show that we not only find direct targets of transcription factors, but must assume that most alterations in expression we observe are a secondary response provoked by expression change in transcription factors regulated directly. Secondary targets are difficult to discriminate without additional data, so we introduce a method to rank genes by their being bound by a transcription factor. For this, we use data from chromatin immunoprecipitation on a chip. The results for WUSCHEL point out likely targets of primary and secondary regulation, including proposals for a binding motif.

The approach shows a lot of promise as it performs as expected on data with known results; confirmation for the WUSCHEL transcription factor targets will be done experimentally on the basis of the list obtained through this method.

# Contents


**1 Introduction** **7**

1.1 Overview . . . . . 8

1.2 Machine Learning for Detecting Co-Regulation . . . . . 8

1.2.1 Composition of Input Vectors . . . . . 9

1.2.2 Sequence window and TFBS Motifs from Gibbs Sampling . 9

1.2.3 Alignment of *Arabidopsis lyrata*, *Populus trichocarpa*, and *Oryza sativa* with *Arabidopsis thaliana* . . . . . 10

1.2.4 Results of SVM Training . . . . . 10

1.3 ChIP-on-chip Binding Data for WUSCHEL . . . . . 11

1.3.1 Support Vector Regression Training . . . . . 12

1.4 Structure of this Work . . . . . 15

**2 Background** **16**

2.1 Biological Context . . . . . 16

2.1.1 Transcription and Regulation . . . . . 16

2.1.2 Transcription Factor Binding Sites . . . . . 17

2.1.3 Microarrays . . . . . 18

2.1.4 Chromatin Immuniprecipiation on a Chip . . . . . 19

2.2 Machine Learning . . . . . 21

2.2.1 Support Vector Machines . . . . . 22

2.2.2 Support Vector Regression . . . . . 27

2.3 Statistical Motif Search . . . . . 28

2.3.1 Data Preparation . . . . . 28

2.3.2 Gibbs Sampling Algorithm . . . . . 29

**3 Materials and Methods** **32**
3.1 Microarray Experiments . . . . . 32
3.1.1 Expression Data . . . . . 32
3.1.2 Gene Lists . . . . . 35
3.2 ChIP-on-chip Data . . . . . 38
3.3 Whole Genome Alignment . . . . . 40
3.4 Shogun Large Scale Machine Learning Toolbox . . . . . 41
3.5 Python Work Flows . . . . . 43
3.5.1 UML diagram specifics . . . . . 43
3.5.2 Classes of the Python Module `ExpressionAnalysis` . . . 44
3.5.3 Classes of the `ChIPchipSVR` analysis pipeline . . . . . 46
3.5.4 Additional Scripts . . . . . 46
3.5.5 Third Party Modules . . . . . 47

**4 Results** **52**
4.1 SVM Training Results . . . . . 52
4.2 SVR Training Results . . . . . 55
4.3 Use Case – Putting Results Together . . . . . 56
4.4 TFBS Candidate Motifs . . . . . 58

**5 Discussion** **60**
5.1 Related Work . . . . . 60
5.1.1 Support Vector Machines for Detecting Co-Regulation . . . 61
5.1.2 Novel Contributions in this Work . . . . . 61
5.1.3 Motif Finding Programs . . . . . 62
5.2 Inferring a Network . . . . . 62
5.3 Comment . . . . . 63
5.4 Outlook . . . . . 63
5.4.1 Future of this implementation . . . . . 63
5.4.2 Regulatory Networks . . . . . 64

**Bibliography** **69**

# List of Figures


1.1 A drawing of a flowering *Arabidopsis thaliana* specimen [1]. . . . 7
1.2 A regulatory network showing the influence of WUSCHEL. . . . . 12
1.3 Binding of DNA previously bound by WUSCHEL. . . . . 13

2.1 Affymetrix GeneChip . . . . . 19
2.2 Cross-linking of DNA and proteins using formaldehyde. . . . . 20
2.3 Deciding on hyperplanes in a support vector machine. . . . . 23
2.4 Example of a hyperplane in a support vector machine. . . . . 24
2.5 Example of a support vector regression. . . . . 28

3.1 Heat map of grid search results. . . . . 48
3.2 Class diagram of the Python package `ExpressionAnalysis` . . 49
3.3 Work flow for the SVM training . . . . . 50
3.4 Components of the python package `ChIPchipSVR` . . . . . 51

4.1 Comparison of SVM performance with and without the use of conservation data. . . . . 55
4.2 Comparison of SVR ranked to genes randomly assigned a WRS in a running average. . . . . 57
4.3 Overall work flow of the use case . . . . . 59

# List of Tables

2.1 Confusion matrix of correct and incorrect predictions [45]. . . . . 25
2.2 Nucleotide frequencies and a transition matrix. . . . . 30

3.1 Microarray expression experiments . . . . . 33
3.3 Gene lists extracted from the microarray experiments in Table 3.1 36
3.5 Genome alignment similarity percentage. . . . . 41

4.1 SVM training results for gene lists from Table 3.3. . . . . 52
4.3 19 genes used for SVR parameter optimization . . . . . 56
4.4 Two candidate motifs from WUSCHEL targets . . . . . 58

# Abbreviations

| | |
|---|---|
| AGI | Arabidopsis Genome Initiative |
| DNA | Deoxyribonucleic Acid |
| kb | one thousand bases (kilo base) |
| Mb | one million bases (mega base) |
| MSA | Multiple Sequence Alignment |
| PWM | Position Weight Matrix |
| SVM | Support Vector Machine |
| SVR | Support Vector Regression |
| TAIR | The Arabidopsis Information Resource |
| WRS | Wuschel Regulatory Score |

# Chapter 1

# Introduction

The goal of this thesis is to find transcription factor binding site motifs through the use of expression data from microarray experiments, protein-DNA binding data from chromatin immunoprecipitation on a microarray chip, and comparison to the genomic DNA sequence of related species through homology and conservation. The techniques developed for this thesis were applied to data from *Arabidopsis thaliana*, but they can be transferred to other species where similar data are available. A drawing of the model plant organism *Arabidopsis thaliana* can be seen in Figure 1.1.

Figure 1.1: A drawing of a flowering *Arabidopsis thaliana* specimen [1].



## 1.1 Overview

To a large extent, the work flow of this thesis has been implemented in Python, a powerful, open source scripting language [2] popular particularly in the field of bioinformatics [3]. The pipeline developed is available internally to the researchers involved with this project and their collaborators, but is scheduled to be released as a web based annotation service, once the issue of computing power will be solved.

The availability of a large number of sequenced genomes has created a demand for solutions to computationally interesting and challenging questions. It may be a disillusioning fact that a sequence itself is not the key to understanding every last detail, but only a step on the way. It is still necessary to unlock the full potential of a sequenced genome by using methods that take into account other data and thereby help to discern the parts of a genome that are coding for proteins from those which are not, or the parts that regulate the transcription of a protein from those which do not. This thesis will try to shed some light on the latter problem by identifying the binding sites of transcription factors in the genome of *Arabidopsis thaliana* and inferring a regulatory network.

From microarray expression experiments, we can learn about the expression levels of certain genes when a transcription factor is *e.g.* over-expressed. From these genes, at least some are bound to have their expression directly regulated by this transcription factor and the sequence of their regulatory region will have measurable properties: a binding site for this transcription factor. If we learn the motif of these binding sites, we can go on to analyze other genes, where the expression levels are too low for detection through microarray experiments or where simply no expression data exists. We can then infer a network where the transcription factor is connected to all the genes whose expression it directly activates or represses. A secondary response is harder to detect, *i.e.* if one of the activated genes is itself a transcription factor, its up-regulation may in turn trigger further expression changes in other genes. This is called a secondary response and without time series experiments it will be much more difficult to detect.

## 1.2 Machine Learning for Detecting Co-Regulation

In addition to sequence data, which has made some of the methods described in this thesis possible in the first place, many biological assays create data that needs to be analyzed in completely new ways, although they still contain a lot of potential. This thesis tries to exploit the potential of all the data involved to find the answer to the question, “Which genes are co-regulated?”

By letting a machine learning algorithm determine the significance of each part of the data we try to find this answer. The machine learning package used in this thesis allows the user to train it and obtain results, but then also to go back and take a look deep inside its learned parameters to find out which parts of the data actually contributed to the classification, and which were ignored.

Support vector machines (SVM) are a well-established machine learning method introduced by Boser, Guyon, and Vapnik in 1992 [4]. An SVM maximizes the so-

called margin between two classes of input vectors that were given to it as a training set. The algorithm looks for the maximal distance between the two classes and puts the hyperplane, which divides the two classes during testing later on, right in the middle between them. This is done in the hopes of giving the SVM the ability to later generalize on the testing data.

### 1.2.1 Composition of Input Vectors

In this thesis, the input vectors of the SVM contain the following data:

1. the part of the upstream regulatory region of a gene that contains a motif, in a 20 base pair window,
2. the conservation of this window in one or all of a number of related plants,
3. the relative position of the motifs among each other and relative to the transcription start, and
4. a score of the particular site the motif was found in.

### 1.2.2 Sequence window and TFBS Motifs from Gibbs Sampling

The first part of the vector, the 20 base pair window, was obtained simply from excising $\pm 10$ characters from the genome sequence of *Arabidopsis thaliana*, which is readily available online [5] and which was entered into a relational database based on MySQL [6] for the purpose of this thesis. The motifs were obtained using one of several Gibbs sampling algorithms, some of which were available as source code and could be incorporated into the Python work flow. Separate classes exist to drive each of the algorithms with their specific requirements for input data.

The motifs were searched for in a list of genes that are thought to be co-regulated. The underlying idea is that among the sequences in these lists, the ones regulated by the transcription factor that was over-expressed or knocked out during the experiment will be over-represented, as their expression level will be changed most dramatically, and consequently the binding site this transcription factor recognizes will be over-represented in the extracted regulatory regions.

The gene lists were created from microarray expression experiments conducted using the Affymetrix GeneChip Arabidopsis ATH1 genome array. The expression level of the control is subtracted in log scale from the expression in the experiment and so an $n$-fold change can be assigned (*cf.* Section 3.1 of this thesis). A selection of threshold fold changes result in lists of different length and stringency, and if combined or intersected with other lists depending on the logic of performed experiments, noise is reduced as well (see Table 3.3 for details on the gene lists).

The lists of genes that reacted in a certain experiment serve as a filter for the sequence extraction. Only the upstream regulatory regions of the genes listed are selected from the data base table and given to the Gibbs motif samplers as input data. Some versions also require homologous sequences from related

organisms [7], a background nucleotide frequency matrix (see Table 2.2a for an example) [8], or some sequences that are not co-regulated as a negative sample [9].

A Python script summarizes the output of the Gibbs samplers in a custom data format that lists the position, score, and length of each motif. The input vector for the SVM is filled with this information and with a 20 base pair window around the center of the motif site. Together with the score of the motif and the respective positions that are also recorded in the custom data format, this constitutes parts 1, 3, and 4 of the SVM input vectors. Part 2 is described in 1.2.3.

### 1.2.3 Alignment of *Arabidopsis lyrata*, *Populus trichocarpa*, and *Oryza sativa* with *Arabidopsis thaliana*

For homology and conservation comparisons of the regulatory region, we aligned the genome of *Arabidopsis thaliana* with the current releases of the genomes of *Oryza sativa*, commonly known as rice, and *Arabidopsis lyrata*, one of the closest relatives of *Arabidopsis thaliana*. The alignment was computed in cooperation with Dr. Gunnar Rätsch of the Friedrich Miescher Laboratory of the Max Planck Society in Tübingen, Germany.

We used BLASTZ [10], a modified version of the gapped BLAST algorithm [11] suitable for whole genome alignments, after a recommendation of Prof. Daniel Huson of the University of Tübingen [12]. BLAST stands for basic local alignment search tool. It was developed for the alignment of the sequences of human and mouse, both genomes being significantly larger as the ones we have aligned. Even though BLASTZ should perform fast on large data sets, it still took about three weeks of computing time until it was finished. The alignments of rice and the two *Arabidopsis* species was subsequently merged into a multiple sequence alignment (MSA) with the threaded blockset aligner [13], indexed by the *Arabidopsis thaliana* chromosomal coordinates. The alignment with *Populus trichocarpa* or poplar [14] was not completed in time to be included in the MSA, but the conservation information was used independently of the other two species, rice and *Arabidopsis lyrata*.

We stored the completed alignments in a matrix data structure that contained each nucleotide of the *Arabidopsis thaliana* genome, indexed by its chromosomal coordinates, a score, and the corresponding gap or nucleotide, be it a match or a mis-match, in the aligned sequences (Section 3.3). This data is part of the input vectors, and the SVM uses this information to a large extent (*cf.* Chapter 4). This was expected, as most researchers looking for transcription factor binding sites ascribe a lot of importance to conservation in related species [15]. Recent studies in *Saccharomyces cerevisiae* show that about 50% of all regulatory elements are conserved among closely related species [16].

### 1.2.4 Results of SVM Training

The trained kernels were able to discern genes that were co-regulated from those that were not, with a score measured as area under the receiver operator

characteristic curve (see Section 2.2 for definitions). This score ranges from 0 to 1 and any number greater than 0.5 signifies a performance better than randomly guessing which class a certain input vector belongs to. The performance of the trained SVMs ranged from 0.60 to 0.85, with the smaller gene lists performing better than the larger ones, this is due to the reduced noise and added information that more stringent lists contain.

The performance was measured first and foremost on a reference data set, a number of gene lists obtained from a heat shock microarray experiment that contains the expression levels of plant tissue that was treated with 38°C medium for a short time. One hour after exposure to heat the expression is measured. The principal heat shock transcription factor's binding site is known [17], and this was used as a reference. If the predictor finds the element and puts the genes that carry it into the correct class, the work flow was considered successful. The same was done for a cold stress microarray experiment, and in both cases, the method worked as expected. See Chapter 3 for experimental details and Chapter 4 for results.

This was a test run for the work flow; the biologists I was working with are most interested in the regulatory network of the transcription factor WUSCHEL, which is involved with stem cell control and maintenance. We applied the work flow to the experiments, but here its results can only partially be confirmed.

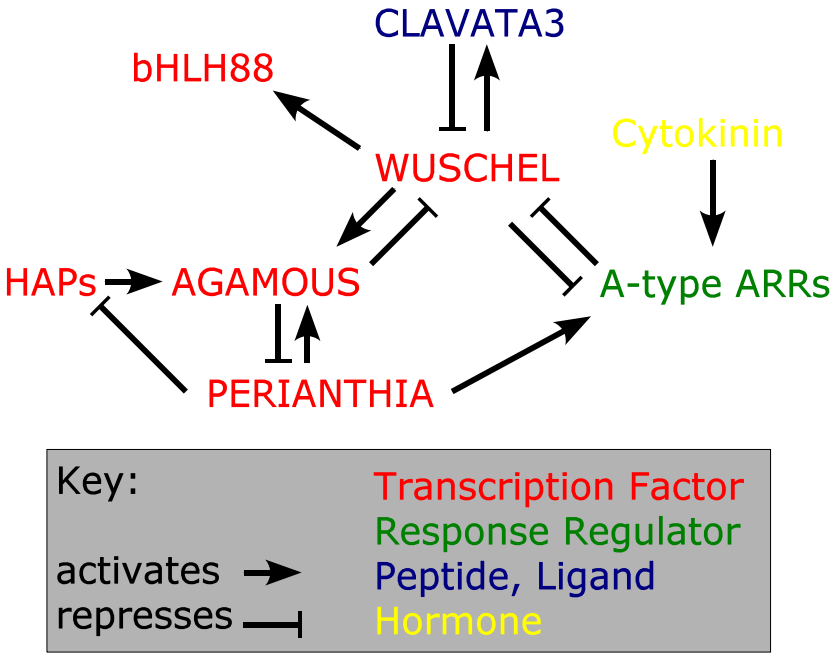
On the WUSCHEL data set, my Gibbs sampling and SVM approach uncovered a motif which has recently been published as a target for a group of transcription factors [18], which were also up-regulated in the WUSCHEL over-expression lines. We believe them to be down-stream targets in the regulatory network that centers around WUSCHEL. The motif the approach identified is TGCATGCA, which is palindromic in the molecular biology sense of the word. This is a good sign for a TFBS because a lot of transcription factor proteins are homodimers and bind to palindromic DNA sequences [19]. It is exactly the motif that was discovered by Suzuki *et al.* [20] in *Zea mays* ssp. *mays* or maize. The researches then found the motif CATGCA, two nucleotides shorter than the original, in *Arabidopsis thaliana* [18]. It is bound by a family of transcription factors called VP1/ABSCISIC ACID INSENSITIVE 3-LIKE B3, and aligning their protein sequence with the maize genome [21] gives exactly those genes with the highest alignment score that bind the longer motif in maize. We believe that the genes they give as targets for the VP1/ABSCISIC ACID INSENSITIVE 3-LIKE B3 genes with the shorter motif are probably also secondary and tertiary responses to the up-regulation, and only a subset, those which actually have the longer 8 nucleotide motif, are the true targets. This would have to be proven experimentally.

## 1.3 ChIP-on-chip Binding Data for WUSCHEL

Another set of experiments that were complemented by the microarrays allowed for a closer look at the WUSCHEL protein, which plays a key role in maintaining the stem cell pool of the shoot apical meristem of *Arabidopsis thaliana* [22]. How the stem cells are maintained in the shoot apical meristem and where

they are located has shown in [23]. A partial regulatory network depicting WUSCHEL's activity is shown in Figure 1.2.

Figure 1.2: A regulatory network showing the influence of WUSCHEL. The figure shows the effects WUSCHEL has on other transcription factors and peptides and *vice versa*. A line ending in an arrow indicates activation of expression of the gene pointed at, while a bar indicates repression of transcription. Red gene names indicate transcription factors, while blue indicates a small peptide, CLAVATA3, which is believed to be a precursor to an excreted hormone, as it is heavily modified after translation [5]. The Arabidopsis Response Regulators are shown in green, they inhibit and are in turn inhibited by WUSCHEL. Cytokinin, yellow, is a plant hormone.




To find out other genes regulated by this key player, an antibody against the WUSCHEL protein is needed. With this, an assay called chromatin immunoprecipitation on a microarray chip (ChIP-on-chip) becomes available. The cells expressing WUSCHEL are exposed to formaldehyde *in vivo*, which cross-links all the proteins bound to the nuclear DNA at the time to the area of the genome they are regulating. The cells and nuclei are subsequently lysed and the DNA is sheared. The fragments that are bound to WUSCHEL can be precipitated through the use of the antibody, which in turn is bound to the WUSCHEL protein. The cross-linking is reversed and the DNA fragments are amplified before putting them on a tiling array chip, where they give a signal in all the areas the protein was bound to. For protocol details, see 2.1.4 in the biological background section.

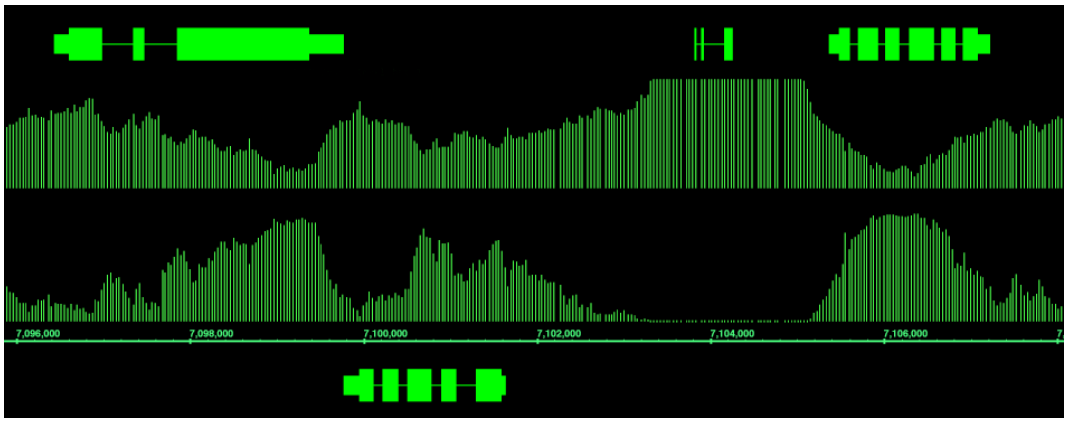
### 1.3.1 Support Vector Regression Training

The resulting binding data was evaluated using a support vector regression on the individual probe values that report bound DNA. An example of the binding

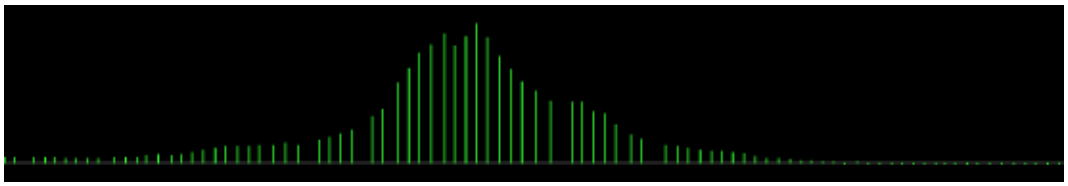
data, shown here in the analysis software provided by Affymetrix for their tiling arrays, can be seen in Figure 1.3.

