University of California
Los Angeles

Contrast Learning on ChIP-Seq Data of Transcription Factors

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Yuju Lee

2014

© Copyright by
Yuju Lee
2014

Abstract of the Thesis

Contrast Learning on ChIP-Seq Data of Transcription Factors

by

Yuju Lee

Master of Science in Statistics

University of California, Los Angeles, 2014

Professor Qing Zhou, Chair

In this study, we analyzed the TF ChIP-Seq data of 105 (i.e., 15 choose 2) pairs. Each pair is based on two TF and three binding-dependent (BD) sequence datasets. The BD were generated from the two TF ChIP-Seq datasets in each pair. That is, the three scenario datasets are containing TFBS sequences of type 1, 2 or both (i.e., 1 and 2) TF.

The objective is to identify motif 1, 2 or even both (i.e., interactive motifs) by contrasting two of the three BD datasets at a time by using the contrast-motif-finder (CMF) algorithm. Each of the CMF's output not only provides estimated consensus motifs based on its full name PWM but also provides likelihood ratios (LRs) as a measure of the enrichment of an identified motif. Using this idea, we construct a dataset where the first column lists the locations of identified enriched motif in the genome, column 2 to n+1 contains the estimated consensus motifs and the last column shows a binary (i.e., 0/1) of which set it is from and n is the number of consensus motifs.

Once these datasets are obtained, we use statistical model such as logistics regression, support vector machine (SVM) and classification tree models to determine their performance (i.e., error rates) and selection power. We have shown

that the SVM Radial kernel seems to have the best performance when using all the motifs in the dataset whereas classification tree selects the fewest motifs in almost every analyzed datasets but at the same time, the error rates and selection power do not drop as much. As a result, we believe the classification tree model is a better model since it not only provides a competitive predictive power with simpler models but also takes far less computational time than the other two models.

The thesis of Yuju Lee is approved.

Rick Paik Schoenberg

Ying Nian Wu

Qing Zhou, Committee Chair

University of California, Los Angeles

2014

To my parents

for their permanent love, and to my friends and teachers

who have given me precious memories and tremendous encouragement.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

One of the most popular and exciting fields to emerge recently in bioinformatics is the development of algorithms that can accurately identify a transcription factor (TF) motif, based on genome-wide ChIP data. Contrast-motif-finder (CMF) (3) and predictive modeling approach (6) are two of such algorithms for finding TF motifs. In both papers, Mason and Zhou have shown that their approaches have a better accuracy rate than other well-known methods for identifying TF motifs.

In this study, we have developed a novel procedure for identifying TF motifs by applying and merging their two ideas together, combining contrast-motif-finder (CMF) techniques with predictive modeling approaches. Based on the CMF's design, we can strategically generate TF pairs (i.e, Oct4/Sox2) and binding-dependent (BD) sets so that we know what TF to expect in each CMF's output and sampling dataset for any two contrast inputs.

Since some predictive models can provide a way to reduce a dataset's dimensionality by selecting important variables/motifs while maintaining its predictions or creating even better ones compared to the full model. We hope to go one step further in determining whether selected motifs in reduced models have expressions similar to their consensus motifs.

Three types of well-known predictive modelings are used: logistics regression, support vector machine (SVM) and classification tree. The goal is to discover a model that would reduce its data dimensionality by the greatest amount (i.e., fewest selection variables) and produce the best performance (i.e., error rates)

based on the summarized results from all 38 analyzed pairs.

1.1 Biological Concept

All living organisms are composed of cells. Each cell contains many chromosome. Cells are the basic unit of life, which contain of deoxyribonucleic acid (DNA) and proteins. DNA is made of two long complimentary nucleotide strands of four nucleic acids twisted around each other in opposite directions forming a double helix. These nucleic acids are Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). They are joined by a sugar and phosphate backbone where hydrogen bond pair Adenine with Thymine (A-T) and Cytosine with Guanine (C-G) from each side of the helix. This structure is called a base pair (bp) or bps in plural.

A gene is a basic unit of heredity in a living organism and resides on a stretch of DNA sequence that codes protein and has a function in the organism. Genes can hold information to build and maintain an organism's cells and pass genetic traits and characteristics. Gene expression is comprised of two major stages. First, ribonucleic acid (RNA) is produced from DNA and then a protein is produced from RNA. This stage is called transcription, the process of making RNA from DNA. The second stage is called translation, where RNA produced by transcription is read by the ribosome to produce a particular amino acid chain which is folded into an active protein. This process of making proteins from DNA is often called the central dogma in molecular biology.

The genome of a living organism contains both genes and non-coding sequences of the DNA. Genes are composed of sequences of DNA in various locations in a genome. The non-coding portion of sequence usually lie in the upstream regulatory regions or before transcription start site (TSS). A regulatory region is the region of the DNA that regulates the transcription of the gene. It can be a positive/negative regulation in which transcription factors (TFs) bind/block the

promoter and enhancer regions and therefore turn on/off its expression. The promoter region is usually at the beginning of a gene and is the region of DNA where the RNA polymerase binds to the DNA to begin to transcribe a gene. The enhancer region is the region of DNA where certain TFs may bind to increase the transcription process.

TF expressions usually share similar patterns with some nucleotide position(s) different. These patterns are often called motifs. A motif is usually short, consisting of 5 to 14 bps, which makes it hard to detect such regions experimentally and computationally. A change at a variable nucleotide position in a TFBS may only have a small effect on the binding affinity whereas a change at an invariant position may have a strong effect. Identifying and characterizing these TFBSs may provide a better understanding of the function of a TF. A recent developed biology experiment method, called ChIP-Seq, is able to locate genome-wide transcription factor binding sites (TFBSs) at the resolution within less than 100 of bps. This is a big step in biology since scientists now can develop motif-finding algorithms or models based on genome-wide ChIP-Seq data to understand gene transcription mechanisms.

1.2 ChIP-Seq Dataset

In this study, we focus on mouse Embryonic Stem Cells (ESC) ChIP-Seq datasets. 13 TFs and 2 transcription regulator datasets are available in the GEO database under ID number GSE11431 [1]. The 13 TF datasets are Nanog, Oct4, STAT3, Smad1, Sox2, Zfx, c-Myc, n-Myc, Klf4, Esrrb, Tcfcp2l1, E2f1, and CTCF. The 2 transcription regulators are p300 and Suz12.

ChIP-Seq is a technology which combines ChIP-chip technology with massively parallel DNA to map the locations of sequence specific TF and transcription regulators. Each ChIP-Seq dataset is composed of two major procedures: 1)

The raw images of each TFBSs have been processed using the Soxlexa Pipeline, and then mapped to the reference genome using Eland software with maximal two mismatches. Each dataset contains chromosome number, peak location and count of reads (signal intensity) of a TFBS. A total of 20 chromosomes are recorded since a mouse has 19 plus an X chromosomes. Each peak location is translated into a sequence region in its genomic coordinates and often refer it to TFBS (1). Each genomic coordinate contains two numbers (i.e., beginning/ending) of a TFBS region so that it is easy to convert them to a nucleotide format.

1.3 Structure of this thesis

This study intends to discover the best performing predictive selection model giving the lowest dimensionality at the same time that selects the greatest number of motifs that accurately match the consensus motif. Chapter 2 gives a list of procedures for constructing contrasted datasets, Chapter 3 discusses the three types of predictive models, Chapter 4 offers the summarized results based on analyzed pairs, and Chapter 5 demonstrates how we analyzed a pair (i.e., Oct4/Sox2) in detail. Finally, Chapter 6 provides conclusions and suggestions.

CHAPTER 2

Methods

In this chapter, our goal is to construct the three sampling datasets for each pair from their TF ChIP-Seq datasets. We summarize our procedures in the following:

1 Expanding TFBSs Window Size,

2 Creating three Binding-Dependent (BD) sets,

3 Converting all TFBSs in BD Sets to DNA Nucleotide Sequences.

4 Once the number of TFBSs was discovered in each BD set from step 3, our analysis only included pairs > 300 TFBSs in all three BD sets. Then, we applied the following steps to those pairs.

5 Contrasting BD sets by CMF,

6 Constructing Predictive Datasets based on the Outputs from step 5,

7 Applying the Sampling Approach to the Resulted Datasets from step 6.

For each pair, we performed the 7 steps describe above, which is going to be discussed greater detail in the following section.

2.1 Expanding TFBS Window Size

For each TF data, we created an expanded set based on the first column of a ChIP-Seq data (See Section 1.2). The first column of each row corresponded to

a binding sequence (BS). Each sequence provided its chromosome number and genomic coordinate in a mouse genome. We parsed and stored them into a $X_{n\times 3}$ matrix where n and 3 are the number of TFBSs in the data and the chromosome numbers, the starting and ending locations in each row, respectively.

Then, we subtracted (starting) and added (ending) 100 bps in column two and three. We searched every combination of two rows/sequences and merged them if they were less than 200 bps apart from either side of a TFBS sequence in the same chromosome. We stored the minimum and maximum of their genomic coordinates into the $X_{n\times 3}$ matrix and removed the two merged sequences. We stopped this process until no more such cases existed. We denoted the final matrix as an expanded set and treated it as our raw data.

2.2 Creating Binding-Dependent (BD) Sets

For each pair, we proposed three subset datasets based on the two TF sets discussed in Section 2.1, that is, for a given TF pair, S_1 and S_2, we defined the three datasets as TFBSs contained only type 1, 2, or both (i.e., 1 and 2) TF in the resulted sets. Each TFBS in the three sets must be at least 200 bps apart from either side of another sequence. To create an interactive set (i.e., 1 and 2), we iteratively merged two sequences from S_1 and S_2 if they were less than 200 bps apart from either side of another sequence and stopped once no more such cases existed. Then, we recorded the minimum and maximum of their merged genomic coordinates and stored them into S_{i_1}.

As a note, it is possible that two or more expanded TFBS sequences are merged. This is the reason we made expanded sets first so that we know it was an interactive sequence when two sequences merged. Obviously, the number of TFBS in S_1 and S_2 were expected to be less after creating S_{i_1}. We denoted them as S_{a_1}, and S_{b_1}, respectively. We call the three subsets as "binding-dependent

(BD)" datasets.

2.3 Converting TFBSs to DNA Nucleotide Sequences

Once the three BD sets were created (See Section 2.2) for each pair, we converted each of their genomic coordinates (TFBSs) to nucleotide sequences using an integrated software called Cisgenome. Specifically, a program name genome_getmaskedseq_c was used on a Mac OS machine (2) to convert each BD sequence in S_{a_1}, S_{b_1}, and S_{i_1} to nucleotide sequences. The program automatically filtered out sequences that were out of a genome range. We believe that the automatically filtered out sequences were difficult to avoid and the chances of detecting them were slim so we did not think it was a big issue to filter them out.

Each nucleotide sequence consisted of a combination of lower and upper case letters of $\{A, a, C, c, G, g, T, t\}$. A position with lower case letter (i.e., a, c, t, g) in a sequence indicated it is masked out so we can safely ignore it. A sequence was also filtered out if it contained less than 20 bps of upper case letters for computation reasons. After these filtering steps, we denoted the three BD sets that composed of nucleotide sequences as S_a, S_b, and S_i.

2.4 Filtering Out Pairs by Number of TFBSs

Once S_a, S_b, and S_i were constructed (See Section 2.3) for each pair, the number of TFBSs in them may be a concern for certain pairs when contrasting these BD sets. In order to ensure there were enough TFBSs in each one of them, we eliminated pairs that contained less than 300 TFBSs in one of the S_a, S_b, or S_i sets and 37 pairs were eliminated.

2.5 Contrasting BD Sets

For each selected pair (See Section 2.4), we contrasted all combinations of two among the three BD sets using CMF. CMF was specifically designed to discriminate between the two sets of TFBSs and provide an estimation of PWMs and consensus motifs. CMF not only corrected for false positives but also masked out dominant motifs signals present in both datasets, allowing weaker signals to be detected (3).

