

# Computational localization of promoters and transcription start sites in mammalian genomes

Thomas Down

This dissertation is submitted for the degree of Doctor of Philosophy

Wellcome Trust Sanger Institute and Sidney Sussex College, Cambridge

# Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

The work in this thesis has not been submitted in whole, or in part, for a degree, diploma, or any other qualification at any other university.

Thomas A. Down

May 2003, Cambridge, UK

# Abstract

A number of large genomes have now been sequenced, and biologists are now faced with the challenge of identifying all the functional pieces of sequence, and understanding how they contribute to the development and life of the organism. While identification of protein coding genes, and annotating their products, has been progressing well, there are a great many open questions relating to the regulatory regions which control the expression of these genes.

Here, I investigate the question of identifying and annotating promoters, one of the most important regulatory signals in the genome, which mark the points where transcription is initiated, and regulate the transcription of genes. I present a new computational method, EponineTSS, which can predict transcription start sites in bulk genomic sequence data with excellent sensitivity and specificity. Unlike the existing methods, it gives an indication of the actual location of the transcription start site. Comparisons with available experimental data suggest that the positional accuracy of these predictions is very good. Results from this method are included as part of the Ensembl human genome annotation.

Having located transcription start sites for genes, I also discuss the use of results from comparative genomics to estimate the extent of the functional promoter region upstream of the start site. I show that the extent of promoters is very variable, and that promoter size is correlated with the function of the gene for whose regulation it is responsible. Genes associated with developmental processes tend to have particularly large, and thus presumably complex, promoters, with the homeobox transcription factors among the most extreme examples.

I also introduce sparse Bayesian learning, a recently developed approach to supervised machine learning which can be applied to the training of a wide range of model types, and embodies the principle of selecting the simplest possible model to explain the observed data. I demonstrate a new technique which makes sparse Bayesian learning much more scalable,

allowing it to be applied to very large and complex problems, and present a convenient, freely available Java library which provides a general-purpose implementation of this technique. This library was used here in the training of the transcription start site predictor, but has a wide range of applications in computational biology and beyond.

# Acknowledgments

Thanks to the Wellcome Trust, and everyone at the Sanger institute who helped and encouraged me as I explored this fascinating area of research. I particularly appreciate the efforts of my project supervisor, Dr. Tim Hubbard, who has helped me along the path towards this thesis, accepted the many digressions, and added a valuable voice of caution and rigour when analyzing new results. Just because you're paranoid, it doesn't mean the data *isn't* out to get you.

All the programs I wrote during this project rest on the strong foundation of the BioJava library. I have to thank everyone who has ever contributed to this project: documentation, code, bug reports and ideas are all vital to the success of an open source effort. Right from the outset, Matthew Pocock has been a guiding light for the project. He's also a great coder, target for trying out new ideas, and friend.

Throughout this all, I've enjoyed the love, support, and good humour of Emily White, who has kept me going through the difficult moments and helped celebrate the joyous ones. Thank you.

Curated annotations for human chromosome 22 were produced by the Chromosome 22 group from the Sanger Institute and were obtained from the World Wide Web at <http://www.sanger.ac.uk/HGP/Chr22/>

# Contents

<b>Chapter 1. Introduction</b>	1
1.1. Objectives of this project	6
1.2. What is a promoter	7
1.3. Resources for studying promoters	13
1.3.1. EPD: the eukaryotic promoter database	14
1.3.2. Full-length cDNA sequences	15
1.3.3. TRANSFAC	17
1.4. Existing computational methods for studying promoter	17
1.4.1. Well-known motifs and Position Weight Matrices	18
1.4.2. CpG islands	21
1.4.3. PromoterInspector	23
1.5. Other resources used in this project	24
1.5.1. BioJava	24
1.5.2. Ensembl	26
1.5.3. The Gene Ontology (GO)	28
<b>Chapter 2. Sparse Bayesian Learning</b>	31
2.1. Generalized Linear Models (GLMs)	32
2.2. Feature selection using pruning priors	35
2.3. A pragmatic approach to handling large spaces	42
2.4. A general-purpose Sparse Bayesian trainer	46