Figure 1.3: Binding of DNA previously bound by WUSCHEL.

**(a)** This image is taken from the software Affymetrix [24] provides to visualize the results of the tiling array experiments. The vertical green bars represent the amount of DNA bound to the probes. Each bar represents one probe on the array and thus on the genome. The running scale shows the chromosomal coordinates. The vertical green bars are gene loci as annotated by TAIR [5], with exons as blocks and introns connecting them with a thin line.
**(b)** This image shows an ideal distribution, taken from real data, of bound DNA in a gene's promoter region. This is most probably the shape we want the support vector regression to learn.




(a)



(b)

The regression label was a score we called "Wuschel Regulation Score", or WRS, a number on an arbitrary scale that was calculated from the behavior of all the genes in several microarray expression experiments, all of them WUSCHEL gain of function experiments (Section 3.1). The idea behind this is that most genes that are bound by WUSCHEL and thus regulated by it will have a high WRS, which is high for those genes that have a high $n$-fold change in the expression experiments.

The reason for using machine learning for this problem may not be obvious. The binding data is full of noise and false positives that would throw a simple threshold clustering algorithm off course. The label we used for the regression is also not exactly what we want, because it is itself derived from selected gain of function expression experiments and thus incomplete. Since we want to find all genes that are bound by WUSCHEL, we used the WRS only as a guide to estimate SVR performance and during the analysis paid particular attention to

genes that are ranked high by the SVR but do not have a high WRS. How we are able to achieve this is outlined in "SVR Results," below.

### Window of Binding Data

The support vector regression couples a high WRS to a certain pattern of binding data from the ChIP-on-chip experiments. We use vectors that include all probes from –3000 bases before the transcription start and +800 into the gene, a tolerant window around the gene's promoter region. The probes are distributed along the genome of *Arabidopsis thaliana* in such a way that there is one approximately every 35 base pairs, with 25 bases in length. We therefore expect a maximum of 108 probes in such a window, this is the input vector for the SVR. Because the input vectors must be of the same length and some of the probes fail to give a signal or have been left out, we would be throwing away a lot of data if we require 100% probe coverage. If we require only 90% of this maximum, *i.e.* 97 probes, a lot more genes can be included. This also means that only 90% of the probes are considered if all of them are present, up to 11 probes were randomly removed from the genes.

As the sheared DNA from the nuclei is usually longer than 25 bases or even longer than 100 bases, the expected signal, should WUSCHEL be bound to a gene's promoter region, looks more like a distribution that trails off at the edges than a sharp peak. Such a distribution can be seen in Figure 1.3b.

### SVR Results

When trained on a number of hand-selected promoters that look like this distribution, complemented by a set that differed largely, the SVR was able to distinguish them with a $q$ error of 0.70 (*cf.* Equation 2.8 in Section 2.2). The error is scaled so that a result of less than 1.0 means the performance was better than a constant prediction. On a genome-wide scale, the performance was not as good as 0.70, but constantly lower than the 1.0 of a constant prediction. This is of very limited importance, because the goal was not to learn to exactly reproduce the WRS, we already have that data for every gene. We were more interested in genes that have probes bound in their promoter region similar to the ones that are definitely regulated by WUSCHEL, but may not have a very high WRS, for example because their expression is too low for a change to be detected by microarray expression analysis. Such a gene would contribute heavily to the $q$ error, but it is a result that we want to see and not get rid of.

A very good fit to the labels from a machine learning perspective is therefore not exactly what we wanted. A much better result in terms of finding more genes with a low WRS but similar binding properties was achieved by selecting the 180 genes with the highest WRS and another 180 from the rest of the dataset at random, as something like a negative sample. The rationale behind this selection is that we are sure that direct WUSCHEL targets are definitely overrepresented in the top 180 ranked by WRS, and we want the SVR to learn how their binding profiles look and have it assign a high regulation score. This will ideally yield an SVR that is able to find genes bound by WUSCHEL, regardless

of their original WRS. On this selection, I performed a grid search and 5-fold cross validation to find out the optimal parameters (2.2.2). This resulted in an assignment of binding-based scores from the SVR, similar to the WRS and containing significantly more genes with a high WRS than would be expected by random selection, but is also not unnecessarily re-creating the WRS, which we already have.

This once again led to lists of genes, for example the top 80 from the SVR. Handed over to the motif evaluation pipeline, I was able to identify several motifs that look like possible WUSCHEL binding sites, in genes that are regulated by WUSCHEL. They will be evaluated in biological experiments and will contribute to the expansion of the regulatory network from Figure 1.2.

## 1.4 Structure of this Work

This thesis starts with an abstract and a number of indices, a table of contents, figures and tables. Abbreviations frequently used are listed on a separate page for reference, but they are also written out the first time they are introduced in the text.

This chapter, the Introduction, gives an overview of the entire thesis. It describes the problem, shows how the concept of machine learning was introduced to solve some parts of it, and briefly describes the data involved. The individual sections end with a summary of the most important results obtained.

Chapter 2 describes the biological and computer science background needed to understand this thesis. The first part gives insight into Arabidopsis thaliana as a model organism and basic concepts of transcriptional regulation and some of the experimental techniques through which we obtained the data. The second section introduces machine learning and its applications especially in this thesis, and gives a theoretical background for the two techniques I used, support vector regression and support vector machines. The last part deals with the Gibbs algorithm for statistical motif search.

In the following chapter, the data sources and analysis methods for the microarray data, the ChIP-on-chip data and the alignments are described, as well as design and implementation of the Shogun machine learning toolbox and my Python work flows.

Chapter 4 describes the results of the two main Python work flows. The discussion in Chapter 5 starts with an overview of related work and then describes how it is possible to infer a network from the results obtained in Chapter 4. It ends with an outlook which gives some information about the future of the programs I wrote and for the next steps required to confirm the results.

The work ends with the mandatory affidavit and an appendix, listing the software programs used. Throughout the text, works from other scientists are cited using numbers in brackets. The citation for the corresponding work can be found on the last pages in the bibliography, in order of appearance in the text.

# Chapter 2

# Background

This chapter is divided into three parts, introducing the underlying biology and the two areas of computer science this work is involved with. Section 2.1 is mostly interesting for scientists who are from outside the biological field but interested in the source of the biological data used in this thesis. The machine learning chapter starts with the basic concepts, so a general audience can understand it. Section 2.3 is about a lesser known area of computer science and may thus be interesting to all readers.

## 2.1 Biological Context

*Arabidopsis thaliana,* commonly known as thale cress or mouse-ear cress, is a flowering plant from the *Brassicaceae* family and an eudicot. It is the most widely-used plant model organism in biology [25], due to some useful characteristics like a rapid life cycle of about six weeks, efficient transformation methods using *Agrobacterium tumefaciens*, a relatively small genome of about 125 Mb, and easy cultivation in restricted space [26]. It was one of the first higher eukaryotic organisms, in fact the first plant, that was completely sequenced in a joint effort of several labs around the world in the Arabidopsis Genome Initiative [27]. Annotation efforts continue to date at the Arabidopsis Information Resource [5].

This thesis is based on experimental data obtained from this model organism and is therefore easily comparable to other works on the same system. The data sets were obtained from experiments conducted in Prof. Dr. Detlef Weigel's department, Department VI – Molecular Biology, of the Max Planck Institute for Developmental Biology in Tuebingen, Germany by Wolfgang Busch in Dr. Jan Lohmann's research group, and from public resources.

### 2.1.1 Transcription and Regulation

In a multicellular organism, regulation of the genes to be transcribed is essential for cell differentiation. To this end, transcription factors bind short sequences of genomic DNA in the cell's nucleus, close to the genes whose transcription they regulate. Most transcription factors facilitate or inhibit the binding

of RNA polymerase and the transcription pre-initiation complex. Usually this happens within the regulatory, untranslated region located between genes or in introns. For an overview of basic transcriptional regulation, see for example the book "Molecular Biology of the Cell" by Alberts *et al.* [19].

As transcription factors are also gene products, they can in turn be regulated by other transcription factors that activate, enhance, suppress or inhibit transcription of the corresponding gene by directly or indirectly affecting the binding of RNA polymerase, or modifying its activity. The identification of direct and indirect target genes of transcription factors is the key to inferring regulatory networks.

### 2.1.2 Transcription Factor Binding Sites

Transcription factors recognize a certain pattern of DNA known as a transcription factor binding site (TFBS). Most transcription factors bind to a pattern that can only be described using a position frequency matrix (PFM): the pattern is not completely invariant but allows for some divergence in at least some positions; these patterns are called *motifs*.

Often, something similar to a consensus sequence is given instead or in addition to a PFM. Consensus sequences are used in the results of multiple sequence alignments, where they show which residues are variable, and more importantly, which residues are conserved across the sequences aligned. A PFM can be converted to a consensus sequence by writing the residue with the highest information content at each position in upper case, sometimes substituting for lower-case letters when information content is very low, or for symbols that stand for several DNA bases such as Y for any of the two pyrimidine bases. Another possible representation of the PFM that is intuitively understood is a sequence logo, which displays all four DNA bases in different colors and the information content is represented as the height of the letter, allowing for several letters in the same position with shared information content [28].

A TFBS motif has a length of about six to twelve bases, rarely more; sometimes a transcription factor needs two or more adjacent motifs to bind, it can therefore be interesting to look at reoccuring pairs of sites by mining item sets. For this thesis, I employed a program written by Dr. Hiroto Saigo in Dr. Koji Tsuda's group at the Max Planck Institute for Biological Cybernetics in Tuebingen, Germany [29], a method that has been used on other biological data with good results [30]. The way the transcription factor protein actually recognizes the stretch of DNA is very different depending on the structure of the protein. Some transcription factors reach into the major groove with an $\alpha$-helix, for example [31]. The biophysical binding process may not be dependent on all bases in the motif, which explains the variation some transcription factors accept at their binding site when recognizing a motif: often, some positions can be completely arbitrary [32].

Since the untranscribed region in particular is subject to greater mutation rates than the coding parts of the genome, similar motifs in different species indicate some need for conservation during replication [33]. In our case, this can be used to identify motifs that have not changed since the model organism *Arabidopsis thaliana* split from *Oryza sativa*, or rice, another plant species whose

genome is readily available since 2005 [34], which was used in this thesis to determine putative TFBS which are conserved among those two species. The first approach, using only the sequence of the close relative *Arabidopsis lyrata*, was not very successful, because the first 700 bases of every gene's upstream region are virtually identical among the two species [35], given that the evolutionary distance to their last common ancestor is only an estimated five million years [36].

Since a transcription factor can only bind to a specific motif, it follows that two genes are regulated by the same transcription factor if they have the same motif in their regulatory region. If we in turn find a similar pattern in two genes' regulatory regions, they will thus be tentatively co-regulated. An over-represented, conserved pattern therefore has a high possibility of being a TFBS. The spatial pattern distribution, over-representation and conservation of regulatory motifs have been examined *in silico* and the methods are explained in Section 2.2 and Section 2.3.

### 2.1.3 Microarrays

The expression data used in this work was obtained from a series of microarray experiments. The goal of expression analysis by microarrays is to find all genes that are expressed in a certain tissue at a given time point. For the experiments examined in this thesis, mRNA levels in tissues of mutated *Arabidopsis thaliana*, for instance over-expression or knock-out mutations of a certain gene (*cf.* Chapter 3, Section 3.1), were looked at, as well as tissues that had been exposed to extreme conditions like heat and cold.

Extraction of the mRNA from these tissues is the first step in this protocol. Further transcription into cDNA is necessary for the molecules to bind to the DNA on the chip. The bound probes then emit a detectable fluorescence signal and the intensity is correlated to the amount of mRNA expression from the tissue extracted. Because these experiments are very prone to noise, they are usually done in several biological and technical replicates.

The commercially available Affymetrix GeneChip Arabidopsis ATH1 genome array (Figure 2.1) is a quartz chip that contains probe sets representing more than 24,000 gene sequences. One probe set consists of 22 oligo-nucleotide probes of 25 bases each. Of these, 11 match perfectly to the target sequence, while the remaining 11 probes harbor a single mismatch in the center [37, 38]. Subtracting the sites used for normalization and some putative genes, this covers about 80% of the known genes [39]. The size of one feature on this chip is 18 $\mu$m, which is made possible by using photolithographic *in situ* synthesis to create the oligonucleotides on the quartz chip, a technique adapted from computer chip manufacturing.

Data from the mismatch probe can be used to estimate unspecific binding to the perfect match probe, *i.e.* its fluorescence signal intensity is subtracted from the perfect match probe's signal by certain normalization algorithms. Several biological replicates were created for each experiment (mostly triplicates); normalization was in most cases conducted according to the GCRMA method, which is a model based algorithm using signals only from the perfect match probes. For details, please refer to Section 3.1.

Figure 2.1: Affymetrix GeneChip




We used data from a heat shock experiment to confirm that this method works as expected, because there is a know heat shock regulation motif [17]. During this experiment, plant tissue was exposed to 38°C warm medium. Expression levels are measured one hour after the exposure to heat. Cold stress data was obtained similarly, after an exposure to 4°C. The other data sets from gain of function and loss of function experiments were obtained using wild type control plants grown under similar conditions.

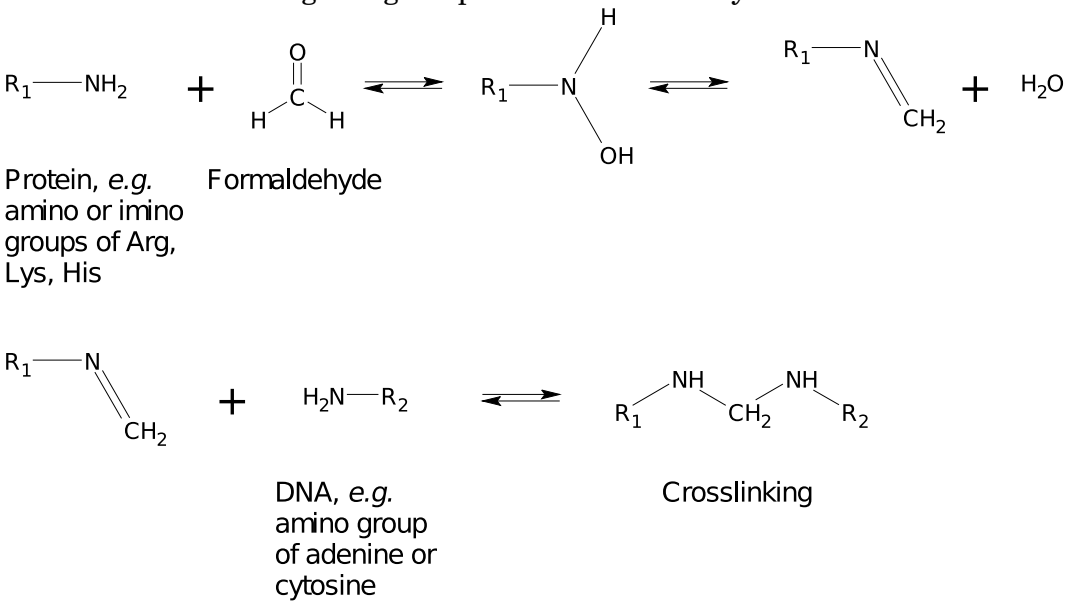
The expression data is used to find genes of similar behavior, either just from one experiment and its control, or from several experiments that have an underlying logic. As the results show, the latter is a cleaner and more rewarding approach. Of course, it can only be used if the experimental logic is known beforehand, and if the setup is more complex than a single exposure to abiotic stress. For this, lists are compiled that contain all the genes that reacted in an experiment with a strong change in expression. Naïvely compiled lists with a strict threshold of a 4-fold expression change, for instance, in a single heat shock experiment did not perform very well, while lists that were based on the behavior of genes in two or three experiments worked very well due to the added stringency and reduced chance of random errors, noise or experimental artifacts (see chapters Chapter 4 and Chapter 5). The lists where used for motif finding (Section 2.3) and the resulting motifs were subjected to the SVM work flow.

### 2.1.4 Chromatin Immuniprecipiation on a Chip

Protein-DNA binding data was obtained from an experimental method called chromatin immunoprecipitation on a chip (ChIP-on-chip), *i.e.* a microarray chip. This method enables studying of protein-DNA binding *in vivo*, on a genome-wide

scale, for a specific protein of interest. The experiments performed by Wolfgang Busch were geared towards finding the genes whose regulatory regions are bound by one transcription factor in particular, WUSCHEL (AGI accession number AT2G17950). An antibody raised against WUSCHEL is necessary to filter out the parts of the DNA that are bound by this transcription factor.

Figure 2.2: Cross-linking of DNA and proteins using formaldehyde.
These reactions take place when cross-linking the proteins on the DNA to the chromatin by means of formaldehyde. The double bond of the formaldehyde's oxygen atom attacks the amino and imino groups of the arginine, histidine, and lysine residues of the protein to form methylol adducts, which are then partially dehydrated, forming another reactive group with a double bond, a Schiff base [40]. The labile imine in turn enables the protein to form a covalent bond with an amino group of one of the DNA bases adenosine or tyrosine, if they are approximately 2 Å away [41]. This covalent bond is stable enough for the purifications that follow, but the cross-linking can later be reversed by heating the molecules while digesting the proteins with an enzyme called Proteinase K.




According to the protocol for this method [24], cells are immersed in a formaldehyde solution, which reacts with the proteins bound to the chromatin in the cell nucleus, cross-linking them to the DNA (Figure 2.2). The cells are lysed, their nuclei isolated and the DNA is sheared into ~1 kb long fragments through sonication. The antibody is added and binds to the WUSCHEL protein. Beads coated with Protein A, which binds to the non-variable domain of the heavy chain of the antibody, are used to extract the antibody-bound pieces of chromatin. This mixture is washed several times and the DNA is finally purified by uncoupling the cross-linked proteins through gentle heating and digesting them with Proteinase K, after which the proteins are cleared away using affinity chromatography or a phenol-chloroform extraction.

The purified DNA is amplified, supplying not only the regular deoxynucleotide triphosphates (dNTP) adenine, cytosine, guanine, and thymine but also a number of uracil triphosphates, replacing some of the thymine triphosphates. This results in amplification products that have a certain amount of U's instead of

some of the T's in their sequence. The low number of uracil bases in the amplified strands is used to cleave the DNA just behind every U, using uracil DNA glycosylase and apurinic/apyrimidinic endonuclease 1 (APE1), yielding fragments that are about 30 to 100 bases long on average. This number can be adjusted by changing the dNTP composition, for instance by including more uracil triphosphate to make them shorter.

In the next step, the molecules are end labeled with biotin and hybridized to the tiling array, where they are stained using fluorescence-tagged streptavidin. The signal is further amplified and then scanned. For information on data analysis, see Section 3.2 in Chapter 3, Materials and Methods.

## 2.2 Machine Learning

Machine learning is a part of artificial intelligence, itself a subfield of theoretical computer science. There are numerous applications for machine learning, and biological data analysis is just one of them. The methods allow computers to automatically extract information from large data sets.

The goal of a general machine learning approach is to make computers "learn" patterns from relatively small data sets and classify comparably large amounts of new data into one of several groups, based on the data seen so far.

Generally, there are supervised and unsupervised learning algorithms. Supervised learning methods need a labeled training data set on which they can learn the classes they are later to distinguish. The label has to be provided on every unit of input data and states exactly which class it belongs to. Later, their performance can be evaluated on data that has a known classification but the label has not been shown to the learning method during training. If input data is sparse or to find good parameters, a method called $n$-fold cross validation can be employed, where alternating fractions of the labeled data is used to train the method and another part is used for testing.