In specific, we contrasted S_a with S_b, S_a with S_i, and S_b with S_i. For each selected pair (See Section 2.4), we denoted these outputs as the three predictive datasets: S_{ab}, S_{ai}, and S_{bi}. Based on the design of BD sets and CMF, we are expected CMF to detect consensus motifs from both a and b in S_{ab}, and from a and b in S_{bi} and S_{ai} accordingly. The reason is that when a contrasted with i, the portion of a in i should be canceled out with a since they should have similar patterns and the b portion in i should be enriched. The same reasoning applies to S_{bi} scenario. However, S_{ab} was a special case where the resulting motifs should contain both a and b motifs. Since nothing can be contrasted from both sets and we did not know exactly how to interpret the results from this type of contrast set, we decided to focus on the results from S_{ai} and S_{bi}. As a note, we filtered out pairs split out errors when applying to CMF. This happened in 3 (i.e., Oct4Esrrb, Stat3K1f4, and cMycTcfcp211) pairs and implied and a total of 64 (=105-37-3) of them were considered for further study. We called them the "selected pairs".

Moreover, the BD data were not compatible with CMF's input, a series of data transformations were needed. In each CMF implementation, we selected the contrast mode and the default parameters. That is, We set the length of a motif seed, the number of mismatch in the seed, FDR level, the number of motif seeds, enrichment mode, lower and upper bound on length of motifs, and c/g content filter to 7 (=w), 2 (=m), .667 (=F), 10 (=t), 2 (=d), 5 (=l), 20 (=u), and 0 (=c),

respectively. A brief summary of the CMF algorithm is outlined in Appendix A and for more information please refer to (3).

In each CMF output, z and enrichment scores were provided for each motif seed. A z and enrichment scores were a measure of significant sequence patterns and enriched sequences in both contrasted sets, respectively. Each dataset we usually found $15 - 25$ consensus motifs. For each motif seed, a list of enriched motifs, their locations and their sequence length were recorded for both contrasted sets.

Enriched motifs were based on an initial motif seed, which consisted of the same sequence length and at most two ($m = 2$) mismatches with each other but the mismatch positions could be different from one to the other. In addition, the length of enriched motifs could be different from the motif seed depending on the flanking positions. This implies that all the enriched motifs could be shorter or longer than a given initial seed. Furthermore, once all the enriched motifs were obtained, a PWM summarized the most probable nucleotide of all the enriched motifs, which we called an "estimated consensus motif."

Moreover, CMF provides likelihood ratio (LR) scores as a measure of detecting subtle regulatory signals within each enriched motif which is a crucial point for making the predictive datasets, which is discussed in the next section.

2.6 Constructing Predictive Modeling Dataset

After getting the predictive datasets for all selected pairs (See Section 2.5), we constructed predictive datasets based on CMF's output by using the estimated consensus motifs and their enriched motifs as LR scores for both contrasted sites.

We believe there may be a few potential obstacles in creating predictive datasets. First, CMF sometimes reported the same consensus motif more one than once, this is a good sign, but we believe they did not give much more additional informa-

tion by including them other than to reduce a dataset's dimensionality. In order to overcome this concern, we chose the one with the highest absolute enrichment score. This was because the enrichment scores of a consensus motif were positive for the first set and negative for the second set.

Moreover, we removed a consensus motif that did not provide enriched motifs from set 1 and/or 2. This was a problem in creating the predictive dataset since its response variable indicated which set an enriched motif belongs to. Furthermore, we chose the maximum LR if two or more LRs (i.e., enriched motifs) were recorded in the same TFBS sequence. This was due to the fact that two enriched motifs were identified in the same enriched sequence but at a different starting or ending point. We denoted the number of selected/filtered consensus motifs to be considered as n.

After filtering consensus motif(s), we extracted the locations and LRs of enriched motif for each consensus motifs individually into a $X_{d\times 2}$ matrix where d was the number of enriched motif locations. The first and second column each contained enriched locations and LRs of those enriched locations. n such sets were created for each CMF's output.

Then, we merged them to form a predictive dataset. We denoted it as $X_{(l+r+n)\times k}$. For each data construction, the first and second column contain enrich motif locations from both TF sets and a binary (-1/1). 1 meant the motif was from the first set and -1 otherwise. Then, we merged each of n $X_{d\times 2}$ matrices by their sequence locations one at a time. If a TFBS location did not have an enriched motif score (LR) for that particular consensus motif, a zero was placed otherwise we placed the log of its LR score instead. At the end, we removed an entire row if a TFBS sequence did not have a LR score among all n consensus motifs. The number of rows, k, remaining at the end implied the number of enriched motifs that were used at least once among n estimated consensus motifs.

Finally, three $X_{(l+r+n)\times k}$ predictive datasets were constructed for each of the

selected pairs based on S_{ab}, S_{ai}, and S_{bi} and the number of enriched motifs in a, b, or i in the three predictive datasets were different. For example, the number of enriched sequences of a in S_{ab} and S_{ai} are different. We denoted the three resulting datasets as predictive datasets (X_{ab_p}, X_{ai_p}, and X_{bi_p}). In each of the predictive datasets, the first column consisted of enriched motif locations from both sets in the genome, column three thru n consisted of LRs from n selected consensus motifs and the second column is composed of a binary (-1/1) response indicator showing whether a TFBS was from both sets. The -1 and 1 in the response variable correspond to the enriched motif in the first and second set. For example, -1 and 1 implied a sequence from a and b in S_{ab} respectively.

2.7 Applying Sampling Approach to Predictive Dataset

After obtaining predictive datasets in Section 2.6, we realized there were two potential problems: 1) the number of -1's in the responses variable of a dataset may dominate the number of 1's or vice versa. This may cause a problem when we build a predictive model, such as the classification tree model, since it was unlikely to split into two groups; and 2) it would be computationally expensive if a predictive dataset contains too many observations especially when applying the selection methods.

Hence, we proposed taking a sampling approach to fix these potential problems. For a predictive dataset, we first subdivided it into two groups according to their class 1 (pos) and -1 (neg) as S_{pos} and S_{neg} and set the ratio of their size to be 0.5 (=r). This controlled the number of enriched motifs in the larger subset to be no more than $(1 + r)$ times of the smaller subset. We also set the maximum data size to be 3000 (=s) for each subset. That is, the number of enriched motifs in both subsets combined is $2 \times s$. Then, the steps are as follows:

1. If the number of enriched motifs in S_{pos} and $S_{neg} > s$, then sample each

subset s times.

2. If the number of enriched motifs in $S_{pos} > s$ and $< S_{neg}$ and, if the ratio between the number of enriched motifs in S_{neg} and $S_{pos} < r$, and

   If the number of enriched motifs in S_{neg} multiplied by $(1 + (1 - r)) < s$, **a)** then randomly sample 1 from S_{neg} $(1+(1-r))$ times, **b)** otherwise sample -1 from S_{neg} s times.

3. If the number of enriched motifs in $S_{neg} > s$ and $< S_{pos}$ and, if the ratio between the number of enriched motifs in S_{neg} and $S_{pos} < r$, and

   If the number of enriched motifs in S_{pos} multiplied by $(1 + (1 - r)) < s$, **a)** then randomly sample 1 from S_{pos} $(1+(1-r))$ times, **b)** otherwise sample -1 from S_{pos} s times.

4. If the number of enriched motifs in S_{pos} and $S_{neg} < s$,

   a) If the number of enriched motifs in $S_{pos} > S_{neg}$, and if the number of enriched motifs in S_{neg} is less or equal to r times of S_{pos}, and if the number of enriched motifs in S_{neg} multiplied by $(1 + (1 - r))$ times is less than s, then randomly sample 1 $(1 + (1 - r))$ times from S_{neg}.

   b) If the number of enriched motifs in $S_{neg} > S_{pos}$, and if the number of enriched motifs in S_{pos} is less or equal to r times of S_{neg}, and if the number of enriched motifs in S_{pos} multiplied by $(1 + (1 - r))$ times is less than s, then randomly sample 1 $(1 + (1 - r))$ times from S_{pos}.

We then combined the two sample subsets and referred them as the sampling dataset. We repeated the same process for all the predictive datasets. We denoted the resulting output as sampling datasets (X_{ab_s}, X_{ai_s}, and X_{bi_s}). We note that if S_{pos} and S_{neg} did not meet the conditions of $1 - 4$, we used the same S_{pos} and S_{neg} as before applying the sampling procedures. Finally, we can apply these "sampling datasets" to a number of different predictive models.

CHAPTER 3

Predictive Modelings

In this chapter, we introduce the three types of predictive models, their variable selection algorithms (if any) and the R (a statistical software language) packages used for this study. The three types of models are the logistics regression (logit), support vector machine (SVM), and classification tree (tree). The variable selection algorithms are the stepwise logistics regression, Scad SVM, and classification tree.

3.1 Logistic Regression

A logistic regression is a transformation of an ordinary least squares (OLS) regression. This transformation is also referred to as a logit function. OLS and logit regressions can be used when a dependent variable is a categorical and independents are of any type. But, the difference is that logit's predictions are between 0 and 1 whereas the OLS may predict values above 1 or below 0. Moreover, logit enables us to estimate the odds of a certain event occurring in terms of probabilities.

We used the glm (generalize linear model) function in R to implement each logit model. We specify logit as the parameter in the glm function and used defaults for other parameters. Since logit regressions are widely used, we only provide an overview of the algorithm in Appendix B.

3.1.1 Variable Selection by Stepwise Logistic Regression

A software package was used to build stepwise regression models. More precisely, we used the stepAIC function in the MASS package in R (10). This function used Akaike information criterion (AIC) as a penalty function when selecting motifs or variables. AIC's definition is: $AIC = 2k - 2logL$, where k is the number of parameters in the model, and L is the maximized value of the likelihood function (MLE) for the estimated model. It is usually described as the tradeoff between bias and variance (accuracy vs complexity) in such a model.

The algorithm is calculated the solution by an iterative process from a set of variables until AIC converges to the minimum AIC. We set the method to check both directions and $k = 2$. It searched forward/adding and backward/deleting in each iterative step until the minimum AIC is obtained. AIC not only rewarded goodness of fit, but also included a penalty when the model is over-fitting. The AIC methodology attempted to find the model, among a candidate set of models, that best explained the data with the fewest free parameters.

3.2 Support Vector Machine (SVM)

Support Vector Machines (SVMs) were first proposed as a classification problem of a binary class (i.e., yes/no) by separating 2 classes with a linear classifier function in a finite space. The ideas were later extended to multiple classes (i.e., yes/maybe/no), and non-linear classifier functions. Then, Vapnik suggested an idea of using soft margin to improve its predictions, which takes mislabeled (by some penalty) data points into account when maximizing the margin. However, the algorithm can still not perfectly label all the data points. Our goal is to find optimal separating hyperplane which a classifier maximizes the margin.

Let's take a look of a concrete example using a linear classifier function and

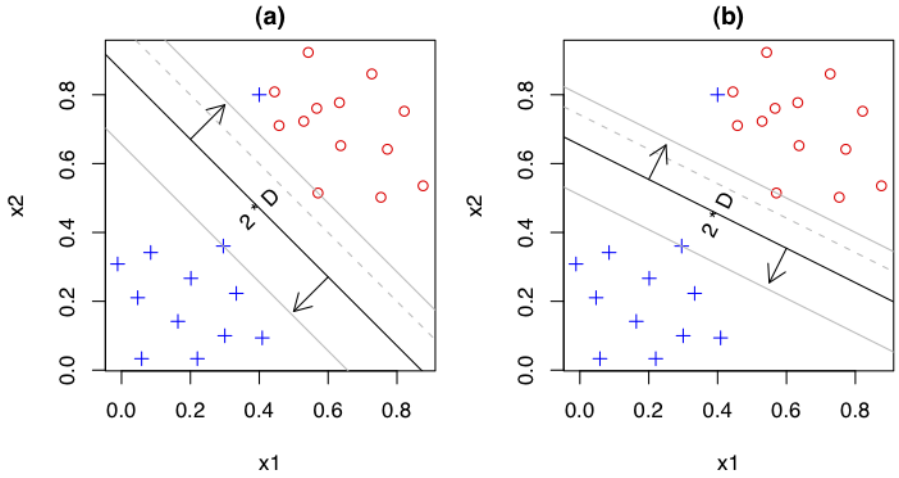


Figure 3.1: SVM Example

binary response variable in Figure 3.1. Both plots are based on simple two dimension SVM analysis. The points annotated by + and o are from two separate classes, there can always be a one line that can separate these classes as far as possible. Notice that one data point in each group is mis-labeled. The algorithm calculated the line by incorporating a slack variable (i.e., soft margin) in the implementation. However, just by looking at the two plots, one can easily tell the line in (a) is better than that in (b). The reason is simply because the former line separates these points further than the latter one.

