Improving Computational Predictions of *Cis*-regulatory Binding Sites in Genomic Data

Faisal Ibne Rezwan
School of Computer Science
University of Hertfordshire

A thesis submitted in partial fulfilment of the requirements of the University of Hertfordshire for the degree of

DOCTOR OF PHILOSOPHY

November 2011

I would like to dedicate this thesis to
Md. Rezwan Ali and Mrs. Samsun Nahar,
my loving parents ...

Acknowledgements

At first I am grateful to the Almighty for giving me the courage, inspiration, and opportunity to finish my PhD. I am deeply indebted to my supervisors, **Neil Davey**, **Rod Adams** and **Yi Sun**. You would not read this thesis without their invaluable support, continual encouragement and willingness to share their vast knowledge and experience from the initial to the final level. I would also like to thank Mark Robinson for all of his supports during the first two-year of my PhD. I am incredibly grateful to Alistair G. Rust for his generosity with his time and effort and not least for his unfailing pertinent feedbacks. I would particularly like to thank Rene teBoekhorst and Reinoud Maex for their advice and timely supports.

I would also like to express my gratitude to my family especially to my wife whose constant care and patience were instrumental in completing this thesis. I am also thankful to my loving niece and sisters for their supports.

In addition to the above, I want to thank The Charles Wallace Trust for giving me the valuable financial support. Lastly, I offer my regards to all of my friends and colleagues who supported me in any respect during the completion of this thesis.

The author was supported by a research studentship from the Biocomputation Group, School of Computer Science, University of Hertfordshire, UK and Overseas Research Student Award Scheme (OSRAS) scholarship.

Abstract

Cis-regulatory elements are the short regions of DNA to which specific regulatory proteins bind and these interactions subsequently influence the level of transcription for associated genes, by inhibiting or enhancing the transcription process. It is known that much of the genetic change underlying morphological evolution takes place in these regions, rather than in the coding regions of genes. Identifying these sites in a genome is a non-trivial problem. Experimental (wet-lab) methods for finding binding sites exist, but all have some limitations regarding their applicability, accuracy, availability or cost. On the other hand computational methods for predicting the position of binding sites are less expensive and faster. Unfortunately, however, these algorithms perform rather poorly, some missing most binding sites and others over-predicting their presence. The aim of this thesis is to develop and improve computational approaches for the prediction of transcription factor binding sites (TFBSs) by integrating the results of computational algorithms and other sources of complementary biological evidence.

Previous related work involved the use of machine learning algorithms for integrating predictions of TFBSs, with particular emphasis on the use of the Support Vector Machine (SVM). This thesis has built upon, extended and considerably improved this earlier work.

Data from two organisms was used here. Firstly the relatively simple genome of yeast was used. In yeast, the binding sites are fairly well characterised and they are normally located near the genes that they regulate. The techniques used on the yeast genome were also tested on the more complex genome of the mouse. It is known that the

regulatory mechanisms of the eukaryotic species, mouse, is considerably more complex and it was therefore interesting to investigate the techniques described here on such an organism.

The initial results were however not particularly encouraging: although a small improvement on the base algorithms could be obtained, the predictions were still of low quality. This was the case for both the yeast and mouse genomes.

However, when the negatively labeled vectors in the training set were changed, a substantial improvement in performance was observed. The first change was to choose regions in the mouse genome a long way (distal) from a gene over 4000 base pairs away - as regions not containing binding sites. This produced a major improvement in performance. The second change was simply to use randomised training vectors, which contained no meaningful biological information, as the negative class. This gave some improvement over the yeast genome, but had a very substantial benefit for the mouse data, considerably improving on the aforementioned distal negative training data. In fact the resulting classifier was finding over 80% of the binding sites in the test set and moreover 80% of the predictions were correct.

The final experiment used an updated version of the yeast dataset, using more state of the art algorithms and more recent TFBSs annotation data. Here it was found that using randomised or distal negative examples once again gave very good results, comparable to the results obtained on the mouse genome. Another source of negative data was tried for this yeast data, namely using vectors taken from intronic regions. Interestingly this gave the best results.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Genes lie at the heart of many complex biological processes in genomes where the interactions between the genes themselves are governed by genetic regulatory networks (GRNs). The translation of a gene into its protein product is relatively simple and well understood (Crick, 1962). However, the quantity of protein generated by a gene within a specific-cell and the change of this protein level over time, a gene's expression profile, is less well understood.

The regulation of gene expression can be controlled by regions of a genome that do not code for genes, simply called non-coding regions. These are large spaces in the genome and it has been estimated that around 50% of the genome of a multi-cellular, eukaryotic organism may have a regulatory function (Markstein *et al.*, 2002).

Specifically, non-coding regions in the vicinity of genes may contain short stretches of DNA sub-sequences to which proteins can bind. These regions are known as *cis*-regulatory binding sites or transcription factor binding sites (TFBSs) and are known to finely regulate gene expression (Arnone & Davidson, 1997; Davidson, 2001). The composition and number of *cis*-regulatory binding sites across multiple non-coding regions give rise to a complex set of GRNs that encode the regulatory program of a cell. There are very few biological processes that are not influenced by regulatory mechanisms.

Deciphering these non-coding regions is significant therefore not only to understand their functional association with gene coding sequences, but also for discovering the regulatory instructions they specify. Hence studying GRNs have been major focuses of studies in the fields of biology and bioinformatics. Over recent years the opportunities to study gene regulation has increased markedly as with the advent of high-throughput experiments from next-generation sequencing technologies, there is now an unprecedented abundance of genomic data available from a large number of publicly accessible databases.

The practical benefits of studying regulatory systems originating from this research are many-fold, including cancer, cell cycle and disease research. The insights gained from this area are also beneficial to pharmaceutical companies and medical researchers, including determining the regulatory regions of a gene that may act as drug targets, predicting the responsiveness of biological pathways to treatment and identifying potential side effects during a drug's development process. Therefore, characterising regulatory systems by identifying their components will have a wide-ranging impact in many other research fields.

1.1 The Importance of Protein-DNA Interaction

Every cell in a living organism contains a genetic blueprint that governs the biological structures and processes for that organism. Gene expression is a key process that uses this information embedded in DNA to produce non-coding RNA or proteins. Figure 1.1 shows the different regulatory control mechanisms in a cells protein synthesis pathway. While all the mechanisms play important roles in regulating transcription, transcriptional control is the main interest here as it is the initial step of the protein synthesis process.

Transcription is a fundamental process that determines a cells morphological and functional attributes. Transcriptional control is achieved via two mechanisms; one by chromatin remodelling and the other by transcription factor activity. Chromatin remodelling acts by inhibiting access to large stretches of a genome to transcription factors, ultimately affecting gene expression (Alberts *et al.*, 1994). This mechanism is far less specific than the interaction of transcription factors and other regulatory proteins with DNA sequences. The later mechanism can however directly influence the level at which a specific gene is expressed.

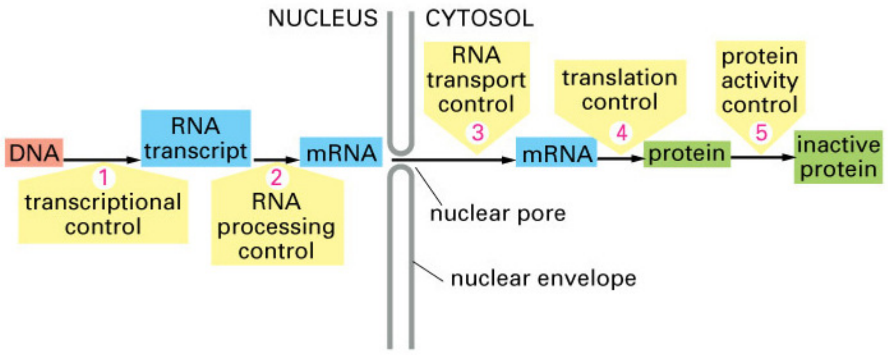


Figure 1.1: Regulation of gene expression along the protein synthesis pathways (*Source: Essential Cell Biology, 2/e (2004 Garland Science)*).

It is through the combinatorial effects of transcription factors binding to spatially localised locations neighbouring genes that are sufficient for transcriptional control in many systems (Arnone & Davidson, 1997; Davidson, 2001; Yuh *et al.*, 1998; Ptashne & Gann, 2002; Davidson, 1999). The location of these binding sites within a genome determines the basic connectivity of an organisms GRNs and ultimately how genes interact. Therefore, identifying these binding sites is an interesting problem both in biology and bioinformatics.

1.2 Limitations of Experimental and Computational Approaches to Identify TFBSs

Transcription factors bind to specific DNA regions, or sites, typically 5-30 base pairs in length and where the DNA composition of each binding site is specific to the factor that binds there. Identifying binding sites across a variety of genomes is a complex task and there are ongoing, complementary approaches from both the experimental (wet-lab) and computational research fields.

There are many experimental approaches for identifying *cis*-regulatory sites, all of which have some limitations regarding their applicability, accuracy, availability and cost. A number of experimental techniques can identify whether a protein binds to particular stretches of DNA and whether these interactions have any regulatory properties. Among them *in vitro* oligo selection (Pollock & Treisman, 1990) and gel-shift assays (Taylor *et al.*, 1994) were often used to determine the DNA binding properties of a protein. But these two techniques often yield poor predictions of *in vivo* binding targets (Lieb *et al.*, 2001; Buck & Lieb, 2004). Once the interaction between DNA and a protein is reliably established, a number of *in vitro* techniques (such as DNA footprinting or chromatin immunoprecipitation, also known as ChIP) can be used to narrow down the search. However, these approaches are not always suitable for genome-wide analysis. Recently the combination of ChIP and whole-genome microarray (ChIP-chip) has reduced these limitations (Ren *et al.*, 2000; Buck & Lieb, 2004; Cawley *et al.*, 2004; Ren & Dynlacht, 2004) and has become the most successful technique to identify *in vivo* binding sites genome-wide. But these types of experimental techniques are costly and in many cases, time consuming (Tompa *et al.*, 2005; Brown *et al.*, 2002). Further, experimental approaches such as ChIP-chip and ChIP-seq, are themselves dependent on the availability of specific antibodies for the binding proteins they aim to analyse and still require additional verification.

From a computational standpoint, there are a number of algorithmic strategies for computationally predicting the location of TFBSs. One class of algorithms search for matches to a motif model of TFBSs (Stormo, 2000) (such as, a position weight matrix) in a given sequence. Another class of algorithms, which extract statistical characteristics of sequence features, requires nothing more than sequence information. However, these algorithms typically exhibit low accuracy with a high level of false positive predictions. Other algorithms try to improve on this statistical approach by searching for common sub-sequences in a set of DNA sequences, where the set of sequences is determined by clustering together genes that share similar patterns of expression for a given biological condition. Another class of algorithms exploits the evolutionary conservation of DNA sequences to infer TFBSs.

Moreover, there are many reasons why binding site prediction is a difficult problem to solve computationally. The binding sites can be very short and of variable sizes. Furthermore a typical transcription factor can bind to a number of different DNA sequences, albeit with very similar DNA composition. These are just some reasons why identifying binding sites computationally is a non-trivial problem rather than a simple pattern recognition or regular expression problem.

All of these algorithmic approaches have their own strengths and weaknesses. One problem is that they can be sensitive to the number of sequences being analysed, sequence length and motif width. It has been shown that some of the algorithms performance decreases with the increase in one of these factors (Hu *et al.*, 2005). Furthermore, introducing more data (sequences) does not necessarily improve the performance but may decrease the accuracy. Also these algorithms often require a number of parameters to be fine-tuned to achieve the greatest accuracy. For a single algorithm for example the set of optimal parameters may vary depending on the dataset. Due to these shortfalls, experimental biologists have to date found little practical use for these algorithms

1.3 Combining Sources of Computational and Biological Data

As described in the previous section, a large number of computational algorithms are available for the prediction of transcription factor binding sites and these can be grouped into a small number of prediction strategies, such as: scanning, statistical, co-regulatory and phylogenetic. Each of these strategies has their own limitations giving rise to many false positives predictions, which can significantly limit their utility. However, there is still scope for improvement by reducing their weaknesses and combining their strengths together. For example, scanning algorithms can only predict those sites that match to the known set of motifs they are supplied with, whereas co-regulatory algorithms can only predict shared, and not unique, binding sites in the regulatory sequences for a set of co-expressed genes. Hence the different subsets of accurately predicted binding sites are complementary and can provide more accurate and reliable predictions if combined together (Tompa *et al.*, 2005).

In earlier works (Sun *et al.*, 2005, 2006a,b, 2007, 2008, 2009a,b; Robinson *et al.*, 2006, 2007a,b, 2008), results from grouping different TFBS prediction algorithms together produced predictions that are better than that of any of the individual predictions alone. In one approach, algorithms and supporting biological data have been classified using Support Vector Machines (SVM). In this thesis, the same approach of integrating multiple sources of evidence has been undertaken. One of the major problems with training a classifier on these combined sources of evidence is that the data constituting the training set can be contradictory and unreliable. Whereas the labelling of a known binding site is normally correct, as it has been experimentally verified, the labelling of the non-binding site, the negative example class, may be much more unreliable. This is the one of the major issues addressed in this thesis.

1.4 Aims and Objectives

This research developed out of the research undertaken by Mark Robinson. Mark Robinson pioneered the approach of integrating different predictors or sources of evidence using an SVM (Robinson, 2006). This integration approach is therefore not novel to this thesis. Robinson proved that attempting to optimise the parameters of the individual predictors prior to combining them in an SVM was of no benefit. The use of default parameters for the predictors was just as good as optimised ones so in this thesis all predictors are used with their default (or commonly used) parameter values.

Robinson, however, only applied his technique to yeast so extending the methods to another organism, mouse, was central to the work to be undertaken for this thesis. However, for the purpose of investigating various methods of improving the results of using an SVM, such as using a one-class SVM and changing the method for finding the appropriate SVM parameters (using cross validation), I initially used the same yeast dataset and predictors as was used by Robinson. Once it became clear that improvements were possible, the mouse genome was used and finally the old yeast data was replaced by more up-to-date yeast data and more up-to-date prediction algorithms in order to verify whether or not the results were generalisable.

Hence the general aims and objectives of this thesis are to develop, investigate and evaluate computational methods to improve the accuracy of the prediction of transcription factor binding sites. Broadly we can divide the objectives into four different parts:

i. The integration of algorithmic predictions and biological evidence to improve transcription factor binding site predictions.

ii. Using a non-linear classifier with a suitable cross-validation method on the combination of algorithmic predictions.

iii. An extension of the computational approach to other organisms (mouse and updated yeast) to investigate the efficacy of the process.

iv. An evaluation of the effectiveness of using negative examples from different sources for improving the accuracy of algorithmic prediction by the process of integration.

1.5 Contribution to Knowledge

The research conducted during my PhD has made the following contributions to the fields of bioinformatics:

- In previous studies, yeast was used as the experimental organism on which to test the computational predictions. In this thesis, I have extended the work to mouse, which has a more complex regulatory system. This proved that the process of the integration of different prediction algorithms to improve *cis*-binding site prediction is not species specific; rather that it can potentially be used on different species.

- The standard cross-validation method used in other classifier approaches was found not to be suitable for classifying TFBSs predictions due to the imbalance of the data. A modified cross-validation method was thereby devised in this thesis. I have introduced a filtering method during the new modified cross-validation method. A wide range of parameters has been searched during the course of this research.

- As already mentioned, we can be reasonably confident about the annotation of transcription factor binding sites (positive examples) that have been experimentally verified. However, we cannot draw the same conclusion about other unverified sites of the genome, the source of negative examples. Therefore, the most interesting and important finding of this thesis is the effect that the different types of negative examples have on the accuracy of predictions. See Chapter 7 for these major results.

- The consistency of the improvement in predicting binding sites largely depends upon the quality of the available data. The yeast data along with the algorithms for integration process that had previously been used were quite old and out of date. Therefore, to improve the prediction further, I have used an updated version of yeast data and more current algorithms (See Chapter 8).

1.6 Structure of the Thesis

A brief outline of the organisational structure of this thesis is now given.

Chapter 2 reviews the background biological knowledge that is related to this thesis. It provides a detailed description of transcription and transcription factors in eukaryotes and the state of knowledge regarding the experimental approaches for identifying *cis*-binding sites. The chapter also provides a brief description of gene regulatory networks.

Chapter 3 provides a literature review of computational approaches of identifying transcription factor binding sites, along with various models for their accurate representation within computational frameworks.

Chapter 4 presents a review of the meta-classifier, the Support Vector Machines (SVM). In this chapter, I describe the different strategies such as two-class and one-class SVMs. I have given a brief description of how to accommodate biological imbalanced data for an SVM. This chapter also contains discussion on cross-validation strategies and methods to measure classification performances.

Chapter 5 then gives a description of the datasets and sources of evidence used in this research. A description and justification of the statistical measures used to evaluate performances of these algorithmic strategies is given, which are used to evaluate the research strategies undertaken in the following chapters.

Chapters 6 through 8 present the research undertaken in this thesis.

Chapter 6 provides the work undertaken to integrate the results from different algorithms and biological evidences and the use of meta-classifiers (as described in Chapter 4). This chapter introduces a new modified cross-validation method and shows the efficacy of using the new method on the integration process. However the results produced up to this point are disappointing. Chapters 7 and 8 contain the major results from this thesis and finally prove that the method described in this thesis produce exceptionally good result.

Chapter 7 reports the results of varying negative examples to improve transcription factor binding site prediction. The chapter also presents the comparison of results from using different negative examples. It also provides the results from a selection of three different sets of training and test data from the same dataset.

Chapter 8 then gives a brief description of a new yeast dataset with improved annotations and updated algorithms/ biological evidences. All the experiments conducted in Chapter 7 will be repeated in this chapter and thus this chapter presents the effect of using varying negative examples on the new yeast data.

Chapter 9 finally summarises the conclusions of each chapter. In addition, a number of suggestions are offered for extending this research.

Chapter 2

A Review of Gene Regulation and *cis*-regulatory Binding Sites

2.1 Introduction

Nature is full of different species containing a vast spectrum of different body shapes and sizes. The differences between species can be described as the variation of the same programme of development that reinitiates again and again. Initially most embryos start as a cluster of nearly identical cells. The embryo then begins the process of partitioning itself into different segments, which results in the final form of the organism. In a nutshell, organisms use similar sets of core genes. This raises the question, if organisms use similar genes, and if genes determine body shape, then why are there differences between species? The answer is that genes are not the only factors that determine body shape. Rather, body shape is the result of the interplay between genes, cells and the surrounding environment in which the organisms exists. It has been found that a small set of genes, a genetic toolkit, lays out the construction of the whole body (Carroll *et al.*, 2005). These toolkits are like genetic switches- either turning a gene on or off. One startling observation is that sets of these genetic toolkits are conserved across species, be they in a fly, a mouse or a human.

Genes are connected to each other in networks, interacting and regulating their functions. These networks of genetic switches are called Gene Regulatory

Networks (GRN) (Alberts *et al.*, 1994). Each cell within an organism is grown under the control of these networks leading to the variety of body plans observed in nature. The products of some of these genes work as transcription factors that start the process of RNA (Ribonucleic acid) synthesis and modulate the expression of other genes. Therefore, gene expression plays a vital role in organism development.

Gene expression is the fundamental level where genotype affirms phenotype of an organism. It drives every stage of development of the body-plan giving rise to cell differentiation and morphogenesis. As a result the same cell organisation can produce different parts of an organism. The coordination of specific gene regulation events during commitment of stem cells and the appropriate control of gene expression in differentiated cell are important for the development and function of all organisms. Inappropriate gene expression may give rise to detrimental and lethal phenotypes. Chemically induced changes in gene regulation are associated with serious and complex human diseases, such as Alzheimers, hypertrophy, cancer, etc. Therefore, understanding gene expression is not only important for developmental biology but also for drug discovery and their potential therapeutic side effects.

The aim of this research project is to use computational methods to identify the components of genetic regulatory networks that drive gene expression. As eukaryotes have been used as the experimental organisms throughout this thesis, the basic mechanisms of gene expression and gene regulation in eukaryotes will be reviewed in this chapter.

2.2 Gene Expression and Regulation

Every cell in living organisms contains a complete set of genes that define how each cell develops and functions. But not all genes are expressed at the same time; only a small subset of genes is expressed at any one time. There are different stages of gene expression:

1. Modification of DNA (Deoxyribonucleic acid) chemically or structurally, which may alter the accessibility of large regions of DNA for binding proteins.

2. Conversion of DNA to RNA known as transcription.

3. Post-transcriptional regulations such as: capping, splicing, and the addition of a polyA tail, etc. (Berg *et al.*, 2007)

4. Conversion of RNA to protein known as translation.

5. Degradation by different factors that affect mRNA lifetime, thereby dynamically reducing the amount of protein in a cell.

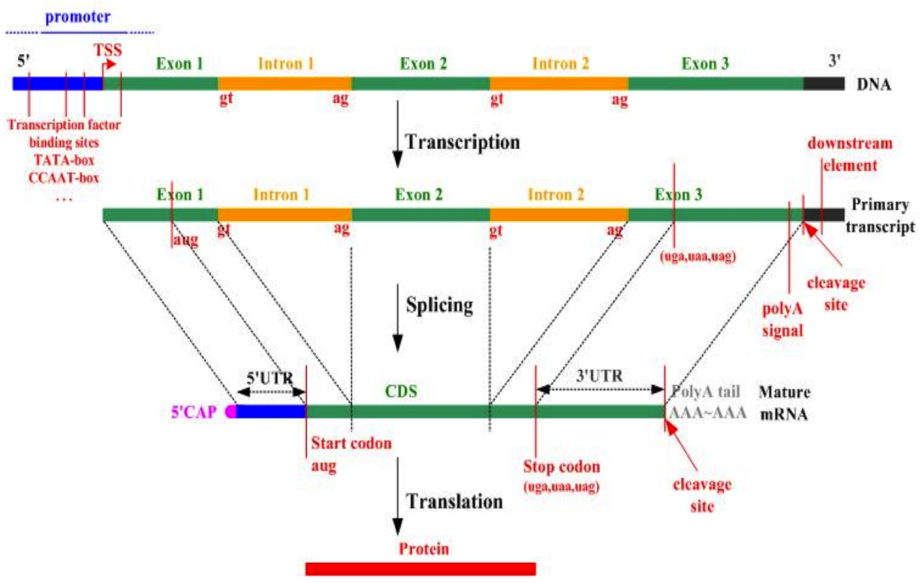


Figure 2.1: The different stages of gene expression in a eukaryote (This figure is created by BioDiscovery Group, LIT, Singapore).

Transcription is the second stage where stretches of DNA are transcribed into RNA. Gene regulation also occurs at the level of transcription. A set of proteins attached to a certain region of the DNA signals the start of transcription. These specific proteins called *transcription factors* (TFs) control gene regulation, which in turn controls the levels of gene expression. The fourth stage, translation, synthesises proteins from these RNAs. Both transcription and translation are important processes taking place in cells.

Figure 2.1 shows a simplified view of gene expression in eukaryotes, where genes are composed of exons (segment of gene containing information for protein coding) and introns (segment of gene does not contain any information for protein coding). A primary transcript is produced through a process of transcription and then introns are removed from the transcripts to produce mature transcripts (mRNA). This mRNA then produces a group of amino acids or proteins.

Understanding transcription is essential as the proteins that take part in transcription are also a part of the transcription process and these form gene regulatory network (discussed in Section 2.6). In the transcription process we are mainly interested in the initiation of gene expression, in which the regulatory protein binds to a specific region of DNA or site and initiates transcription. Therefore, our primary focus will be on the initiation of the transcription process rather than on the subsequent stages of gene expression.

2.3 Transcription Factors and *cis*-regulatory Sites

Transcription factors play an essential part in the process of transcription as mediation of transcription factors increases rates of transcription significantly. They are typically proteins that bind to DNA, preparing a gene for transcription. There are different types of transcription factors that can broadly be divided into three different classes (shown in Figure 2.2) according to their mechanism of action, regulatory function and structural similarity. The mechanistic class comprises general transcription factors and upstream transcription factors. General transcription factors form the pre-initiation complex for transcription (Orphanides *et al.*, 1996) and upstream transcription factors enhance or repress the transcription process (Boron, 2005). The functional transcription factors can be divided in two classes namely constitutively active and conditionally active (Brivanlou & Darnell, 2002). Constitutively active factors are continuously present in all cells and act as activators of transcription. In comparison, conditionally active transcription factors depend on external signals. These signals can be generated from other regulated transcriptions. Transcription factors can also be classified by their tertiary structure (Stegmaier *et al.*, 2004). These include basic-helix-loop-helix, zinc coordinate DNA binding domain, helix-turn-helix, beta-scaffold factors with minor groove contacts, etc.

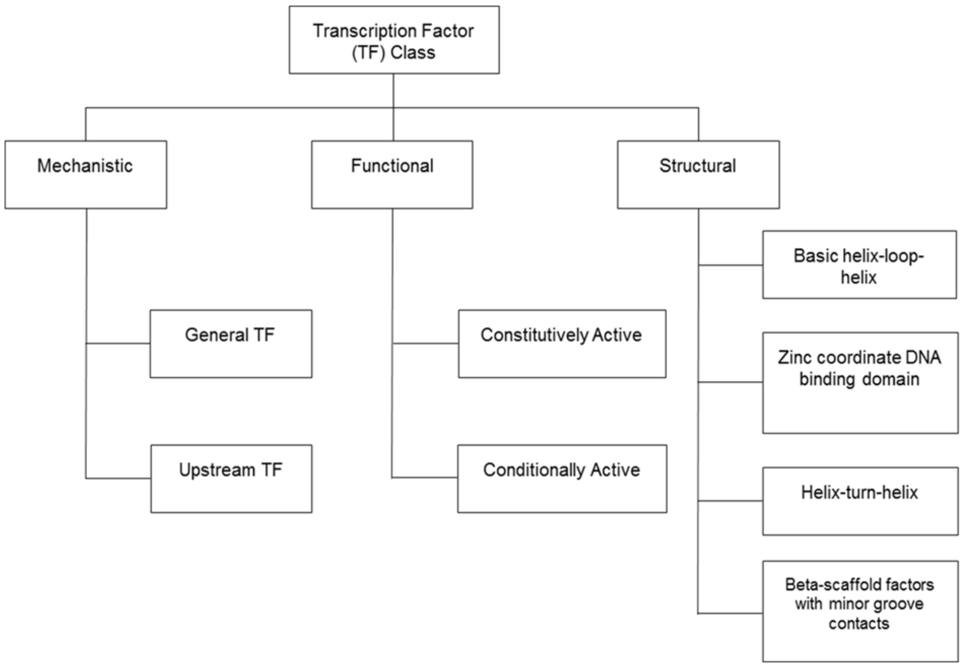


Figure 2.2: Classification of transcription factors.

2.3.1 Transcription factors in eukaryotes

The transcription process in eukaryotes seems to be quite straight forward, but it varies between different eukaryotic organisms. This is in part due to RNA polymerase (a type of enzyme that produces RNA), which performs different actions in an organism during the transcription process. For example, RNA polymerase I produces larger ribosomal RNA (rRNA), RNA polymerase II produces messenger RNA (mRNA) and most of the small nuclear RNAs (snRNA), and RNA poymerase III produce transfer RNAs (tRNA) and small ribosomal RNA (small rRNA). Some organisms use RNA polymerase I to transcribe DNA to rRNAs while others use several RNA polymerases for transcription, making eukaryotic transcription more complex (White, 2000).

However, the recruitment of RNA polymerase with promoters results in very low basal transcription rate and as mentioned above mediation of transcription factors can increase the transcription rate significantly(Ptashne & Gann, 2002). In eukaryotes, there is a common set of proteins that bind to the promoters (an upstream region) of most genes, known as general transcription factors. These transcription factors comprise of basal transcription factors and TATA binding proteins (TBP). This complex of transcription factors and binding proteins (see Figure 2.3) recruits the RNA polymerase II enzyme to the promoter and collectively they initiate transcription.

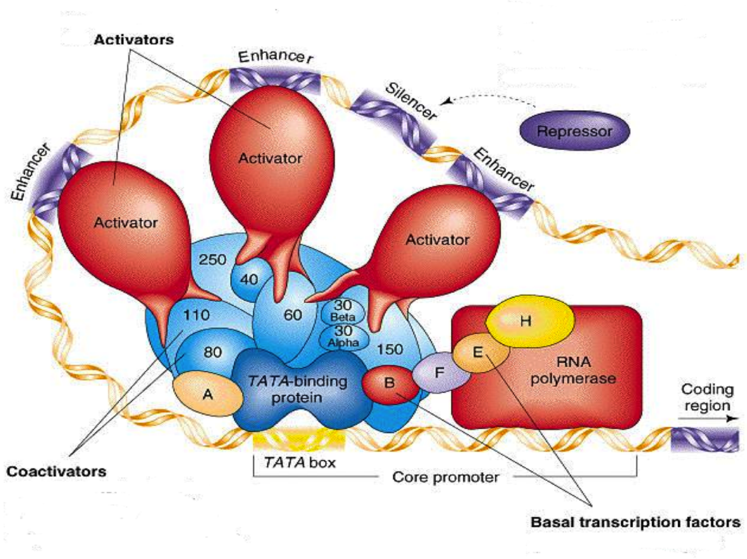


Figure 2.3: Transcription factors involved in eukaryotic transcription initiation (*Source: modified version from Tjian (1995)*) .

Eukaryotic promoters contain a specific conserved sequence, which is essential for transcription initiation. This is generally known as the TATA box (sometimes GC box or CCAAT box) (Lifton *et al.*, 1978; Goldberg, 1979) and they are found typically around 40 - 120 base pairs upstream of genes transcription start site (Struhl, 1995). A specific protein called the TATA binding protein binds to this site and forms a complex with a group of other basal transcription factors namely TFIIA, TFIIB, TFIIE, TFIIF, TFIIH. Generally the complex of TBP and its associate binding proteins is known as TFIID.

There is a second class of binding proteins called co-activators, which are actually TBP-associated factors (TAFs). Different combinations of TAFs and TBP bind to different promoters and activate them with different strengths. There are other factors, which enhance the transcription rate from normal, basal level to enhanced level. These types of factors are called activators.

Another class of factors is enhancers or silencers. Enhancers (or silencers) are short stretches of DNA sequences that are hundreds of base pairs upstream or downstream of the transcription initiation site (see Figure 2.3). Transcription factors bind to these sites to control the level of transcription. Activators bind to the enhancers and determine which gene to turn on for transcription and can also speed the rate of transcription. Alternatively, repressors bind to silencer sequences that disrupt the function of activators and thus slow transcription. Enhancers and silencers are also known as *cis*-regulatory sites.

2.3.2 The organisation of *cis*-regulatory sites

For the transcription process to be initiated, typically a set of transcription factors needs to bind to specific sites of DNA, the previously mentioned *cis*-regulatory sites. These are normally small stretches of nucleotides of variable lengths ranging from 5 to 30 base pairs. Unfortunately locating the binding site(s) for a particular transcription factor is difficult. They maybe upstream or downstream of the genes transcription start site and further may be located thousands of base pairs away from it. Moreover a specific regulatory protein may have many sites that it binds too in the genome, but these sites may not have a common consensus pattern of DNA to which it binds.

The organisation of *cis*-regulatory binding sites in eukaryotes is complex. These *cis*-regulatory binding sites responsible for regulating certain expression patterns can form spatially localised clusters along the DNA. The position of these clusters or modules may not necessarily be near the promoter of a gene. Rather they can be located a significant distance from the promoter of the regulated gene (Alberts *et al.*, 1994; Yuh & Davidson, 1996). The combinatorial nature of many of the DNA binding interactions is mainly responsible for the diversity and complexity observed in eukaryotes.

2.3.3 The number of transcription factors across different eukaryotes

In general terms, the more complex an organism is, the greater the size of genome it possesses. This is also reflected in the number of protein coding genes encoded in the genome, with complex species such as human and mouse, in general having more genes than simpler species, such as fruit fly and yeast. A summary of supporting statistics for five species is presented in Table 2.1.

To appreciate the complexity of identifying transcription factor binding sites and regulatory regions in general, at least two genomic properties need to be considered. Firstly, the potential regions of genomic DNA to which transcription factors can bind to can be large. At one extreme, if all non-coding regions (i.e. non-exonic) are considered potential regulatory regions, then in the case of humans this could be over 2.9G bps of DNA sequence (Table 2.1(d)). Even in simpler species such as fruit fly and worm, this equates to more than 67% of their genomes as being potential regulatory regions. At the other extreme, restricting the potential search space to just the upstream regions of genes (Table 2.1(e)), still require the analysis of tens to hundreds of millions of base pairs. Secondly, the number of genes that are thought to function as transcription factor binding sites increases with organism complexity (Table 2.1(f)). Taken together, even in simple organisms the interplay between the number of estimated transcription factors and the potential genomic regions to which they can bind, results in a large and complex computational search space when trying to predict transcription factor binding sites.

Rows (f) and (g) in Table 2.1 also highlight the gap between the actual estimated number of transcription factors that are thought to exist and well characterised transcription factors that are curated in publicly-available resources, in this case *ORegAnno* (Montgomery *et al.*, 2006). For simpler species the gap between known and curated transcription factors is narrow, for example in yeast, but for human and mouse, less than 10% of transcription factors have been sufficiently characterised to be included in this snapshot of the *ORegAnno* database. Experimental methods, reviewed in Section 2.5 are beginning to close these gaps but computational prediction methods can also provide complementary analyses.

	Human	Mouse	Fruit fly	Worm	Yeast
(a)Genome length Golden Path (bps)	3,101,804,739	2,716,965,481	168,736,537	100,286,070	12,162,995
(b)Known protein-coding genes	20,930	21,637	13,781	20,389	6,696
(c)Total length of coding regions encoded by exons (bps)	154,618,343 (5.0%)	112,973,327 (4.2%)	34,102,698 (20.2%)	32,278,848 (32.2%)	9,056,064 (74.5%)
(d)Maximum length of non-coding regions (bps)	2,947,186,396 (95.0%)	2,603,992,154 (95.8%)	134,633,839 (79.8%)	68,007,222 (67.8%)	3,106,931 (25.5%)
(e) Estimated length of potential regulatory regions (bps)	104,650,000 (3.4%)	108,185,000 (4.0%)	27,562,000 (16.3%)	40,778,000 (40.7%)	13,392,000 (¿100%)
(f) Estimated number of transcription factor genes	1508 (7.2%)	1426 (6.6%)	601 (4.4%)	698 (3.4%)	172 (2.3%)
(g) Transcription factor genes curated in ORegAnno	134	95	103	22	119

Table 2.1: A summary of genomic and transcription factor statistics for five species. Version 62 of the Ensembl genome project was used to collect genomic statistics for (a), (b) and (c). The total length of exons in (c) was calculated by summing the length of each unique exon in the specific genome using their predicted start and end genomic coordinates. (e) is an over-simplified estimate of potential regulatory regions, assuming that each protein-coding gene has an upstream, promoter region that transcription factors can bind to. In human and mouse the promoter region was defined as 5K bps and was 2Kbps in the other three species. (f) The estimated numbers of transcription factors were taken from *DBD: Transcription factor prediction database*. (g) summarises the number of transcription factors curated in the publicly available *ORegAnno* regulatory element database.

2.4 Identification of *cis*-regulatory Regions and Binding Sites Using Biological Cues

Cis-regulatory sites have some biological properties that can be used to identify their locations. Sequence conservation is one such property. Short stretches of nucleotides that are conserved across the non-coding DNA across different species can indicate that they may be functional. One way of searching for sequence conservation can be undertaken by sequence alignment. There are two types of sequence alignments, global and local, which are typically carried out in a pair-wise fashion. Global alignment aligns every residue (nucleotides) in every sequence among equal size DNA sequences. The conventional global alignment algorithm (for example: Needleman-Wunsch algorithm (Needleman & Wunsch, 1970)) uses a dynamic programming approach. This kind of approach is mainly suitable for highly similar but small regions of homologous DNA. Alternatively, local alignment finds regions of similarity or similar sequence motifs within larger sequences. Local alignment finds the local regions with highest similarities regardless of the rest of the sequence. One advantage of local alignment over global alignment is that, local similarities may indicate a functional module, (for example: transcription factor binding sites) within the sequences.

If more than two sequences are to be aligned then multiple sequence alignment is necessary. Multiple sequence alignment is an alignment technique where three or more relevant sequences are aligned together. This is useful to find the evolutionary relationship between the sequences and therefore determine their evolutionary origin. Dynamic programming has been used for multiple sequence alignment incorporating gap penalties and substitution matrices. This can be done, using the Carrillo-Lipman algorithm (Carrillo & Lipman, 1988). In this algorithm, pair-wise alignments are created between different sequences and an optimisation of the sum of the pair score at the position of the alignment is done to get an optimum alignment. The progressive technique (also known as the hierarchical or tree method) is another technique for multiple sequence alignment. In this method, a pair-wise alignment is performed on similar pairs and extends the alignment to more distantly related sequences (Mount, 2004).

CLUSTALW (Chenna *et al.*, 2003; Larkin *et al.*, 2007) is the most popular multiple sequence alignment program that uses the progressive technique. One problem with the progressive method is that it cannot find the globally optimum alignment. This can be achieved by the iterative method, which repeatedly realigns the initial sequences and then add new sequences to the growing multiple sequence alignment. In addition, Hidden Markov Model and genetic algorithms are also used for multiple sequence alignment.

Histone modification is another way to identify *cis*-regulatory sites. Histone modifications occur primarily within the histone amino-terminal tails protruding from the surface of the nucleosome as well as on the globular core region (Cosgrove *et al.*, 2004). This can lead to two mechanisms, which may affect chromosomal function. One of these is the alteration of the electrostatic charge of the histone that leads to structural changes of histones or their binding to DNA. Another mechanism is that these modifications result in binding sites for the protein recognition module. This is known as the Histone Code hypothesis proposed by Strahl & Allis (2000). According to this hypothesis, a modification (for example, methylation) at the unstructured tail of histone proteins can be correlated with transcriptional activities. Therefore, if there is any kind of modification is present in histone proteins, there is a possibility of transcription at that site.

DNA methylation is another mechanism, which causes DNA modification. In DNA methylation, a methyl group is added to either cytosine or adenine at 5´ position. Methylation generally occurs in CpG islands, which are rich in CG content. Normally these CpG islands are found upstream of promoter regions. DNA methylation mainly represses the initiation of transcription by directly binding with the transcriptional activators or indirectly by binding with the proteins (Weber *et al.*, 1990; Razin, 1998; Ng & Bird, 1999). So any region of DNA in which DNA methylation occurs is less likely to have binding sites and therefore can be less significant in determining the location of binding sites.

2.5 Identification of *cis*-Regulatory Sites Experimentally

Experimental approaches to identify transcription factor binding sites are important to understand their biological functions, the complexity of tissue-specific interactions and the temporal effects that binding has on gene expression (Levine & Tjian, 2003). There are two types of approaches currently available (i) when the regulatory proteins involved are not known and (ii) when the transcription factor that binds to a specific DNA sequence has been putatively identified. In the first case, analysis of the alteration of chromatin structure and experimental manipulation of specific DNA segment are carried out. For the latter case, protein-DNA interactions are directly measured.

DNase hypersensitivity maps the changes in chromatin structure of DNA. The degree of response that DNA gives to DNase is known as hypersensitivity and this is present in all actively expressed genes (Elnitski *et al.*, 2006). Hypersensitivity actually acts as a marker for functional regions in non-coding sequences enabling the detection of promoters, enhancers, silencers, etc (Cereghini *et al.*, 1984; Gross & Garrard, 1988). DNaseI footprinting (Galas & Schmitz, 1978) can precisely identify the localisation of protein binding sites without prior knowledge of the binding preferences of the protein. Promoter analysis is another experimental method, which can be used if the regulatory protein involved is not known. In this experimental method a gene expression assay measures changes in response to regulatory signal.

Other approaches to identify *cis*-regulatory sites experimentally include using protein-binding assays. Electrophoretic mobility shift assays (EMSA) (Fried & Crothers, 1981; Garner & Revzin, 1981) were one of the earliest methods that utilises the screening technique of nondenaturing polyacrylamide gels for the separation of protein-bound DNA from other non-binding DNA.

Currently ChIP (Chromatin immunoprecipitation) assay techniques are more popular than the traditional techniques described above. ChIP is an immunoprecipitation method that determines the location of the binding sites of a particular protein of interest to DNA in a genome (Aparicio *et al.*, 2004). In this process DNA binding proteins are cross-linked with formaldehyde, then the regions of the

DNA where the protein has bound to, are isolated by shearing the DNA along with the binding protein into small fragments. The binding protein is then bound to antibodies to isolate the complex by precipitation and a reverse process releases the DNA fragments. PCR (Plolymerase Chain Reaction) is used to amplify the DNA sequences.

ChIP-seq technology is based upon both the ChIP method summarised above and incorporates subsequent gene sequencing (Jothi *et al.*, 2008). The small fragment of DNA attached to a particular protein acts as an oligonucleotide adapter to enable massive parallel sequencing of the ChIP-DNA fragments using a genome sequencer. One other technique is ChIP-on-chip technology, where the single stranded DNA fragments are labeled with fluorescent tags after amplifying the DNA (Buck & Lieb, 2004). These coloured DNA fragments are then hybridised over the surface of a DNA-microarray and the array is then further analysed for identifying the binding sites of regulatory factors. Pair-Ended Tags (PET) are short DNA or cDNA fragments that map to the genome and thus represent the whole DNA fragment of interest. ChIP-PET combines the ChIP and PET technologies together and this is another option to identify binding sites (Fullwood *et al.*, 2009).

All of these technologies for identifying binding sites are quite successful, but there are a number of factors that should be taken into consideration. For example, tissue ChIP assays need sufficient source tissue samples and it is not feasible when those tissues are rare (Elnitski *et al.*, 2006). Experimental conditions and the quality of reagent also affect the results in these experimental techniques. Also if the binding affinity or strength with which a transcription factor binds to DNA is weak then it can be difficult to obtain reliable results. Moreover, though the brute force techniques generate impressive results in identifying transcription factor binding sites, they are also costly and time consuming. Some of the techniques also need processing of the raw data. For example, ChIP-chip raw data needs to be processed to find the best binding sites among a collection of DNA targets and in this case, a number of statistical approaches have been used (Buck & Lieb, 2004; Lieb *et al.*, 2001; Ren *et al.*, 2000). Computational approaches can be used to fine-tune the identification of *cis*-regulatory elements experimentally. Though they are still in their preliminary stages, these have opened a new dimension of research. Some of these computational approaches will be discussed in the next chapter.

2.6 Gene Regulatory Networks

The regulatory relationships between genes and their products are interlinked, where the products of the expression of one gene can act as a regulatory factor for another and thus form a network. This type of network is known as a Gene Regulatory Network (GRN). GRNs dynamically control the level of expression for each gene in the genome by controlling whether and how that gene will be transcribed into RNA, determining the functional role of the produced proteins.

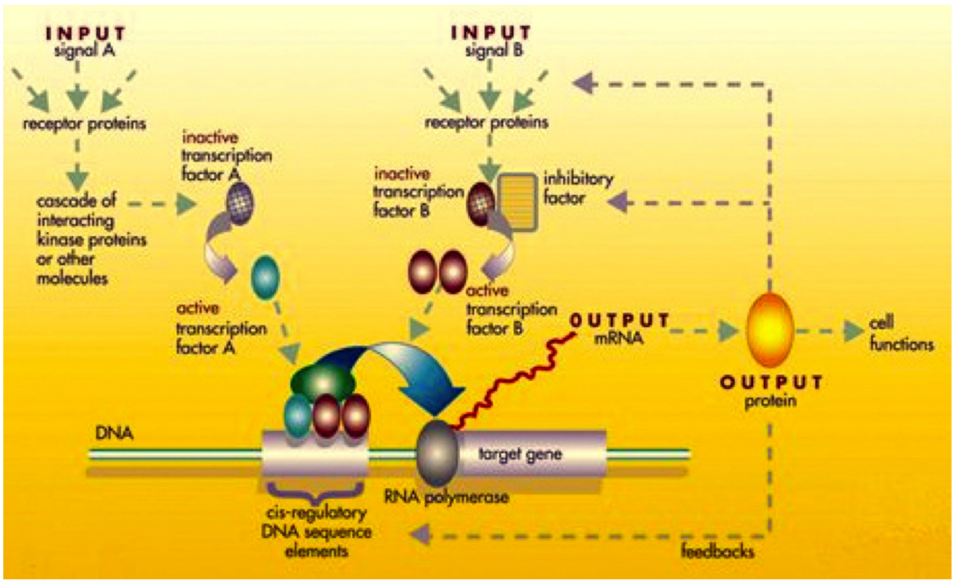

INPUT signal A
receptor proteins
cascade of interacting kinase proteins or other molecules
inactive transcription factor A
active transcription factor A
INPUT signal B
receptor proteins
inactive transcription factor B
inhibitory factor
active transcription factor B
OUTPUT mRNA
OUTPUT protein
cell functions
DNA
target gene
RNA polymerase
cis-regulatory DNA sequence elements
feedbacks

Figure 2.4: A gene regulatory network (*Source: U.S. Department of Energy Genomics:GTL Program, http://genomics.energy.gov*).

A simple GRN can be viewed as cellular input-output device containing an input signal reception and transduction system, a core gene regulatory network component and output in the form of RNAs and proteins. The core gene regulatory network component consists of regulatory proteins and *cis*-acting DNA sequences.

These regulatory proteins bind to the specific *cis*-acting DNA sequence to start transcription.

Figure 2.4 shows an example of the structure of a GRN where the transcription factors (transcription factor A and B) enable the target gene to be transcribed into mRNA and its end products, proteins. The transcribed protein may also act as a feedback to regulate those transcription factors themselves and other cellular functions.

Figure 2.5 shows the general control process of a typical, single level of GRN. The input signal reception and transduction system induces intra-cellular and/or extra cellular signals to a group of transcription factors to activate them. The GRN component comprises these activated regulatory proteins and *cis*-acting DNA sequences of their target genes. This component can up-regulate or down-regulate the synthesis of the corresponding primary output such as RNAs or proteins. The primary output changes the phenotypes or cell functions, which are the terminal outputs. Direct and indirect feedbacks can modulate the level of input. Though here only a single level of GRN has been shown, GRNs can however be composed of multiple GRNs resulting in complex interactions, where the products of one level can regulate the expression of another level.

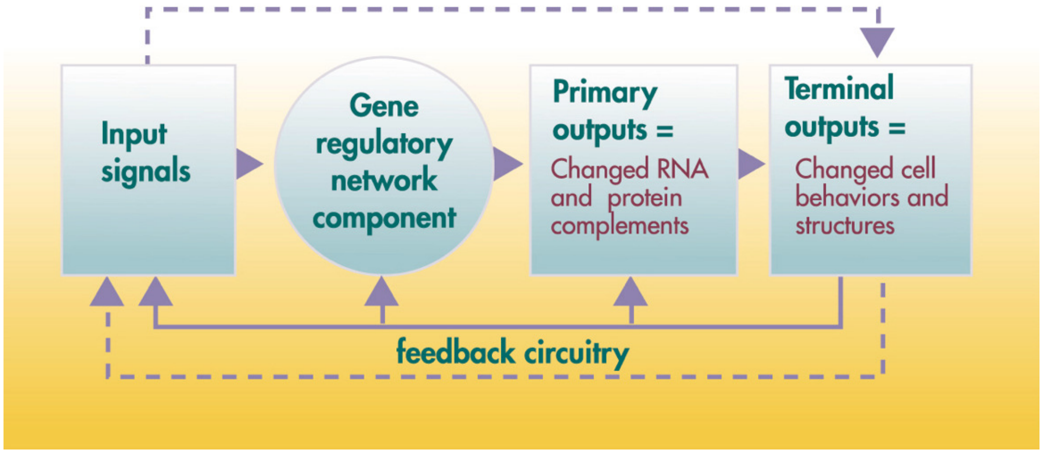
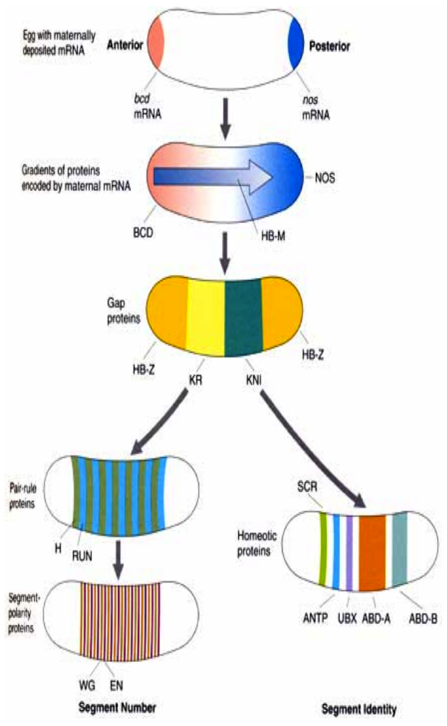
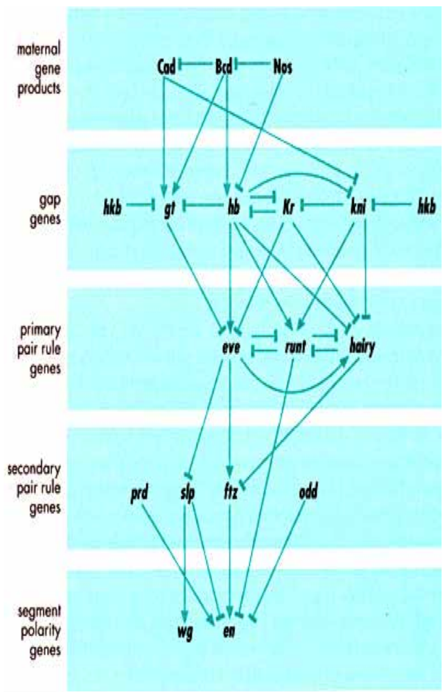

Input signals
Gene regulatory network component
Primary outputs = Changed RNA and protein complements
Terminal outputs = Changed cell behaviors and structures
feedback circuitry

Figure 2.5: The control process of a gene regulatory network *(Source: U.S. Department of Energy Genomics:GTL Program, http://genomics.energy.gov)*.

One example of a highly studied and modelled GRN is that of the developmental stages of drosophila embryogensis. Gene expression occurs in different body parts and segments discretely during the developmental stages of drosophila. Various gene products (proteins) are produced during the different stages of the drosophila embryo development which themselves regulate gene expression. As a result the embryo generates different gene expression levels of the gene products at different developmental stages. Figure 2.6 shows the developmental gene regulatory network controlling segmentation in drosophila development.



(a)



(b)

Figure 2.6: A gene regulatory network during drosophila embryogenesis.

2.7 Summary

Transcription is an essential stage in gene expression and the initiation of transcription is the preliminary stage for gene regulation to occur. Multiple transcription factors bind to DNA in the upstream regions of genes and actively take part in regulation by initiating, enhancing or suppressing the level of transcription. Identifying these binding sites is an important and interesting problem that biologists are facing today. The interaction between transcription factors and their binding sites will help advance our understanding of gene regulatory networks. However, identifying transcription factor binding sites is a difficult problem. The transcription factors do not always bind to a specific sites and it depends upon different factors. Also, they are variable in size. According to the statistics (shown in Table 2.1), the search space (non-coding regions in the genome) for transcription factor binding sites in complex organisms is also huge. As a result, finding these *cis*-sequences by laboratory methods is an expensive and time-consuming process. This reflects why the curated data for transcription factors in well-known repositories (for example, *ORegAnno*) is not sufficient with respect to estimated number of transcription factors and their binding sites.

Chapter 3

A Review of *cis*-regulatory Binding Sites Prediction Strategies

3.1 Introduction

As reviewed in the previous chapter, genes are regulated by proteins binding to specific stretches of DNA, or sites, on the genome according to whether specific stretches of DNA, or sites on the genome have a regulatory protein bound to them (see Figure 3.1). These sites, called transcription factor binding sites (TFBSs), are fundamental in the way cells and their genes interact.

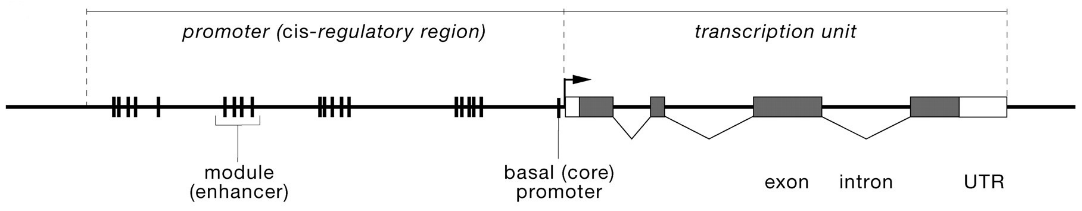


Figure 3.1: Relative position of the gene, basal promoter region and cis-binding sites in the organisation of a eukaryotic gene. The arrow indicates the direction of transcription (*source:* *Wray* et al. *(2003)*).

Unfortunately, locating TFBSs for a particular gene is non-trivial for various reasons:

i. The spatial locations of TFBSs relative to a gene are notoriously variable. They can be found nearby (upstream or downstream or even inside the coding region of a gene) or far away, sometimes hundred of thousands of nucleotides, from the genes they regulate.

ii. The genome size varies between species. As described in Section 2.3.3 (Table 2.1), the larger in size a genome is, the more it is difficult to reliably detect TFBSs due to their increase in the DNA to search.

iii. A specific regulatory protein may bind to more than one site, but these sites do not necessarily have a unique, unambiguous DNA sequence.

iv. In a regulatory module, there can be many sites that cooperate together and the boundary of the module can be difficult to determine. Moreover, though some *cis*-regulatory modules have similar functions, they do not contain exactly the same TFBSs.

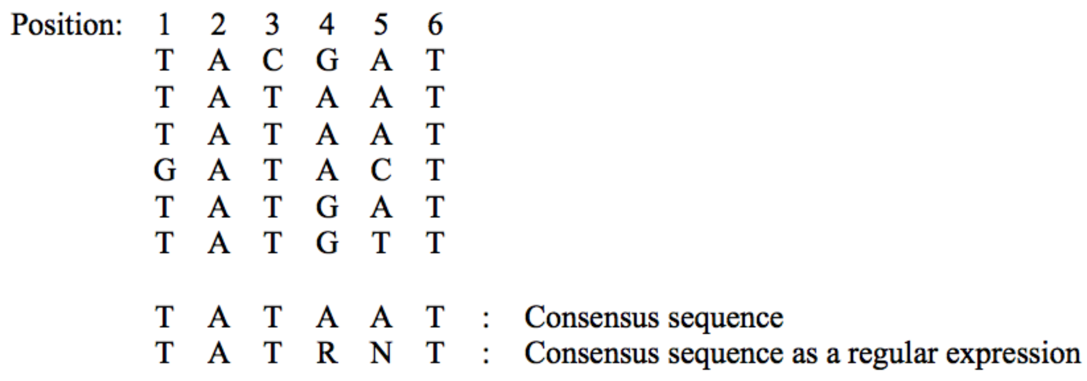
A number of experimental methods and technologies for identifying TFBSs have been developed. Conventional methods for recognising binding sites mainly depend upon footprinting methods (Blanchette *et al.*, 2002). These methods identify those regions of DNA that specifically bind particular proteins and characterise them by various procedures including nitrocellulose binding assays, gel shifting, etc. Currently, there are also high throughput methods such as EMSA, ChIP-chip, ChIP-seq, etc. available for experimentally identifying binding sites. These methods however, are costly, time consuming and some of them do not scale up to genome-wide analysis. Hence the need for computational approaches to identify TFBSs has become eminent. Nevertheless, laboratory techniques are still essential to establish the ground-truth and computational assessments are complementary to, rather than substituting the experimental approach. In this chapter I will discuss different computational approaches for recognising TFBSs. Firstly, I will discuss how genomic (DNA) sequences in general (and binding sites in particular) can be represented computationally and then introduce a number of different binding site identification algorithms.

3.2 Representation of DNA Sequences

There are a number of different ways to represent and characterise a genomic sequence. These representations, each with their own strengths and weaknesses, capture different statistical and structural information. Algorithms for TFBSs prediction are sometimes dependent upon these models. Therefore, a number of these representations are discussed below.