Unsupervised learning methods do not rely on labeled data, they classify data without knowing anything about it. They can of course be tested with known labels as to their performance, but would not use this additional information. This is the major advantage of unsupervised methods: they do not need data sets that are already labeled. Supervised learning methods, in contrast, rely on this additional information and use it to tune their classification function; the programs used in this thesis are all supervised learning methods and use labeled data (with varying certainty) as training and testing data.

Training of such methods is known to users of speech recognition, where the software has to learn to recognize the voice of a new user by having texts read to it. The ultimate goal of both methods is to find a classifier that performs well on completely new, unlabeled data, so it is a helpful tool for classification. In the speech recognition example, this means that the software is ultimately able to understand any text read to it by the user (limited by its vocabulary), not only the training samples. Trained methods that perform well in testing are subsequently used on such data that has not yet been classified. We want them to do as well as possible on testing and training data sets, because there we can

compute a prediction error with ease, but with new, unclassified data we have to verify findings experimentally. If we use methods that already performed well in testing, *i.e.* on data not seen during training, we assume that the predictor will also do well on new data.

### 2.2.1 Support Vector Machines

The method used in this thesis is called support vector machine (SVM), a classifier that measures the distance of the input feature vectors through linear combination and puts the input into two or more classes [4]. For a complete overview, see the book "Learning with Kernels" by B. Schölkopf and A. Smola [42].

For example, the input feature vector $\vec{x}$ is multiplied with a weight vector $\vec{w}$, which has been adjusted during training of the SVM, yielding a scalar product. Offsets from the origin are added by a constant $b$. The general linear classification function can be written as

$$y = \mathrm{sgn}(\langle \vec{x}, \vec{w} \rangle + b), \tag{2.1}$$

classifying the input $\vec{x}$ into one of two classes, –1 and 1, as determined by the *signum* function. During training, the labeled data set $(\vec{x}_i, y_i)$ with $i = 1, ..., n$ is used as input, and the algorithm tries to learn the mapping from $\vec{x}$ to $y$ by balancing a perfect fit of the training data and its ability to generalize to unseen data.

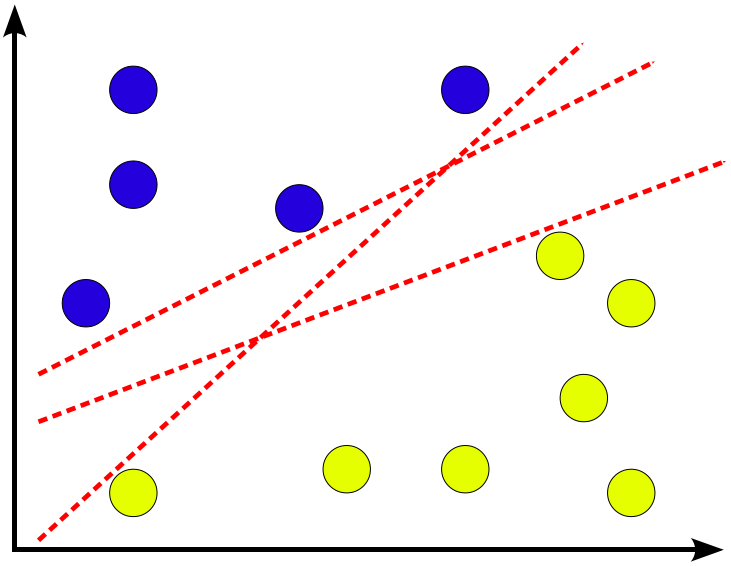
A classifier that is untrained or underfitted to the training data will usually not perform much better than a method that randomly assigns a class to every input vector. On the other hand, if the classifier performs perfectly on the training data, it is most likely overfitted and may perform worse than randomly guessing the class for every input vector. A good balance has to be maintained between these two extremes by tuning the so-called regularization parameters. These parameters can be adjusted by holding back enough data for validation. Critically assessing the performance of a trained method is key to creating a good classifier.

The underlying rationale will become clearer in an example. For a two-dimensional input space, the data can be visualized as points on a coordinate system. The classification function divides the data points by a hyperplane, where all points on one side of the plane belong to the same class (see Figure 2.3 for an example).

#### Performance

The classification is scalable to very high dimensions in the input space, and still remains very fast, because even though training has a theoretical complexity of $\mathcal{O}(n^3)$, the prediction performs with $\mathcal{O}(n)$, where $n$ is the number of examples. The theoretical complexity is the worst-case scenario, in practice this depends on the input data, and an exponent of 3 is rarely reached. SVMs are the ideal choice for the problems presented in this thesis, as there are a large

Figure 2.3: Deciding on hyperplanes in a support vector machine.
The two-dimensional data points of two different classes are displayed in a coordinate system. Any of the hyperplanes shown in red separate the two classes, but which one allows for the best generalization later on?



number of data points that can be used for classification, and the problem is posed in such a way that two classes suffice. Alternatives in machine learning would be neural networks, an older technique that is usually outperformed by SVMs. Additionally, we have a lot of experience to use SVMs on sequence data directly [43, 44], whereas this experience is lacking for neural networks.

### Training

The input data is normalized so that a single large entry in the input vectors cannot dominate the resulting scalar product. Obviously, the individual positions may nevertheless be of different importance for classification.

During training, a support vector machine tries to find a maximal separating hyperplane between the two classes of input vectors by adjusting the weight vector in the classification function. This plane is found by constructing two parallel planes to the separating hyperplane that still separate the data points. If there are absolutely no points in between, this is called a hard margin. The hyperplane is adjusted to maximize the distance between the two parallel planes that delimit the margin, which is defined by the closest data points of the two classes, called support vectors (*cf.* Figure 2.4). Other data points are subsequently ignored. A soft margin classifier accepts that some input vectors may be inside the margin around the hyperplane, and also that some vectors are falsely classified, to obtain a larger margin. The balance between a large margin and too many falsely classified examples has to be adjusted using parameters.

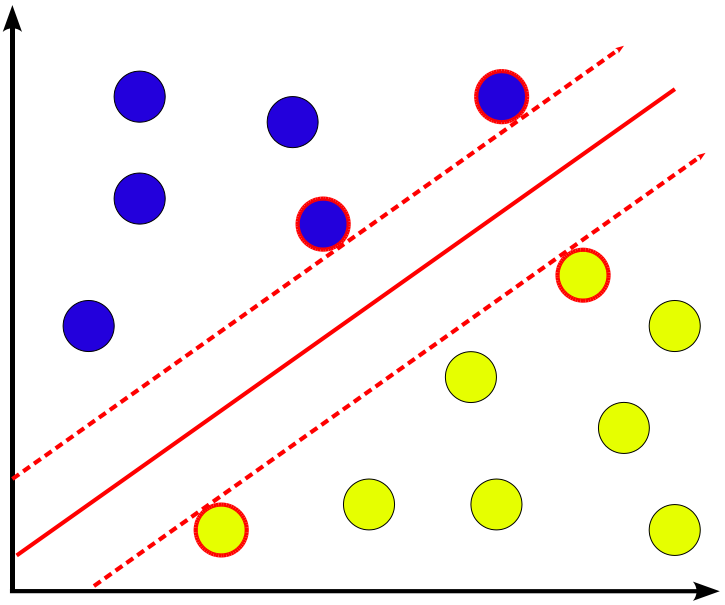
The assumption is that the greater the distance between the points of the two classes that already lie closest to each other, *i.e.* the largest possible margin, the better the SVM can generalize for examples drawn from the same distribution.

This is because points that lie in the middle, between the bulk of the two classes, are hardest to classify. We assume of course, that the data used for training and testing is a good representation of what is to come – so the classifier will not be derailed by new data that is completely dissimilar to what it learned to classify.

The adjustment of the hyperplane during training is calculated according to the objective function $E$ (Equation 2.2). The labeled data set is $(\vec{x}_i, y_i)$, where $\vec{x}_i$ is the input vector and $y_i$ its corresponding label. The regularization term $\|\vec{w}\|^2$ is used to determine the softness of the margin, i.e. how many data points the classifier accepts on the "wrong" side of the hyperplane. The other term of the function is the margin violation of all data sets $(\vec{x}_i, y_i)$ over all $i = 1, ..., n$. $C$ is a parameter of the SVM that determines the trade-off between the two terms, margin violation and regularization, in the result of the objective function $E$. The objective function is being minimized to obtain the hyperplane with the largest margin.

$$E = \|\vec{w}\|^2 + C \sum_i \max(0,\ 1 - (\langle \vec{w}, \vec{x}_i \rangle + b)\, y_i) \tag{2.2}$$

Figure 2.4: Example of a hyperplane in a support vector machine. The hyperplane (solid) is adjusted so the distance between the closest "representatives" of each class is maximal. These representatives are called support vectors, they are circled in the figure. The two dashed lines are the boundaries of the margin; they are parallel to the hyperplane. The way this adjustment is calculated is shown in Equation 2.2. The offset of the hyperplane from the origin is called $b$.



As stated before, SVMs are a supervised learning method. Labeled input data will be used as training data. This information is used to adjust the weight vector and the classification function, while maintaining the balance between perfectly classifying the training data, *i.e.* overfitting, and the ability to generalize on unseen examples.

### Evaluation

To evaluate the performance of a classifier, some of the available labeled data is knowingly held back and used for testing later on. A popular technique, also employed in this thesis, which formalizes the "holding back," is $n$-fold cross validation, in which $\frac{1}{n}$ input vectors are held back for testing, and the remaining $\frac{n-1}{n}$ are used for initial training. This is repeated $n$ times for every non-overlapping set of the input vectors, called split, of size $\frac{1}{n}$. The prediction error is then averaged after these $n$ repeats.

The testing error is evaluated for instance by using the area under either the precision recall curve, especially suited for unbalanced problems with largely different numbers of positives and negatives, or the receiver operating characteristic curve [45]. The two measurements are related, as the underlying metrics are the same. For all of these curves, it is important to look at the rates of correct and incorrect predictions, split into four groups, as seen in Table 2.1.

Table 2.1: Confusion matrix of correct and incorrect predictions [45].

| | **actual positive** | **actual negative** |
|---|---|---|
| **predicted positive** | true positive (TP) | false positive (FP) |
| **predicted negative** | false negative (FN) | true negative (TN) |

This matrix allows for some derivations that appear under various names, depending on the field of research. Some of these metrics are shown here:

$$\text{Precision} = \text{Positive Predictive Value} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.3}$$

$$\text{Recall} = \text{True Positive Rate} = \text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.4}$$

$$\text{False Positive Rate} = (1 - \text{Specificity}) = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{2.5}$$

The precision recall curve is a plot of the recall (Equation 2.4) and precision (Equation 2.3) scores as points in an $(x, y)$ coordinate system. The receiver operating characteristic has been used for signal detection and analysis, as well as in evaluating diagnostic tests and drugs [46]. Today it is widely used in the field of machine learning. It can be plotted as sensitivity *versus* false positive rate (Equation 2.4, Equation 2.5), both values normalized to the [0, 1] interval, and the area under this curve should be significantly greater than 0.5 for the performance to be better than random guessing, *i.e.* a diagonal line, which amounts to randomly assigning one of the two classes to every data point.

### Kernel Functions

The linear scalar product can also be replaced by non-linear variants, called kernel functions, such as the radial basis function. For this, it is necessary to

project the input vectors $x_i$ into a higher-dimensional space through a projection $\varphi(\vec{x}_i)$. The transformation allows for more complex, non-linear classification functions that are able to classify very diverse input features, making it a powerful machine learning method. The transformed input space is usually very high dimensional, in the case of the Gaussian radial basis function even infinite-dimensional, where the scalar product of two vectors would be impossible to compute explicitly. The kernel function is chosen in such a way that its result for $K(\vec{x}_1, \vec{x}_2)$ is equal to computing the scalar product $\varphi(\vec{x}_1) \cdot \varphi(\vec{x}_2)$ in the transformed, potentially infinite feature space.

The non-linear kernel function is therefore an elegant way around this, such a good way in fact that it has been dubbed the "kernel trick." Even though the hyperplane is linear in the new space, transformed back into the input space it is a non-linear function [47].

The mathematical theory for this transformation is based on Mercer's theorem, its application to this problem, according to [48], was first proposed in [49], where they called the convolution of the scalar product Potential Functions. The scalar product in this new, high dimensional space may be difficult or impossible to compute, but thanks to the kernel function, which replaces the linear scalar product in every calculation in the algorithm, it will never be necessary to do so explicitly. According to Mercer's theorem, this mapping of input features into a higher dimensional space is valid as long as the kernel function has some specific properties which are stated in this theorem. Because the scalar product of the two transformed input vectors to be compared evaluates exactly to the kernel function, it can be used instead of the scalar product in the SVM algorithm: $K(\vec{x}_1, \vec{x}_2) = \varphi(\vec{x}_1) \cdot \varphi(\vec{x}_2)$. For a proof, see for example Section 3.2 of Smola's thesis [50].

### Using the Gaussian Radial Basis Function as a Kernel

The Gaussian radial basis kernel, for example, is an exponential function that uses the Euclidean distance between the input vectors and divides it by a parameter $\sigma$, also called the kernel width. The input features are mapped into a higher dimensional space, where the computation of the linear scalar product may be impossible to compute explicitly for reasons stated above. The kernel function is identical to the explicit computation of the scalar product in the projected space. Since this is based on a provable mathematical rationale, *i.e.* Mercer's theorem, the input vectors and the algorithm can remain unchanged, only the linear scalar product is replaced by a kernel. For many functions, the requirements for Mercer's theorem can be shown and they can thus be used as kernels, like the Gaussian radial basis function, for which a common notation is

$$K(\vec{x}_1, \vec{x}_2) = \exp\left(-\frac{\|\vec{x}_1 - \vec{x}_2\|^2}{2\sigma^2}\right) \tag{2.6}$$

The string kernel and weighted degree kernels, taken from [51] and also used in this thesis, are described in Materials and Methods, Section 3.4.

### 2.2.2 Support Vector Regression

A powerful extension of the SVM machine learning approach is called support vector regression (SVR), which was used for classifying the ChIP-on-chip data in this thesis [50]. While the classifier of an SVM classifies input into only two very distinct classes because its output is thresholded, the goal of a support vector regression function is to predict and real number for every input vector. Training data should be labeled accordingly.

A regression function assigns a real number to each input vector and needs to be trained and tested with labeled training data, almost like an SVM, as it is also a supervised learning method. The algorithm takes as parameters the threshold $\varepsilon$, the kernel width $\sigma$, and the smoothness of the regression function $C$. In the input space, a band or buffer zone of width $\varepsilon$ is constructed around the regression function. The errors of all input vectors that come to lie inside this zone are subsequently ignored in prediction error calculations and for the adjustment of the regression function during training, *i.e.* they are not support vectors. The threshold $\varepsilon$ is a parameter of the SVR and in this application set to $\varepsilon = 0.01$, a default setting that was not optimized for the purpose of this work. The input vectors that do play a role for prediction error calculation and parameter adjustment are the ones outside this zone, whose distance to the regression function is greater than $\varepsilon$, they are the support vectors for the SVR (Figure 2.5).

#### Parameter Selection and Performance Evaluation

The regularization parameter $C$ describes the trade-off between a function that tries to hit each training data point, creating a very complex curve, and the opposite. It is generally a good idea to try different settings for $C$ and the kernel width $\sigma$ and to scale them according to the input data. Both of these parameters have been evaluated in this thesis using cross validation and grid search, a very straightforward parameter optimization technique yielding a matrix that can be represented nicely as a heat map of parameters (Figure 3.1).

The performance of a support vector regression can be evaluated using for example the $q^2$ error:

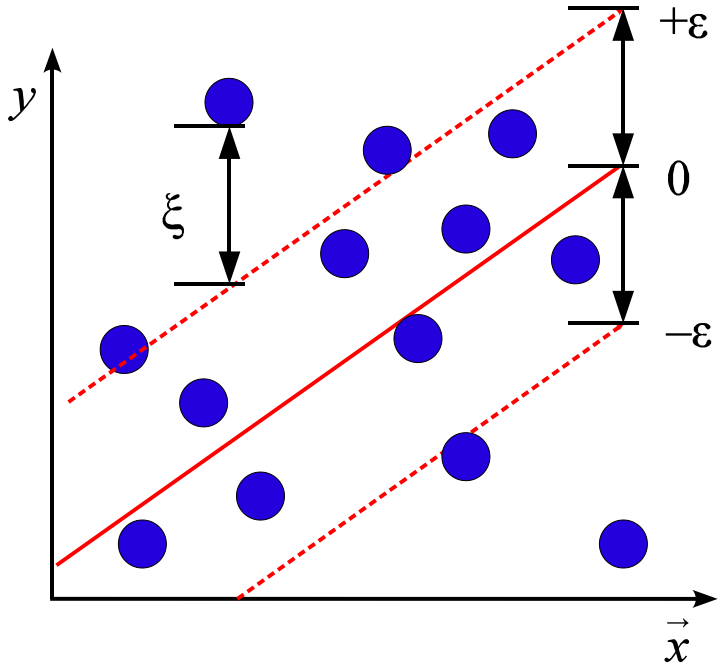
$$q^2 = \frac{\sum_i (f(\vec{x}_i) - y_i)^2}{\sum_i (\mathrm{mean}(y) - y_i)^2} \tag{2.7}$$

or the similar $q$ error:

$$q = \frac{\sum_i |f(\vec{x}_i) - y_i|}{\sum_i |\mathrm{median}(y) - y_i|} \tag{2.8}$$

In these equations, the predicted value for $f(\vec{x}_i)$ is subtracted by the actual label $y_i$ assigned to $x_i$, and for normalization is divided by the error of the mean predictor (in the case of the $q^2$ error) or the median predictor (in the case of the $q$ error) of all $y$ minus the current $y_i$. This error is smaller than 1 if the prediction is better than that of the best constant predictor, and larger than 1 otherwise.

Figure 2.5: Example of a support vector regression.
The solid line is the hyperplane, a function fitted to the labels of the training data. The two parallel lines are indicators of the zone where the distance to the hyperplane is smaller than $\varepsilon$. Only the examples outside this zone are support vectors. The distance between the hyperplane and the vectors outside the zone is subtracted by $\varepsilon$, so that only the difference of the labeled value $y$ for this vector and the predicted value $f(\vec{x}_i)$ indicated by $\xi$ counts towards the prediction error. Vapnik called this the $\varepsilon$-insensitive loss function [42].




## 2.3 Statistical Motif Search

One problem that had to be solved in the course of this research was to extract motifs, *i.e.* interesting $k$-mers, from sequences. This is often the only approach for some methods to find transcription factor binding sites, but as we also have other data like conservation available for this thesis, obviously not the only one to use. The gene lists obtained from the microarray expression experiments (Section 3.1) were the basis for the motif search. See Table 3.3 for detailed descriptions of the lists.

### 2.3.1 Data Preparation

We extracted the –1000 base pair upstream region of each gene according to the current TAIR annotation [5]. This is a feasible length, as experiments that compared different lengths and areas showed [52], and is also indicated by the fact that the –700 upstream region of matching genes is highly conserved in *Arabidopsis lyrata* [35], which is *Arabidopsis thaliana*'s closest relative. This resulted in groups of several sequences that were used to find over-represented motifs.

### 2.3.2 Gibbs Sampling Algorithm

The main assumption made by statistical motif search is that motifs of transcription factor binding sites will be over-represented to a statistically significant degree. The Gibbs sampling algorithm developed in 1993 [53] still is the most widely-used method for this task, and many different implementations have been published. See for example [54, 55, 56, 57, 58, 59, 7], many others exist. A comparative performance evaluation has been done by Martin Tompa in [60]. This algorithm looks for over-represented motifs, for instance in a set of DNA sequences. The output is a position frequency matrix for every motif the algorithm identified. If it is a true TFBS, the motif shows the bases that a transcription factor binds to on a DNA molecule.

The algorithm uses a Markov chain Monte Carlo procedure, and uses a Bayesian method to infer the number and location of several motifs simultaneously, *i.e.* during one iteration. To filter out any bias in nucleotide distribution, it also uses a background model, which is either computed from the input data before every run or given as an argument during execution. Some implementations use a Wilcoxon signed-rank test to score the motifs the algorithm found and assess their statistical significance [54]. The following is a summary [52] of the description of the original Gibbs sampling algorithm for nucleotide data as published by C. E. Lawrence, S. F. Altschul, *et al.* [53].

The algorithm looks for ungapped patterns in each of the input sequences and tries to find occurrences of that pattern in the other sequences. A single pattern is computed as a probabilistic model of nucleotide frequencies at each position (PFM).