In this study, we used the svm function in the e1071 package to implement SVM models (8). We used defaults parameters except using the classification mode, both linear and radial kernel functions. An overview of SVM algorithm can be found in Appendix C.

3.2.1 Variable Selection by Scad-SVM

A function called scadsvc in the penalizedSVM package is able to select important motifs (7). This is an add-on from the svm function. It borrows the idea of the

SVM algorithm and scad penalty function to do the predictions. This implies that this package uses the linear classifier in the SVM. We chose to use the smoothly clipped absolute deviation (Scad) to select significant features since it has been shown to have better theoretical properties than the $L1$ function (9) and used the selected motifs to do predictions. However, it sometimes does not choose any motif, so no predictions were given.

The optimal hyperplane with maximal margin is solved by convex optimization. Maximizing the margin can be achieved by solving:

$$min_{b,w} \sum [1 - y_i f(x_i)] + pen_\lambda(\beta_i) \tag{3.1}$$

The penalization term for SCAD SVM has the form

$$pen_\lambda(\beta_i) = \sum_{i=1}^{d} p_\lambda(\beta_i) \tag{3.2}$$

where the Scad penalty function for each coefficient β_i is defined as

$$p_\lambda(\beta_i) = \begin{cases} \lambda|\beta_i| & \text{if } |\beta_i| \leq \lambda \\ -\frac{(|\beta_i|^2 - 2a\alpha|\beta_i| + \lambda^2)}{2(a-1)} & \text{if } |\beta_i| \leq \text{a}\lambda \\ \frac{(a+1/\lambda^2)}{2} & \text{if } |\beta_i| > a\lambda \end{cases} \tag{3.3}$$

with a linear kernel function, tuning parameters $a = 3.7$ and $\lambda = 0.01$. $p\lambda(\beta)$ corresponds to a quadratic spline function with knots at λ and a. For small coefficients, the scad takes on the same behavior as the $L1$. For large coefficients, however, the Scad applies a constant penalty, in contrast to the $L1$ penalty, which increases linearly as the coefficient increases. This absolute maximum of the scad penalty, which is independent from the input data, decreases the possible bias for estimating large coefficients (7).

3.3 Classification Tree - Recursive PARTitioning Algorithm

Until now, many people have developed classification algorithms and most algorithms are implemented based on the ideas of CART (classification and regression tree). It has a classification or regression implementation depending on the characteristics (i.e., continuous, or categorical) of the response variable. A similar algorithm is also available in R which is called Recursive PARTitioning (rpart) algorithm. It gives an option to choose the two implementations. We use the default parameters except setting the classification mode since the dependent variable in the every sampling dataset is categorical (8). The algorithm has two major procedures:

- It first finds a predictor variable that best splits data into two groups, then applies the same process separately to each subgroup or other variables recursively until the subgroups either reach a minimum size or until no improvement can be made.
- The method above could make the tree too complicated, so the algorithm then trims the full tree from the back using cross-validation (CV). The final model contains subtrees with the lowest estimate of risk and is much simpler than the one before.

A brief summary of the Tree algorithm is outlined in Appendix D and for more information, please refer to (8).

Evidently, rpart automatically created and pruned a tree, so we used the nodes of the tree as its selected motifs. We did not have the full model like in the logistics regression or support machine vector models.

CHAPTER 4

Empirical Analysis

In this chapter, we present the consensus motifs obtained from Transfac (13), the analyzed pairs and their training and 5-fold CV analysis results for each predictive models. A total of 8 variations of predictive models are used in this study. Once all the error rates are obtained from each analyzed pair, we summarize each model's result in a boxplot and compute the accuracy by comparing each data's motif with known consensus motifs.

4.1 Consensus Motifs

Transfac provided 12 different TF's PWM(s) and each TF may have a multiple of PWMs (13). The goal was to translate each TF's PWM(s) into a motif consensus. For each PWM, we intended to find a nucleotide (i.e., A, C, G, T) in each position which occurred more frequently than the other three nucleotides. We randomly picked a nucleotide if there were two or more nucleotides that had the same maximum count of that position. We assumed they are invariant positions. Then, if there was more than one PWM for a TF, we summarized them into one consensus since there was always a good portion of bps that were overlapping. It was often the case that the consensus motif's length was shorter than the original ones. The 12 motifs are shown in Table 4.1.

We used these consensus motifs in Table 4.1 to compare with the motifs selected by a model.

Oct4	Sox2	Nanog	Esrrb
TATGCAAAT	CATTGTT	CGATTA	AAGGTCA
Stat3	cMyc	nMyc	E2f1
TTCCCGGAA	ACCACGTG	CCACGTGAC	TTTGGCGCG
Smad1	Zfx	Tcfcp211	p300
CAGACA	AGGCCTGG	AACCAGT	GGGAGTG

Table 4.1: Summarized Consensus Motifs

4.2 Analyzed Pairs

We want to correctly identify motifs among the motifs selected by each predictive model. That is, we compare each selected motifs from the model with the expected TF's consensus. Since there were only 12 consensus motifs (See Section 4.1), among the 66 (i.e., 12 choose 2) pairs only 38 of them were "selected pairs" (See Section 2.4) however, we got results from 51 "selected pairs". This number was reduced from 64 because some dataset could not convergent points when applying to certain datasets of a selected pair.

In order to have consistent results when comparing each model's performance (i.e., error rates) and selection power (i.e., selected vs consensus motifs), we focused on these 38 pairs instead. We called these pairs as "analyzed pairs". As we mentioned before, we cannot draw a solid conclusion about the results from S_{ab_s} (See Section 2.7) data so we mainly focused on the results from S_{ai_s} and S_{bi_s}.

For each S_{ai_s} and S_{bi_s} in the analyzed pairs, we compared each model's performance using training and CV data to conclude the best model. The training data was the input data which was the same as the sampling dataset (See Section 2.7). A CV data contained approximately a 1/5 of dimension in observation (i.e., enriched motifs) size than training's dataset. To create the 5-fold data, we first split the training data by their class (i.e., -1/1), and then we created five subsets

(randomly selected without replacement) for both classes. Finally, we combined the two subsets of different class to form the five datasets, with the size of each dataset being roughly equal to each other. These data were used for the 5-fold CV analysis.

We summarized the number of TFBSs of S_1, S_2, S_a, S_b, and S_i in Table **??** for each pair, which was discussed in Chapter 2. We assigned each pair a number between 1 and 105 in the first column. The two TF names for that pair is shown in the second column. For example, Oct4Sox2 means Oct4 and Sox2 is from S_1 and S_2, respectively. The number of TFBSs in S_1, S_2, S_a, S_b, and S_i is presented in the 3^{rd} through 7^{th} columns. The 8^{th} and 9^{th} columns indicate whether a pair was a selected or analyzed pair, respectively.

4.3 Eight Predictive Models

We proposed 8 different variations of predictive models based on logistics regression, support machine vector and the classification tree model as follows:

1. Logistics Regression (L)
2. SVM by Linear Kernel (LS)
3. SVM by Radial Basis Kernel (RS)
4. Stepwise Logistics Regression (Sl)
5. SVM by Scad (S)
6. RS by variables selected in L (SlR)
7. RS by variables selected in S (SR)
8. Classification tree (T)

The parentheses at the end of a line indicates the abbreviation for that model. For example, L indicates Logistics regression and S represents Scad SVM. We applied these 8 models described above to each training and 5-fold CV to get summarize results.

We note that some models sometime may not converge (i.e., LS or S) when applying to some datasets. For the results shown in the following sections, we first checked which contrast dataset(s) in a selected pair followed such case, then we filtered out those pairs in our analysis if at least one model could not find a convergent point of a contrast dataset.

4.4 Computing Error Rates

For each predictive model, we wanted to construct a 2×2 contingency table to compute its error rates. The "predict" function in R, a statistical software, can predict each motif class (i.e., -1/1) directly. However, it did not work for L, Sl, and S. These three types of models provided fitted values of a range between 0 and 1 for L or $-\infty$ and ∞ for S instead. We chose a cutoff value to be .5 for L and Sl and 0 for S. That is, if a fitted value was greater than .5 in L, we set it as 1 and -1 otherwise. The same reasoning applied to S. Also, sometimes R may give warning when the "predict" function predict 0 or 1 for a particular enriched motif of a logit models. This was due to the fact that CMF tried to discriminate the 2 sets of datasets. Once we had the predictions, we constructed a 2×2 contingency table based on the predicted and observed class frequency counts.

We also applied the 5-fold CV method to quantity predictive power. We used a combination of four folds as the training set and used the remaining one as the testing set to make predictions (i.e., -1/1) one at a time and for a total of five cycles. Then, we constructed a 2×2 contingency table from the five observed and predicted CV sets. It is worth noting that for Sl, S and T, we used four CV

sets' selected motifs and made predictions based on the testing set for each CV procedure.

Once the counts of a contingency table were obtained for both the training and 5-fold CV analysis, we can easily calculate a model's error rates. It is calculated by taking the sum of the two off-diagonal values and dividing it by the sum of all four values in a contingency table. The two diagonal counts indicate that it was accurately identified.

In the following sections, we aim to discuss the model performance when using all variables, model selection methods, and reduced model methods. We also calculate the percentage of correctly and incorrectly identified motifs for each model.

4.5 Performance Testing - Original Method

We have demonstrated the overall error rates of L, LS, RS, T, TL, and TR of S_{ai_s}, and S_{bi_s} in all analyzed pairs in Table 4.1. We note that L, LS, RS used all the motifs in S_{ai_s} or S_{bi_s} to do the predictions. Here, we included T, TL, and TR to show that TL and TR had better performance (i.e., error rates) than T's predictions since T is one of the three types of models that we are studying.

For the two, S_{ai_s}, S_{bi_s}, datasets in each pair, we computed its error rates (See Section 4.4) and summarized each model's error rates in Figure 4.1. Each boxplot consisted of 76 ($= 38 \times 2$) values. We multiplied by two because the error rates from S_{ai_s}, and S_{bi_s} data. The error rates between S_{ai_s}, and S_{bi_s} were very similar for the same type of predictive model so we aggregated them altogether. Also, we generally drew the same conclusions if we only ran models on training data based on the 55 pairs' result. It was 9 pairs short because some models did not predict two classes (i.e., -1/1) or converge based on a classifier function (i.e., linear kernel) in both training or 5-fold CV methods.

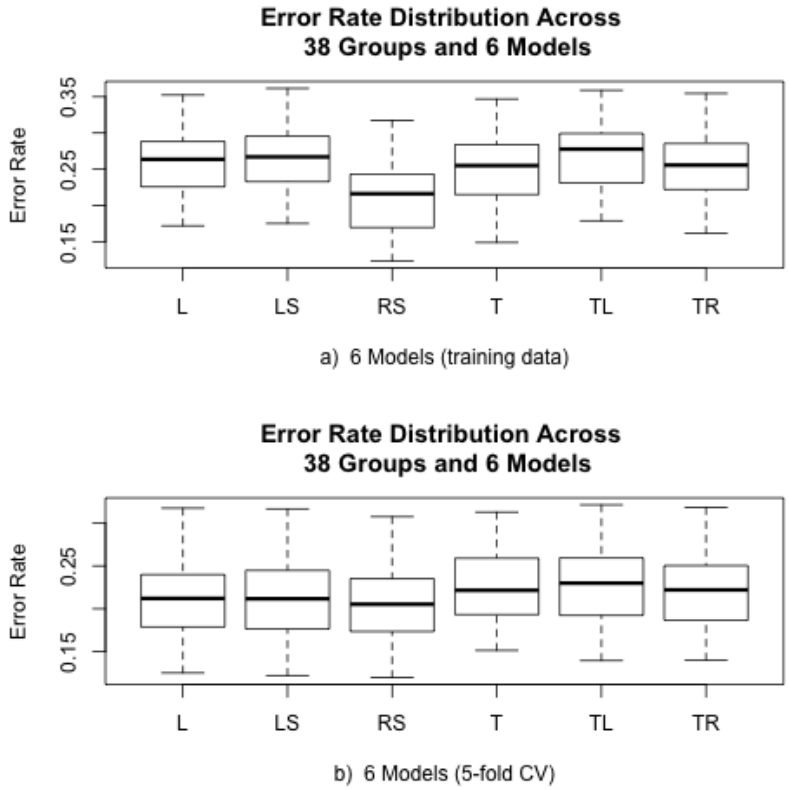


Figure 4.1: a and b are the distributions error rates of the training and CV based on the original method.