2.5. Sparse Bayesian Learning Discussion .. .. .	48
<b>Chapter 3. Modeling of transcription start sites .. .. .</b>	<b>50</b>
3.1. The Eponine Anchored Sequence (EAS) model .. .. .	52
3.1.1. Learning EAS models .. .. .	55
3.1.2. Implementation and validation of EAS .. .. .	57
3.2. Training a transcription start site model .. .. .	60
3.3. Model refinement using mouse cDNA sequence data .. .. .	62
3.4. Validation and testing of EponineTSS .. .. .	65
3.4.1. Testing on human chromosome 22: bulk genomic performance	66
3.4.2. Testing on EPD: calibration of positional accuracy .. .. .	71
3.4.3. Comparison with other methods .. .. .	72
3.5. Analysis of cases where promoters were not detected by EponineTSS ..	74
3.5.1. Modeling of non-detectable promoters .. .. .	75
3.5.2. Correlation of promoter-detectability with gene type and function	77
3.6. EponineTSS discussion .. .. .	83
<b>Chapter 4. Learning from comparative genomics .. .. .</b>	<b>88</b>
4.1. The Eponine Windowed Sequence (EWS) model family .. .. .	90
4.2. Training from non-coding homologies between human and mouse .. ..	93
4.3. Evaluating the function of mouse-similarity models as promoter predictors	100
4.4. An attempt to discover a second signal in mouse-human homologies ..	104
4.5. Comparative Genomics Discussion .. .. .	109

<b>Chapter 5. Evolutionary conservation of promoter regions</b>	.. .. .	113
5.1. Alignment of promoter regions	.. .. .	115
5.2. Relationship of promoter alignments to regulatory roles	.. .. .	121
5.3. Discussion of promoter conservation	.. .. .	130
<b>Chapter 6. Conclusions</b>	.. .. .	134
<b>References</b>	.. .. .	139



# List of Figures

1.1. Flow of biological information from the genome to the proteome. In this case, the pre-mRNA can be spliced in two possible ways, leading to different protein products. .. .. .	2
1.2. Illustrative diagram showing the basic structure of a eukaryotic protein-coding gene. .. .. .	3
1.3. X-ray structure of transcription factor Pu.1 (shown in red) bound to a synthetic DNA substrate (green). .. .. .	10
1.4. A (simplistic) overview of keys steps in transcription initiation at a <i>polIII</i> promoter.	11
1.5. X-ray structure of DNA (shown in green) coiled around an octamer of core histone proteins (red). .. .. .	12
1.6. A protocol for oligonucleotide-labeling of mRNAs with intact cap structures, used to preferentially clone full-length mRNA sequences. .. .. .	16
1.7. Hidden Markov Model of a motif (states m1 to m5) embedded in a longer sequence. .. .. .	20
1.8. Logo view of a Position Weight Matrix model of the TATA box (redrawn from data on the EPD website) .. .. .	21
1.9. Example of a directed acyclic graph of Gene Ontology terms. .. .. .	29
2.1. Plot of the logistic function (equation 2.1.2). .. .. .	34
2.2. Plots of the Gaussian distribution, $\mathcal{G}(\beta_i   0, \alpha_i^{-1})$ , for various values of $\alpha$ . .. .. .	39
2.3. Example of sparse Bayesian learning in Cartesian 2-space. The data points chosen as centers for the final set of basis functions are highlighted in green. .. .. .	41
2.4. Example of a data set used for testing of training speed. .. .. .	44

2.5. Training time vs. problem size for full-set and incremental sparse Bayesian learning. . . . . 44

3.1. Example schematic architecture of an Eponine Anchored Sequence model. . . 54

3.2. Schematic of an EAS model learned from the synthetic dataset, showing the three spiked motifs. . . . . 58

3.3. Learning curves for the training of an EAS model on the synthetic dataset. . . 59

3.4. Accuracy vs. coverage (ROC) curve for a model trained on the synthetic dataset. 60

3.5. The EponineTSS\_1 model, trained from 389 mammalian sequences from EPD. . . 62

3.6. Histogram showing offsets of DBTSS start sites relative to those of corresponding EPD entries. . . . . 63

3.7. The EponineTSS\_2 model, trained from 599 mouse sequences. . . . . 65

3.8. Selection of regions to include when the pseudochromosome sequence. . . . . 68

3.9. Accuracy vs. coverage for two EponineTSS models on the pseudochromosome. 69

3.10. Density of prediction from the EponineTSS\_2 model relative to annotated gene starts. . . . . 70

3.11. Accuracy vs. coverage for the EponineTSS\_2 model on pseudochromosomes based on old (2.3) and new (3.1b) curated annotation of chromosome 22. . . . 71