The input is a number of sequences $S_1$, ..., $S_N$ and a width $W$ of the motif to look for. The algorithm keeps the PFM pattern description in memory during each iteration. Each column $i$ from 1 to $W$ consists of the variables $q_{i,A}$, $q_{i,C}$, $q_{i,G}$, $q_{i,T}$, representing the probability of each nucleotide. Another instance of the same data structure contains the background frequencies as $p_A$, $p_C$, $p_G$, $p_T$. There is a second data structure that describes an alignment. It contains positions from 1 to $N$, indicating the occurrence of the common pattern within the sequences. The algorithm tries to find the most probable common pattern by maximizing the ratio of pattern probability to background probability.

#### Iteration

Initialized with a random starting position, the algorithm proceeds through many iterations of the following two steps:

1. *Predictive Update*: One of the $N$ sequences is chosen at random, the pattern description $q_{i,j}$ is calculated as described in Equation 2.9, from the current position of $a_k$ in all sequences except the current one, and background frequencies $p_j$ taken from Table 2.2a. For the $i$th position of the pattern, there are $N-1$ observed nucleotides. Let $c_{i,j}$ be the count of nucleotide $j$ in this position. Bayesian statistical analysis suggests the $c_{i,j}$ be

supplemented with pseudo counts $b_j$ to yield pattern probabilities, where $B$ is the sum of the $b_j$.

$$q_{i,j} = \frac{c_{i,j} + b_j}{N - 1 + B} \tag{2.9}$$

The $p_j$ are calculated analogously, with the corresponding counts taken over all positions not included in the pattern. The addition of pseudo counts is necessary to avoid assigning zero probability to nucleotides not occurring in the current pattern. Pseudo counts have to be defined *a priori* and should reflect a general tendency of the organism towards certain nucleotides. In the case of *Arabidopsis thaliana*, this means AT-rich regulatory regions. A frequency matrix for the background can be found in Table 2.2a.

2. *Sampling:* A sliding window of width $W$ moves over the current sequence $z$ and the probability $Q_x$ of generating each segment $x$ according to the current pattern probabilities $q_{i,j}$ are calculated as are the probabilities $P_x$ of generating these segments by the background probability $p_j$. The weight $A_x = Q_x/P_x$ is assigned to segment $x$, and a random segment is selected to become the new position of $a_x$. After normalization, $A_x$ gives the probability that the pattern sequence $z$ belongs at position $x$. The algorithm finds the most probable alignment by selecting a set of $a_k$s that maximizes the product of these ratios, where the $c_{i,j}$ and $q_{i,j}$ are calculated from the complete alignment.

$$F = \sum_{i=1}^{W} \sum_{j=\mathtt{A}}^{\mathtt{T}} c_{i,j} \log \frac{q_{i,j}}{p_j} \tag{2.10}$$

Table 2.2: Nucleotide frequencies and a transition matrix.
**(a)** Nucleotide frequencies in the intergenic, non-coding regions of *Arabidopsis thaliana*. This matrix is used by the Gibbs sampling algorithm as a background model.

**(b)** The transition matrix to score occurrences of a position weight matrix in a sequence. Due to the nature of the Gibbs sampling algorithm, it is possible that some occurrences of a pattern in a list of sequences is missed. It is therefore a good idea to look for every occurrence of a pattern of interest, not only in the whole genome, but also in the original set of sequences, lest some occurrences will be missed.

**(a)**

| Nucleotide | Frequency Proportion |
|---|---|
| **A** | 0.3449 |
| **C** | 0.1581 |
| **G** | 0.1556 |
| **T** | 0.3414 |

**(b)**

| | Transition Matrix | | | |
|---|---|---|---|---|
| | **A** | **C** | **G** | **T** |
| **A** | 0.3911 | 0.1516 | 0.1482 | 0.3091 |
| **C** | 0.3760 | 0.1703 | 0.1268 | 0.3269 |
| **G** | 0.3630 | 0.1389 | 0.1671 | 0.3311 |
| **T** | 0.2756 | 0.1678 | 0.1711 | 0.3855 |

The main idea of this iterative algorithm is that the more accurate a pattern description becomes in step 1, the more accurate the determination of its lo-

cation in step 2 will be and *vice versa*. Once some correct positions $a_k$ have been selected by chance, the $q_{i,j}$ will converge towards a pattern existent within other sequences. This will in turn lead to further correct $a_k$ and will improve the discriminating ability of the emerging pattern.

The width of the pattern is a mandatory input parameter and cannot be given to the algorithm in a range. The reason lies within the increased dimensionality and the fact that *e.g.* in Equation 2.10, the calculation of $F$ is not useful if $W$ can be increased (it will always be better to increase it in this case). In the work flow, a number of separate calculations is run for plausible widths from 6 to 12 to obtain patterns with different lengths. In order to score the findings, the transition matrix from Table 2.2b was used. For more details about the work flow, see Chapter 3.

# Chapter 3

# Materials and Methods

## 3.1 Microarray Experiments

The microarray data that were obtained from the expression experiments Wolfgang Busch conducted are best summarized in two tables, describing the experiment itself and the extracted gene lists. For most of the lists, some form of $n$-fold change was used as a threshold. It was computed from the absolute level of the experimental probe or gene, $p$, and the one for the control, $c$, as shown in Equation 3.1. In this case the direction of the change is represented by the sign of $n$, + is up and – is down relative to the control. If several replicates were available, the mean after normalization is taken for every gene, for all replicates of $p$ and $c$ respectively.

$$n = \begin{cases} -2^{-\log_2(p/c)} & \text{if } \log_2(p/c) < 0 \\ 2^{\log_2(p/c)} & \text{if } \log_2(p/c) \geq 0 \end{cases} \tag{3.1}$$

The other important measurement is the $Z$ score *vel* normal score, also calculated with log ratios. The raw intensity data is $\log_{10}$ transformed and then used in Equation 3.2. The $Z$ score for gene $p_i$ is calculated by subtracting the average intensity $\mu$ of all genes $P$ within an experiment from the value of $p_i$, and dividing this by the standard deviation $\sigma$ of all intensities in that experiment [61].

$$Z_i = \frac{p_i - \mu(P)}{\sigma(P)},\ P = \{p_1, ..., p_n\} \tag{3.2}$$

### 3.1.1 Expression Data

Fold change and normal score were the basis of extracting the gene list in Table 3.3 from the microarray experiments in Table 3.1. In this table, the # column contains a running number identifying every experiment, it is used to cross-reference the gene lists from Table 3.3. In the gene column, the genes that are relevant for this experiment are listed. They are either mutated, or over-expressed, *etc*. Lower case letters in the gene name indicate a knock out

mutant, while the AlcA prefix indicates an over-expression mutant, which enables the experimenter to induce over-expression of the gene coupled to AlcA by exposing the plant to ethanol.

The coverage column lists the number of experimental replicates for this row, usually two or three for both control and experiment. The description contains details about the experiment itself, for example which tissue was used and how long it was exposed to the experimental conditions, while the series column shows some identification information. LOF stands for a loss of function experiment, complemented by the lower case gene name that stands for a knock out mutant, and GOF stands for a gain of function experiment.

Table 3.1: Microarray expression experiments

| # | Gene | Coverage | Description | Series |
|---|---|---|---|---|
| 1 | stm | biological triplicates | dissected apices of 3 day old stm -/- seedlings were compared to apices from wild type plants grown in parallel | LOF1 |
| 2 | wus | biological triplicates | same as above, with wus -/- seedlings | LOF1 |
| 3 | clv3 | biological triplicates | same as above, with clv3 -/- seedlings | LOF1 |
| 4 | wig | biological triplicates | same as above, with wig -/- seedlings | LOF1 |
| 5 | stm | biological triplicates | same as above, with stm -/- seedlings | LOF2 |
| 6 | wus | biological triplicates | same as above, with wus -/- seedlings | LOF2 |
| 7 | clv3 | biological triplicates | same as above, with clv3 -/- seedlings | LOF2 |
| 8 | wig | biological triplicates | same as above, with wus -/- seedlings | LOF2 |
| 9 | zll_no | biological triplicates | dissected apices of 4 day old zll -/- seedlings showing the “no meristem” phenotype were compared to apices of wild type plants (LER) grown in parallel | zll |
| 10 | zll_pin | biological duplicates | dissected apices of 4 day old zll-/- seedlings showing the pin-like phenotype were compared to apices of wild type plants (LER) grown in parallel | zll |

| # | Gene | Coverage | Description | Series |
|---|---|---|---|---|
| 11 | AlcA::WUS | biological triplicates | dissected apices with young leaves of 14 day old 35S::AlcR, AlcA::WUS seedlings, induced for 12 hours in ethanol, were compared to similarly treated 35S::AlcR, AlcA::GUS control plants | GOF2 |
| 12 | AlcA::CLV3 | biological triplicates | same as above, with 35S::AlcR, AlcA::CLV3 seedlings | GOF2 |
| 13 | AlcA::LFY | biological triplicates | same as above, with 35S::AlcR, AlcA::LFY seedlings | GOF2 |
| 14 | clv3 | biological triplicates | shoot apex, inflorescence (after bolting) of 21 day old clv3-7 mutants were compared to corresponding wild type samples | AtGe_apex |
| 15 | lfy-12 | biological triplicates | shoot apex, inflorescence (after bolting) of 21 day old lfy-12 mutants were compared to corresponding wild type samples | AtGe_apex |
| 16 | clv3 | biological triplicates | flower stage 12 of the clv3-7 mutant (multi-carpel *gynoeceum*, enlarged meristem, increased number of organs) were compared to corresponding wild type samples | AtGe_flower |
| 17 | lfy-12 | biological triplicates | flower stage 12 of the lfy-12 mutant (flowers converted to leaf-like structures, some shoot characteristics) were compared to corresponding wild type samples | AtGe_flower |
| 18 | STM:VP16 | biological duplicates | dissected apices with young leaves of 14 day old 35S::AlcR, AlcA::STM:VP16 seedlings, induced for 12 hours in ethanol, were compared to similarly treated 35S::AlcR, AlcA::GUS control plants | GOF1_veg |
| 19 | AlcA::STM | biological duplicates | same as above, with 35S::AlcR, AlcA::STM seedlings | GOF1_veg |
| 20 | AlcA::WUS | biological duplicates | same as above, with 35S::AlcR, AlcA::WUS seedlings | GOF1_veg |
| 21 | LFY+WUS | biological triplicates | same as above, with 35S::AlcR, AlcA::LFY, AlcA::WUS seedlings | GOF1_veg |

| # | Gene | Coverage | Description | Series |
|---|---|---|---|---|
| 22 | AlcA::LFY | biological duplicates | same as above, with 35S::AlcR, AlcA::LFY seedlings | GOF1_veg |
| 23 | LFY:VP16 | biological duplicates | same as above, with 35S::AlcR, AlcA::LFY:VP16 seedlings | GOF1_veg |
| 24 | AlcA::WUS | biological triplicates | same as above, with 35S::AlcR, AlcA::WUS seedlings | GOF1_flower |
| 25 | AlcA::CLV3 | biological duplicates | inflorescences of 35S::AlcR, AlcA::CLV3 seedlings, induced for 12 hours in ethanol, were compared to similar treated 35S::AlcR, AlcA::GUS control plants | AlcA::CLV3 |
| 26 | Heat Shock | biological duplicates | leaves of wild type plants exposed to medium at 38°C, measurement taken after one hour | HS |
| 27 | Cold Stress | biological duplicates | leaves of wild type plants exposed to medium at 4°C, measurement taken after one hour | CS |

### 3.1.2 Gene Lists

Table 3.3 has a slightly different layout, it describes gene lists that are extracted from the results of one or several experiments. The name column is the identifier in the file system and the list as such can be found in the digital supplementary material under this name. The experiments column refers to the microarray experiments from Table 3.1 via their number in the # row. The experiments whose data contributed to this list are referenced here.

The lists are created mostly by intersection of the sets of genes obtained from the experiments by applying the filtering function stated in the description column. This filtering consists of measuring the $n$-fold change in expression of a gene between the actual experiment and the control. To rule out genes that are generally not highly expressed, the average $Z$ score in all experimental conditions should be greater than 2 for a gene to be considered for a list. Genes with lower expression are under the detection threshold of microarray experiments. Noise in any part of the process can interfere with a probe, and $n$-fold change of a gene that is only expressed so low cannot be detected reliably, it is an inexact measure for lowly expressed genes.

In the measurement column, one of the two analysis methods for expression changes are listed. The Gene-Chip Robust Multi-Array (GCRMA) method combines the stochastic robust multi-array analysis (RMA) with a physical model, taking into account the G and C content of the nucleotide probes on the microarray chip [62]. The other expression change measurement used is logiT

test, described in [63]. The len column lists the number of genes in the gene list.

Table 3.3: Gene lists extracted from the microarray experiments in Table 3.1

| **Name** | **Experiments** | **Meas.** | **Description** | **len** |
|---|---|---|---|---|
| all CLV WUS loop logic | 2, 3, 6, 7, 11, 12 according to wus regulatory logic | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 104 |
| CLV3 gof (GOF) - CLV3 gof (jan) | 12, 25 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 142 |
| CLV3 GOF (gof, jan) - wus lof - loop logic | 12, 25, 2, 3, 6, 7 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 42 |
| CLV3 lof - CLV3 lof atgenEX 46 | 3, 7, 14 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 149 |
| CLV lof CLV3 GOF jan | 3, 7, 12 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 190 |
| GOF changed in LFY | 13, 22, 23 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 32 |
| GOF changed in WUS | 11, 20, 24 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 33 |
| GOF old new LFY down overlap | 13, 22, 23 only down-regulated | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 5 |
| GOF old new LFY up overlap | 13, 22, 23 only up-regulated | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 4 |
| GOF up in clv | 12, 25 only up-regulated | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 419 |
| LFY lof (atgenX) - LFY GOF (GOF, jan) | 15, 17, 13, 22, 23 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 8 |
| LFY 3fold tended | 15, 17, 13, 22, 23 | gcRMA | $n$-fold change > 3 | 282 |

| **Name** | **Experiments** | **Meas.** | **Description** | **len** |
|---|---|---|---|---|
| LOF 5 ideal genes | changed in 2, 3, 6, 7; not changed in 1, 4, 5, 8 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 5 |
| LOF down in stm and not in wus | down in 1, 5; not down in 2, 6 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 65 |
| LOF only down in clv | 3, 7 only down | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 29 |
| LOF only up in clv | 3, 7 only up | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 14 |
| LOF up in stm and not in wus | up in 1, 5; not up in 2, 6 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 23 |
| logiT 0.01 PIN zll lof | 9 | logiT test | $p$ value < 0.01 | 26 |
| logitT0.01 nomem zll | 10 | logiT test | $p$ value < 0.01 | 154 |
| Overlap wus old and new GOF | 24, 11, 20 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 318 |
| STM lof - STM gof (vp16, AlcA) | 1, 5, 18, 19 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 18 |
| STM 3fold tended | 1, 5 | gcRMA | $n$-fold change > 3 | 11 |
| wig(234) - wig (567) | 4, 8 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 318 |
| WUS gof (GOF) - WUS gof (jan) | 20, 11 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 101 |
| wus LOF - wus gof (GOF, jan) | 2, 6, 11, 20 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 12 |
| WUS lof (234) - WUS lof (567) | 2, 6 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2, join | 478 |
| WUS LOF (23456) - CLV LOF (23456) - loop logic | 2, 3, 6, 7 | gcRMA | $Z$ score of $n$-fold change > 1 in each condition, average $Z$ score of all conditions > 2 | 42 |

| Name | Experiments | Meas. | Description | len |
|---|---|---|---|---|
| WUS 3fold tended | 20 | gcRMA | $n$-fold change > 3 | 89 |
| WUS+LFY 3fold tended | 21 | gcRMA | $n$-fold change > 3, join | 212 |
| heat shock 9fold | 26 | gcRMA | $n$-fold change > 9 | 16 |
| heat shock 8fold | 26 | gcRMA | $n$-fold change > 8 | 23 |
| heat shock 7fold | 26 | gcRMA | $n$-fold change > 7 | 48 |
| heat shock 6fold | 26 | gcRMA | $n$-fold change > 6 | 120 |
| heat shock 5fold | 26 | gcRMA | $n$-fold change > 5 | 259 |
| heat shock 4.01fold | 26 | gcRMA | $n$-fold change > 4.01 | 540 |
| heat shock 1.8fold negative | 26 | gcRMA | $n$-fold change < 2.0 | 20k |
| heat shock 1.2fold negative | 26 | gcRMA | $n$-fold change < 1.2 | 18k |
| heat shock 1fold negative | 26 | gcRMA | $n$-fold change < 1.01 | 16k |
| cold stress 24fold | 27 | gcRMA | $n$-fold change > 24 | 33 |
| cold stress 16fold | 27 | gcRMA | $n$-fold change > 16 | 75 |
| cold stress 12fold | 27 | gcRMA | $n$-fold change > 12 | 151 |
| cold stress 8fold | 27 | gcRMA | $n$-fold change > 8 | 324 |
| cold stress 1.2fold negative | 27 | gcRMA | $n$-fold change < 1.2 | 17k |
| cold stress 1fold negative | 27 | gcRMA | $n$-fold change < 1.01 | 15k |

## 3.2 ChIP-on-chip Data

The data from the ChIP-on-chip experiments are individual probe signals that are represented by a real number. A distribution of these signals can be seen in Figure 1.3, while the idea behind the experiments is outlined in Section 1.3. 2.1.4 describes the protocol used for these experiments.

To identify the genes that are regulated by WUSCHEL, a support vector regression method was used to learn the patterns of DNA fragments that once were bound by WUSCHEL. After the performing the experiment, the data we get consists of spots on a tiling array that represent every part of the genomic DNA. The probes are set approximately every 35 base pairs, but especially in repetitive regions, some of them are missing.

The support vector regression couples a high WRS to a certain pattern of binding data from the ChIP-on-chip experiments. As the sheared DNA from the nuclei is usually longer than 25 bases or even 100 bases, the expected signal, should WUSCHEL be bound to a gene's promoter region, looks more like a distribution that trails off at the edges than a sharp peak. Such a distribution can be seen in Figure 1.3b.

For this specific approach, I used the binding data in the range of –3000 bases before the transcription start and +800 bases into the gene, a broad window around the gene's promoter region. Because the input vectors have to be of the same length, and we do not want to forego all the regions where some probes are missing, only a coverage $c$ of 90% of probes is required for our input vector. In a region of $n$ base pairs, this is $c = (n/35) \cdot 0.9$ probes, which is then also the length of the vector of input data (multiplied by the number of experiments that we want to look at). As stated in Section 1.3, we expect a maximum of 108 probes in a window of 3800 bp. If we consider only 90% of this maximum, we can include genes that have only 97 probes in this window. This also means that even if all 108 probes are present, only 90% of them are considered, so we randomly remove probes from genes that have more probes than 90% of the maximum coverage until their input vector is shortened to the uniform length.

During training of the support vector regression kernels, the individual probe intensities were accessed from a MySQL data base. One intensity value is the average of 9 pixels from the scanned image where this probe is located on the chip. Python scripts can access this kind of data base using a module called MySQLdb, an open source software project. The WUSCHEL regulation score (WRS) was pre-calculated by Wolfgang Busch in R and stored in the data base. In preliminary trials, we averaged the probe data over all experimental replicates in a sliding window of 250 base pairs. Later calculations were based on the individual probes from each experiment. For comparative performance evaluations, see Section 4.2.

The WRS was calculated for every gene on the tiling array chip. A gene's WRS is based on the individual $Z$ scores from those expression experiments where either WUSCHEL was over-expressed or CLAVATA3 was over-expressed or knocked out. These are:

- AlcA::WUS apex
- AlcA::CLV3 apex
- clv3 INF apex
- clv3 flower
- AlcA::WUS veg
- AlcA::LFY and AlcA::WUS veg
- AlcA::WUS flower
- AlcA::CLV3 flower

Descriptions for these experiments can be found in Table 3.1. The range of the WRS is from $18.3$ to $6.7 \cdot 10^{-5}$. This is an arbitrary scale, and the trained support vector regression is not judged by the absolute value it assigns to each gene, but by the ranking it gives the genes. For this, I calculated the average WRS of the 10, 11, 12, ... top ranked genes from the SVR and plotted it in comparison to a random selection of genes with the same running average. All of the measured, trained SVRs performed better than this random list and the ones trained on approximately the top 180 genes from the WRS list and another 180 selected at random performed best (Figure 4.2). With optimized parameters from the grid search and cross validation, which is explained in Figure 3.1, there was another measurable improvement.