Both plots in Figure 4.1 have clearly shown that the RS (Radial SVM) had the lowest overall mean and median error rates when using all motifs/variables in the dataset since the radial kernel was designed to make good predictions in high dimensions. However, it was very interesting to see that TL (L using selected motifs from Tree) and TR's (RS using selected motifs from Tree) predictions are not lower than T's. We conclude that using T's model, one does not need to apply its selected motifs to the RS model to improve its prediction rates.

Model	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
L_n	0.1721	0.2260	0.2637	0.2594	0.2882	0.3524
LS_n	0.1753	0.2337	0.2670	0.2636	0.2934	0.3615
RS_n	0.1233	0.1697	0.2162	0.2115	0.2428	0.3175
T_n	0.1490	0.2159	0.2550	0.2511	0.2837	0.3468
TL_n	0.1790	0.2327	0.2778	0.2709	0.2989	0.3589
TR_n	0.1616	0.2235	0.2557	0.2561	0.2852	0.3545
L	0.1253	0.1790	0.2121	0.2109	0.2394	0.3172
LS	0.1223	0.1772	0.2117	0.2129	0.2447	0.3165
RS	0.1198	0.1738	0.2054	0.2062	0.2347	0.3073
T	0.1515	0.1933	0.2216	0.2252	0.2586	0.3126
TL	0.1397	0.1933	0.2299	0.2286	0.2588	0.3211
TR	0.1402	0.1875	0.2220	0.2200	0.2493	0.3180

Table 4.2: 5-Number Summary of Error Rate

The five-number summary for these models are shown in Table 4.2. The subscript n indicated the results were from the training data and the 5-fold CV otherwise. In general, the error rates from CV tended to be lower than the training's and both methods indicated that RS_n seemed to have the lowest error rates among all. However, it was quite interesting to note that T performed the second best overall and the number of motifs used for each dataset was 4.5 on average.

4.6 Performance Testing - Selection Method

We intended to find a model that minimized the error rates and number of selected motifs, and at the same time maximized the number of matched motifs after applying model selection methods. We demonstrated the performance of Sl, S, and T. Then, we compared their selected motif with the consensus motifs described in Section 4.1.

In Figure 4.2a, the prediction errors for T seemed to be better than Sl and S. However, SlR and SR have yet better predictions than T. In the last section, we have shown that the predictions did not improve if we applied the selected motifs from T to TL or TR. This is why we only showed T's predictions in Figure 4.2a and it is the same as the boxplot shown in Figure 4.1.

By looking at S, Sl and T's results, S's selection method seemed to be unstable and sometimes did not select motifs because of the way the method chose λ for the its penalty function. We used λ=0.01 for all the datasets and it was very computationally intensive to apply a reasonable λ to each dataset, especially when we performed the CV procedures. This is reason we didn't compute CV results for S model. However, we usually drew the same conclusion between the training and the 5-fold CV results in terms of performance when comparing the same models. In Figure 4.2b, we can see that SlR's prediction was better than T's, but the number of motifs selected in SIR was usually about four times more than T's on average.

Let's take a closer look at the five-number summary of these models' error rates in Table 4.3. The mean of the overall error rates in SlR_n and SR_n were 0.2291 and 0.2241, respectively. Each selected about 12 or 14.5 motifs on average for each training dataset. That was about 44.4% or 32.4% of dimension reduction from the original dataset. Each original dataset contained about 21.4 motifs. In general, SR_n gave a better prediction than SlR_n because it selected more variables/motifs

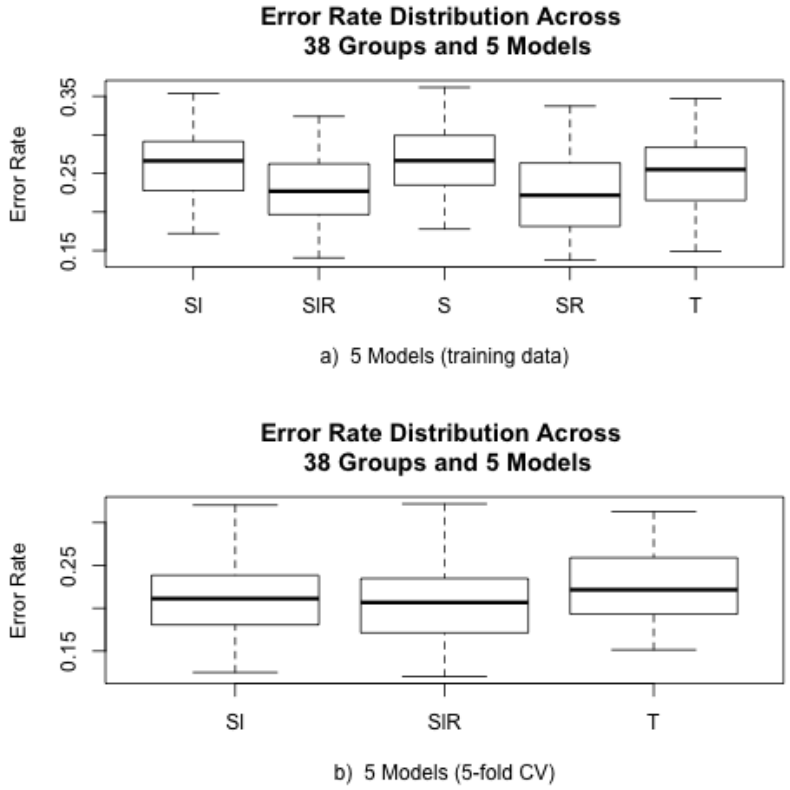


Figure 4.2: a and b are the distributions error rates of the training and CV based on the selection method.

Model	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Sl_n	0.1721	0.2278	0.2662	0.2606	0.2913	0.3539
SlR_n	0.1407	0.1966	0.2269	0.2291	0.2611	0.3239
S_n	0.1778	0.2360	0.2665	0.2646	0.2989	0.3615
SR_n	0.1376	0.1820	0.2216	0.2241	0.2635	0.3372
T_n	0.1490	0.2159	0.2550	0.2511	0.2837	0.3468
Sl_n	0.1252	0.1813	0.2113	0.2113	0.2384	0.3203
SlR_n	0.1202	0.1714	0.2067	0.2067	0.2347	0.3218
T_n	0.1515	0.1933	0.2216	0.2252	0.2586	0.3126

Table 4.3: 5-Number Summary of Selection Error

in the same dataset but this was not always the case since Sl and S may sometimes select different motifs. One thing to note is that the mean of T_n's error rate only differs by about 2.7% (0.2511) with SR_n's but the number of selected motifs drops significantly. The two models differ by almost 10 motifs per model.

Moreover, S did not produce any kind of ranking system like in SlR or T, nor did it always select motifs for us to evaluate. We attempted using a brute force method by iteratively setting λ equal to .0001, .0005, .001, .005, .01 and increased from .05 to 2 by .05 increment at a time. We stopped the iteration if the number of selected motif(s) was/were less than or equal to T. Otherwise, we did not consider that particular dataset since we cannot compare the variable selection methods between the three (i.e., Sl, S, and T) models. There were many problems to this approach, since T often selected only one motif so it was almost impossible to compare them. Therefore, we did not include the results in this study.

We believe SR may not be an ideal method since the computation time was not very stable and occasionally did not select motifs since every dataset should have at least one important motif identified by CMF. The second-best performance model selection method was SlR, because its overall average error rate was 0.2291

and it selected at least one motif in most of datasets. In addition, we can use the p-value of each selected motif as rankings to conclude their motif selection. Lastly, the error distribution range was smaller in SlR_n than in SR_n.

4.7 Selection Method - Motif Selection Power

In the previous two sections, we have seen that RS, SlR and T seemed to give a good performance under different situations. In this section, we want to compare the consensus motifs (See Section 4.1) with the ones selected by the model's selection and reduced model methods based on the error rates from each training dataset.

For any comparison, we assumed a selected motif is matched with the consensus motif (See Section 4.1) only if a selected motif is at least five bps long and matched the consensus motif in a sliding window containing at most two mismatches. Our procedure was as follows: Given two (selected vs consensus) motifs, we used the longer sequence to compute its reverse complement. Then, we used the shorter motif to compare with the longer one with both its original strand and reverse complement by every possible sliding window. A sliding window is matched if it was at least five bps long and contained at most two positions that are mismatched.

A special case was when we were comparing the starting and ending positions. If the first or last five positions of a shorter motif are matched and the shorter one was more than five bps long, we considered them as matched motifs. For example, ACGTGCAT and GGACGCG is a match since the last five positions of the shorter motif is matched with the first five positions in the longer one.

We recorded motif(s) selected by Sl, S and T. Then, we compared them with the consensus motifs (See Section 4.1) based on the two analyzed TFs in each pair. For any pair 1 and 2 TFs, we are expected to find 2 and 1 TF motifs in X_{ai_s}

and X_{bi_s}, respectively. For the sake of simplicity, we always checked motifs 1 and 2 regardless of the type of dataset (ai, or bi). In this way, we are able to calculate if CMF identified an unexpected TF motif.

The expected and unexpected percentage for each model is calculated based on the 38 pairs. The procedure of calculating correct and incorrect percentage is as follows:

1 For each model and dataset, if there was at least one matched motif from b in X_{ai_s} or a in X_{bi_s}, then we recorded 1 as correct, and 0 otherwise.

2 For each model and dataset, if there is at least one matched motif from a in X_{ai_s}, and b in X_{bi_s}, then we recorded 1 as incorrect, and 0 otherwise.

3 We computed the expected and unexpected percentage by summing up the total counts divided by the number of datasets analyzed.

The results in Table 4.4 indicate that each dataset was composed of about 21.4 variables/motifs on average among the 76 ($= 38 \times 2$) datasets. In addition, T seemed to have better overall predictions among the three selection models (Sl_n and S_n) and its overall mean is about 0.2511. It selected about four motifs on average with 79% of dimension reduction.

4.8 Selection Method - Reduced Model

In the last section, the results indicated that T and Sl may be good methods to use. However, Sl usually selected too many motifs per dataset compared to T. In order to make a reasonable comparison in terms of the number of motifs for each model, we proposed two types of models since L and Sl provide p-values for each estimated coefficient as follows:

1 Select motif(s) with p-value $\leq .005$ for both the L and Sl model and

use the selected motifs to run SR. We denote them as rL and rSl.

2 Select motif(s) with p-value $\leq .005$ for both L or Sl model and motifs use in T and used the selected motifs to run SR. We denote them as rLT and $rSlT$.

In this way, we can reduce the number of selected motif(s) and have a better comparison with the T. The expected and unexpected percentage of these models are shown in Table 4.4.

Model	Expected	Unexpected	# of Motif
T	0.816	0.421	4.47
rL	0.829	0.329	6.123
rSl	0.842	0.474	6.97
rLT	0.882	0.566	7.61
rSlT	0.882	0.632	8.27
Sl	0.882	0.855	11.91
S	0.908	0.737	14.40
All	0.934	0.947	21.35

Table 4.4: Selection Error Rates

In Table 4.4, *All* used all the motifs in a dataset. This gave us a way to see the upper bound of the correct and incorrect percentages. It turned out that 93% of the dataset had at least one motif matched with consensus motif. This was equivalent to 71 of the 76 datasets contained at least one estimated consensus motif correctly identified by CMF.