3.2.1 Consensus Sequence

One method of representing the pattern (nucleotide composition) of known binding sites is by constructing consensus sequences. It is the simplest representation of binding site model. In this method, the sequences of binding sites, which have typically been obtained by wet-lab experiments, are aligned together and consensus nucleotide letters are assigned to each column to represent nucleotide composition. There are two broad ways by which a consensus sequence can be constructed, both of which have been illustrated in Figure 3.2. Both methods simply use the frequency of the nucleotides in the positions of the aligned sequences. In one method the nucleotide with the highest frequency for each position is taken as the representative nucleotide (the consensus sequence in Figure 3.2). A second approach represents the nucleotides using the IUPAC (International Union of Pure and Applied Chemistry) notation (authors listed, 2001). Using this notation, in a consensus sequence the A, C, G, and T represent the individual nucleotide in a specific position. Other letters represent the ambiguity between nucleotides. For example, R represents a position that contains either an A or a G (purines), Y represents a position that contains either a C or a T (pyrimidine), N represents a position that has any of the four possible nucleotides. A detailed description of these notations can be found in Appendix A. To search for matches for a consensus sequence in a DNA sequence, a simple regular expression search is undertaken. Though consensus sequences provide a better representation than a single pattern, there are drawbacks (Stormo, 2000; Schneider, 2002). A consensus sequence is not always representative of the majority of binding sites (Schneider, 1997).



Position:	1	2	3	4	5	6		
	T	A	C	G	A	T		
	T	A	T	A	A	T		
	T	A	T	A	A	T		
	G	A	T	A	C	T		
	T	A	T	G	A	T		
	T	A	T	G	T	T		
	T	A	T	A	A	T	:	Consensus sequence
	T	A	T	R	N	T	:	Consensus sequence as a regular expression

Figure 3.2: A simple example illustrating the creation of a six nucleotides consensus sequence.

For example in Figure 3.2, there are six sequences used to build the consensus sequence -TATAAT. If we use the consensus sequence with no mismatch taking the most frequent nucleotides in each position, only two sites can be identified among the six sequences. If we allow one mismatch, we find three sites and two mismatches will identify all the sites. If we use the alternate consensus sequence (TATRNT) with no mismatches, four out of six sites can be identified.

A well-known consensus sequence of the -10 promoter region of the bacterium *Escherischia coli* is TATAAT and is derived from 291 sequences described in Ben-Gal *et al.* (2005). However, only 14 of these sequences follow the consensus sequence without any mismatch, which is insufficient for a reliable identification. For the six sequences in Figure 3.2, we need to introduce two mismatches to match the consensus sequence. Obviously, the larger the number of sequences, the more mismatches will give rise to the number of false identifications and alternatively consensus sequence may also reduce the number of real sites detected. The advantages of consensus sequences are that they are concise, simple to detect and easily remembered and displayed. However, using consensus sequences usually result in a loss of information, and it is difficult to quantitatively evaluate partial matches.

3.2.2 Position Weight Matrices

Generating profiles of binding sites is another way to summarise the information contained in a set of TFBSs. A Position Weight Matrix (PWM) is such a profiling tool. A PWM is a powerful method to model the binding specificity of a transcription factor. It provides a quantitative description of the known binding sites for a given TF. The common way to construct a PWM is to divide the nucleotide probabilities by the expected background probabilities and convert the values to a log-scale. The quantitative PWM score for a putative binding site is the sum of the PWM values for each nucleotide in the site. The nucleotide probability values can be then used to determine the total information content for each position.

For a set of sites of length n, a PWM can take the form of a $4 \times n$ matrix with scores assigned to the sequence by the following formula (Hertz & Stormo, 1999):

$$score = \sum_{i=1}^{n} \mathbf{W}_{b,i} \tag{3.1}$$

Here,

$$\mathbf{W}_{b,i} = \log_e \frac{\mathbf{A}_{b,i}}{\mathbf{B}_{b,i}} \tag{3.2}$$

$\mathbf{A}_{b,i}$ = Conditional probability that the position is found to be base b in the binding site sequences,
$\mathbf{B}_{b,i}$ = Conditional probability that the position is found to be base b in the non-binding site sequences.
For larger number of aligned sequences:

$$\mathbf{A}_{b,i} = \frac{\mathbf{C}_{b,i}}{Z} \tag{3.3}$$

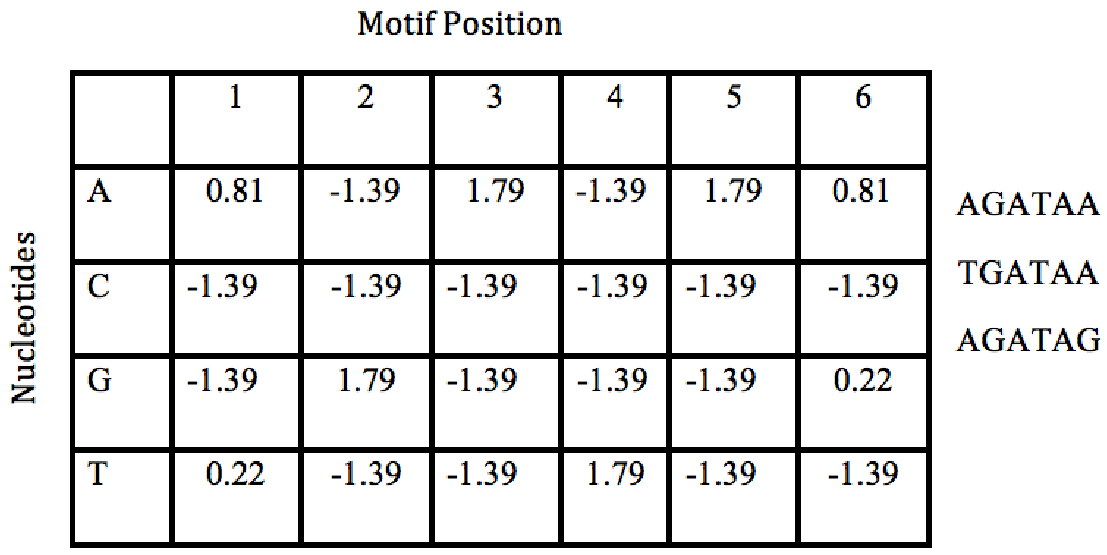
Here, $\mathbf{C}_{b,i}$ = Number of b nucleotide at position i
Z = Total number of aligned sequences
For smaller numbers of aligned sequences:

$$\mathbf{A}_{b,i} = \frac{\mathbf{C}_{b,i} + \mathbf{B}_{b,i}}{Z + 1} \tag{3.4}$$

For an example, let us take three short sequences: AGATAA, TGATAA, and

AGATAG.

By performing a simple alignment, we can construct a motif [AT]GATA[AG] indicating that the middle four bases are always GATA and the first position is either an A or a T and the last position is either an A or a G. But the problem with this motif is that it leads to a less specific search by allowing mismatches and it does not imply how detected sites should be ranked. A PWM attempts to overcome these issues by computing the log odd weights for a match score. For the three sequences the following PWM can be constructed:



Motif Position

Nucleotides

	1	2	3	4	5	6
A	0.81	-1.39	1.79	-1.39	1.79	0.81
C	-1.39	-1.39	-1.39	-1.39	-1.39	-1.39
G	-1.39	1.79	-1.39	-1.39	-1.39	0.22
T	0.22	-1.39	-1.39	1.79	-1.39	-1.39

AGATAA
TGATAA
AGATAG

Figure 3.3: An example of a Position-Weight Matrix.

A detailed explanation of how the PWM is constructed is given in Appendix B.

From this matrix we can calculate the score of different possible motifs and accept those motifs, which have score above a specified threshold. For example, we can determine how closely the following four sequences match the PWM by calculating their scores. The scores are:

AGTAGG:	0.81+1.79-1.39-1.39-1.39+0.22	=	-0.96
AGATAT:	0.81+1.79+1.79+1.79+1.79-1.39	=	6.58
TGATAA:	0.22+1.79+1.79+1.79+1.79+0.81	=	8.19
GGATAA:	-1.39+1.79+1.79+1.79+1.79+0.81	=	6.58

If we select the cut-off threshold value as 8.00 then, three sequences will be removed from the list and one will be left as a candidate of interest. If we decrease the value of the cut-off threshold, the number of matching sites increases. If we decrease the threshold to 6.00, only one sequence will be removed. Determining a suitable threshold value is the main challenge to reliably detect TFBSs using this technique, since by choosing an inappropriate threshold this may give rise to many false positives. Hence the challenge is to select the optimal threshold that maximises true positives while minimising false positives.

PWMs for TFBSs can be obtained from a number of public databases such as TRANSFAC (Matys *et al.*, 2003), JASPAR (Sandelin *et al.*, 2004), PAZAR (Portales-Casamar *et al.*, 2009), Frankel Lab [1], etc. These databases are often derived from experimentally verified TFBSs. Table 3.1 shows a summary of different public databases where PWMs are available.

	Number of TFs	Number of organisms	PWMs
TRANSFAC	17, 811 (includes miRNAs)	180	1,551
JASPAR	460	20	460
Fraenkel Lab	88	1	124
ORegAnno	561	17	-

Table 3.1: A summary of TFBSs databases.

TRANSFAC is the biggest database, which has more than 1500 PWMs across 180 organisms. JASPAR has far less number of PWMs for the binding sites of 460 TFs. A third public source of data is the Fraenkel Lab from MIT, which has 124 PWMs yeast.

The PWM is a robust representation for TFBS prediction without explicitly knowing other biological properties. As it is related to the binding energy of the

[1]http://fraenkel.mit.edu/

DNA-protein interaction, it can be considered both as a statistical and an energy-based model. One problem with the PWM is the assumption that the positions in the site contribute additively to the total activity and this may lead to an over-prediction of binding sites (Stormo & Fields, 1998; Tompa *et al.*, 2005), which has already been discussed before. Moreover PWMs are also heavily dependent on the number and quality of the sequences derived from the experimental data.

3.2.3 Representing TFBSs using Information Theory

Information theory is an alternate and arguably much better approach to using consensus sequences. The information content (Schneider *et al.*, 1986) at a position in a site is can be represented as:

$$I_i = 2 + \sum_{b=A}^{T} f_{b,i} \log_2 f_{b,i} \tag{3.5}$$

Where, i = position within the site
b = each possible bases (A, C, G, and T)
$f_{b,i}$ = observed frequency of each base at that position

However as defined, Equation 3.5 is only suitable for nucleotides that occur with equal probabilities, whereas in most organisms nucleotides are not distributed with equal probabilities across the genome. In that case, Equation 3.5 can be re-written in a more general form as:

$$I_{seq(i)} = \sum_{b=A}^{T} f_{b,i} \log_2 \frac{f_{b,i}}{p_b} \tag{3.6}$$

Here, p_b = frequency of base b in the whole genome

Equation 3.6 can be used as the estimate for binding energy contribution. As each position in the genome contributes independently to the total binding energy, a matrix $H(b, i)$ is used whose elements define the binding energy contribution as its elements (Heumann *et al.*, 1994). Here,

$$H(b,i) = -\ln \frac{f_{b,i}}{p_b} \tag{3.7}$$

For the collection of known binding sites for a particular protein, $H(b,i)$ is the maximum probability estimate for binding energy contribution of each base at each position and I_{seq} is the average binding energy for all the known sites (Stormo, 2000).

3.2.4 Sequence Logo

A sequence logo is a graphical representation of the biological information of an aligned set of binding sites, developed by Tom Schneider and Mile Stephens (Schneider & Stephens, 1990). In other words, a sequence logo is a visual representation of a PWM, where the aligned sequences are displayed as a set of stacked characters at each position. The logo shows the frequency of the bases and the relative height indicates the degree of sequence conservation. The height is measured in bits of information, with a maximum of 2 bits at each position. The frequency of any bases is not lost in the sequence logo as it does in a consensus sequence.

The height of the base at each different position is defined in Schneider & Stephens (1990). The height of base b at position l is $f(b,l)R_{sequence}(l)$.

Where,
$f(b,l)$ = Frequency of base b at position l
$R_{sequence}(l) = 2 - (H(l) + e(n))$ *bits per position*

And here,
$H(l) = \sum_{b=A}^{T} f(b,l)\log_2 f(b,l)$ *bits per position*
$e(n)$ = Uncertainty of the pattern for a small sample size (n) (Correction Factor) (Schneider *et al.*, 1986)

Figure 3.4 shows the sequence logo (generated by WEBLOGO (Crooks *et al.*, 2004)) of the -10 promoter region of the bacterium *Escherischia coli* (TATAAT) from 291 sequences described in Ben-Gal *et al.* (2005).

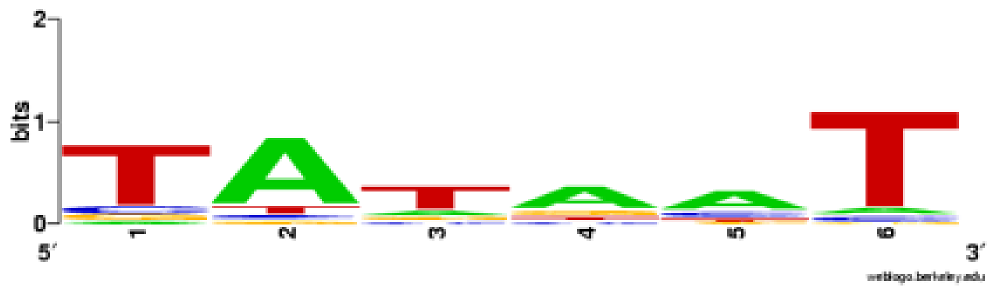


Figure 3.4: Sequence logo of -10 region of bacterial promoter (Ben-Gal *et al.*, 2005).

3.2.5 Markov Models

Markov chains and Hidden Markov Models (HMM) are two probabilistic models used for motif representation. A simple Markov model, composed of a number of states with the transition probabilities between the states (Durbin *et al.*, 1999), can be used for modelling a DNA sequence.

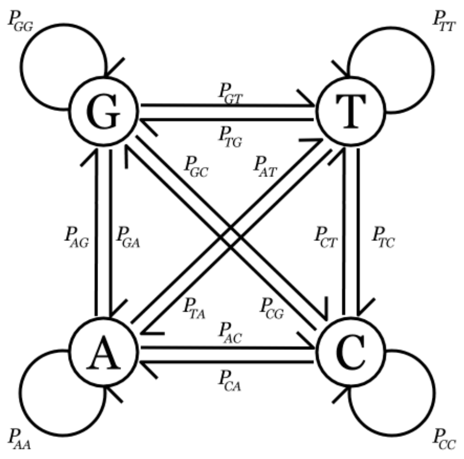


Figure 3.5: A Markov model of a DNA sequence. Any path from one node to another node will produce (emit) a DNA sequence.

Figure 3.5 shows a Markov chain for modelling DNA sequences, where each nucleotide represents a state and the arrows represent state transitions. Here the transitional probability is the conditional probability that a particular state will occur given the previous state in a sequence of states. Therefore, the transition probability from a previous state (s) to a particular state (t) in a sequence of states $\mathbf{x}$ is:

$$a_{st} = P\left(\mathbf{x}_i = t|\mathbf{x}_{i-1} = s\right) \tag{3.8}$$

In case of larger data, higher order Markov models are required and therefore a practical limit on the order needs to be placed in most applications (Eddy, 1996). The probability of a given sequences generated by using the Markov chain is as follows:

$$P(\mathbf{x}) = P\left(\mathbf{x}_l, \mathbf{x}_{l-1}, \ldots, \mathbf{x}_1\right) = P\left(\mathbf{x}_1\right) \prod_{i=2}^{L} a_{x_{i-1}x_i} \tag{3.9}$$

Here, $\mathbf{x}$ = a given sequence and L = length of the sequence

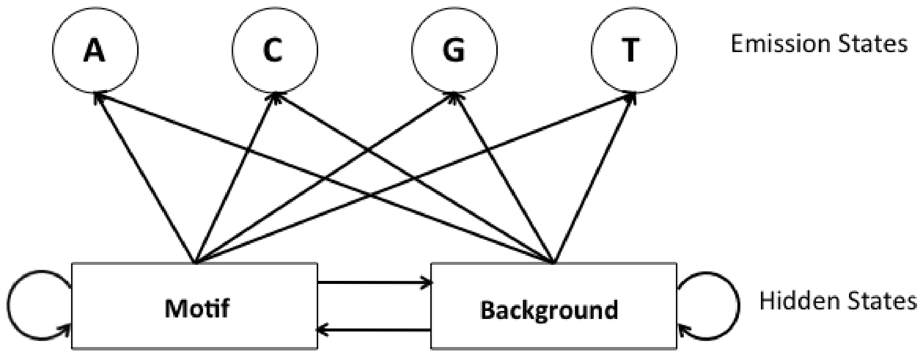


Figure 3.6: A simple HMM with two hidden states and four observable states. The square boxes represent the internal states (TFBSs or background). The circles represent the emission states (A,T, G, and C). The arrows show the transitions. Here the transition probabilities are not shown.

The Hidden Markov Model (HMM) is a powerful extension to the Markov model as it allows a model to contain a number of different states with potentially differing transition probabilities (Yada *et al.*, 1998; Eddy, 1996; Kochanski, 2004).

Figure 3.6 shows an example of a HMM that describes a promoter sequence consisting of a background sequence with short TFBSs.

This model can be used to predict the sequence annotations, *i.e.* TFBSs. The state transitions, which are the paths through the two states (here k and l) that would maximise the probability of generating such a sequence, can be determined by a classical Markov chain described in Equation 3.10:

$$a_{kl} = P\left(\boldsymbol{\pi}_i = l | \boldsymbol{\pi}_{i-1} = k\right) \tag{3.10}$$

The emission state of the HMM can be determined as:

$$e_k(b) = P\left(\mathbf{x}_i = b | \boldsymbol{\pi}_i = k\right) \tag{3.11}$$

Therefore, the joint probability of an observed sequence, $\mathbf{x}$ and a state sequence, $\boldsymbol{\pi}$:

$$\boldsymbol{P}\left(\mathbf{x}, \boldsymbol{\pi}\right) = \boldsymbol{a}_{\boldsymbol{\pi}_1} \prod_{\boldsymbol{i}=1}^{\boldsymbol{L}} \boldsymbol{e}_{\boldsymbol{\pi}_i}\left(\mathbf{x}_{\boldsymbol{i}}\right) \boldsymbol{a}_{\boldsymbol{\pi}_i \boldsymbol{\pi}_{i+1}} \tag{3.12}$$

The annotation (defined as the most probable state path) of an observed sequence can be then determined by the Viterbi algorithm (Eddy, 1996, 2004; Durbin *et al.*, 1999). There are different variations of HMMs used in Bioinformatics namely, profile HMM, phylo-HMM, etc. The profile hidden Markov model (Eddy, 1998), where each position in a motif is represented by a unique state with associated emission probabilities for that position, can be a well-suited candidate for the representation of TFBSs. The phylo-HMM (Phylogenetic Hidden Markov Model) is another variation of HMM, which can detect conserved elements based on multiple genome alignments. In the case of the phylo-HMM, a phylogenetic tree replaces the multinomial distribution and a new column in a multiple alignment is emitted at each time step (Siepel & Haussler, 2004). Similar to PWMs, HMMs are largely dependent on the amount of experimental binding site data for the estimation of accurate transition and emission probabilities. The data requirement for training a HMM can be so restrictive that sometimes HMM is impractical to use in many situations (Eddy, 1996).

3.3 Transcription Factor Binding Site Prediction Algorithms

A large number of algorithmic approaches have been introduced to identify candidate transcription factor binding sites *in silico* reviewed in Wei & Yu (2007); Tompa *et al.* (2005); Nguyen & Androulakis (2009); Das & Dai (2007); Pavesi *et al.* (2004); Elnitski *et al.* (2006); Blanchette *et al.* (2002); Hu *et al.* (2005). Many algorithms have been developed to exploit the various sources of experimental information available, and the various statistical properties that appear to distinguish regulatory regions from the genome in general using DNA sequence representation (discussed in the previous section). There are more than 100 TFBSs prediction algorithms currently available – almost 50% of them use PWMs for their match models for predictions, 10% of them use regular expressions, and the rest uses other strategies for predictions (such as HMM, phylogenetic, etc). Broadly, these algorithms can be classified into four main groups based on the approach to the problem *scanning*, *statistical*, *co-regulatory*, and *phylogenetic* algorithms.

3.3.1 Scanning algorithms

Scanning algorithms search for sequences that match with experimentally verified binding sites (Quandt *et al.*, 1995; Rajewsky *et al.*, 2002; Kel *et al.*, 2003; Yan *et al.*, 2005) by using binding site motif representations. The scanning algorithm simply performs a regular expression search on the target sequences and in this case a consensus sequence representation is used. Due to the conservative nature of a consensus sequence, these algorithms can produce predictions with a low rate of false positives as well as a high rate of false negatives. Mismatches can be introduced to reduce the false negative predictions as explained in Section 3.2.1.

Scanning algorithms can use PWM representations that can be used to construct a probability distribution along the length of a target sequence by calculating the log likelihood of each starting point in the sequence providing the match to the motif model. In this case, the choice of a threshold value is quite crucial. As discussed in Section 3.2.2, any sequence with a score above some predetermined threshold can be considered as a putative binding site. The performance of a scanning algorithm is normally dependent on the quality of the data used to generate the motif representation, the accuracy of the background model and the threshold value. In this thesis a number of scanning algorithms have been used: *Fuzznuc*, *MotifScanner*, *MotifLocator*, *Ahab*, *EvoSelex*,*P-match* etc.

3.3.2 Statistical algorithms

Statistical algorithms attempt to predict the location of *cis*-regulatory binding sites based exclusively on the statistical properties of genomic sequences, and utilising no prior information. These algorithms particularly play important roles in characterising the promoter regions of an organism where the prior information (such as binding site motif models, expression profiles, orthologous sequence from related species, etc) is not available and prediction based on the statistical properties of the sequence is the only available option. Statistical algorithms represent a diverse array of approaches applied to the problem of binding site prediction and often incorporate biological knowledge about DNA-protein interactions in regulatory systems (Galas *et al.*, 1985; Brazma *et al.*, 1998a,b; van Helden *et al.*, 1998; Marsan & Sagot, 2000; Papatsenko *et al.*, 2002; Sinha & Tompa, 2002; Apostolico *et al.*, 2004; Frith *et al.*, 2004).

The underlying hypothesis for determining TFBSs, for statistical algorithms, is based on the following two observations:

i. Regulatory regions often have multiple copies of a particular binding motif, leading to statistical over-representation of the motif locally (Berman *et al.*, 2002; Papatsenko *et al.*, 2002);

ii. Functional binding motifs should be restricted to the regulatory regions of the genes they regulate to prevent titration of the *trans*-acting transcription factors, resulting in the under-representation of the binding motifs over larger genomic stretches (Schneider *et al.*, 1986).

Statistical algorithms mainly rely on an enumeration of all statistically improbable words that occur in a sequence- in this case a set of TFBSs sequences. The determination of the probability of words can be estimated by a number of different approaches- such as a direct count of word frequencies, or matches to some variable representation and then is compared to an estimation of the expected frequency given the background model for the sequence. An alternative approach is to calculate the probability of occurrence for all words in a sequence, given a Markov model for the background.

Although statistical algorithm can correctly predict the location of experimentally verified binding sites (Papatsenko *et al.*, 2002; Sinha & Tompa, 2002; Apostolico *et al.*, 2004), they can produce predictions with high false positive rate (Tompa *et al.*, 2005). Moreover, over and under-representation alone are not sufficient to distinguish *cis*-regulatory elements from the background sequence. Other sources of biological information, such as functional annotation of regulated genes (Cora *et al.*, 2004), the clustering of predicted binding sites, the location of predictions relative to a promoter (Kiełbasa *et al.*, 2001; Hampson *et al.*, 2002), the use of structural family binding profiles (Sandelin & Wasserman, 2004) can be used for further refinement of predictions. In this thesis, I have used several statistical algorithms namely *PARS*, *DREAM*, and *Verbumculus*.

3.3.3 Co-regulatory algorithms

Any statistical algorithm that can make predictions for single sequences can be trivially extended to make predictions for a set of genes clustered on the assumption described in the previous section. Moreover, the inclusion of information, described in the previous section, can extend the predictive power of statistical algorithms. Algorithms based on this approach are among the most efficient prediction tools currently available. In this approach, the main hypothesis in this case is that if a set of genes is regulated by the same transcription factors, then the associated binding motifs are expected to be statistically over-represented in the promoter regions of the set (Markstein *et al.*, 2002; Ptashne & Gann, 2002). In practice, microarray data for genomic expression profiling is used to assess this and here the assumption is made that genes clustered by their expression profiles may be regulated by the same transcription factors (Roth *et al.*, 1998; Hampson *et al.*, 2000; Bussemaker *et al.*, 2001). Though we can rationally assume that genes regulated simultaneously are very likely to be responding to the same genetic signals, it is not true for all the case (Bussemaker *et al.*, 2001). Moreover, determination of gene clusters in this way is strongly dependant on the effectiveness of the clustering algorithms used in this process (Dougherty *et al.*, 2002).

Two powerful approaches that are often used are *Gibbs sampling* and *Expectation Maximisation* and they are almost solely used for co-regulatory analysis and therefore will be discussed here.

3.3.3.1 Gibbs Sampling

Gibbs sampling, an iterative stochastic sampling technique, is based on the application of Bayesian theory and it is used for solving optimization problems (Lawrence *et al.*, 1993; Neuwald *et al.*, 1995; Durbin *et al.*, 1999). It can be used in predicting *cis*-regulatory binding sites on a set of co-regulated gene regulatory sequences and hence formulated as follows:

The algorithm iterates through two steps- the predictive update and the sampling step. The *predictive update* step works by selecting one sequence from the set of promoter sequences randomly. The substrings of a predetermined length, starting from positions contained in a set of starting positions, are then aligned, and a probabilistic profile is generated. This profile represents the current model of the binding motif. A model of the background sequence is also generated from all sequences, excluding the sequence in use to generate the profile. Then the *sampling* step continues to exploit the generated motif profile and background model to calculate the likelihood ratio for each possible subsequence in the selected sequence. These likelihood ratios are then used as probabilistic weightings allowing a new motif start position in the sequence concerned. The stochasticity of the sampling step ensures that the evolving solution does not get stuck in local optimum. Once an optimal motif profile has been generated, all matches above a threshold can be masked and the algorithm is re-run for allowing predictions for multiple binding motifs. An alternative strategy is to fit the parameters for multiple motif profiles simultaneously. This helps the sampler to avoid the difficulty of modelling a site with a variable gap using a matrix representation and model conserved regions of the binding site separately.

Gibbs sampling algorithm requires no prior knowledge about binding sites to predict both the locations and identity of a binding motif. However, one thing should be noted that this type of algorithm is not sufficient for an exhaustive motif search as it is not suitable for finding rare motifs. In summary, the Gibbs sampling algorithm represents a very efficient and powerful heuristic for the detection of over-represented motifs.

3.3.3.2 Expectation Maximisation

The Expectation Maximisation (EM) algorithm is a deterministic approach to the problem of identifying over-represented patterns in a set of sequences. It is actually the maximum likelihood estimation of the parameters for a two state finite mixture model, which describes a set of sequences (Bailey & Elkan, 1994; Durbin *et al.*, 1999). In this context the two models correspond to the motif profile and the background sequence model. The starting positions of the two components within the sequences are also considered as unknown data and need to be estimated from the observed sequences.

EM takes a set of unaligned sequences and a motif length as inputs and returns a probabilistic model of the motif with the highest maximum likelihood score given the input sequences. The idea behind the use of EM for finding motifs is that ideally each sequence contains an example of the motif whose position is unknown. It is assumed that the motif is generated by a sequence of independent and multinomial random variables. As the sequences are not aligned in the dataset, the offset needs to be determined by estimation. The EM algorithm estimates the probability of the motif that starts in some position of a sequence and is then re-estimated.

The EM algorithm is guaranteed to find a local maximum for the likelihood of the model parameters and the missing data, given the original sequence data (Durbin *et al.*, 1999). For this reason it is far more sensitive to the choice of the initial parameter estimates than a stochastic algorithm such as Gibbs sampling, and additional algorithms are typically used to calculate the optimal initial parameter estimates (Bailey & Elkan, 1995). This is perhaps less of a problem in the implementation of the EM algorithm that iteratively searches for multiple motifs in a set of sequences (Bailey & Elkan, 1995). In such an example it is perhaps more likely that each local maximum explored is biologically interesting. In this thesis some co-regulatory algorithms have been used namely *MEME*, *AlignACE*, and *Sampler*.

3.3.4 Phylogenetic and other alignment based approaches

Phylogenetic approaches to binding site prediction use the assumption that binding sites are likely to show sequence conservation in closely related orthologous species (Fickett & Wasserman, 2000; Wasserman & Sandelin, 2004). In this case, the choice of a comparison species with an appropriate evolutionary distance is essential. Figure 3.7 shows the conservation of CTCF binding site between mouse, rat, and human H19 DMR regions (Bell & Felsenfeld, 2000).

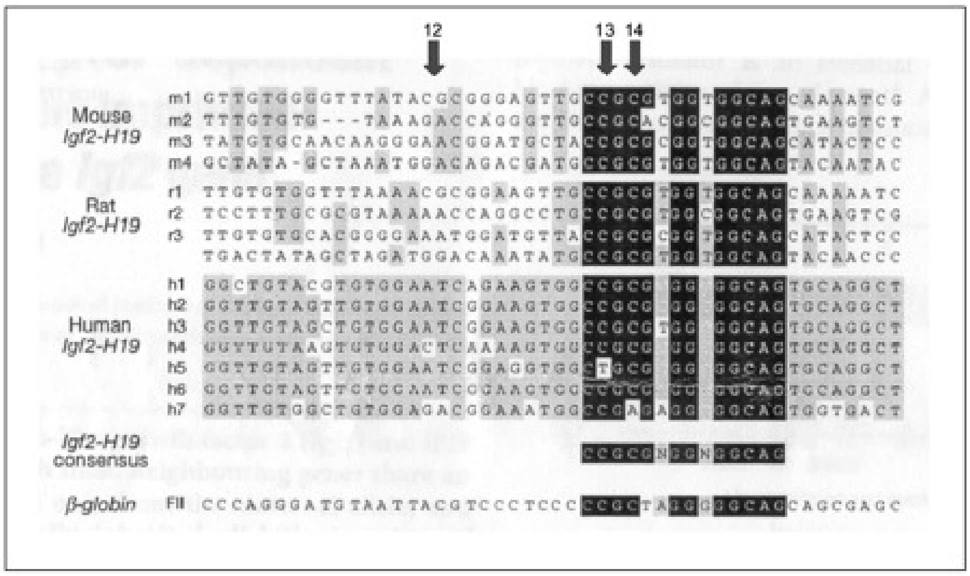


Figure 3.7: Sequence conservation across different species. Species-specific identities are shown in grey and cross-species conservation is shown in black. This figure is taken from Bell & Felsenfeld (2000).

If the evolutionary distance between the species is too small, the non-functional sequence will not have had a chance to diverge through the accumulation of mutations. Hence large stretches of DNA will perfectly match, making the identification of the short, functional TFBS sequences impossible to find. Alternatively, too large a distance and the regulatory inputs for the gene in question may have mutated such that they are dissimilar across species, undermining the basic assumption of identifiable, shared DNA sequences (Fickett & Wasserman, 2000).

The alignment of evolutionary related sequences, either orthologous or paralogous (genes at different chromosomal locations in the same organism that have structural similarities, indicating that they are derived from a common ancestral gene), utilises a set of well-established techniques (Durbin *et al.*, 1999; Jareborg *et al.*, 1999; Bray & Pachter, 2003). These techniques are often used to predict the biological identity or functionality of novel sequences based on their homologies with sequences of known identity or functionality. Algorithms for both the global and local alignment (described in Section 2.4) of such sequences have been very successfully used for many years. Alignment models can give the explanation for the various evolutionary changes (such as deletion, insertion, mutations, etc.) that a sequence might be subject to. These models use parameters, for example, a gap penalty, to enable optimal alignments to be efficiently calculated using dynamic programming techniques (Durbin *et al.*, 1999; Eddy, 2004).

A study by Wasserman *et al.* (2000) showed that approximately only 19% of non-coding DNA was contained within phylogenetic alignments, or footprints while comparing human and rodent sequences. However, 98% of experimentally determined binding sites were located within this subset of sequence. This observation suggests a new strategy for binding site prediction, where the phylogenetic analysis of all orthologously paired non-coding sequence is done followed by a Gibbs sampling/ EM/ motif scanning strategy on the subset of sequence contained within the footprints (McCue *et al.*, 2001; Sinha *et al.*, 2004). Thus the accuracy for identifying binding sites is be expected to improve by focusing the search algorithms on regions with a higher probability of containing binding sites and binding site prediction over a larger genomic scale also becomes much more feasible. However, such a strategy is greatly dependent on both the amount of available sequence and the identification of orthologously related non-coding regions.

In this thesis, I have used two phylogenetic algorithms namely *SeqComp* and *Footprinter*. I have also used *Regulatory Potential*, and *PhastCons* as indirect sources of evidence.

3.4 Combining Sources of Evidence

The algorithmic strategies described above are diverse and incorporate differing sources of biological information in the predictive process. They all have there own strengths and weaknesses. Therefore, there is good reason to consider that the set of binding sites predicted correctly by the individual sources of evidence are likely to form non-identical sets. If these predictions do really complement each other, then they can potentially provide more significant information when taken in combination. Therefore, combining their outputs may lead to better predictions. If one algorithm misses any binding site another algorithm may be able to capture that site. There are a number of approaches where the results from different algorithms have been combined together to improved predictions. Among them the most notable are *BEST* (Che *et al.*, 2005), *Multifinder* (Huber & Bulyk, 2006), *WebMOTIFS* (Romer *et al.*, 2007), and *MEMOFinder* (Wilczynski *et al.*, 2008) .

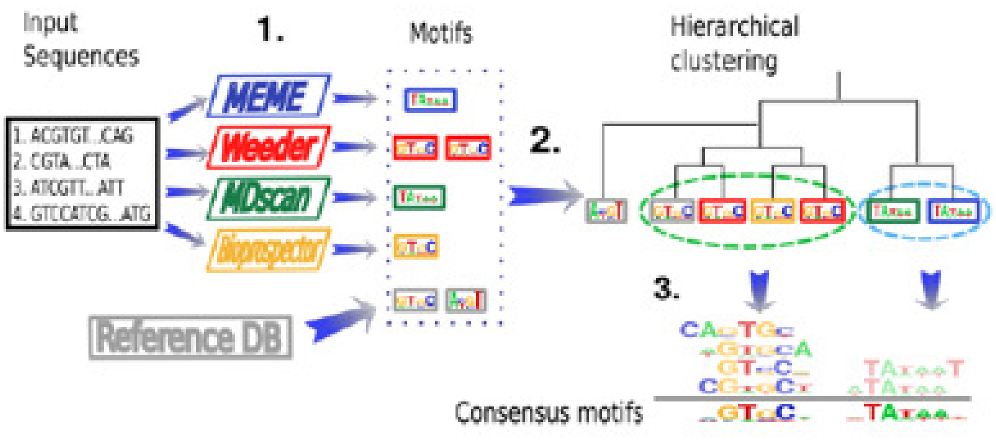


Figure 3.8: MEMOFinder. This figure is taken from (Wilczynski *et al.*, 2008).

BEST (Binding-site Estimation Suite of Tools) includes the co-regulated algorithms *AlignACE*, *BioProspector* (Liu *et al.*, 2001), *CONSENSUS* (Hertz & Stormo, 1999), and *MEME*, as well as the optimisation program *BioOptimizer* (Jensen & Liu, 2004). In *BEST*, *BioOptimizer* ranks the predictions from the different motif finding algorithms and presents the top 10 motifs with motif score, width, number of predicted sites and consensus sequence from both the original algorithms and *BioOptimizer*. *Multifinder* uses four motif discovery programs (*AlignACE*, *MDScan* (Liu *et al.*, 2002), *BioProspector* and *MEME*) and combines their results by using a clustering method. *WebMOTIFS* is another tool, which works in a similar way as Multifinder, but it includes *Weeder* (Pavesi *et al.*, 2001) in place of *BioProspector*. *MEMOFinder* (see Figure 3.8) is the most recent approach for combining the output from motif discovery algorithms using *MEME*, *Weeder*, *MDScan* and *BioProspector*.

Figure 3.8 shows that *MEMOFinder* takes the output from the base algorithms, produces a distance matrix from them and clusters them accordingly. It also generates consensus motifs from the outputs. The key theme with these approaches is to use multiple algorithms to generate sets of putative predictions and they differ in the strategies used to combine the predictions together.

3.5 Summary

This chapter has reviewed the main representation of biological features integral to the computational binding site predictions, along with the main algorithmic strategies that have been used in the literature. The algorithms described have been selected to represent both the major contributions in the field and also provide necessary background on the algorithmic strategies that are utilised later in the course of this thesis. At the end of this chapter, the popular approach of combining these sources of evidence has been discussed. The combination of algorithms produces promising results and this idea will be adapted and described broadly in later chapters.

Chapter 4

Machine Learning and Sampling Techniques

4.1 Introduction

The existence of hundreds of public databases, with enormous amount of data, needs proper cataloging and representations with respect to its biological significance. This gives rise to the necessity of computational tools to analyse this data in an efficient manner. The importance of classification techniques to the bioinformatics community has long been recognised. The application of these classification techniques to various biological problems is of increasing importance (Workman & Stormo, 2000; Jensen & Liu, 2004; Radivojac *et al.*, 2004; Ahmad & Sarai, 2005; Beiko & Charlebois, 2005; Dietterich, 2002). There are many classification techniques developed by the machine learning community to tackle the problem of data classification.

A number of classifier methods have recently been employed and among these the Support Vector Machine (SVM) is currently enjoying popularity. SVMs have been used to predict regulatory motifs (Sun *et al.*, 2006a; Vert *et al.*, 2005; Holloway *et al.*, 2005; Jiang *et al.*, 2007), gene regulatory networks (Qian *et al.*, 2003) and to detect functionally similar proteins (Leslie *et al.*, 2002, 2004). They have been used for classification of tissue examples (*e.g.* type of cancer) based on microarray data (Furey *et al.*, 2000; Guyon *et al.*, 2002), prediction of the function of uncharacterised genes (Pavlidis *et al.*, 2001), and prediction of protein sub-cellular localisation (Yu *et al.*, 2006; Hua & Sun, 2001b). SVMs have also been used for predicting protein secondary structure (Hua & Sun, 2001a), protein folding (Ding & Dubchak, 2001), protein super family (Jaakkola *et al.*, 2000), etc.

Prediction of *cis*-regulatory binding sites in regulatory DNA sequences can be formulated in terms of a classification problem and it can be a good candidate problem for the application of these algorithms. Classification algorithms can be quite helpful in classifying this metadata. The major aim of this thesis is to demonstrate the utility of classification algorithms for improving the performance of individual binding site prediction algorithm. Integrating multiple binding sites prediction and learning the correct classification given the initial prediction can achieve this. This chapter will, therefore, undertake a review of SVMs and some of the related techniques such as data sampling to enhance their training capabilities.

4.2 Two-class Kernel Method

The kernel method for classification is a recently developed technique in the Machine Learning field (Cristianini & Shawe-Taylor, 2010). Here the word kernel is related to mapping data into a higher dimensional space. The simplest form of SVM to consider is a two-class classifier, where objects belong to two categories - positive examples and negative examples (Boser *et al.*, 1992). The general idea is to separate the data just by drawing a separator (for example, a hyperplane) between them and dividing them into two classes.

4.2.1 Maximum margin separator

There can be many separators that can make a distinction between the two classes. In Figure 4.1(a), there are two classes of data, which can be separated by a single separator. However, Figure 4.1(b) shows a number of different separators can also separate these two classes. Now the question is: which one should be chosen as an ideal separator?

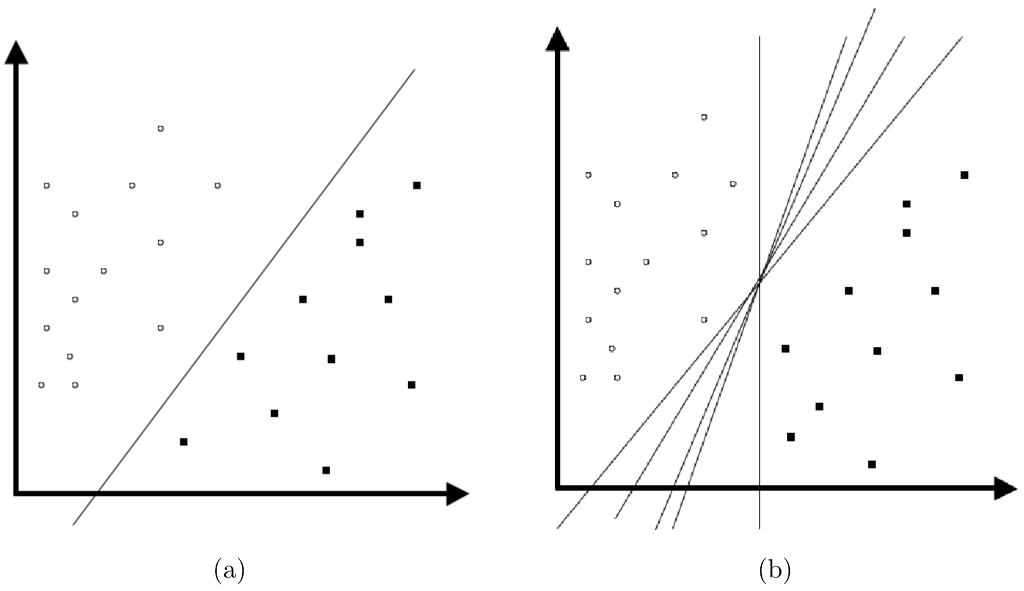


Figure 4.1: Two-class classifier.

A solution is to choose the separator with the maximum margin. Here, the margin is defined as the distance of the closest data point to the separator in this case a hyperplane. By having maximum margin this separator can create maximum separation between the two classes. Figure 4.2 shows the separator with maximum margin and the data points push up against the margin are the Support Vectors. A better separation of data, that minimises the risk of over-fitting (discussed in Section 4.4), can be obtained by allowing some misclassifications. Hence, a term called cost can be incorporated with it to prioritise the importance of misclassifications. For larger value of cost, a larger penalty is assigned to the errors, whereas a smaller value of cost leads to larger margin. Therefore, a large cost may cause over-fitting and a small one may cause under-fitting. Further effect of cost will be discussed in the next section.

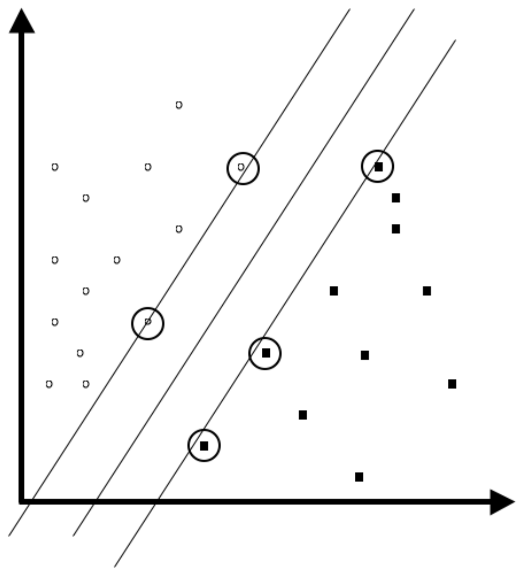


Figure 4.2: Maximum margin separator.

4.2.2 Two-class SVM

A Support Vector Machine (SVM) is a maximum margin classifier with a tuneable cost parameter. Let, a training set of a number of patterns be $\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \cdots, \mathbf{x}_{\boldsymbol{n}}\}$ with known labels $\{\boldsymbol{y_1}, \boldsymbol{y_2}, \boldsymbol{y_3}, \cdots, \boldsymbol{y_n}\}$ where $\boldsymbol{y_i} \in \{\mathbf{-1}, \mathbf{+1}\}$. The training patterns are used to build a decision function, $\boldsymbol{D}(\mathbf{x})$ such that,
$\mathbf{x} \in \boldsymbol{class}(+)$ when,$\boldsymbol{D}(\mathbf{x}) \geq \mathbf{+1}$; if $\boldsymbol{y_i} = \mathbf{+1}$
$\mathbf{x} \in \boldsymbol{class}(-)$ when, $\boldsymbol{D}(\mathbf{x}) \leq \mathbf{-1}$; if $\boldsymbol{y_i} = \mathbf{-1}$
Here, $\boldsymbol{i} = \mathbf{1, 2, 3}, \cdots, \boldsymbol{n}$
$\mathbf{x}$ is on decision boundary when, $\boldsymbol{D}(\mathbf{x}) = \mathbf{0}$
Here,

$$\boldsymbol{D}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + \boldsymbol{b} \tag{4.1}$$

$\mathbf{w}$: weight vector
$\boldsymbol{b}$: bias value
$\mathbf{w} \cdot \mathbf{x}$: dot product between the two vectors $\mathbf{w}$ and $\mathbf{x}$. This convention will be followed all through this thesis.
Equation 4.1 is the *discriminant function* of the hyperplane that divides the

data points into two classes. In classification with large margin, a boundary of the hyperplane can be set and we can adjust the value of bias such that the hyperplane fits in the middle of the margin (see Figure 4.3).

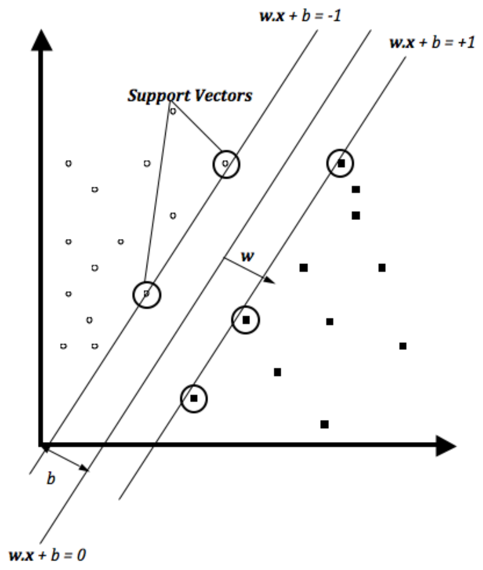


Figure 4.3: Two-class SVM.

For Equation 4.1, the margin is $\frac{1}{\|\mathbf{w}\|}$ and here $\|\mathbf{w}\|$ is the length of $\mathbf{w}$. To calculate the value of $\mathbf{w}$ and $\boldsymbol{b}$, we need to solve the following optimisation problem:
$\boldsymbol{min \frac{1}{2} \|\mathbf{w}\|^2}$
subject to: $\boldsymbol{y_i(\mathbf{w} \cdot \mathbf{x} + b) \geq 1}$ for $\boldsymbol{i = 1, 2, \cdots, n}$
Here by minimizing $\|\mathbf{w}\|^2$, we are actually maximising the margin to classify examples correctly. This is called *hard margin* problem. A hard margin problem does not allow any misclassifications during building the training model.

However in practice, the data may not be linearly separable or even linearly separable data may need a greater margin for classification. In theory and experimental results it is found that a larger margin can provide better classification performance than the hard margin SVM (Ben-Hur *et al.*, 2008). Therefore, A *soft margin* problem, by allowing some misclassifications during training, can provide better classification performance. To do this a set of *slack variables* ($\boldsymbol{\xi}$), one for each data point, is introduced. For correctly classified points the slack variable is set to zero (see Figure 4.4), whereas if it is in the decision boundary the value is in between 0 to 1. But when the instance is misclassified the value of slack variable used is greater than 1.

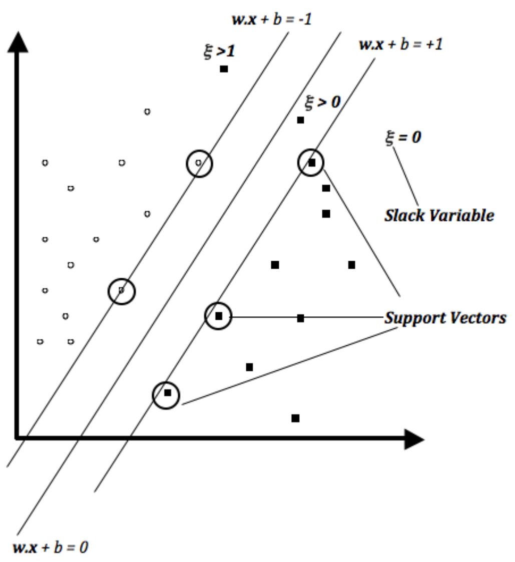


Figure 4.4: Use of slack variable in two-class SVM.

As mentioned before, to discourage too much use of *slack variable*, a term called cost ($\boldsymbol{C}$) has been incorporated with it to maximise the margin and minimise the slack variable. Now the optimisation problem after using slack variable is:

$$\boldsymbol{min}\left[\frac{1}{2}\left\|\mathbf{w}\right\|^2 + \boldsymbol{c}\sum_{i=1}^{n}\boldsymbol{\xi_i}\right] \tag{4.2}$$

subject to: $\boldsymbol{y_i(\mathbf{w}\cdot\mathbf{x}+b)\geq 1-\xi_i}$, for $\boldsymbol{i=1,2,\cdots,n}$ and $\boldsymbol{\xi_i\geq 1}$
A decision boundary for an SVM with a very high cost value has a narrow margin around the decision boundary that may lead to under-fitting the data. On the other hand, a smaller value of cost of the decision boundary for an SVM increases the margin and therefore may introduce over-fitting.

Equation 4.2 can be reformulated in to dual form by using *Lagrange multipli-*

ers (Boyd & Vandenberghe, 2004). The dual form is as follows:

$$\boldsymbol{max} \left[\sum_{i=1}^{n} \boldsymbol{\alpha_i} - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{y_i y_j \alpha_i \alpha_j} \mathbf{x}_i \cdot \mathbf{x}_j \right] \tag{4.3}$$

subject to: $\sum_{i=1}^{n} \boldsymbol{y_i \alpha_i = 0, 0 \leq \alpha_i \leq c}$
From this dual form, it can be proved that,

$$\mathbf{w} = \sum_{i=1}^{n} \boldsymbol{y_i \alpha_i} \mathbf{x}_i \tag{4.4}$$

For $\boldsymbol{\alpha_i > 0}$, $\mathbf{x}_i$ is called *support vector*. (Ben-Hur *et al.*, 2008)

4.2.3 Feature space

The problem with a linear classifier is that it may not be able to satisfactorily deal with non-linearly separable data (see left hand side of Figure 4.5). Sometimes non-linear classifiers give a better classification than linear classifiers in these cases. One solution to this is to map the data into a high dimensional *feature space* including non-linear features and then use a linear classifier.

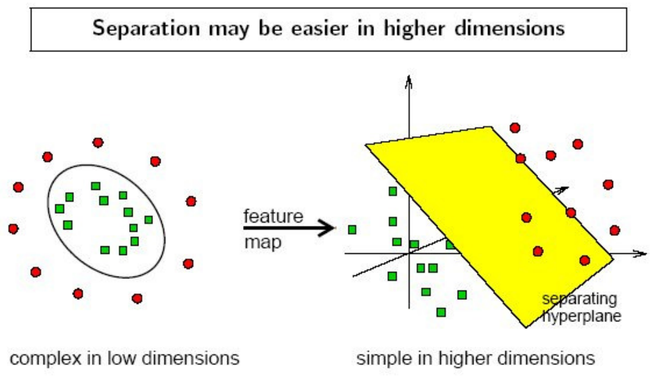

Separation may be easier in higher dimensions
feature
map
separating
hyperplane
complex in low dimensions
simple in higher dimensions

Figure 4.5: Concept of non-linear data classification. Here $\boldsymbol{\phi}$ is a mapping function (*Source: http://www.dtreg.com/svm.htm*).

Figure 4.5 shows that using a non-linear classifier can separate the data. But this data can be implicitly mapped to another space (called a *feature space*) to make it linearly separable. A mapping function ($\boldsymbol{\phi}$) maps the data to feature space with higher dimension making the data more likely separable. A linear separator in the feature space may correspond to a non-linear separator in the original space.

But now the question is: how many features are needed to be computed and which feature should they be? The answer may be to generate nonlinear decision boundaries by using kernel methods. Figure 4.5 shows that using a non-linear classifier can separate the data. But these data can be implicitly mapped to feature space to make it linearly separable.

4.2.4 Kernel functions

To compute a hyperplane, we need to compute dot products in the data space. But this dot product can be replaced by other functions known as *kernel functions*. The interesting property of a kernel function is that by using it we do not need to explore the feature space and rather all computations can be done on the original data.

If the kernel function is defined as $\boldsymbol{k}(\mathbf{x}, \mathbf{x}')$, then $\boldsymbol{k}(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}')$and it can be computed efficiently as it solves the problem of mapping data into a very high-dimensional space. Actually, we do not need to compute $\boldsymbol{\phi}(\mathbf{x})$ or even know what it is.

The two most widely used kernel functions are *polynomial kernel* and *Gaussian kernel*. The *polynomial kernel* of degree $\boldsymbol{d}$ is defined as:

$$\boldsymbol{k}_{\boldsymbol{d},\boldsymbol{x}}^{\boldsymbol{polynomial}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + \boldsymbol{\delta})^{\boldsymbol{d}} \tag{4.5}$$

Here, $\boldsymbol{\delta}$ is zero($\mathbf{0}$) if homogenous and one ($\mathbf{1}$) if chosen to be inhomogeneous.
If $\boldsymbol{d} = \mathbf{1}$and $\boldsymbol{\delta} = \mathbf{0}$, the kernel function is defined as *linear kernel* function, $\boldsymbol{k}^{\boldsymbol{linear}}$, which is the original dot product. The bigger the value of the $\boldsymbol{d}$, maps the function to the higher is the dimension. The $\boldsymbol{d}$ of a kernel controls the flexibility of the classifier (Ben-Hur *et al.*, 2008). Normally $\boldsymbol{d} = \mathbf{2}$ is sufficient enough to discriminate between two classes with a good margin.

Another widely used kernel function is *Gaussian kernel* function, which can be defined as:

$$k_{\gamma}^{Gaussian}(\mathrm{x}, \mathrm{x}') = exp(-\frac{1}{\gamma}\|\mathrm{x} - \mathrm{x}'\|^2) \tag{4.6}$$

Here, γ is a hyperparameter, which controls the width of the *Gaussian kernel.* If γ is large, the data point has a non-zero value relative to any data points in the dataset. But as we decrease its value, the kernel becomes more local. A smaller value of γ gives a non-zero value of the discriminant function only in the close vicinity of each support vector (Ben-Hur *et al.*, 2008). If γ is very small there is a tendency for all the data points to be support vectors.

There is another kernel function called *sigmoid kernel* function which can be defined as:

$$k_{\alpha}^{sigmoid}(\mathrm{x}, \mathrm{x}') = \tanh(\alpha \mathrm{x} \cdot \mathrm{x}' + \delta) \tag{4.7}$$

Here, α(similar to d and γ) and δ works in the same way as previous kernel functions.

There are more kernel functions available (Laplacian kernel, ANOVA kernel, circular kernel, etc). But these four kernel functions have been discussed as I have used them in this thesis.

4.3 One-class Kernel Methods

Two-class SVM can be used where the training set is well specified, i.e. data either belongs to positive or negative class. But some data may not be as well characterised as is needed for the two-class SVM. In some problems one can be confident on the label of only one class of data and one-class classifier can be a better choice for classification in this case. One class classification is actually the special case of two-class classification problems (Alashwal *et al.*, 2006). In this thesis, the data we are using as positive examples (part of TFBSs) are experimentally verified and thus we have a certain level of confidence about them being positive examples. But the rest of the data may not belong to one negative class only. Actually we cannot be sure about the distribution of these data.

In one-class classification approach, data from only one class is available and this class is also better sampled than any other classes present in the dataset. The data, which are in well-sampled class, is called target class and others are known as outliers. There are many approaches for one-class classification. Schölkopf *et al.* (2001) proposed using traditional hyperplane method (One-class SVM). On the other hand, Tax & Duin (2004) suggested creating outlier uniformly in and around target class (Support Vector Data Description- SVDD). In this thesis I have used the one-class SVM and hence I am going to give the detailed description of the one-class SVM only.