The parameters in Figure 3.1 were trained on a data set of eight hand selected genes that had a distribution similar to Figure 1.3b. These genes were complemented by ten examples that differed greatly from this distribution, mostly by staying completely flat or showing a wide valley of unbound probes. When trained on these hand-selected promoters, the SVR was able to distinguish them with a $q$-error of 0.70 (*cf.* Equation 2.8 in Section 2.2). More details can be obtained from Section 4.2.

## 3.3 Whole Genome Alignment

For the alignment of *Arabidopsis lyrata*, *Populus trichocarpa* (poplar), and *Oryza sativa* (rice) with *Arabidopsis thaliana*, we used the latest releases of the respective genomes. The alignment was computed in cooperation with Dr. Gunnar Rätsch.

The sequencing project at the Joint Genome Initiative of the United States Department of Energy (JGI) of *Arabidopsis lyrata* is still ongoing, but my examiner Prof. Weigel is a part of this sequencing effort. I was able to obtain the current contigs from the 4-fold coverage from his lab. They were provided by Korbinian Schneeberger.

The other genomes are available on-line, for rice the current release is TIGR5, and the poplar release is called 1.1 [34, 14]. The alignment was performed with BLASTZ as stated in 1.2.3, an independent implementation of the gapped BLAST algorithm [10]. It was developed for the alignment of whole genomes, but the run still took a very long time, especially for the genomes that were divided in sections that were smaller than chromosomes, *e.g.* the contigs of *Arabidopsis lyrata*. The complete set of pairwise alignments needed three weeks of computation time.

The completed pairwise alignments of the two *Arabidopsis* species and rice were merged into a multiple sequence alignment by a program called threaded block set aligner [13]. Its result was saved in a MATLAB matrix file, containing the position, contig and match/mis-match information of each nucleotide of the *Arabidopsis thaliana* genome, indexed by chromosome coordinates, in the aligned sequences of *Arabidopsis lyrata* and rice. The poplar alignment was not completed in time to be included in this multiple sequence alignment, but is instead available in the work flow as a single matrix.

The similarity was quite good for most genomes, the amount of matched sequences can be seen in Table 3.5. The similarity for *Arabidopsis lyrata* and *Arabidopsis thaliana* may seem a little low compared to the other genomes, given that *Arabidopsis lyrata* is the other plant's closest relative, they are only five million years apart. This can be explained in part by the different karyotypes of the two plants: *Arabidopsis thaliana* has only five chromosomes and a genome of about 125 Mb, while *Arabidopsis lyrata* is a diploid with 8 chromosomes and a genome of 230 Mb. More than half of the genome of *Arabidopsis thaliana* can be found in its relative, but there are a lot of repetitive regions that account for most of the size increase in *Arabidopsis lyrata*. These cannot be mapped to anything in *Arabidopsis thaliana*'s genome, but are probably of limited significance for the genetic functions and regulatory mechanisms.

Table 3.5: Genome alignment similarity percentage.
Genome A was aligned with Genome B, resulting in a similarity percentage listed in the % column. The calculation only looks at the –1000 base pairs from the transcription start of each gene in *Arabidopsis thaliana*.

| **Genome A** | **Genome B** | *%* |
|---|---|---|
| *Arabidopsis thaliana* | *Arabidopsis lyrata* | 52 |
| *Arabidopsis thaliana* | *Populus trichocarpa* | 31 |
| *Arabidopsis thaliana* | *Oryza sativa* | 27 |

## 3.4 Shogun Large Scale Machine Learning Toolbox

For the training of the SVM kernels, we used the software package 将軍 (*shōgun*) Large Scale Machine Learning Toolbox (Shogun) through its Python interface and its MATLAB interface [64]. It works on the basis of either the LIBSVM [65] or SVM$^{light}$ [66] implementation, both available as source code or binaries for various systems. The object interface of Shogun provides to these libraries is generic, so they can be freely interchanged. For the purpose of this thesis, we used LIBSVM.

The SVMs can be combined with several kernel functions, and my implementation makes heavy use of this. Of special note is the weighted degree kernel with shifts (WDS), a string kernel that attaches importance to the position of a certain sub-string. Contrary to the regular weighted degree kernel, it allows for a shift in this position. The shift can be handed to the kernel as a parameter. The WDS kernel has been designed with sequence analysis tasks in mind [51]. It is able to efficiently compute similarities between sequences in addition to the positional information of substrings of length $k$, called $k$-mers, which it extracts from the sequence in the input vectors. After a modification created in cooperation with Dr. Gunnar Rätsch, the kernel is also able to accept a score for the conservation of each position in the sequence, as shown in Equation 3.6.

The WDS kernel $K_{\mathrm{WDS}}$ of length $d$ with shift $S$ is defined as

$$K_{\mathrm{WDS}}(\vec{x}_1, \vec{x}_2) = \sum_{k=1}^{d} \beta_k \sum_{i=1}^{l-k+1} \sum_{\substack{s=0 \\ s+i \leq l}}^{S} \frac{\mu_{k,i,s,\vec{x}_1,\vec{x}_2}}{2(s+1)} \tag{3.3}$$

$$\mu_{k,i,s,\vec{x}_1,\vec{x}_2} = \mathbf{I}\left[u_{k,i+s}\left(\vec{x}_1\right) = u_{k,i}\left(\vec{x}_2\right)\right] + \mathbf{I}\left[u_{k,i}\left(\vec{x}_1\right) = u_{k,i+s}\left(\vec{x}_2\right)\right], \tag{3.4}$$

where $\vec{x}_1$ and $\vec{x}_2$ are the input vectors. It considers sub-strings up to length $d$, and allows them to be shifted up to $S$ positions in the input sequences. Since there is an inner sum over all the possible shifts, the kernel is computationally proportional to $S$, having a run time complexity of $\mathcal{O}(lS)$, with $l$ being the length of the input sequences $\vec{x}_1$ and $\vec{x}_2$. The coefficients $\beta_k$ weighting the *k*-mer matches are defined as

$$\beta_k = 2\frac{d-k+1}{d(d+1)}, \tag{3.5}$$

as suggested in [43], and $u_{k,i}(\vec{x})$ is the sub-string of $\vec{x}$ of length *k* that starts at position $i$. The idea of $\mu_{k,i,s,\vec{x}_1,\vec{x}_2}$ is to count the matches in two sequences $\vec{x}_1$ and $\vec{x}_2$ between the sub-strings $u_{k,i}(\vec{x}_1)$ and $u_{k,i}(\vec{x}_2)$, where $u_{k,i}(\vec{x}) = x_i x_{i+1} ... x_{i+k-1}$ for all $i = 1, ..., l-k+1$ and $1 \leq k \leq d$. This comparison is realized by the indicator function $\mathbf{I}\left[arg\right]$, which evaluates to $0$ if its argument $arg$ is false, and to $1$ otherwise. In the case of Equation 3.4, the argument is the comparison of the two sub-strings.

The WDS kernel consists of a weighted summation of only the matches between two input sequences $\vec{x}_1$ and $\vec{x}_2$, depending on its length and relative position. The contribution becomes smaller if the match is not at the same position in both sequences, and not at all if it is more than $S$ positions apart. A similar implementation is the spectrum kernel [67], but here the sub-strings are all of a fixed length and the matches are completely independent of the relative positions of each other. The WDS kernel is better suited for the tasks in this thesis and therefore the higher computational requirement is acceptable. The WDS kernel has a time complexity of $\mathcal{O}(lS)$ compared to the spectrum kernel's $\mathcal{O}(l_{x_1} + l_{x_2})$, or the even more frugal weighted degree kernel (without shifts), which can be computed in $\mathcal{O}(l)$.

To include the conservation information, we extended the $K_{WDS}$ in Equation 3.6 with a term to multiply the score of the local alignment onto the inner sum, looking it up in the alignment matrix. The new kernel is called weighted degree with shifts and conservation. Keeping the definitions for $\mu$ from Equation 3.4 and $\beta$ from Equation 3.5 the same as for Equation 3.3, we get

$$K_{\mathrm{WDSC}}(\vec{x}_1, \vec{x}_2) = \sum_{k=1}^{d} \beta_k \sum_{i=1}^{l-k+1} \sum_{\substack{s=0 \\ s+i \leq l}}^{S} \frac{\mu_{k,i,s,\vec{x}_1,\vec{x}_2}}{2(s+1)} \cdot \gamma_{k,i,\vec{x}_1} \cdot \gamma_{k,i,\vec{x}_2} \tag{3.6}$$

$$\gamma_{k,i,\vec{x}} = 1 + a \sum_{g=1}^{G} \frac{\sum_{j=0}^{k} \mathbf{I}[u_{1,i+j}(\vec{x}) = u_{1,i+j}(A_g(\vec{x}))]}{k} \tag{3.7}$$

where $u_{1,i+j}(A_g(\vec{x}))$ is a substring consisting of a single character *j* from genome $g$, the part of the sequence of $g$ that was aligned to character $u_{1,i+j}(\vec{x})$ in the alignment matrix $A$ described above. To find the correct genome in $A$ we give it the parameter $g$. The alignment of genome $g$ is part of the total number of aligned genomes *G*. In this thesis, there is a maximum of two genomes aligned to the original in this matrix. The fact that we add 1 means we only value an existing alignment positively, but do not further punish the absence of an alignment. The influence of the alignment data can be adjusted through the factor $a$, which is $0 \leq a \leq 1$. The results shown in this thesis were obtained with the setting of $a = 1$. The indicator function $\mathbf{I}\,[arg]$ works as described above; in this case, mismatches, gaps and the absence of an alignment were treated as "false."

As described in 1.2.1, the input vectors in the work flow in this thesis also consist of some real numbers. For these, a linear kernel was used. These are in particular the relative distances among the found motifs, the score as assigned by the MotifLocator program, the length, and number of occurrences in the selected gene.

## 3.5 Python Work Flows

The work flow for SVM training and evaluation is the largest implementation in this thesis. It is complemented by several smaller scripts used for example in the SVR parameter optimization or data format parsing. The whole project is called `org.metatfbs`.

The Python package called `org.metatfbs.ExpressionAnalysis` is used to run the SVM training. It is divided into several large class files that contain multiple functions each. A UML class diagram is shown in Figure 3.2. In this diagram, the individual packages of `ExpressionAnalysis` can be identified through their color-coded classes.

### 3.5.1 UML diagram specifics

The name of a class is written in the top section of a box, followed by its attributes. These are given with their respective data type, and not all classes have attributes. The next section in the box contains the class methods, they are given with their parameters and the type of their return value. Some methods span multiple lines, this is indicated by indentation. A diamond at the end of an arrow indicates a class that is part of another class and does not exist separately.

**Composition:** A filled diamond shows that the class the arrow originates from is constituted entirely of this class, *e.g.* a `FastaFile` consists of only `FastaEntry`s.

**Aggregation:** An empty diamond shows that this class is contained somehow in the other class, *e.g.* the `main` class has a `Database` that consists of connections, a cursor, *etc*. Classes that aggregate others can have more than one aggregation.

**Generalization:** An empty triangle points towards a generalization of a class, *e.g.* `MotifWrapper` is a special case of a `Wrapper`, it inherits all the `Wrapper` properties but contains additional methods and attributes or overwrites existing ones. `Wrapper` is the generalization of `MotifWrapper`.

### 3.5.2 Classes of the Python Module `ExpressionAnalysis`

The individual parts of `ExpressionAnalysis` are `Inputs`, `Outputs`, `Converters`, `Wrapper`, `Tools` and `main`.

**`Inputs`** contains classes that represent file formats that are used for input and exchange between the modules of this work flow:

- **`FastaFile`** contains one or more instances of `FastaEntry`, a class that contains both the header and the sequence of a FASTA file. `FastaFile` has several methods for accessing the `FastaEntry`s inside. The class also provides a method to parse FASTA files and to recognize such a file by looking for the first ">" character, with which every FASTA file starts.
- **`FastaEntry`** contains `header` and `sequence` strings, and methods to access and modify them. Noteworthy are methods to retrieve a certain sequence window around a specific center, *i.e.* the sub-string around this position. This is used in creating the input vectors for the SVM.
- **`GffFile`** is composed of several `GffEntry`s. In this respect, it is similar to the class `FastaFile`, though it represents a file in general feature format (`.gff`), which is a tabulator delimited table of features in a sequence. The output files of some of the Gibbs sampling programs come in this format. `GffFile` has methods to retrieve the gene names of the genes it lists features for, and to retrieve the best hit for a gene.
- **`GffEntry`** contains all the columns of the GFF specification and some additional information like the corresponding matrix file name whose features are listed and the length of a feature.
- **`Microarray`** contains a table of a microarray or tiling array experiment, complete with identifiers and expression values for the experimental conditions and the control. It uses the `A2A` class from `Converters` so it has identifiers both in the Affymetrix specific format and the TAIR AGI accession numbers.
- **`Genelist`** is a simple class that contains a list of gene names with methods to add individual genes. `Microarray` instances return a `Genelist` if asked for all genes above a certain $n$-fold change in expression level.
- **`PredOptions`** is part of every `Wrapper` instance. It defines the command line options that are given to the binary of the sampling programs.

**Wrapper** is the generalized parent class for all Gibbs sampling programs and it also contains the Shogun SVM class `SVMTraining`.

- **DIPSWrapper** contains the specifics needed to start the DIPS binary, a motif sampler that uses positive and negative FASTA files for motif finding.
- **GibbsWrapper** calls the Perl interface `BioPerl::TFBS` to search for motifs using the Gibbs binary and is able to make sequence logos (`.png`) from both `.imm` and `.pfm` files.
- **MotifWrapper** calls the INCLUSive MotifLocator and MotifSampler programs. MotifSampler is a Gibbs motif finding program and MotifLocator is used in conjunction with an inclusive motif model (`.imm`) file, which contains a position frequency matrix, to create `.gff` files given a certain threshold.
- **SVMTraining** accesses Shogun through its Python interface.

**Converters** contains two classes that prepare data and convert file formats. Additional conversion is provided by the `Wrapper` class, which is able to convert `.pfm` and `.imm` files.

- **PrepareData** is a rewritten MATLAB script that creates the input vectors for SVM training.
- **A2A** contains dictionaries (Python hashes) that map Affymetrix accessions to AGI codes to identify *Arabidopsis thaliana* genes.

**Outputs** are `org.metatfbs` formats that can be displayed or saved as results.

- **ResultSet** is the class in which a probable candidate for a TFBS is presented. It contains the PFM, a representation of the same as sequence logo, a distribution obtained from the `.gff` file relative to the gene start, and statistical analyses. These sets can be saved as files.
- **HTMLOutput** displays the results as an `.html` file, it can format `ResultSets` directly.
- **SVMOut** stores a trained SVM kernel in a file and can retrieve it at a later time.

**Tools** contains some smaller scripts that do useful work on sequence files.

**Database** connects to a MySQL database and provides a cursor to execute commands. The results can be obtained either as lists from the MySQLdb module or in one of the `Inputs` formats.

**main** ties all the modules together and takes many command line arguments, including the option to proceed either on previously compiled gene lists or on raw microarray expression data. It is the interface for the `org.metatfbs` web annotation service.

### 3.5.3 Classes of the `ChIPchipSVR` analysis pipeline

The SVR package is called `org.metatfbs.ChIPchipSVR`. This package is not as intricate as `ExpressionAnalysis` and the connections between the classes are loose. Most of the parameters have to be adjusted in the source code. The class diagram in Figure 3.4 shows some classes that only exist within the other files, but are shown in their own box for clarity.

**`main`** is the principal class to execute a training of a single SVR. It takes the length of the sequence that is extracted from the transcription start downstream as well as upstream, the number of top genes to use WUSCHEL regulation score, the level of cross validation, and the SVR kernel parameters `C` and `width`. It is also able to compute the $q^2$ error and the $q$ error from Equation 2.7 and Equation 2.8, respectively.

**`mainGrid`** searches for good values for `C` and `width`, two parameters of the SVR's kernel function. It uses the ranges and the increment size of these two parameters to call a function called `subGrid`. This became necessary as the Shogun objects, even though the same variables are reassigned to new objects, do not seem to be freed in memory, and if called repeatedly such as in this grid search, amount to several gigabytes. Since this is not feasible, the slower but more robust approach of calling an outside function through a system call was implemented in this grid search class.

**`mainGW`** uses a trained SVR kernel to calculate regulation scores for every gene on the chip. It then stores this score in a new data base table along with the AGI accessions.

**`average`** calculates the running average of all the genes ranked by their predicted regulatory score. The score that is averaged, however, is taken from the original WUSCHEL regulation score table. This shows how many genes with an above average score the SVR identified and ranked similarly.

**`assembleGrid`** is used to display the results from the `mainGrid` class. This results in a heat map like the one in Figure 3.1.

### 3.5.4 Additional Scripts

Several smaller scripts exist that perform a number of tasks associated with the work flow above. Although some of them are incorporated in the Tools class of the ExpressionAnalysis package, there are some separate files.

A part of the SVM work flow was originally programmed in MATLAB, and this implementation still exists. It has been used for initial testing and prototyping, as well as studying the feasibility of the approach. Rough-hewn prototype scripts were initially composed by Dr. Gunnar Rätsch and later revised and extended by me.

### 3.5.5 Third Party Modules

The third party Python modules used in conjunction with Shogun and the Python work flow are listed below.

**wxPython,** a programming interface to wxWidgets, a toolkit for graphical user interfaces in Python. It is native to the host computer's operating system, because wxWidgets is written in C++.

**matplotlib** (`pylab`), a MATLAB-compatible 2D plotting library that uses the wxPython interface to display graphical plotting results, *e.g.* Figure 3.1.

**Numerical Python** (`numpy`), adds support for large, multi-dimensional arrays to Python, complemented by a number of mathematical functions like matrix multiplication, transformation, *etc.*, to operate on these arrays.

**MySQLdb** is a module that enables users to connect to a MySQL data base and query it, retrieving either the ID of the last inserted row or the result set in a list of lists.

Figure 3.1: Heat map of grid search results.
This is the result of a grid search over the parameter space of $\sigma$ and $C$ of the support vector regression with a Gaussian radial basis function kernel. The regression was applied to the ChIP-on-chip dataset with genes that were hand-selected for positive and negative features (for details see Section 3.2). The $x$ axis shows the individual settings for the value of the $C$ parameter ranging from $2^{-4}$ at the origin to $2^{11}$ in a total of $70$ steps and the $y$ axis shows the values for the $\sigma$ parameter, in an interval of $2^{-7}$ and $2^{8}$ in a total of $48$ steps. The minimal value of this plot lies in $(15, 38)$, what seems to be the middle of a minimum that is achievable using parameter optimization with this dataset. The minimum translates to parameter settings for $\sigma = 1.5$ and $C = 1.4$, and a $q$ error of $0.70$. The kernels seem to favor parameters about equal in size for this dataset, indicated by the diagonal from about $(10, 25)$ to $(40, 70)$ that remains dark blue, while the rest of the image turns lighter in color as we reach the edges. Dark blue indicates values $q < 0.8$ for the $q$ error (*cf.* Equation 2.8), and yellow to red indicate high values, *i.e.* greater than one.