The Table 4.4 is sorted by the average motifs used in each model. The expected percentage in rLT, $rSlT$ and Sl were the same, which was 88.2% and about 5.3% different when comparing the results to the correctness of *All* motifs. The number of motifs selected by rLT was about 13.7 less than

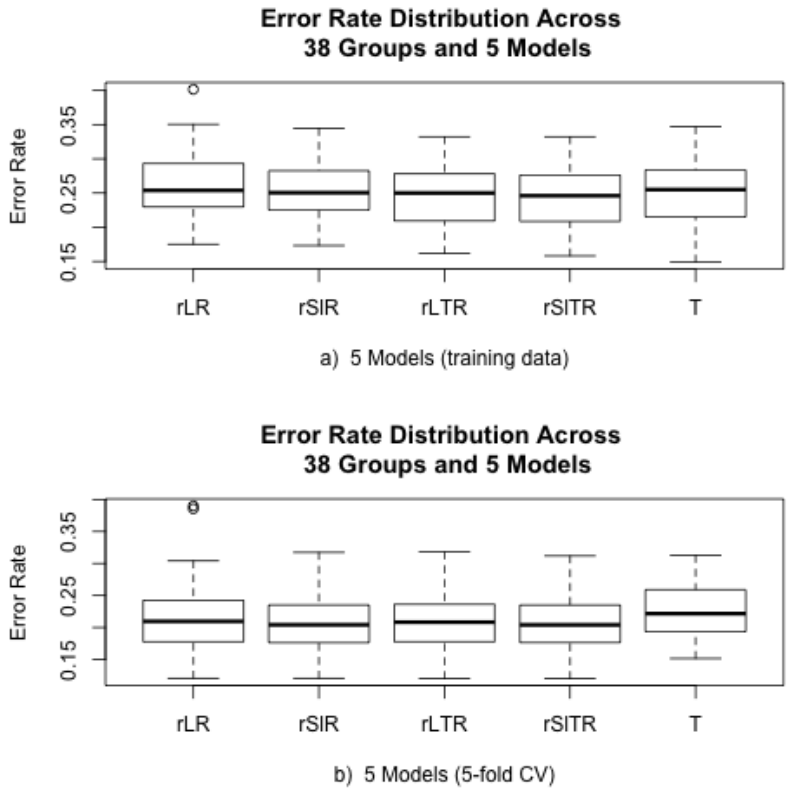


Figure 4.3: a and b are the error rates and CV distributions based on the reduced method.

All. One interesting fact was that the expected % and the number of motifs in T and rL were the lowest and closest. The unexpected % and the number of selected motifs were the lowest in model rL and the expected % was about 12% lower than the *All* model but the difference between the motifs used was about 17. Also, the error rates obtained from 5-fold CV analysis for T was lower than rL. Therefore, we strongly believe that T was a much better model.

Let's take a look at the overall distributions of what each of these models in Figure 4.3. We constructed the 5-fold CV analysis results for $rSlT$. For each of the four CV combinations as training data, we used the union of the

motifs/coefficient whose p-value less or equal to .005 in L. Then, we selected T and then used RS (i.e., SVM radial kernel) to fit a model; We used the testing set to get its prediction errors. Once we obtained the predictions for all five testing sets, we computed a contingency table and error rates (See Section 4.4). A similar reasoning can be applied to rL, rSl and rLT. We note that rL and rSl did not take the selected motif from T into account.

As we can see from both plots in Figure 4.3, the results for each model was relatively similar and we have not found one method that was significantly better than the other in terms of predictions power. However, we have shown in the previous sections that T seemed have the best performance by considering the number of motifs used for each dataset.

CHAPTER 5

Demonstrating Our Analysis Procedures Using Oct4/Sox2 Pair

In this chapter, we demonstrate our analysis procedures by choosing Oct4/Sox2 pairs since these are well-known TFs and it is quite difficult to present each pair's findings. In each sampling dataset in Oct4/Sox2, we present 1) the number of enriched TFBSs before and after applying the sampling procedures (See Section 2.7), 2) the performance of each model (See Section 4.3), 3) the number of selected motifs that are matched with the consensus motifs in the contrast data (See Section 4.1), and 4) the number of expected and unexpected motifs identified by model selection methods.

5.1 Number of Enriched TFBSs in the Sampling Datasets

The number of enriched Oct4 and Sox2 TFBSs in the sampling dataset before (original) and after (sampling) applying the sampling method (See Section 2.7) are shown in Table 5.1.

type	Original Oct4	Original Sox2	Sampling Oct4	Sampling Sox2
ab	2306	1946	2306	1946
ai	1048	1768	1048	1768
bi	1199	2327	1199	2327

Table 5.1: Number of Enriched TFBSs in Oct4 and Sox2 Datasets

The number of TFBSs in the Original and Sampling sets were the same for both TFs since the preset cutoff was 3000 bps (See Section 2.7). We note that the S_{ab} dataset was not used in any analysis but we presented it here for completion.

5.2 Model Performance

For each data type (i.e., S_{ab}, S_{ai}, S_{bi}), we constructed 8 predictive models which were discussed in Section 4.3. Let's take a closer look of each model's 2×2 contingency table and their corresponding error rates in the next few sections.

5.2.1 Oct4 contrasts with Sox2

In Table 5.2, RS model had the lowest error rates for both training and 5-fold CV results among all the models. The Sl and T had the lowest error rates in the training and CV among the three (i.e., SR, S and T) selection models, respectively. The training results have shown that T used only 2 motifs to make predictions while Sl used 13 and the error rate was only about 1.4% higher. We believe T was a better model based on the 5-fold CV results. The table also showed that Sl improved about 2.5% by applying SlR and reduction rates were approximately 40% (=1-13/21) and 29% (=1-15/21). We have shown in the last chapter that the λ in S was very sensitive so that its predictions were very unstable from data to data. We believe that SlR's result was more robust. In addition, the motifs selected in Sl and S may not be the same. Even though the number of selected motifs for S is higher, SR did not always have a lower error rate compared to SlR which was the case in the table shown above.

rLR, $rSlR$, $rLTR$ and $rSlTR$ are shown to indicate that T may be a better

	Training Result						CV Result				
Model	1_1	-1_1	1_-1	-1_-1	**Error**	# of Motif	1_1	-1_1	1_-1	-1_-1	**Error**
L	1942	364	519	1427	0.2076670	21	1935	371	526	1420	0.210959548
LS	1986	320	574	1372	0.2102540	21	1990	316	587	1359	0.212370649
RS	2015	291	484	1462	0.1822672	1950	356	541	1405	0.210959548	
S	1994	312	587	1359	0.2114299	15	NA	NA	NA	NA	NA
SR	2015	291	489	1457	0.1834431	15	NA	NA	NA	NA	NA
Sl	1943	363	518	1428	0.2071966	13	1926	380	522	1424	0.2121355
SIR	2025	281	496	1450	0.1827375	13	1962	344	540	1406	0.2079022
T	1690	616	323	1623	0.2208373	2	1775	531	435	1511	0.2271872
TL	1981	325	658	1288	0.2311853	2	2008	298	728	1218	0.2412982
TR		1957	349	617	1329	0.2271872	2	1932	374	572	1374
0.2224835											
rLR	2002	304	591	1355	0.2104892	5	1996	310	612	1334	0.2168391
rSlR	1983	323	494	1452	0.1921449	7	1965	341	510	1436	0.2001411
rLTR	2002	304	591	1355	0.2104892	5	1987	319	595	1351	0.2149577
rSlTR	1983	323	494	1452	0.1921449	7	1965	341	510	1436	0.2001411

Table 5.2: Error Rate for S_{ab} Dataset

model model to choose from since the number of motifs selected lower but at the same time and it correctly identify the consensus motif more than 80% of the time.

5.2.2 Oct4 contrasts with Oct4 co-bound with Sox2

In the Table 5.3, the lowest error rates were RS, T and SR in their comparison groups. The results here again were consistent with previous ones. However, SR had a lower overall error rate than SlR's error rate for both methods. Since the difference is not much, we still believe SlR's prediction was more robust.

	Training Result						CV Result				
Model	1_1	-1_1	1_-1	-1_-1	**Error**	# of Motif	1_1	-1_1	1_-1	-1_-1	**Error**
L	572	476	337	1431	0.2887074	21	1940	366	519	1427	0.2081373
LS	571	477	343	1425	0.2911932	21	2000	306	580	1366	0.2083725
RS	666	382	345	1423	0.2581676	21	1943	363	552	1394	0.2151929
S	577	471	337	1431	0.2869318	10	NA	NA	NA	NA	NA
SR	664	384	358	1410	0.2634943	10	NA	NA	NA	NA	NA
Sl	570	478	343	1425	0.2915483	10	1940	366	519	1427	0.2081373
SlR	665	383	363	1405	0.2649148	10	1961	345	544	1402	0.2090781
T	541	507	277	1491	0.2784091	4	1763	543	408	1538	0.2236595
TL	543	505	315	1453	0.2911932	4	1995	311	669	1277	0.2304798
TR	572	476	330	1438	0.2862216	4	1918	388	592	1354	0.2304798
rLR	592	456	371	1397	0.2936790	2	1997	309	622	1324	0.2189558
rSlR	608	440	354	1414	0.2819602	4	1962	344	521	1425	0.2034337
rLTR	619	429	364	1404	0.2816051	5	1992	314	610	1336	0.2173095
rSlTR	632	416	354	1414	0.2734375	7	1962	344	520	1426	0.2031985

Table 5.3: Contingency table and Error Rate in S_{ai} Dataset

5.2.3 Sox2 contrasts with Oct4 co-bound with Sox2

In Table 5.4, the model with the lowest error rates were RS, T and SlR relative to their comparison models. Again, the results here were consistent with previous findings. However, SR's error rate was lower in the training and higher in the 5-fold CV analysis. Since the CV result was less biased, we believe SlR's prediction was better.

5.3 Number of Matched and Selected Motifs

In the last section, we have seen the prediction performance of each predictive model. In this section, we want to show the number of matched motif(s) in each type of dataset. The criteria for matching was explained in Section

	Training Result						CV Result				
Model	1_1	-1_1	1_-1	-1_-1	**Error**	# of Motif	1_1	-1_1	1_-1	-1_-1	**Error**
L	546	653	261	2066	0.2592172	20	1935	371	522	1424	0.2100188
LS	513	686	253	2074	0.2663074	20	1986	320	581	1365	0.2119003
RS	601	598	206	2121	0.2280204	20	1949	357	544	1402	0.2119003
S	528	671	254	2073	0.2623369	12	NA	NA	NA	NA	NA
SR	603	596	230	2097	0.2342598	12	NA	NA	NA	NA	NA
Sl	546	653	261	2066	0.2592172	13	1940	366	529	1417	0.2104892
SlR	624	575	228	2099	0.2277368	13	1961	345	545	1401	0.2093133
T	613	586	307	2020	0.2532615	6	1734	572	396	1550	0.2276576
TL	533	666	264	2063	0.2637550	6	2022	284	740	1206	0.2408278
TR	597	602	276	2051	0.2490074	6	1919	387	590	1356	0.2297742
rLR	566	633	267	2060	0.2552467	5	1994	312	618	1328	0.2187206
rSlR	566	633	267	2060	0.2552467	5	1964	342	522	1424	0.2031985
rLTR	597	602	276	2051	0.2490074	6	1994	312	618	1328	0.2187206
rSlTR	597	602	276	2051	0.2490074	6	1964	342	522	1424	0.2031985

Table 5.4: Contingency table and Error Rate in S_{bi} Dataset

4.1. Let's us just take a look at the number of matched motifs in Sl, S, and T, which is shown in the Table 5.5.

Type	Sl_{Oct4}	Sl_{Sox2}	S_{Oct4}	S_{Sox2}	T_{Oct4}	T_{Sox2}
ab	1	5	1	5	0	2
ai	0	4	0	5	0	2
bi	7	12	6	11	5	5

Table 5.5: Number of Matched and Selected Motifs in Oct4 and Sox2 Data

The consensus motif of Oct4 and Sox2 are TATGCAAAT and CATTGTT, respectively.