3.12. Density of EponineTSS\_2 predictions relative to the annotated TSS of EPD entries. . . . . 72

3.13. Intersection of “correct” predictions of promoters by EponineTSS, PromoterInspector, and CpG islands. . . . . 75

3.14. Model trained on negative-selected FANTOM data. . . . . 76

3.15. Accuracy vs. coverage for model trained on negative-selected FANTOM data, with random predictions for comparison. . . . . 77

3.16. Example GO term lineage. . . . . 79

4.1. Length distribution of sequence regions aligned between human and mouse by ensembl-compara version 5. .. .. . 94

4.2. Scatter of scores from two different EWS models. .. .. . 97

4.3. Ensembl contigview displays for selected portions of human chromosome 22, showing windows with high scores for one of the homology models (labeled “ews1\_0.95”), and predictions from the EponineTSS\_2 model (labeled “Eponine”).  
.. .. . 98

4.4. ROCs for EWS models trained from sets of 2000 or 4000 human-mouse homologies. .. .. . 100

4.5. Intersection of transcription start sites correctly predicted by the EponineTSS, EponineHomol, and CpG methods. .. .. . 103

4.6. A set of basis functions learned by training the EWS-scaffold system on human-mouse homologous sequences. Each cell of this table shows an individual basis function, made up of between one and three sequence motifs. .. .. . 105

4.7. Scores for a model trained from the reverse-selected dataset vs. EponineHomol\_1. 106

4.8. Contigview displays showing both EponineHomol\_1 and EponineHomol\_2 predictions. .. .. . 108

4.9. Intersection of “correct” promoter predictions from EponineTSS, EponineHomol\_1, and EponineHomol\_2. In each compartment, the first figure indicates the number of start sites which were also detected by a CpG island predictor, while the second figure gives the total number. .. .. . 109

5.1. Agreement of promoter orthology with protein orthology. Panel a shows coverage (proportion of sequences correctly paired) at high levels of accuracy, with and without additional repeat-masking using **etandem**. Panel b shows a wider range of coverages, and only includes the results with **etandem** masking. .. .. . 119

5.2. Comparison of orthology in the windows [-2000:0] and [-4000:-2000] .. .. 121

5.3. Histograms of sequences binned by number of bases of promoter sequence included  
in blastn alignments to the orthologous promoter. .. .. 125