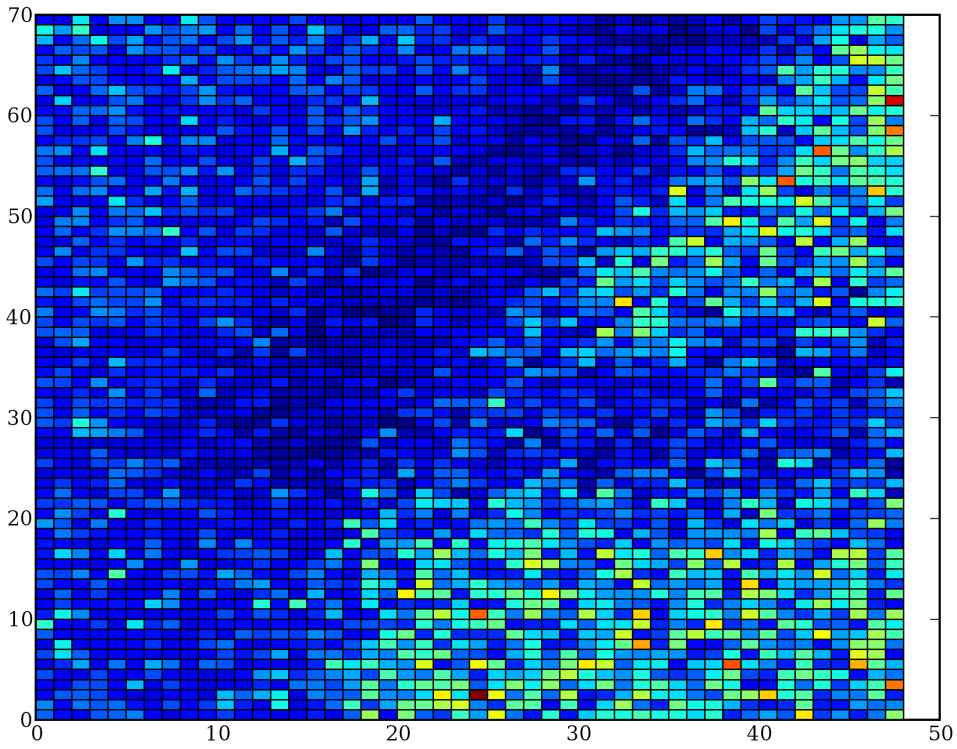


Figure 3.2: Class diagram of the Python package `ExpressionAnalysis`

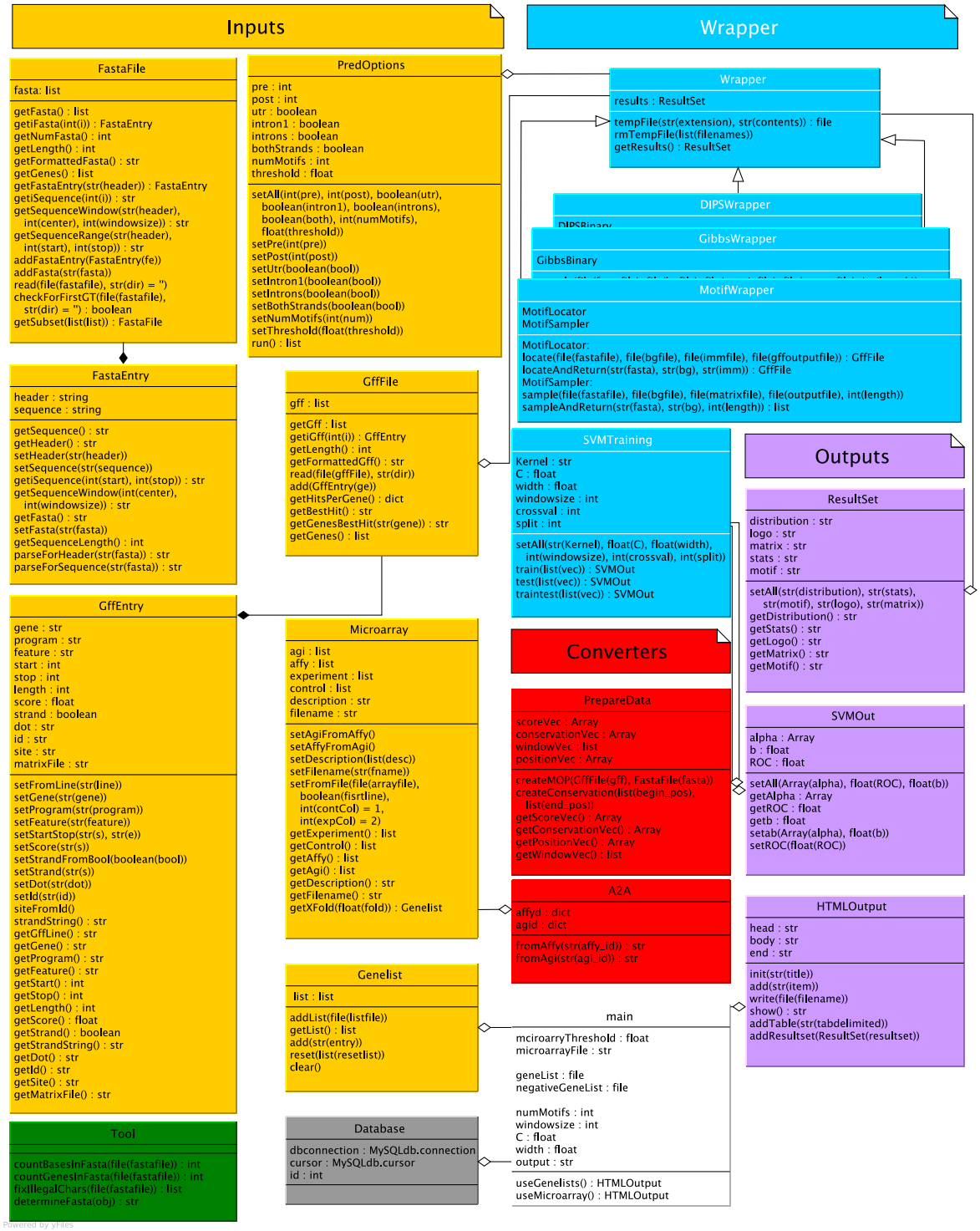


Figure 3.3: Work flow for the SVM training

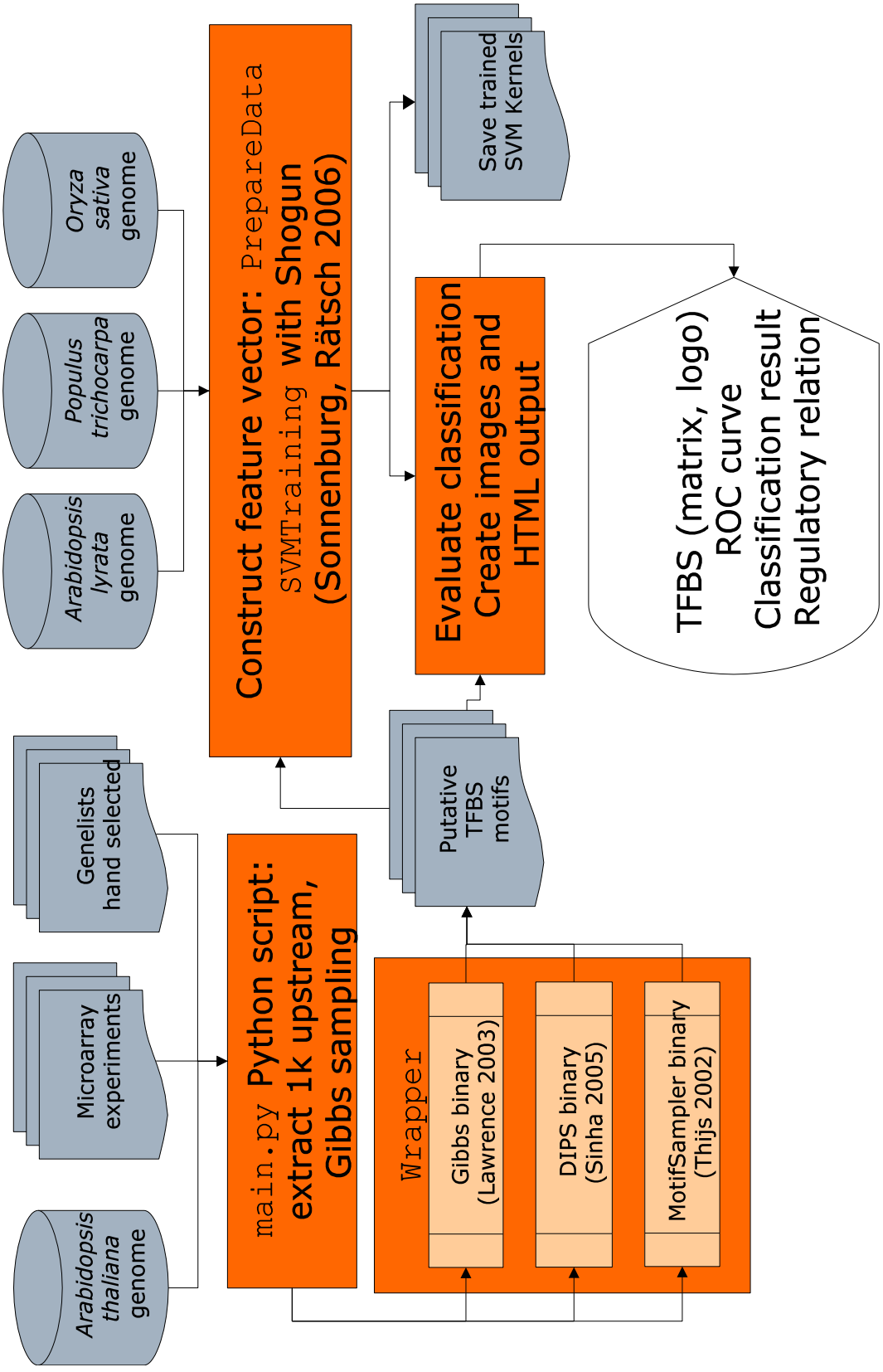
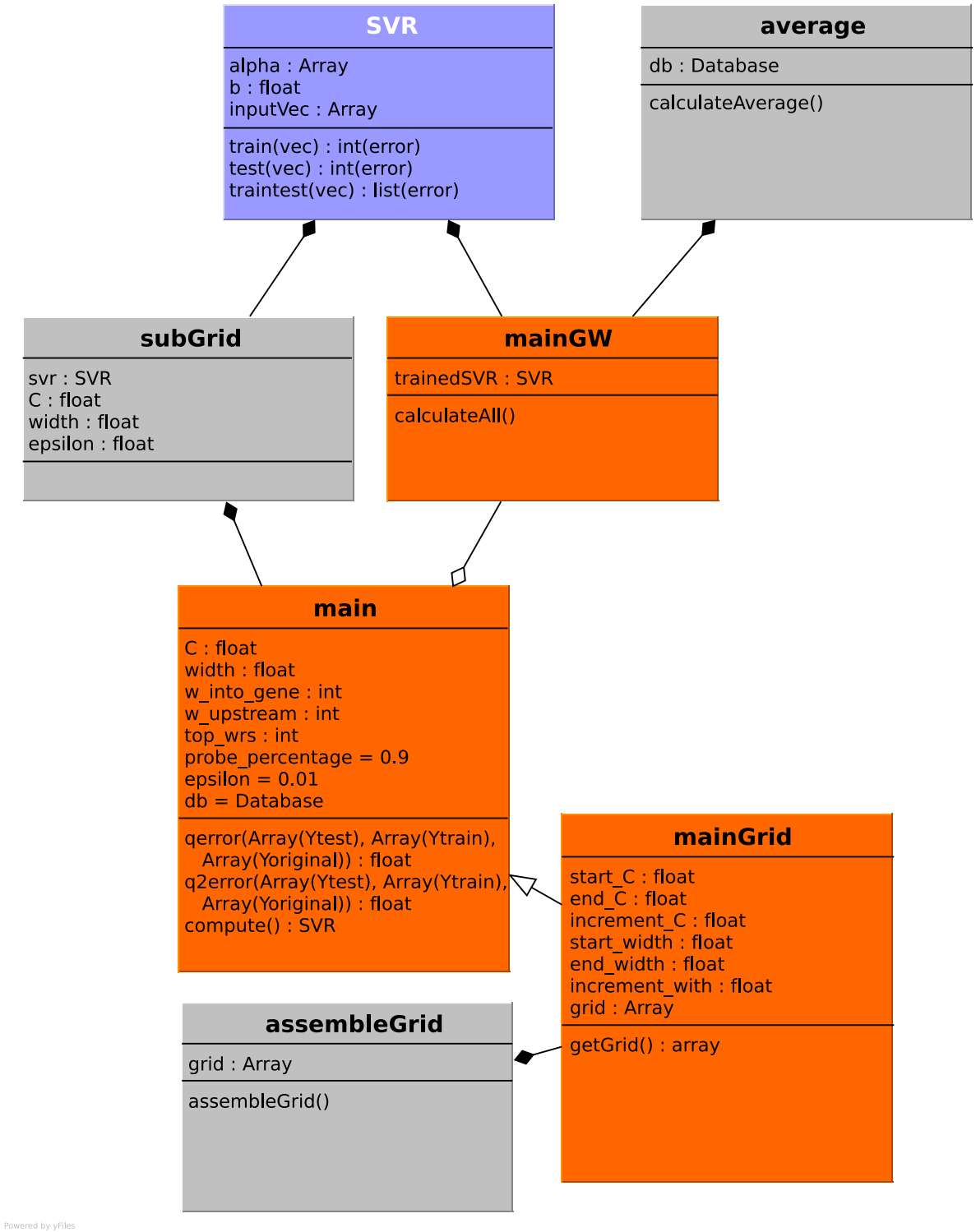


Figure 3.4: Components of the python package `ChIPchipSVR`



# Chapter 4

# Results

The results are presented separately for the microarray expression data and the ChIP-on-chip protein-DNA binding data in Section 4.1 and Section 4.2 respectively. The bigger picture of bringing these results together is described in Section 4.3. The final part of this chapter, Section 4.4, shows some genes and candidate motifs found in their promoter region that could be targets and binding sites for the transcription factor WUSCHEL or its downstream targets of regulation.

## 4.1 SVM Training Results

Successful training of an SVM kernel depends on the input vectors. There is nothing the machine can do to find meaning in noisy or contradicting data. Therefore, an effective filtering method is a good way to get rid of noise. As can be seen from the gene lists in Table 3.3, the ones using data from several experiments usually become smaller and of a higher quality.

The SVM training work flow described in 3.5.2, Figure 3.3 and summarized in Section 4.3 yielded many results for the gene list input data. For an overview, they are summarized in Table 4.1. The first column, labeled S, lists the area under the receiver operator characteristic curve, a score to measure prediction accuracy, described in Section 2.2. The columns labeled D, G and MS list the numbers of motifs that this classification is based on, found by the DIPS program, the Gibbs sampler, or the MotifSampler binary. Name lists the gene list that was used for the classification.

Table 4.1: SVM training results for gene lists from Table 3.3.

| S | D | G | MS | Name | S | D | G | MS | Name |
|---|---|---|---|---|---|---|---|---|---|
| 0.73 | 26 | 19 | 31 | STM 3fold | 0.66 | 43 | 31 | 50 | STM lof - STM gof (vp16, AlcA) |

| S | D | G | MS | Name | S | D | G | MS | Name |
|---|---|---|---|---|---|---|---|---|---|
| 0.51 | 504 | 360 | 591 | WUS+LFY 3fold | 0.52 | 1137 | 813 | 1132 | WUS lof 234 - WUS lof 567 |
| 0.64 | 212 | 151 | 248 | WUS 3fold | 0.80 | 19 | 14 | 22 | LFY lof atgenX - LFY gof |
| 0.71 | 117 | 83 | 137 | LFY 3fold | 0.72 | 452 | 323 | 529 | CLV lof - CLV3 GOF |
| 0.84 | 78 | 56 | 92 | GOF changed in WUS | 0.62 | 338 | 241 | 396 | CLV3 gof - CLV3 gof |
| 0.80 | 12 | 9 | 14 | GOF old new LFY down overlap | 0.67 | 29 | 20 | 33 | wus LOF wus gof (GOF) |
| 0.75 | 10 | 7 | 11 | GOF old new LFY up overlap | 0.74 | 67 | 48 | 78 | CLV3 GOF - wus lof loop logic |
| 0.85 | 14 | 10 | 7 | GOF old new WUS overlap | 0.71 | 100 | 71 | 117 | WUS LOF 23456 - CLV LOF 23456 loop logic |
| 0.57 | 997 | 712 | 878 | GOF up in clv | 0.69 | 756 | 541 | 886 | wig 234 wig 567 no rps |
| 0.66 | 76 | 54 | 81 | GOF changed in LFY | 0.57 | 240 | 172 | 281 | WUS gof - WUS gof |
| 0.69 | 247 | 177 | 290 | all CLV WUS loop logic | 0.61 | 747 | 534 | 675 | CLV3 lof CLV3 lof atgenEX46 |
| 0.85 | 13 | 9 | 15 | LOF 5 ideal genes | 0.58 | 38 | 27 | 45 | heat shock 9fold |
| 0.65 | 155 | 111 | 181 | LOF down in stm not in wus | 0.77 | 55 | 39 | 64 | heat shock 8fold |
| 0.62 | 200 | 143 | 243 | LOF down in wus not in stm | 0.76 | 114 | 82 | 134 | heat shock 7fold |
| 0.74 | 69 | 49 | 81 | LOF only down in clv | 0.70 | 285 | 204 | 334 | heat shock 6fold |
| 0.81 | 33 | 24 | 39 | LOF only up in clv | 0.65 | 616 | 440 | 722 | heat shock 5fold |

| S | D | G | MS | Name | S | D | G | MS | Name |
|---|---|---|---|---|---|---|---|---|---|
| 0.79 | 55 | 39 | 64 | LOF only up in stm not in wus | 0.63 | 983 | 918 | 1193 | heat shock 4.01fold |
| 0.72 | 68 | 61 | 100 | LOF up in wus not in stm | 0.51 | 105 | 75 | 132 | cold stress 24fold |
| 0.69 | 366 | 262 | 429 | logiT 0.01 PIN zll lof | 0.79 | 22 | 15 | 23 | cold stress 16fold |
| 0.68 | 62 | 44 | 72 | logiT 0.01 nomem zll | 0.71 | 25 | 18 | 19 | cold stress 12fold |
| 0.53 | 354 | 253 | 415 | Overlap WUS old and new GOF | 0.53 | 155 | 75 | 148 | cold stress 8fold |

Negative data sets for any of the lists not from either heat shock or cold stress data were obtained from the microarray experiments in a similar manner as the original list, from all the genes with a fold change smaller than 1.01 and a generally high expression level, a $Z$ score equal to the one for the original list. This results in an imbalance between positives and negatives, but the performance is still acceptable. Sub-sampling of negatives, by whichever manner, can usually not be justified acceptably.

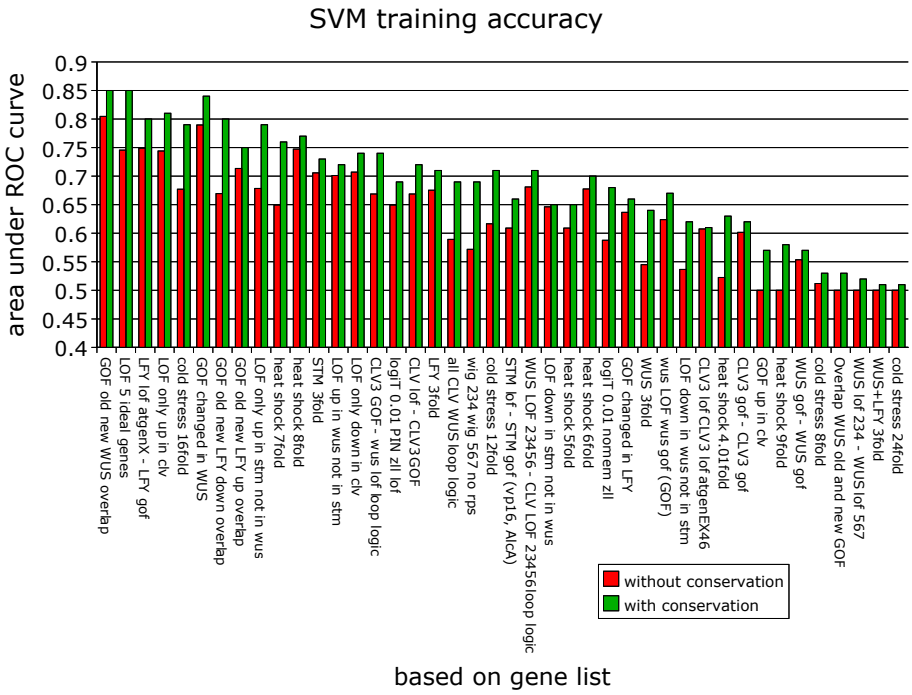
The classifications based on 5-fold cross validation were conducted with completely independent splits of the data set. The technique is outlined in 2.2.1 under the Evaluation heading. During training, the remaining split held back for testing is never seen by the classifier, neither the labels nor the input vectors. To make the split truly independent, we also let the work flow search the motifs on only 80% and 20% respectively.

This is one reason the number of found motifs by each sampling program may seem high. The work flow also looks for motifs of every length between 6 and 12, and tries to find up to ten over-represented ones for each setting. This makes some of the stronger, longer motifs appear several times in shorter versions. The numbers given in the D, G and MS columns are the sum of all motifs discovered in any of the five splits and all of the various lengths.