In S_{ab} data, we expected matched motifs from both Oct4 and Sox2. A motif named ATTATGCAAAT was matched with consensus motif of Oct4 in both

Sl and *S*. However, *T* did not select any motif. The same five matched motifs in *Sl* and *S* were AGAACAATGG, GAACAAAGGA, AGAACAATGGA, AGAACAAAGGA, and AGAACAATGGG. *T*'s matched with AGAACAATGG, and GAACAAAGGA. These two matched motifs were also in the union of *Sl* and *S*.

In S_{ai} data, we expected matched motifs from Sox2 only. As we can see from Table 5.5, we did not find Oct4 motif matched in all three models. The four matched motifs in *Sl* were CATAACAAAGG, ACAAAGG, CCATTGTTAT and ACAAAGAA. The five matched motifs in *S* were the same matched motifs in *Sl* plus ATGACAAAGG. The two matched motifs in *T* were ACAAAGAA, and CCATTGTTAT. These two motifs were also included in *S* and *Sl*'s selection.

In S_{bi} data, we expected matched motifs from Oct4 only. The five matched motifs in *T* were ATGCAAA, ATTTGCATAACAATGGC, CATTTGCATGACAATGGA, CCTTTGTTATGCAAAT and CATTTGCATAACAAAGG. The six matched motifs for *S* were those matched in *T*, plus CATTTGCATAACAATGG. The seven matched motifs for *Sl* were those matched in *S*, plus CATTTGCATGACAATGG.

In addition, Table 5.5 also has shown that there were unexpected motifs. *T*'s unexpected motifs were ATTTGCATAACAATGGC, CATTTGCATAACAAAGG, CATTTGCATGACAATGGA, CCTTTGTTATGCAAAT, and GAACAATGGA. *S*'s were the same as the motifs in *T*'s plus AGAACAATGGG, CATTTGCATAACAAAGG, GGAACAATGGG, GAGAACAATGGA, GAGAACAATGGG, and TCCATTGTTCC. *Sl*'s are AGAACAATGGA, CATTTGCATAACAAAGG, CATTTGCATAACAATGG, GAGAACAATGGG, GAGAACAATGGA, GGAACAATGGG, and TCCATTGTTCC.

Since there was so many unexpected motifs identified in all three models, we believe there may be Context-Dependent (CD) in *bi* data which was

discuss in Mason's paper. For this data, we went ahead to find if such CD motifs exist, we found that AACAATG and AACAAAG were CD scenarios where AACAATG is an unexpected motif (Sox2), and AACAAAG was a CD motif (Oct4 co-bound with Sox2) where the position of these two motifs were aligned by their positions.

Furthermore, based on this detailed analysis, we can see that T's motif selection gave the least motifs and the number of motifs used to do predictions was less than the other ones. Again, it seemed T can effectively find motifs that correctly match the consensus motifs.

CHAPTER 6

Conclusion and Recommendations

6.1 Conclusion

In this study, we have demonstrated how to generate the three BD sets, predictive and sampling datasets for each analyzed pair. Our finding suggested that *RS* (i.e., SVM with the Radial kernel) was the best method when including all variables/motifs but it did not improve prediction rates (i.e., low error rates) when running *RS* on the motifs selected from the classification tree model. Moreover, the predictive model of selection methods based on CMF's output was a good way to reduce a data's dimension and its false positive motifs, while keeping its performance and selection power competitive.

We specifically compared the *RS* including all variables in the dataset (the best performance model) and with the classification tree model in every analyzed dataset. The dimensionality of the classification tree model (i.e., *T*) selected about 17 fewer motifs than *RS*. In addition, about 82% of each *T*'s model contained at least one correct motif compared with *RS*'s 93%. As we can see, there was only an 11% difference. Moreover, at least one unexpected motif selected by T was 42% compared with RS's 94%. In terms of error rates, *RS*'s training and 5-fold CV was 21.15% and 20.62%, compared with *T*'s 25.11% and 22.52%, of which *RS* was only better by less than 1% and 2% in both methods, respectively. Also, we performed 50 simulations to test the

total time to construct the same dataset for both models, and we found that T's time is 48.731 seconds compared with RS's 182.549 seconds, which was about 3 times longer than T's. Based on careful investigation, we conclude that classification tree models are generally time efficient to construct and provide reasonable error rates while correctly identifying motifs compared with the other predictive models.

6.2 Recommendations

Although our study did not focus on S_{ab_s} data, we believe the selected motifs in this type may contain some portion motifs from both a and b and had a different motif expression, which can be explored in our future studies.

The classification tree model had a high chance of selecting expected motifs for each dataset; however, we believe the classification tree can also correctly identify unknown or new motif(s) that no one has discovered thus far. This may be one of the reasons why the number of unexpected motif rates is so high for all the models. One can confirm our claim by using our findings based on stepwise logistics regression to perform a series of actual experiments to confirm whether or not it is correct. Of course, it is not necessary to test all GC enriched motifs since stepwise logistics regression usually includes all the selected motifs as the ones in the classification tree model and gives a nice ranking system (p-value) of each selected motifs in its output table.

CMF seems to correctly identify motifs, but there is much improvement needed. The unexpected motif rate for each model is pretty high assuming all the unexpected motifs are wrong. This is why Mason mentioned context-dependent (CD) motifs which do not happen often. Another issue is that by setting a different seed number in CMF's input, the performance of pre-

dictive models may give a worse results. Of course, the result for motif (i.e., selected vs consensus) comparison may also change. In the future, we may adapt CMF or related algorithms for studies more suitable for our purposes, so that not as many input parameters should be allowed.

APPENDIX A

CMF Algorithm

CMF algorithm takes in 2 sets of nucleotide sequences, S_1 and S_2. The algorithm first computes all possible w-mers of each sequence in both sets and each position contains one of the 4 nucleotides ({A, C, G, T}). For every w-mer in a TFBS sequence, we define it as X = (x_1, ..., x_w) where w is 7 in this project. LR is computed as follows:

$$LR(x) = \frac{\sum_{i=1}^{w} \theta_{ix_i}}{\sum_{i=1}^{w} \theta_0(x_{i-1}, x_i)} \tag{A.1}$$

where $\theta_0(x_{i-1}, x_i)$ is the background model. Then, it predicts BSs by computing likelihood ratios (LRs) of all w-mer greater than τ (=2/3) to prevent too many false positives contributing to the resulting PWM. The author claims that $\tau = 2/3$ works the best. (3). The author claims by correcting for the influence of false positives in Equation A.2, the algorithm becomes robust to the discretiztion of τ.

In addition, CMF finds enriched w-mers in S_1 and compare with w-mers in S_2. Then, a z-score is computed for each w-mer as follows:

$$z(x) = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1 - \hat{p}(\frac{1}{L_1} + \frac{1}{L_2})}} \tag{A.2}$$

where L_1 and L_2 are defined as the total number of w-mer in S_1 and S_2 and $\hat{p_1} = \frac{C_1}{L_1}$, $\hat{p_2} = \frac{C_2}{L_2}$ and $\hat{p} = \frac{C_1+C_2}{L_1+L_2}$ with C_1 and C_2 the number of time a w-mer in S_1 and S_2, respectively. It does not use t-score distribution since

the value of L_1 and L_2 are usually large enough to follow as close to a z distribution as possible.

Furthermore, the author defines a seed center at a w-mer x by incorporating neighboring w-mers, where a w-mer y is considered a neighbor of x if it matches x with at most m=2 mismatches and include only those neighbors y that are overrepresented in the same set of sequences as the center w-mer, x, is included in the seed (i.e., $\{y : z(y)z(x) > 0\}$). The algorithm then defines sub-neighborhood y_j of the central w-mer x as those neighbors with the same mismatch position(s) k, where k is a size m subset of {1, ..., m}. For each sub-neighborhood y_k, it computes the z-score with the counts C_1 and C_2 being the total number of counts of w-mers in y_k and the best sub-neighborhood is selected. Each seed is summarized by two m by 4 count matrices, $N_1^{(1)}$ and $N_2^{(2)}$, where $N_1^{(1)}$ and $N_2^{(2)}$ are composed of the best sub-neighborhood sites in S_1 and S_2, respectively. This step is called seed creation in the paper.

Once the best sub-neighborhoods are determined, Mason uses an iterative approach to update the PWM for each seed whose sub-neighborhood z-score ranks within the T=10, the number of top seeds to test. The iterative algorithm is summarized in the following:

For t = 1, 2, 3, ..., T

1. Given initialize $N_1^{(1)}$ and $N_2^{(2)}$ by seed creation, add 5% pseudo-counts to each position in $N^{(t)}$ and normalize each row into probabilities to prevent CMF from being trapped in a local mode;

2. Update PWM, $\Theta^{(t)}$ from normalized pseudo-count matrices and

$$N^{(t)} = (N_{ij}^{(t)})_{w\times 4} = max(N_t^{(1)} - N_t^{(2)}, 0); \tag{A.3}$$

3. Scan S_1 and S_2 with $\Theta^{(t)}$ and determine $\tau^{(t)}$. The false discovery rate of the sites in both S_1 and S_2 are taking into account;

4. Use sites with LR(x) $> \tau^{(t)}$ to create $N_{t+1}^{(1)}$ and $N_{t+1}^{(2)}$;

5. Algorithm stops once the distance between $\Theta^{(t+1)}$ and $\Theta^{(t)}$ is less than some ϵ by $d^{(t)} = \max|\theta_{ij}^{(t+1)} - \theta_{ij}^{(t)}|$.

At each iteration, the algorithm determines whether the motif should grow or shrink by 1 bp on either side based on the Bayes factors at the flanking positions (3). For each request seed, the output provides the location of the enriched sequence locations, its features and LRs from both sets, and the estimated PWM that is calculated by using each enriched feature.

APPENDIX B

Logistics Regression Algorithm

Logistic regression (logit) function has a excellent property which predicts the log odds of an observation equal to 1 (i.e, $P(Y = 1)$). The odds of an event is defined as the ratio of the probability that an event occurs (i.e., $P(Y = 1)$) to the probability that it fails to occur as follows:

$$odds(Y = 1) = \frac{Pr(Y = 1)}{1 - Pr(Y = 1)} = \frac{P}{1 - P} \tag{B.1}$$

Let P = π. The log odds is a logit function as follow:

$$log(odds(Y = 1)) = logit(\pi) = log\left(\frac{\pi}{1 - \pi}\right) \tag{B.2}$$

where,

$$log\left(\frac{\pi}{1 - \pi}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k = x_i^{'}\beta \tag{B.3}$$

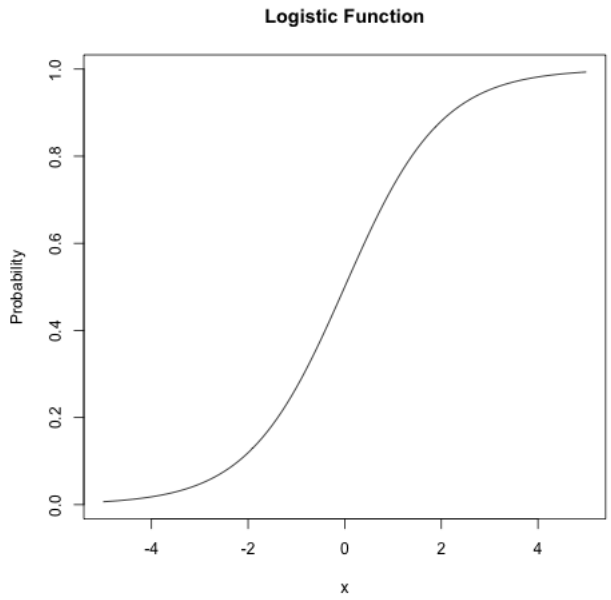
$$\text{where } x_i = (1, x_{i1}, \ldots, x_{ik})^{'} \text{ and } \beta = (\beta_0, \beta_1, \ldots, \beta_k)^{'}$$

and,

$$\pi_i(Y = 1) = \frac{exp(x_i^{'}\beta)}{1 + exp(x_i^{'}\beta)} \tag{B.4}$$

The odds scale is bounded by 0 and ∞ shown in equation B.1. Each value between 0 and 1 can be transformed when the roles of the response variable is switched by taking its inverse (i.e., 1/value) to a value in the range of 1 and ∞. The log odds scale varies from (-∞, ∞). The roles of the response variable is switched by multiplying -1 of the log odds. The outcomes (i.e., 0/1) are equally likely in both odds and log odds.