5.4. Aligned regions from 250 transcription-associated genes, sorted by number of  
aligning bases. .. .. 126

5.5. Dot plot between human and mouse upstream regions with low similarity. .. .. 127

5.6. Dot plot between human and mouse upstream regions with moderate similarity. 128

5.7. Dot plot between human and mouse upstream regions with strong similarity. .. 129

# List of Tables

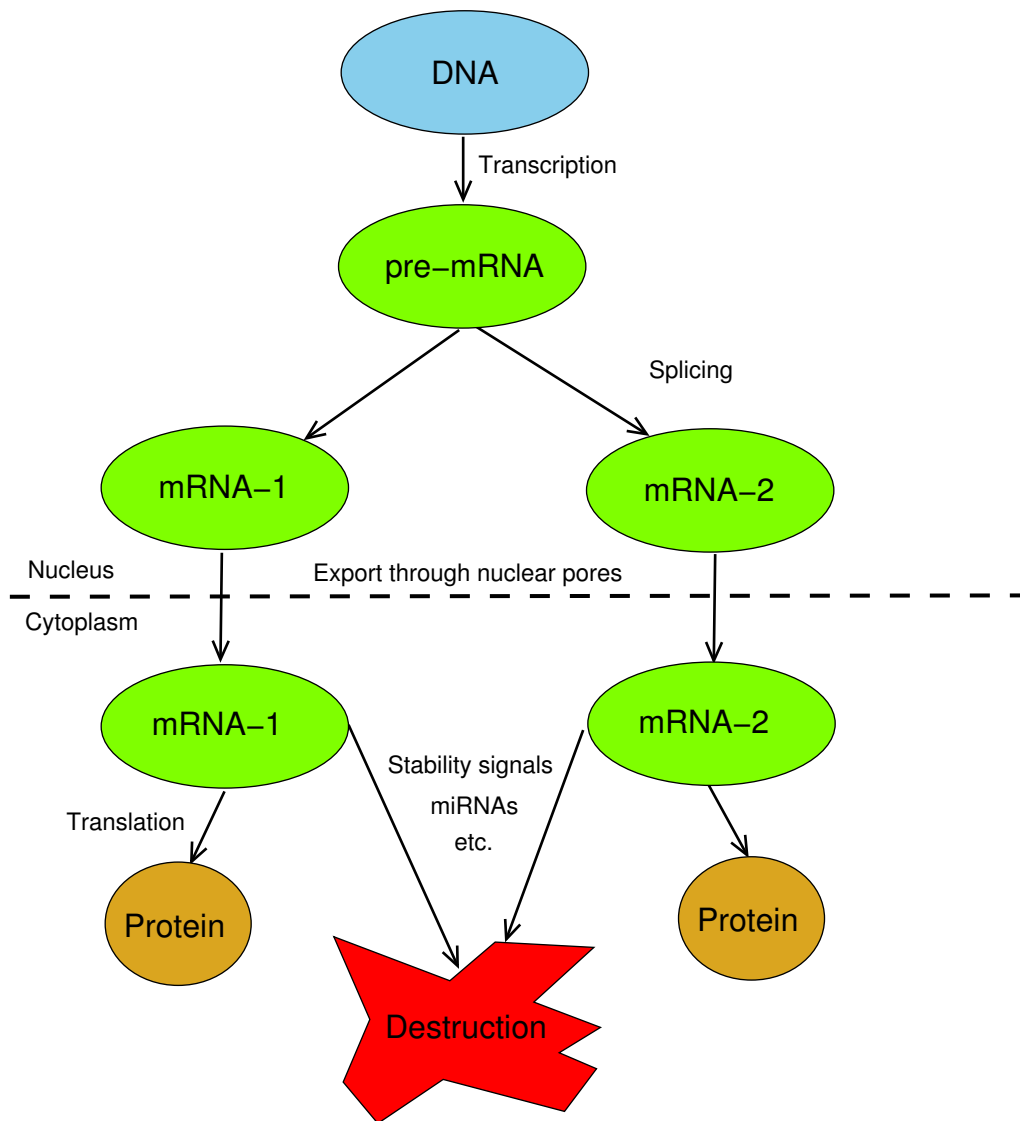
3.1. The four position weight matrices used in the EponineTSS_2 model. . . . .	66
3.2. Sensitivity and selectivity of various promoter-prediction mechanisms on the human pseudo-chromosome. . . . .	74
3.3. GO terms applied preferentially to <b>found</b> genes. . . . .	80
3.4. GO terms applied preferentially to <b>unfound</b> genes. . . . .	81
4.1. Words learned from three EWS models trained from mouse-human homologies.	96
4.2. Logo view of the motifs used in the EponineHomol_1 model. . . . .	102
4.3. Logo view of the motifs in the EponineHomol_2 model. . . . .	107
5.1. Default <b>blastn</b> parameters used . . . . .	117
5.2. GO terms which are overrepresented in the long-aligning promoter set. . . . .	123
5.3. GO terms which are overrepresented in the short-aligning promoter set. . . . .	124

# Chapter 1. Introduction

Large scale analysis of genomes remains a very new field of study. The publication at the end of 1998 of the 102 megabase *C. elegans* genome provided our first glimpse at a near-complete sequence for a multicellular organism [*C. elegans* Sequencing Consortium 1998]. This was followed in 2001 and 2002 by draft sequences for the human [IHGSC 2001] and mouse [MGSC 2002] genomes respectively. Both of these are – at least in terms of number of nucleotides – around 30 times larger than *C. elegans*. Since then, the rate of genome sequencing has continued to accelerate and many more sequences have been published, including a number of additional vertebrates. This genomic revolution has promised great developments, both in terms of pure science and in practical developments in the fields of medicine and commercial biotechnology. However, unlocking this potential requires additional experimental work, and also high-throughput analysis and data-mining methods.