4.3.1 One-class SVM

This method has been proposed in Schölkopf *et al.* (2001). In this method, the origin is treated as the only member of the outlier class. Then relaxation parameters are used to separate the target class from the origin. The main idea is to separate the surface region containing data from the region containing no data. The hyperplane is constructed maximally away from the origin. The points on the other side of the hyperplane are considered as positives. A function is constructed in such a way that it returns $+1$ when it captures those data from one class in the small region and returns -1 otherwise.

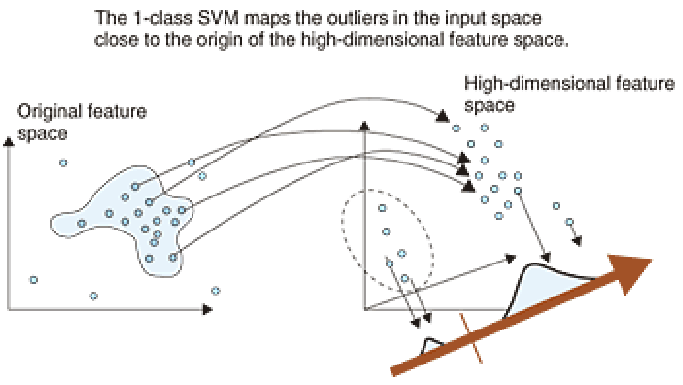

The 1-class SVM maps the outliers in the input space close to the origin of the high-dimensional feature space.
Original feature space
High-dimensional feature space

Figure 4.6: Concept of one-class SVM- Schölkopf's method (*Source:* Sudo et al. *(2008)*).

The algorithm can be summarised as follows:
Let $\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \cdots, \mathbf{x}_n\}$ are the training examples for a class $\boldsymbol{X}$ where, $\boldsymbol{X} \subset \Re^n$. If the mapping function is ϕ, then , where $\phi : \boldsymbol{X} \longrightarrow \boldsymbol{H}$ is the feature space. To separate the dataset from the origin we need to solve the following quadratic

programming problem:

$$\boldsymbol{min}\left[\frac{1}{2}\left\|\mathrm{w}\right\|^{2}+\frac{1}{\nu n}\sum_{i=1}^{n}\xi_{i}-\rho\right] \tag{4.8}$$

subject to: $\mathbf{w}\cdot\phi(\mathrm{x}_i)\geq\rho-\xi$, $i=1,2,\cdots,n$ and $\xi_i\geq 0$
The solution tends to the decision function,

$$D(\mathrm{x})=\boldsymbol{sign}(\mathrm{w}\cdot\phi(\mathrm{x})-\rho) \tag{4.9}$$

Here, **w**: weight vector
ν : upper bound of fraction on the outliers and lower bound on the fraction of the support vector and $\nu\in(0,1]$
ξ: slack variable to penalize misclassification
ρ: bias
n: number of examples
The sign of $D(x)$ will be positive for most of the examples in the training set.

As the representing data is coming from only positive examples, the small amount of data is not feasible to determine the boundary. Therefore, a huge amount of good sampled data is needed to be available (Manevitz & Yousef, 2002). This is one big problem with Schölkopfs one-class SVM. Another problem with Schölkopfs method is that it is sensitive to γ (for RBF kernel) and ν and also to the kernel selection. The classification performance of it completely depends upon a good selection of γ and ν. The performance also dramatically changes for different kernels.

4.3.2 LIBSVM: A popular SVM implementation

In the thesis, I have used LIBSVM (Chang & Lin, 2011) for implementing support vector machines. LIBSVM is a package, which contains implementations of different types of support vector machines. It has the implementation of both two-class and one-class SVM. It also implemented the four types of kernel functions that we discussed in Section 4.2.4. It also provides methods for model optimisation and scaling of the data. The one-class SVM of LIBSVM has actually implemented

Schölkopf's one-class SVM algorithm. Recently it has also implemented Tax and Duin's SVDD (Tax & Duin, 2004).

4.4 Model Optimisation

The biggest problems of trainable classifiers are over-fitting and under-fitting problems. Over-fitting occurs when the training model cares too much about misclassification and thus includes so many feature vectors that it cannot generalise the unseen data well. In case of over-fitting, the classification performance on the validation set is always very high. However, the prediction on unseen data is not as good as shown on the validation set. Therefore, over-fitting occurs when a learning algorithm is more accurate in fitting training examples, but less accurate in predicting new examples. Under-fitting basically occurs when the model cannot fit well on any data including training examples, which means the model does not match well with the underlying distribution of the data.

Figure 4.7(a) is an example of model over-fitting where the model cares too much about the misclassification and thus separates the data well in the training set. However, the separator (curved line in Figure 4.7(b)) cannot predict anything beyond the data points in the training set and therefore having a very bad classification performance on the test set. However, in Figure 4.8(a), the separator (straight line) is quite straight forward, as it does not care much about the misclassification. But as a result, it has failed to fully detect the supportive data points in the training data (Figure 4.8(b)). Here, the model under-fits the data. In both cases, the model will not be efficient enough to perform an efficient classification on unseen data.

Therefore, a suitable model optimisation process is needed to find the best model for classification. The cross-validation method used during training over a range of parameters can be an efficient way to avoid over-fitting and under-fitting. In this thesis, we have used cross-validation method from LIBSVM and also devised a *modified cross-validation* technique (described in Section 6.4.5). The standard cross-validation method (implemented in LIBSVM) is described in the next section.

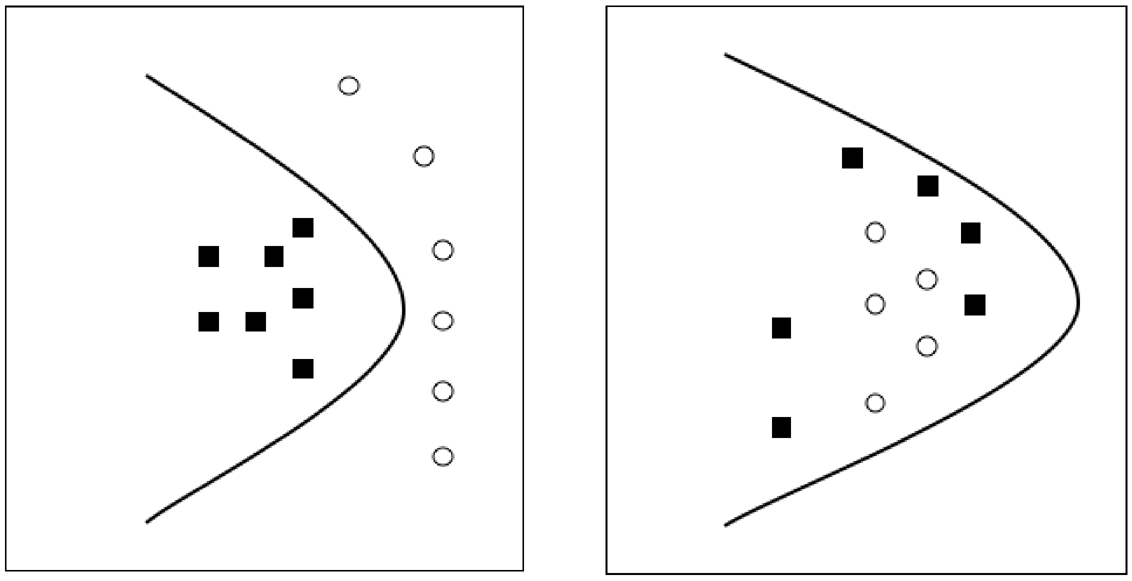

(a) (b)

Figure 4.7: Model over-fitting in SVM.

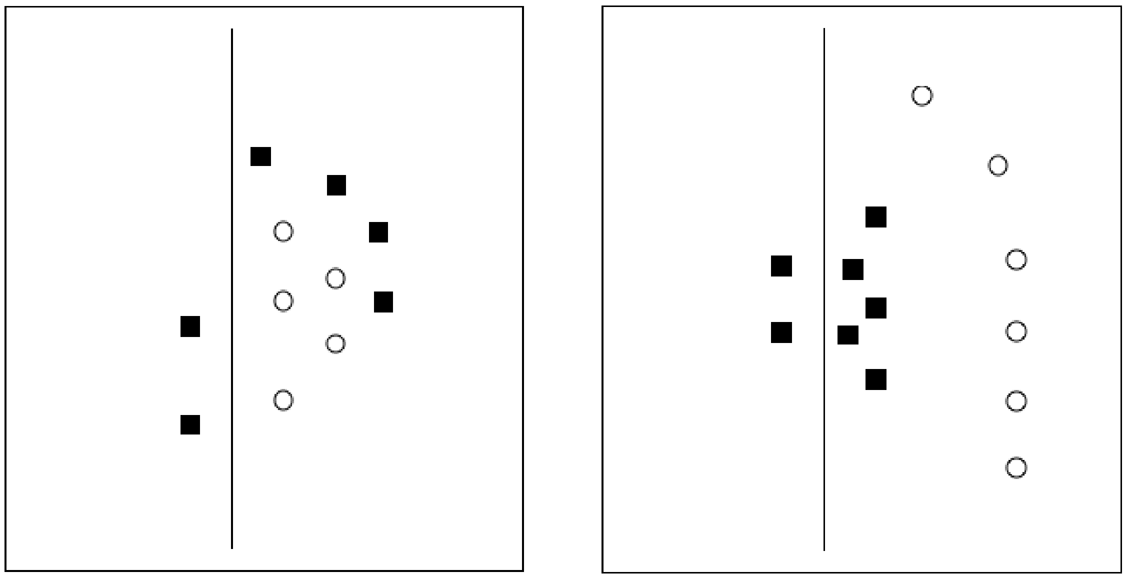

(a) (b)

Figure 4.8: Model under-fitting in SVM.

4.4.1 Cross-validation method

In this thesis we have used two types of cross-validation: i) cross-validation of the split of training and test set for the generalisation of classification results on dataset, and ii) cross-validation for finding the best hyper-parameter.

For the first criteria, we divided the data into three parts and took one part as test set and the other two parts combined as a training set. We alternate the different part of the dataset to get different training and test set. The test set will remain unknown to the model generated from the training set. For finding the best hyper parameters, we have done $\boldsymbol{v}$-fold cross validation, where the training set is further divided into $\boldsymbol{v}$ disjoint (no overlaps) parts and one of its subset is used as the validation set (for testing). The rest of the $(\boldsymbol{v}-\mathbf{1})$ subsets act as the training set. Each subset of the whole training set is predicted once so the cross-validation efficiency is the average performance measure. The average performance measure on the validation set can be based on *Accuracy*, *F-score*, or any other of the performance measures of the classification (discussed in Section 4.5.3).

The standard cross-validation, which had been used in earlier experiments (Sun *et al.*, 2005, 2006a,b, 2007, 2008, 2009a,b; Robinson *et al.*, 2006, 2007a,b, 2008), may not be efficient enough for generalisation due to the dissimilar nature of validation sets and test sets. Therefore, a modified cross-validation method has been devised in this work. In this new method, the training set is created after some pre-processing and the validation set is taken from the original biologically meaningful dataset. A further discussion on this method has been given in Section 6.4.5.

4.4.2 Finding best hyperparameters

A good classification performance of an SVM depends on the selection of the hyper parameters (discussed in Section 4.2.4). To find an optimum model, an exhaustive search on the parameters has been done. It tries the values of a set of parameters across a specific range (known as a Grid Search, Chang & Lin (2011)). It is quite straightforward through a naïve process (Chang & Lin, 2011). For each different hyper parameter set, the validation rate is calculated and the hyper parameters

are picked based on the best average validation rate. For example, for Gaussian kernel function, there are two hyper-parameters and . The best pair of $\boldsymbol{C}$ and $\boldsymbol{\gamma}$ is picked from an exponentially growing sequence of hyperparameter sets based on the best validation rate. This parameter search helps to evaluate fitting provided by a set of parameter values using cross-validation. The limitation of this kind of search is it takes a lot of computational time if we want to search a big range of hyper parameter values.
The pseudocode for finding the best hyper parameters is given below:

Pseudocode 1 Finding the best hyper-parameters

```
 1: Split the training data into $\boldsymbol{v}$ partitions
 2: This gives $\boldsymbol{v}$ different training sets and the corresponding validation sets
 3: for each of the training set do
 4:     Pre-process the data to produce balanced training set
 5: end for
 6: for each combination of hyper-parameter values do
 7:     for each of the pre-processed $\boldsymbol{v}$ training sets do
 8:         Train an SVM
 9:         Measure performance on the corresponding validation set
10:     end for
11:     Average the Performance Measure
12: end for
13: Choose the combination of hyper-parameter with the best average
```

4.5 Imbalanced Data

When training a supervised classification algorithm on a dataset it is important to consider the proportional representation of the various features being considered. If a label of interest is significantly under-represented within the dataset, we can call the dataset imbalanced. The datasets used in this thesis are imbalanced in nature. Therefore, now I am going to discuss the problem of using the imbalanced data in a supervised learning algorithm (*i.e.* an SVM) and the sampling techniques to overcome it.

4.5.1 Problems with imbalanced data

If a dataset is imbalanced, then training is likely to result in a classifier that has been over-trained on the majority class and can only act as a weak classifier for the minority class (Chawla *et al.*, 2002, 2003; Japkowicz, 2003). The output of the classifier will tend to represent the majority class rather than minority class, which may in turn produce poor performance on the test set. In the dataset we have used, very few of the base pairs are positive examples (binding sites) and the rest of them are negative examples. This made the dataset imbalanced which is not suitable for supervised learning technique and it will over predict the majority class - the non-binding sites. In order to compensate for imbalanced data it is often possible to sample from the original dataset in such a way so as to provide a new dataset that can be used to train the classifier more efficiently.

4.5.2 Sampling

In Japkowicz (2003) it was shown that oversampling by simply repeating elements of the minority class might not improve the recognition of this class. So a data based sampling method (Chawla *et al.*, 2002; Radivojac *et al.*, 2004) had been chosen in which the minority class (here, binding site examples) were over-sampled and majority class (non-binding site examples) were under-sampled. The Synthetic Minority Over-sampling TEchnique (SMOTE) (Chawla *et al.*, 2002) had been previously used with this data to over-sample the minority class in the training set (Sun *et al.*, 2006a,b, 2007, 2008, 2009a,b; Robinson *et al.*, 2006, 2007a,b, 2008) . In this method, nearest neighbors are identified for each data point and thus new instances were created using a Heterogeneous Value Difference Metric (HVDM) (Wilson & Martinez, 1997).

For features that are continuous in nature the *Euclidean distance* function is used for identifying $\boldsymbol{K}$-nearest neighbour. There are other distance functions available for such uses, like Camberra, Chebychev, Quadratic, Correlation, and Chi-square distance metrics (Michalski *et al.*, 1981; Diday, 1974); hyperrectangle distance functions (Salzberg, 1991; Domingos, 1995); the Context-Similarity measure (Biberman, 1994); the Contrast Model (Tversky, 1977); Minkowsky (Batchelor, 1978); Mahalanobis Distance (Nadler & Smith, 1993), etc. However, *Eu-*

clidean distance function is widely used and feasible for continuous data, which is defined as:

$$\boldsymbol{Eculidean\ distance} = \sqrt{\sum_{d=1}^{m} (x_d - y_d)^2} \tag{4.10}$$

Here, **x** and **y** are two input vectors and $\boldsymbol{m}$ is the number of the attributes.
But *Euclidian distance* is not feasible for the data with nominal attributes. So HVDM is used to calculate the distance between two input vectors which may have both continuous and nominal values. It can be defined as follows:

$$\boldsymbol{HVDM}(\mathrm{x}, \mathrm{y}) = \sqrt{\sum_{a=1}^{m} d_a{}^2(x_a, y_a)} \tag{4.11}$$

Here, **x** and **y** are two input vectors with number of attributes $\boldsymbol{m}$ and $\boldsymbol{d_a}(\mathrm{x}, \mathrm{y})$ is a function that returns distance.

$$d_a(\mathrm{x}, \mathrm{y}) = \begin{cases} \boldsymbol{normalized\ vdm_a(x_a, y_a)}, & \text{if } a \text{ is nominal} \\ \boldsymbol{normalized\ difference_a(x_a, y_a)}, & \text{if } a \text{ is continuous} \end{cases} \tag{4.12}$$

In Equation 4.12, VDM is the Value Difference Metric (Stanfill & Waltz, 1986). For two values x and y of an attribute a, the simplified version of VDM is as follows:

$$\boldsymbol{vdm_a(x, y)} = \sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^q \tag{4.13}$$

Here, $\boldsymbol{N_{a,x}}$: the number of instances in the training set T that have value $\boldsymbol{x}$ for attribute $\boldsymbol{a}$;
$\boldsymbol{N_{a,x,c}}$: the number of instances in T that have value $\boldsymbol{x}$ for attribute $\boldsymbol{a}$ and output class $\boldsymbol{c}$;
$\boldsymbol{C}$: the number of output classes in the problem domain;
$\boldsymbol{q}$: a constant, usually 1 or 2;

For example, we have two classes of data, each containing nucleotides, which can be part of either a binding site or a non-binding site. Here, nucleotides that

are part of non-binding sites will be denoted as $\boldsymbol{x}$ and nucleotides that are part of binding sites will be denoted as $\boldsymbol{y}$.
There are two classes of data:
$\boldsymbol{C1} = \mathbf{4}(\boldsymbol{x}), \mathbf{1}(\boldsymbol{y})$, 4 nucleotides that are the part of non-binding sites and 1 nucleotide that is the part of binding sites.
$\boldsymbol{C2} = \mathbf{2}(\boldsymbol{x}), \mathbf{6}(\boldsymbol{y})$, 2 nucleotides that are the part of binding sites and 6 nucleotide that is the part of non-binding sites.
For this problem the VDM is calculated as follows:
$\boldsymbol{vdm_a}\,(\mathrm{x,y}) = \left|\frac{4}{6} - \frac{1}{7}\right| + \left|\frac{2}{6} - \frac{6}{7}\right| = \frac{22}{21}$

After calculating the distance using the HVMD, the oversampling is done using the following steps: (i) First, we search for its $\boldsymbol{K}-$nearest neighbours in the minority using above-mentioned methods. Since the dataset is a mixture of continuous and binary features, we took the following measures as suggested in SMOTE.
(ii) For continuous features, a new feature value denoted by $\boldsymbol{x_d}^{\boldsymbol{new}}$ is given by:

$$\boldsymbol{x_d}^{\boldsymbol{new}} = \boldsymbol{x_d}^{\boldsymbol{n}} + \boldsymbol{rand}(\mathbf{0},\mathbf{1}) \times (\boldsymbol{x_d}^{\boldsymbol{n}} - \boldsymbol{x_d}^{\boldsymbol{NN}}) \tag{4.14}$$

where, the difference of each feature between the pattern $(\boldsymbol{x_d}^{\boldsymbol{n}})$ and its nearest neighbor $(\boldsymbol{x_d}^{\boldsymbol{NN}})$ is taken, and then multiplied by a random number between 0 and 1, and added to the corresponding feature of the pattern.Here $\boldsymbol{x_d}^{\boldsymbol{NN}})$ is calculated based on the HVDM functions mentioned earlier.
(iii) For binary features, the majority voting principle is applied to each element of the $\boldsymbol{K}$-nearest neighbours in the feature space.

On the other hand, a randomly selected subset of data points from the majority class was selected for under-sampling. The SMOTE technique significantly improves minority class recognition compared with just oversampling done by replacement (Chawla *et al.*, 2002).

4.5.3 Confusion Matrix and Performance Measures

Confusion matrix is a visualisation tool used for supervised learning like SVM. Each row of confusion matrix represents the prediction class and each column represents the actual class. A confusion matrix has the following entries:

True Negative ($\boldsymbol{TN}$) : correct predictions that an instance is negative.

False Positive ($\boldsymbol{FP}$) : incorrect predictions that an instance is positive.

False Negative ($\boldsymbol{FN}$) : incorrect predictions that an instance is negative.

True Positive ($\boldsymbol{TP}$) : correct predictions that an instance is positive.

	Predictive Negatives	Predictive Positives
Actual Negatives	True Negatives (TN)	False Positives (FP)
Actual Positives	False Negatives (FN)	True Positives (TP)

Table 4.1: Confusion Matrix.

There are different performance measures that can be used to measure the efficiency of a classifier, which are as follows:

Recall is the proportion of positive cases that were correctly identified.

Precision is the proportion of the predicted positive cases that were correct.

F-score is the harmonic mean of *Recall* and *Precision*.

False Positive Rate is he proportion of negatives cases that were incorrectly classified as positive.

Accuracy is the proportion of the total number of predictions that were correct.

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}} \tag{4.15}$$

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}} \tag{4.16}$$

$$\boldsymbol{F\text{-}score} = \frac{\boldsymbol{2 \times Recall \times Precision}}{\boldsymbol{Recall + Precision}} \tag{4.17}$$

$$\boldsymbol{FP\text{-}rate} = \frac{\boldsymbol{FP}}{\boldsymbol{FP + TN}} \tag{4.18}$$

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + FN + FP + TN}} \qquad (4.19)$$

Accuracy (correct classification) can be an ideal performance measure to report the efficiency of a classifier. But as we are dealing with an imbalanced dataset, simply using *Accuracy* as the performance measure is not appropriate. Predicting everything as not belonging to a binding site will give a very good *Accuracy* rate. *Recall*, *Precision*, *F-score*, and *FP-rate* derived from the confusion matrix are more important for the classification problem used in this thesis.
For example, there are 1200 base pairs and amongst these 10 base pairs are annotated as binding sites. If the classifier predicts all the base pairs as non-binding sites, the performance measures are calculated as follows:

$\boldsymbol{TP = 0}$; $\boldsymbol{TN = 1190}$; $\boldsymbol{FN = 10}$; and $\boldsymbol{FP = 0}$
In this case, Recall = 0, Precision = infinite, F-score = infinite, and FP-rate = 0

But, Accuracy = 0.99 even though the other measures showing bad performance. So for this kind of problem Accuracy is not a very suitable performance measure.

In previous studies(Sun *et al.*, 2005, 2006a,b, 2007, 2008, 2009a,b; Robinson *et al.*, 2006, 2007a,b, 2008), it was found that most of the original algorithms have high *Recall* and this is possibly caused by simply over predicting the binding sites.
In the previous example, if we predict everything as a binding site, then

$\boldsymbol{TP = 10}$; $\boldsymbol{TN = 0}$; $\boldsymbol{FN = 0}$; and $\boldsymbol{FP = 1190}$
In this case, Recall = 1.0, Precision = 0.0008, F-score = 0.00016, and FP-rate = 1.0

It shows measuring the performance of our meta-classifier with just *Recall* is not correct. It is easy to get high *Recall* by predicting everything as binding site. On the other hand, *Precision* can be a very good measure for measuring classifier performance, as it is the proportion of actual predictive samples from the binding sites. Increasing *Precision* value can be one of our main goals, but increase of *Precision* occurs at the cost of decrease in *Recall*. Again, it is easy to get high *Precision* by predicting nothing as a part of binding sites. If in the previous example, the confusion matrix becomes as follows,

$\boldsymbol{TP = 9}$; $\boldsymbol{TN = 1180}$; $\boldsymbol{FN = 4}$; and $\boldsymbol{FP = 7}$
In this case, Recall = 0.69, Precision = 0.56, F-score = 0.62, and FP-rate = 0.006

But if the number of false-positive decreases, the Precision will increase at the cost of Recall.

So, for $\boldsymbol{TP = 9}$; $\boldsymbol{TN = 1180}$; $\boldsymbol{FN = 8}$; and $\boldsymbol{FP = 3}$
Recall = 0.53, Precision = 0.75, F-score = 0.62, and FP-rate = 0.002

Both of the classifiers are good and the *F-score* values in both cases are better, though the *FP-rate* for the later example is less than the previous one.

Therefore, taking account of both *Recall* and *Precision* using the *F-score* should give a good measure of classification performance since the *F-score* is actually a kind of weighted average of *Recall* and *Precision*. In addition reducing the *FP-rate* should also be another major concern to verify a classifier's performance.

4.6 Summary

In this chapter, I have presented a brief discussion on various classification methods that have been used in this thesis. The main aim of the discussion was to introduce different Support Vector Machine techniques as well as different issues on the datasets that might decrease classification efficiency. The issue on imbalanced dataset has been addressed and the techniques for processing this type of data properly has been discussed. The databased technique has been described as the remedy to the problem of imbalanced data and an oversampling technique (SMOTE) has been explained thoroughly. The performance measure for the classification has been described along with the rationale behind the selected performance measure that would be ideal for analysing the results. I have also mentioned about a new cross-validation technique that has been employed in different experiments along side with the standard cross-validation technique implemented in LIBSVM and this will be discussed thoroughly in later chapters.The results from the new cross-validation method may potentially be an interesting outcome of the thesis.

Chapter 5

Description of Datasets

5.1 Introduction

The generation of annotated sequences for the testing and evaluation of computational transcription factor binding site predictions is a crucial, non-trivial step. To generate a high quality annotated promoter dataset, it is essential to select an organism which have a large set of sequences and are associated with high quality, experimentally determined annotations of *in vivo cis*-regulatory binding sites. Therefore it is important that any annotation used should be experimentally verified. However, experimental determination of *cis*-regulatory binding sites is currently an expensive and time-consuming process. Such functional annotation of non-coding sequences is typically only available for particularly well-studied model species. A number of species, such as *E. coli*, *S. cerevisiae*, *D. melanogaster* or *M. musculus*, etc., have not only had their genome sequenced, but have also been the subject of intense regulatory characterisation. But it is important to understand that even for the best studied systems there is no assurance that this characterisation has incorporated all biologically relevant binding sites.

Many computational algorithms can make implicit assumptions about the type of *cis*-regulatory organisation found in the model organism. There are algorithms that use clustering of predictions to indicate a regulatory module. However, it is not yet clear to what extent the additional levels of complexity found in the *cis*-regulatory regions of advanced multi-cellular organisms will be amenable to different computational approaches for predictions. These computational approaches for transcription factor binding site predictions are also prone to many false predictions.

This chapter and the next describe the base prediction algorithms used in this thesis and most of the initial experiments undertaken while trying to find improvements in prediction accuracy. Both yeast data and mouse data used in this chapter has been inherited from previous studies (Sun *et al.*, 2005, 2006a,b, 2007, 2008, 2009a,b; Robinson *et al.*, 2006, 2007a,b, 2008). Both of these datasets will be discussed in this chapter. This chapter then continues with an analysis of the prediction accuracy of the base algorithms used separately while the next chapter considers the combination of the base algorithms analysed via an SVM. I have used both of the datasets presented here with new experiments described in Chapter 7, and Chapter 8 will introduce a newer and more up to date dataset and the experiments performed with it.

5.2 Choice of Experimental Organism

A simple and well-suited regulatory system is preferable for the initial test of proof of concept of the approaches presented in this thesis. The first studies, exploring the biochemical mechanisms on gene regulation, were conducted in *E. coli* (Jacob & Monod, 1959; Ptashne & Gann, 2002) and therefore can be an obvious choice in this context. *E. coli* has a compact genome and this genome is also one of the best-annotated genomes currently available. Its promoters are small and simple and found immediately upstream of the gene or operon (a cluster of genes regulated by single promoter). It also does not have any modular organisation of the binding sites. Furthermore, bacterial genes are typically regulated by only 2-3 TFBS per gene and these sites are contained within 200-300 bps of the sequence. Therefore, finding TFBSs in a prokaryotic model has significantly reduced challenge compared with eukaryotes.

However, moving from a prokaryotic model to a eukaryote model does not represent a simple extension of basic regulatory principles. The organisation of both the genome and the cellular environment is fundamentally different in this case. Given these considerations, the eukaryote yeast (*S. cerevisiae*) has been chosen as the first experimental organism. Yeast has the most completely annotated sequence and is therefore a more appropriate model for initial evaluation of binding site predictions. It has many of the advantages of *E. coli* as a model organism. It has a small compact genome with a typical size for genes regulatory sequence of approximately 500 bps. It has also been well studied, and experimentally annotations are readily available from various public databases (Zhu & Zhang, 1999; Montgomery *et al.*, 2006). It is particularly rich in information such as co-regulated clusters of genes or orthologous sequences (sequence from genes that have evolved directly from an ancestral gene) for phylogenetic footprinting, which is used by a number of algorithms used in this thesis. Thus it represents a good initial model for testing approaches where there is a possibility for expanding the work to the more complex regulatory organisation found in eukaryotic multi-cellular organisms.

As mentioned binding sites differ from one species to another, some organisms have a much more complex organisation of their gene regulatory regions, which makes the positions of their binding sites more difficult to predict than yeast data. Unlike in yeast, the location of binding sites may not be proximal to the promoter site and can even be thousands of base pairs away, both upstream and downstream as well as inside intronic regions. Complex organisms can have a number of other biological features in the non-coding sequence, which are not related to gene regulation or transcription factors. I have therefore chosen a complex multi-cellular organism, mouse (*M. musculus*), as the second model organism to validate my method. The mouse genome has all of the above properties and in addition it has more non-coding DNA sequences than the yeast genome.

5.3 Selection of Dataset

Generation of appropriate datasets for use in evaluating the performance of binding site prediction algorithms is a challenging problem with no clear solution (Tompa *et al.*, 2005). As mentioned before, promoter sequences that have been experimentally annotated are commonly used in this case with no assurance of the completeness of sequence annotations. In the initial experiments, to determine suitable approaches to improve binding site predictions, I used an existing yeast dataset (Sun *et al.*, 2005, 2006a,b, 2009a,b; Robinson *et al.*, 2006, 2007a,b, 2008). This consisted of 112 annotated promoter sequences, which were selected for training and testing the algorithms, a total of 67,782 bps of sequence data. These 112 annotated promoter sequences were extracted from the *S. cerevisiae Promoter Database* (SCPD) (Zhu & Zhang, 1999). For each promoter, 500 bps of sequence taken immediately upstream from the transcriptional start site were considered sufficient to typically allow full regulatory characterisation in yeast (Zhu & Zhang, 1999). If annotated binding sites lay outside of this range, then the range was expanded accordingly. Likewise, where a 500 bps upstream region would overlap a coding region then it was truncated accordingly. See Table 5.1 for details.

Total number of sequences	112
Total sequence length	67,782 bps
Average sequence length	605 bps
Average number of TFBSs per sequence	3.6
Average TFBSs width	13.2 bps
Total number of TFBS	400
TFBS density in total dataset	7.8%

Table 5.1: A summary of the yeast dataset.

I used an existing mouse dataset (Sun *et al.*, 2007, 2008; Robinson *et al.*, 2008) as well.The mouse dataset was constructed from the *ABS* (Blanco *et al.*, 2006) and *OregAnno* (Montgomery *et al.*, 2006) databases. 47 annotated promoters sequences were taken with TBFS for mouse from these databases and merged together into a single dataset.

The sequence length in base pairs extracted from ABS is typically 500 bps and those from *OregAnno* are around 2000 bps in length. Most of the promoters are upstream of their associated gene and a few of them are extended over the first exon including intronic regions. There are 60851 nucleotides in total in the dataset. See Table 5.2 for details.

Total number of sequences	47
Total sequence length	60851 bps
Average sequence length	1294.70 bps
Average number of TFBSs per sequence	2.87
Average TFBSs width	12.78 bps
Total number of TFBS	135
TFBS density in total dataset	2.85%

Table 5.2: A summary of the mouse dataset.

Having constructed the datasets, two-third of each dataset (both yeast and mouse) has been used as training set in the experiments that are discussed in Section 6.4. The other one-third of the data has been used as a test set. In this chapter, this biologically meaningful test set has been used, that is the full test set, since I am not constructing vectors of data and not training a classifier. This has been done to allow comparison between the results in this chapter and those in Chapter 6.

5.4 Description of the Algorithms Used

A full range of computational approaches to the binding site prediction problem has already been presented in Chapter 3. This chapter describes a wide diversity of binding site prediction algorithms selected for the analysis. Most of the selected algorithms were published and well established in the bioinformatics research community. The exceptions were a small number of algorithms, which were either developed in-house or by collaborating institutions.

The different algorithmic strategies are dependent on the values of various parameters. These parameters being expected to relate in differing ways to the underlying organisation of the DNA sequences being analysed. A previous study (Robinson, 2006) showed that attempting to optimise the parameters had little or no effect on the performance of the integrated process. Hence default parameter values taken from the literature were used in this study. These parameter values are therefore already selected to be good values in the literature. It was important that this study was carried out using a wide range of different algorithmic strategies as possible. The intention was to feed the output from these evidences into the integration process described in Chapter 6, for which again, maximising the diversity of the set of prediction strategies was a key requirement.

5.4.1 Algorithms used for the yeast data

For yeast, the selected algorithms were typically taken from the literature although some were developed in-house or by the collaborators as mentioned in the previous section. Table 5.3 lists the algorithms used with the yeast dataset. Parameter settings for the algorithms were taken from the literature, if not available, default settings were used.

Strategy	Algorithms
Scanning Algorithms	Fuzznuc MotifScanner Ahab
Statistical Algorithms	PARS Dream (2 versions) Verbumculus
Co-regulatory Algorithms	MEME AlignACE Sampler
Evolutionary Algorithms	SeqComp Footprinter

Table 5.3: The 12 Prediction Algorithms used with the yeast dataset.

Fuzznuc [1] , *MotifScanner* (Thijs *et al.*, 2001, 2002) and *Ahab* (Rajewsky *et al.*, 2002) were chosen as scanning algorithms (described in Section 3.3.1). *Fuzznuc*, developed as a part of EMBOSS bioinformatics software analysis package, is a simple scanning algorithm. It performs a regular expression search, using IUPAC (International Union of Pure and Applied Chemistry) codes (discussed in Section 3.2.1), on a DNA sequence for the set of provided consensus motifs. *Fuz-*

[1]http://www.hgmp.mrc.ac.uk/Software/EMBOSS/

znuc does not use any model of the background sequence and allows a number of user-defined mismatches.

MotifScanner, a part of the INCLUSive [1] suite (Thijs *et al.*, 2002) of bioinformatics tools, searches for the sequences that have a high likelihood score given a weight matrix motif representation (described in Section 3.2.2) along with higher order Markov model (described in Section 3.2.5). Unlike *Fuzznuc*, it uses the model of the background sequences for the Markov model. Finally, *Ahab* is an algorithm designed to search for both enhancers and binding sites in the genome of multi-cellular organisms. *Ahab* generates a local Markov model of the background within a sliding window and then within that window it searches for matches to a weight matrix motif model.

The statistical algorithms (described in Section 3.3.2) selected for the single sequence analysis in this study were *PARS* [2], *DREAM* (Abnizova *et al.*, 2006), and *Verbumculus* [3] (Apostolico *et al.*, 2000, 2004). *PARS* is a heuristical algorithm designed specifically to search for patterns exhibiting the kinds of symmetry that might be associated with dimerically (in which two macromolecules, such as proteins or nucleic acids, bind by non-covalent bonds) binding transcription factors. *PARS* was developed by one of my supervisors, Dr. Mark Robinson (Robinson, 2006). *DREAM* (Detection of Regulatory Elements and Modules) is an algorithm developed in collaboration with Dr. Irina Abnizova. *DREAM* first generates a local Markov model (described in Section 3.2.5) of the background sequence within a sliding window and then searches for clusters of words within the window, which are significantly unlikely to occur given the background model. Finally, The statistical algorithm *Verbumculus* searches for over- and under-represented words in a sequence. It uses a suffix tree (a data structure that presents substring of a given string as nodes) to identify statistically unlikely patterns in a computational efficient manner and thus avoids the scaling issues that arise when enumerating all words over a range of sizes in large sequences

AlignACE (Roth *et al.*, 1998; Hughes *et al.*, 2000) , *MEME* (Bailey & Elkan, 1995), and *Sampler/ Mosta* (Reiss & Schwikowski, 2004) are the co-regulatory

[1]http://homes.esat.kuleuven.be/ sistawww/bioi/thijs/download.html
[2]http://sourceforge.net/projects/pars
[3]http://www.cs.ucr.edu/ stelo/Verbumculus

algorithms (described in Section 3.3.3) used in this study. *AlignACE*, an implementation of a Gibbs sampling algorithms (Lawrence *et al.*, 1993) (described in Section 3.3.3.1), allows searches for multiple different binding motifs within a set of sequences by using an iterative masking procedure. Motifs located on the complementary strand are also included in the search. *MEME* is an extension of the Expectation maximisation algorithm (described in Section 3.3.3.2). *MEME* incorporates a heuristic for selecting an optimal start point preventing convergence to a locally optimal solution. Finally, Sampler is another algorithm that uses Gibbs sampling algorithm and included in the set of algorithms as part of the collaboration (with Institute for Systems Biology). Sampler, being a part of the *netmosta* motif searching tools, incorporates a higher order background model and requires that the number of matching motifs fall within a predetermined range.

Two phylogenetic algorithms(described in Section 3.3.4) that were included in this study are *SeqComp* (Brown *et al.*, 2002) and *Footprinter* (Blanchette & Tompa, 2003). *SeqComp* is a simplistic pair-wise comparison algorithm that compares two sequences and detects if they contain any stretches of a predetermined size. A *similarity threshold* value is used in this algorithm (Brown *et al.*, 2002). It was designed specifically to search for regulatory modules rather than individual sites. *Footprinter* is a multiple alignment (described in Section 2.4) algorithm that identifies the best-conserved motifs in a set of homologous sequences (sequences with highly similarity). The phylogeny of the homologous sequences is used to factor in the expected evolutionary distance and therefore divergence between the sequences, which enables more accurate predictions.

5.4.2 Algorithms used for the mouse data

Seven sources of evidence were used as input in this study. Table 5.4 lists the algorithms used with the mouse dataset. A number of sources of evidence were extracted from the UCSC genome bioinformatics website [1] (Karolchik *et al.*, 2003).

[1]http://genome.ucsc.edu/cgi-bin/hgGateway

Strategy	Algorithms
Scanning Algorithms	MotifLocator EvoSelex
Evolutionary Algorithms	Regulatory Potential PhastCons (Conserved) PhastCons (Most Conserved)
Indirect Evidence	CpGIsland
Negative Evidence	Exon

Table 5.4: The 7 Prediction Algorithms used with the mouse dataset.

The two scanning algorithms included in this study are: *MotifLocator* (Thijs *et al.*, 2001, 2002) and *EvoSelex*. *MotifLocator* scans using the *PHYLOFACTS* matrices from the *JASPAR* database (Sandelin *et al.*, 2004) (mentioned in Section 3.2.2). The 174 matrices were assembled from the paper Xie *et al.* (2005). A scanning procedure is designed to produce a distribution for each individual matrix against random sequence and selected threshold values. This reduces the number of potential false positive predictions. The *Evoselex* algorithm uses a simple wrapper around *Fuzznuc* (described in the previous section) to identify motifs assembled from the paper Ettwiller *et al.* (2005) and these motifs do not have any degeneracy in the consensus sequences.

Three phylogenetic evidence namely *Regulatory Potential* (Kolbe *et al.*, 2004) and two versions of *PhastCons* (conserved and most-conserved) (Siepel & Haussler, 2004; Siepel *et al.*, 2005) are used. *Regulatory Potential* (RP) is used to compare frequencies of short alignment patterns between known regulatory elements and neutral DNA. The *RP* scores were calculated using alignments from the mouse, rat, human, chimpanzee, macaque, dog, and cow. One the other hand, *PhastCons* is an algorithm that computes sequence conservation from multiple alignments using a phylo-HMM strategy (mentioned in Section 3.2.5). It is the part of the *PHAST* (PHylogenetic Analysis with Space/Time models) package. The algorithm was used with two levels of stringency: conserved and most conserved, which are used as separate sources of evidence.

The *CpGIsland* algorithm is a kind of indirect evidence of existence of regulatory regions in the genome region (mentioned in Section 2.4). The *CpGIsland* algorithm finds CG sequences in the regulatory that are typically found near transcription start sites and are rare in vertebrate DNA. The scores were obtained from the *UCSC* genome browser custom track which calculates the ratio of observed to expected CpGs (Gardiner-Garden & Frommer, 1987). Finally, *Exon* predictions are included for those sequences where the sequence extends over the first exon and into the next intronic region and should be considered a type of negative evidence.

5.5 Statistics of the Algorithms

This section will detail the results obtained during the course of this research. A critical analysis of the results will be given in the discussion in Section 5.6. The performance of each algorithm was calculated by comparing the prediction at each individual nucleotide position in the sequence with the annotated values. In this way the frequencies of the four possible outcomes at a given sequence position could be calculated, both across an individual sequence and the entire dataset:

1. Binding site predicted and also annotated in the database: True Positive
2. Binding site predicted but not annotated in the database: False Positive
3. Binding site not predicted and also not annotated in the database: True Negative
4. Binding site not predicted but is annotated in the database: False Negative.

This then allowed the calculation of the statistical measures detailed in Section 4.5.3.

5.5.1 Algorithm performance for the yeast dataset

A full evaluation of the baseline performance of the twelve algorithms used in the case of yeast is obviously a necessary prerequisite for any comparative analysis. Table 5.5 contains the details of the performance of each of the algorithms using the range of statistics chosen to explore the different aspects of classification performance for each of the algorithms. These statistics were calculated based on performance across the test dataset only and taken from Robinson (2006). *Fuzznuc* achieves the best *F-score* followed by *MotifScanner* and *MEME*. *MEME* also achieves the lowest *FP-rate* value.

Algorithm	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate	Accuracy
Fuzznuc	683	2213	962	18655	0.415	0.236	0.301	0.098	0.859
MotifScanner	448	1682	1197	19186	0.272	0.210	0.237	0.075	0.872
Ahab	1108	10806	537	10062	0.674	0.093	0.163	0.480	0.496
PARS	189	1551	1456	19317	0.115	0.109	0.112	0.069	0.866
Verbumculus	349	2545	1296	18323	0.212	0.121	0.154	0.113	0.829
Dream(over)	472	5273	1173	15595	0.287	0.082	0.128	0.234	0.714
Dream(under)	474	5967	1171	14901	0.288	0.074	0.117	0.265	0.6843
Sampler	78	1489	912	20034	0.079	0.049	0.061	0.069	0.893
MEME	262	1305	789	20157	0.249	0.167	0.200	0.061	0.907
AlignACE	174	1393	837	20109	0.172	0.111	0.135	0.065	0.901
SeqComp	352	1215	3421	17525	0.093	0.225	0.131	0.065	0.794
Footprinter	460	1107	4974	15972	0.085	0.294	0.131	0.065	0.729

Table 5.5: Performance measures of sources of evidence on the yeast data.

Figure 5.1 illustrates the variation in *Precision*, *Recall* and *F-score* across the different algorithms. Note that larger values are preferable for all of these measures.

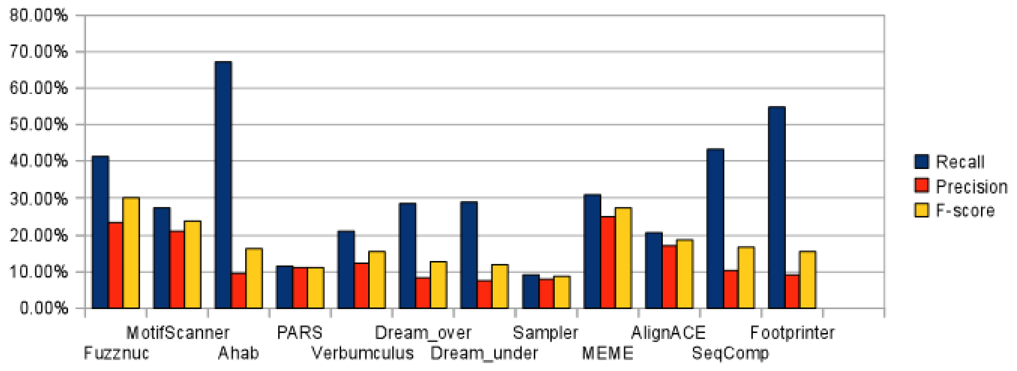


Figure 5.1: Comparison between *Recall*, *Precision* and *F-score* from different base algorithms on the yeast data.

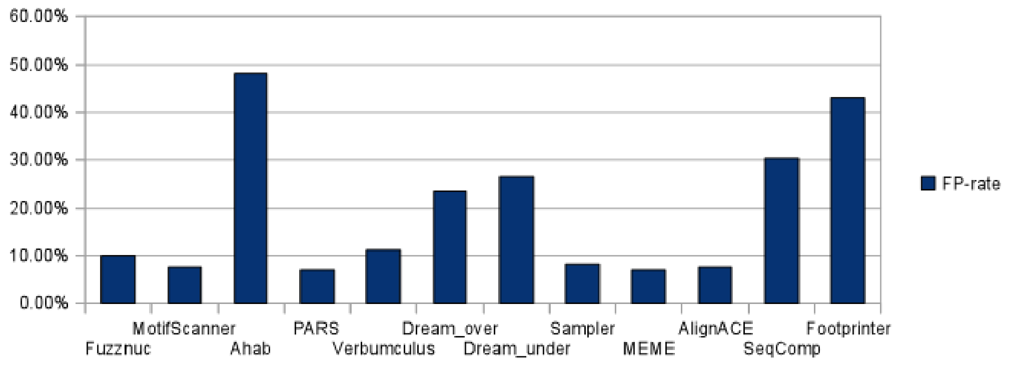


Figure 5.2: Comparison between *FP-rate* from different base algorithms for the yeast data.

Figure 5.2 shows the *FP-rate* scores for each of the algorithms. Smaller values are to be preferred for this measure.

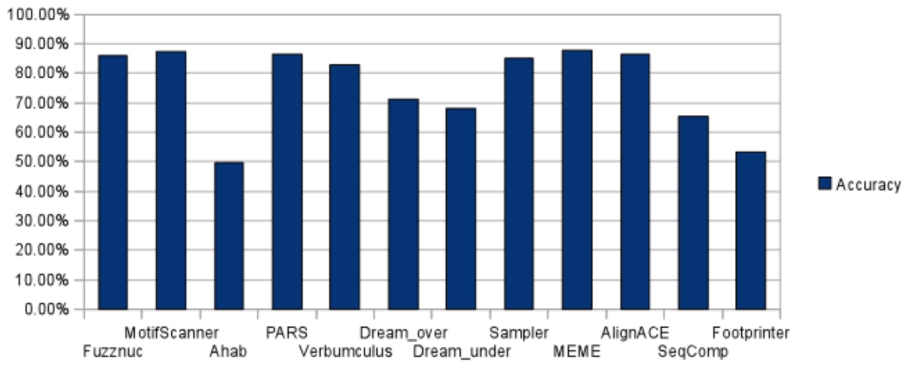


Figure 5.3: Comparison between *Accuracy* from different base algorithms for the yeast data.

Figure 5.3 shows the *Accuracy* scores for each of the algorithms. Larger values are to be preferred for this measure.

5.5.2 Algorithms performance for the mouse dataset

Like yeast, a full evaluation of the baseline performance of the seven algorithms used in case of mouse has been undertaken for comparative analysis. Table 5.6 contains details of the performance of algorithms for the mouse data using the range of statistics chosen to explore the different aspects of classification performance for each of the algorithms. These statistics were calculated based on performance across the test dataset only. *PhastCons (conserved)* achieves the best *F-score* closely followed by *EvoSelex*. However, *EvoSelex* has better *FP-rate* than *PhastCons (conserved)*. *CpG Island* and *Regulatory Potential* achieve the lowest *FP-rate* value.

Algorithm	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate	Accuracy
MotifLocator	333	4385	451	13739	0.425	0.071	0.121	0.242	0.744
EvoSelex	273	3139	511	14985	0.348	0.080	0.130	0.173	0.807
Regulatory Potential	52	455	732	17669	0.066	0.103	0.081	0.025	0.937
PhastCons (conserved)	526	6560	258	11564	0.668	0.074	0.134	0.362	0.639
PhastCons (most conserved)	157	1686	627	16438	0.200	0.085	0.119	0.094	0.878
CpG Island	27	436	757	17688	0.034	0.058	0.043	0.024	0.937
Exon	26	855	758	17269	0.033	0.029	0.031	0.047	0.915

Table 5.6: Performance measures of sources of evidence on the mouse data.

Figure 5.4 illustrates the variation in *Precision*, *Recall* and *F-score* across the different algorithms. Note that larger values are preferable for all of these measures.

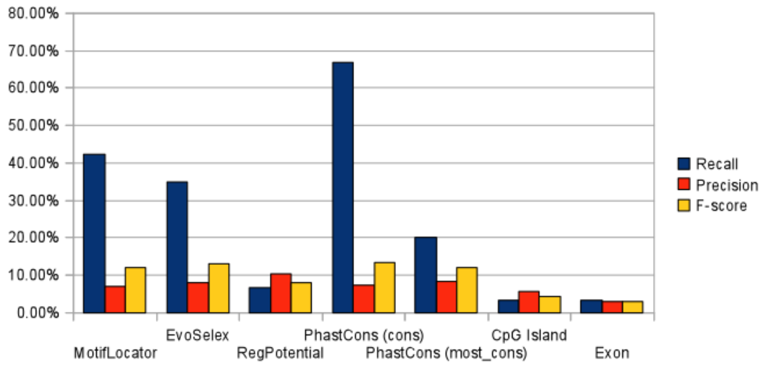


Figure 5.4: Comparison between *Recall*, *Precision*, and *F-score* from different base algorithms for the mouse data.

Figure 5.5 shows the *FP-rate* scores for each of the algorithms. Smaller values are to be preferred for this measure.

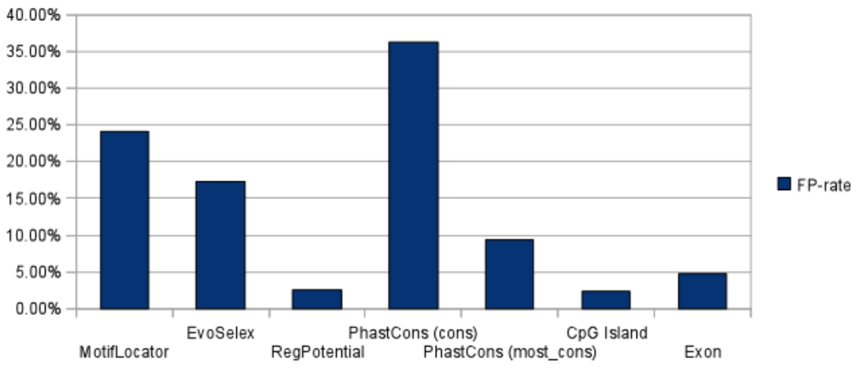


Figure 5.5: Comparison of *FP-rate* from different base algorithms for the mouse data.

Figure 5.6 shows the *Accuracy* scores for each of the algorithms. Larger values are to be preferred for this measure.

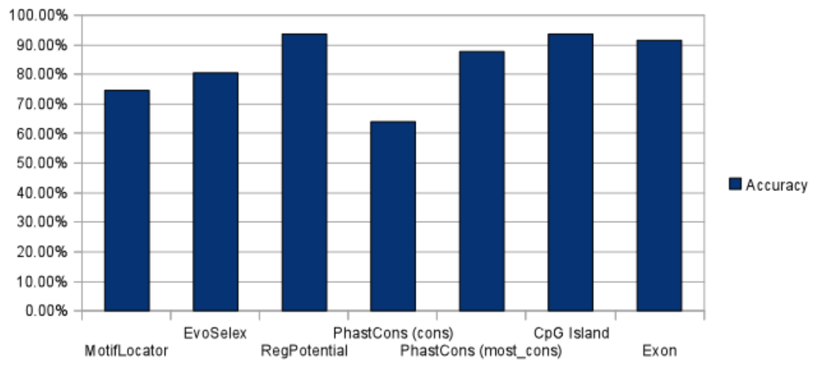


Figure 5.6: Comparison between *Accuracy* from different base algorithms for the mouse data.

5.6 Discussion

The set of results obtained for the yeast data illustrates a wide range of performances among the base algorithms: *Recall* ranges between 9-68%; *Precision* ranges between 7-25%; *F-score* ranges between 8-30%; *FP-rate* ranges between 7-48%, and *Accuracy* ranges between 50% - 88%. It is really interesting to see that the *Accuracy* rate is really high for most of the algorithms. But the *Precision* and *Recall* is not that high as expected. This supports the claim that has been made in Section 4.5.3 that for this kind of dataset that has very small percentage of minority class data, *Accuracy* is not a very good performance measure.

The set of results obtained for mouse data also illustrates a wide range of performances among the base algorithms: *Recall* ranges between 3%-42%; *Precision* ranges between 2%-10%; *F-score* ranges between 3%-13%; *FP-rate* ranges between 2%-24%, and *Accuracy* ranges between 63% - 94%.

As we know, *Recall* (True positive rate) is the proportion of positive cases that were correctly identified. But *Recall* can easily be increased by over-prediction. So, we cannot rely on *Recall* only for better prediction measures. On the other hand, increase in *Precision* can improve the prediction result considerably, but it may decrease the True Positives in the prediction. Combining both *Recall* and *Precision* should be a solution. Hence, *F-score* ought to be a useful performance measure, which can have a higher value (say more than 50%) if both *Recall* and *Precision* have higher values. A higher *F-score* with reasonable values of *Recall*, *Precision* and *FP-rate* could be good criteria for choosing the best prediction algorithms. From the results of yeast data (Table 5.5), it is evident that *Fuzznuc* is the best prediction algorithm with its reasonably better *F-score* and other performance measures than the other prediction algorithms. Among the prediction algorithms from mouse data (Table 5.6), *EvoSelex* can be chosen as the best algorithm for its higher *F-score* with reasonable *Recall*, *Precision* and *FP-rate*. Though, *PhastCons (conserved)* has the highest *F-score*, the *FP-rate* is higher than EvoSelex.

Individual algorithms exhibit some improvement in *Recall*. But the variation of the *Recall* value is really wide. The scanning algorithms show the same type of performance. Both of them have higher *Recall* values. But *Precision* values are really low in the cases of all the algorithms. *Regulatory Potential*, a co-regulatory algorithm gives the best *Precision* value though it is not much better than other base algorithms. So, it can be said that all the algorithms are unable to reliably identify real binding sites.

As a result of overall small *Precision* values, a higher value for *F-score* is only possible where there are higher *Recall* values. The three algorithms (*MotifLocator*, *EvoSelex*, and *PhastCons (conserved)*) have the highest *F-score* as all of them have higher *Recall* values. This implies that the increases in *Recall* must have involved some kind of trade-off and increases in the *FP-rate* indicate that it was not possible to improve the performance of this measure in any instance.

It is important to remember that a key limitation of all scanning algorithms is that they require good quality and a large number of position weight matrices or consensus sequences available to them. In the dataset used for this study, 27 position weight matrices were extracted from the *TRANSFAC* database (Wingender, 2008). But the promoter sequences extracted from the *SCPD* database contain annotations for 69 unique transcription factor binding sites. It can be immediately seen that the scanning algorithms are fundamentally limited in the number of sites in this dataset that they could possibly predict. Another important issue is that the quality of position weight matrices and consensus sequences is heavily dependent on the amount and quality of the data used to generate them. As a result, it might be expected that the optimal threshold settings for a scanning algorithm might vary from binding site to binding site thus making it very hard for the optimisation process to find a global optimal value.

Another reason for such unreliable results can be due to the fact that the algorithms were unable to efficiently search the parameter space. Again, there are no sets of experimentally annotated regulatory sequences for which there is absolute confidence that no binding sites have been missed, or even that all positive annotations are reliable. The dataset selected (*SCPD* for the yeast and *ABS-ORegAnno* for the mouse) represents one of the highest confidence dataset available but uncertainty still remains even here. Where inaccuracies are present, this will cause the algorithms to be unfairly penalised. Furthermore, sequences may differ with respect to the ease with which the algorithms are able to predict binding sites within them. For example, the presence of DNA features such as repeats, scaffold attachment sites; etc. may influence the case, which with the various algorithms are able to detect genuine binding sites against the background of the non-regulatory sequence.

5.7 Summary

In this chapter, I have given a brief description of the experimental organisms used in this thesis, their genomic datasets, and the various base algorithms, which will be used in the later chapters for further experiments. I have also discussed and analysed a comparative analysis of the performance measures of different prediction algorithms for both yeast and mouse datasets using a selection of performance metrics. The results showed some variation in the prediction results and it is not clear whether this is due to a lack of precision on the part of the algorithms or flaws and omissions in the experimental annotations. But these results certainly provided the motivation for the research introduced in the following chapters. In these chapters I have explored whether integrating the results from all the base prediction algorithms can provide a much better overall prediction than each of them gave individually.