The logit function shown in equation B.4 as an example maps any value of the OLS estimates to a probability between 0 and 1. If log odds are linearly related to Xs, then the relation between Xs and π_i(Y=1) is nonlinear, and has the form of the S-shaped curve like in the following:



Now the question is: How do we measure the fit of the model? In linear regression, one can use R^2 of the model, but it is not available in the logistics regression. A deviance is provided in each model instead. In general, the smaller the deviance the better the model is. Logistics regression uses Maximum Likelihood (ML) to find the smallest possible deviance between the observed and predicted values. This is an iterative method in which the algorithm tries all possible solutions until it gets the smallest possible deviance. Once it finds the solution, it provides a final value for the deviance which is known as $-2logL$. L is ML. The ML equation is derived from the probability distribution of the dependent variable. Since each y_i represents a binomial count in the i_{th} population, the joint probability density function of $\mathbf{Y}$ for n observations is:

$$L = \prod_{i=1}^{n} \left(\frac{exp(x_i'\beta)}{1 + exp(x_i'\beta)} \right)^{\sum_{i=1}^{n} Y_i} \left(1 - \frac{exp(x_i'\beta)}{1 + exp(x_i'\beta)} \right)^{n - \sum_{i=1}^{n} Y_i} \tag{B.5}$$

A convenient way to conclude the overall fit of a model is comparing the null ($-2logL_{Null}$) and residual ($-2logL_p$) deviances. People often call the differences between these two terms a likelihood ratio (LR):

$$LR = -2log \left(\frac{L_{Null}}{L_p} \right) \sim \chi^2 \tag{B.6}$$

where, the degree of freedom (df) = the difference between the model with predictors and the null model. LR is assumed to follow a χ^2 distribution.

The idea here is to see if the model with predictors (i.e. residual model) fits significantly better than a model with just an intercept (i.e. a null model). One can test the goodness of fit using χ^2 statistic on LR. The χ^2 p-value can be obtained based on LR and df differences. If $\alpha = .05$ and if an associated p-value is less than 0.05, then it signifies that the model fits significantly better than a null model as a whole, and one should include the independent variable(s) in the model.

The material described here is based on (4) and (5).

APPENDIX C

Support Machine Vector Algorithm

Let us consider consider a training data D containing a set of n observations

$$D = \left\{(x^1, y^1), \ldots, (x^n, y^n)\right\}, \quad x \in \mathbb{R}^p,\ y_i \in \{-1, 1\} \tag{C.1}$$

where y_i indicates which X_i belongs to a group and each X_i is a p-dimensional real vector. The goal is to have a hyperplane that divides the points of 1 and -1 in $\mathbb{R}^p$. A hyperplane is defined as $\mathbf{w} \cdot \mathbf{x}$ + b = 0. $\mathbf{w}$ is a normal vector and perpendicular to the hyperplane.

It is desirable to obtain $\mathbf{w}$ and b that maximizes the margin while still separating the data. If the training data are linearly separable, we can select the two hyperplanes of the margin in a way that there are no points between them and then try to maximize their distance. We can find the distance between these two hyperplanes is $\frac{2}{\|\mathbf{w}\|}$, so we need to minimize $\mathbf{w}$. We have to prevent data points falling into the margin, so the algorithm adds the following constraint for both classes:

$$min_i|\mathbf{w} \cdot \mathbf{x}_i + b| = 1 \tag{C.2}$$

A hyperplane then must satisfy the following constraints,

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots n \tag{C.3}$$

Thus, a method called soft margin is introduced. It tries to split the classes as cleanly as possible while still maximizing the margin to the nearest cleanly

split examples by introducing a slack variable, ξ. It is a measure of the degree of misclassification errors. The modified constraint is:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots n \tag{C.4}$$

The way to have a hyperplane that optimally separates the data is by minimizing $\|\mathbf{w}\| + C \sum \xi_i$ (objective function) but it is very difficult to solve. However, it is possible to alter it by solving $\frac{1}{2}\|\mathbf{w}\|^2 + C \sum \xi_i$ instead without changing the solution. The objective function is increased by a function which penalizes non-zero ξ_i, and the optimization becomes a tradeoff between a large margin and a small error penalty. If the penalty function is linear, the optimization problem becomes:

$$\begin{aligned} & min_{\mathbf{w},b} \tfrac{1}{2}\|\mathbf{w}\|^2 + C \textstyle\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i > 0 \end{aligned} \tag{C.5}$$

The above optimization problem can be solved by Lagrangian as:

$$\begin{aligned} \Phi(w, b, \alpha, \xi, \beta) = & \tfrac{1}{2}\|\mathbf{w}\|^2 + C \textstyle\sum_{i=1}^{n} \xi_i \\ & - \textstyle\sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i] - \textstyle\sum_{i=1}^{n} \beta_i \xi_i \end{aligned} \tag{C.6}$$

α_i and β_i are Lagrange multipliers. It is reasonable to solve the problem in dual form. The previous problem can be thought as:

$$max_\alpha W(\alpha, \beta) = max_{\alpha,\beta} \left\{ min_{w,b,\xi} \Phi(w, b, \alpha, \xi, \beta) \right\} \tag{C.7}$$

By taking the partial derivatives of w, b, and ξ. We obtain the following:

$$max_\alpha W(\alpha) = max_\alpha - .5 \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + \sum_{k=1}^{n} a_k \tag{C.8}$$

Hence, the solution to the problem is:

$$\alpha^* = argmin_\alpha \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \sum_{k=1}^{n} a_k \tag{C.9}$$

with constraints,

$$\sum_{j=1}^{n} \alpha_j y_j = 0 \quad and \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots n \tag{C.10}$$

It?s not difficult to generalize this linear case to the nonlinear by replacing $(\mathbf{x}_i \cdot \mathbf{x}_j)$ with a nonlinear function $K(\mathbf{x}_i \cdot \mathbf{x}_j)$:

$$\alpha^* = argmin_\alpha \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i \cdot \mathbf{x}_j) - \sum_{k=1}^{n} a_k \tag{C.11}$$

where $K(\mathbf{x}, \mathbf{x}')$ is the kernel function performing the non-linear mapping into feature space and the constraints are the same as in equation C.10.

Solving equation C.11 with constraints equation C.10 determines the Lagrange multipliers, and a hard classifier implementing the optimal separating hyperplane in the feature space is given by,

$$f(x) = sgn(\sum \alpha_i y_i K(x_i, x) + b) \tag{C.12}$$

where, $(\mathbf{w}^* \cdot \mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x)$, $b^* = -\frac{1}{2} \sum_{i=1}^{n} \alpha_i y_i [K(x_i, x_r) + K(x_i, x_r)]$

There are several kernel functions people use, but we applied linear and radial basis kernel functions in this study:

1 Linear Kernel:

$$K(x_i, x_j) = x_i' \times x_j \tag{C.13}$$

2 Radial Basis Kernel:

$$K(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2), \gamma > 0 \tag{C.14}$$

Radial basis kernel is a tool for measuring similarity of two sample cases. It's quite natural to think that if an observation is more similar to group 1 than group 2, we should classify it into group 1. We used linear kernel to compare the difference between the 2 kernel functions. The concept presented here is based on (11) and (12).

APPENDIX D

Recursive Partitioning and Regression Trees Algorithm

The algorithm first looks for a node A to split two childs into A_L (left child) and A_R (right child) using the following formula:

$$P(A_L)r(A_L) + P(A_R)r(A_R) \leq P(A)r(A) \tag{D.1}$$

where $\mathrm{P(A)} \approx \sum_{i=1}^{C} \pi_i n_{iA}/n_i$,

- π_i is the prior probabilities of each class,
- n_i and n_A are the number of observations in the sample that are class i and node A, respectively,

The idea is to choose a split which decreases risk of $r(A)$. There are some problems with this approach. One way to fix it is to use a measure of impurity of a node A as follows:

$$I(A) = \sum_{i=1}^{C} f(p_{ia}) \tag{D.2}$$

where f is some impurity function and p_{ia} is the proportion of those in A that belong to class i for future sample. These can be thought as lookahead rules. The goal here is to have $I(A) = 0$ when A is pure, so f must be concave function with $f(0) = f(1) = 0$. Then, rpart uses maximal impurity reduction to split node A between information index $f(p) = -plog(p)$ and Gini index $f(p) = p(1-p)$ as follows:

$$\Delta I = p(A)I(A) - p(A_L)I(A_L) - P(A_R)I(A_R) \tag{D.3}$$

It makes sense to incorporate losses in impurity function. The algorithm incorporates losses using altered priors method to extend the impurity function.

Let's first denote risk of A as:

$$R(A) = \sum_{i=1}^{C} p_{ia} L(i, \tau(A)) = \sum_{i=1}^{C} \pi_i L(i, \tau(A))(n_{ia}/n_i)(n/n_a) \tag{D.4}$$

where $R(A)$ is chosen to minimize the risk

- $\tau(A)$ is the class assigned to A as a final node,
- L(i, τ(A)) is the loss matrix for incorrectly classifying an i as a τ(A).

It assumes there exists $\tilde{\pi}$ and $\tilde{L}$

$$\tilde{\pi_i} \tilde{L}(i, j) = \pi_i L(i, j) \quad \forall i, j \in C \tag{D.5}$$

It's clear that $R(A)$ is unchanged under the new losses and priors. If $\tilde{L}$ is proportional to the loss matrix then the priors $\tilde{\pi}$ should be used in the splitting criteria. This is possible only if L is of the form

$$L(i, j) = \begin{cases} L_i & \text{i} \neq \text{j} \\ 0 & \text{i} = \text{j} \end{cases} \tag{D.6}$$

where,

$$\tilde{\pi_i} = \frac{\pi_i L_i}{\sum_{i=j} \pi_j L_j} \tag{D.7}$$

An impurity function $I(A) = \sum f(p_i)$ has its maximum at $p_1 = p_2 = \ldots = p_c = 1/C$. When altered priors are used, they affect only the choice of split. The ordinary losses and priors are used to compute the risk of the node. The altered priors simply help the impurity rule choose splits that are likely to be as good as in terms of the risk. After building the tree, the tree structure is often too complicated and the algorithm then tries to prune the tree.

- Let $T_1, T_2, \ldots, T_k$ be the terminal nodes of a tree T and,
- Let $|T|$ = number of terminal nodes.
- Let $T = R(T) = \sum_{i=1}^{k} P(T_i)R(T_i)$ be the risk of a model.
- Let $\alpha \in [0, \infty]$ which measures the cost of adding another variable to the model. α is call the complexity parameter.
- Let R(T_0) be the risk for the zero split tree

Define,

$$R_\alpha(T) = R(T) + \alpha|T| \tag{D.8}$$

to be the cost for the tree. Let T_α to be that subtree of the full model which has minimal cost. Obviously, T_0 = the full model and T_∞ = the model with no splits at all. The following results are shown in [].

- If T_1 and T_2 are subtrees of T with $R_\alpha(T_1) = R_\alpha(T_2)$, then either T_1 is subtree of T_2 or T_2 is subtree of T_1.
- If $\alpha > \beta$ then either $T_\alpha = T_\beta$ or T_α is a strict subtree of T_β.

The result above suggests that one can uniquely define T_α as the smallest tree T for which R_α(T) is minimized and all possible values of α can be grouped into m intervals, m $< |T|$ as $I_1 = [0, \alpha_1]$, $I_2 = (\alpha_1, \alpha_2]$, ..., $I_m = (\alpha_{m-1}, \infty]$ where all $\alpha \in I_i$ share the same minimizing subtree.