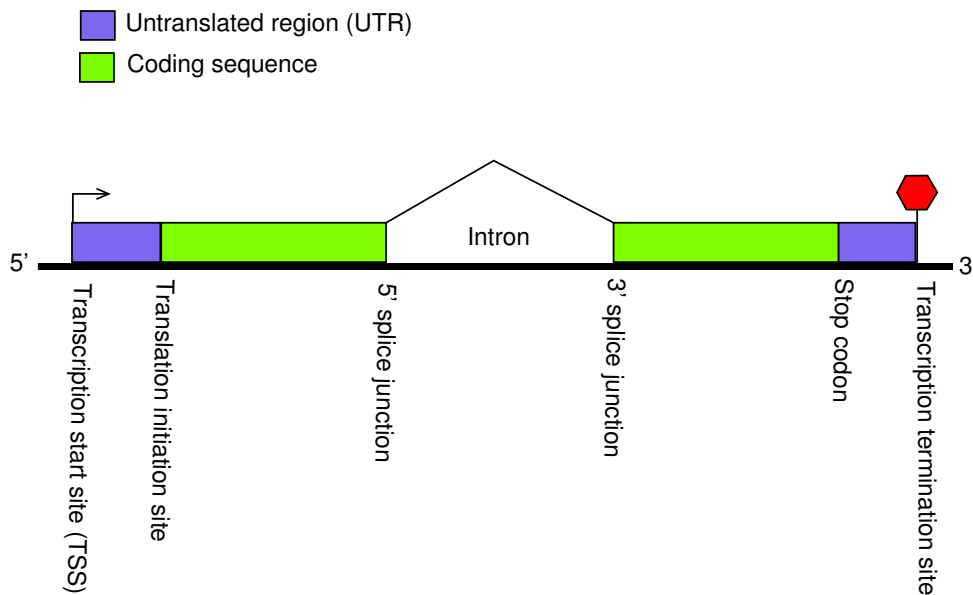
The function of the genetic material in the cell is part of the field known as molecular biology. The core processes of molecular biology can be seen as a pipeline of information flowing from the genomic DNA molecules, via a pool of RNA messengers (the transcriptome) to the set of proteins which mediate most of the cell's biochemical functions (the proteome). This has become known as the central dogma of molecular biology, and a schematic of the information flow is shown in figure 1.1. This model is something of a simplification, since while the primary role of RNA is to provide an intermediate stage in information flow between DNA and protein, many RNA molecules perform important functions – catalytic, structural or regulatory – in their own right [Eddy 2001].

An alternative, sequence-centric, view of this process is given in figure 1.2. A primary transcript is produced from the region between the transcription start and termination sites,



**Figure 1.1.** Flow of biological information from the genome to the proteome. In this case, the pre-mRNA can be spliced in two possible ways, leading to different protein products.

then introns are spliced out. Note that this diagram is somewhat schematic: in real mammalian genes, introns are usually much longer than exons, and most genes have a number of introns. In many cases, different sets of introns can be spliced to give alternative products from the same primary transcript [Ladd and Cooper, 2002]. As well as interrupting the coding regions, introns often occur in the 5' untranslated region, but only very rarely in the 3' UTR [Pesole *et al.* 2001]. Finally, at least in some cases, a single gene has more than one possible transcription start site (for example, [Laurinn *et al.* 2000]), although it is not certain just what fraction of genes possess this extra dimension of complexity.



**Figure 1.2.** Illustrative diagram showing the basic structure of a eukaryotic protein-coding gene.

In bacteria, the basic flow of information is similar, but the details are a little simpler: since there is no distinct nucleus, there is no specific export step. Bacterial messenger RNAs generally have no introns, removing the need for the splicing step, but eliminating the possibility for multiple splice variants from the same primary transcript. The rest of this thesis, however, concentrates exclusively on the molecular biology of eukaryotic cells, and in particular on the recently sequenced mammalian genomes.

Having identified the basic form of the molecular biology pipeline, the challenge to biologists today is to gain a full understanding of how pure, digital, information encoded as a string over the four-letter DNA alphabet leads can define the biochemistry (and larger-scale properties) of a living organisms. Clearly, this knowledge will play a major part in understanding the action of genetic diseases, particularly the polygenic diseases which cannot be traced back to a single, obvious, defect in a single gene [Wright *et al.* 2003]. As our understanding of molecular biology grows, it also seems likely that an increasing range of manipulations and interventions will be possible. For instance, detailed understanding of the pathways which lead cells in higher organisms to form tissues and organs could ultimately lead to *in vitro* production of replacement



organs [Risbud 2001]. A good test of the state of biological knowledge would be to build a computational model the processes of life in sufficient detail that it is possible to predict the effects of any perturbation in the genome.

To date, most analysis of the genome sequences has focused on the protein-coding regions – those portions of the sequence which are expressed as messenger RNAs and act directly as templates for protein synthesis. At a basic level, at least, these pose rather easy targets for study: there is a simple, well-defined code which relates groups of three nucleotide “symbols” in the DNA genome (or the RNA messenger) to one amino acid “symbol” in the protein product. This is often called the universal genetic code, since it is conserved with only a few minor variations throughout all known branches of terrestrial life. Detecting coding sequences is a fairly well-understood problem. Typical methods combine a model of coding sequence (often expressed as a table of hexamer frequencies) with additional logic to detect likely splice sites [reviewed in Mathé *et al.* 2002]. A rather different approach, which emphasizes just how distinctive coding genes can be from the genomic background, is described in [Pocock 2001]. Here, an unsupervised machine learning approach is used to partition the genome of *Plasmodium falciparum* (a malarial parasite) into several classes, with no *a priori* definition of what these classes should actually represent. This approach consistently identified the parasite’s coding regions as a single, distinct, class.