Chapter 6

Integration of Algorithmic Predictions Using Non-linear Classification Techniques

6.1 Introduction

As seen in the last chapter, the maximum precision of established binding site prediction algorithms, as tested on annotated yeast and mouse sequences, remains poor. The algorithmic strategies represented in this study are diverse and incorporate differing sources of biological information to aid the predictive process. In Chapter 3, it was shown that these algorithms have their own weaknesses and strengths and combining these outputs may lead to better predictions (Che *et al.*, 2005; Huber & Bulyk, 2006; Romer *et al.*, 2007; Wilczynski *et al.*, 2008) . If one algorithm misses any binding sites another algorithm may catch that site. There is, therefore, good reason to believe that the set of binding sites predicted correctly by the individual algorithms are likely to form non-identical sets. If these predictions do indeed complement each other they could be expected to provide significantly more information when taken in combination. Hence this chapter explores the possibility that the combination of all the base algorithms will give better predictions than the algorithms themselves.

Section 3.4 describes a number of different approaches (*BEST*, *Multifinder*, *WebMOTIFS*, *MEMOFinder*, etc.) where the results from different algorithms have been combined together for improved prediction and in the previous chapter, the datasets to be used in this thesis and the basic prediction algorithms that will be combined in a meta-predictor were discussed. In this chapter, I am going to describe the classification approach that has been undertaken. This approach is somewhat similar to those approaches described above. Meta-data (motif scores) are taken from different algorithms and are combined together anticipating that using them together would combine the strengths of their different algorithms. Here I have applied classification instead of clustering. This whole chapter will describe the classification technique I have used, the first set of experiments performed, as well as a comparative analysis with the results from the base prediction algorithms.

6.2 Classification Approach

As mentioned before, each of these basic algorithms has their own particular limitations and strengths. Taken in combination, it might be expected that they provide more information about TFBSs than they do individually. These combining algorithms are called ensemble algorithms in the machine learning field and have proven to be extremely successful (Dietterich, 2000). The initial approaches (Sun *et al.*, 2005, 2006a,b, 2007, 2008, 2009a,a,b; Robinson *et al.*, 2006, 2007a,b, 2008) were to provide these algorithmic predictions as input to a Support Vector Machine (SVM), which has been trained to use the original predictions to make higher specificity predictions on the yeast data (described in Section 5.3). Among these approaches most of them adopted sampling techniques (described in Section 4.5.2). Some of the approaches (Robinson *et al.*, 2006, 2007a, 2008; Sun *et al.*, 2007, 2008, 2009a,b) used post-processing (described in Section 6.4.4). Using negative examples from different sources was undertaken in Sun *et al.* (2008). Among these approaches only few of them (Sun *et al.*, 2007, 2008; Robinson *et al.*, 2008) used the mouse data and rest of them used the yeast data described in Chapter 5. Contextualisation of data (windowing technique) was only used for the yeast data (Sun *et al.*, 2005, 2009a,b; Robinson *et al.*, 2007b).

My work will adopt the same approach (integration, sampling, training an

SVM model, post-processing, etc.) but will considerably extend it with newer and up-to-date datasets, techniques and improved cross-validation method. Firstly in this chapter, I will run experiments with the mouse data inherited from the previous research. I will also use the windowing technique on the mouse data, which has never been applied. The most interesting extension of my work over the previous research is the new modified cross-validation and it will be applied on the both datasets.

As noted above the output from the different sources of evidences discussed in the previous chapter has been combined and used as input to a SVM. In fact, two types of SVM have been used– a two-class SVM and a one-class SVM (both described in Chapter 4).

The approach can be divided into three major steps:

- Pre-processing data
- Training and testing an SVM model
- Post-processing data

Figure 6.1 shows the complete workflow of the method and these will be discussed in detail in Section 6.4.

6.3 Representation of Data

There are a number of possible ways to present the base algorithm predictions to the classification algorithms (motif scores). Therefore, experiments were repeated using different representation of data in order to clarify these issues.

6.3.1 Data structure

The predictions obtained, mentioned in the previous chapter, were used as the input datasets for the classifiers. The data is presented to the classification algorithms as an $\boldsymbol{n} \times (\boldsymbol{m}+\mathbf{1})$ matrix with each row vector representing the respective predictions for a given position within the sequence of the promoter region (see

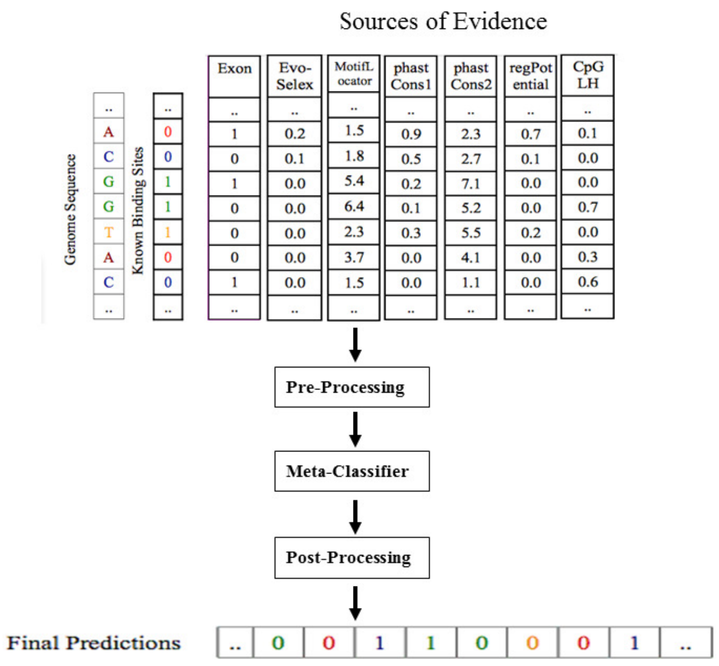


Figure 6.1: Workflow of integration sources evidence.

Figure 6.1). Here, $\boldsymbol{n}$ is number of base pairs and $\boldsymbol{m}$ is the number of features with an extra column representing the label for the data. This column, the first column of the matrix, contains annotations from an appropriate database (*SCPD* for yeast, and *ABS* and *ORegAnno* for mouse) giving our best estimate for the known binding site positions. The rest of the columns give the $\boldsymbol{m}$ predictions from the $\boldsymbol{m}$ base algorithms at each binding site position. The matrix was built by simply concatenating all of the sequences used. The label column is used when training the SVM, but not used when testing the SVM except of course to evaluate its ability at the end.

6.3.2 Windowing input vectors to include contextual information

It is not clear whether contextual information about neighbouring positions will be significant when working with a meta-analysis of the base predictions. It is possible that this kind of contextual information has already been summarised in the process of making the raw algorithmic predictions. Then again it may be the case that further contextual information can further guide accurate classification. To resolve this question, the experiments were first performed with classification performed on the set of predictions made at each base position as described in Section 6.3.1. The experiments were then repeated but this time each base position was represented by a collection of predictions. These predictions represent the sets of predictions within a window centred on the base position of interest (see Figure 6.2).

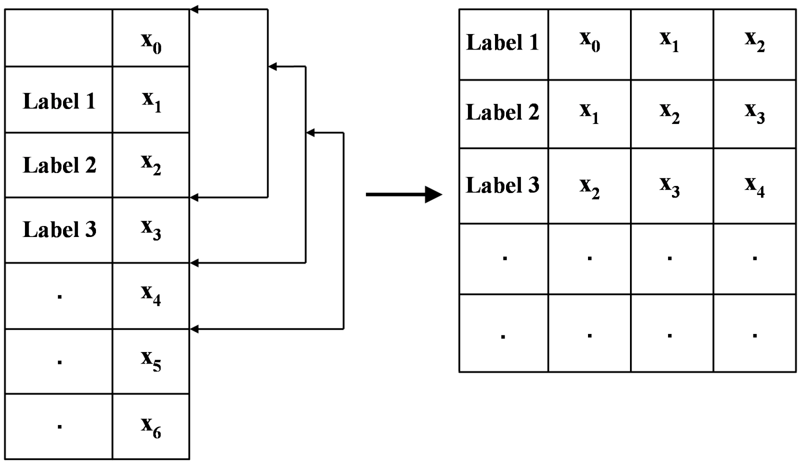


Figure 6.2: Contextualising input vectors with window size 3.

As mentioned earlier, this technique was applied on the yeast data described in Sun *et al.* (2005, 2009a,b). In this thesis, I have extended the same approach to the mouse data. I, therefore, have contextualised the training and test data by windowing the vectors. For example, in Figure 6.2, one location either side has been included, giving a window size of 3. With this window size, if there are $\boldsymbol{m}$ algorithms then each input vector is now $\boldsymbol{3 \times m}$ plus the label for the middle base pair position. The $\boldsymbol{3 \times m}$ matrix consisting of the predictions from the base pair previous to the one being trained, the predictions for this particular base pair itself and the predictions from the base pair after the trained one taken in this order. In this work, a window size from 3 to 7 in increments of 2 has been set. Therefore, for window size 5 we include two locations either side and for window size 7 we include three locations either side and so on.

6.4 Methodology for Two-class SVM

In this section, I have described the two types of experiments that were run on the yeast and the mouse data. The same experiments had been performed on yeast data in previous studies and taken from Sun *et al.* (2005, 2006a,b, 2009a,a,b); Robinson *et al.* (2006, 2007a,b). The experiments are:

(a) Using part of the original promoter region deemed to be not in a TFBSs as negative examples as described above.

(b) Using windowed data containing data mentioned in a with windowing (see Section 6.3.2) so that the contexualised data also contain its surrounding information.

Therefore, there are 2 different training sets based on each case mentioned above:
Case 1: Original yeast and mouse data, which will be denoted as *yeast* for yeast data and *mouse* for mouse data.
Case 2: *Case 1* with the addition of windowing. Window sizes 3, 5 and 7 have been used in the experiments. For the yeast data, I will denote the data as *yeast+w3* (for window size =3), *yeast+w5* (for window size = 5) and *yeast+w7* (for window size = 7). For mouse data, it will be *mouse+w3*, *mouse+w5*, *mouse+w7*, etc.

In addition some pre-processings (data division, normalisation and sampling) have been undertaken on the training set and post-processing on the prediction set. However, before discussing these steps at first I will present the statistics of both yeast and mouse data.

6.4.1 Statistics of the datasets

Table 6.1 shows the statistics of both yeast and mouse datasets. Here, 7.8% of the whole yeast data (also known as positive examples) contains base pairs that are parts of binding sites. Whereas, for mouse it is only 2.9%. In both datasets (yeast and mouse) there are a number of vectors that are repeated. Vectors of this type (repeats that occur in both negative and positive example classes) are called *inconsistent* vectors, which make up about 69% of the yeast data and 20% of the mouse data (see Table 6.1). Only the same 870 vectors, in the yeast dataset, repeated in two different classes (negative and positive classes) comprise these inconsistent data vectors and some examples are very abundant. For example, only 8 inconsistent data vectors make almost 25% of the yeast dataset. For the mouse data, only 77 vectors act as inconsistent and among them merely 3 inconsistent vectors make almost 17% of the mouse dataset.

Organism		Original	Inconsistent	Repeats	Unique
Yeast	Negative examples	62,502	40,656	17,064	5,671
	Positive examples	5,280	6,039	1,794	850
	Total	67,782	46,695	18,858	6,521
Mouse	Negative examples	59,070	11,963	20,731	31,262
	Positive examples	1,781	156	238	1,484
	Total	60,851	12,119	20,969	32,764

Table 6.1: Statistics of inconsistencies and repetitions in yeast and mouse datasets. All the numbers in the table are in base pairs.

From Figure 6.3, we can see that, in the yeast dataset, one data vector is repeated 5460 times in the negative example class labeled as the part of non-binding sites whereas the same vector is present 414 times in the positive example class where it is labeled as the part of binding sites. This single vector, which is inconsistent, constitutes more than 8.5% of the whole yeast dataset. There are also other inconsistent data vectors like this present in the dataset.

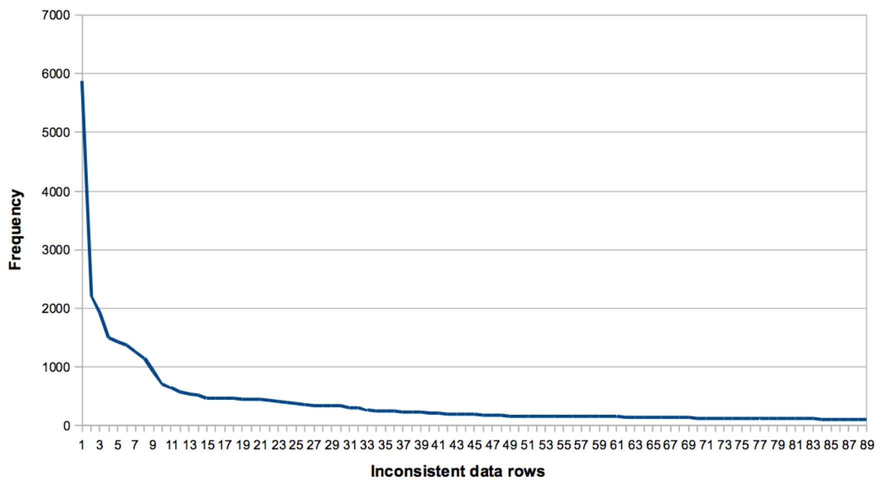


Figure 6.3: Frequency of inconsistent data rows in the yeast data. First 100 inconsistent vectors are included in the figure.

The same is true for the mouse data. Here, one data vector is repeated 6,337 times in the negative example class whereas the same vector is present 19 times in the positive example class (see Figure 6.4).

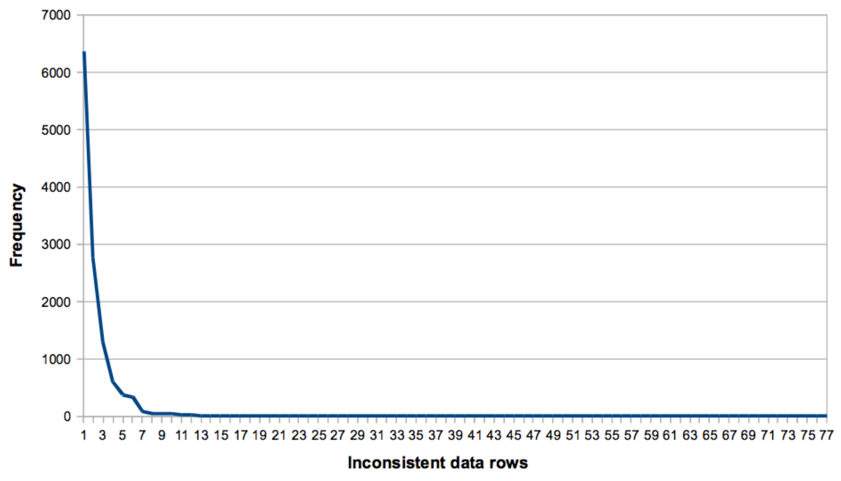


Figure 6.4: Frequency of inconsistent data rows in the mouse data.

There are also repeats that occur in only one class and these are simply called *repeats*, which are 27.8% for the yeast data and 34.5% for the mouse data. The vectors that occur only once overall are called *unique*. Surprisingly the yeast data has only 9.6% unique data vectors. However, the mouse data has 54% unique vectors.

6.4.2 Pre-processing data

The training data used in the SVM consists of a vector of predictions taken from the different original (base) algorithms, together with a label that represents the best available evidence for that particular base pair being, or not being, part of a TFBSs. Most of the biological training data used are imbalanced in nature (for details see Section 5.3). This type of data can be misleading for an SVM. An SVM may not be able to find an optimum way of identifying the patterns. Therefore, some pre-processing is needed before using the data as input into an SVM. This pre-processing is only carried out on the training dataset and it may enhance the chances of better prediction and decreases the chance of over-fitting and under-fitting that an SVM normally faces (described in Section 4.4). The pre-processing process consists of dataset division, normalisation, sampling, etc.

6.4.2.1 Dataset division and normalisation

At first, repetitive or inconsistent data was identified in the input matrix and was eliminated from the training set, as these inconsistent and repetitive data are the source of misleading prediction results. When using contextualised data, the windowing technique was used to produce larger vectors and then searched for repetitive and inconsistent data point needed for deletion. There should be fewer repetitions once the vectors are larger.

As mentioned previously, the data was divided into two sets training and test set. Two-third of the dataset has been used as a training set. Both training and test sets have been normalised to isolate any statistical error and standardise the data. This has been done by subtracting the population mean of each algorithm from an individual score of the algorithm and then dividing the difference by the population standard deviation of the that particular algorithm, in other words each feature is turned into *Z-score*.

6.4.2.2 Variation in test sets

In each experiment two types of test sets had been used. One is with only the consistent data points as described above, which will demonstrate the correct efficiency of the classifier and it will be denoted as *filtered test set* as it completely lacks in biological properties since it is no longer contiguous data. This test set will be of interest to machine learning practitioners, as it will demonstrate the classification efficiency of our SVM models on the data suitable for machine learning.

The second test set is produced by keeping the repetitive and inconsistent data points to give a biologically meaningful contextualised genome sequence and it will be denoted is as *biological test set*. Whilst I realise that this dataset contain a lot of repetitions and inconsistent data vectors, it is realistic that the measures on this set will show how our process will work on real world data. Therefore, this test set will demonstrate how good our prediction model is trained to predict binding sites from biological data and ultimately it is the biologists that are most interested in the practical application of the method described in this thesis.

6.4.2.3 Sampling techniques in the training set

As mentioned in Chapter 5, the dataset used for this study is highly imbalanced. Unless this situation is properly accounted for, the supervised classification algorithms may be expected to trivially over predict the majority class. In order to mitigate this problem a databased sampling method (Chawla *et al.*, 2002; Radivojac *et al.*, 2004) was utilised for this study. A combination of over-sampling (*SMOTE*) of the minority class and under-sampling of the majority class were used to balance the training dataset, allowing for more efficient and useful training to take place (described in Section 4.5.2). Here one thing should be noted that the number of representatives of the majority class included in the training set was calculated to ensure a constant ratio – majority to minority. A range of different ratios has been chosen during the course of this thesis.

6.4.3 Training and testing data

After constructing the training set using pre-processing, the training set was trained using an SVM on the data. For the two-class SVM, the radial basis kernel was used and the two parameter cost ($\boldsymbol{C}$) and gamma ($\boldsymbol{\gamma}$) for the kernel were chosen from a very large sequence of hyper-parameter sets. But for the one-class SVM, all four kernels supported by LIBSVM were tested and a wide range of values of nu ($\boldsymbol{\nu}$) and gamma ($\boldsymbol{\gamma}$) were used. For further information about these parameters see Chapter 4. I modified the standard approach to use an exhaustive search of some discrete values of $\boldsymbol{C}$ and $\boldsymbol{\gamma}$ ($\boldsymbol{\nu}$ and $\boldsymbol{\gamma}$ in case of the one-class SVM), which covers a wide range of values of the variables in order to find the optimum values of the variables. The ranges for cost and gamma were chosen to give reasonable boundaries, keeping in mind the value of $\boldsymbol{C}$ will have a high penalty for non-separable points and a very low value of $\boldsymbol{\gamma}$ is also not desirable as the scalar multiplication between different data points will actually be the same data points. However, we explored the whole range of $\boldsymbol{\nu}$ (from 0 to 1 with an interval of 0.05) for the one-class SVM. A detail description of these parameters is given in Section 4.2.2. All the values of the variables during this exhaustive search were selected by standard 5-fold cross validation, which has been described in details in Section 6.4.5.

6.4.4 Post-processing Data

The original biological algorithms predict contiguous sets of base pairs as binding sites. However in the classification approach undertaken here, each base pair is predicted independently of their neighbouring nucleotides. As a result, the classifier may output many short predictions sometimes even with the length of only one or two. It is not clear whether these very short stretches are feasible or have any biological meaning. From both yeast and mouse dataset, it can be seen that the shortest binding site is 5 bps in length and the longest one is 13 bps. Therefore, predictions with a length equal or smaller than a threshold value had been removed (replaced the positive prediction with a negative one) and then the effect of the performance was measured.

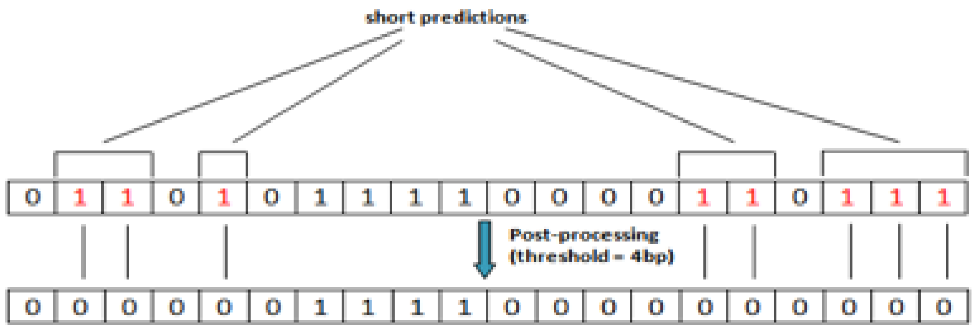


Figure 6.5: Filtering on prediction by using post-processing

In this study, different threshold values (from 4 bps to 7 bps) had been used rather than only one to explore possible feasible threshold sizes. This post-processing can obviously only be carried out on the biological test set, since that is the only test set which is still contiguous and still a subset of the original genome whereas filtered test set contains random data vectors. Figure 6.5 shows how the post-processing performed on the prediction with threshold value 4 bps.

6.4.5 Cross-validation

In all the previous experiments mentioned in 6.2 and some of experiments presented in this thesis, standard cross-validation technique has been used. In standard cross-validation, the data has been divided in five subsets for 5-fold cross validation. Then four subsets are taken as training during cross-validation and the rest of the data part is used as the validation set. However, the cross-validation method used so far may not be efficient for the kind of problem that I am dealing within this thesis. As mentioned, the data I have used is imbalanced in nature, the data is needed to be processed before training to make it balanced (discussed in Section 4.5). But the unseen test set is still imbalanced. During cross-validation, the validation set is also balanced which is not the same as the test set used finally for prediction. So the nature of the test set is quite different from the validation set used during cross-validation. The differences are two-fold:

- In the validation data the minority class is oversampled and the majority class is under-sampled. This is not the case in the test set.
- In the test data, short sequences of binding site predictions are removed. This is not the case in the validation set.

Therefore, a *modified cross-validation* method has been devised (mentioned in Section 4.4.1) in which the model is validated in exactly the same way, as it will be tested. The model is validated with non-sampled validation sets and short predictions were removed. In this new modified version, the data is divided in five subsets for 5-fold cross validation and four subsets are taken as training during cross-validation, but now the validation set is not drawn from the rest of the pre-processed data. Rather it is taken from the corresponding original dataset. The training set in the cross-validation undergoes the same pre-processing to make it balanced. The prediction from the validation also undergoes post-processing (described in Section 6.4.4 in details). The pseudo-code of this modified cross-validation method is given below:

Pseudocode 2 Finding the best hyper-parameters with modified cross-validation method.

```
1:  Split the training data into 5 partitions
2:  This gives 5 different training sets (4 of 5) and the corresponding validation
    sets (1 of 5)
3:  for each of the 5 training set do
4:      Pre-process the data to produce balanced training set
5:  end for
6:  for each combination of hyper-parameter values do
7:      for each of the pre-processed 5 training sets do
8:          Train an SVM
9:          Measure performance on the corresponding validation set, exactly as the
            final test will be measured. So use a Performance Measure, after the
            predictions on the validation set have been filtered (post-processing)
10:     end for
11:     Average the Performance Measure
12: end for
13: Choose the combination of hyper-parameter with the best average
```

One thing should be noted, in all the previous experiments, *Accuracy* was used as the cross-validation criterion or performance measure as shown in the pseudocode above. As *F-score* is more feasible a performance measures than *Accuracy* (see Section 4.5.3) for the problem I am dealing with, I have used it as the cross-validation criterion in all the experiments undertaken in this thesis. However, this code is flexible enough to allow any criteria of performance. Therefore, *Accuracy* has also been used as a cross-validation criterion for comparison in this chapter.

Having set up the training and test data as described above some results for the two-class SVM have been produced. Section 6.5 will have the results from the yeast dataset, which are just repeats of those done in Sun *et al.* (2005, 2006a,b, 2009a,a,b); Robinson *et al.* (2006, 2007a,b). Section 6.6 will give the results for the mouse data. These results are all new to me so a much more detailed set of experiments were carried out, which involved both the filtered and the biological test data. Finally in Section 6.7, I will look at the results produced using the enhancement to the cross-validation process described in this section.

6.5 Results for Two-class SVM on the Yeast Data using Standard Cross-validation (biological test set)

The standard cross-validation procedure is to analyse the performance in terms of the *Accuracy* value. This is the default setting, though later with the mouse data this cross-validation criterion was changed to *F-score*. As the results presented in this section have been repeated from the previous studies mentioned in Section 6.2, the new modified cross-validation procedure has not been used. The confusion matrix of the best algorithm (*Fuzznuc*), among the 12 prediction algorithms (see Table 5.5), is as follows:

	Predictive Negatives	Predictive Positives
Actual Negatives	TN = 83%	FP = 10%
Actual Positives	FN = 4%	TP = 3%

Table 6.2: Confusion matrix of the best base algorithm *Fuzznuc* on the yeast data.

The Performance measure of *Fuzznuc* is:

	Recall	Precision	F-score	FP-rate
Fuzznuc	0.4	0.222	0.245	0.245

Table 6.3: Performance measures of the best base algorithm *Fuzznuc* on the yeast data.

From Table 6.2, it can be seen that the False Positives are three times more than the True Positives, which makes the best algorithm unreliable.

The results of combining prediction results using an SVM are given in Table 6.4. The results are generated from the prediction made on the biological test set.

	Recall	Precision	F-score	FP-rate
yeast	0.305	0.317	0.334	0.044
yeast+w3	0.132	0.628	0.218	0.008
yeast+w5	0.207	0.511	0.295	0.017
yeast+w7	0.221	0.499	0.307	0.019

Table 6.4: Results of two-class SVM (cross-validation criterion: *Accuracy*) on the yeast data.

Figure 6.6 shows the comparisons between the *F-scores* of the best prediction algorithm and the two-class SVM approaches (cross-validation criterion: *Accuracy*). In most of the cases, the *F-scores* are higher than that of the original prediction algorithm, but not by much.

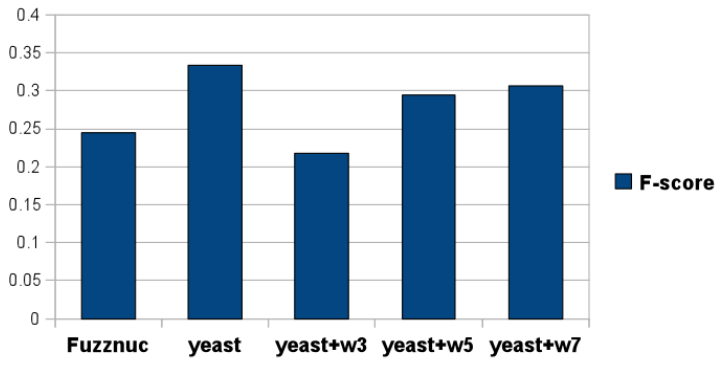


Figure 6.6: Comparison of *F-scores* between *Fuzznuc* and the two-class SVM approach (cross-validation criteria: *Accuracy*) on the yeast data.

Figure 6.7 shows the comparisons between the *FP-rates* of the best prediction algorithm and the two-class SVM approaches (cross-validation criterion: *Accuracy*). In all the cases, the *FP-rate* is lower than that of the original prediction algorithm. These results also show that while windowing improves the *FP-rates*, it does not improve the *F-scores*.

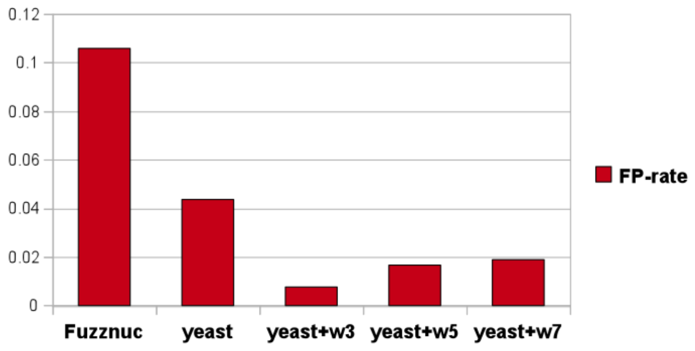


Figure 6.7: Comparison of *FP-rate* between *Fuzznuc* and two-class SVM approach (cross-validation criteria: *Accuracy*) on the yeast data.

6.6 Results for Two-class SVM on the Mouse Data using the Standard Cross-validation

For the mouse data, I have chosen two different criteria for cross-validation: i) *Accuracy* and ii) *F-score* and compared the performances. I have also run the trained model on both the filtered test set (containing no repetitions and inconsistencies) and the biologically meaningful test set. Again the new modified cross-validation process has not been used here. One thing should be noted; I have explored a set of ratios (negative examples: positive examples) for under sampling the negative examples and in the results given below the ratio that has given the best classification performance has been used.

6.6.1 Results for the best original algorithm

Before presenting the experimental results, let us see how the original base algorithms perform for identifying *cis*-binding sites. The results are given for the best algorithm called *EvoSelex* (see Table 5.6). From Section 5.5.2, the confusion matrix is as follows:

	Predictive Negatives	Predictive Positives
Actual Negatives	TN = 79.25%	FP = 16.60%
Actual Positives	FN = 2.70%	TP = 1.44%

Table 6.5: Confusion matrix of the best base algorithm *EvoSelex* on the mouse data.

Therefore, the Performance measures of *EvoSelex* is:

	Recall	Precision	F-score	FP-rate
EvoSelex	0.348	0.08	0.13	0.173

Table 6.6: Performance measures of the best base algorithm *EvoSelex* on the mouse data.

From the results in Tables 6.5 and 6.6, it is evident that the *FP-rate* is very high, but that *Precision* is not that high. As a result the *F-score* is also not high.

This happens because the algorithm tries to annotate many non-binding sites as binding sites. This leads to a high *Recall* but leaves the *Precision* low.

6.6.2 Results for two-class SVM on the filtered test set

Initially the standard cross-validation procedure using *Accuracy* is used and thus the results for the filtered test set are as follows:

	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
mouse	177	1452	524	9881	0.253	0.109	0.152	0.128
mouse+w3	156	889	603	14169	0.206	0.149	0.172	0.059
mouse+w5	171	1717	591	14269	0.224	0.091	0.129	0.107
mouse+w7	147	1406	618	15117	0.192	0.095	0.127	0.085

Table 6.7: Results of filtered test set from the mouse dataset (cross-validation criterion: *Accuracy*).

Next the cross-validation criterion was changed to *F-score* and the results are as follows:

	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
mouse	149	1412	552	9921	0.213	0.095	0.132	0.125
mouse+w3	156	891	603	14167	0.206	0.149	0.173	0.059
mouse+w5	431	3892	331	12094	0.556	0.099	0.169	0.244
mouse+w7	419	3672	346	12851	0.548	0.102	0.173	0.222

Table 6.8: Results of filtered test set from the mouse dataset (cross-validation criterion: *F-score*).

The results are mixed and show some improvements on the best original algorithm. However, the results are not as good as expected. Changing to using *F-score* as the cross-validation criterion in the cross-validation method has not done much to improve the results.

6.6.3 Results for two-class SVM on the biological test set

6.6.3.1 Using *Accuracy* as the cross-validation criterion

Table 6.9 shows the results of two-class SVM approaches where the cross-validation criterion is *Accuracy*.

	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
mouse	167	1313	617	16811	0.213	0.113	0.148	0.073
mouse+w3	140	648	644	17436	0.179	0.178	0.178	0.036
mouse+w5	117	937	667	17107	0.149	0.111	0.127	0.052
mouse+w7	90	691	694	17313	0.115	0.115	0.115	0.038

Table 6.9: Results of biological test set from the mouse dataset (cross-validation criterion: *Accuracy*).

If these results are compared with the previous result in Table 6.6, we can see that the SVM has improved the prediction results. The *FP-rate* has decreased and the *F-score* value has increased. Using only mouse data gives a slightly better *F-score* and the *FP-rate* has also decreased. But *mouse+w3* gives decreased *FP-rate* with the best *F-score*, which makes it the best prediction in Table 6.9.

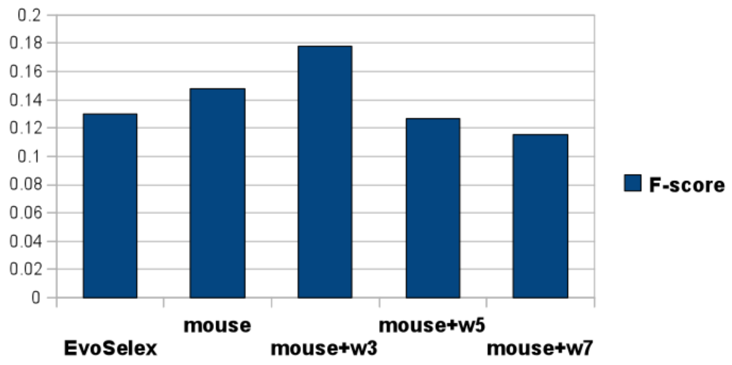


Figure 6.8: Comparison of *F-scores* between *EvoSelex* and the two-class SVM approach (cross-validation criteria: *Accuracy*) on the mouse data.

Figure 6.8 shows the comparisons between the *F-scores* of the best prediction algorithm and the two-class SVM approaches (cross-validation criterion: *Accuracy*). In only two cases is *F-score* better than that of the original prediction algorithm, with the 3 window mouse data showing the best result.

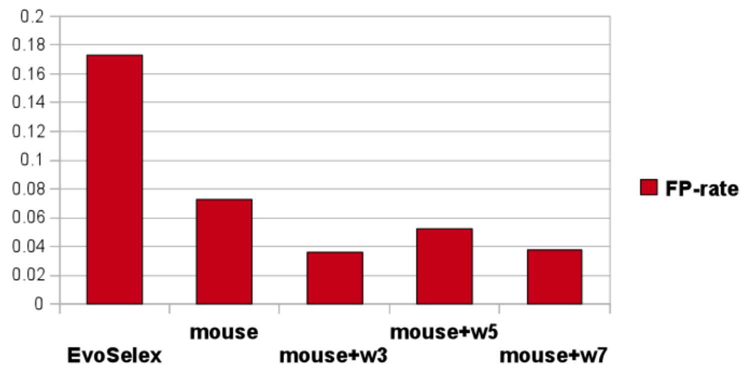


Figure 6.9: Comparison of *FP-rates* between *EvoSelex* and the two-class SVM approach (cross-validation criteria: *Accuracy*) on the mouse data.

Figure 6.9 shows the comparisons between the *FP-rates* of the best prediction algorithm and the two-class SVM approaches (cross-validation criterion: *Accuracy*). In all the cases, the *FP-rate* is lower than that of the original prediction algorithm.

6.6.3.2 Using *F-score* as the Cross-validation Criterion

The same experiments were run taking *F-score* as the cross validation criterion. Table 6.10 shows the results of the two-class SVM approaches where the cross-validation criterion is *F-score*.

	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
mouse	167	1313	617	16811	0.213	0.113	0.148	0.073
mouse+w3	140	648	644	17436	0.179	0.178	0.178	0.036
mouse+w5	341	2430	443	15614	0.435	0.123	0.192	0.135
mouse+w7	373	2518	411	15486	0.476	0.129	0.203	0.139

Table 6.10: Results of biological test set from the mouse dataset (cross-validation criterion: *F-score*).

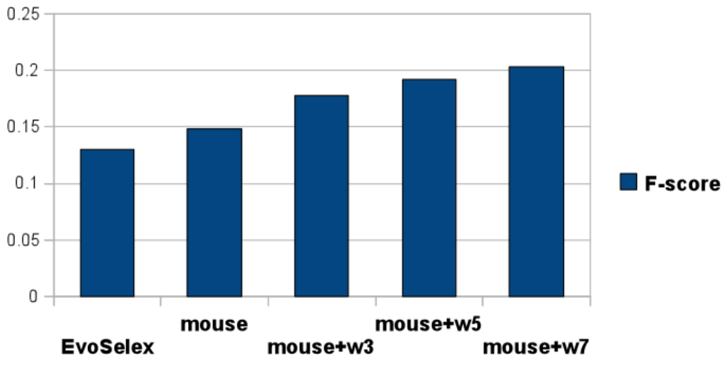


Figure 6.10: Comparison of *F-scores* between *EvoSelex* and the two-class SVM approach (cross-validation criteria: *F-score*) on the mouse data.

Figure 6.10 shows the comparisons between the *F-scores* of the best prediction algorithm and the two-class SVM approaches (cross-validation criterion: *F-score*). In all the cases, the *F-score* is higher than that of the original prediction algorithm.

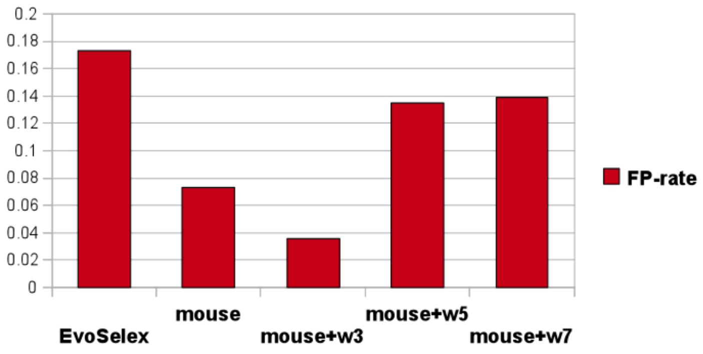


Figure 6.11: Comparison of *FP-rates* between *EvoSelex* and the two-class SVM approach (cross-validation criteria: *F-score*) on the mouse data.

Figure 6.11 shows the comparisons between the *FP-rates* of the best prediction algorithm and the two-class SVM approaches (cross-validation criterion: *F-score*). In all the cases, the *FP-rate* is lower than that of the original prediction algorithm.

It can be seen that the best results in Table 6.10 are very similar to the best results of Table 6.9. It can be concluded that using the *F-score* instead of the *Accuracy* has not brought that much benefit. However, this approach of integrating algorithmic predictions gave better result than the individual base algorithm.

6.7 Using the Modified Cross-validation Method

I have used standard cross-validation method in all the two-class SVM experiments described so far. As we mentioned earlier, datasets with an imbalanced nature has been used. To balance the data for training an oversampling technique has been used. So the oversampled data for training contains a lot of synthetic data produced during oversampling, which may hamper the model in predicting from the test set that contains just biological data.

For this reason, I will now use the modified cross-validation process as described in Section 6.4.5. Here, both *F-score* and *Accuracy* were being used as the cross-validation criteria (with the biological validation set) and the results are compared. These results are for the biologically meaningful test set only.

Previously, I have explored a set of ratios (negative examples: positive examples) for under sampling the negative examples and have used the ratio that gave the best classification performance. However, from previous experiments mentioned above, it was observed that only the two ratios 1:1 and 2:1 for under sampling the negative examples give better results than any other ratios I had used. Therefore I will only use these two ratios for under sampling the negative examples from now on. Again I have used the ratio that gave the best classification performance in the results presented below. This will be the same for both the yeast and the mouse data.

6.7.1 Results of Two-class SVM on Yeast Data using Modified Cross-validation (biological test set)

Table 6.11 shows the results on yeast data using the modified cross-validation using *F-score* and *Accuracy*.

Cross-validation criterion	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Accuracy+ post-processing	524	876	2284	18910	0.187	0.374	0.249	0.044
F-score+ post-processing	643	1409	2165	18377	0.229	0.313	0.265	0.071

Table 6.11: Results of two-class SVM on the yeast data using modified cross-validation method.

Figure 6.12 shows the comparison between *F-scores* and *FP-rates* while using the new modified cross-validation method. Here, both *F-score* and *Accuracy* have been used as cross-validation criteria. It is quite clear that using *F-score* as a cross-validation criterion gives a slightly better *F-score* than where *Accuracy* is used as a cross-validation criterion. This may indicate that the SVM could identify the True Positives better in this case. On the other hand the *FP-rate* increases while using *F-score* as a cross-validation criterion. So the results are mixed regarding this new cross-validation process.

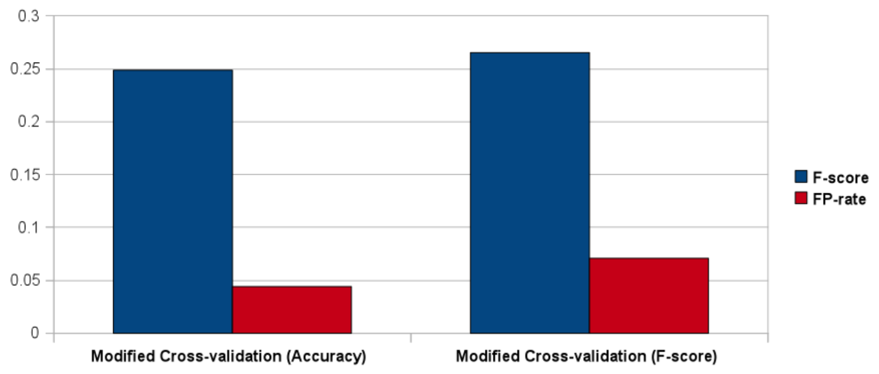


Figure 6.12: Comparison of *F-scores* and *FP-rates* using modified cross-validation method using both *Accuracy* and *F-score* as cross-validation criteria for the yeast data.

In conclusion the original method in Table 6.4 has better results than the new results in Table 6.11, but not by much, and besides all the results are fairly poor.

6.7.2 Results of two-class SVM on the mouse data using modified cross-validation (biological test set)

Table 6.12 shows the results on the mouse data using the modified cross-validation using *F-score* and *Accuracy*.

Cross-validation criterion	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Accuracy+ post-processing	157	976	627	17148	0.200	0.139	0.164	0.054
F-score+ post-processing	191	908	593	17216	0.244	0.174	0.203	0.050

Table 6.12: Result of two-class SVM on mouse data using modified cross-validation method

Figure 6.13 shows the comparison between *F-scores* and *FP-rates* while using the new modified cross-validation method. For the mouse data, the same improvement occurs in *F-score* like that of the yeast data. The *FP-rate* did not increase as in case of the yeast data, which is quite promising.

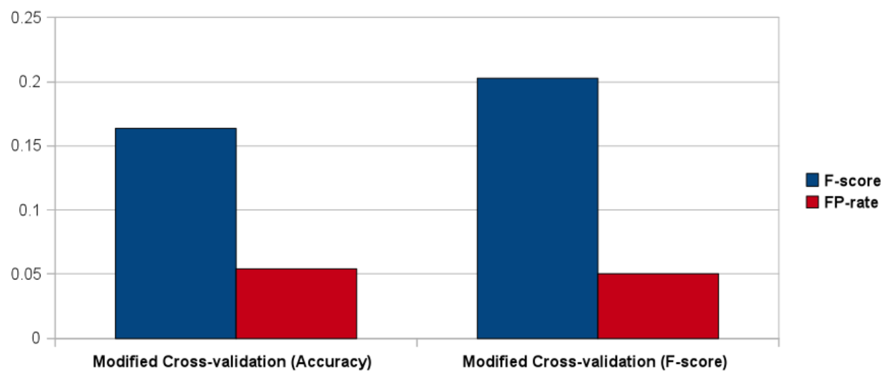


Figure 6.13: Comparison of *F-scores* and *FP-rates* using modified cross-validation method using both *Accuracy* and *F-score* as cross-validation criteria for the mouse data.

In conclusion the new results in Table 6.12 are slightly better than the original results in Tables 6.9 and 6.10. Therefore, for the mouse data the new cross-validation process is perhaps worth considering more in later experiments.

6.8 Methodology for the One-class SVM

A number of one-class classification experiments were run using a one-class SVM. The same yeast and mouse data, described in Section 5.3, have been used in these experiments. One thing should be noted is that the one-class SVM has been used with different kernels on each dataset.

In addition, some pre-processings (data division, normalisation and sampling) were undertaken on the training set and post-processing on the prediction set. These steps are almost the same as those described in Section 6.4 except in the case of making the training sets. Here, we took out all the negative examples (not annotated as binding sites) and thus the training set contained only positive examples, as required by a one-class SVM. The new modified cross-validation method was used during training the data.

6.9 Results for One-class SVM (biological test set)

As mentioned earlier, experiments using different kernels were run to see the effects on training and prediction. The run time for the one-class SVM was quite quick, so the training and prediction could be run with the same training and test dataset on a wide range of parameters to determine the best kernel with parameter values that would give better prediction.

6.9.1 One-class SVM result for the yeast data (biological test set)

The results of using the one-class SVM on the yeast data are as follows:

Kernel Type	Recall	Precision	F-score	FP-rate
Linear	0.136	0.151	0.143	0.058
RBF	0.663	0.076	0.136	0.61
Sigmoid	0.411	0.117	0.183	0.234
Polynomial(degree = 2)	0.769	0.075	0.137	0.72
Polynomial(degree = 3)	0.423	0.072	0.123	0.533

Table 6.13: One-class SVM results on the yeast data with different kernels.

The results (in Table 6.13) show that, for the yeast data the *sigmoid* kernel gives better classification results than the other kernels as measured by *F-score*. Although the other kernels have very similar *F-score* values, they have greater values of *FP-rates* than the *sigmoid* apart from the *linear* kernel, which has the lowest *FP-rate*.

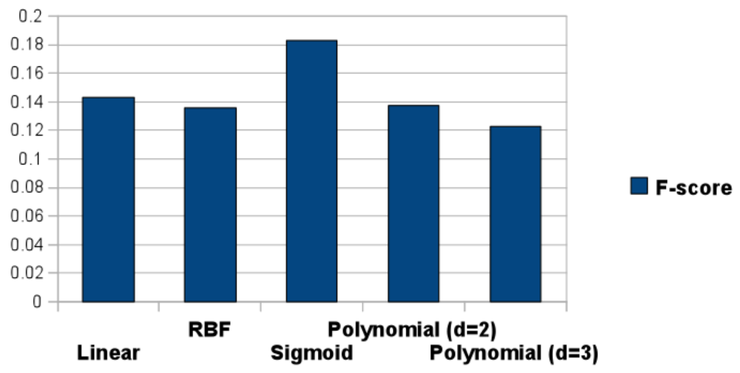


Figure 6.14: Comparison of *F-scores* from the one-class SVM on the yeast data with different kernels.

Figure 6.14 shows the comparisons between the *F-scores* from one-class SVM approaches on the yeast data with different kernels. Here the prediction using *sigmoid* kernel gives the best *F-score*.

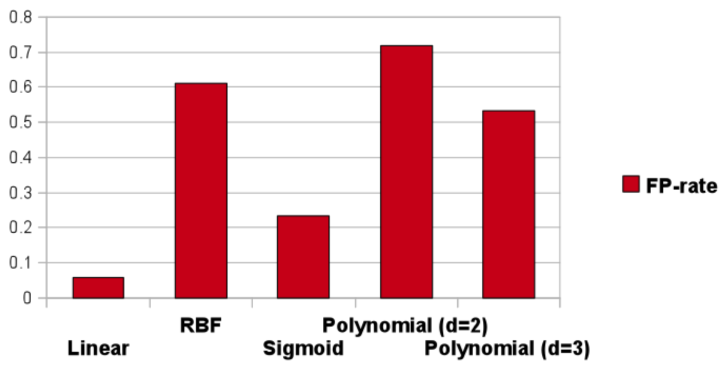


Figure 6.15: Comparison of *FP-rates* from the one-class SVM on the yeast data with different kernels.

Figure 6.15 shows the comparisons between the *FP-rates* from one-class SVM approaches on the yeast data with different kernels. Here the prediction using *linear* kernel has the lowest *FP-rate*.

Interestingly, all the better results were produced using smaller values of γ. The smaller values of gamma correspond to the smaller number of outliers and as a result they can cover most of the training points. This makes the one-class SVM really sensitive to proper parameter choice.

6.9.2 One-class SVM result for the mouse data (biological test set)

The results of using one-class SVM on the mouse data are as follows:

Kernel Type	Recall	Precision	F-score	FP-rate
Linear	0.291	0.143	0.191	0.076
RBF	0.327	0.152	0.207	0.079
Sigmoid	0.394	0.128	0.193	0.116
Polynomial(degree = 2)	0.979	0.041	0.149	0.99
Polynomial(degree = 3)	0.171	0.124	0.144	0.052

Table 6.14: One-class SVM results on the mouse data with different kernels

The one-class SVM results (Table 6.14) for the mouse data show that, unlike the yeast data, the *Gaussian* kernel (RBF) yields the best classification results. The *Gaussian* kernel produces both a better *F-score* and *FP-rate* than the other kernels. The *linear* kernel gives very similar results to the *Gaussian* kernel. The *FP-rate* is the same, but the *Gaussian* kernel produces a better *Precision* value, which may lead to detection of some novel patterns in mouse data. One thing should be noted that, the *polynomial* kernel (with degree = 2) produced the worst results among all the kernels. This can be due to the fact that the training model using *polynomial* kernel (degree =2) tried to predict everything as positive examples, therefore it produces a very high *Recall* along with low *Precision* and a very high *FP-rate*. The result for the yeast data using *polynomial* kernel (degree = 2) followed the same trend.

Figure 6.16 shows the comparisons between the *F-scores* from one-class SVM approaches on the mouse data with different kernels. Here the prediction using RBF kernel gives the best *F-score*.

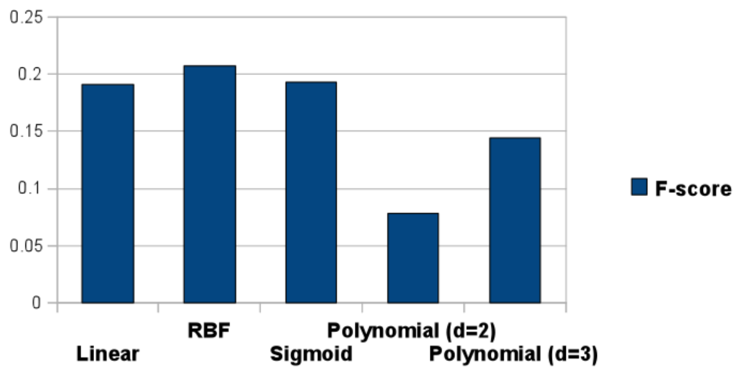


Figure 6.16: Comparison of *F-scores* from the one-class SVM on the mouse data with different kernels.

Figure 6.17 shows the comparisons between the *FP-rates* from one-class SVM approaches on the mouse data with different kernels. Here the prediction using *linear* and RBF kernels have the lowest *FP-rate*.

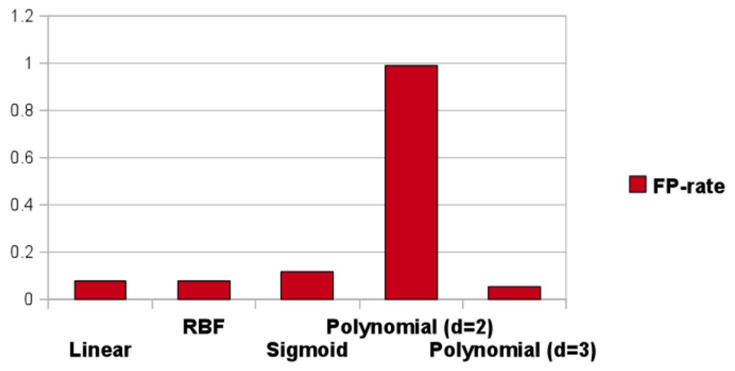


Figure 6.17: Comparison of *FP-rates* from the one-class SVM on the mouse data with different kernels.

6.10 Discussion

In order to determine the appropriate parameters to use in an SVM a standard cross- validation procedure was carried out. The cross-validation criteria previously used were based on *Accuracy* (percentage of right classifications). But for the datasets used here, where positive examples are far fewer than negative examples, *Accuracy* may not be a very good measure. The cross-validation based on *Accuracy* can give classification choice with higher *Accuracy*, but it may not yield a better classification result for our datasets. *F-score* was then tried as an alternative cross-validation criterion. This method improved results in some cases, but the overall result is not consistent. So choosing different cross-validation criteria may not produce better results as I had expected. Along with this, a modified cross-validation method was also used for getting improved predictions. Now we are going to compare the best results from the experiments that have been undertaken so far.

6.10.1 Prediction Algorithm vs. Two-class SVM vs. One-class SVM (biological test set)

The results from the two-class SVM and the one-class SVM need to be compared with that of original algorithm and those from the previous studies where the algorithms has been combined and used in a two-class SVM.

First we look at the yeast data. Table 6.15 has the results for the biological test set with the following rows:

i The first are the results from the best original algorithm (*Fuzznuc*) – Table 6.3

ii The second are the results from previous studies using a two-class SVM with the standard cross-validation using *Accuracy* – Table 6.4

iii The third are the results from the two-class SVM with the modified cross-validation (cross-validation criterion: *F-score*) – Table 6.11

iv The fourth are the results from the one-class SVM with the modified cross-validation (cross-validation criterion: *F-score*) and a *sigmoid* kernel – Table 6.13

In each case, the results presented are the best of that type.

	Recall	Precision	F-score	FP-rate
Fuzznuc	0.400	0.222	0.245	0.106
Two-class SVM (standard cross-validation)	0.305	0.371	0.334	0.044
Two-class SVM (modified cross-validation)	0.229	0.313	0.265	0.071
One-class SVM (modified cross-validation)	0.411	0.117	0.183	0.234

Table 6.15: Comparison of the best results of base algorithm, two-class SVM and one-class SVM approaches for the yeast data.

If the best result from one-class SVM is compared with other results, it becomes clear that one-class classification does not give better classification result for the yeast dataset. It does not even give better result than the original algorithm.

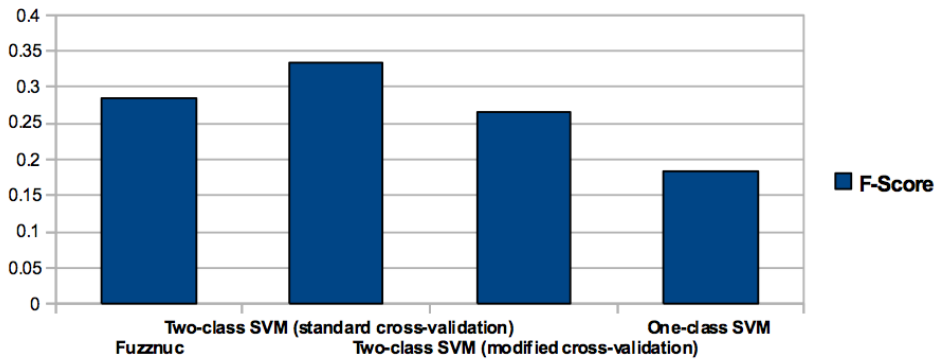


Figure 6.18: Comparison of the *F-scores* of the best base algorithm, two-class SVM and one-class SVM approaches for the yeast data.

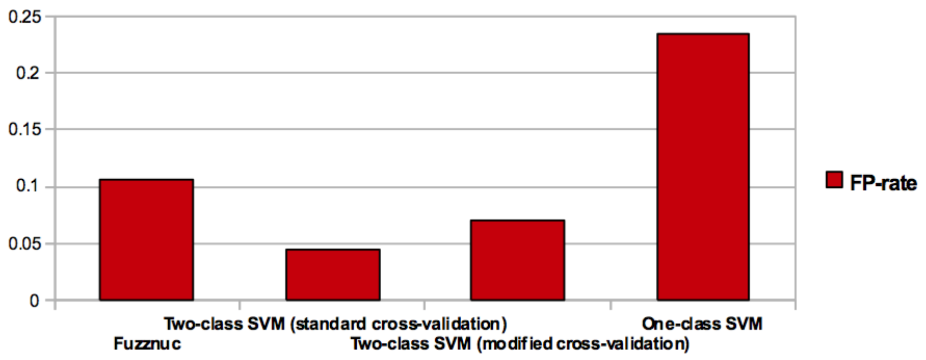


Figure 6.19: Comparison of the *FP-rates* of the best base algorithm, two-class SVM and one-class SVM approaches for the yeast data.

To my surprise, the two-class SVM using standard cross-validation gives the best result. The two-class SVM using the modified cross-validation gives results not much better than the original algorithms in that it improves the *FP-rate* but has slightly worse *F-score*. However, it does produce better *Precision* than the original algorithm.