The weaker performance of even the highly filtered lists from the heat shock and cold stress data shows that fold change alone is not the best measure to go by. Many of the other lists perform better and are much more helpful in deciding which genes are co-regulated. Episodic proof that the work flow does the correct thing can be seen from the results of the heat shock lists with 9-, 8-, and 7-fold change in expression: Here the heat shock motif from [17] was found repeatedly and all of the 15 direct target genes that were identified in [17] were classified as co-regulated if they were included in the list. A mock experiment with only the 15 direct target genes and the correct motif, as well as the most stringent 1.01-fold gene list from the heat shock experiment as a negative set had a score of 0.92 in a 5-fold cross validation setup.

An interesting comparison is highlighted in Figure 4.1. Here, the area under the receiver operator characteristic curve is plotted, once for the SVMs with the weighted degree kernel with shifts, in orange, and once as seen in Table 4.1, with added conservation data from the alignment matrix. One can easily see that there is a constant improvement almost every time. On average the melioration is 7 percentage points, with the maximum being a change of 14 percentage points. It clearly pays off to use the conservation data. The sequence window itself still remains the single biggest contributor in the input vector to good prediction scores, however.

Figure 4.1: Comparison of SVM performance with and without the use of conservation data.




## 4.2 SVR Training Results

The results of the regression were obtained by running the programs from 3.5.3 with the binding data outlined in Section 3.2. Initially, the performance was barely below that of a constant predictor, so we trained the SVR kernel by using a list of 19 hand-selected genes that are good examples of great binding profiles, average binding, and genes without any enrichment. Since we did not want to see too much of a correlation between the WUSCHEL regulation score, which we use as a label, and the binding, the genes were selected without looking at the WRS. The 19 genes are listed by their AGI code in Table 4.3. They were selected by Wolfgang Busch.

Using this list, I performed a grid search over the parameter space of $\sigma$ and $C$ of a Gaussian radial basis function kernel. A heat map with the results of this

Table 4.3: 19 genes used for SVR parameter optimization

| Good profile | Average binding | No enrichment | |
|---|---|---|---|
| AT3G20370 | AT1G14370 | AT1G08730 | AT3G59990 |
| AT1G17380 | AT5G05410 | AT5G39970 | AT3G44340 |
| AT2G25735 | AT5G54490 | AT3G60740 | AT2G20860 |
| AT3G04630 | | AT1G67210 | AT4G20200 |
| AT5G40170 | | AT2G07020 | AT1G26340 |
| | | AT3G03810 | |

grid search can be seen in Figure 3.1. The values where the error reaches a minimum are $\sigma = 1.5$ and $C = 1.4$, the $q$ error reached $0.70$ for this setting. This is the average of a 5-fold cross validation on the input data set.

Using these parameter settings and a binding profile from $-3000$ upstream from the transcription start of a gene and $+800$ downstream into the gene, the SVR ranked all the genes in the genome after being trained on 360 of them (the top 180 ranked by their WRS and another random 180). This was still better than a constant prediction with a $q$ error of 0.91, but not as good as on the hand-selected data set. Compared to the running average of a random assignment of genes to a WRS, it is a much better result, as can be seen in Figure 4.2. Since we are not really interested in recalculating the WRS perfectly, we deemed this sufficient. The ultimate goal is to predict the probability of the observed signal being a real regulation enrichment on the promoter. The WRS is our only approximation for this, but by no means perfect.

The ranked genes were then used as a list to intersect with the genes predicted by the SVM work flow to be co-regulated, and motifs were searched. See Section 4.4 for results of this undertaking.

## 4.3 Use Case – Putting Results Together

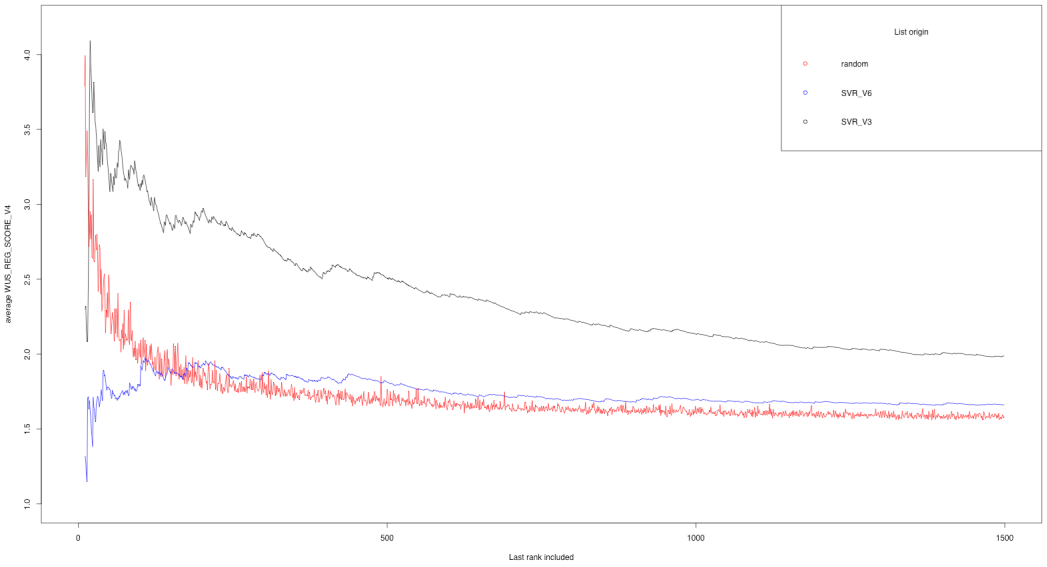
The steps to create a network from such experimental data as expression and binding using this work flow are the following:

1. Use microarray expression experiments to extract high-quality lists of co-regulated genes and negative examples, *i.e.* of genes that are highly expressed but do not change, to exclude genes below the sensitivity in the negative examples.

2. Start the Python work flow by submitting gene lists, both positive and negative: motifs will be created from the FASTA files of the regulatory regions of the listed genes, SVM classification of genes into co-regulated and not co-regulated takes place, based on motifs, conservation, *etc.*

3. Good results from the SVM warrant a search of the motifs found in the co-regulated genes in the whole genome, once again comparing to expression experiments: genes are deemed positive if an expression change is measurable or the level is below sensitivity.

Figure 4.2: Comparison of SVR ranked to genes randomly assigned a WRS in a running average.
The red line indicates the average WRS in genes ranked randomly. It is compared to the performance of an SVR with optimized parameters (black) and trained on the 360 examples described above. The blue line is a comparative SVR that was not optimized and performed about the same as a constant prediction, with a $q$ error of 1.05.




4. Ideally, data from experiments like ChIP-on-chip should be available for one or more transcription factors and we can then confirm direct targets by intersecting the bound genes with the lists from step 3. All other targets are most probably secondary responses to changes in the expression of a transcription factor among the primary targets.

5. Time series experiments can increase the resolution of detecting secondary and higher level responses. Repeat steps 1 through 4 for different time points.

If the work flow is laid out like this, it becomes clear that there have to be manual steps at several points. For instance, at point 1 the experimental logic of the expression data has to be considered in creating the gene lists. During the work flow in step 2, the user can adjust several parameters. In step 3, the results themselves have to be evaluated by the user, as the SVM classification output implies a ranking. That is part of what we wanted it to learn from our labels.

If data for step 4 are available, the user can proceed with the ChIP-on-chip work flow and rank the bound genes. An example of the intersection can be seen in Section 4.4. Unfortunately, I did not have access to any time series experiments and could not test this part of the use case.

## 4.4 TFBS Candidate Motifs

With the results from both the expression analysis work flow and the ChIP-on-chip regression analysis, several candidates were identified that are probably regulated by WUSCHEL, the transcription factor in *Arabidopsis thaliana* that affects stem cell maintenance in the shoot apical meristem's organizing center.

To arrive at the results from step 4 of the use case in Section 4.3, I took the 80 top-ranking genes from the binding data and compared them to selected lists from step 3. To obtain these, I analyzed the gene lists from the WUSCHEL over-expression experiments with my SVM work flow from Figure 3.3. The motifs from the genes classified there I used to look in the –1000 promoter region of the whole *Arabidopsis thaliana* genome. This in turn led to lists of genes that contained these motifs. Compared with the motifs from the 80 genes of the SVR, I was able to identify two motifs that appeared repeatedly and are probably the binding motifs of either WUSCHEL or a strong downstream target of it. The sequence logo and additional statistical information can be seen in Table 4.4. Intersecting the lists of the SVR and the genes that have this motif may be the key to finding the targets of the transcription factor that recognizes this binding site.

The second of the two motifs identified has a similar core (GATCC) to a known pattern from the ARR10 gene in *Arabidopsis thaliana* [68], a congruence that was also found by JASPAR CORE , a binding site data base [32].

This is one example of how such a work flow can be of use to a biological scientist. The genes from the intersected lists can be used for further biological analyses, *e.g.* in knock-out mutants, *in situ* hybridization and other methods.

Table 4.4: Two candidate motifs from WUSCHEL targets

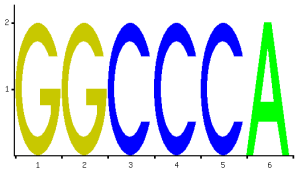
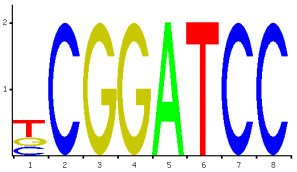
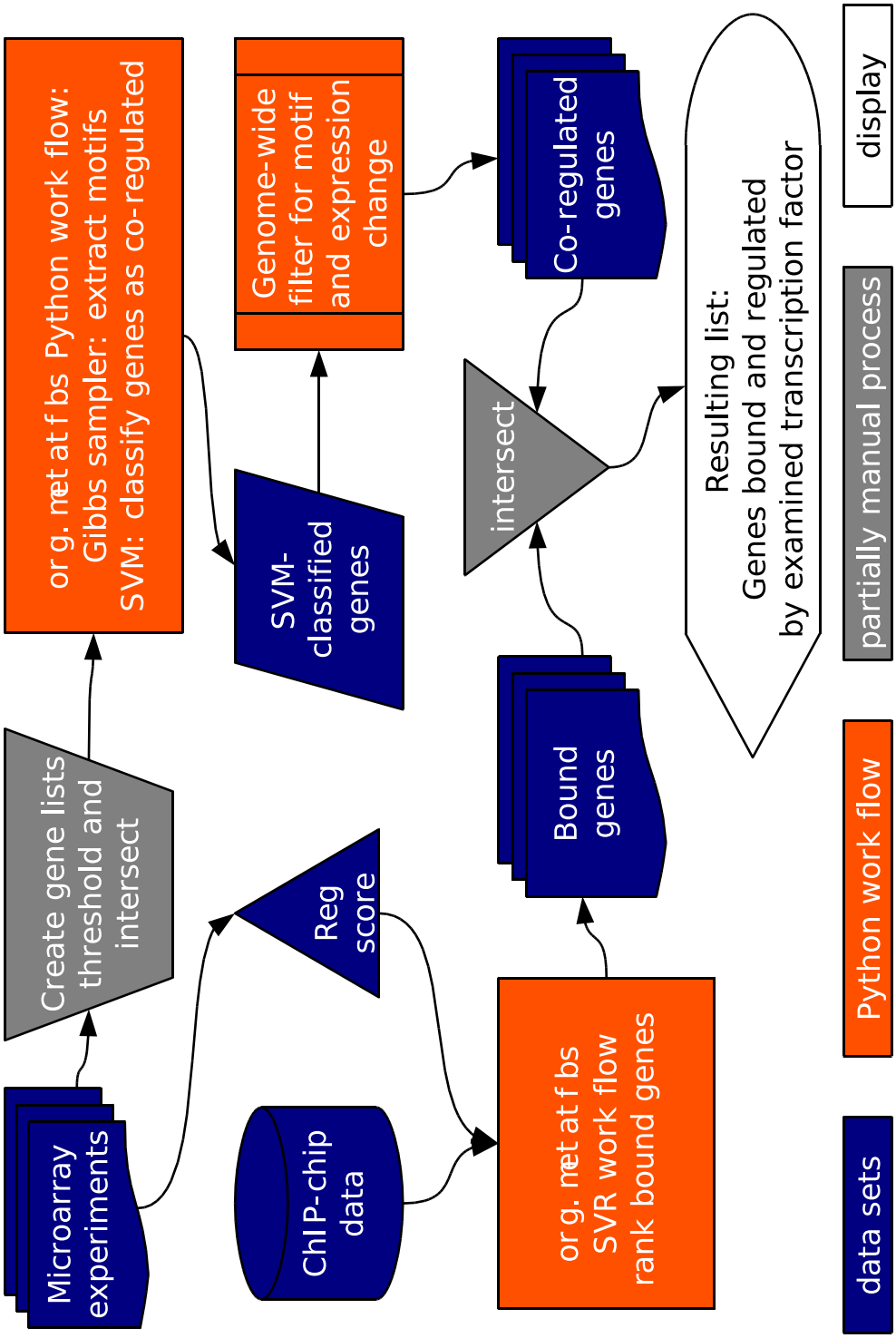
| **Program used** | found by Gibbs sampler | found by INCLUSive MotifSampler |
|---|---|---|
| **Sequence logo** |  |  |
| **Hits per gene** | 2.2 hits | 1.9 hits |
| **Genes contributing** | 20 genes from SVR top 80, 28 genes genome-wide | 18 genes from SVR top 80, 43 genes genome-wide |
| **Similarity in JASPAR, first hit:** JASPAR CORE | 84% MA0112 nuclear receptor | 80% ARR10 *Arabidopsis thaliana* TRP-cluster |
| JASPAR FAM | 67% MF0004 nuclear receptor class family | 67% ETS class family |
| JASPAR PHYLOFACTS | 85% PF0159 | 79% PF0021 |

Figure 4.3: Overall work flow of the use case



# Chapter 5

# Discussion

## 5.1 Related Work

The idea to extract motifs from regulatory sequences with computational means is quite old, and there is a plethora of publications on this topic. The oldest methods use Gibbs sampling, as described in Section 2.3, others use genetic algorithms [69] and again other groups rely on phylogenetic information [70], or a very large number of microarray data [71], and there are still more.

No specific method of those above has become standard among researchers trying to uncover regulatory networks: it is still very hard to do so. The methods developed are definitely not useless, they show promise and yield some good, albeit small-scale results. It is thus no surprise that some groups have tried to integrate several of these approaches to combine the smaller results and arrive at a more comprehensive picture of regulation, *i.e.* a regulatory network [72, 73, 74, 7, 75]. They either use data from different biological assays or integrate several computational approaches.

Like piecing together small clues from a crime scene, they hope to find out who is responsible. We can assume that the regulatory relationships we know and are already familiar with are probably the easiest and most obvious to understand. For example, the *lacZ* operon in *Escherichia coli*, which is the standard example in biology textbooks to introduce regulation of genes [19], is trivially simple. Most other processes, especially the ones that are completely unknown, must have a much higher complexity, especially in eukaryotes, where different cell types co-exist and communicate.

If we are to understand these regulations, it is definitely a good idea to extract information from several independent sources. A machine learning method is then a good choice, because unlike statistical methods, it is able to integrate many different data sources and decide on what is important and what to leave out. Other methods that extract valuable information from raw biological data are a good pre-processing step, because their filtered information suits the machine learning methods better. Noise is always a problem, because the machine will be unable to distinguish between results that are due to noise in the data set and those that are due to clean results. In our case, the prediction would not be very useful if there is a lot of noise in the data set, because we want to limit false positives.

### 5.1.1 Support Vector Machines for Detecting Co-Regulation

Only a few published papers exist where researchers tried to use support vector machines or something similar for this task. The authors of [76] use a single support vector classifier to predict targets of transcription factors by looking only at their expression profiles. They look towards ChIP-on-chip results for confirmation, but according to the results section of their paper, the agreement is rather low.

A trio of researchers, Holloway, Kon, and DeLisi, [77] has already published at least two papers promoting the same method. They integrate several heterogeneous data sources for their SVM training, like I did. Their approach is to use as many data sources as possible, mostly without pre-processing, and putting them into several combined kernels, creating one predictor for every transcription factor they analyze. Since they need positive and negative examples for training and testing (*i.e.* two classes), they try hard to balance the data sets and extract only representatives of the negatives and repeatedly test the same positives, resulting in a strong under-sampling. A balanced data set makes the prediction easier for the classifier.

Linear discriminants are used by [78], and they do a similar form of under-sampling as Holloway *et al.*, but in contrast to that work, they are quite sure the sets they use are in fact true negatives. They do not predict new targets of a transcription factor, but rather filter out false positives from lists of co-regulated genes. For all of their analyses, they only rely on sequence data.

### 5.1.2 Novel Contributions in this Work

One novel idea in this thesis is to only use windows around pre-sampled putative binding sites, instead of the whole promoter region, for the SVM kernel to work on. This results in a great reduction of noise through repetitive and highly mutable regions that do not have to be considered by the kernel, because only interesting, over-represented motifs are included in the results of the Gibbs sampling programs. This explains the continuously high prediction precision we achieve.

Another source of noise are unfiltered microarray experiments that other researchers have used as input vectors. It is obviously a good idea to use the experimental rationale to reduce potential false-positives, because the more replicates we incorporate into the information distilling, the fewer false positives due to chance we can get. Here, false positives are all the genes that appear regulated by a transcription factor due to experimental error or noise.

Conservation information is rarely used in conjunction with microarray data. This is another novelty presented in this work. While some Gibbs sampling approaches use this information [7], this is one of the the first implementation of it into a machine learning kernel, previous attempts for example by Vert *et al.* in 2006 [79] with the spectrum kernel [67].

Recent studies [16] show that among species, binding sites are highly mutable, and variations in genetic expression most probably account for the high difference among the different organisms we know. They calculated that 50% of

all binding sites in *Saccharomyces cerevisiae* are not conserved among its subspecies. We believe it is valuable to identify conserved binding sites, because they are almost guaranteed to be of some kind of importance. We must keep in mind however that not all binding sites will be conserved. We respect this fact in the approach by rewarding the presence of an alignment, but do not discount the absence of one. Using conservation data has greatly improved the performance of the SVM approach, as can be seen in Figure 4.1.

Considering the relative distances, lengths and number of occurrences has been done similarly by [77], and we improve upon their initial ideas.

### 5.1.3 Motif Finding Programs

There is a large number of motif finding programs, and Martin Tompa did a comparative analysis of several of them [60]. Unfortunately, the comparison is lacking both the DIPS algorithm that Tompa's student, Saurabh Sinha, developed after the publication of [60], and the Gibbs sampler from [54], both of which were used in this thesis. We can assume that Sinha incorporated strengths from all the algorithms under scrutiny in the comparative paper and thus created a strong algorithm that performs at least equally to the best one they tested in 2005, namely Weeder [80]. This algorithm was unfortunately unavailable during the six months of working on this thesis. At their referenced web site `http://www.pesolelab.it` one could only see a page from an as of yet unconfigured Mac OS X server (last attempt on May 20, 2007).

Included in Tompa's comparison is the MotifSampler algorithm, which does fairly well in their analysis. Unfortunately, their benchmark data set is taken from the public data base for transcription factors TRANSFAC [81] and does not include any sites found within plants. Their findings are thus of limited significance for this work.

As seen in my previous work [52], both of the packages analyzed there perform very well when presented with FASTA files obtained through good gene lists. This justifies their use also for this approach, although their performance may be limited for the straightforward, thresholded lists as for instance from the heat shock data set. Here especially the negative lists are extremely large and the motif samplers come back with different motifs after every run.

## 5.2 Inferring a Network

The method presented here is not in itself able to infer a regulatory network, in the biologist's sense of the terms. This will, probably for a long time, remain a manual process. Finding transcription factor binding site patterns is an interesting task for computational methods, but as I have shown in Section 5.1, it is not sufficient to reconstruct a network. Most of the regulation processes still not uncovered are probably much more complex than what we already know.

Conducting well-designed biological experiments with an underlying logic is a very promising step towards finding out more about regulation, given the noise and errors that are inherent to these data. The logic can be exploited to exclude

false positives and reduce noise; this will allow us to efficiently distribute the limited resources of working time and project cost on useful biological experiments that are suggested by the results of my work flow. How the experimental logic can benefit the quality of the results has been shown in the SVM results in Table 4.1.