Detecting vertebrate genes – which tend to be large entities with many big introns separating relatively small (often less than 100 nucleotides) fragments of actual coding sequence – is arguably the greatest challenge in gene prediction. However, a combination of purely computational methods such as the widely used Genscan algorithm [Burge and Karlin, 1997] with experimental evidence such as Expressed Sequence Tag (EST) sequences [Adams *et al.* 1991] can give good quality gene annotation. For the human genome, a manual curation process is currently underway to create a definitive annotation set, but in the mean time there are also fully-automated methods, notably the Ensembl [Hubbard *et al.* 2002] annotation pipeline, which uses a range of experimental data (protein, cDNA, and EST

sequences) plus some computational input to provide a good quality first-pass annotation set. While Ensembl was created primarily to analyze human sequences, the annotation pipeline is now routinely applied to a number of other metazoans.

Before the sequencing of the human genome, there was a great deal of speculation about the number of human genes – especially once the *C. elegans* gene was published, confirming that this simple nematode worm has around 19,000 genes. Estimates from 30,000 to 120,000 human genes have been widely circulated, with many commentators favouring the higher end of this range (see, for example, [Fields *et al.* 1994]). A final figure will have to wait for more comprehensive curated annotation of the genome, but at the time of writing, the current Ensembl human genome release (version 13.31) contains predictions for 24,847 protein-coding genes. Ensembl annotation methods are, by design, relatively conservative so this is probably an underestimate, but many researchers today expect a final count of around 30,000 protein-coding genes. Relatively speaking, this is only a small increase compared to the worm. This number does not seem to reflect the apparent difference in complexity between the human body (with around  $10^{13}$  cells and a complex nervous system) and the worm (with a fixed developmental pathway culminating in an organism with just over  $10^3$  cells). The conclusion here has to be that while the set of building blocks for a human may not be substantially larger than that for some much lower organisms, the networks of regulatory molecules which process information during the developmental process and mark out the parts of the body are substantially more complex. To understand biology, and especially developmental processes, it will be important to look at complete networks as well as their individual building blocks.

All the processes shown in figure 1.1 are regulated to a greater or lesser extent, either by proteins, small molecules (usually interacting with DNA via a protein), or by regulatory RNA molecules such as the micro-RNAs (miRNAs) [Lewin 2000, Grosshans and Slack, 2002]. So the interplay of the cell's current population of proteins and RNA influences the population at some point in the future, providing the basis for regulatory networks. The basis for most of the known regulated processes in molecular biology – transcription, alternate splicing, and RNA

stability – involves interactions between the regulator model and specific regulatory regions of a nucleic acid molecule, which might be either the genomic DNA or an RNA transcript. In either case, sequences which function as regulatory targets share the genome with (and in some cases, such as exonic splice enhancers, actually overlap) the protein-coding regions [Blencowe 2000]. Therefore, they can be seen as a second genetic code which conveys information about when, where, and how much of a protein should be produced. Today, our techniques of genome analysis make it possible, given some raw sequence data, to identify protein-coding genes and to predict the sequence of the protein they code for. This can then be compared with known protein sequences to give at least some indication of the protein's likely structure and function. An important step towards complete understanding of the genome will be to perform an analogous process on the regulatory regions: first to identify these regions in bulk genomic sequence, and then to “read” the regulatory codes, gaining some information about the set of regulator molecules which interact with a particular region, and thus where it fits into the cell's regulatory network.

### **1.1. Objectives of this project**

An important direction for computational biology over the coming years is to develop methods for identifying and decoding regulatory regions in genomic sequences – especially large and complex genomes, such as those of vertebrates. The order here is significant: it is necessary to identify the regions of interest before detailed study and decoding are possible. For this reason, in this project I concentrated primarily on this first identification step. I also decided to focus on one particular class of regulatory signals: the promoters, which are located near (primarily upstream of) the transcription start site and regulate transcription initiation. While these are certainly not the only significant type of regulatory region in the genome, they are clearly vitally important since every gene has one. This property also means that the set of examples available for study is large, and that it is reasonable to assume that every point in the genome which can be shown to function as a transcription start site (TSS) must have some kind

of associated promoter region.