Next let us look at the mouse data. Table 6.16 has the results for the biological test set with the following rows:

i The first are the results from the best original algorithm (*EvoSelex*) – Table 6.6

ii The second are the results from the two-class SVM with the standard cross-validation (cross-validation criterion: *F-score*) – Table 6.10

iii The third are the results from the two-class SVM with the modified cross-validation cross-validation criterion: *F-score*) – Table 6.12

iv The fourth are the results from the one-class SVM with the modified cross-validation (cross-validation criterion: *F-score*) and a *Gaussian* kernel – Table 6.14

In each case, the results presented are the best of that type.

	Recall	Precision	F-score	FP-rate
EvoSelex	0.348	0.080	0.130	0.173
Two-class SVM (standard cross-validation)	0.476	0.129	0.203	0.139
Two-class SVM (modified cross-validation)	0.244	0.174	0.203	0.050
One-class SVM (modified cross-validation)	0.327	0.152	0.207	0.079

Table 6.16: Comparison of the best results of base algorithm, two-class SVM and one-class SVM approaches for the mouse data.

From these results (see Table 6.16) it is evident that, overall the two-class SVM (using the modified cross-validation method) gives the best result and it is certainly better than the original algorithms. The result from the one-class SVM is also very similar but its *FP-rate* is lower than the two-class SVM (using the modified cross-validation). The two-class SVM (using the modified cross-validation method) can characterise the positive example more efficiently than other methods; as a result, the *Precision* is higher as well as the *FP-rate* being lower than other methods. Whereas the *Precision* from the one-class SVM is slightly lower and the *Recall* is higher.

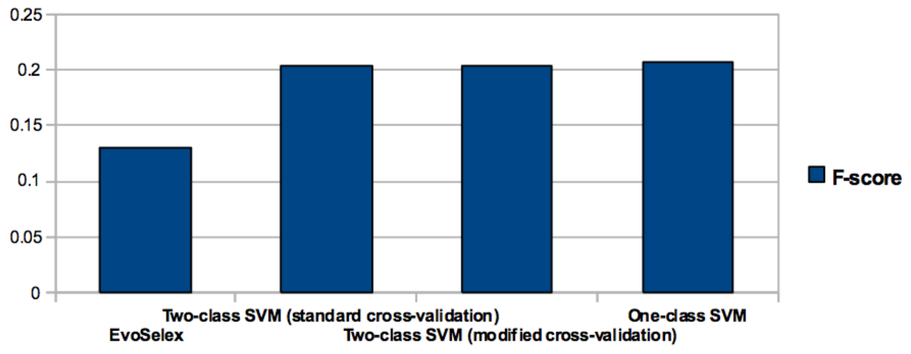


Figure 6.20: Comparison of the *F-scores* of the best base algorithm, two-class SVM and one-class SVM approaches for the mouse data.

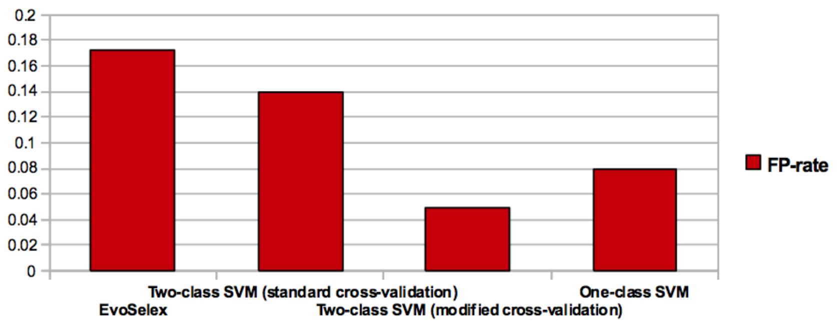


Figure 6.21: Comparison of the *FP-rates* of the best base algorithm, two-class SVM and one-class SVM approaches for the mouse data.

Though slightly better result can be obtained for the one-class SVM in the case of mouse data, better results have not been achieved in case of the yeast data (compared to the base algorithms). So, the question arises, is the one-class SVM that has been devised too specific for certain dataset? This may not be the case. There can be many reasons for getting undesired results for the yeast data. One problem with Schölkopf's method is that it is sensitive to parameter selection. Sensitivity of the parameters sometimes makes the method difficult for generalisation. This is the reason that different set of data required different kernels to produce better classification. The number of features is also crucial for this method. In the one-class SVM method any feature that is not representative for classification could be removed. A better choice of feature selection could improve the classification. Also selection of kernels depends upon the feature size. *Linear* and *sigmoid* kernel did not seem to be sensitive to this and gave steadier results than other kernels. *Polynomial* kernels gave poor results for all the data indicating that this type of kernel is not feasible for this data.

Schölkopf's method also needs a lot of positive examples to determine the boundary between the positive class and other classes. After removing the inconsistent and repetitive data, the number of positive examples decreased a lot which may be an obstacle for getting better classification result. For this reason, experiments were run without removing inconsistent and repetitive data points, which gave even worse classification results. The result can be due to the fact that some positive examples were wrongly classified as other classes. Finally I tried oversampling the positive example using SMOTE for the training set. But it again reduced the classification performance.

Five fold cross validation (using *Accuracy* as cross-validation criterion) has been used. This can be another problem, as the nature of data does not support *Accuracy* as the optimum metric for judging classification performance. However, the modified cross-validation method failed to produce promising result for yeast data. The reason could be the dataset itself. The yeast data, which I have been using, is quite old and lacks in proper annotations, for this reason a new yeast dataset has been produced and the results of the experiments on this new dataset will be presented in Chapter 8.

6.11 Summary

In this chapter, I have described the two-class and one-class classification approaches on the combination of predictions from basic algorithms. I have given a brief description of different techniques (pre-processing) involved before an SVM has been introduced for training. The results on the yeast have been disappointing. All the changes tried, namely using *F-score* instead of *Accuracy* as the cross-validation criterion, using the same method in the cross-validation as will be used for the final test set and using a one-class SVM have all failed to improve the results given by the standard cross-validation method used previously. It seemed to me that the improvements in the cross-validation procedure should have improved the results, but I have been proved wrong. Particularly disappointing was the trial of the one-class SVM so much that I have decided to ignore it from now on. Results for the mouse are not much better, but showed some possible potential for the new modified cross-validation procedures.

The inconsistencies in results for the yeast have given a reason to think about the dataset. The same yeast data has been used in this chapter as used in the previous studies and will be again used in the following chapter. But in Chapter 8, an updated yeast dataset will be introduced and I am going to re-run the whole experiments again to see the impact on the results. In the next chapter other changes in procedures will be tried out and as we shall see considerable improvements are then found.

Chapter 7

Improving Transcription Factor Binding Sites Predictions by Using Negative Examples from Different Sources

7.1 Introduction

From the results and discussion of Chapter 5, we observed that computational approaches are not of of good quality and typically prone to predicting many false positives, which significantly limit their utility. In the earlier research (described in Chapter 6), the results from different predictors have been combined together to produce a prediction by using a meta-classifier, an SVM. It has been confirmed that the results from the combination of predictions are better than that of any of the individual algorithm and improved results have been achieved from the previous studies.

However, these improvements are not as significant as expected. The results showed that this approach is still generating lots of false predictions if we compare them to original the annotated binding sites. This raises some questions : a) are the methods (mentioned in Chapter 6) being used properly? b) are reliable datasets being used? In this chapter, I will address these issues and investigate some alternative approaches. In detail, I will describe the effect of inconsistent and repetitive data vectors (described in Section 6.4.1) and the sources of negative examples (sequences of DNA that are not believed to be transcription factor regulatory regions) in the process. I have already introduced the new modified cross-validation procedure in Chapter 6, but will now use it with different sets of negative examples. I will extend this work by replacing the promoter negative examples with ones taken further from the promoter region and ones produced by a randomisation process. In this chapter, I will give results that will show major improvement shown on original algorithms.

7.2 Anticipated Problems and Solutions

As mentioned in the previous chapter, the improvements achieved were marginal and this leads to the belief that there might be some problems in our technique or with the reliability of the dataset. So the anticipated problems can be divided into two categories:

1. problems with the dataset; and
2. problems with the technique.

7.2.1 Problems with the dataset and solution

In the datasets that I have used so far, one can be reasonably confident that the base pairs labeled as being part of a binding site are accurate. But no such confidence can be extended to the rest of the promoter region. There may be many, as yet undiscovered, sites therein. This implies that the base pairs labeled as not being part of a binding site could be incorrect. From Table 7.1 (a summary of Table 6.1), we can see that in both datasets (yeast and mouse) there are a number of vectors that are repeated and inconsistent, and about 69% of the yeast data and 20% of the mouse data are inconsistent.

Species	Original	Inconsistent	Unique	Repeat
Yeast	67,782	46,695 (69%)	6,521 (9.6%)	18,858 (27.8%)
Mouse	60,851	12,119 (20%)	32,747 (54%)	20,969 (34.5%)

Table 7.1: Summary of yeast and mouse dataset. All the numbers in the table are in base pairs.

The yeast dataset has many inconsistent data points, and this suggests that this dataset is particularly unreliable. So far to deal with inconsistent and repeated data, I have taken the simplest approach by removing all such vectors (keeping one copy of the consistent vectors). As a result, nearly 90% of the yeast data and 46% of the mouse data has been lost. However, the inconsistent and repeated vectors may be kept in the test set to make it more biologically meaningful whilst using the above mentioned approach for training sets only.

The unreliability of the promoter negative examples and the over-abundance of inconsistent and repeated data points can both be dealt with by introducing the concept of synthetic negative examples. This chapter will detail the results produced by this approach. One of the major changes in the experimental approach and one of the main contributions to knowledge contained in this thesis was to vary the source of the negative examples. In both the yeast and mouse datasets described so far negative examples are just the promoter regions that are not annotated as TFBSs (referred to as promoter negative examples). Here I will introduced two further sources of negative examples namely: *distal negative examples* and *randomised negative examples.* The details of these negative examples are explained below.

7.2.1.1 Distal Negative examples

For this source of negative examples, selected regions have been taken from the mouse genome that are at least 4500 – 5000 bps away from their associated genes. Care has been taken to avoid overlap with other genes and promoter regions. This set of negative examples has been used only for the mouse data from the previous studies. . For the yeast genome the issue is rather complicated as the genes are relatively close together and identifying distal negative examples is not trivial.

This issue is, however, addressed in Chapter 8, where a more up-to-date set of yeast data is examined.

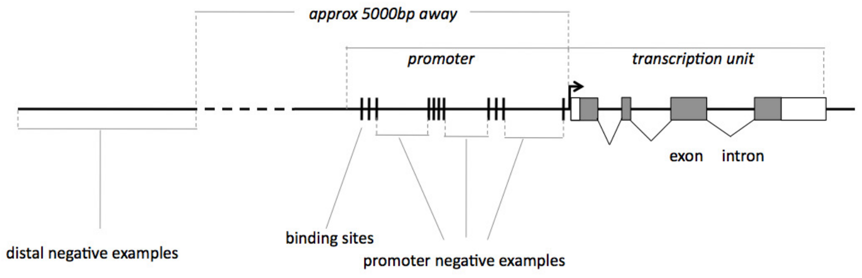


Figure 7.1: Sources of promoter and distal negative examples.

The negative examples were taken from the distal regions of the genome that is from 250 non-coding upstream sequences of the mouse genome and therefore referred to as distal negative examples (see Table 7.2 for details). The first 500bp was picked from each of the distal sequences so that the extracted sequences should be non-coding. Since they are far away from the promoter region, we can be reasonably confident that these regions also have non-regulatory properties. There are in total 124,467 nucleotides in this negative dataset.

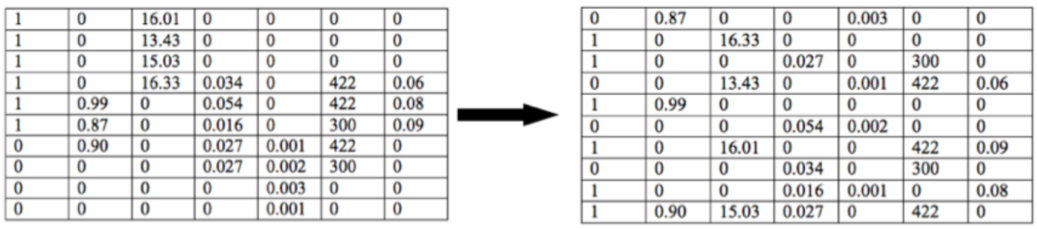
Total number of sequence	250
Total sequence length	124,467 bps
Average sequence length	497.87 bps

Table 7.2: Summary of the mouse dataset that contain distal negative examples.

This second set of mouse data was constructed with the same positive examples (annotated TFBSs) as before taken from *ABS* and *ORegAnno* database.

7.2.1.2 Randomised Negative Examples

The distal negative examples may still have some properties, which may have the function of a binding site and thus may still act as noisy data. To produce the randomised negative examples, all the data vectors labeled as non-binding sites, from distal negative examples, have been placed into a matrix with a column for each base algorithm. Each column is then independently randomly reordered. This randomises each column vector but maintains the overall statistical properties of each algorithm since all the original algorithm values are still there. It is



1	0	16.01	0	0	0	0
1	0	13.43	0	0	0	0
1	0	15.03	0	0	0	0
1	0	16.33	0.034	0	422	0.06
1	0.99	0	0.054	0	422	0.08
1	0.87	0	0.016	0	300	0.09
0	0.90	0	0.027	0.001	422	0
0	0	0	0.027	0.002	300	0
0	0	0	0	0.003	0	0
0	0	0	0	0.001	0	0

0	0.87	0	0	0.003	0	0
1	0	16.33	0	0	0	0
1	0	0	0.027	0	300	0
0	0	13.43	0	0.001	422	0.06
1	0.99	0	0	0	0	0
0	0	0	0.054	0.002	0	0
1	0	16.01	0	0	422	0.09
0	0	0	0.034	0	300	0
1	0	0	0.016	0.001	0	0.08
1	0.90	15.03	0.027	0	422	0

Figure 7.2: Creating randomised negative examples.

unlikely that a real binding site would elicit such randomly joint predictions. The vectors are now correctly labeled as negative, as all the information in the original non-binding site data has been completely lost. These randomised negative examples have been used for both mouse and yeast data. For the mouse data, the dataset for randomised negative examples has been generated by randomly reordering the distal negative examples dataset. This type of dataset was not available for yeast. Therefore, the dataset of randomised negative examples for yeast has been generated by randomly reordering the original yeast dataset.

7.2.2 Problem with the current approach and proposed solution

As previously described (in Section 6.4), we combined the results from different algorithms and performed the pre-processing (discussed in Section 6.4.1) before training. This pre-processed data was then trained to create a model by using standard cross-validation method (described in Section 4.4.1) and made prediction on the biological data. Post-processing (discussed in Section 6.4.4) was done on the predictions to filter out false positives as much as possible. But while establishing a prediction model using standard cross-validation, both the training and validation sets were generated from the same pre-processed data. The validation set was not biologically meaningful whilst the test set being used for final prediction was biologically meaningful. The datasets used so far in all the experiments are imbalanced in nature. Pre-processing makes a balance between two types of examples (positive and negative), which lead to a balanced validation set also. Whereas the final test set is still imbalanced. Thus the training models being used for prediction were not fully attuned for predicting biological data. This could be one reason that the validation rate is so high and in contrast the prediction performance is poor.

To remove this discrepancy during cross-validation, a new modified cross-validation method has been devised (described in Section 6.4.5). This new cross-validation process used validation data of the same type as the final test set including using *F-score* to evaluate performance and using post-processing. However in this chapter we are also changing the training data by using different sources of negative data. It is important to use the real data for the validation set during the cross-validation. Therefore, the negative data is replaced by the original promoter negative data while using this part of the dataset as a validation set.

7.3 Methodology When Replacing Negative Examples

In this method, negative examples have been replaced with either randomised or distal negative examples in the dataset. However, one copy of the real dataset is kept and it will be the source of test sets and validation sets. After mixing the positive examples with the synthetic negative examples, the new dataset is then divided into three parts. The first two-third is used as a training set. The training set undergoes all the pre-processing routines like removing inconsistent and repetitive data points, normalisation, over sampling the minority class and under sampling the majority class, etc.

This training set builds an SVM model using LIBSVM tools, which can then be used to test further data. As mentioned earlier, the cross-validation method used is the new modified cross-validation method. During the cross-validation process, the original dataset provides me with the ability to reconstitute the validation sets for testing. In order to compare results I have used four versions of the cross-validation method. I have either used *Accuracy* or *F-score* as the metric

and have performed cross-validation both with post-processing and without post-processing. In all cases, I have used the resultant model to predict binding sites in the original data using post-processing to filter out short predictions. Figure 7.3 shows a complete workflow of the whole process.
For the yeast data, I have undertaken one experiment namely:

- Replacing the negative examples with randomised negative examples

For the mouse data, I have undertaken two experiments namely:

- Replacing the negative examples with distal negative examples
- Replacing the negative examples with randomised negative examples

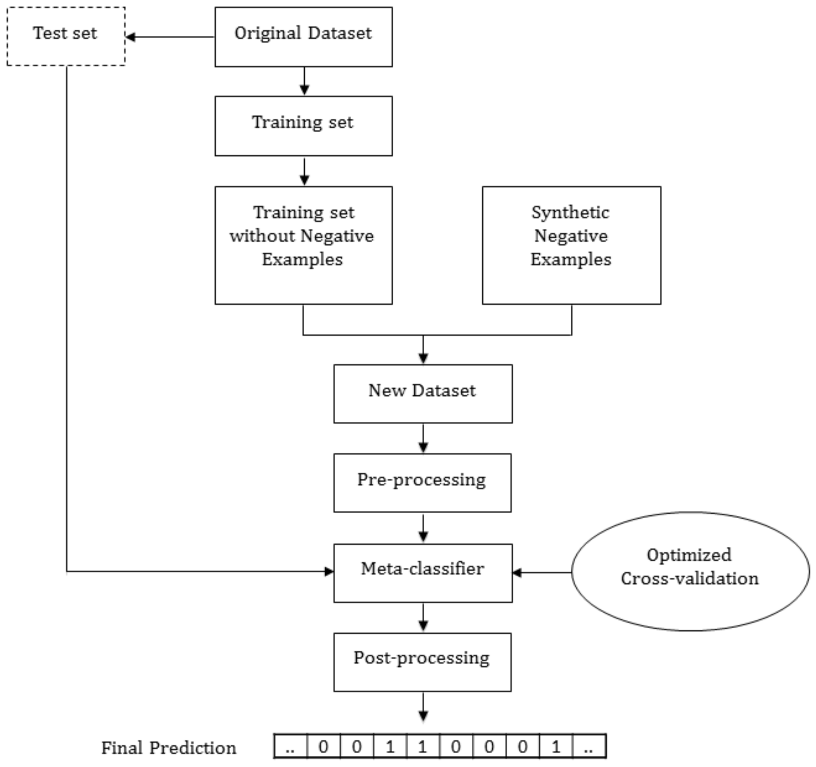


Figure 7.3: Workflow of applying two-class SVM when replacing negative examples.

7.4 Results When Replacing Negative Examples

In this experiment, the negative examples have been replaced in the training set with distal and randomised negative examples (discussed in Section 7.2.1). I have explored two ratios (negative examples: positive examples) 1:1 and 2:1 for under sampling the negative examples and given the result, which has the best classification performance. Distal negative examples have only been used for the mouse data and randomised negative examples have been used for both yeast and mouse data. The results are produced using both filtered and biological test set and post-processing is used only in case of biological test set, as it is obviously not possible to use post-processing on filtered test set as explained in Section 6.4.4.

7.4.1 Results using filtered test Set

7.4.1.1 Replacing negative examples with distal negative examples

Table 7.3 shows the result of classifier performance on filtered test set from the mouse data while using distal negative examples and Table 7.4 shows a brief comparison between the best result from the previous chapter and this.

Cross-validation criterion	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Accuracy	299	240	822	10673	0.267	0.555	0.361	0.022
F-score	705	436	439	10454	0.616	0.619	0.618	0.040

Table 7.3: Results of two-class SVM using the distal negative examples with modified cross-validation methods in the mouse dataset (filtered test set).

	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Mouse+promoter (standard cross-validation)	177	1452	524	9881	0.253	0.109	0.152	0.128
Mouse+distal (modified cross-validation)	705	436	439	10454	0.616	0.619	0.618	0.040

Table 7.4: Comparison between the best results for the mouse data (promoter negative examples vs. distal negative examples) using filtered test set.

The results are very interesting. The first thing to note is that changing the negative examples from proximal to distal brings immediate benefit. Comparing the first row of Table 7.3 and the first row of Table 7.4 we see that the precision has been increased from 0.109 to 0.555 without losing recall. And this is due to the fact that the FPs have fallen from 1452 to 240. With the new data the SVM predicts the presence of binding sites much less often, which is good since most of its original predictions were wrong.

This is the first time in this body of work that a serious dent has been made in the number of FPs produced by any of our predictors.

The next important point to note is that changing the cross-validation criterion with the distal negative examples gives another major jump in performance. This can be seen from rows one and two of Table 7.3. The major change here is that the TPs rises and the FNs falls. It appears that now previously missed binding sites are being found. Overall this produces a huge jump in Recall, which more than doubles and a concomitant jump in F-Score to 0.618 by some way the best predictor so far found for mouse TFBSs. It is interesting to observe that changing the cross-validation criterion with the original promoter negative examples did not bring much benefit at all (see Section 6.7).

Finally, Table 7.4 gives an overall comparison of the new meta classifier against the original SVM. The major improvement in the performance of the trained model is obvious.

It is really exciting to see that, the new cross-validation method with using distal negative examples bring a huge improvement in the classifiers performance.

The *Recall* improves from 25% to 62% and the *Precision* from a merely 11% to an impressive 62%. As a result the *F-score* also observes considerable improvement.

7.4.1.2 Replacing negative examples with randomised negative examples

I now examine whether similar effects can be produced using randomized negative examples. The first set of results in Table 7.5 shows the classifier performance on the filtered test set using the yeast data.

Cross-validation criterion	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Accuracy	9	8	505	9355	0.018	0.529	0.034	0.001
F-score	335	652	179	8711	0.652	0.339	0.446	0.070

Table 7.5: Results of two-class SVM using the randomised negative examples with modified cross-validation methods in the yeast dataset (filtered test set).

As the first row of the data in Table 7.6 is taken from previous studies(Sun *et al.*, 2009a), the confusion matrix was not obtainable. Therefore, no confusion matrices are presented in Table 7.6.

	Recall	Precision	F-score	FP-rate
Yeast+promoter (standard cross-validation)	0.321	0.245	0.278	0.075
Yeast+ randomised (modified cross-validation)	0.652	0.339	0.446	0.070

Table 7.6: Comparison between the best results for the yeast data (promoter negative examples vs. randomised negative examples) using filtered test set.

The first thing to note in Table 7.5 is the very poor performance of the model optimised using *Accuracy*. It has simply learnt to predict the negative class for almost all vectors. However when the model is optimised using *F-Score* a much better classifier is produced, and once again this is the best predictor of yeast binding sites that I have so far produced. Having said that is not as big an improvement as was produced (see Table 7.6) using distal negatives on the mouse genome. The next set of results addresses the question of whether the randomised data works as well with the mouse data as did the distal negatives.

Table 7.7 shows the result of classifier performance on filtered test set from the mouse data while using randomized negative examples and Table 7.8 shows a brief comparison between the best result from the previous chapter and this.

Cross-validation criterion	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Accuracy	119	676	456	10783	0.796	0.787	0.791	0.021
F-score	836	176	191	10831	0.814	0.827	0.821	0.016

Table 7.7: Results of two-class SVM using the randomised negative examples with modified cross-validation methods in the mouse dataset (filtered test set).

	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Mouse+promoter (standard cross-validation)	177	1452	524	9881	0.253	0.109	0.152	0.128
Mouse+ randomised (modified cross-validation)	836	176	191	10831	0.814	0.827	0.821	0.016

Table 7.8: Comparison between the best results for the mouse data (promoter negative examples vs. randomised negative examples) using filtered test set.

The main result here is that the classifier simply does extremely well with this data set. Optimising using *F-Score* is a little better than *Accuracy* and the best classifier has both very high *Recall* and *Precision*. Table 7.8 shows the dramatic improvement over my original mouse predictor.

The following section will present this impact of new negative examples and cross-validation method on the biological test set. Replacing the negative examples in mouse and yeast data caused huge improvement in performance of SVM.

In the next section, I will observe how this result can be extended on biological test set.

7.4.2 Results using biological test set

7.4.2.1 Replacing negative examples with distal negative examples

Table 7.9 shows the results of using distal negative examples in the mouse data with different versions of the new modified cross-validation method and Table 7.10 shows a brief comparison between the best result from the previous chapter and this.

Cross-validation criterion	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Accuracy	289	10	495	18114	0.369	0.967	0.534	0.0006
F-score	513	3	271	18121	0.654	0.994	0.789	0.0002
Accuracy + post-processing	530	2	254	18122	0.676	0.996	0.806	0.0001
F-score + post-processing	530	2	254	18122	0.676	0.996	0.806	0.0001

Table 7.9: Results of two-class SVM using the distal negative examples with modified cross-validation methods in the mouse dataset (biological test set).

	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Mouse+promoter (modified cross-validation)	191	908	593	17216	0.244	0.174	0.203	0.050
Mouse+ distal (modified cross-validation)	530	2	254	18122	0.676	0.996	0.806	0.0001

Table 7.10: Comparison between the best results for the mouse data (promoter negative examples vs. distal negative examples) using biological test set.

It is exciting to see that the results improve considerably when using distal negative examples rather than the previous results using promoter negative examples. It shows that introducing distal negative examples helps the classifier to characterise positive and negative examples properly. However, it is not only introducing distal negative examples that is beneficial but also the new modified cross-validation technique has played a vital role as well. In the absence of post-processing the use of *F-score* in the cross-validation improves results a lot. However filtering out short predictions during cross-validation improves the result in all cases. This may be due to the fact that removing short predictions in the validation set could determine the best parameters for the meta-classifier. A full comparison of results between this chapter and the previous chapter is given in Section 7.6.

7.4.2.2 Replacing negative examples with randomised negative examples

Table 7.11 shows the results of using the randomised negative examples with different versions of the modified cross-validation methods in the yeast dataset and Table 7.12 shows a brief comparison between the best result from the previous chapter and this.

Cross-validation criterion	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Accuracy	1426	49	1382	19736	0.508	0.967	0.666	0.003
F-score	1594	65	1214	19720	0.568	0.961	0.714	0.003
Accuracy+ post-processing	1748	67	1060	19718	0.622	0.963	0.756	0.003
F-score+ post-processing	1748	67	1060	19718	0.622	0.963	0.756	0.003

Table 7.11: The result of using the randomised negative examples with varying cross-validation methods in the yeast dataset (biological test set).

As the first row of the data in Table 7.12 is taken from previous studies(Sun *et al.*, 2005, 2006a,b, 2009a,a,b; Robinson *et al.*, 2006, 2007a,b), the confusion matrix was not obtainable. Therefore, no confusion matrices are presented in Table 7.12.

	Recall	Precision	F-score	FP-rate
Yeast+promoter (standard cross-validation)	0.305	0.371	0.334	0.044
Yeast+ randomised (modified cross-validation)	0.622	0.963	0.756	0.003

Table 7.12: Comparison between the best results for the yeast data (promoter negative examples vs. randomised negative examples) using biological test set.

The results using the yeast data are again greatly improved from the previous promoter negative data of the last chapter.

Table 7.13 shows the results of using randomised negative examples in the mouse data with different versions of the new modified cross-validation method and Table 7.14 shows a quick comparison between the best result from the previous chapter and this. Using randomised negative examples on mouse data has further improved the result.

Cross-validation criterion	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Accuracy	602	264	182	17860	0.768	0.695	0.730	0.020
F-score	542	19	242	18105	0.691	0.966	0.806	0.001
Accuracy+ post-processing	594	0	190	18124	0.758	1.0	0.862	0.00
F-score+ post-processing	594	0	190	18124	0.758	1.0	0.862	0.00

Table 7.13: The result of using the randomised negative examples with varying cross-validation methods in the mouse dataset (biological test set).

	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Mouse+promoter (modified cross-validation)	191	908	593	17216	0.244	0.174	0.203	0.050
Mouse+ randomised (modified cross-validation)	594	0	190	18124	0.758	1.0	0.862	0.00

Table 7.14: Comparison between the best results for the mouse data (promoter negative examples vs. randomised negative examples) using biological test set.

Here replacing promoter negative examples with randomised negative examples that have very low probability of being the part of a binding site, has characterised the data more efficiently than using distal negative examples. One important observation is there are very few false predictions and in case of using post-processing during cross-validation, the *FP-rates* are zero. There is also no difference between using *Accuracy* and *F-score* as cross-validation criterion when using post-processing during cross-validation for both yeast and mouse data. However, the *F-score* does improve without post-processing during cross-validation, and is therefore a better candidate as a cross-validation criterion than *Accuracy*.

7.4.3 Visualisation of the Predictions

Apart from assessing the prediction based on performance measures, I have produced a visualisation of the predictions on the mouse data to see if our predictions are as good as are reflected in our results. The predictions on the yeast data have not been presented due to the lack of proper annotations in the dataset. A fraction of the mouse genome (upstream region of the gene *MyoD1*, *Q8CFN5*, *Vim*, and *U36283*) has been taken and compared best results from different experiments along with prediction algorithms and annotations.

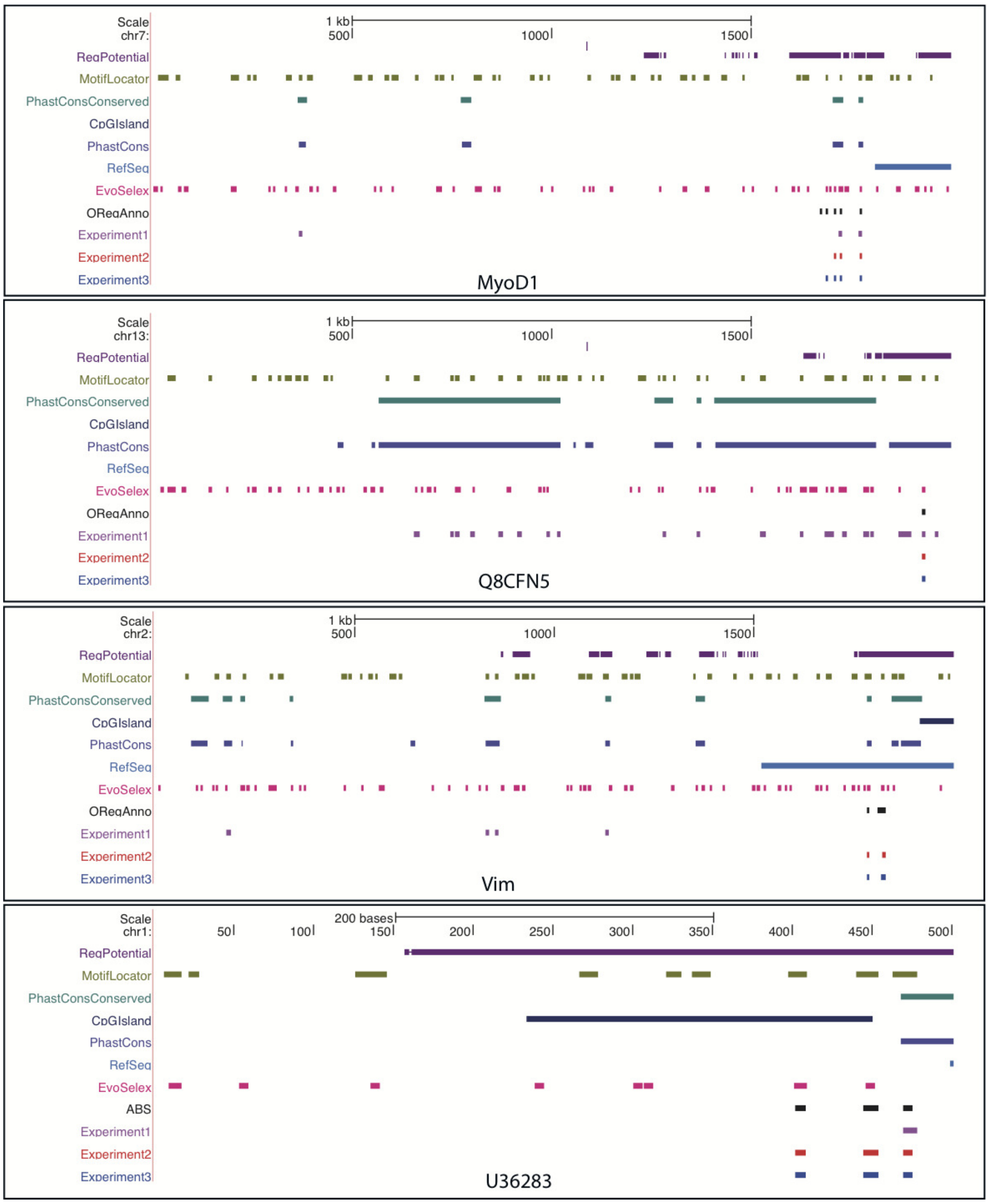


Figure 7.4: Visualization of prediction results on the mouse data.

In Figure 7.4, the upper seven results are from the original prediction algorithms (described in Section 5.4.2) and the next one is experimentally annotated binding sites from *ABS* or *ORegAnno*. The last three results are our best prediction results from three different types of experiments

Experiment 1 is using promoter negative examples (described in Section 6.7.2);

Experiment 2 is using distal negative examples (described in Section 7.4.2.1);

Experiment 3 is using randomised negative examples (described in Section 7.4.2.2).

The figure shows that the prediction algorithms generate a lot of false predictions. On the other hand, using original mouse data (Experiment 1) does not make good predictions. Whereas, using distal or randomised negative examples (Experiment 2 or 3) improves the predictions considerably. The predictions are almost identical to the annotations with the experiment with randomised negative example giving slightly better predictions than that with distal negative examples.

The results in this section are exceptionally pleasing and justify all the experiments using different methodologies. In the next two sections I will expand and analyse the results further and then a full discussion of the results is given in Section 7.6.2.

7.5 Effect of Repetitions and Inconsistent Vectors

Adding new sources of negative examples has a considerable impact on the result. There is one further issue that needs to be investigated. As noted in Table 7.1, there are a lot of repetitive and inconsistent data in the full dataset. These are removed from the training set prior to training. Since the biological test set has the full set of contiguous data replaced in it there is an obvious question that arises regarding whether these repeats bias the results. For this reason, I need to repeat the experiment for the filtered test set since this no longer has the bias of all the repetitions.

While doing so I explored an alternative method of generating training data. Up to this point all the repeated and inconsistent data are removed from the training set after changing the source of the negative data. When the negative data is taken either from distal regions or by randomising the number of repetitions and inconsistent data vectors is likely to be a lot less after this has been done. The question arises whether it makes a difference if the repetitive and inconsistent vectors are removed before replacing the negative examples or after. So I intended to compare the results obtained when removing the repetitions and inconsistencies either before or after using new negative vectors. However, the results generated by this method did not show much improvement in predictions. The methodology, statistics for the dataset sizes under the different replacement and the results on both filtered and biological test sets are given in Appendix C.

7.6 Comparison of Results and Discussion

7.6.1 Comparisons between the base algorithm and all the two-class SVM results

Now let us compare all the results that have been gathered so far on the yeast and mouse datasets using the two-class SVM. Here I have accumulated best results of the base algorithm from Chapter 5, best results of two-class SVM with standard and modified cross-validation method from Chapter 6 and compared with the results obtained in this chapter. In all cases the biological test set has been used. Table 7.15 shows the comparison between results obtained from different types of experiments using the two-class SVM methods on the yeast data. From the results, it is evident that using randomised negative examples with the modified cross-validation method improved the results substantially for the yeast dataset. The *F-score* improved from 29% to 76%.

	Recall	Precision	F-score	FP-rate
Fuzznuc	0.400	0.222	0.285	0.106
Yeast+ standard	0.305	0.371	0.334	0.044
Yeast+ w	0.221	0.499	0.307	0.019
Yeast+ mod	0.229	0.313	0.265	0.071
Yeast+ mod +rand	0.622	0.963	0.756	0.003

Table 7.15: Comparison of performance measures between the best base algorithm, two-class SVM, and two-class SVM with replacing negative examples (yeast data).

In Table 7.15:

Fuzznuc = Best base algorithm

Yeast+standard = Yeast data using standard cross-validation

Yeast + w = Yeast data using windowing

Yeast + mod = Yeast data using modified cross-validation

Yeast + mod + rand = Yeast data with randomised negative examples using new modified cross-validation

Figure 7.5 shows the comparison between *F-scores* obtained from the best base algorithm and different two-class SVM methods on the yeast data. Initially, using the original promoter negative examples produces a poor classification performance, some worse than the base algorithm (*Fuzznuc*). But after replacing negative examples with the randomised negative examples the results improve dramatically. Therefore, we can see that using randomised negative examples and using the new modified cross-validation give by far the best *F-score*.

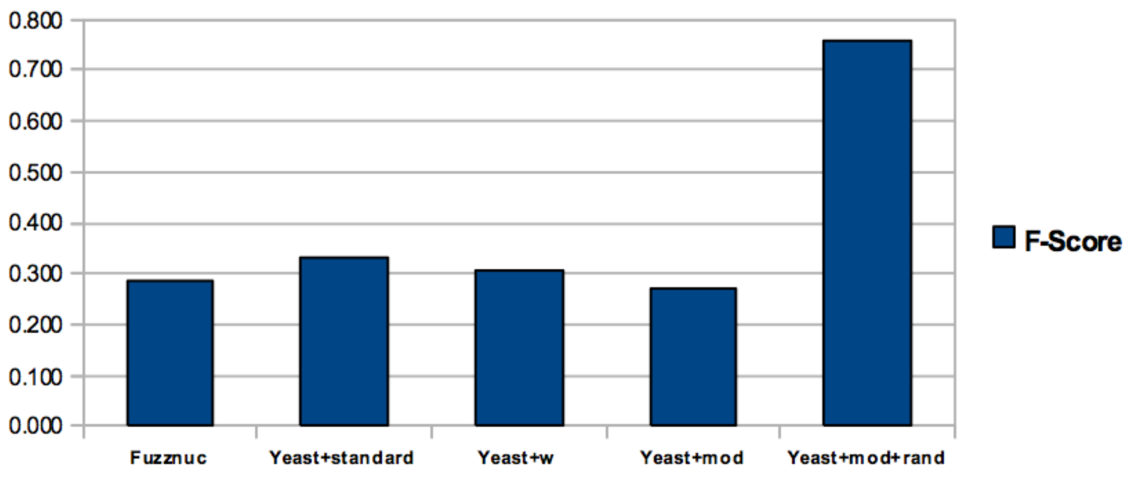


Figure 7.5: Comparison of *F-score* between the best base algorithm, two-class SVM, and two-class SVM with replacing negative examples (yeast data).

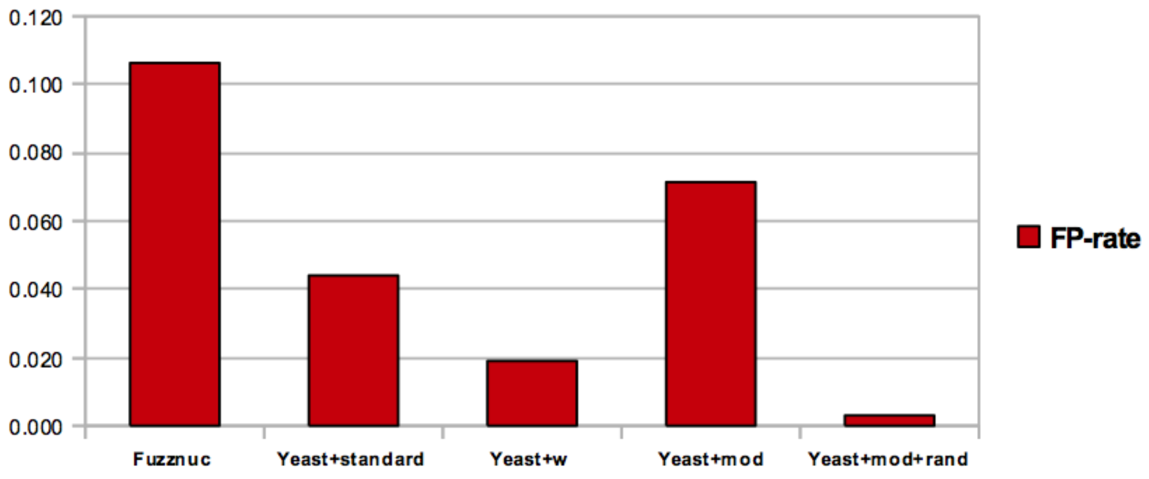


Figure 7.6: Comparison of *FP-rate* between the best base algorithm, two-class SVM, and two-class SVM with replacing negative examples (yeast data).

Figure 7.6 shows the comparison between *FP-rate* obtained from the best base algorithm and different two-class SVM methods on the yeast data. The last method where negative examples have been replaced by randomised negative examples using modified cross-validation has the lowest *FP-rate*. Here, the *FP-rate* has decreased considerably. This proves that using negative examples that completely lack biological properties can characterise the data quite well, as the classifier could predict the binding sites in the unseen test set properly. It indicates that the new method of cross validation is efficient enough to find the best hyper-parameter for training the model for any cross-validation criterion with post-processing.

Table 7.16 shows the comparison between results obtained from different two-class SVM methods on the mouse data. Clearly, using randomised negative examples with modified cross-validation method improved the results substantially for the mouse dataset. The *F-score* improved from 13% to 86% while comparing to the best base algorithm.

	Recall	Precision	F-score	FP-rate
EvoSelex	0.348	0.080	0.130	0.172
Mouse+ standard	0.213	0.113	0.148	0.073
Mouse+ w	0.476	0.129	0.203	0.139
Mouse+ mod	0.244	0.174	0.203	0.050
Mouse+ mod +dist	0.676	0.996	0.806	0.0001
Mouse+ mod +rand	0.758	1.0	0.862	0.000

Table 7.16: Comparison of performance measures between the best base algorithm, two-class SVM, and two-class SVM with replacing negative examples (mouse data).

In Table 7.16:

EvoSelex = Best base algorithm

Mouse = Mouse data using standard cross-validation

Mouse + w = Mouse data using windowing

Mouse + mod = Mouse data using modified cross-validation

Mouse + mod + dist = Mouse data with distal negative examples using new modified cross-validation

Mouse + mod + rand = Mouse data with randomized negative examples using new modified cross-validation

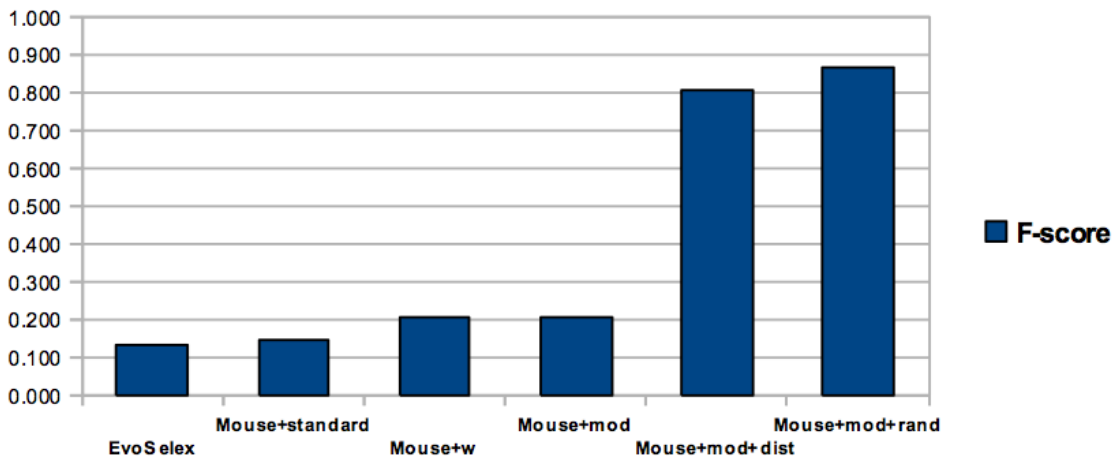


Figure 7.7: Comparison of *F-score* between the best base algorithm, two-class SVM, and two-class SVM with replacing negative examples (mouse data).

Figure 7.7 shows the comparison between *F-scores* obtained from the best base algorithm and different two-class SVM methods on the mouse data. The last method where negative examples have been replaced by randomised negative examples using modified cross-validation has the best *F-score*. Unlike the yeast data, the new modified cross-validation method clearly gave better classification performance than that of the base algorithm (*EvoSelex*) and the two-class SVM method with standard cross-validation. The use of distal negative data is nearly as good as using the randomised negative data.

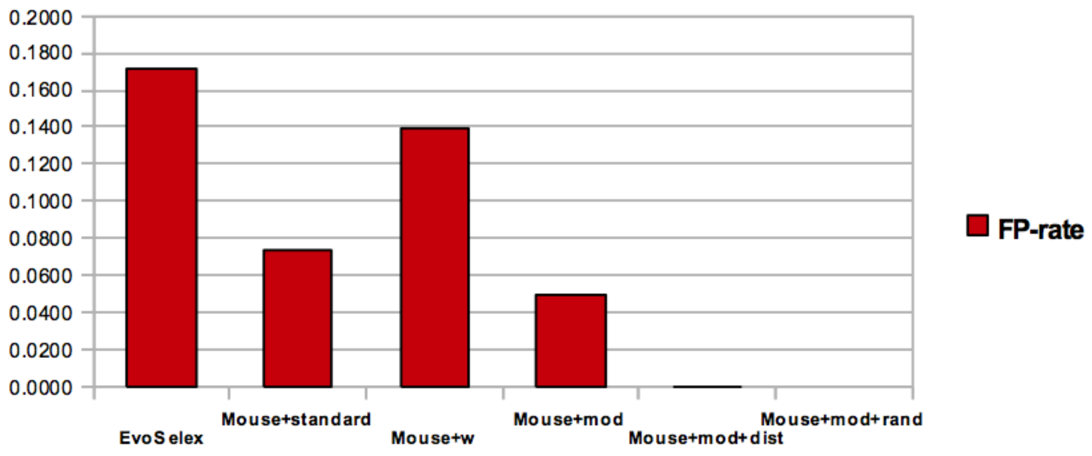


Figure 7.8: Comparison of *FP-rate* between the best base algorithm, two-class SVM, and two-class SVM with replacing negative examples (mouse data).

Figure 7.8 shows the comparison between *FP-rate* obtained from the best base algorithm and different two-class SVM methods on the mouse data. The last method where negative examples have been replaced by randomised negative examples using modified cross-validation has the lowest *FP-rate*, with the results using distal negative examples not far behind.

7.6.2 Discussion

In previous studies, it was shown that basic algorithms individually could not produce accurate predictions and consequently produced many false positives. Though the combination of these algorithms using two-class SVM (described in Chapter 6) gave better results than each individual prediction algorithm, there were still a lot of false positives due to the vulnerability of the negative examples in the datasets. Therefore, the idea of replacing negative examples is introduced in this chapter. At first I use a filtered test set for this approach to understand how good the classifier is after training it with new negative examples along with the modified cross-validation method. Tables 7.4 and 7.8 show remarkable improvements on the classifier performance on the mouse data. While using distal negative examples in the mouse data the *F-score* improves from 17% to almost 62%. This is due to the fact that the classifier improved almost all the measures (TP, FP, and FN except TN) in the confusion matrix. The True Negative prediction is decreased by 50% as the previous classifier (mouse+promoter (standard cross-validation)) tried to predict everything as negative examples. However, the False Positive prediction decreases three times which increases the proportion of correct predicted positive examples. The same is true when using random negative examples in the mouse data (see Table 7.8).

By extending the same approach to the biological test sets, we actually observe the same trend of improvements. Due to the nature of the dataset (containing a lot of repeats), the prediction performances are more improved than that of with the filtered test sets. Accumulating all the results together (see Tables 7.15 and 7.16), we can observe that a more than two times improvement in *F-score* occurred by using randomised negative examples for the yeast data. This is due to a huge drop in false predictions (FNs and FPs). The False Negative prediction is decreased by four times and there is very little False Positive prediction, in the case of mouse data it is even zero. The mouse data also observed a huge improvement in *F-score* as well. However as there is a big reduction in *FP-rate*, the possibility of predicting new novel sites become less. This can be due to the fact that the algorithms combined can characterise the annotated examples (examples from the positive class), but cannot do the same promoter negative examples well. These negative examples act as noisy data for the classifier, which is an obstacle to the classifier performance. However, this combination technique works fine with other negative examples (distal and randomised), which have less possibilities of having binding sites in them. This also proves that the provided annotation that had been used are quite correct. As my technique could successfully predict both the binding sites and non-binding sites in promoter regions, this can be a very interesting application on unlabelled data for predicting novel binding sites.

My present results show that a change in the provenance of the negative examples significantly improves the resulting predictions and that implementation of the new cross-validation technique can bring further improvements. Consequently the major result presented here is the beneficial effect of changing the source of the negative examples used in the training data. Along with the modified cross-validation method our procedure can be a step in the right direction for dealing with this type of biological data.

7.7 Summary

In this chapter I have introduced a new technique for generating negative examples along with a new modified cross-validation method. The results have improved considerably from those reported in Chapter 6. It is not only the new cross-validation method that is improving the result but also these dramatic improvement actually come after using negative examples in the training set either from distal or randomised negative examples. The original yeast dataset that I inherited from the previous research is now fairly dated and newer prediction algorithms have recently been applied to yeast genome. I, therefore, have decided to create a new yeast dataset containing predictions from more recent algorithms and replicate all the experiments on the updated yeast dataset in the next chapter.

Chapter 8

Application of Varying Negative Examples on Updated Yeast Datasets

8.1 Introduction

In the previous chapters (Chapters 6 and 7), I gave a comprehensive description of approaches where biological sources of evidence were combined together and this input was classified using a meta-classifier for improving the prediction of transcription factor binding sites. It was shown that the combination of the predictions from different computational algorithms and evidence gives better results than those of the algorithms (evidence) independently. Two datasets from yeast and mouse with several sources of evidence were used in those chapters and the results considerably improved the prediction of TFBSs.

However, when reviewing the results obtained from the yeast dataset, I did not feel they were as reliable as the mouse ones. This is due to the fact that the yeast dataset, its annotated binding sites and the algorithms used in the previous studies (Sun *et al.*, 2005, 2006a,b, 2007, 2008, 2009a,a,b; Robinson *et al.*, 2006, 2007a,b, 2008) are a number of years old and so lacks the most current set of annotations and lacks the benefit of all the most recent work done in the yeast community. The current yeast dataset was mainly used as a test bed to investigate new methods of dealing with the classification task. Moreover, this yeast data has a larger proportion of inconsistent and repetitive vectors in comparison with the mouse data, that is relatively new and has a lower percentage of inconsistent and repetitive vectors in the dataset (see Section 7.2.1).

	Original	Inconsistent	Unique	Repeat
Yeast	67,782	46,695 (69%)	6,521 (9.6%)	18,858 (27.8%)

Table 8.1: Statistics of inconsistencies in the yeast dataset. All the numbers in the table are in base pairs.

From Table 8.1, it is noticeable that 69% of the current yeast dataset has inconsistent vectors which can lead to misleading classifications. Only 9.6% of its data vectors are unique since the dataset has a lot of repetitions. Therefore, the improved results obtained in the previous chapters may be due to the training of a lot of repeated vectors (see Section 6.4.1). In conjunction with this, the algorithms used for the previous yeast data have also not been updated. The aim of this chapter is to introduce a new updated yeast dataset, new and updated prediction algorithms and to incorporate this into the combinatorial approach that I have described so far in this thesis.

8.2 Genomic Data

8.2.1 Data sources

The yeast data in this chapter has been collected from the resources at the UCSC Genome bioinformatics website. The majority of data is originally from the *Saccharomyces Genome Database* [1] and is based on the assembly of sequence of the *S288C* strain (dated June 2008). As of June 18, 2011, the yeast database contains more than 6600 genes, among which 74.60% are verified, 13.15% uncharacterised and 12.24% are annotated as dubious.

[1]http://downloads.yeastgenome.org/

The annotations for transcription factor binding sites have been collected from an open source database for curated, known regulatory elements, *ORegAnno*. The locus of transcription factor binding sites in different chromosomes from this source has been collected from the UCSC Genome bioinformatics website.

8.2.2 Data selection

As seen from the previous section, the yeast data has a large number of verified protein coding genes. It will be computationally intensive to run the combinatorial prediction approach on the full set of yeast genes, so a criterion was set for choosing a subset of genes. The first 200 genes were selected on the basis of those genes having the highest frequency of transcription factors binding to them. For this criterion, I used the mapping of conserved regulatory sites from MacIsaac *et al.* (2006) [1]. This improved mapping was created by analysing genome-wide chromatin immuno-precipitation data for 203 transcription factors for yeast, using two sequence coservation-based motif discovery algorithms, *PhyloCon* and *Converge* (MacIsaac *et al.*, 2006). Detailed information on transcription factors that have different binding p-values (probability) cutoffs and different levels of sequence conservation between different species was obtained from the same source.

There are three types of binding p-value cutoffs reported in the *MacIsaac* dataset: *0.001*, *0.005* and *no cutoff*. For conservation, there are also three different levels calculated, which are *stringent*, *moderate*, and *no conservation*. The combination of these two parameters of the dataset encodes nine different subsets of data with different levels of stringency. From these nine sets of data, I selected two datasets. One with a p-value cutoff 0.001 and stringent conservation, and another one with a p-value cutoff 0.005 and moderate conservation. This has been done to introduce variation in the datasets. From this point, I will denote the yeast data with the p-value cutoff 0.001 and stringent conservation as *Yeast_p0.001_stringent* and the yeast data with the p-value cutoff 0.005 and moderate conservation as *Yeast_p0.005_moderate*.

The *Yeast_p0.001_stringent* contains the most stringent transcription factor binding sites to search for and thereby the lowest number of examples for training.

[1]source:http://fraenkel.mit.edu/

Alternatively, the *Yeast_p0.005_moderate* has more variations in the mapping than the previous one. The TF-Gene mapping dataset from MacIsaac *et al.* (2006) was used to generate the following number of transcription factors versus genes for both datasets (see Table 8.2).

Number of TFs	Number of genes	
	Yeast_p0.001_stringent	Yeast_p0.001_stringent
21-25	-	3
16-20	1	12
11-15	8	56
6-10	100	247
1-5	1817	2597

Table 8.2: Number of TFs vs. Number of genes for *Yeast_p0.001_stringent* and *Yeast_p0.005_moderate* datasets.

From Table 8.2, for each dataset the first 200 genes were selected that had the most transcription factors binding them. One further criterion for selecting genes was that any associated TFBS had to lie within the 500bps upstream regions of a gene. Based on this, 176 genes of the initial 200 satisfied this criterion for *Yeast_p0.001_stringent* set. Using the same criterion, 148 gene promoter sequences were selected for *Yeast_p0.005_moderate* set. The details of these datasets are given below in Table 8.3.

	Yeast_p0.001_stringent	Yeast_p0.001_stringent
Total number of promoter sequences	176	148
Total sequence length	88,000 bps	74,000 bps
Average sequence length	500 bps	500 bps
Average number of TBFS sites per sequence	4.05	3.87
Average number of TBFS sites per sequence	12.34 bps	11.68 bps
Total number of TFBS sites	714	573
TBFS density in total dataset	10.01%	9.04%

Table 8.3: Summary of *Yeast_p0.001_stringent* and *Yeast_p0.005_moderate* datasets.

In this chapter, I used three types of negative examples apart from the promoter negative examples:

i. Distal negative examples

ii. Randomised negative examples

iii. Intronic negative examples

For *distal negative examples*, I have taken intergenic regions from the yeast data mentioned in Section 8.2.1. The average distance between genes in yeast is around 493 bps and the minimum size for a regulatory region is around 172bps (Chen *et al.*, 2011). Therefore, to avoid any overlaps with the regulatory regions, I selected intergenic regions that are more than 1,000bps in length and took 50 bps from either side of the midpoint of the intergenic regions. There are in total 59,994 nucleotides in this negative dataset. The *randomised negative example* dataset is produced by randomising this distal negative example dataset.