Then, CV is used to choose a best value for α by the following steps:

- Fit the full model on the dataset and compute $I_1, I_2, \ldots, I_m$ and set $\beta_1 = 0$, $\beta_2 = \sqrt{\alpha_1\alpha_2}$, ..., $\beta_{m-1} = \sqrt{\alpha_{m-2}\alpha_{m-1}}$, $\beta_m = \infty$.
- Divide the data set into s groups $G_1, G_2, \ldots, G_s$ each of size s/n, and for each group separately:

  1) fit a full model on the data set everywhere except G_i and determine $T_{\beta 1}, T_{\beta 2}, \ldots, T_{\beta m}$ for this reduce dataset,

2) compute the predicted class for each observation in G_i, under each of the models T_{β_j} for $1 \leq \mathrm{j} \leq \mathrm{m}$,

3) compute the risk for each subject from 2).

– Sum over the G_i to get an estimate o risk for each β_j. For that β with smallest risk compute T_β for the full data set, this is chosen as the best trimmed tree.

We do not discuss rpart's implementation about missing values since it never happens in our case. For more information, please refer to (8).

APPENDIX E

Tables

Table E.1: Number of TFBSs in Each Group

Pair		**Raw Data**		**BD Data**			**Analysis**	
Number	Name	S_1	S_2	S_a	S_b	S_i	Selected?	Analyzed?
1	Oct4Sox2	3761	4526	2134	2547	1331	Yes	Yes
2	Oct4Nanog	3761	10343	2014	7263	1447	Yes	Yes
3	Oct4Esrrb	3761	21647	2569	19079	878	No	No
4	Oct4Stat3	3761	2546	3049	1891	391	Yes	Yes
5	Oct4cMyc	3761	3422	3107	3006	333	Yes	Yes
6	Oct4nMyc	3761	7182	2796	6338	645	Yes	Yes
7	Oct4K1f4	3761	10875	2541	9323	904	Yes	No
8	Oct4E2f1	3761	20699	2154	18002	1306	Yes	Yes
9	Oct4Smad1	3761	1126	2986	542	455	Yes	Yes
10	Oct4Zfx	3761	10338	2902	9470	543	Yes	Yes
11	Oct4Tcfcp211	3761	26910	2362	23213	1097	Yes	Yes
12	Oct4Ctcf	3761	39609	3100	35663	342	Yes	No
13	Oct4p300	3761	524	3280	315	161	No	No
14	Oct4Suz12	3761	4215	3391	4156	46	No	No
15	Sox2Nanog	4526	10343	1643	6478	2259	Yes	Yes
16	Sox2Esrrb	4526	21647	2838	18922	1061	Yes	Yes
17	Sox2Stat3	4526	2546	3443	1864	423	Yes	Yes

Continued on next page...

Table E.1 – Continued from Previous Page

Pair		Raw Data		BD Data			Analysis	
Number	Name	S_1	S_2	S_a	S_b	S_i	Selected?	Analyzed?
18	Sox2cMyc	4526	3422	3730	3209	130	No	No
19	Sox2nMyc	4526	7182	3577	6702	283	No	No
20	Sox2K1f4	4526	10875	3022	9381	852	Yes	No
21	Sox2E2f1	4526	20699	2890	18316	997	Yes	Yes
22	Sox2Smad1	4526	1126	3313	439	562	Yes	Yes
23	Sox2Zfx	4526	10338	3566	9711	306	Yes	Yes
24	Sox2Tcfcp211	4526	26910	2595	23014	1313	Yes	Yes
25	Sox2Ctcf	4526	39609	3597	35738	265	No	No
26	Sox2p300	4526	524	3678	292	184	No	No
27	Sox2Suz12	4526	4215	3837	4181	20	No	No
28	NanogEsrrb	10343	21647	6944	18200	1810	Yes	Yes
29	NanogStat3	10343	2546	8070	1674	617	Yes	Yes
30	NanogcMyc	10343	3422	8492	3150	185	No	No
31	NanognMyc	10343	7182	8268	6575	413	Yes	Yes
32	NanogK1f4	10343	10875	7388	8933	1325	Yes	No
33	NanogE2f1	10343	20699	7262	17865	1472	Yes	Yes
34	NanogSmad1	10343	1126	7938	247	755	No	No
35	NanogZfx	10343	10338	8259	9576	440	Yes	Yes
36	NanogTcfcp211	10343	26910	6742	22304	2039	Yes	Yes
37	NanogCtcf	10343	39609	8410	35731	277	No	No
38	Nanogp300	10343	524	8463	255	220	No	No
39	NanogSuz12	10343	4215	8654	4182	21	No	No
40	EsrrbStat3	21647	2546	19199	1530	756	Yes	Yes
41	EsrrbcMyc	21647	3422	19330	2725	604	Yes	Yes

Continued on next page...

Table E.1 – Continued from Previous Page

Pair		Raw Data		BD Data			Analysis	
Number	Name	S_1	S_2	S_a	S_b	S_i	Selected?	Analyzed?
42	EsrrbnMyc	21647	7182	18649	5691	1291	Yes	No
43	EsrrbK1f4	21647	10875	17422	7709	2561	Yes	No
44	EsrrbE2f1	21647	20699	16890	16186	3129	Yes	Yes
45	EsrrbSmad1	21647	1126	19503	548	450	Yes	Yes
46	EsrrbZfx	21647	10338	18142	8202	1823	Yes	Yes
47	EsrrbTcfcp211	21647	26910	16091	20408	3971	Yes	No
48	EsrrbCtcf	21647	39609	18444	34510	1518	Yes	No
49	Esrrbp300	21647	524	19774	310	168	No	No
50	EsrrbSuz12	21647	4215	19610	3876	324	Yes	No
51	Stat3cMyc	2546	3422	2109	3162	170	No	No
52	Stat3nMyc	2546	7182	1926	6624	360	Yes	Yes
53	Stat3K1f4	2546	10875	1587	9514	706	No	No
54	Stat3E2f1	2546	20699	1380	18358	914	Yes	No
55	Stat3Smad1	2546	1126	2040	752	243	No	No
56	Stat3Zfx	2546	10338	1917	9647	366	Yes	Yes
57	Stat3Tcfcp211	2546	26910	1494	23500	803	Yes	Yes
58	Stat3Ctcf	2546	39609	2100	35824	181	No	No
59	Stat3p300	2546	524	2167	365	112	No	No
60	Stat3Suz12	2546	4215	2269	4195	8	No	No
61	cMycnMyc	3422	7182	788	4437	2547	Yes	No
62	cMycK1f4	3422	10875	2337	9212	995	Yes	No
63	cMycE2f1	3422	20699	836	16765	2506	Yes	Yes
64	cMycSmad1	3422	1126	3308	967	25	No	No
65	cMycZfx	3422	10338	1887	8567	1439	Yes	Yes

Continued on next page...

Table E.1 – Continued from Previous Page

Pair		Raw Data		BD Data			Analysis	
Number	Name	S_1	S_2	S_a	S_b	S_i	Selected?	Analyzed?
66	cMycTcfcp211	3422	26910	2536	23495	801	No	No
67	cMycCtcf	3422	39609	2939	35610	395	Yes	No
68	cMycp300	3422	524	3314	458	17	No	No
69	cMycSuz12	3422	4215	3310	4182	21	No	No
70	nMycK1f4	7182	10875	4876	8104	2107	Yes	No
71	nMycE2f1	7182	20699	2257	14529	4732	Yes	No
72	nMycSmad1	7182	1126	6900	914	79	No	No
73	nMycZfx	7182	10338	4278	7306	2700	Yes	Yes
74	nMycTcfcp211	7182	26910	5271	22578	1716	Yes	Yes
75	nMycCtcf	7182	39609	5955	34976	1036	Yes	No
76	nMycp300	7182	524	6945	441	34	No	No
77	nMycSuz12	7182	4215	6843	4065	135	No	No
78	K1f4E2f1	10875	20699	5889	14922	4366	Yes	No
79	K1f4Smad1	10875	1126	9841	615	380	Yes	No
80	K1f4Zfx	10875	10338	8070	7865	2149	Yes	No
81	K1f4Tcfcp211	10875	26910	7484	21531	2799	Yes	No
82	K1f4Ctcf	10875	39609	9018	34814	1201	Yes	No
83	K1f4p300	10875	524	10068	331	146	No	No
84	K1f4Suz12	10875	4215	10037	4031	170	No	No
85	E2f1Smad1	20699	1126	18930	657	347	Yes	Yes
86	E2f1Zfx	20699	10338	14264	5013	5001	Yes	No
87	E2f1Tcfcp211	20699	26910	15052	20075	4273	Yes	Yes
88	E2f1Ctcf	20699	39609	17565	34297	1727	Yes	No
89	E2f1p300	20699	524	19139	352	126	No	No

Continued on next page...

Table E.1 – Continued from Previous Page

Pair		Raw Data		BD Data			Analysis	
Number	Name	S_1	S_2	S_a	S_b	S_i	Selected?	Analyzed?
90	E2f1Suz12	20699	4215	19159	4097	106	No	No
91	Smad1Zfx	1126	10338	914	9934	79	No	No
92	Smad1Tcfcp211	1126	26910	562	23861	442	Yes	Yes
93	Smad1Ctcf	1126	39609	960	35971	31	No	No
94	Smad1p300	1126	524	855	338	138	No	No
95	Smad1Suz12	1126	4215	987	4199	4	No	No
96	ZfxTcfcp211	10338	26910	8005	22289	2026	Yes	Yes
97	ZfxCtcf	10338	39609	9090	35077	927	Yes	No
98	Zfxp300	10338	524	9972	438	37	No	No
99	ZfxSuz12	10338	4215	9836	4025	174	No	No
100	Tcfcp211Ctcf	26910	39609	22328	34046	1990	Yes	No
101	Tcfcp211p300	26910	524	24139	328	152	No	No
102	Tcfcp211Suz12	26910	4215	23934	3846	355	Yes	No
103	Ctcfp300	39609	524	35984	459	17	No	No
104	CtcfSuz12	39609	4215	35856	4058	142	No	No
105	p300Suz12	524	4215	472	4200	3	No	No

Bibliography

[1] Chen et al. (2008). Integration of External Signaling Pathways with the Core Transcriptional Network in Embryonic Stem Cells. Cell, 133(6):1106-1117, 2008.

[2] Ji et al. (2008). An integrated software system for analyzing ChIP-chip and ChIP-seq data. Nature Biotechnology. Nature Biotechnology, 26(11):1293-1300, 2008.

[3] Mason et al. (2010). Identification of Context-Dependent Motifs by Contrasting ChIP Binding Data. Bioinformatics, 26(22):2826-2832, 2010.

[4] Czepiel et al. (2011). Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation. http://czep.net/stat/mlelr.pdf, 2011

[5] Weisberg et al. (2005). Applied Linear Regression. Wiley Series in Probability and Statistics, 3rd Edition, 2005.

[6] Zhou et al. (2008). Extracting sequence features to predict protein-DNA interactions: a comparative study. Nucleic Acids Research, 36(12):4137-4148, 2008.

[7] Becker et al. penalizedSVM: a R-package for feature selection SVM clasifcation. Bioinformatics, 22(13):1711-1712, 2009.

[8] Therneau et al. (2006) rpart: Recursive Partitioning. http://CRAN.R-project.org/, 2006.

[9] Fan et al. (2001). Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties. Journal of the American Statistical Association, 96(12):1348-1360, 2001.

[10] Venables et al. (2002). Modern Applied Statistics with S. Springer, ISBN 0-387-95457-0, 2002.

[11] Alpaydin et al. (2004). Introduction to Machine Learning. MIT Press, 2004.

[12] Karatzoglou et al. (2006). Support Vector Machines in R. Journal of Statistical Software, 15(9), 2006.

[13] V et al. (2006). TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes. Nucleic Acids Res., 34(Database issue):D108-10, 2006.