Having chosen this objective, chapter 3 of this thesis describes a novel method for predicting transcription start sites: the key points around which promoters are organized. A second predictive method is described in chapter 4. Chapter 5 discusses how sequence from a second species – in this case, mouse – can be used to highlight regions of extremely high evolutionary conservation which are believed to indicate the cores of the functional promoter regions.

I also hoped that, in achieving my primary objective, I would develop technologies that will prove useful in the coming years for dissecting regulatory regions, identifying the sequences which make these regions targets for particular regulatory pathways, and therefore providing the knowledge to start actively decoding novel regulatory regions. Chapter 2 describes a recent, highly versatile, approach to machine learning which can be shown, on the basis of results in chapters 3 and 4, to be effective for sequence analysis problems. This approach may well be applicable to further analysis of regulatory regions.

The remainder of this chapter gives a brief background on the current state of knowledge about promoter regions, and discusses some tools and resources which were helpful for studying them.

## 1.2. What is a promoter

Briefly, a promoter is a region of DNA sequence close to a gene's transcription start site, at which molecular events occur leading to the initiation of transcription. They are sometimes described as *cis*-regulatory regions, as opposed to *trans*-regulation which implies regulators situated some distance from the gene itself. Needless to say, few if any promoter regions, even in the simplest organisms, blindly trigger the production of new transcripts. Even single cells will have many genes which must be turned on and off under specific circumstances, for example

enzymes forming the pathways for metabolizing various different energy-source compounds. And no protein is required in an unlimited amount, so feedback mechanisms which turn off transcription once an adequate concentration has been achieved will make the cell function more efficiently. Multicellular organisms require proportionately more complex regulation – at a minimum, each cell type will have its own pattern of gene expression, and there is usually also extra complexity in the form of signaling mechanisms exchanging information between cells and tissues. A number of enhancer sequences, which control the rate of transcription from locations some distance from the transcription start site (and thus represent a form of *trans*-regulation) have also been identified. While these are clearly an important part of the transcriptional regulation machinery, their location further from the transcription start site and the fact that many genes may not have any associated enhancer region at all mean that they are currently rather difficult to study, and they are not considered further in this thesis.

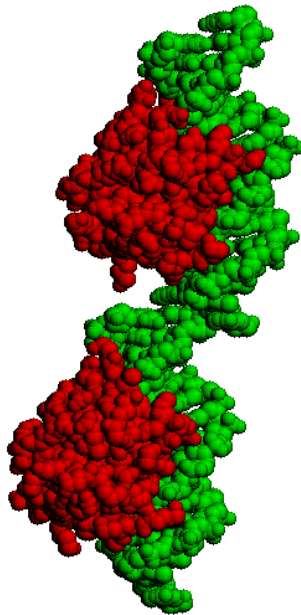
Promoters have been studied in the laboratory for many years, first in bacteria [Pardee *et al.* 1959] and then in eukaryotic cells. A wide variety of experimental techniques have been used, including directly mapping the interactions of proteins to DNA [Galas and Schmitz, 1978], and attempting to carry out transcription *in vitro* [Hayashi 1965]. In some senses, promoter biology is advancing reasonably well. Given a gene, and some laboratory time, it is often possible to clone the promoter region then couple it to a reporter gene – typically either a fluorescent protein or an enzyme which produces a coloured product – to identify the expression pattern. It is then possible to delineate the actual boundaries of the promoter experimentally, for example by performing targeting deletions. A number of strong (in the sense of causing high levels of transcription) promoters with useful and well-characterized expression patterns are available “off the shelf”, and are used to drive the expression of transgenes for experimental purposes, or in practical biotechnology applications. This is especially relevant in plants, where a small number of promoters, such as the Cauliflower Mosaic Virus promoter, are used for many purposes. However, for this kind of application it is usually sufficient to treat the promoter as a black box, assuming that if it gives a particular expression level and pattern for

one gene, it will do the same for a second gene in the same (or even a related) organism. This does not directly improve our ability to decode arbitrary promoters, or to engineer new promoters from scratch to achieve a specific expression pattern. In addition, promoter cloning and delineation remains a fairly complex and labour-intensive process. And methods for determining the expression pattern, which are generally most effective when dealing with reasonably high levels of expression, may fail for genes which are only expressed at the level of a few copies per cell, or which are only active under a very small range of circumstances. It seems unlikely that the full set of promoters from any complex organism will be identified in this manner in the foreseeable future.