Following the success in using negative examples from different sources, I have introduced another new negative examples set named *intronic negative examples* in this chapter. For intronic negative examples, I have randomly selected 75 intronic regions from the yeast data mentioned in Section 8.2.1 and trimmed 10bps of either end off a selected intron as there is possibility of some bps that are close to the end of an exon that have a high degree of sequence conservation. There are in total 33,091 nucleotides in this negative dataset. The details of these two non-randomised datasets are given below:

	Distal negative example dataset	Intronic negative example dataset
Total number of sequences	594	75
Total sequence length	59,994 bps	33,091 bps
Average sequence length	101 bps	441.21 bps

Table 8.4: Summary of the intronic and distal negative example dataset.

8.3 Sources of Evidence

Seven sources of evidence were used as input in this study. Table 8.5 lists the algorithms and biological evidence used in this chapter.

Strategy	Algorithms
Scanning algorithms	Fuzznuc
	MotifLocator
	P-match
Co-regulatory algorithms	MEME
	AlignACE
Phylogenetic data	PhastConsC (conserved)
	PhastConsMC (most conserved)

Table 8.5: The seven sources of evidence used with the new yeast dataset.

I have tried to cover algorithms and evidence from different strategies. These strategies have already been discussed in Chapter 3. After reviewing several literature sources to find good algorithms for the new yeast dataset, I believe that the algorithms and sources of evidence that have been used so far are well-established. Therefore these can be good candidates for using on the new yeast dataset. The scanning algorithm, *Fuzznuc*, has been chosen as one of the seven sources of evidence because it was the best prediction algorithm on the older yeast data (described in Chapter 5). 91 patterns and regular expressions [1] were searched using *Fuzznuc*. *MotifLocator* was also one of the best prediction algorithms on the mouse data and therefore was chosen as another candidate algorithm. For this scanning algorithm, PWMs were used from the *Saccharomyces Cerevisiae Promoter Database* (SCPD) [2]. The two co-regulatory algorithms (*MEME* and *AlignACE*) are well-known prediction algorithms. Default parameters were used for both of the algorithms and all sequences were used together while running the algorithms. The maximum number of motifs (according to Table 8.2) was explored while running the *MEME*. The phylogenetic sources of evidence (*Phast-*

[1]source:http://biochemie.web.med.uni-muenchen.de/YTFD/
[2]http://rulai.cshl.edu/SCPD/

ConsC for conserved and *PhastConsMC* for most conserved) were downloaded from the UCSC genome bioinformatics website.

The only new prediction algorithm that has been introduced in this chapter is *P-match* (Chekmenev *et al.*, 2005) [1]. *P-match* is based on the simultaneous use of a position weight matrix (PWM) taken from *TRANSFAC*. It uses 142 matrices from *TRANSFAC* public database (Chekmenev *et al.*, 2005). The matrix is derived from the alignment of a set of experimentally determined TFBSs and then calculates *d-score* from all the PWMs. A *d-score* measures the similarities between subsequences by calculating from the weights of the specific nucleotide of a sequence from its corresponding PWM. Two types of *d-score* are calculated one for the whole sequence and another for the single core position of the binding site. Two independent cut-off values are estimated for these *d-scores* to reduce false-negatives and false-positives. The overlapping regions are removed by taking the site, which has the highest *d-score* among those sites. A further description of this algorithm is available in Chekmenev *et al.* (2005).

8.4 Statistics of the Algorithms

This section details the results, on the sources of evidence, obtained during the course of this research. The performance of each algorithm has been calculated by comparing the prediction at each individual nucleotide position in the sequence with the annotated values from *ORegAnno*. The calculation of the statistical measures was previously detailed in Section 4.5.3.

8.4.1 Algorithm performance for *Yeast_p0.001_stringent*

These statistics, presented in Table 8.6, were calculated based on the performance across the biological test dataset only. The test sets were generated in the same manner as described in Section 6.4.2.*Fuzznuc* achieves the best *F-score* closely followed by *MotifLocator* and *PhastConsC*. Though *P-match* has the strongest *Precision*, it has a very low *Recall* value. Here, *MEME* has the lowest *FP-rate*.

[1]http://www.gene-regulation.com/cgi-bin/pub/programs/pmatch/bin/p-match.cgi

Algorithm	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Fuzznuc	249	912	647	8192	0.278	0.215	0.242	0.100
MotifLocator	269	1102	627	8002	0.300	0.196	0.237	0.121
P-match	84	275	812	8829	0.094	0.234	0.134	0.030
MEME	9	126	887	8987	0.010	0.067	0.018	0.014
AlignACE	176	1388	720	7716	0.196	0.113	0.143	0.153
PhastConsC	624	3808	272	5296	0.696	0.141	0.234	0.418
PhastConsMC	521	3231	375	5873	0.582	0.139	0.224	0.355

Table 8.6: Performance measures of sources of evidence on *Yeast_p0.001_stringent*.

Figure 8.1 illustrates the variation in *Precision*, *Recall* and *F-score* across the different algorithms and sources of evidence. Note that larger values are preferable for all of these measures.

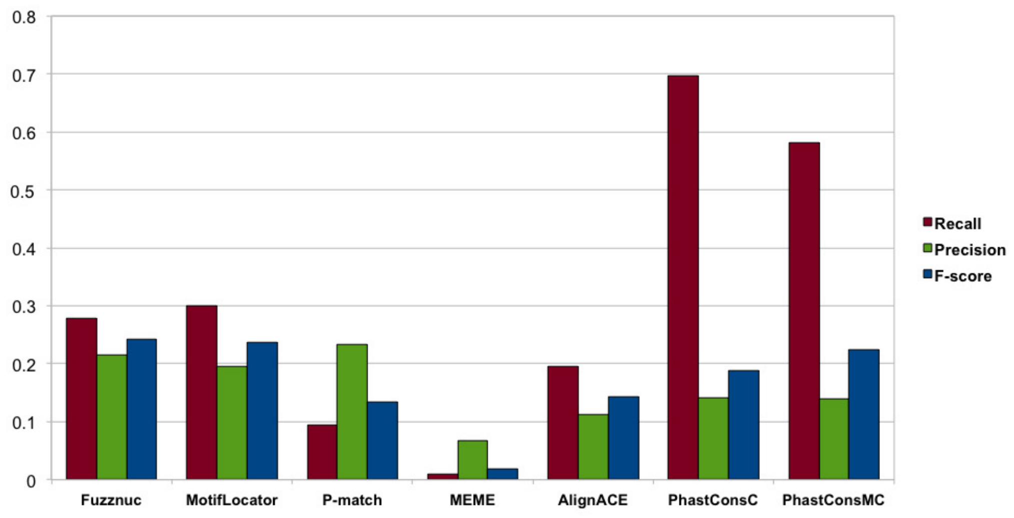


Figure 8.1: Comparison of *Recall*, *Precision* and *F-score* for *Yeast_p0.001_stringent*

Figure 8.2 shows the *FP-rate* scores for each of the algorithms and sources of evidence. Smaller values are to be preferred for this measure.

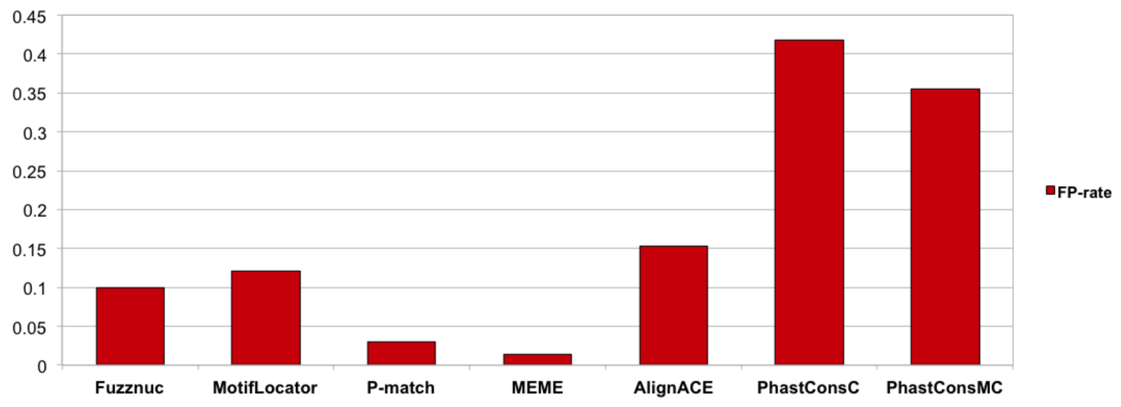


Figure 8.2: Comparison of *FP-rate* for *Yeast_p0.001_stringent*

8.4.2 Algorithm performance for *Yeast_p0.005_moderate.*

Table 8.7 contains the details of the performance of each of the algorithms using the range of statistics chosen to explore the different facets of classification performance. These statistics were calculated based on the performance across the biological test dataset only. In this dataset, *PhastConsMC* achieves the best *F-score* closely followed by *PhastConsC* and *Fuzznuc*. Hence, *Fuzznuc* has the strongest Precision while *P-match* has the lowest *FP-rate*.

Algorithm	TP	FP	FN	TN	Recall	Precision	F-score	FP-rate
Fuzznuc	250	974	682	8094	0.268	0.204	0.232	0.107
MotifLocator	158	988	774	8080	0.169	0.138	0.152	0.109
P-match	46	228	886	8840	0.049	0.168	0.076	0.025
MEME	100	787	126	8978	0.442	0.112	0.178	0.081
AlignACE	281	1400	651	7668	0.302	0.167	0.215	0.154
PhastConsC	749	4351	183	4717	0.804	0.147	0.248	0.480
PhastConsMC	711	3884	221	5184	0.763	0.155	0.257	0.428

Table 8.7: Performance measures of sources of evidence on *Yeast_p0.005_moderate.*

Figure 8.3 illustrates the variation in *Precision*, *Recall* and *F-score* across the different algorithms and sources of evidence. Note that larger values are preferable for all of these measures.

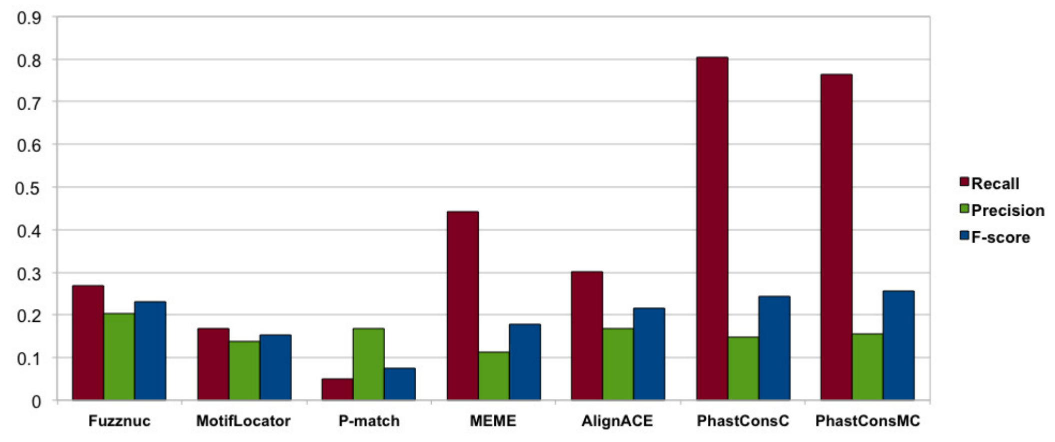


Figure 8.3: Comparison of *Recall*, *Precision* and *F-score* for *Yeast_p0.005_moderate*.

Figure 8.4 shows the FP-rate scores for each of the algorithms and sources of evidence. Smaller values are to be preferred for this measure.

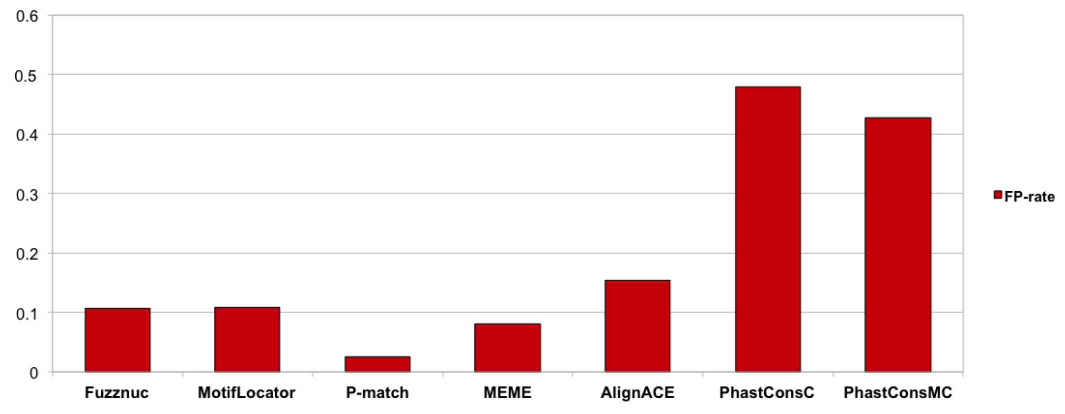


Figure 8.4: Comparison of *FP-rate* for *Yeast_p0.005_moderate*.

8.5 Dealing with Inconsistent and Repetitive Data Vectors

In Section 7.2.1 of the previous chapter, I discussed the problems of the two TF-BSs datasets (yeast and mouse), one of which was having inconsistent and repetitive data vectors. These data vectors were a major setback to the process. The new yeast datasets are also not without this problem. Similar characteristics still persist in the new yeast datasets. Table 8.8 shows the statistics of inconsistent, repetitive and unique data vectors in both yeast datasets, *Yeast_p0.001_stringent* and *Yeast_p0.005_moderate*.

	Original	Inconsistent	Unique	Repeat
Yeast_p0.001_stringent	88,000	34,443 (39.14%)	27,567 (31.33%)	25,990 (29.53%)
Yeast_p0.005_moderate	74,000	28,286 (38.22%)	23,470 (31.72%)	22,244 (30.06%)

Table 8.8: Statistics of inconsistencies and repeats in new yeast datasets (*Yeast_p0.001_stringent* and *Yeast_p0.005_moderate*). All the numbers in the table are in base pairs.

More than 39% of the *Yeast_p0.001_stringent* dataset are inconsistent vectors and almost 30% are repetitive data vectors. Similarly, the percentage of inconsistent, repetitive and unique vectors for the second yeast dataset (*Yeast_p0.005_moderate*) is almost the same as the first one. Figure 8.5 shows the comparison between the inconsistent and unique vectors among all of the three yeast datasets.

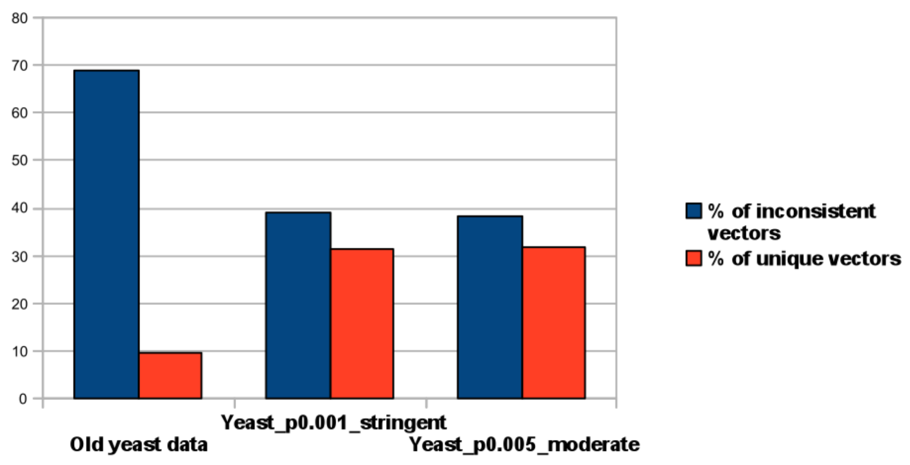


Figure 8.5: Comparison of the different data vectors between the old and the new yeast datasets.

From Figure 8.5, it is apparent that the previous yeast dataset was quite unreliable compared to the new yeast datasets. The new yeast datasets have less inconsistent vectors than the previous one. Therefore, when inconsistent vectors are eliminated, more than 60% of new the datasets could be retained.

8.6 Methodology for a Two-class SVM on the New Yeast Datasets

This section describes the four types of experiments that I have run on both of the new yeast datasets. The experiments are:

a. Using part of the original promoter region deemed to be not a TFBS as negative examples

b. Replacing negative examples with distal negative examples

c. Replacing negative examples with intronic negative examples

d. Replacing negative examples with randomised negative examples

The modified cross-validation method with five-fold cross-validation has been used in all experiments. In addition some pre-processing (data division, normalisation and sampling) has been undertaken on the training set (described in section 6.4). Here I have only explored two ratios 1:1 and 2:1 for under-sampling and reported the results with the ratio, that gave the best classification performance. An exhaustive search method using *F-score* has been used and searched for the best cost ($\boldsymbol{C}$) and gamma ($\boldsymbol{\gamma}$) values. The best model from training with the best $\boldsymbol{C}$ and $\boldsymbol{\gamma}$ values has been used to predict the test set. A post-processing step (with threshold size= 4, 5, 6, and 7) has been done on the prediction of the biological validation and test sets.

8.7 Results for a Two-class SVM on New Yeast Datasets

I have taken the first 30,000 data vectors from both of the new datasets and run the experiments to make them less computationally intensive. The statistics of the subsets of both of the new yeast dataset (*Yeast_p0.001_stringent* and *Yeast_p0.005_moderate*) are given below:

	Original	Inconsistent	Unique	Repeat
Yeast_p0.001_stringent	30,000	10,408 (34.70%)	9,837 (32.79%)	9,755 (32.52%)
Yeast_p0.005_moderate	30,000	10,333 (34.44%)	9,791 (32.64%)	9,876 (32.92%)

Table 8.9: Statistics of inconsistencies and repeats in subsets of new yeast datasets (*Yeast_p0.001_stringent* and *Yeast_p0.005_moderate*). All the numbers in the table are in base pairs.

Two-third of the both datasets (*Yeast_p0.001_stringent* and *Yeast_p0.005_moderate*) are taken as training sets (containing 20,000 data vectors) and one-third is used as a test set (containing 10,000 data vectors). In this case, the test sets are biological test sets containing contiguous data points. Varying different negative examples (mentioned in Section 8.6) on the training sets will form four training sets for each of the two yeast datasets. The statistics for each of these cases are given below in Table 8.10:

Type	Description	Dataset	Size (bps)	
			Training set (after sampling)	Test set
Original	Yeast data containing	Yeast_p0.001_stringent	13,752	10,000
	promoter negative examples	Yeast_p0.005_moderate	13,874	
Distal	Non-binding sites drawn	Yeast_p0.001_stringent	14,209	
	from distal regions	Yeast_p0.005_moderate	13,540	
Randomised	Non-binding sites formed	Yeast_p0.001_stringent	16,015	
	by random permutations	Yeast_p0.005_moderate	15,297	
Intronic	Non-binding sites drawn	Yeast_p0.001_stringent	13,209	
	from intronic regions	Yeast_p0.005_moderate	12,336	

Table 8.10: Statistics of training and biological test sets when varying negative examples.

Now I will present the results of the two-class SVM on the new yeast datasets when varying the source of the negative examples.

8.7.1 Using promoter negative examples

The new modified cross-validation procedure (introduced in Section 6.4.5) was used to analyse the performance in terms of the *F-score* value for both of the new yeast datasets. For the dataset *Yeast_p0.001_stringent*, *Fuzznuc* has the best performance according to the F-score among the seven sources of evidence:

	Recall	Precision	F-score	FP-rate
Fuzznuc	0.278	0.215	0.242	0.100

Table 8.11: Best results from original algorithm (*Fuzznuc*) for *Yeast_p0.001_stringent*.

The results of combining prediction results using an SVM with the new modified cross-validation are given below. The results are generated from the predictions made on the same biological test set mentioned in Section 8.4.1. The confusion matrix for the result is as follows:

	Predictive Negatives	Predictive Positives
Actual Negatives	TN = 6699	FP = 2405
Actual Positives	FN = 378	TP = 518

Table 8.12: Confusion matrix for *Yeast_p0.001_stringent*.

	Recall	Precision	F-score	FP-rate
Yeast_p0.001_stringent	0.578	0.177	0.271	0.264

Table 8.13: Results of the two-class SVM on *Yeast_p0.001_stringent*.

Figure 8.6 shows the comparisons between the *F-scores* and *FP- rates* of the best base algorithm (*Fuzznuc*) and the two-class SVM approach using promoter negative examples for *Yeast_p0.001_stringent*. The *F-score* is only slightly higher than that of the best prediction algorithm. However, the *FP-rate* has worsened in comparison to that of the original prediction algorithms.

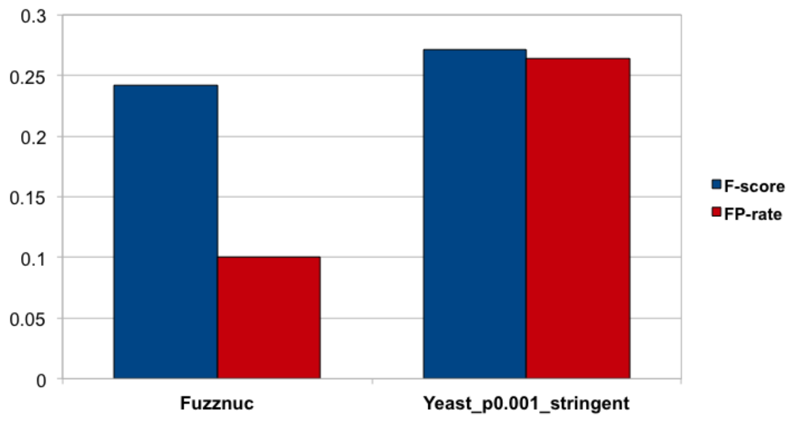


Figure 8.6: Comparison between *F-scores* and *FP-rates* of the best base algorithm (*Fuzznuc*) and *Yeast_p0.001_stringent*.

For the dataset *Yeast_p0.005_conserved*, *PhastConsMC* has the best performance according to *F-score* among the seven sources of evidence:

	Recall	Precision	F-score	FP-rate
PhastConsMC	0.763	0.155	0.257	0.428

Table 8.14: Best results from original algorithm (*PhastConsMC*) for *Yeast_p0.005_moderate*.

The results from combining prediction results using an SVM with the new modified cross-validation are given below. The results are generated from the prediction made on the biological test set. The confusion matrix for the result is as follows:

	Predictive Negatives	Predictive Positives
Actual Negatives	TN = 7554	FP = 1514
Actual Positives	FN = 488	TP = 444

Table 8.15: Confusion matrix for *Yeast_p0.005_moderate*.

	Recall	Precision	F-score	FP-rate
Yeast_p0.005_moderate	0.476	0.227	0.307	0.167

Table 8.16: Results of two-class SVM on *Yeast_p0.005_moderate*.

Figure 8.7 shows the comparisons between the *F-scores* and *FP-rates* of the best base source of evidence (*PhastConsMC*) and the two-class SVM approach using promoter negative examples for *Yeast_p0.005_moderate*.

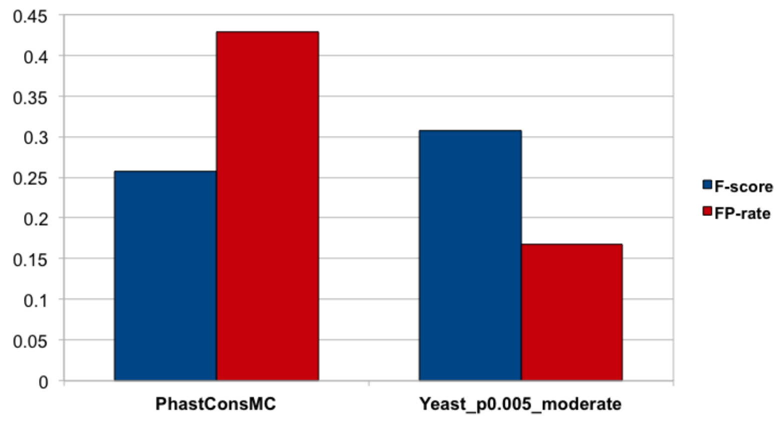


Figure 8.7: Comparison between *F-scores* and *FP-rates* of the best base algorithm (*PhastConsMC*) and *Yeast_p0.005_moderate*.

The *F-score* is only slightly higher than that of the original prediction algorithm (*PhastConsMC*). However, the *FP-rate* is still lower than that of the original prediction algorithms but higher than the lowest *FP-rate* in Table 8.7. As we have not observed any significant improvement just by combining the results of the base algorithms, the next section will show the results after using distal negative examples in place of promoter negative examples on this combined result.

8.7.2 Using distal negative examples

The result of combining prediction results by replacing the negative examples with distal negative example using an SVM with the new modified cross-validation are shown in Tables 8.17 and 8.18. The results are generated from the predictions made on the same biological test set mentioned in Section 8.4.1. The confusion matrix from the result is as follows:

	Predictive Negatives	Predictive Positives
Actual Negatives	TN = 9092	FP = 12
Actual Positives	FN = 324	TP = 572

Table 8.17: Confusion matrix of *Yeast_p0.001_stringent* when using distal negative examples.

	Recall	Precision	F-score	FP-rate
Yeast_p0.001_stringent +distal	0.638	0.979	0.773	0.001

Table 8.18: Results of two-class SVM on *Yeast_p0.001_stringent* when using distal negative examples.

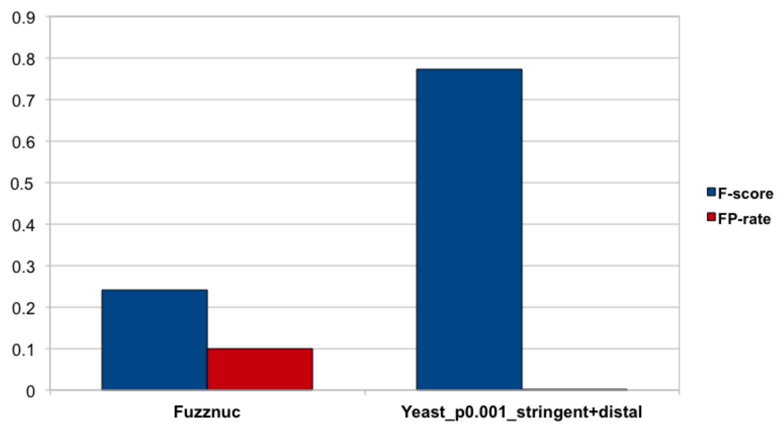


Figure 8.8: Comparison between *F-scores* and *FP-rates* of the best base algorithm (*Fuzznuc*) and *Yeast_p0.001_stringent* with distal negative examples.

Figure 8.8 shows the comparisons between the *F-scores* and *FP-rates* of the best base algorithm (*Fuzznuc*) and the two-class SVM approach using distal negative examples for *Yeast_p0.001_stringent*. The result shows a huge improvement over the best base algorithm as expected. The *F-score* has improved from 24% to 77% and the classifier can predict almost all the positive examples present in the test set (as the *Precision* is almost 98%). There is also a huge reduction in the *FP-rate* as only a few predictions as positive examples have been predicted wrongly.

The results using *Yeast_p0.005_moderate* are given in Tables 8.19 and 8.20:

	Predictive Negatives	Predictive Positives
Actual Negatives	TN = 9031	FP = 37
Actual Positives	FN = 209	TP = 723

Table 8.19: Confusion matrix of *Yeast_p0.005_moderate*when using distal negative examples.

	Recall	Precision	F-score	FP-rate
Yeast_p0.005_moderate +distal	0.775	0.951	0.855	0.004

Table 8.20: Results of two-class SVM on *Yeast_p0.005_moderate* when using distal negative examples.

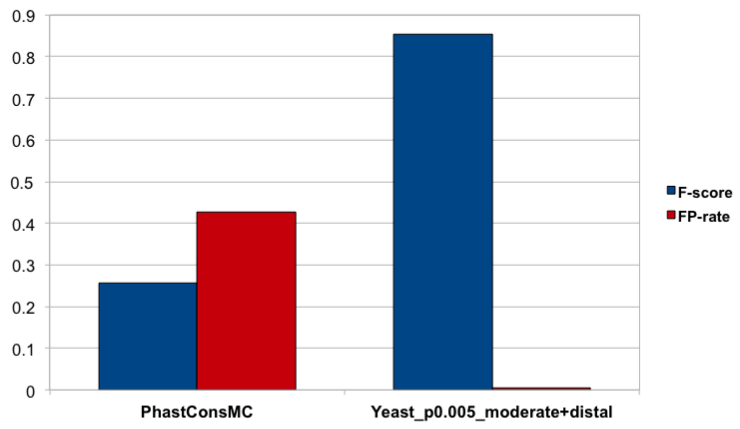


Figure 8.9: Comparison between *F-scores* and *FP-rates* of the best base algorithm (*PhastConsMC*) and *Yeast_p0.005_moderate* with distal negative examples.

Figure 8.9 shows the comparisons between the *F-scores* and *FP- rates* of the best base source of evidence (*PhastConsMC*) and the two-class SVM approach using distal negative examples for *Yeast_p0.005_moderate*. Using distal negative examples on *Yeast_p0.005_moderate* also improves the result tremendously. The *Recall* improves four times from that of the base algorithm and the *F-score* also improves from 26% to 85%. Now we will see the results when using randomised negative examples that are generated from the distal negative examples based on random reordering.

8.7.3 Using randomised negative examples

The result of combining prediction results by replacing the negative examples with randomised negative example using an SVM with the new modified cross-validation are shown in Tables 8.21 and 8.22. The results are generated from the predictions made on the same biological test set mentioned in Section 8.4.1. The confusion matrix from the result is as follows:

	Predictive Negatives	Predictive Positives
Actual Negatives	TN = 9101	FP = 3
Actual Positives	FN = 326	TP = 570

Table 8.21: Confusion matrix of *Yeast_p0.001_stringent* when using randomised negative examples.

	Recall	Precision	F-score	FP-rate
Yeast_p0.001_stringent +randomised	0.636	0.995	0.776	0.0004

Table 8.22: Results of two-class SVM on *Yeast_p0.001_stringent* when using randomised negative examples.

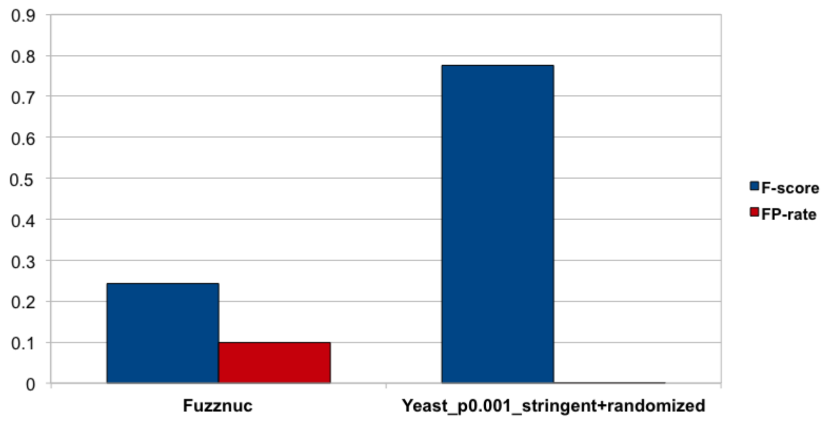


Figure 8.10: Comparison between *F-scores* and *FP-rates* of the best base algorithm (*Fuzznuc*) and *Yeast_p0.001_stringent* with randomised negative examples.

Figure 8.10 shows the comparisons between the *F-scores* and *FP-rates* of the best base algorithm (*Fuzznuc*) and the two-class SVM approach using randomised negative examples for *Yeast_p0.001_stringent*. The result also shows a huge improvement over the base algorithms as expected. The *F-score* has improved from 24% to 78%. There is also a huge reduction in *FP-rate* as only few predictions as positive examples have been predicted wrongly. There is a very slight improvement in the results compared to that of using distal negative examples (see Table 8.18).
The results using *Yeast_p0.005_moderate* are given in Tables 8.23 and 8.24:

	Predictive Negatives	Predictive Positives
Actual Negatives	TN = 9059	FP = 9
Actual Positives	FN = 219	TP = 713

Table 8.23: Confusion matrix of *Yeast_p0.005_moderate*when using randomised negative examples.

	Recall	Precision	F-score	FP-rate
Yeast_p0.005_moderate +randomised	0.765	0.988	0.862	0.001

Table 8.24: Results of two-class SVM on *Yeast_p0.005_moderate* when using randomised negative examples.

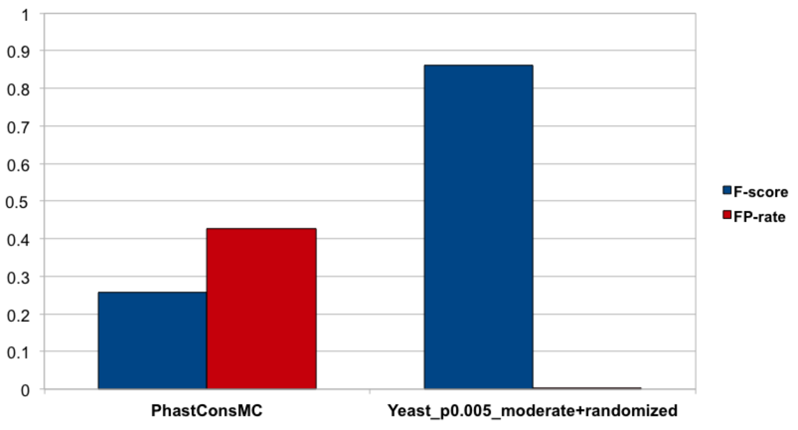


Figure 8.11: Comparison between *F-scores* and *FP-rates* of the best base algorithm (*PhastConsMC*) and *Yeast_p0.005_moderate* with randomised negative examples.

Figure 8.11 shows the comparisons between the *F-scores* and *FP- rates* of the best base algorithm (*PhastConsMC*) and the two-class SVM approach using randomised negative examples for *Yeast_p0.005_moderate*. Using randomised negative examples on *Yeast_p0.005_moderate* also improves the result tremendously and there is a very slight improvement of the results compared to that of using distal negative examples (see Table 8.24).

As mentioned before, this chapter introduces a new source of negative examples; intronic negative examples. We will observe the effect of using intronic negative examples in the following section.

8.7.4 Using intronic negative examples

The result of combining prediction results by replacing the negative examples with intronic negative example using an SVM with the new modified cross-validation are shown in Tables 8.25 and 8.26. The results are generated from the predictions made on the same biological test set mentioned in Section 8.4.1. The confusion matrix from the result is as follows:

	Predictive Negatives	Predictive Positives
Actual Negatives	TN = 9068	FP = 36
Actual Positives	FN = 235	TP = 661

Table 8.25: Confusion matrix of *Yeast_p0.001_stringent* when using intronic negative examples.

	Recall	Precision	F-score	FP-rate
Yeast_p0.001_stringent +intronic	0.737	0.948	0.829	0.004

Table 8.26: Results of the two-class SVM on *Yeast_p0.001_stringent* when using intronic negative examples.

Figure 8.12 shows the comparisons between the *F-scores* and *FP-rates* of the best base algorithm (*Fuzznuc*) and the two-class SVM approach using intronic negative examples for *Yeast_p0.001_stringent*.

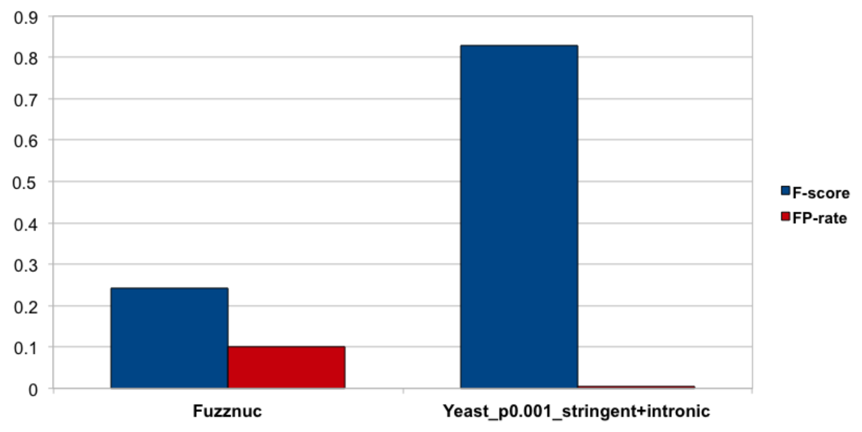


Figure 8.12: Comparison between *F-scores* and *FP-rates* of the best base algorithm (*Fuzznuc*) and *Yeast_p0.001_stringent* with intronic negative examples.

The result also shows a huge improvement over the base algorithms as expected. The *F-score* has improved from 24% to 83%. There is also a very slight improvement of the results compared to that of using distal negative and randomised examples (see Tables 8.18 and 8.22).
The results using *Yeast_p0.005_moderate* are given in Tables 8.27 and 8.28:

	Predictive Negatives	Predictive Positives
Actual Negatives	TN = 9049	FP = 19
Actual Positives	FN = 200	TP = 732

Table 8.27: Confusion matrix of *Yeast_p0.005_moderate*when using intronic negative examples.

	Recall	Precision	F-score	FP-rate
Yeast_p0.005_moderate +intronic	0.785	0.975	0.870	0.002

Table 8.28: Results of two-class SVM on *Yeast_p0.005_moderate* when using intronic negative examples.

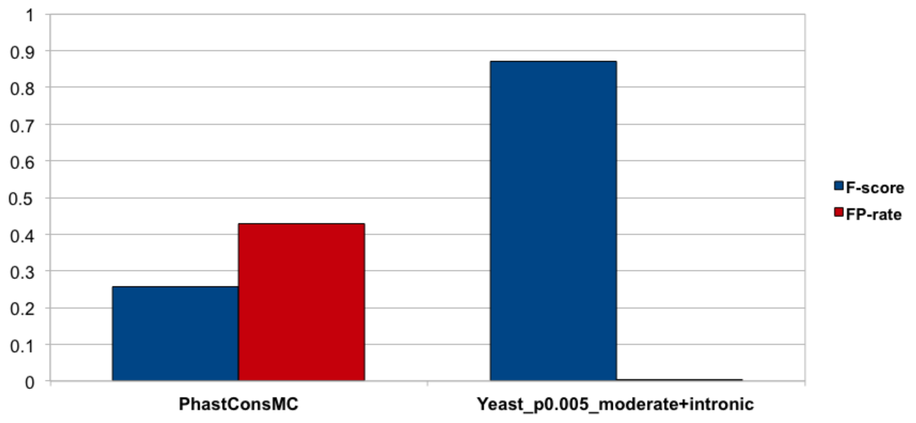


Figure 8.13: Comparison between *F-scores* and *FP-rates* of the best base algorithm (*PhastConsMC*) and *Yeast_p0.005_moderate* with intronic negative examples.

Figure 8.13 shows the comparisons between the *F-scores* and *FP- rates* of the best base source of evidence (*PhastConsMC*) and the two-class SVM approach using intronic negative examples for *Yeast_p0.005_moderate*. Using intronic negative examples on *Yeast_p0.005_moderate* also improves the result greatly compared to the best result of the base algorithm and there is a very slight improvement of the results compared to that of using distal and randomised negative examples (see Tables 8.20 and 8.24).

8.8 Comparisons of the Results and Discussion

Now let us compare all the results that have been gathered so far on the new yeast datasets using the two-class SVM with different negative examples. In this section, I have collected the results of the best base algorithms and the two-class SVM with the modified cross-validation method when using negative examples from the different sources. In all cases the same biological test set has been used. Table 8.29 shows the comparison between results obtained from the different types of experiments using the two-class SVM method on the dataset, *Yeast_p0.001_stringent*.

	Recall	Precision	F-score	FP-rate
Fuzznuc	0.278	0.215	0.242	0.100
Yeast_p0.001_stringent	0.578	0.177	0.271	0.264
Yeast_p0.001_stringent +distal	0.638	0.979	0.773	0.001
Yeast_p0.001_stringent +randomised	0.636	0.995	0.776	0.0004
Yeast_p0.001_stringent +intronic	0.737	0.948	0.829	0.004

Table 8.29: Comparison between the performance measures of the best base algorithm and the two-class SVM with varying negative examples in *Yeast_p0.001_stringent*.

Figure 8.14 shows the comparisons between the *F-scores* of the best prediction algorithm (*Fuzznuc*) and the two-class SVM approach discussed so far. In all cases, the *F-scores* are higher than that of the original prediction algorithm alone. As before the promoter negative data case is only weakly better, while the much more reliable negative data taken from the other sources produces exceptional performances.

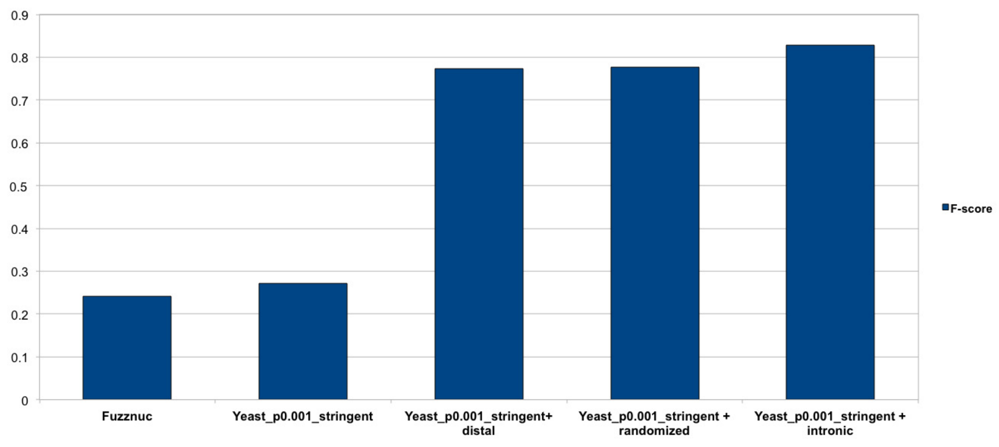


Figure 8.14: Comparison between *F-scores* of the original algorithm (*Fuzznuc*) and *Yeast_p0.001_stringent* with varying negative examples.

Figure 8.15 shows the comparisons between the *FP-rates* of the best prediction algorithm (*Fuzznuc*) and the two-class SVM approach. In all cases, apart from the unreliable promoter negative data result, the *FP-rate* is substantially lower than that of the original prediction algorithm.

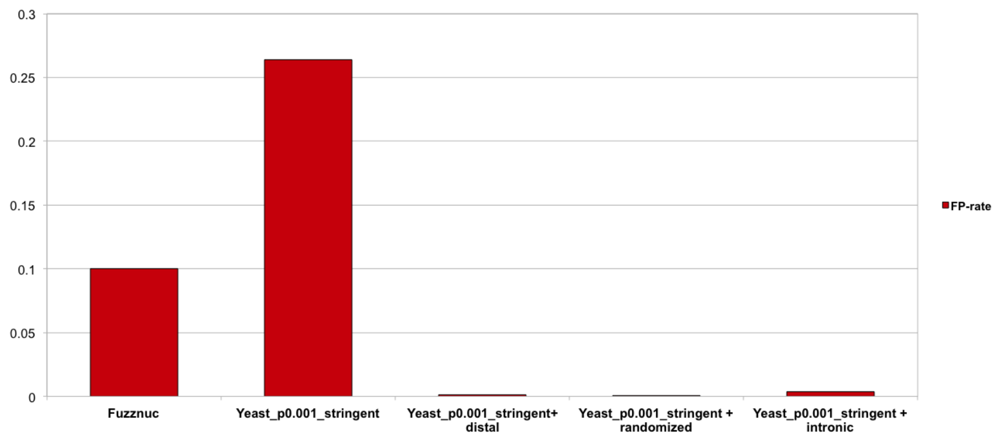


Figure 8.15: Comparison between *FP-rates* of the original algorithm (*Fuzznuc*) and *Yeast_p0.001_stringent* with varying negative examples.

Table 8.30 shows the comparison between results obtained from different types of experiments using the two-class SVM method on the dataset, *Yeast_p0.001_moderate*.

	Recall	Precision	F-score	FP-rate
PhastConsMC	0.763	0.155	0.257	0.428
Yeast_p0.005_moderate	0.476	0.227	0.307	0.167
Yeast_p0.005_moderate +distal	0.775	0.951	0.855	0.004
Yeast_p0.005_moderate +randomised	0.765	0.988	0.862	0.001
Yeast_p0.005_moderate +intronic	0.785	0.975	0.870	0.002

Table 8.30: Comparison between the performance measures of the best base algorithm and the two-class SVM with varying negative examples in *Yeast_p0.005_moderate*.

Figure 8.16 shows the comparisons between the *F-scores* of the best prediction algorithm (*PhastConcMC*) and the two-class SVM approach on *Yeast_p0.005_moderate* discussed so far. As before in all cases, the *F-scores* are much higher than that of the original prediction algorithm, apart from the unreliable promoter negative data case.

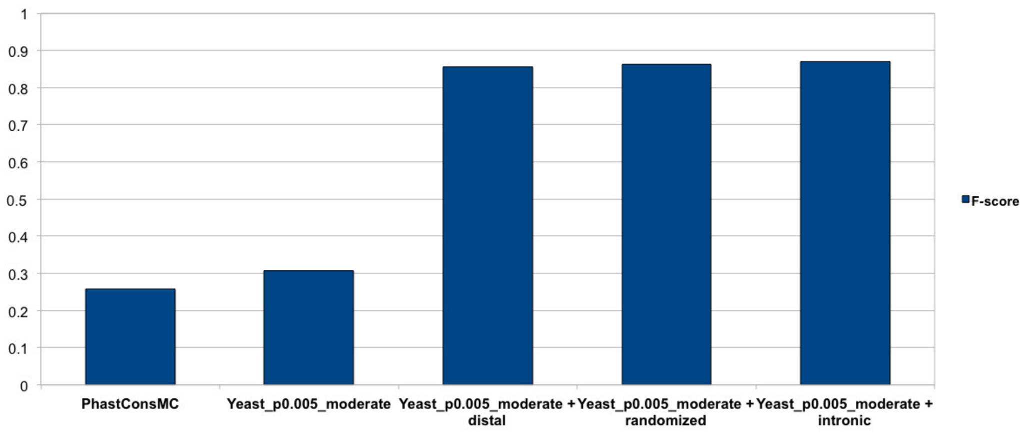


Figure 8.16: Comparison between *F-scores* of the original algorithm (*PhastConsMC*) and *Yeast_p0.005_moderate* with varying negative examples.

Figure 8.17 shows the comparisons between the *FP-rates* of the best prediction algorithm (*PhastConsMC*) and the two-class SVM approach. Again in all cases, the *FP-rate* is lower than that of the original prediction algorithm, apart from the unreliable promoter negative data case.

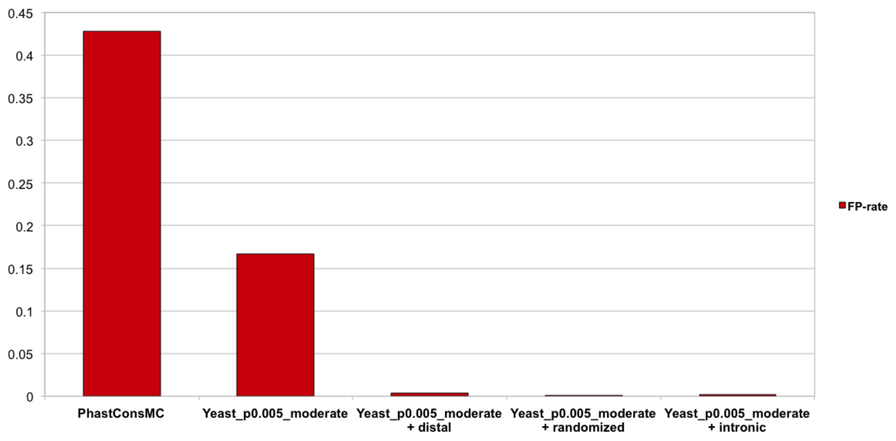


Figure 8.17: Comparison between *FP-rates* of the original algorithm (*PhastConsMC*) and *Yeast_p0.005_moderate* with varying negative examples.

From the results, it is evident that using appropriate negative examples together with an intelligent choice for the modified cross-validation method, as presented in this thesis, improves the TFBS prediction results substantially. In fact if we compare the results with that of *Fuzzznuc* or *PhastConsMC*, we can see that combining the results of different sources of evidence has improved the prediction results better than I could have ever hoped for when I started work on this thesis.

It is clear that when using promoter negative examples, there is only a slight improvement in *F-score* and the *FP-rate* even has a higher value than that of the base algorithms. Clearly the use of promoter negative examples is unreliable. All the examples taken from distal regions and using randomised examples are giving results that are a different order of magnitude in prediction improvement to that of the promoter examples. For *Yeast_p0.001_stringent* dataset (see Table 8.29), the *F-score* jumps to 77% from 24% when using distal negative examples. Using randomised negative examples improves the result slightly again to nearly 78%. In the case of the *Yeast_p0.005_moderate* dataset (see Table 8.30), this follows the same trend of improvement with a best result of 86% using randomised negative examples. As expected the yeast using moderate conservation and a less strict cut-off produces higher values of *F-score*, but this is true across all the results including the base algorithm and just reflects its more relaxed method of production.

The substantial improvements in prediction performance when using other sources of negative data that I had shown for the mouse, as shown in the previous chapter, I have now also shown in the case of the new, updated yeast data as well. Interestingly the best results for yeast were for the case where the negative examples were taken from intronic regions. It gives the best result for the *Yeast_p0.001_stringent* dataset with 83% *F-score* and only 0.4% *FP-rate*, and the best result for the *Yeast_p0.005_moderate* dataset with 87% *F-score* and 0.2% *FP-rate*. This new source of negative data is an innovation introduced in this chapter. This proves the claim made in Section 7.6.2 where it was hoped that combining results from different sources of evidence should substantially improve the prediction result for the new yeast data as well as for the previous datasets.

8.9 Visualisation of the Predictions

Similar to Section 7.4.3, I have produced a visualisation of parts of the new yeast data to see if our predictions are as good as are reflected in our numerical results. A fraction of the yeast genome has been taken and a comparison of the best results from the different experiments along with the prediction algorithms and annotations are shown. For each dataset, upstream regions from five genes have been chosen randomly. For *Yeast_p0.001_stringent* upstream regions of genes *CDS1*, *HAP4*, *MET3*, *SRL1* are selected.

In Figure 8.18, the upper seven result are tracks from the original prediction algorithms (mentioned in Section 8.3) and the next four tracks are our best prediction results from the four different types of experiments mentioned in Section 8.6

Promoter is using promoter negative examples (from Section 8.7.1)

Distal is using distal negative examples (from Section 8.7.2)

Randomised is using randomized negative examples (from Section 8.7.3)

Intronic is using intronic negative examples (from Section 8.7.4)

The last track contains experimentally annotated binding sites from *ORegAnno*. The figures show that the original prediction algorithms generate a lot of false predictions. Also using promoter negative examples from the original yeast data does not produce good predictions. Whereas, using distal, randomised and intronic negative examples improves the predictions considerably. The predictions are almost identical to the annotations with the experiments using randomised negative example giving slightly better predictions than that with the distal negative examples and with the intronic negative examples clearly giving the best results.

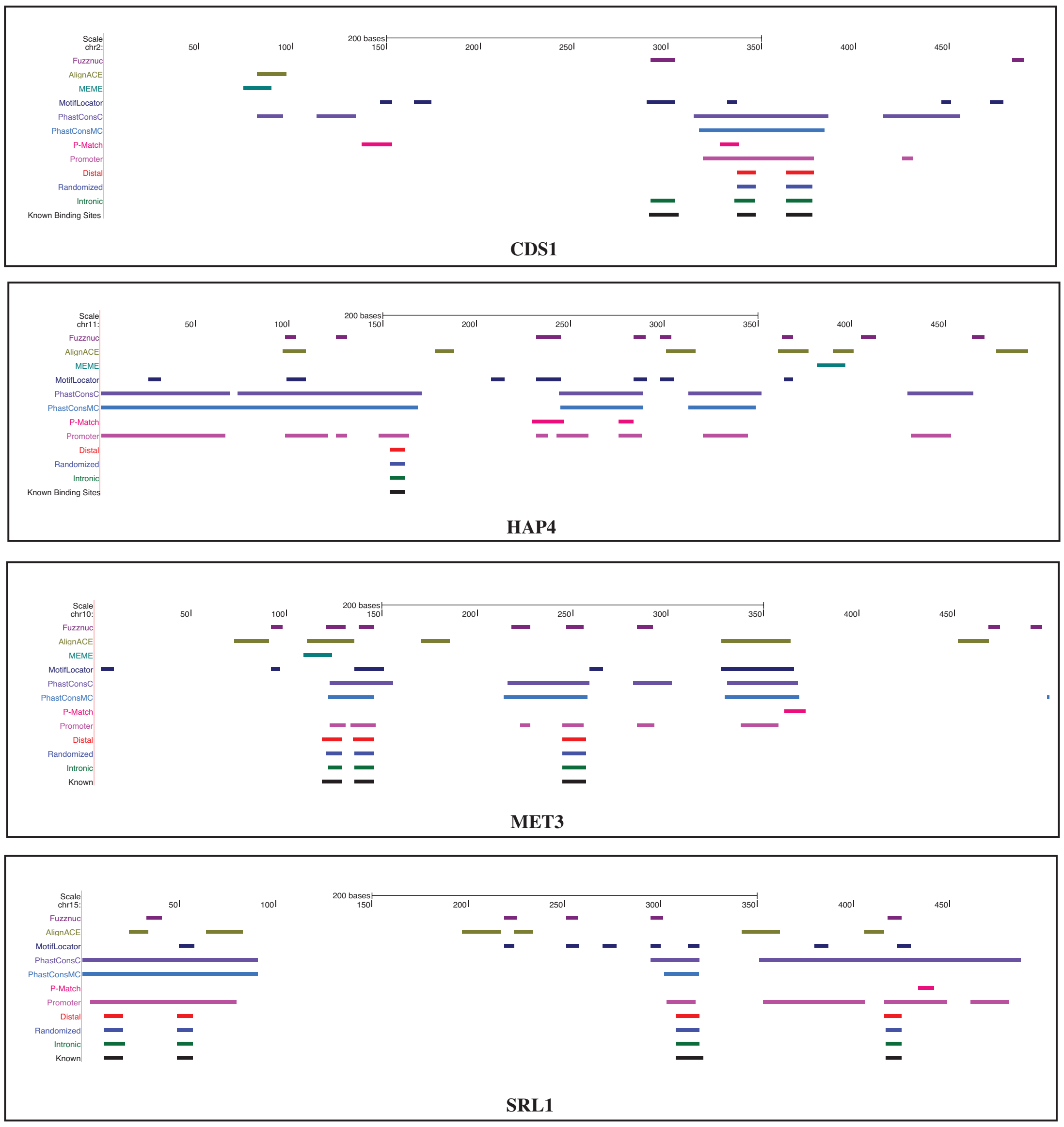


Figure 8.18: Visualisation of computational prediction results on the *Yeast_p0.001_stringent* dataset.