The use case in Section 4.3 is a good example of how the programs developed for this thesis can be applied to real-world data. So to finally infer a network, the binding and co-regulation gene lists have to be intersected. This leads to genes that are both bound by the transcription factor with a very high probability and co-regulated, because they share a motif and have a similar expression pattern. This makes them high-confidence targets that warrant further biological investigations, effectively reducing time-consuming, expensive experiments. For such an investment, it is best to be as certain as possible about which genes to look at, and my approach tries to extract as much information as possible from all the data available.

## 5.3 Comment

Section 5.1 on related work shows that scientists from the biological field and from the computer science field have developed some well-working methods, but generally their potential remains untapped. This is largely due to incompatibility of interfaces among the programs and due to a lack of availability in a simple-to-use manner.

Source code for command line utilities is not a good way to attract users, and mostly the scientific community has itself to blame for this. For a good paper, it is sufficient to apply one's own software to an example and then leave it unmaintained, because only novel discoveries are valued by the community.

To fully harness the capabilities of the developed methods, one needs to find interesting applications and develop extensive programs known as "glue" that integrate these methods. For the work flows presented here, the parts that connect the different third party programs are indeed very complex and involved. Maybe scientists would do each other a big favor to only accept programs for publication that show something of a promising future of being developed further, or at least being in a final state of development before the first release. Unavailable web sites and uncompilable code are just some of the problems I have encountered.

## 5.4 Outlook

### 5.4.1 Future of this implementation

The situation addressed in Section 5.3 should preferably not become true for this project, so I will try to make it available to a general audience of users. This has already been realized for the labs that I worked with: the work flow is available to them for their analyses. We will definitely open it to collaborators

and other interested people, but since it requires a lot of resources even for a single run, there may be some limitations.

My goal is to make it available both as an independent web annotation service and open to queries via SOAP, a web service communication interface I became familiar with during another project [82]. The SOAP interface allows a programmer to easily integrate the web service in her own project and this way create a grid of web resources.

If the issue of computing power can be resolved, a web interface for the Python classes can be easily set up. To make the demands more realistic, that is smaller, the service can be limited to pre-compiled gene lists and maybe also already trained SVM kernels. This could make the time and memory complexity of the work flow significantly smaller.

### 5.4.2 Regulatory Networks

It will become more and more important to look towards the regulation and cell differentiation in the basic biological sciences. Most of all for our own species, we need to learn how expression is controlled, because variations in expression are probably largely responsible for the different species that exist. Many genes are similar among higher eukaryotes, but the organisms clearly are not. This will be one of the greatest challenges in the next few years as we understand more and more how genes are controlled and what networks can be inferred.

# Acknowledgments

I would like to thank my examiners Prof. Dr. Oliver Kohlbacher and Prof. Dr. Detlef Weigel, and my advisors Dr. Jan Lohmann and Dr. Gunnar Rätsch, for the guidance and support they have given me during the creation of this thesis and its manuscript. I am indebted to the Rätsch Lab and the Lohmann Lab, especially Wolfgang Busch for providing the experimental data and advice, Fabio De Bona for help with the Shogun Toolbox, and Korbinian Schneeberger from the Weigel World, for providing data for my work.

For countless discussions over lunch, I would like to thank my fellow students Chris Malisi, Dave Engel, Ewa Willing, Julia Fischer, Marc Röttig, and many more. For help with this treatise, I express my appreciation to Chris Malisi, Michael M. Helten, and Dave Engel.

Finally, I wish to thank my parents, my brother Nico, and my one love, Geri for invaluable moral support.

# Versicherung

**nach § 19 Abs. 6 der Diplomprüfungsordnung Bioinformatik**

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegeben Quellen und Hilfsmittel benutzt zu haben.

Tübingen, den 31. Mai 2007

Sebastian J. Schultheiß

# Appendix

Appendix A contains a list of resources and software that were used to create this thesis and the programs described. The source code is available from the author or on a CD that accompanies the printed version of this thesis, as described in Appendix B.

## Appendix A – Software used for this thesis

Web resources used for this thesis:

- NCBI PubMed, NCBI Entrez, Google Scholar
- TIGR, JGI, TAIR

These are the software packages used to program the scripts for this thesis:

- Python 2.4 with NumPy, MatPlotLib, wxPython, MySQLdb
- MySQL, SVN, Perl, PHP
- Eclipse 3.2 with Pydev, Subclipse, DbEdit
- Shogun with SVM$^{light}$, LIBSVM, SWIG
- Math Works MATLAB 7.4, R 2.4.1
- Gibbs, PhyloGibbs, DIPS, INCLUSive, exMotif, sMotif
- BioPerl::TFBS
- Mac OS X 10.4.9, Ubuntu Linux Dapper Drake

Several software packages were used during of writing this thesis. It is set in Century Schoolbook.

- LyX 1.4.4
- LaTeX, bibTeX, bibit, WIKINDX3
- NeoOffice 2.1, Inkscape 1.4, yEd 2.4.2.2
- Smultron 3.0.4 and TextWrangler 2.2.1
- Adobe Reader 8, Preview, Journler 2.5

# Appendix B – Results and Source Code

The source code from the python work flows and other scripts are available from the author. A CD which accompanies the printed version of this thesis contains this code and additionally all the result files mentioned in the text.

# Bibliography


[1] European Commission (2007) Arabidopsis thaliana - First plant genome sequenced. http://ec.europa.eu/research/quality-of-life/arabidopsis.html *Date: May, 2007*.

[2] Python Software Foundation (2007) Python. http://python.org *Date: May, 2007*.

[3] The Biopython Consortium (2007) Biopython. http://www.biopython.org *Date: May, 2007*.

[4] Boser B. E., Guyon I. M. and Vapnik V. N. (1992) A training algorithm for optimal margin classifiers: *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory, pp.* 144–152. ACM Press, Pittsburgh, Pennsylvania, United States.

[5] Arabidopsis Information Resource (TAIR) (2007) Arabidopsis Genome Annotation TAIR7. http://arabidopsis.org *Date: April, 2007*.

[6] Axmark D., Larsson A. and Widenius M. (2007) MySQL. http://www.mysql.com *Date: May, 2007*.

[7] Siddharthan R., Siggia E. D. and Nimwegen E. (2005) PhyloGibbs: a Gibbs sampling motif finder that incorporates phylogeny. *PLoS Comput Biol* **1**(7), 67.

[8] Thijs G., Lescot M., Marchal K., Rombauts S. *et al.* (2001) A higher-order background model improves the detection of promoter regulatory elements by Gibbs sampling. *Bioinformatics* **17**(12), 1113–22.

[9] Sinha S. (2006) On counting position weight matrix matches in a sequence, with application to discriminative motif finding. *Bioinformatics* **22**(14), e454–63.

[10] Schwartz S., Kent J. W., Smit A., Zhang Z. *et al.* (2003) Human-mouse alignments with BLASTZ. *Genome Res* **13**(1), 103–107.

[11] Altschul S. F., Madden T. L., Schäffer A. A., Zhang J. *et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* **25**(17), 3389–3402.

[12] Huson D. (2007) Software programs suitable for whole genome alignment. Personal communication, unpublished work. http://www-ab.informatik.uni-tuebingen.de *Date: May, 2007*.

[13] Blanchette M., Kent J. W., Riemer C., Elnitski L. *et al.* (2004) Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res* **14**(4), 708–715.

[14] Tuskan G. A., Difazio S., Jansson S., Bohlmann J. *et al.* (2006) The genome of black cottonwood, Populus trichocarpa (Torr. & Gray). *Science* **313**(5793), 1596–1604.

[15] Scherf M. (2007) Bioinformatics of Expression Regulation. *Date: March, 2007*.

[16] Doniger S. W. and Fay J. C. (2007) Frequent Gain and Loss of Functional Transcription Factor Binding Sites. *PLoS Computational Biology* **3**(5), 932–942.

[17] Nover L. and Scharf K. D. (1997) Heat stress proteins and transcription factors. *Cell Mol Life Sci* **53**(1), 80–103.

[18] Suzuki M., Wang H. H.-Y. and McCarty D. R. (2007) Repression of the LEAFY COTYLEDON 1/B3 regulatory network in plant embryo development by VP1/ABSCISIC ACID INSENSITIVE 3-LIKE B3 genes. *Plant Physiol* **143**(2), 902–911.

[19] Alberts B., Johnson A., Lewis J., Raff M., Roberts K. and Walter P. (2002) Molecular Biology of the Cell. Garland Science, New York, New York, United States.

[20] Suzuki M., Ketterling M. G., Li Q.-B. and McCarty D. R. (2003) Viviparous1 alters global gene expression patterns through regulation of abscisic acid signaling. *Plant Physiol* **132**(3), 1664–1677.

[21] Lawrence C. J., Dong Q., Polacco M. L., Seigfried T. E. and Brendel V. (2004) MaizeGDB, the community database for maize genetics and genomics.. *Nucleic Acids Res* **32**(Database issue), D393–D397.

[22] Lohmann J. U., Hong R. L., Hobe M., Busch M. A. *et al.* (2001) A molecular link between stem cell regulation and floral patterning in Arabidopsis. *Cell* **105**(6), 793–803.

[23] Leibfried A., To J. P. C., Busch W., Stehling S. *et al.* (2005) WUSCHEL controls meristem function by direct regulation of cytokinin-inducible response regulators. *Nature* **438**, 1172–1175.

[24] Affymetrix (2006) Affymetrix Chromatin Immunoprecipitation Assay Protocol. *Affymetrix, Inc.*

[25] Meyerowitz E. M. and Somerville C. R. (1994) Arabidopsis. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York.

[26] Arabidopsis Information Resource (TAIR) (2007) About Arabidopsis. `http://www.arabidopsis.org/portals/education/aboutarabidopsis.jsp`it Date: *April, 2007*.

[27] Arabidopsis Genome Initiative (2000) Analysis of the genome sequence of the flowering plant Arabidopsis thaliana. *Nature* **408**(6814), 796–815.

[28] Schneider T. D. and Stephens R. M. (1990) Sequence logos: a new way to display consensus sequences. *Nucleic Acids Res* **18**(20), 6097–100.

[29] Saigo H. and Tsuda K. (2007) Mining Itemsets of Transcription Factor Binding Site Motifs. (Personal communication, unpublished work). *Date: April, 2007*.

[30] Pham T. H., Satou K. and Ho T. B. (2004) Mining yeast transcriptional regulatory modules from factor DNA-binding sites and gene expression data. *Genome Inform* **15**(2), 287–295.

[31] Matthews B. W., Ohlendorf D. H., Anderson W. F. and Takeda Y. (1982) Structure of the DNA-binding region of lac repressor inferred from its homology with cro repressor. *Proc Natl Acad Sci U S A* **79**(5), 1428–1432.

[32] Sandelin A., Alkema W., Engstrom P., Wasserman W. W. and Lenhard B. (2004) JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Res* **32**(Database issue), D91–4.

[33] Derose-Wilson and Gaut (2007) Transcription-related mutations and GC content drive variation in nucleotide substitution rates across the genomes of Arabidopsis thaliana and Arabidopsis lyrata. *BMC Evol Biol* **7**(1), 66.

[34] Ouyang S., Zhu W., Hamilton J., Lin H. *et al.* (2007) The TIGR Rice Genome Annotation Resource: improvements and new features.. *Nucleic Acids Res* **35**(Database issue), D883–D887.

[35] Schneeberger K. and Ossowski S. (2007) Sequence of Arabidopsis lyrata and alignment with Arabidopsis thaliana. (Personal communication, unpublished work). *Date: March, 2007*.

[36] Nasrallah M. E., Yogeeswaran K., Snyder S. and Nasrallah J. B. (2000) Arabidopsis species hybrids in the study of species differences and evolution of amphiploidy in plants. *Plant Physiol* **124**(4), 1605–1614.

[37] Affymetrix (2004) Data Sheet of the GeneChip Arabidopsis ATH1 Genome Array. *Affymetrix, Inc.*

[38] Redman J. C., Haas B. J., Tanimoto G. and Town C. D. (2004) Development and evaluation of an Arabidopsis whole genome Affymetrix probe array. *Plant J* **38**(3), 545–561.

[39] Busch W. and Lohmann J. U. (2007) Profiling a plant: expression analysis in Arabidopsis. *Curr Opin Plant Biol* **10**(2), 136–141.

[40] Metz B., Kersten G. F. A., Hoogerhout P., Brugghe H. F. *et al.* (2004) Identification of formaldehyde-induced modifications in proteins: reactions with model peptides. *J Biol Chem* **279**(8), 6235–6243.

[41] Voet J. G. and Voet D. (2004) Biochemistry. John Wiley & Sons, Inc., Hoboken, New Jersey, USA.

[42] Schoelkopf B. and Smola A. J. (2001) Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA.

[43] Rätsch G. and Sonnenburg S. Accurate Splice Site Prediction for Caenorhabditis elegans: *MIT Press Series on Computational Molecular Biology, pp.* 277–298. MIT Press.

[44] Rätsch G., Sonnenburg S. and Schäfer C. (2006) Learning interpretable SVMs for biological sequence classification. *BMC Bioinformatics* **7 Suppl 1**, 9.

[45] Davis J. and Goadrich M. (2006) The Relationship Between Precision-Recall and ROC Curves: *Proceedings of the 23rd International Conference on Machine Learning, pp.* 233–240, Pittsburg, Pennsylvania, United States.

[46] Zweig M. H. and Campbell G. (1993) Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine. *Clin Chem* **39**(4), 561–577.

[47] Vapnik V. N. (1995) The nature of statistical learning theory. Springer-Verlag New York, Inc., New York, NY, USA.

[48] Cortes C. and Vapnik V. N. (1995) Support-vector networks. *Machine Learning* **20**(3), 273–297.

[49] Aizerman M. A., Braverman E. A. and Rozonoer L. (1964) Theoretical foundations of the potential function method in pattern recognition learning: *Automation and Remote Control, pp.* 821–837. Nauka/Interperiodica, Moscow, Russia.

[50] Smola A. J. (1996) Regression estimation with support vector learning machines. *Thesis,* Technische Universität München, Munich, Germany.

[51] Sonnenburg S., Rätsch G. and Rieck K. (2006) Large Scale Learning with String Kernels. *In* Bottou L., Chapelle O., DeCoste D. and Weston J. *(Eds.)*: *Large Scale Kernel Machines*, pp. 1–38. The MiT Press, Cambridge, MA, USA.

[52] Schultheiss S. J. (2005) Transcriptional Regulation of Stem Cells in Arabidopsis thaliana. *Thesis,* Eberhard Karls University, Tübingen, Germany.

[53] Lawrence C. E., Altschul S. F., Boguski M. S., Liu J. S., Neuwald A. F. and Wootton J. C. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* **262**(5131), 208–14.

[54] Thompson W., Rouchka E. C. and Lawrence C. E. (2003) Gibbs Recursive Sampler: finding transcription factor binding sites. *Nucleic Acids Res* **31**(13), 3580–5.

[55] Thijs G., Moreau Y., De Smet F., Mathys J. *et al.* (2002) INCLUSive: integrated clustering, upstream sequence retrieval and motif sampling. *Bioinformatics* **18**(2), 331–2.

[56] Sinha S. and Tompa M. (2000) A statistical method for finding transcription factor binding sites. *Proc Int Conf Intell Syst Mol Biol* **8**, 344–354.

[57] Reddy C. K., Weng Y.-C. and Chiang H.-D. (2006) Refining motifs by improving information content scores using neighborhood profile search. *Algorithms Mol Biol* **1**, 23.

[58] Hughes J. D., Estep P. W., Tavazoie S. and Church G. M. (2000) Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *J Mol Biol* **296**, 1205–1214.

[59] Brynildsen M. P., Tran L. M. and Liao J. C. (2006) A Gibbs sampler for the identification of gene expression and network connectivity consistency. *Bioinformatics* **22**(24), 3040–3046.

[60] Tompa M., Li N., Bailey T. L., Church G. M. *et al.* (2005) Assessing computational tools for the discovery of transcription factor binding sites. *Nat Biotechnol* **23**(1), 137–44.

[61] Cheadle C., Vawter M. P., Freed W. J. and Becker K. G. (2003) Analysis of microarray data using Z score transformation. *J Mol Diagn* **5**(2), 73–81.

[62] Wu Z., Irizarry R. A., Gentleman R., Murillo F. M. and Spencer F. (2004) A model based background adjustment for oligonucleotide expression arrays. *John Hopkins University, Dept. of Biostatistics Working Papers* (Working Paper 1), 2–26.

[63] Lemon W. J., Liyanarachchi S. and You M. (2003) A high performance test of differential gene expression for oligonucleotide arrays. *Genome Biol* **4**(10), 67.

[64] Sonnenburg S., Rätsch G., Schäfer C. and Schölkopf B. (2006) Large Scale Multiple Kernel Learning. *Journal of Machine Learning Research* **7**(Jul), 1531–1565.

[65] Chang C.-C. and Lin C.-J. (2007) LIBSVM: a library for support vector machines. `http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html` *Date: May, 2007*.

[66] Joachims T. (1999) Making large-Scale SVM Learning Practical. *In* Schölkopf B., Burges C. and Smola A. J. *(Eds.)*: *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, USA.

[67] Leslie C., Eskin E. and Noble W. S. (2002) The spectrum kernel: a string kernel for SVM protein classification. *Pac Symp Biocomput* , 564–575.

[68] Hosoda K., Imamura A., Katoh E., Hatta T. *et al.* (2002) Molecular structure of the GARP family of plant Myb-related DNA binding motifs of the Arabidopsis response regulators. *Plant Cell* **14**, 2015–2029.

[69] Wei Z. and Jensen S. T. (2006) GAME: detecting cis-regulatory elements using a genetic algorithm. *Bioinformatics* **22**(13), 1577–84.

[70] Lenhard B., Sandelin A., Mendoza L., Engstrom P., Jareborg N. and Wasserman W. W. (2003) Identification of conserved regulatory elements by comparative genome analysis. *J Biol* **2**(2), 13.

[71] Brilli M., Fani R. and Lió P. (2007) MotifScorer: using a compendium of microarrays to identify regulatory motifs. *Bioinformatics* **23**, 493–495.

[72] Xiong M., Li J. and Fang X. (2004) Identification of genetic networks. *Genetics* **166**(2), 1037–52.

[73] Sun J., Tuncay K., Haidar A. A., Ensman L. *et al.* (2007) Transcriptional regulatory network discovery via multiple method integration: application to E. coli K12. *Algorithms Mol Biol* **2**(1), 2.

[74] Steffens N. O., Galuschka C., Schindler M., Bülow L. and Hehl R. (2005) AthaMap web tools for database-assisted identification of combinatorial cis-regulatory elements and the display of highly conserved transcription factor binding sites in Arabidopsis thaliana. *Nucleic Acids Res* **33**(Web Server issue), W397–W402.

[75] Palaniswamy S. K., James S., Sun H., Lamb R. S., Davuluri R. V. and Grotewold E. (2006) AGRIS and AtRegNet: A platform to link cis-regulatory elements and transcription factors into regulatory networks. *Plant Physiol* **140**, 818–829.

[76] Qian J., Lin J., Luscombe N. M., Yu H. and Gerstein M. (2003) Prediction of regulatory networks: genome-wide identification of transcription factor targets from gene expression data. *Bioinformatics* **19**, 1917–1926.

[77] Holloway D. T., Kon M. and DeLisi C. (2005) Integrating genomic data to predict transcription factor binding. *Genome Inform* **16**, 83–94.

[78] Simonis N., Wodak S. J., Cohen G. N. and Helden J. (2004) Combining pattern discovery and discriminant analysis to predict gene co-regulation. *Bioinformatics* **20**, 2370–2379.

[79] Vert J. P., Thurman R. and Noble W. S. (2005) Kernels for gene regulatory regions: *Neural Information Processing Systems, pp.* 1401–1408. MIT Press.

[80] Pavesi G., Mereghetti P., Mauri G. and Pesole G. (2004) Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic Acids Res* **32**, W199–W203.

[81] Matys V., Fricke E., Geffers R., Gossling E. *et al.* (2003) TRANSFAC: transcriptional regulation, from patterns to profiles. *Nucleic Acids Res* **31**(1), 374–8.

[82] Schultheiss S. J. and Malisi C. U. (2006) CoMPAS Pro: Comprehensive Meta Prediction and Annotation Services for Proteins. `http://compas-pro.sourceforge.net` *Date: December, 2007*.