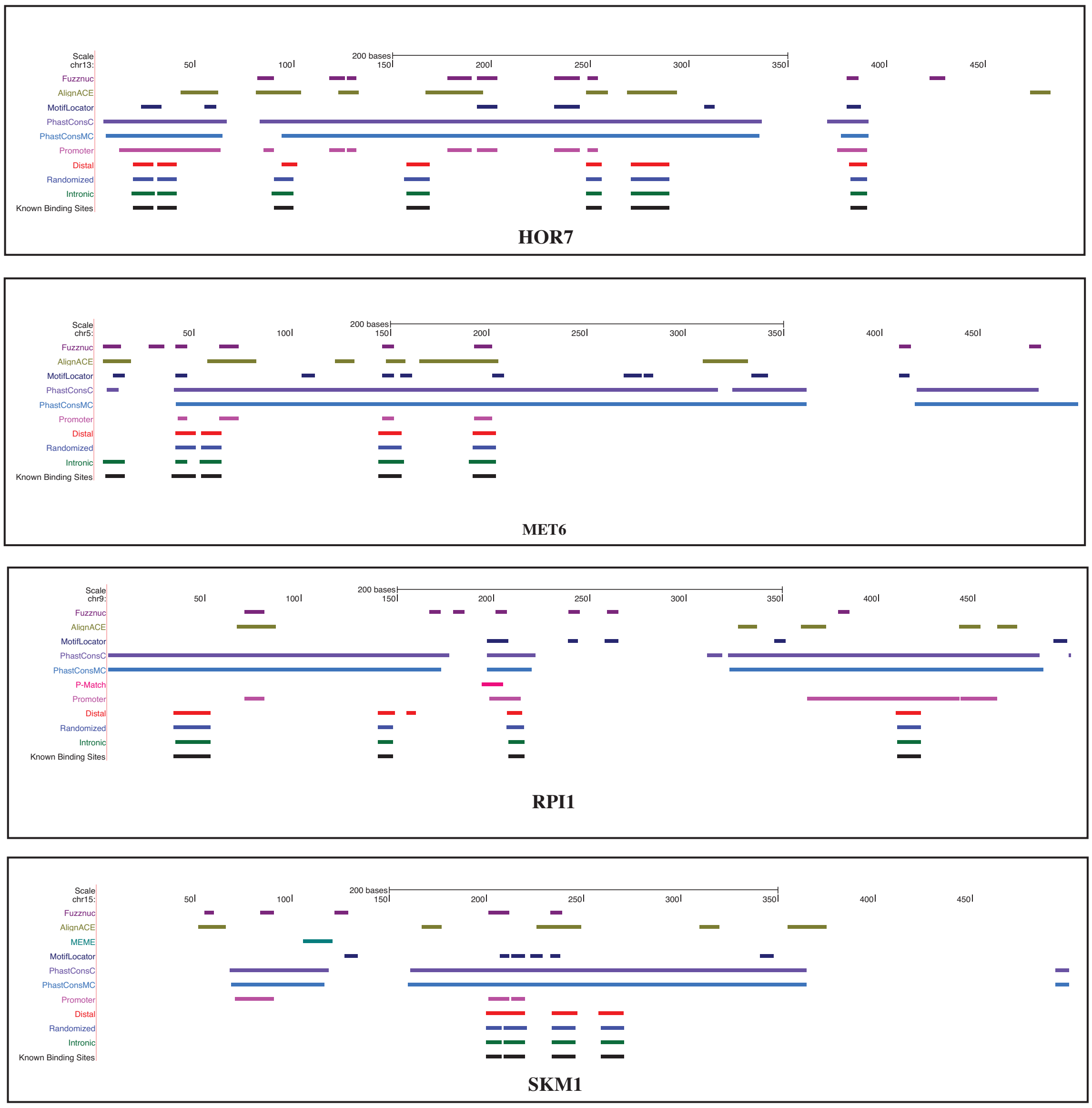


Figure 8.19: Visualisation of computational prediction results on the *Yeast_p0.005_moderate* dataset.

For *Yeast_p0.005_moderate* upstream regions of genes *HOR7*, *MET6*, *RPI1*, and *SKM1* are selected. Figure 8.19 shows the visualisations of the base algorithms along with the predictions by using different negative examples and annotated binding sites from *ORegAnno*.

The results follow the same trend as that of *Yeast_p0.001_stringent* dataset with the intronic negative examples again clearly the best. The results are exceptionally pleasing and therefore justify all the experiments using different methodologies and, in particular, different sources of negative data. The high resolution figures are available in Appendix D.

8.10 Summary

This chapter established the proof of concept produced in Chapter 7, where using negative examples from different sources proved extremely beneficial. New yeast datasets with some variations with binding p-values and conservation stringency have been used with an improved mapping of binding sites with their transcription factors. New, improved and updated algorithms and biological sources of evidence have been used on these datasets to generate two sets of data for experiments. I repeated the same experiments undertaken in the previous chapter (different negative examples using the appropriate modified cross-validation method for classification) and the results obtained showed that using negative examples from different source have a significant effect on improving transcription factor binding site predictions. A new source of negative examples, originating from intronic regions, has also been introduced and this improves the prediction results more than that of using the other negative examples. In summary, combining multiple sources of evidence with more reliable negative examples and an appropriate modified cross-validation method has brought substantial and pleasing improvements in predicting transcription factor binding sites in yeast.

Chapter 9

Conclusions

9.1 Discussion

This thesis addresses three distinct areas in the research of transcription factor binding site (TFBSs) predictions: integrating multiple sources of evidences and using classification technique to improve TFBS predictions; an improved cross-validation method; and an investigation of the sources of negative examples.

In fact the major contribution of my research can be stated quite simply: for the yeast and mouse genome I can predict the position of binding sites with high confidence. Moreover, my predictions are of much higher quality than the predictions of the original base algorithms.

In Chapter 5, the performance of individual algorithms or biological sources evidence, were presented. The results showed that a lot of incorrect predictions had been made. However, different algorithms varied in their strengths and weaknesses. This encouraged the idea that combining the algorithms might bring better predictions. As mentioned previously, this idea was already presented in some earlier works (mentioned in Chapter 6). Most of the previous work was done using yeast as the model organism and some work was also done using the mouse genome. This thesis took things forward by undertaking extensive research on the mouse genome and applying all the techniques for integration (windowing, post-processing) mentioned earlier in the thesis.

In Chapter 6, I ran the basic experimental techniques and found a slight improvement in prediction. The best *F-score* obtained, for the mouse data, was improved from 13% to 20%. According to the sources of evidence, it was assumed that the standard two-class SVM might not be an efficient enough classification technique for the improvement of the predictions of TFBSs. Moreover, there were some questions regarding the quality of the negative data set. Therefore it was interesting to investigate how a one-class SVM performed on this data. However, using a one-class SVM was not as beneficial as expected and it showed very little or no improvement relative to that of a two-class SVM. Nonetheless, an original and important contribution was made in this chapter:

- As mentioned in Chapter 6, the validation data used during the standard cross-validation method was not of the same type as the test set that had been used for prediction. The validation data was taken from the set of pre-processed data whereas the test set was drawn from a contiguous biologically meaningful data set. Moreover, the prediction made on the biological test set was also filtered during post-processing. For this reason, I devised a new modified cross-validation technique, which has biological validation set and post-processing was implemented during cross-validation. This is an important contribution for this thesis. Although the use of the modified cross-validation method did not bring much improvement to results, this seemed to be the right method for cross-validation due to the nature of the classification problem undertaken on the course of this thesis.

This marginal improvement in predictions led to the idea that some part of the negative examples that had been used so far might contain TFBSs, which had not yet been annotated. Therefore, these might act as noisy data for the classifier. Taking this into account, the first step was to change the sources of negative examples. Chapter 7 dealt with this problem and a number of original and important contributions were made:

- Two types of negative examples were introduced from different sources, namely distal, and randomized negative examples. However, using standard cross-validation with these negative examples was not feasible as the validation set in this case would be just a synthetic data set rather than a biological one. Hence, modified cross-validation was an ideal candidate with these new negative examples.

- Using modified cross-validation with different negative examples gave an impressive improvement on TFBSs prediction. For the yeast data set, the *F-score* jumped from 29% to 76% and for the mouse data set, it jumped from 13% to 86%.

- Remarkably for the mouse data set all the predicted binding sites were actually labeled as such: giving a *Precision* of 100% and no False Positives. Therefore, the classifier could not find any newer sites. Some false predictions were still available for the yeast data set, where it may be appropriate to look for novel binding sites.

The question that naturally arises is: why we are now seeing such substantial improvements. Our original hypothesis that the negatively labeled promoter regions might contain many, as yet undiscovered, binding sites has proved to be incorrect. The achieved predictions largely coincide with the original label, hence our high *F-score* values. The results show that the algorithms collectively can identify the binding sites in the promoter regions, but collectively they cannot predict non-binding sites in the promoter regions. However, outside the promoter region (distal region) the algorithms do collectively characterise these regions as containing no binding sites, as the results with the distal negative examples show. It was also found that in fact using randomised negative examples perform even better than distal negative examples.

One thing should be noted: in both Chapters 6 and 7, two types of test data set (filtered and biological) had been used. The filtered test set had been used to demonstrate the classification efficiency of our SVM models on the data suitable for machine learning. The biological test set demonstrated how good our prediction model is at predicting binding sites in biological data.

In Chapter 8, a new yeast data set had been used, in order to validate the techniques I had used in the previous chapter. A new type of negative examples, namely intronic negative examples, was also introduced. As expected, the idea of using negative examples from a source different than the promoter region, with the modified cross-validation method gave much benefit. For both datasets, the *F-score* was more than 80% with almost 98% predicted binding sites were correct.

9.2 Contributions to Knowledge

I now give a brief overview of the major contributions of my thesis:

- I have shown that a synthetically constructed negative data set can bring about a substantial improvement in the prediction of binding site locations.
- I have proposed a modified cross-validation method that gives further improvement to the performance.
- I have demonstrated that the meta-classifier works with not only with the yeast genome but also more complex multi-cellular mouse genome.
- I have shown that my approach also works very well with the latest yeast prediction algorithms and sources of evidence. Here, the best source of negative data proved to be the intronic region.

9.3 Publications

The results contained in this thesis, except the one in Sections 5.5.1 and 6.5, were all generated, obtained and analysed by me. Some of these results had been published in journals and conference proceedings.

- I was a co-author of: Combining experts in order to identify binding sites in yeast and mouse genomic data was published in the journal, Neural Networks (2008). The paper contains the results for the mouse data with windowing mentioned in Section 6.6.3.
- I was the first author and presenter of: Combining experts in order to identify binding sites in genomic data at the workshop, UKCI 2008. This paper contains the initial results for the yeast and mouse data presented in Chapter 6.
- I was the first author of: Using Randomised Vectors in Transcription Factor Binding Site Predictions presented at the conference, ICMLA 2010. This paper contains the results of the effect of repetitions and inconsistent vectors

in the yeast and mouse data, using randomised negative examples on filtered the test set described in Appendix C.

- Again I was the first author and presenter of my fourth paper entitled Effect of Using Varying Negative Examples in Transcription Factor Binding Site Predictions at EvoBIO11 conference. This paper contains the impressive results obtained by using distal and randomised negative examples on the mouse data presented in Sections 6.7.2 and 7.4.2.1. This paper was selected as one of the two best papers in the conference.

A detailed list of publications is given in Appendix E.

9.4 Future Works

According to the knowledge and understanding of the key issues and research directions in the field of TFBSs predictions gained by undertaking the research presented in this thesis, there are scopes to extend the research presented here further. The following main areas would seem to be key starting points for further research:

- One very important extension of the work is the validation of the predictions experimentally. Though in some of cases, 100% *Precision* was achieved, still some false positives could be observed in the predictions. These false predictions can be a good source of novel binding sites where experimentalists could explore. This meets one of the main aims of this thesis about scaling time and cost of experimental approaches.

- Another important extension can be implementing this technique on genomic data that is not yet annotated. This may help to find novel binding sites in the unlabelled data.

- A particularly important extension of this research can be the application of these strategies to other available genomic data. The initial approach, using the yeast data, was able to establish the proof of concept on an organism with simple regulatory organisation. The technique was then assessed using the far more complex genome of the eukaryotic organism, the mouse (which has a more complicated regulatory organisation than that of yeast). However, while the approach undertaken is proven more generally useful, it is essential to demonstrate that the method is successful when applied to a genome with more complex *cis*-regulatory organisation. Taking this into consideration, *D. melanogaster* (fruit fly), *C. elegans* (worm), *R. norvegicus* (brown rat), or even *H. sapiens* (human) can be ideal candidates for choice of such organisms. There are two large projects (ENCODE and modENCODE) that have huge amounts of very specific and focused data and these can be potential sources of both complementary biological evidence and experimentally verified binding sites.

- It will be interesting to see the effect of the classifier with other species. Thus the classifier can be applied on phylogenetically close species. For example: the classifier used for mouse can be used on rat or other close species.

- It can be expected that the approach presented in this thesis is not restricted to the problem of finding transcription factor binding sites. This approach could be applied to other problems with a similar profile. For example, this integration approach could be used in predicting the tertiary structure of a protein, given its primary sequence. Therefore, the integrative process presented in this thesis, if appropriately modified for the problem domain, could be useful to improve prediction accuracy in other fields too.

Appendix A

Nucleic Acid Notations

IUPAC nucleotide code	Base
A	Adenine
C	Cytosine
G	Guanine
T (or U)	Thymine (or Uracil)
R	A or G
Y	C or T
S	G or C
W	A or T
K	G or T
M	A or C
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or T
N	any base
. or -	gap

Table A.1: IUPAC notations for nucleic acids

Appendix B

Position Weight Matrix

B.1 Generating position weight matrix

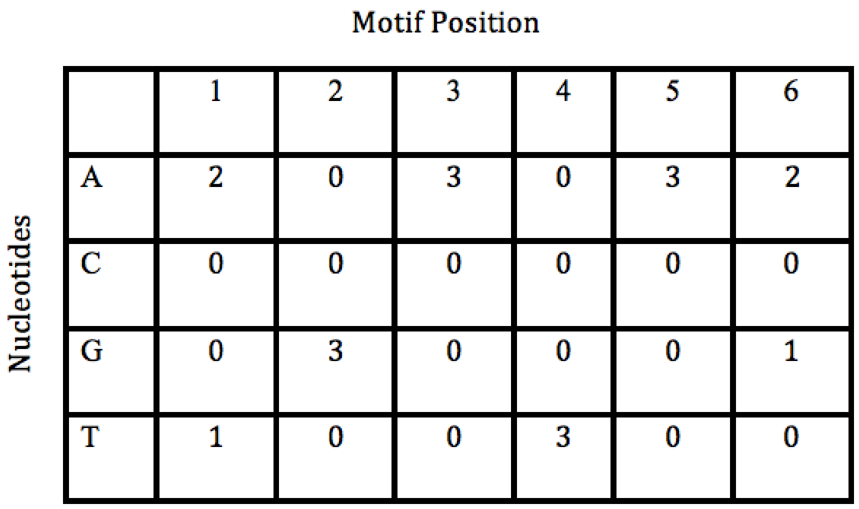
The three sequences are:

Sequence 1 : AGATAA

Sequence 2 : TGATAA

Sequence 3 : AGATAG

The Position Frequency Matrix (PFM) is as follows:



Motif Position

Nucleotides

	1	2	3	4	5	6
A	2	0	3	0	3	2
C	0	0	0	0	0	0
G	0	3	0	0	0	1
T	1	0	0	3	0	0

Figure B.1: Position Frequency Matrix

$$\mathbf{W}_{\boldsymbol{b,i}} = \boldsymbol{log_e}\frac{\mathbf{A}_{\boldsymbol{b,i}}}{\mathbf{B}_{\boldsymbol{b,i}}} = \boldsymbol{log_e}\frac{(\mathbf{C}_{\boldsymbol{b,i}} + \mathbf{B}_{\boldsymbol{b,i}})/(\boldsymbol{Z}+\mathbf{1})}{\mathbf{B}_{\boldsymbol{b,i}}}$$

Here,

$\mathbf{A}_{\boldsymbol{b,i}}$ = Conditional probability that the position is found to be base $\boldsymbol{b}$ in the binding site sequences,

$\mathbf{B}_{\boldsymbol{b,i}}$ = Conditional probability that the position is found to be base $\boldsymbol{b}$ in the non-binding site sequences.

$\mathbf{C}_{\boldsymbol{b,i}}$ = Number of $\boldsymbol{b}$ nucleotide at position $\boldsymbol{i}$

$\boldsymbol{Z}$ = Total number of aligned sequences

For i = 1 ,Profile matrix values (assume $\mathbf{B}_{\boldsymbol{b,i}}$=0.25 for all nucleotides):

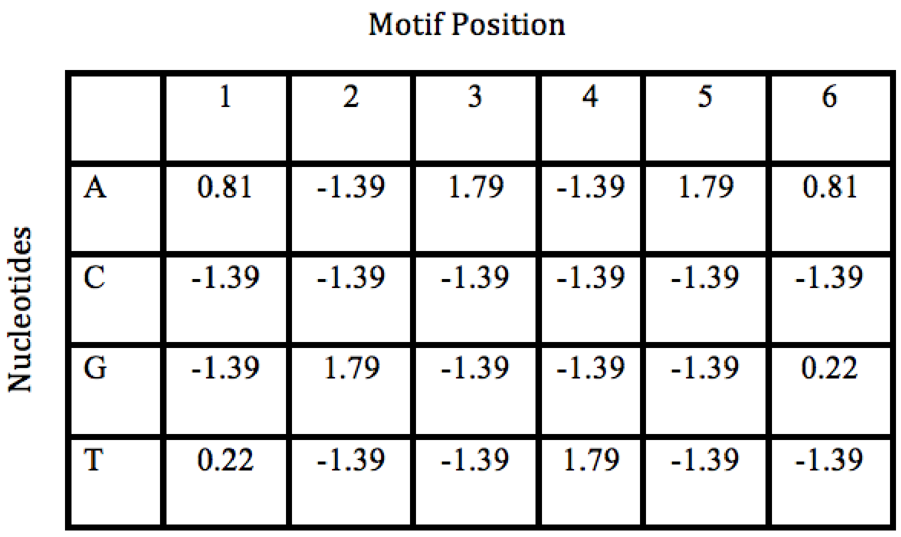
$$\mathbf{W}_{\boldsymbol{A},\mathbf{1}} = \boldsymbol{log_e}\frac{(\mathbf{C}_{\boldsymbol{A},\mathbf{1}} + \mathbf{B}_{\boldsymbol{A},\mathbf{1}})/(\boldsymbol{Z}+\mathbf{1})}{\mathbf{B}_{\boldsymbol{A},\mathbf{1}}} = \boldsymbol{log_e}\frac{(\mathbf{2}+\mathbf{0.25})/(\mathbf{3}+\mathbf{1})}{\mathbf{0.25}} = \mathbf{0.81}$$

$$\mathbf{W}_{C,1} = log_e \frac{(\mathbf{C}_{C,1} + \mathbf{B}_{C,1})/(Z+1)}{\mathbf{B}_{C,1}} = log_e \frac{(0+0.25)/(3+1)}{0.25} = -1.39$$

$$\mathbf{W}_{G,1} = log_e \frac{(\mathbf{C}_{G,1} + \mathbf{B}_{G,1})/(Z+1)}{\mathbf{B}_{G,1}} = log_e \frac{(0+0.25)/(3+1)}{0.25} = -1.39$$

$$\mathbf{W}_{T,1} = log_e \frac{(\mathbf{C}_{T,1} + \mathbf{B}_{T,1})/(Z+1)}{\mathbf{B}_{T,1}} = log_e \frac{(1+0.25)/(3+1)}{0.25} = 0.22$$

The scores at other positions can be calculated in the same manner. Therefore, the Position Weight Matrix (PWM) is:



Motif Position

Nucleotides

	1	2	3	4	5	6
A	0.81	-1.39	1.79	-1.39	1.79	0.81
C	-1.39	-1.39	-1.39	-1.39	-1.39	-1.39
G	-1.39	1.79	-1.39	-1.39	-1.39	0.22
T	0.22	-1.39	-1.39	1.79	-1.39	-1.39

Figure B.2: Position Weight Matrix

Appendix C

Effect of Repetitions and Inconsistent Vectors

C.1 Methodology of Adding Negative Examples Before/After Making Training Data Consistent

In this experiment, both yeast and mouse data have been used. For the yeast data only promoter and randomized negative examples have been used and for the mouse data promoter, distal, and randomised negative examples have been used.

In each experiment, the data has been divided into three parts. Two parts are taken as training sets and one part as test set for filtered test set. This is repeated three times once for each selection of the test set. For the biological test set, the corresponding part has been reconstructed from the original data set, which has consecutive data points. In the training set the minority class (positive examples) has been over-sampled and the majority class (negative examples) has been under-sampled. Here I have continued to explore two ratios 1:1 and 2:1 for under-sampling and quote the results with the best classification performance. The cross-validation has always been done using F-score to evaluate the results on the validation set. An appropriate method has been used in the cross-validation depending on the final test set. That is, for the biological test set the validation set is reconstructed to contain promoter negative data and is post-processed (with size = 4, 5, 6, and 7) to remove small predictions. For the filtered test set the validation set is just reconstructed to contain promoter negative data.

The pseudo-code of the whole process with modified cross-validation using different negative examples and selection of training and test sets is given below: In the training set the minority class (positive examples) has been over-sampled

Pseudocode 3 Finding the best hyper-parameters with modified cross-validation method and and selection of training and test sets i

```
 1: Replace negative examples in the original data set with distal/randomized
    negative examples
 2: Remove repeats and inconsistent vectors
 3: Split data into 3 equal subsets. Take two of the sets as training and one set
    from the original data as test set. Do this 3 times
 4: for each of 3 training-test set splits do
 5:    Split the training data into 5 partitions
 6:    This gives 5 different training sets (4 of 5) and the corresponding validation
       sets (1 of 5). The validation set is reconstituted from the corresponding
       data in the original data set.
 7:    Using sampling to produce more balanced training set
 8:    for each of (cost, gamma) values do
 9:       for each of the 5 training set do
10:          Train an SVM
11:          Measure performance (F-score /Accuracy) on the corresponding re-
             constituted validation set
12:       end for
13:       Average the F-score/Accuracy over the 5 trials
14:    end for
15:    Average the F-score/Accuracy over the 5 trials
16:    Reform the complete training set and train an SVM with the best (cost,
       gamma)
17:    Test the trained model on the unseen test set
18: end for
19: Test the trained model on the unseen test set
```

and the majority class (negative examples) has been under-sampled. Here I have continued to explore two ratios **1 : 1** and **2 : 1** for under-sampling and quote the results with the best classification performance. The cross-validation has always been done using *F-score* to evaluate the results on the validation set. An appropriate method has been used in the cross-validation depending on the final test set. That is, for the biological test set the validation set is reconstructed to contain promoter negative data and is post-processed (with size = 4, 5, 6, and 7) to remove small predictions. For the filtered test set the validation set is just reconstructed to contain promoter negative data.

C.2 Statistics for the yeast data

For yeast data, three types of experiments have been run to obtain the performance measures by applying the model on both the filtered and biological test set. The three experiments I have undertaken are as follows:

a. Yeast data with non-binding sites drawn from promoter regions, which will be denoted as *Yeast_const_data_with_promoter*.

b. Making the yeast data consistent and then replacing negative examples by non-binding sites formed by random permutations, which will be denoted as *Yeast_replace_rand_after_making_const_data*.

c. Replacing negative examples by non-binding sites formed by random permutations and then making yeast data consistent, which will be denoted as *Yeast_replace_rand_before_making_const_data*.

One thing should be noted is that Experiments a and c had been done before and the results are described in the previous sections. The main difference with the previous experiments is that this time I have used three sets of training and test sets and averaged the result. Whereas in the previous experiments, I only used one set of training and test data.
A statistics on each of the case has been given in Table C.1.

Type	Dataset	Negative data points (bps)	Positive data Points (bps)	Size (bps)
Original	Yeast data	59677	8174	67851
Randomised	Non-binding sites formed by random permutation	59608	0	59608
Yeast_const_data_with_promoter	Consistent yeast data	5675	850	6525
Yeast_replace_rand_after_making_const_data				
Yeast_replace_rand_before_making_const_data	Consistent yeast data	24155	1410	25565
Yeast_const_data_with_promoter (after sampling)	Training set (ratio = 1)	3200	3200	6400
		2715	2715	5430
		2585	2585	5170
Yeast_replace_rand_after_making_const_data (after sampling)	Training set (ratio = 1)	3200	3200	6400
		2715	2715	5170
		2585	2585	5170
Yeast_replace_rand_before_making_const_data (after sampling)	Training set (ratio = 2)	10570	5285	15855
		8960	4480	13440
		8670	4335	13005

Table C.1: Statistics of using negative examples in the training set after and before removing inconsistent and repetitive data in the yeast dataset

C.3 Statistics for the mouse data

For mouse data, five types of experiments have been run for obtaining the performance measures by applying the model on filtered and biological test set. The five experiments are as follows:

a. Mouse data with non-binding sites drawn from promoter regions, which will be denoted as *Mouse_const_data_with_promoter*.

b. Mouse data with replacing negative examples by non-binding sites drawn from distal regions after making data consistent, which will be denoted as *Mouse_replace_dist_after_making_const_data*.

c. Mouse data with replacing negative examples by non-binding sites drawn from distal regions before making data consistent, which will be denoted as *Mouse_replace_dist_before_making_const_data*.

d. Mouse data with replacing negative examples by non-binding sites formed by random permutations after making data consistent, which will be denoted as *Mouse_replace_rand_after_making_const_data*.

e. Mouse data with replacing negative examples by non-binding sites formed by random permutations before making data consistent, which will be denoted as *Mouse_replace_rand_before_making_const_data*.

One thing should be noted is that Experiments a, c, and e had been done before and the results are described in the previous sections. The main difference with the previous experiments is that this time I have used three sets of training and test sets and averaged the result. Whereas in the previous experiments, I only used one set of training and test data.
A statistics on each of the case has been given in Table C.2. The key thing to notice from these tables (Tables C.1 and C.2) is that when removing inconsistent and repeated vectors after replacing the negative examples then the data set is much larger. This gives the SVM a better training set.

C.4 Results of Adding Negative Examples Before/After Making Training Data Consistent

Previously in this chapter all the results were given for the biological test set, because ultimately it is the biologists that are most interested in the practical application of our method and they are only interested in biological data. Here, though, we first give the pure results, with no repetitions in the test set, which are of interest to machine learning practitioners.

Type	Dataset	Negative data points (bps)	Positive data points (bps)	Size (bps)
Original	Mouse data	59070	1781	60851
Original	Non-binding sites drawn from distal regions	124467	0	124467
Randomised	Non-bindingsites formed by random permutation	124467	0	124467
Mouse_const_data_with_promoter	Consistent mouse data	31263	1484	32747
Mouse_replace_dist_after_making_const_data	Consistent mouse data	31263	1484	32747
Mouse_replace_rand_after_making_const_data	Consistent mouse data	31263	1484	32747
Mouse_replace_dist_before_making_const_data	Consistent mouse data	36169	1549	37718
Mouse_replace_rand_before_making_const_data	Consistent mouse data	40577	1546	42123
Mouse_const_data_with_promoter (after sampling)	Training set (ratio = 1)	9450	9450	18900
		6132	6132	12264
		5194	5194	10388
Mouse_replace_dist_after_making_const_data	Training set (ratio = 1)	9450	9450	18900
		6132	6132	12264
		5194	5194	10388
Mouse_replace_rand_after_making_const_data	Training set (ratio = 2)	18900	9450	28350
		12264	6132	18396
		10388	5194	15582
Mouse_replace_dist_before_making_const_data	Training set (ratio = 2)	16884	8442	25326
		13846	6923	20769
		12614	6307	18921
Mouse_replace_rand_before_making_const_data	Training set (ratio = 1)	7427	7427	14854
		7840	7840	15680
		6363	6363	12726

Table C.2: Statistics of using negative examples in the training set after and before removing inconsistent and repetitive data in the mouse dataset

C.4.1 Results on the Filtered Test Sets

The average of performance measures of yeast data on the filtered test sets is given below in Table C.3:

	Recall	Precision	F-score	FP-rate
Yeast_const_data_with_promoter	0.767	0.188	0.289	0.568
Yeast_replace_rand_after_making_const_data	0.632	0.511	0.562	0.084
Yeast_replace_rand_before_making_const_data	0.773	0.33	0.448	0.097

Table C.3: Effect of adding negative examples before/after making training data consistent (yeast filtered test set)

Table C.4 shows the average of performance measure of mouse data on the filtered test set.

	Recall	Precision	F-score	FP-rate
Mouse_const_data_with_promoter	0.495	0.089	0.142	0.348
Mouse_replace_dist_after_making_const_data	0.489	0.224	0.294	0.061
Mouse_replace_dist_before_making_const_data	0.672	0.468	0.521	0.077
Mouse_replace_rand_after_making_const_data	0.547	0.417	0.473	0.029
Mouse_replace_rand_before_making_const_data	0.771	0.679	0.695	0.031

Table C.4: Effect of adding negative examples before/after making training data consistent (mouse filtered test set)

From Table C.3, we can see that adding negative examples after removing inconsistent and repetitive data actually gives the best result in yeast set. However, this is quite the opposite of what I had expected. Because, if the negative examples have been replaced after removing inconsistent and repetitive data, we may lose some data points that may characterise positive and negative examples. Therefore, it may become difficult for the meta-classifier to characterise the examples by finding suitable parameters. Therefore, replacing negative examples before making the data set consistent should have given better classification performance. However, Table C.4 gives the expected results where replacing distal and randomised negative examples in the mouse data before making it consistent gives better results than that of replacing them after making the data set made consistent.

With this filtered data set here is still a considerable improvement to be seen in using either distal or randomised examples of negative data in either case. So the use of a new source of negative data is still of considerable benefit even for data with no advantageous repetitions. These predictions have been done on filtered test sets. Now let us discuss the effect on biological test set.

C.4.2 Results on Biological Test Set

Table C.5 shows the average of performance measure of yeast data on biological test set.

	Recall	Precision	F-score	FP-rate
Yeast_const_data_with_promoter	0.359	0.231	0.254	0.248
Yeast_replace_rand_after_making_const_data	0.699	0.974	0.814	0.003
Yeast_replace_rand_before_making_const_data	0.549	0.977	0.701	0.002

Table C.5: Effect of adding negative examples before/after making training data consistent (yeast biological test set)

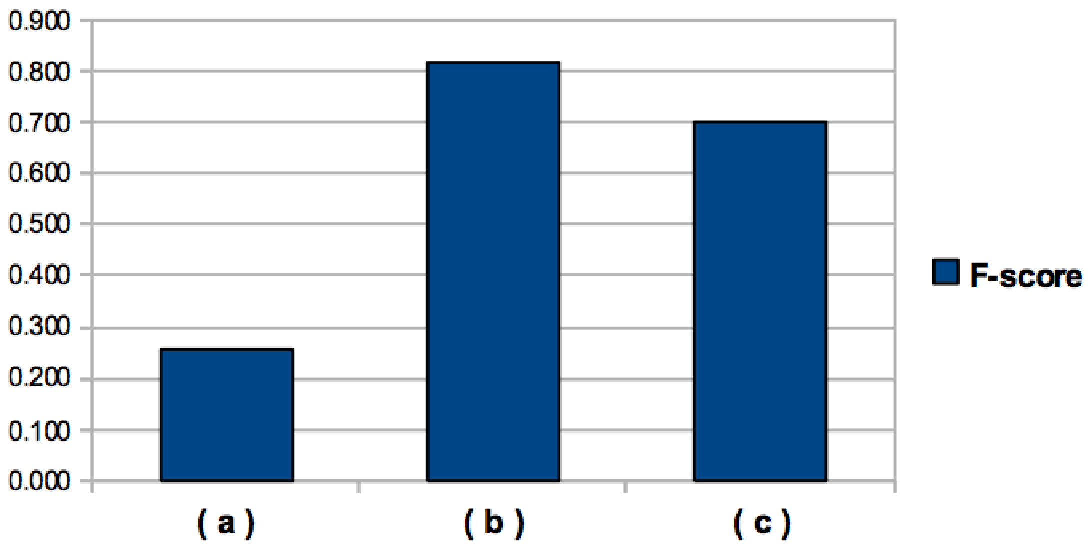


Figure C.1: Comparison of *F-score* by adding negative examples before/after making training data consistent (biological yeast test set)

In Figure C.1,

(a) is Yeast_const_data_with_promoter

(b) is Yeast_replace_rand_after_making_const_data

(c) is Yeast_replace_rand_before_making_const_data

Figure C.1 shows the comparisons between the *F-scores* by adding negative examples before/after making training data consistent in case of biological test set from yeast dataset. There is an increase in *F-score* by using randomised negative examples as expected. However, adding randomised negative examples after removing the inconsistent and repetitive data vectors still gives the best result.

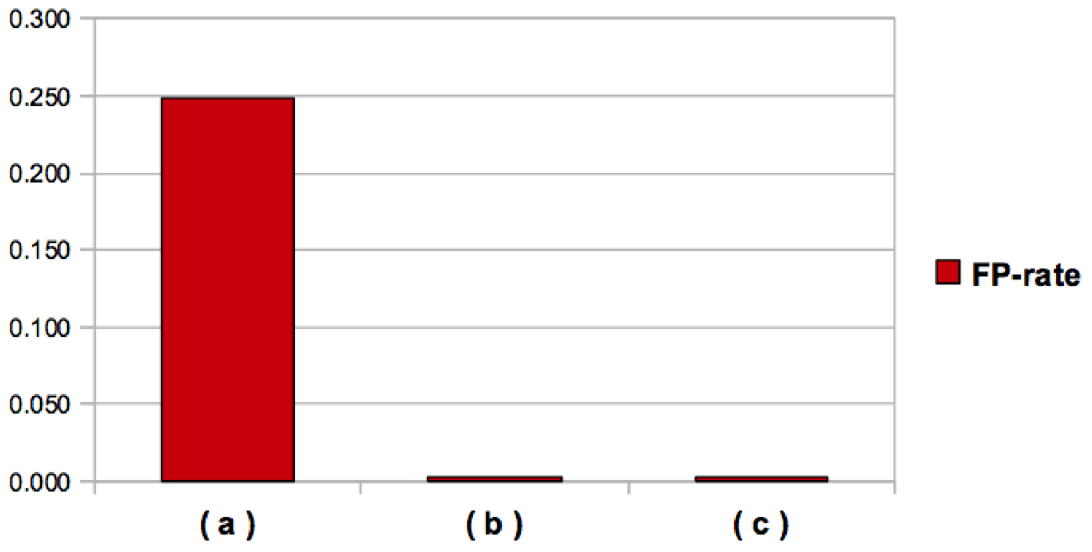


Figure C.2: Comparison of *FP-rate* by adding negative examples before/after making training data consistent (biological yeast test set)

In Figure C.2,

(a) is Yeast_const_data_with_promoter

(b) is Yeast_replace_rand_after_making_const_data

(c) is Yeast_replace_rand_before_making_const_data

Figure C.2 shows the comparison of *FP-rate* by adding negative examples before/after making training data consistent (biological yeast test set). There is a considerable decrease in *FP-rate* by using randomised negative examples as seen before.

Table C.6 shows the average of performance measure of mouse data on the biological test set.

	Recall	Precision	F-score	FP-rate
Mouse_const_data_with_promoter	0.638	0.139	0.227	0.174
Mouse_replace_dist_after_making_const_data	0.709	0.994	0.819	0.0002
Mouse_replace_dist_before_making_const_data	0.738	0.997	0.846	0.0001
Mouse_replace_rand_after_making_const_data	0.767	0.998	0.866	0.00008
Mouse_replace_rand_before_making_const_data	0.815	0.999	0.883	0.00003

Table C.6: Effect of adding negative examples before/after making training data consistent (mouse biological test set)

In Figure C.3,

(a) is Mouse_const_data_with_promoter

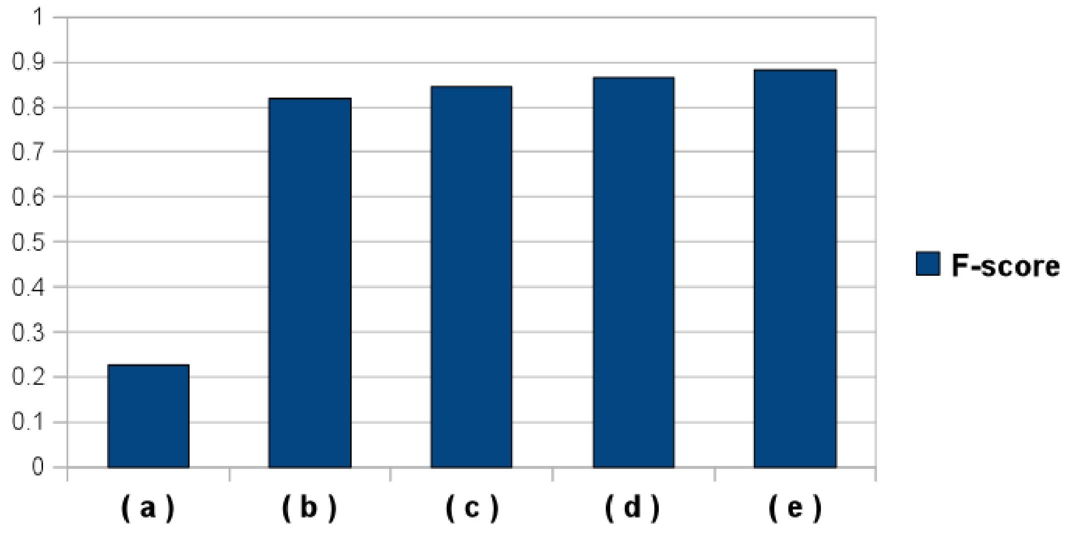


Figure C.3: Comparison of *F-score* by adding negative examples before/after making training data consistent (biological mouse test set)

(b) is Mouse_replace_dist_after_making_const_data

(c) is Mouse_replace_dist_before_making_const_data

(d) is Mouse_replace_rand_after_making_const_data

(e) is Mouse_replace_rand_before_making_const_data

Figure C.3 shows the comparison of *F-score* by adding negative examples before/after making training data consistent, in case of the biological test set from the mouse dataset. There is a large increase in *F-score* after using distal and randomised negative examples as expected. The results of replacing negative examples before making the mouse data consistent produce better results and this is quite expected. In all the cases, randomised negative examples give better results than that of distal negative examples for mouse data and this is consistent with the previous results, showed in Table 7.9 and Table 7.13 in this chapter.

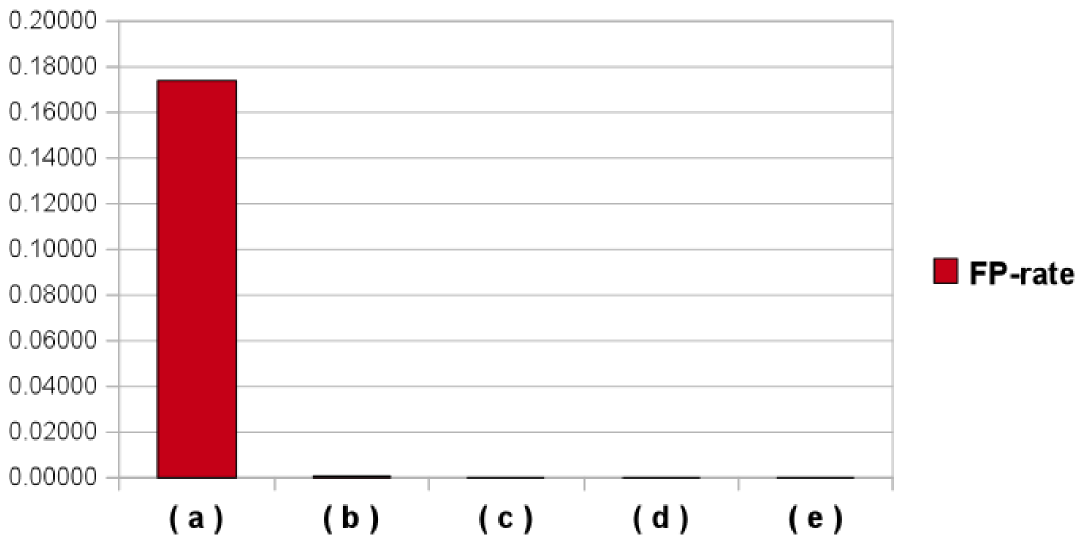


Figure C.4: Comparison of *FP-rates* by adding negative examples before/after making training data consistent (biological mouse test set)

In Figure C.4,

(a) is Mouse_const_data_with_promoter

(b) is Mouse_replace_dist_after_making_const_data

(c) is Mouse_replace_dist_before_making_const_data

(d) is Mouse_replace_rand_after_making_const_data

(e) is Mouse_replace_rand_before_making_const_data

Figure C.4 shows the comparison of *FP-rate* by adding negative examples before/after making training data consistent, in case of the biological test set from the mouse dataset. There is a considerable decrease in *FP-rate* after using distal and randomised negative examples as expected.

As seen before, the improvement found by using distal or randomised negative examples was very good in all cases. However the improvement was not as great when using the filtered data set as when using the biological data set. This can be seen by comparing Table C.3 with Table C.5 and comparing Table C.4 with Table C.6. This is as expected. The biological data set contains repetitions that, once we are predicting them correctly, bias the results upwards.

C.5 Comparison of Results regarding Repetitions and Inconsistent Vectors

Now let us discuss the effect of adding negative examples along with removing repetitions and inconsistent data vectors. Here, the *F-score* of *Yeast_with_random* is the best result that has been taken from Table 7.11, where only one set of training and test sets have been used. The rest of the results, which averages three sets of training and test sets are taken from Table C.5. The results are compared in the following table (Table C.7):

	Recall	Precision	F-score	FP-rate
Yeast_with_random	0.622	0.963	0.756	0.003
Yeast_replace_rand_after_making_const_data	0.699	0.974	0.814	0.003
Yeast_replace_rand_before_making_const_data	0.549	0.977	0.701	0.002

Table C.7: Comparison of the effect of adding negative examples before/after making training data consistent for the yeast data

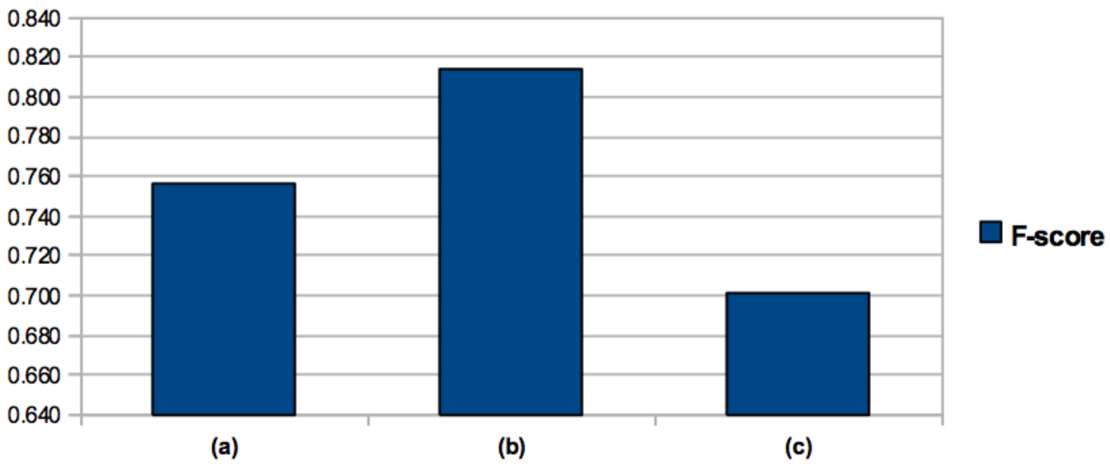


Figure C.5: Comparison of *F-scores* from experiments adding randomised negative examples before/after making training data consistent (yeast data)

In Figure C.5,

(a) is Yeast_with_random

(b) is Yeast_replace_rand_after_making_const_data

(c) is Yeast_replace_rand_before_making_const_data

According to the Figure C.5, the *F-score* has improved when adding randomised negative examples after eliminating the repetitions and inconsistent data vectors. This means the SVM can characterise the positive examples better after removing its repetitions and inconsistent vectors. But as we remove the repetitions and inconsistent data vectors the number of positive vectors also decreased considerably, which may not be enough to characterise the positive examples.

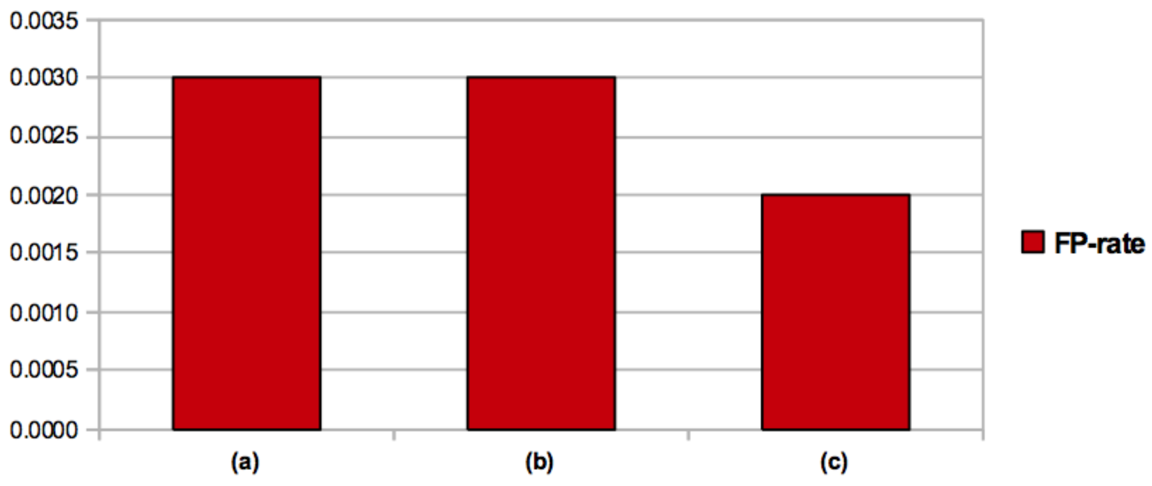


Figure C.6: Comparison of *FP-rates* from experiments adding randomised negative examples before/after making training data consistent (yeast data)

In Figure C.6,

(a) is Yeast_with_random

(b) is Yeast_replace_rand_after_making_const_data

(c) is Yeast_replace_rand_before_making_const_data

The FP-rates are almost the same in each case shown in Figure C.6 It should be noted that the scale on the vertical axis means most of the results are nearly zero.

One interesting observation should be noted that the classifier performance improved even by just using three sets of training and test sets. This implies that the new cross-validation technique along with replacing negative examples is an efficient method to identify the *cis*-binding sites.

	Recall	Precision	F-score	FP-rate
Mouse_with_distal	0.676	0.996	0.806	0.001
Mouse_replace_dist_after_making_const_data	0.709	0.994	0.819	0.0002
Mouse_replace_dist_before_making_const_data	0.738	0.997	0.846	0.0001
Mouse_with_random	0.758	1.0	0.862	0.00
Mouse_replace_rand_after_making_const_data	0.767	0.998	0.866	0.00008
Mouse_replace_rand_before_making_const_data	0.815	0.999	0.883	0.00003

Table C.8: Comparison of the effect of adding negative examples before/after making training data consistent for the mouse data

With the mouse data, in both cases (distal negatives examples and randomised negative examples) a desirable result has been obtained (see Table C.8). Since both results are good, I have combined them in one graph. The *F-score* of *Mouse_with_distal* and *Mouse_with_random* are taken from Tables 7.9 and 7.13 respectively. The other results, which are average from three sets of training and test sets, are taken from Table C.6. The results are compared in Table C.8. In Figure C.7,

(a) is Mouse_with_distal

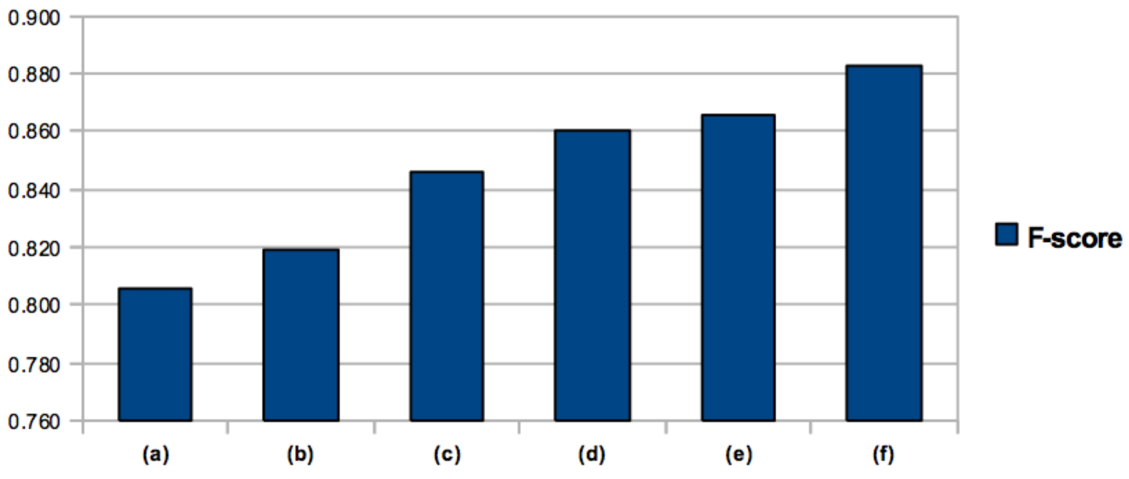


Figure C.7: Comparison of *F-scores* from experiments adding distal and randomised negative examples before/after making training data consistent (mouse data)

(b) is Mouse_replace_dist_after_making_const_data

(c) is Mouse_replace_dist_before_making_const_data

(d) is Mouse_with_random

(e) is Mouse_replace_rand_after_making_const_data

(f) is Mouse_replace_rand_before_making_const_data

Figure C.7 shows that replacing negative examples after removing the repetitions and inconsistent data vectors can bring further improvement to the method in both cases of using distal and randomised negative examples. However, it is not better than the method where distal negative examples are replaced before removing the repetitions and inconsistent data vectors (Figures C.7(b) and C.7(c)). The later process gives enough negative and positive examples to the meta-classifier to characterise the data. On the other hand, using randomised negative examples also exhibits the same type of results. Replacing randomised negative examples before eliminating repetitions and inconsistent data rows actually produces the best *F-score* so far (see Figure C.7(f)). The *F-score* is even better than the best results obtained by using the modified cross-validation method with replacing randomised negative examples (see Table C.8). In Figure C.8,

(a) is Mouse_with_distal

(b) is Mouse_replace_dist_after_making_const_data

(c) is Mouse_replace_dist_before_making_const_data

(d) is Mouse_with_random

(e) is Mouse_replace_rand_after_making_const_data

(f) is Mouse_replace_rand_before_making_const_data

Figure C.8 shows the comparisons between *FP-rates*. The *FP-rates* are almost the same in each case (shown in Figure C.8) and they are all very small in magnitude. The best result (Figure C.8(d)) exhibits no false positives. Replacing both distal and randomised negative examples before and after making the mouse

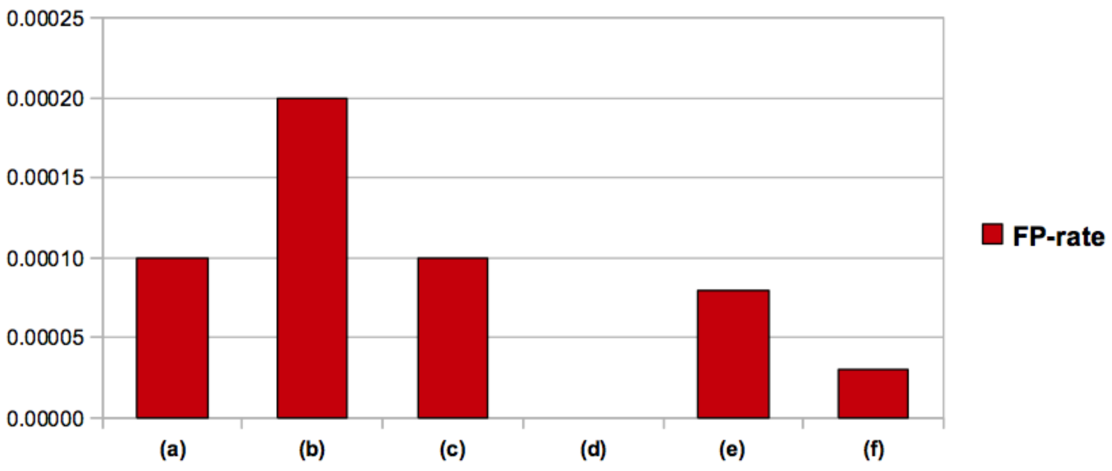


Figure C.8: Comparison of *FP-rates* from experiments adding distal and randomised negative examples before/after making training data consistent (mouse data)

data consistent shows some false positives, which are also almost negligible. The results from the yeast data are not as consistent as that of the mouse data set. It has been already mentioned that this may be due the nature of the data set as it is not updated and may not properly annotated.

Appendix D

Visualisations

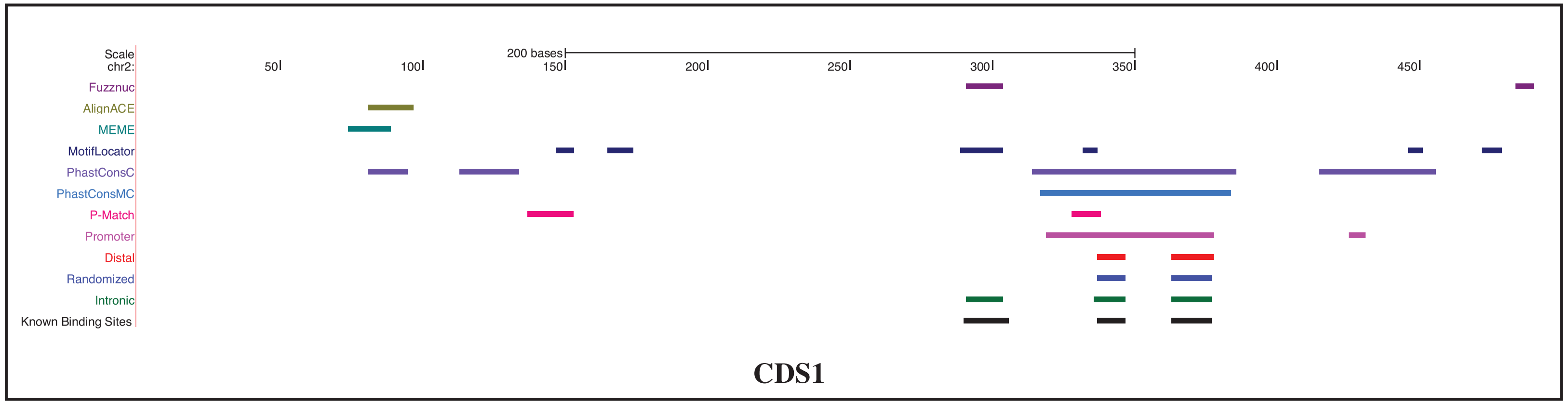


Figure D.1: Visualisation of computational prediction results on the gene CDS1 (*Yeast_p0.001_stringent* dataset)

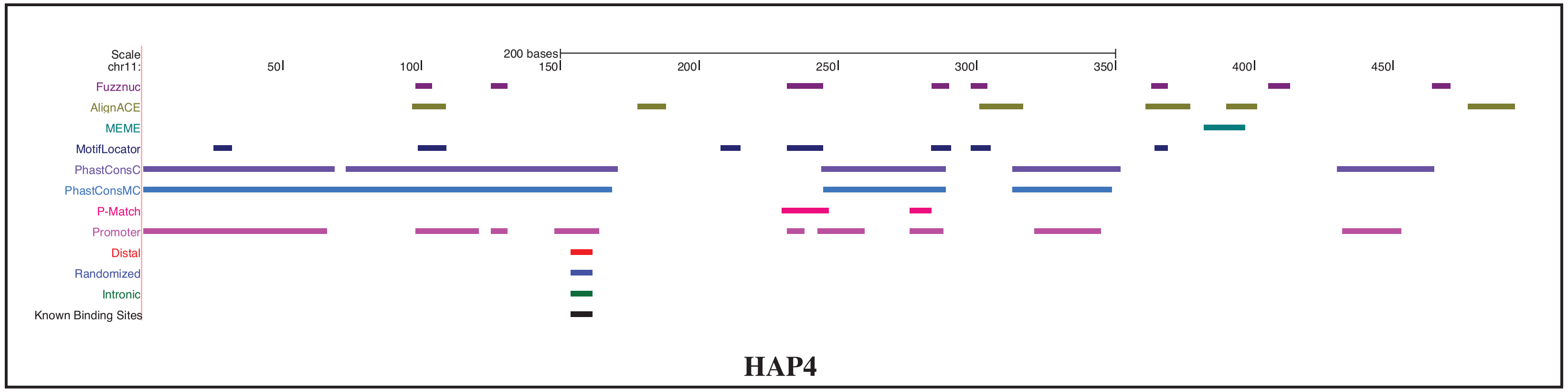


Figure D.2: Visualisation of computational prediction results on the gene HAP4 (*Yeast_p0.001_stringent* dataset)

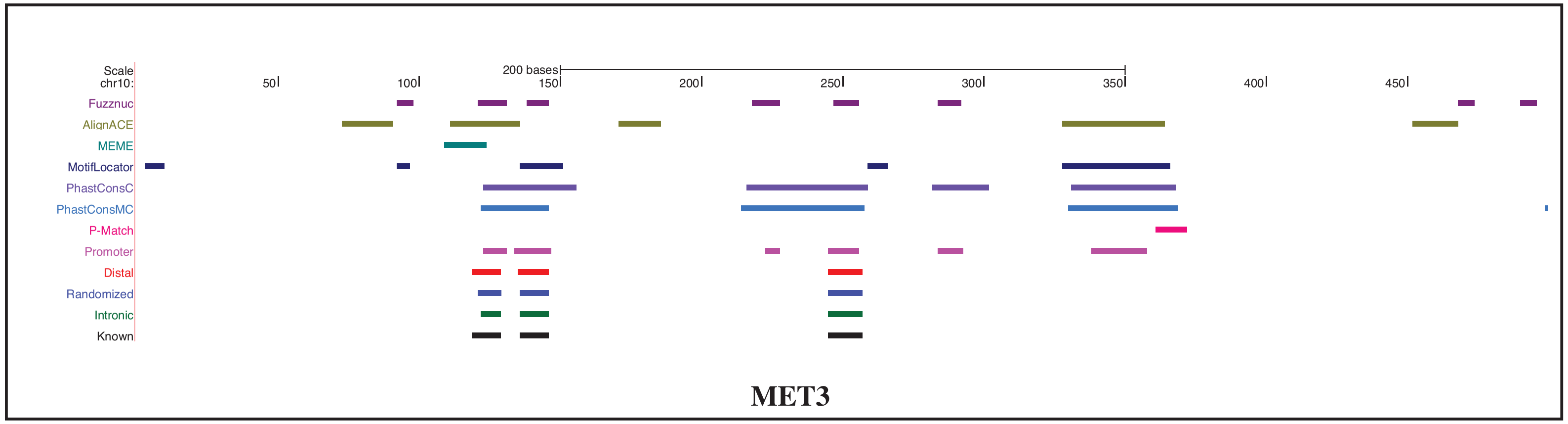


Figure D.3: Visualisation of computational prediction results on the gene MET3 (*Yeast_p0.001_stringent* dataset)

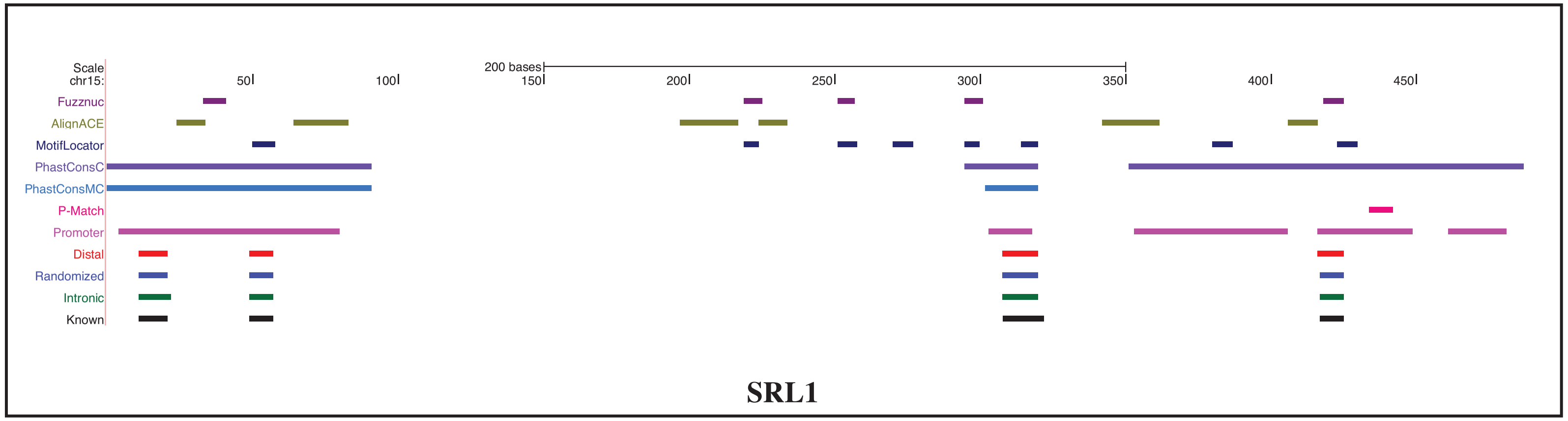


Figure D.4: Visualisation of computational prediction results on the gene SRL1 (*Yeast_p0.001_stringent* dataset)

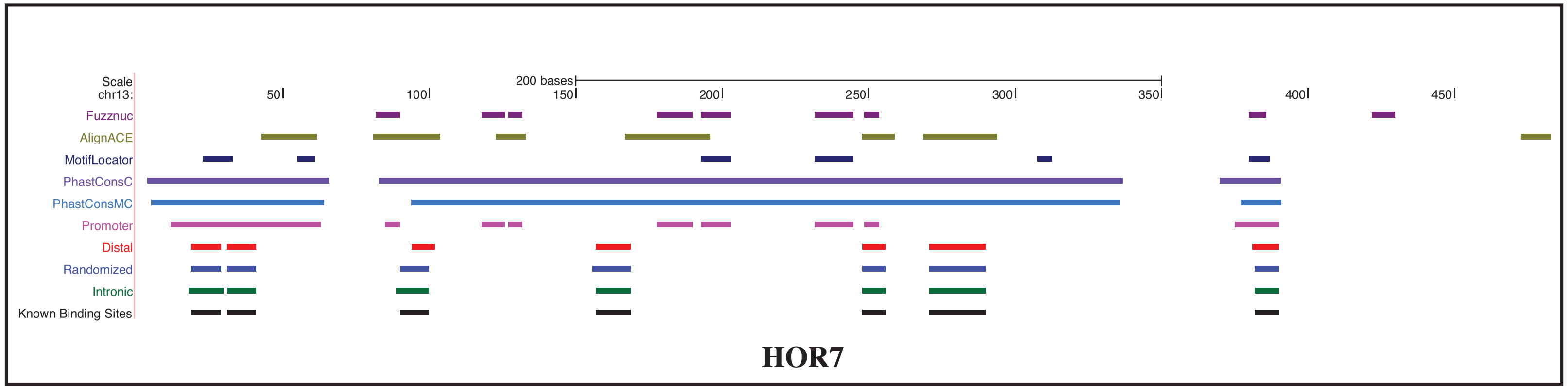


Figure D.5: Visualisation of computational prediction results on the gene HOR7 (*Yeast_p0.005_moderate* dataset)

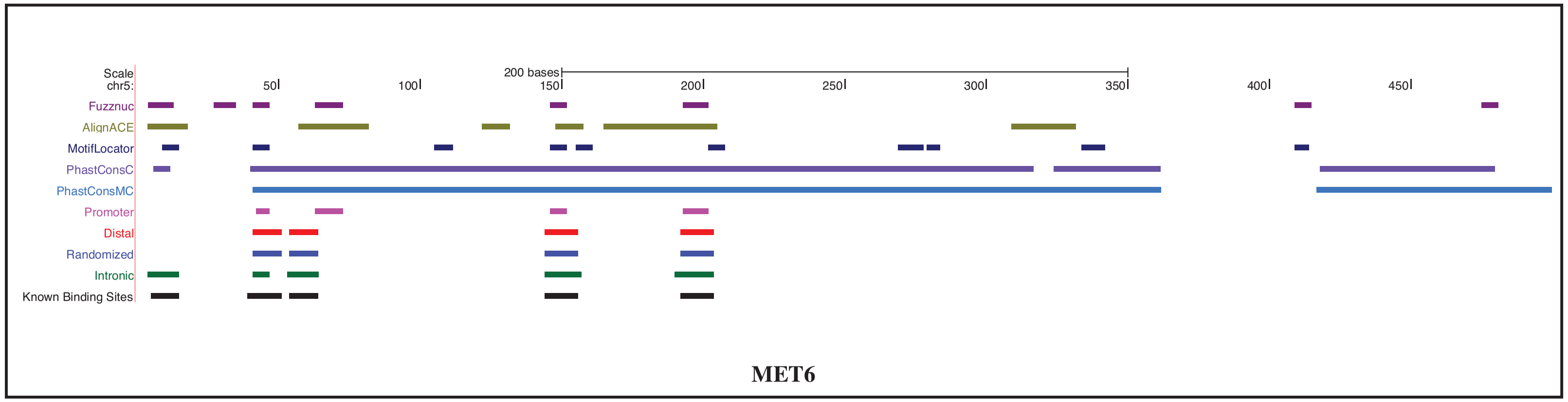


Figure D.6: Visualisation of computational prediction results on the gene MET6 (*Yeast_p0.005_moderate* dataset)

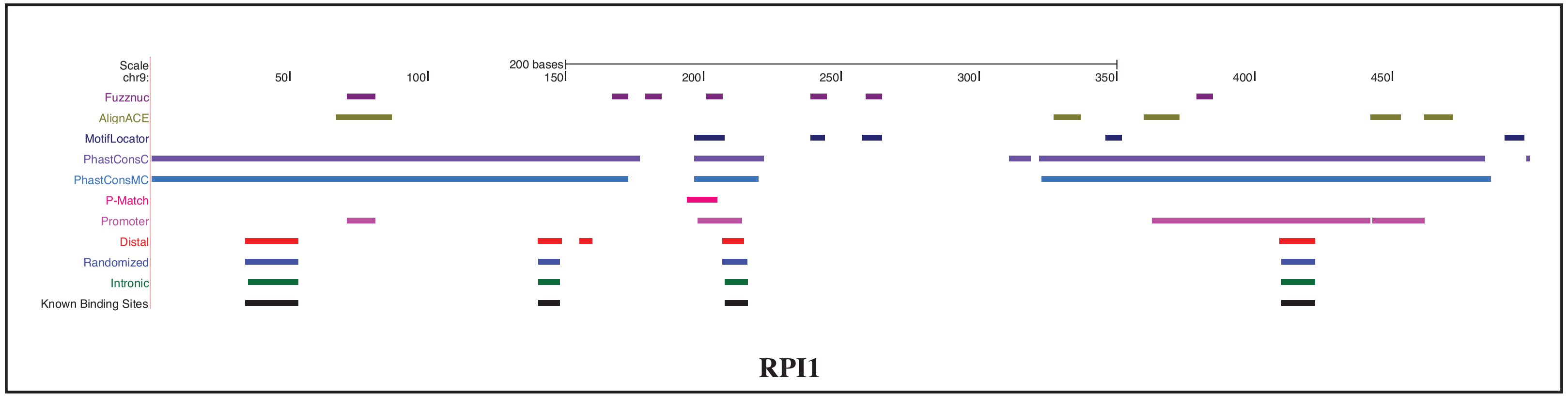


Figure D.7: Visualisation of computational prediction results on the gene RPI1 (*Yeast_p0.005_moderate* dataset)

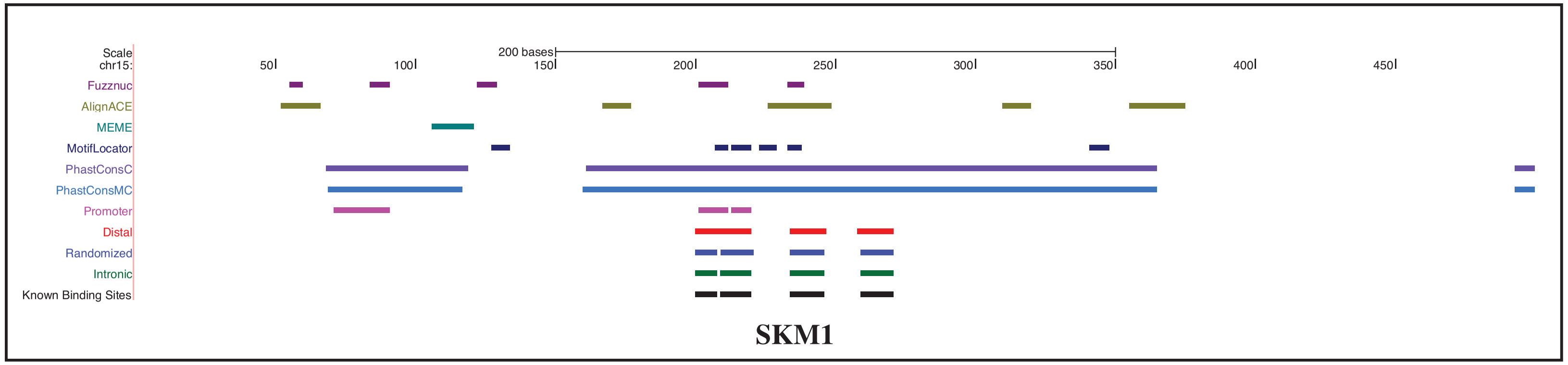


Figure D.8: Visualisation of computational prediction results on the gene SKM1 (*Yeast_p0.005_moderate* dataset)

Appendix E

List of Publications

E.1 Journal

1. Mark Robinson, Cristina Gonzlez Castellano, Faisal Rezwan, Rod Adams, Neil Davey, Alistair Rust, Yi Sun (2008). Combining experts in order to identify binding sites in yeast and mouse genomic data. Neural Networks, 2008: 856-861.

E.2 Selected Conferences

1. Faisal Rezwan, Rod Adams, Neil Davey, Yi Sun, Alistair G. Rust, Mark Robinson (2011). Effect of Using Varying Negative Examples in Transcription Factor Binding Site Predictions: Proceedings of 9th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics (EvoBIO11). Torino, Italy.
   Nominated for the best paper award

2. Faisal Rezwan, Rod Adams, Neil Davey, Yi Sun, Alistair G. Rust, Mark Robinson. Using Randomised Vectors in Transcription Factor Binding Site Predictions: Proceedings of IEEE The Ninth International Conference on Machine

Learning and Applications (ICMLA 2010). Hyatt Regency Bethesda, Washington DC, USA.

3. Faisal Rezwan, Rod Adams, Neil Davey, Yi Sun, Alistair Rust, Mark Robinson (2008). Combining experts in order to identify binding sites in genomic data: Proceedings of the 2008 UK Workshop on Computational Intelligence (UKCI 2008). De Montfort University, Leicester, UK.

References

Abnizova, I., Rust, A. G., Robinson, M., Te Boekhorst, R., & Gilks, W. R. 2006. Transcription binding site prediction using Markov models. *J Bioinform Comput Biol*, **4**(Apr), 425–441. 76

Ahmad, S., & Sarai, A. 2005. PSSM-based prediction of DNA binding sites in proteins. *BMC Bioinformatics*, **6**, 33. 48

Alashwal, H. T., Deris, S., & Othman, R. M. 2006. One-Class Support Vector Machines for Protein-Protein Interactions Prediction. *Int'l Journal Biomedical Sciences*, **1**(2), 120–127. 56

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., & Watson, J. D. 1994. *Molecular Biology of the Cell, 3rd edition*. New York: Garland Science. 2, 11, 16

Aparicio, O., Geisberg, J. V., & Struhl, K. 2004. Chromatin immunoprecipitation for determining the association of proteins with specific genomic sequences in vivo. *Curr Protoc Cell Biol*, **Chapter 17**(Sep), Unit 17.7. 21

Apostolico, A., Bock, M. E., Lonardi, S., & Xu, X. 2000. Efficient detection of unusual words. *J. Comput. Biol.*, **7**, 71–94. 76

Apostolico, A., Gong, F., & Lonardi, S. 2004. Verbumculus and the Discovery of Unusual Words. *J. Comput. Sci. Technol.*, 22–41. 40, 41, 76

Arnone, M. I., & Davidson, E. H. 1997. The hardwiring of development: organization and function of genomic regulatory systems. *Development*, **124**(May), 1851–1864. 1, 3 authors listed, No. 2001. IUPAC-IUB joint commission on biochemical nomenclature abbreviations and symbols for the description of conformations of polynucleotide chains. *Curr Protoc Nucleic Acid Chem*, **Appendix 1**(May), Appendix 1C. 29

Bailey, T. L., & Elkan, C. 1994. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc Int Conf Intell Syst Mol Biol*, **2**, 28–36. 43

Bailey, T. L., & Elkan, C. 1995. The value of prior knowledge in discovering motifs with MEME. *Proc Int Conf Intell Syst Mol Biol*, **3**, 21–29. 43, 44, 76

Batchelor, Bruce G. 1978. *Pattern Recognition: Ideas in Practice*. Plenum Pub Corp. 63

Beiko, R. G., & Charlebois, R. L. 2005. GANN: genetic algorithm neural networks for the detection of conserved combinations of features in DNA. *BMC Bioinformatics*, **6**, 36. 48

Bell, A. C., & Felsenfeld, G. 2000. Methylation of a CTCF-dependent boundary controls imprinted expression of the Igf2 gene. *Nature*, **405**(May), 482–485. 44

Ben-Gal, I., Shani, A., Gohr, A., Grau, J., Arviv, S., Shmilovici, A., Posch, S., & Grosse, I. 2005. Identification of transcription factor binding sites with variable-order Bayesian networks. *Bioinformatics*, **21**(Jun), 2657–2666. 30, 36

Ben-Hur, A., Ong, C. S., Sonnenburg, S., Scholkopf, B., & Ratsch, G. 2008. Support vector machines and kernels for computational biology. *PLoS Comput. Biol.*, **4**(Oct), e1000173. 52, 54, 55, 56

Berg, J., Tymoczko, J. L., & Stryer, L. 2007. *Biochemistry*. New York: WH Freeman and Co. 12

Berman, B. P., Nibu, Y., Pfeiffer, B. D., Tomancak, P., Celniker, S. E., Levine, M., Rubin, G. M., & Eisen, M. B. 2002. Exploiting transcription factor binding site clustering to identify cis-regulatory modules involved in pattern formation in the Drosophila genome. *Proc. Natl. Acad. Sci. U.S.A.*, **99**(Jan), 757–762. 40

Biberman, Y. 1994. A context similarity measure. *Pages 49–63 of: Proceedings of the European conference on machine learning on Machine Learning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc. 63

Blanchette, M., & Tompa, M. 2003. FootPrinter: A program designed for phylogenetic footprinting. *Nucleic Acids Res.*, **31**(Jul), 3840–3842. 77

Blanchette, M., Schwikowski, B., & Tompa, M. 2002. Algorithms for phylogenetic footprinting. *J. Comput. Biol.*, **9**, 211–223. 28, 39

Blanco, E., Farre, D., Alba, M. M., Messeguer, X., & Guigo, R. 2006. ABS: a database of Annotated regulatory Binding Sites from orthologous promoters. *Nucleic Acids Res.*, **34**(Jan), D63–67. 73

Boron, Walter F. 2005. *Medical Physiology: A Cellular And Molecular Approach*. Elsevier/Saunders. 13

Boser, B. E., Guyon, I., & Vapnik, V. 1992. A Training Algorithm for Optimal Margin Classifiers. *Pages 144–152 of: COLT*. 49

Boyd, S., & Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press. 54

Bray, N., & Pachter, L. 2003. MAVID multiple alignment server. *Nucleic Acids Res.*, **31**(Jul), 3525–3526. 45

Brazma, A., Jonassen, I., Eidhammer, I., & Gilbert, D. 1998a. Approaches to the automatic discovery of patterns in biosequences. *J. Comput. Biol.*, **5**, 279–305. 40

Brazma, A., Jonassen, I., Vilo, J., & Ukkonen, E. 1998b. Predicting gene regulatory elements in silico on a genomic scale. *Genome Res.*, **8**(Nov), 1202–1215. 40

Brivanlou, A. H., & Darnell, J. E. 2002. Signal transduction and the control of gene expression. *Science*, **295(5556)**, 813–818. 13

Brown, C. T., Rust, A. G., Clarke, P. J., Pan, Z., Schilstra, M. J., De Buysscher, T., Griffin, G., Wold, B. J., Cameron, R. A., Davidson, E. H., & Bolouri, H. 2002. New computational approaches for analysis of cis-regulatory networks. *Dev. Biol.*, **246**(Jun), 86–102. 4, 77

Buck, M. J., & Lieb, J. D. 2004. ChIP-chip: considerations for the design, analysis, and application of genome-wide chromatin immunoprecipitation experiments. *Genomics*, **83**(Mar), 349–360. 4, 22

Bussemaker, H. J., Li, H., & Siggia, E. D. 2001. Regulatory element detection using correlation with expression. *Nat. Genet.*, **27**(Feb), 167–171. 41, 42

Carrillo, H., & Lipman, D. J. 1988. The Multiple Sequence Alignment Problem in Biology. *SIAM Journal of Applied Mathematics*, **48 (5)**, 1073–1082. 19

Carroll, S. B., Grenier, J. K., & Weatherbee, S. D. 2005. *From DNA to diversity:molecular genetics and the evolution of animal design.* Oxford, UK: Wiley-Blackwell. 10

Cawley, S., Bekiranov, S., Ng, H. H., Kapranov, P., Sekinger, E. A., Kampa, D., Piccolboni, A., Sementchenko, V., Cheng, J., Williams, A. J., Wheeler, R., Wong, B., Drenkow, J., Yamanaka, M., Patel, S., Brubaker, S., Tammana, H., Helt, G., Struhl, K., & Gingeras, T. R. 2004. Unbiased mapping of transcription factor binding sites along human chromosomes 21 and 22 points to widespread regulation of noncoding RNAs. *Cell*, **116**(Feb), 499–509. 4

Cereghini, S., Saragosti, S., Yaniv, M., & Hamer, D. H. 1984. SV40-alpha-globulin hybrid minichromosomes. Differences in DNase I hypersensitivity of promoter and enhancer sequences. *Eur. J. Biochem.*, **144**(Nov), 545–553. 21

Chang, Chih-Chung, & Lin, Chih-Jen. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, **2**, 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 58, 61

Chawla, Nitesh V., Bowyer, Kevin W., Hall, Lawrence O., & Kegelmeyer, W. Philip. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res. (JAIR)*, **16**, 321–357. 63, 65, 96

Chawla, Nitesh V., Lazarevic, Aleksandar, Hall, Lawrence O., & Bowyer, Kevin W. 2003. SMOTEBoost: Improving Prediction of the Minority Class in Boosting. *Pages 107–119 of:* Lavrac, Nada, Gamberger, Dragan, Blockeel, Hendrik, & Todorovski, Ljupco (eds), *PKDD*. Lecture Notes in Computer Science, vol. 2838. Springer. 63

Che, D., Jensen, S., Cai, L., & Liu, J. S. 2005. BEST: binding-site estimation suite of tools. *Bioinformatics*, **21**(Jun), 2909–2911. 46, 87

Chekmenev, D. S., Haid, C., & Kel, A. E. 2005. P-Match: transcription factor binding site search by combining patterns and weight matrices. *Nucleic Acids Res.*, **33**(Jul), W432–437. 153

Chen, W. H., Wei, W., & Lercher, M. J. 2011. Minimal regulatory spaces in yeast genomes. *BMC Genomics*, **12**, 320. 151

Chenna, R., Sugawara, H., Koike, T., Lopez, R., Gibson, T. J., Higgins, D. G., & Thompson, J. D. 2003. Multiple sequence alignment with the Clustal series of programs. *Nucleic Acids Res.*, **31**(Jul), 3497–3500. 19

Cora, D., Di Cunto, F., Provero, P., Silengo, L., & Caselle, M. 2004. Computational identification of transcription factor binding sites by functional analysis of sets of genes sharing overrepresented upstream motifs. *BMC Bioinformatics*, **5**(May), 57. 41

Cosgrove, M. S., Boeke, J. D., & Wolberger, C. 2004. Regulated nucleosome mobility and the histone code. *Nat. Struct. Mol. Biol.*, **11**(Nov), 1037–1043. 20

Crick, F. H. 1962. The genetic code. *Sci. Am.*, **207**(Oct), 66–74. 1

Cristianini, Nello, & Shawe-Taylor, John. 2010. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press. 49

Crooks, G. E., Hon, G., Chandonia, J. M., & Brenner, S. E. 2004. WebLogo: a sequence logo generator. *Genome Res.*, **14**(Jun), 1188–1190. 35

Das, M. K., & Dai, H. K. 2007. A survey of DNA motif finding algorithms. *BMC Bioinformatics*, **8 Suppl 7**, S21. 39

Davidson, E. H. 1999. A view from the genome: spatial control of transcription in sea urchin development. *Curr. Opin. Genet. Dev.*, **9**(Oct), 530–541. 3

Davidson, Eric H. 2001. *Genomic Regulatory Systems: Development and Evolution*. Academic Press. 1, 3

Diday, E. 1974. Recent Progress in Distance and Similarity Measures in Pattern Recognition. *Pages 534–539 of: Pattern Recognition*. 63

Dietterich, T. G. 2002. Machine Learning for Sequential Data: A Review. *Pages 15–30 of: Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*. London, UK, UK: Springer-Verlag. 48

Dietterich, Thomas G. 2000. Ensemble Methods in Machine Learning. *Pages 1–15 of:* Kittler, Josef, & Roli, Fabio (eds), *Multiple Classifier Systems*. Lecture Notes in Computer Science, vol. 1857. Springer. 88

Ding, C. H., & Dubchak, I. 2001. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, **17**(Apr), 349–358. 49

Domingos, P. 1995. Rule induction and instance-based learning a unified approach. *Pages 1226–1232 of: Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. 63

Dougherty, E. R., Barrera, J., Brun, M., Kim, S., Cesar, R. M., Chen, Y., Bittner, M., & Trent, J. M. 2002. Inference from clustering with application to gene-expression microarrays. *J. Comput. Biol.*, **9**, 105–126. 42

Durbin, R., Eddy, S. R., Krogh, A., & Mitchison, G. 1999. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press. 36, 38, 42, 43, 45

Eddy, S. R. 1996. Hidden Markov models. *Curr. Opin. Struct. Biol.*, **6**(Jun), 361–365. 37, 38

Eddy, S. R. 1998. Profile hidden Markov models. *Bioinformatics*, **14**, 755–763. 38

Eddy, S. R. 2004. What is dynamic programming? *Nat. Biotechnol.*, **22**(Jul), 909–910. 38, 45

Elnitski, L., Jin, V. X., Farnham, P. J., & Jones, S. J. 2006. Locating mammalian transcription factor binding sites: a survey of computational and experimental techniques. *Genome Res.*, **16**(Dec), 1455–1464. 21, 22, 39

Ettwiller, L., Paten, B., Souren, M., Loosli, F., Wittbrodt, J., & Birney, E. 2005. The discovery, positioning and verification of a set of transcription-associated motifs in vertebrates. *Genome Biol.*, **6**, R104. 78

Fickett, J. W., & Wasserman, W. W. 2000. Discovery and modeling of transcriptional regulatory regions. *Curr. Opin. Biotechnol.*, **11**(Feb), 19–24. 44, 45

Fried, M., & Crothers, D. M. 1981. Equilibria and kinetics of lac repressor-operator interactions by polyacrylamide gel electrophoresis. *Nucleic Acids Res.*, **9**(Dec), 6505–6525. 21

Frith, M. C., Fu, Y., Yu, L., Chen, J. F., Hansen, U., & Weng, Z. 2004. Detection of functional DNA motifs via statistical over-representation. *Nucleic Acids Res.*, **32**, 1372–1381. 40

Fullwood, M. J., Wei, C. L., Liu, E. T., & Ruan, Y. 2009. Next-generation DNA sequencing of paired-end tags (PET) for transcriptome and genome analyses. *Genome Res.*, **19**(Apr), 521–532. 22

Furey, T. S., Cristianini, N., Duffy, N., Bednarski, D. W., Schummer, M., & Haussler, D. 2000. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, **16**(Oct), 906–914. 48

Galas, D. J., & Schmitz, A. 1978. DNAse footprinting: a simple method for the detection of protein-DNA binding specificity. *Nucleic Acids Res.*, **5**(Sep), 3157–3170. 21

Galas, D. J., Eggert, M., & Waterman, M. S. 1985. Rigorous pattern-recognition methods for DNA sequences. Analysis of promoter sequences from Escherichia coli. *J. Mol. Biol.*, **186**(Nov), 117–128. 40

Gardiner-Garden, M., & Frommer, M. 1987. CpG islands in vertebrate genomes. *J. Mol. Biol.*, **196**(Jul), 261–282. 79

Garner, M. M., & Revzin, A. 1981. A gel electrophoresis method for quantifying the binding of proteins to specific DNA regions: application to components of the Escherichia coli lactose operon regulatory system. *Nucleic Acids Res.*, **9**(Jul), 3047–3060. 21

Goldberg, M.L. 1979. *Sequence Analysis of Drosophila Histone Genes*. Ph.D. thesis, Stanford University. 15

Gross, D. S., & Garrard, W. T. 1988. Nuclease hypersensitive sites in chromatin. *Annu. Rev. Biochem.*, **57**, 159–197. 21

Guyon, Isabelle, Weston, Jason, Barnhill, Stephen, & Vapnik, Vladimir. 2002. Gene Selection for Cancer Classification using Support Vector Machines. *Mach. Learn.*, **46**(March), 389–422. 48

Hampson, S., Baldi, P., Kibler, D., & Sandmeyer, S. B. 2000. Analysis of yeast's ORF upstream regions by parallel processing, microarrays, and computational methods. *Proc Int Conf Intell Syst Mol Biol*, **8**, 190–201. 41

Hampson, S., Kibler, D., & Baldi, P. 2002. Distribution patterns of over-represented k-mers in non-coding yeast DNA. *Bioinformatics*, **18**(Apr), 513–528. 41

Hertz, G. Z., & Stormo, G. D. 1999. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, **15**, 563–577. 31, 47

Heumann, J. M., Lapedes, A. S., & Stormo, G. D. 1994. Neural networks for determining protein specificity and multiple alignment of binding sites. *Proc Int Conf Intell Syst Mol Biol*, **2**, 188–194. 35

Holloway, D. T., Kon, M., & DeLisi, C. 2005. Integrating genomic data to predict transcription factor binding. *Genome Inform*, **16**, 83–94. 48

Hu, J., Li, B., & Kihara, D. 2005. Limitations and potentials of current motif discovery algorithms. *Nucleic Acids Res.*, **33**, 4899–4913. 5, 39

Hua, S., & Sun, Z. 2001a. A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach. *J. Mol. Biol.*, **308**(Apr), 397–407. 49

Hua, S., & Sun, Z. 2001b. Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, **17**(Aug), 721–728. 49

Huber, B. R., & Bulyk, M. L. 2006. Meta-analysis discovery of tissue-specific DNA sequence motifs from mammalian gene expression data. *BMC Bioinformatics*, **7**, 229. 46, 87

Hughes, J. D., Estep, P. W., Tavazoie, S., & Church, G. M. 2000. Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. *J. Mol. Biol.*, **296**(Mar), 1205–1214. 76

Jaakkola, T., Diekhans, M., & Haussler, D. 2000. A discriminative framework for detecting remote protein homologies. *J. Comput. Biol.*, **7**, 95–114. 49

Jacob, F., & Monod, J. 1959. [Genes of structure and genes of regulation in the biosynthesis of proteins]. *C. R. Hebd. Seances Acad. Sci.*, **249**(Oct), 1282–1284. 71

Japkowicz, Nathalie. 2003. Class imbalances: are we focusing on the right issue. *Workshop on Learning from Imbalanced Data Sets II*, 1723. 63

Jareborg, N., Birney, E., & Durbin, R. 1999. Comparative analysis of noncoding regions of 77 orthologous mouse and human gene pairs. *Genome Res.*, **9**(Sep), 815–824. 45

Jensen, S. T., & Liu, J. S. 2004. BioOptimizer: a Bayesian scoring function approach to motif discovery. *Bioinformatics*, **20**(Jul), 1557–1564. 47, 48

Jiang, B., Zhang, M. Q., & Zhang, X. 2007. OSCAR: one-class SVM for accurate recognition of cis-elements. *Bioinformatics*, **23**(Nov), 2823–2828. 48

Jothi, R., Cuddapah, S., Barski, A., Cui, K., & Zhao, K. 2008. Genome-wide identification of in vivo protein-DNA binding sites from ChIP-Seq data. *Nucleic Acids Res.*, **36**(Sep), 5221–5231. 22

Karolchik, D., Baertsch, R., Diekhans, M., Furey, T. S., Hinrichs, A., Lu, Y. T., Roskin, K. M., Schwartz, M., Sugnet, C. W., Thomas, D. J., Weber, R. J., Haussler, D., & Kent, W. J. 2003. The UCSC Genome Browser Database. *Nucleic Acids Res.*, **31**(Jan), 51–54. 77

Kel, A. E., Gössling, E., Reuter, I., Cheremushkin, E., Kel-Margoulis, O. V., & Wingender, E. 2003. MATCH: A tool for searching transcription factor binding sites in DNA sequences. *Nucleic Acids Res.*, **31**(Jul), 3576–3579. 39

Kiełbasa, S. M., Korbel, J. O., Beule, D., Schuchhardt, J., & Herzel, H. 2001. Combining frequency and positional information to predict transcription factor binding sites. *Bioinformatics*, **17**(Nov), 1019–1026. 41

Kochanski, G. 2004. *Markov Models, Hidden and Otherwise.* 37

Kolbe, D., Taylor, J., Elnitski, L., Eswara, P., Li, J., Miller, W., Hardison, R., & Chiaromonte, F. 2004. Regulatory potential scores from genome-wide three-way alignments of human, mouse, and rat. *Genome Res.*, **14**(Apr), 700–707. 78

Larkin, M. A., Blackshields, G., Brown, N. P., Chenna, R., McGettigan, P. A., McWilliam, H., Valentin, F., Wallace, I. M., Wilm, A., Lopez, R., Thompson, J. D., Gibson, T. J., & Higgins, D. G. 2007. Clustal W and Clustal X version 2.0. *Bioinformatics*, **23**(Nov), 2947–2948. 19

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., & Wootton, J. C. 1993. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, **262**(Oct), 208–214. 42, 77

Leslie, C., Eskin, E., & Noble, W. S. 2002. The spectrum kernel: a string kernel for SVM protein classification. *Pac Symp Biocomput*, 564–575. 48

Leslie, C. S., Eskin, E., Cohen, A., Weston, J., & Noble, W. S. 2004. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, **20**(Mar), 467–476. 48

Levine, M., & Tjian, R. 2003. Transcription regulation and animal diversity. *Nature*, **424**(Jul), 147–151. 21

Lieb, J. D., Liu, X., Botstein, D., & Brown, P. O. 2001. Promoter-specific binding of Rap1 revealed by genome-wide maps of protein-DNA association. *Nat. Genet.*, **28**(Aug), 327–334. 4, 22

Lifton, R. P., Goldberg, M. L., Karp, R. W., & Hogness, D. S. 1978. The organization of the histone genes in Drosophila melanogaster: functional and evolutionary implications. *Cold Spring Harb. Symp. Quant. Biol.*, **42 Pt 2**, 1047–1051. 15

Liu, X., Brutlag, D. L., & Liu, J. S. 2001. BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. *Pac Symp Biocomput*, 127–138. 47

Liu, X. S., Brutlag, D. L., & Liu, J. S. 2002. An algorithm for finding protein-DNA binding sites with applications to chromatin-immunoprecipitation microarray experiments. *Nat. Biotechnol.*, **20**(Aug), 835–839. 47

MacIsaac, Kenzie D., Wang, Ting, Gordon, D. Benjamin, Gifford, David K., Stormo, Gary D., & Fraenkel, Ernest. 2006. An improved map of conserved regulatory sites for *Saccharomyces cerevisiae*. *BMC Bioinformatics*, **7**, 113. 149, 150

Manevitz, L. M., & Yousef, M. 2002. One-class svms for document classification. *J. Mach. Learn. Res.*, **2**(March), 139–154. 58

Markstein, M., Markstein, P., Markstein, V., & Levine, M. S. 2002. Genome-wide analysis of clustered Dorsal binding sites identifies putative target genes in the Drosophila embryo. *Proc. Natl. Acad. Sci. U.S.A.*, **99**(Jan), 763–768. 1, 41

Marsan, L., & Sagot, M. F. 2000. Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. *J. Comput. Biol.*, **7**, 345–362. 40

Matys, V., Fricke, E., Geffers, R., Gossling, E., Haubrock, M., Hehl, R., Hornischer, K., Karas, D., Kel, A. E., Kel-Margoulis, O. V., Kloos, D. U., Land, S., Lewicki-Potapov, B., Michael, H., Munch, R., Reuter, I., Rotert, S., Saxel, H., Scheer, M., Thiele, S., & Wingender, E. 2003. TRANSFAC: transcriptional regulation, from patterns to profiles. *Nucleic Acids Res.*, **31**(Jan), 374–378. 33

McCue, L., Thompson, W., Carmack, C., Ryan, M. P., Liu, J. S., Derbyshire, V., & Lawrence, C. E. 2001. Phylogenetic footprinting of transcription factor binding sites in proteobacterial genomes. *Nucleic Acids Res.*, **29**(Feb), 774–782. 45

Michalski, R.S., Stepp, R. E., & Diday, E. 1981. A Recent Advance in Data Analysis: Clustering Objects into Classes Characterized by Conjunctive Concepts. *Progress in Pattern Recognition*, **1**, 33–56. 63

Montgomery, S. B., Griffith, O. L., Sleumer, M. C., Bergman, C. M., Bilenky, M., Pleasance, E. D., Prychyna, Y., Zhang, X., & Jones, S. J. 2006. ORegAnno: an open access database and curation system for literature-derived promoters, transcription factor binding sites and regulatory variation. *Bioinformatics*, **22**(Mar), 637–640. 17, 72, 73

Mount, David W. 2004. *Bioinformatics: Sequence and Genome Analysis, Second Edition*. 2nd edn. Cold Spring Harbor Laboratory Press. 19

Nadler, Morton, & Smith, Eric P. 1993. *Pattern Recognition Engineering*. Wiley-Interscience. 63

Needleman, S. B., & Wunsch, C. D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, **48**(Mar), 443–453. 19

Neuwald, A. F., Liu, J. S., & Lawrence, C. E. 1995. Gibbs motif sampling: detection of bacterial outer membrane protein repeats. *Protein Sci.*, **4**(Aug), 1618–1632. 42

Ng, H. H., & Bird, A. 1999. DNA methylation and chromatin modification. *Curr. Opin. Genet. Dev.*, **9**(Apr), 158–163. 20

Nguyen, T. T., & Androulakis, I. P. 2009. Recent Advances in the Computational Discovery of Transcription Factor Binding Sites. *Algorithms*, **2(1)**, 582–605. 39

Orphanides, G., Lagrange, T., & Reinberg, D. 1996. The general transcription factors of RNA polymerase II. *Genes Dev.*, **10**(Nov), 2657–2683. 13

Papatsenko, D. A., Makeev, V. J., Lifanov, A. P., Regnier, M., Nazina, A. G., & Desplan, C. 2002. Extraction of functional binding sites from unique regulatory regions: the Drosophila early developmental enhancers. *Genome Res.*, **12**(Mar), 470–481. 40, 41

Pavesi, G., Mauri, G., & Pesole, G. 2001. An algorithm for finding signals of unknown length in DNA sequences. *Bioinformatics*, **17 Suppl 1**, S207–214. 47

Pavesi, G., Mauri, G., & Pesole, G. 2004. In silico representation and discovery of transcription factor binding sites. *Brief. Bioinformatics*, **5**(Sep), 217–236. 39

Pavlidis, P, Cai, J, Weston, J, & Grundy, W. N. 2001. Gene functional classification from heterogeneous data. *In: In Proceedings of the Fifth Annual International Conference on Computational Biology: April 22-25, 2001.* 49

Pollock, R., & Treisman, R. 1990. A sensitive method for the determination of protein-DNA binding specificities. *Nucleic Acids Res.*, **18**(Nov), 6197–6204. 4

Portales-Casamar, E., Arenillas, D., Lim, J., Swanson, M. I., Jiang, S., McCallum, A., Kirov, S., & Wasserman, W. W. 2009. The PAZAR database of gene regulatory information coupled to the ORCA toolkit for the study of regulatory sequences. *Nucleic Acids Res.*, **37**(Jan), 54–60. 33

Ptashne, M., & Gann, A. 2002. *Genes and Signals*. New York: Cold Spring Harbour Laboratory Press. 3, 14, 41, 71

Qian, J., Lin, J., Luscombe, N. M., Yu, H., & Gerstein, M. 2003. Prediction of regulatory networks: genome-wide identification of transcription factor targets from gene expression data. *Bioinformatics*, **19**(Oct), 1917–1926. 48

Quandt, K., Frech, K., Karas, H., Wingender, E., & Werner, T. 1995. MatInd and MatInspector: new fast and versatile tools for detection of consensus matches in nucleotide sequence data. *Nucleic Acids Res.*, **23**(Dec), 4878–4884. 39

Radivojac, P., Chawla, N. V., Dunker, A. K., & Obradovic, Z. 2004. Classification and knowledge discovery in protein databases. *J Biomed Inform*, **37**(Aug), 224–239. 48, 63, 96

Rajewsky, N., Vergassola, M., Gaul, U., & Siggia, E. D. 2002. Computational detection of genomic cis-regulatory modules applied to body patterning in the early Drosophila embryo. *BMC Bioinformatics*, **3**(Oct), 30. 39, 75

Razin, A. 1998. CpG methylation, chromatin structure and gene silencing-a three-way connection. *EMBO J.*, **17**(Sep), 4905–4908. 20

Reiss, D. J., & Schwikowski, B. 2004. Predicting protein-peptide interactions via a network-based motif sampler. *Bioinformatics*, **20 Suppl 1**(Aug), i274–282. 76

Ren, B., & Dynlacht, B. D. 2004. Use of chromatin immunoprecipitation assays in genome-wide location analysis of mammalian transcription factors. *Meth. Enzymol.*, **376**, 304–315. 4

Ren, B., Robert, F., Wyrick, J. J., Aparicio, O., Jennings, E. G., Simon, I., Zeitlinger, J., Schreiber, J., Hannett, N., Kanin, E., Volkert, T. L., Wilson, C. J., Bell, S. P., & Young, R. A. 2000. Genome-wide location and function of DNA binding proteins. *Science*, **290**(Dec), 2306–2309. 4, 22

Robinson, M. 2006. *Automated de-novo Prediction of cis-regulatory Binding Sites in Putative Regulatory Sequences.* Ph.D. thesis, University of Hertfordshire, UK. 6, 74, 76, 80

Robinson, M., Sun, Y., te Boekhorst, R., Kaye, P., Adams, R., N.Davey, & Rust, A. G. 2006. Improving Computational Predictions of Cis- Regulatory Binding Sites. *Pages 391–402 of:* Altman, Russ B., Murray, Tiffany, Klein, Teri E., Dunker, A. Keith, & Hunter, Lawrence (eds), *Pacific Symposium on Biocomputing.* World Scientific. 5, 61, 63, 67, 71, 73, 88, 92, 100, 134, 147

Robinson, M., Castellano, C. G., Adams, R., Davey, N., & Sun, Y. 2007a. Identifying Binding Sites in Sequential Genomic Data. *Pages 100–109 of:* de Sá, Joaquim Marques, Alexandre, Luís A., Duch, Wlodzislaw, & Mandic, Danilo P. (eds), *ICANN (2).* Lecture Notes in Computer Science, vol. 4669. Springer. 5, 61, 63, 67, 71, 73, 88, 92, 100, 134, 147

Robinson, M., Sharabi, O., Sun, Y., Adams, R., te Boekhorst, R., Rust, A. G., & Davey, N. 2007b. Using Real-Valued Meta Classifiers to Integrate and Contextualize Binding Site Predictions. *Pages 822–829 of: Proceedings of the 8th international conference on Adaptive and Natural Computing Algorithms, Part I.* ICANNGA '07. Berlin, Heidelberg: Springer-Verlag. 5, 61, 63, 67, 71, 73, 88, 92, 100, 134, 147

Robinson, M., Castellano, C. G., Rezwan, F., Adams, R., Davey, N., Rust, A., & Sun, Y. 2008. Combining experts in order to identify binding sites in yeast and mouse genomic data. *Neural Networks*, **21**(6), 856–861. 5, 61, 63, 67, 71, 73, 88, 147

Romer, K. A., Kayombya, G. R., & Fraenkel, E. 2007. WebMOTIFS: automated discovery, filtering and scoring of DNA sequence motifs using multiple programs and Bayesian approaches. *Nucleic Acids Res.*, **35**(Jul), W217–220. 46, 87

Roth, F. P., Hughes, J. D., Estep, P. W., & Church, G. M. 1998. Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nat. Biotechnol.*, **16**(Oct), 939–945. 41, 76

Salzberg, S. 1991. A Nearest Hyperrectangle Learning Method. *Mach. Learn.*, **6**(May), 251–276. 63

Sandelin, A., & Wasserman, W. W. 2004. Constrained binding site diversity within families of transcription factors enhances pattern discovery bioinformatics. *J. Mol. Biol.*, **338**(Apr), 207–215. 41

Sandelin, A., Alkema, W., Engstrom, P., Wasserman, W. W., & Lenhard, B. 2004. JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Res.*, **32**(Jan), D91–94. 33, 78

Schneider, T. D. 1997. Information content of individual genetic sequences. *J. Theor. Biol.*, **189**(Dec), 427–441. 29

Schneider, T. D. 2002. Consensus sequence Zen. *Appl. Bioinformatics*, **1**, 111–119. 29

Schneider, T. D., & Stephens, R. M. 1990. Sequence logos: a new way to display consensus sequences. *Nucleic Acids Res.*, **18**(Oct), 6097–6100. 35

Schneider, T. D., Stormo, G. D., Gold, L., & Ehrenfeucht, A. 1986. Information content of binding sites on nucleotide sequences. *J. Mol. Biol.*, **188**(Apr), 415–431. 34, 35, 40

Schölkopf, Bernhard, Platt, John C., Shawe-Taylor, John, Smola, Alex J., & Williamson, Robert C. 2001. Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, **13**(7), 1443–1471. 57

Siepel, A., & Haussler, D. 2004. Combining phylogenetic and hidden Markov models in biosequence analysis. *J. Comput. Biol.*, **11**, 413–428. 38, 78

Siepel, A., Bejerano, G., Pedersen, J. S., Hinrichs, A. S., Hou, M., Rosenbloom, K., Clawson, H., Spieth, J., Hillier, L. W., Richards, S., Weinstock, G. M., Wilson, R. K., Gibbs, R. A., Kent, W. J., Miller, W., & Haussler, D. 2005. Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. *Genome Res.*, **15**(Aug), 1034–1050. 78

Sinha, S., & Tompa, M. 2002. Discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic Acids Res.*, **30**(Dec), 5549–5560. 40, 41

Sinha, S., Blanchette, M., & Tompa, M. 2004. PhyME: a probabilistic algorithm for finding motifs in sets of orthologous sequences. *BMC Bioinformatics*, **5**(Oct), 170. 45

Stanfill, C., & Waltz, D. 1986. Toward memory-based reasoning. *Commun. ACM*, **29**(December), 1213–1228. 64

Stegmaier, P., Kel, A. E., & Wingender, E. 2004. Systematic DNA-binding domain classification of transcription factors. *Genome Inform*, **15**, 276–286. 13

Stormo, G. D. 2000. DNA binding sites: representation and discovery. *Bioinformatics*, **16**(Jan), 16–23. 4, 29, 35

Stormo, G. D., & Fields, D. S. 1998. Specificity, free energy and information content in protein-DNA interactions. *Trends Biochem. Sci.*, **23**(Mar), 109–113. 34

Strahl, B. D., & Allis, C. D. 2000. The language of covalent histone modifications. *Nature*, **403**(Jan), 41–45. 20

Struhl, K. 1995. Yeast transcriptional regulatory mechanisms. *Annu. Rev. Genet.*, **29**, 651–674. 15

Sudo, Kyoko, Osawa, Tatsuya, Wakabayashi, Kaoru, Koike, Hideki, & Arakawa, Kenichi. 2008. Estimating Anomality of the Video Sequences for Surveillance Using 1-Class SVM. *IEICE - Trans. Inf. Syst.*, **E91-D**(July), 1929–1936. 57

Sun, Y., Robinson, M., Adams, R., Kaye, P., Rust, A. G., & Davey, N. 2005. Using Real-Valued Metaclassifiers To Integrate Binding Site Predictions. *Pages 481–486 of:* Yao, Yiyu, Shi, Zhongzhi, Wang, Yingxu, & Kinsner, Witold (eds), *IEEE IJCNN*. IEEE. 5, 61, 67, 71, 73, 88, 91, 92, 100, 134, 147

Sun, Y., Robinson, M., Adams, R., te Boekhorst, R., Rust, A. G., & Davey, N. 2006a. Using Feature Selection Filtering Methods for Binding Site Predictions. *Pages 566–571 of:* Yao, Yiyu, Shi, Zhongzhi, Wang, Yingxu, & Kinsner, Witold (eds), *IEEE ICCI*. IEEE. 5, 48, 61, 63, 67, 71, 73, 88, 92, 100, 134, 147

Sun, Y., Robinson, M., Adams, R., te Boekhorst, R., Rust, A. G., & Davey, N. 2006b. Using sampling methods to improve binding site predictions. *Pages 533–538 of: ESANN*. 5, 61, 63, 67, 71, 73, 88, 92, 100, 134, 147

Sun, Y., Robinson, M., Adams, R., Davey, N., & Rust, A. G. 2007. Predicting Binding Sites in the Mouse Genome. *Pages 476–481 of:* Wani, M. Arif, Kantardzic, Mehmed M., Li, Tao, Liu, Ying, Kurgan, Lukasz A., Ye, Jieping, Ogihara, Mitsunori, Sagiroglu, Seref, wen Chen, Xue, Peterson, Leif E., & Hafeez, Khalid (eds), *ICMLA*. IEEE Computer Society. 5, 61, 63, 67, 71, 73, 88, 147

Sun, Y., Robinson, M., Adams, R., Rust, A. G., & Davey, N. 2008. Prediction of Binding Sites in the Mouse Genome Using Support Vector Machines. *Pages 91–100 of:* Kurková, Vera, Neruda, Roman, & Koutník, Jan (eds), *ICANN (2)*. Lecture Notes in Computer Science, vol. 5164. Springer. 5, 61, 63, 67, 71, 73, 88, 147

Sun, Y., Robinson, M., Adams, R., te Boekhorst, R., Rust, A. G., & Davey, N. 2009a. Integrating genomic binding site predictions using real-valued meta classifiers. *Neural Comput. Appl.*, **18**(September), 577–590. 5, 61, 63, 67, 71, 73, 88, 91, 92, 100, 131, 134, 147

Sun, Y., Castellano, C. G., Robinson, M., Adams, R., Rust, A. G., & Davey, N. 2009b. Using pre & post-processing methods to improve binding site predictions. *Pattern Recogn.*, **42**(September), 1949–1958. 5, 61, 63, 67, 71, 73, 88, 91, 92, 100, 134, 147

Tax, D. M. J., & Duin, R. P. W. 2004. Support Vector Data Description. *Mach. Learn.*, **54**(January), 45–66. 57, 59

Taylor, J. D., Ackroyd, A. J., & Halford, S. E. 1994. The gel shift assay for the analysis of DNA-protein interactions. *Methods Mol. Biol.*, **30**, 263–279. 4

Thijs, G., Lescot, M., Marchal, K., Rombauts, S., De Moor, B., Rouze, P., & Moreau, Y. 2001. A higher-order background model improves the detection of promoter regulatory elements by Gibbs sampling. *Bioinformatics*, **17**(Dec), 1113–1122. 75, 78

Thijs, G., Marchal, K., Lescot, M., Rombauts, S., De Moor, B., Rouze, P., & Moreau, Y. 2002. A Gibbs sampling method to detect overrepresented motifs in the upstream regions of coexpressed genes. *J. Comput. Biol.*, **9**, 447–464. 75, 76, 78

Tjian, R. 1995. Molecular machines that control genes. *Sci. Am.*, **272**(Feb), 54–61. 15

Tompa, M., Li, N., Bailey, T. L., Church, G. M., De Moor, B., Eskin, E., Favorov, A. V., Frith, M. C., Fu, Y., Kent, W. J., Makeev, V. J., Mironov, A. A., Noble, W. S., Pavesi, G., Pesole, G., Regnier, M., Simonis, N., Sinha, S., Thijs, G., van Helden, J., Vandenbogaert, M., Weng, Z., Workman, C., Ye, C., & Zhu, Z. 2005. Assessing computational tools for the discovery of transcription factor binding sites. *Nat. Biotechnol.*, **23**(Jan), 137–144. 4, 5, 34, 39, 41, 72

Tversky, A. 1977. Features of Similarity. *Pages 327–352 of: Psychological Review*, vol. 84. 63

van Helden, J., Andre, B., & Collado-Vides, J. 1998. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J. Mol. Biol.*, **281**(Sep), 827–842. 40

Vert, Jean-Philippe, Thurman, Robert, & Noble, William Stafford. 2005. Kernels for gene regulatory regions. *In: NIPS*. 48

Wasserman, W. W., & Sandelin, A. 2004. Applied bioinformatics for the identification of regulatory elements. *Nat. Rev. Genet.*, **5**(Apr), 276–287. 44

Wasserman, W. W., Palumbo, M., Thompson, W., Fickett, J. W., & Lawrence, C. E. 2000. Human-mouse genome comparisons to locate regulatory sites. *Nat. Genet.*, **26**(Oct), 225–228. 45

Weber, H., Ziechmann, C., & Graessmann, A. 1990. In vitro DNA methylation inhibits gene expression in transgenic tobacco. *EMBO J.*, **9**(Dec), 4409–4415. 20

Wei, W., & Yu, X. D. 2007. Comparative analysis of regulatory motif discovery tools for transcription factor binding sites. *Genomics Proteomics Bioinformatics*, **5**(May), 131–142. 39

White, R. J. 2000. *Gene Transcription: Mechanism and Control.* Oxford, UK: Willey-Blackwell. 14

Wilczynski, B., Darzynkiewicz, M., & Tiuryn, J. 2008. MEMOFinder: combining de novo motif prediction methods with a database of known motifs. *Nature Precedings*, Dec, 1–6. 46, 87

Wilson, D. Randall, & Martinez, Tony R. 1997. Improved Heterogeneous Distance Functions. *Journal of Artificial Intelligence Research (JAIR)*, **1**, 1–34. 63

Wingender, Edgar. 2008. The TRANSFAC project as an example of framework technology that supports the analysis of genomic regulation. *Briefings in Bioinformatics*, **9**(4), 326–332. 85

Workman, C. T., & Stormo, G. D. 2000. ANN-Spec: a method for discovering transcription factor binding sites with improved specificity. *Pac Symp Biocomput*, 467–478. 48

Wray, G. A., Hahn, M. W., Abouheif, E., Balhoff, J. P., Pizer, M., Rockman, M. V., Romano, L. A., & Wray, G. A. 2003. The evolution of transcriptional regulation in eukaryotes. *Mol. Biol. Evol.*, **20**(Sep), 1377–1419. 27

Xie, X., Lu, J., Kulbokas, E. J., Golub, T. R., Mootha, V., Lindblad-Toh, K., Lander, E. S., & Kellis, M. 2005. Systematic discovery of regulatory motifs in human promoters and 3' UTRs by comparison of several mammals. *Nature*, **434**(Mar), 338–345. 78

Yada, T., Totoki, Y., Ishikawa, M., Asai, K., & Nakai, K. 1998. Automatic extraction of motifs represented in the hidden Markov model from a number of DNA sequences. *Bioinformatics*, **14**, 317–325. 37

Yan, T., Yoo, D., Berardini, T. Z., Mueller, L. A., Weems, D. C., Weng, S., Cherry, J. M., & Rhee, S. Y. 2005. PatMatch: a program for finding patterns in peptide and nucleotide sequences. *Nucleic Acids Res.*, **33**(Jul), W262–266. 39

Yu, C. S., Chen, Y. C., Lu, C. H., & Hwang, J. K. 2006. Prediction of protein subcellular localization. *Proteins*, **64**(Aug), 643–651. 49

Yuh, C. H., & Davidson, E. H. 1996. Modular cis-regulatory organization of Endo16, a gut-specific gene of the sea urchin embryo. *Development*, **122**(Apr), 1069–1082. 16

Yuh, C. H., Bolouri, H., & Davidson, E. H. 1998. Genomic cis-regulatory logic: experimental and computational analysis of a sea urchin gene. *Science*, **279**(Mar), 1896–1902. 3

Zhu, J., & Zhang, M. Q. 1999. SCPD: a promoter database of the yeast Saccharomyces cerevisiae. *Bioinformatics*, **15**, 607–611. 72, 73