Some promoters have been studied in much more detail, to the level of understanding many of the interactions between DNA and protein. Simple model organisms, notably yeast, are the convenient and popular choice for this kind of research, although more complex model organisms, and even human cell lines, have been studied in some cases. To date, this kind of work has been our most important source of understanding in how promoters and the transcriptional machinery actually function. While it is unlikely that the full repertoire of promoters will be studied in depth by individual experiments, there is now an approach to mapping protein-DNA interactions, at least at a moderate resolution, which appears amenable to high-throughput application. Briefly, preparations of chromatin are fragmented, then the fragments associated with a protein of interest are precipitated using appropriate antibodies. DNA fragments extracted from the precipitate can then be mapped back to the genome by hybridization onto a DNA microarray [Ren *et al.* 2000]. The future significance of this technique, known as ChIP-on-chip, is discussed in the concluding chapter of this thesis.

Eukaryotic cells actually have three different forms of RNA polymerase, the enzyme responsible for transcribing RNA from DNA. Two of these, *polII* and *polIII* have only very specialized roles relating to transcription of certain noncoding RNA genes. All protein-coding RNAs, and many others, are transcribed by *polIII*, making this the most interesting (and best studied) of the three. Studying the complexes which the *polIII* core forms with DNA, it is

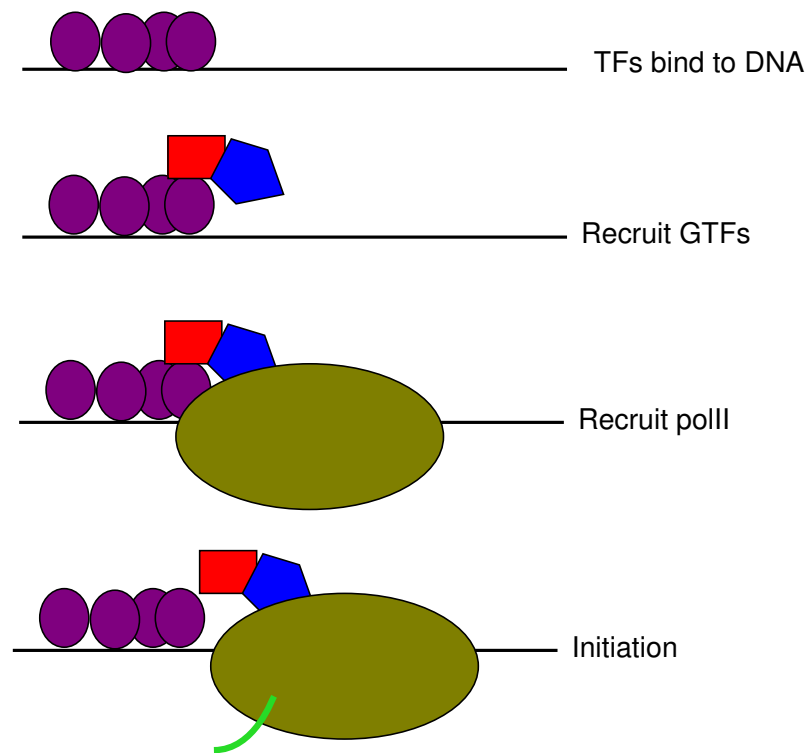
found to be associated with a wide range of proteins. The bulk of these are characterized as *transcription factors* (TFs) [Lewin 2000]. These proteins containing DNA-binding domains from one of several large families such as the zinc finger and leucine zipper families, and recognize particular elements in promoter regions. The structure of one transcription factor (PU.1, a form of helix-turn-helix factor), bound to a short synthetic DNA molecule, is shown in figure 1.3. This was drawn from the PDB entry 1pue [Kodandapani *et al.* 1996].



**Figure 1.3.** X-ray structure of transcription factor Pu.1 (shown in red) bound to a synthetic DNA substrate (green).

Transcription factors are very common: the GO annotation of Ensembl human gene predictions (see page 28) lists 1028 transcription factors, and since not every gene is annotated in this way, this is almost certain to be an underestimate. Also involved in *polIII* complexes are the general transcription factors (GTFs) and adapters, which do not necessarily recognize specific DNA sequences, but are required parts of the functional transcription complex. A naive view of transcription initiation is shown in figure 1.4.

In fact, we know that figure 1.4 is a significant over-simplification in several respects. Genomic DNA in its natural context is certainly not a simple linear molecule, and the

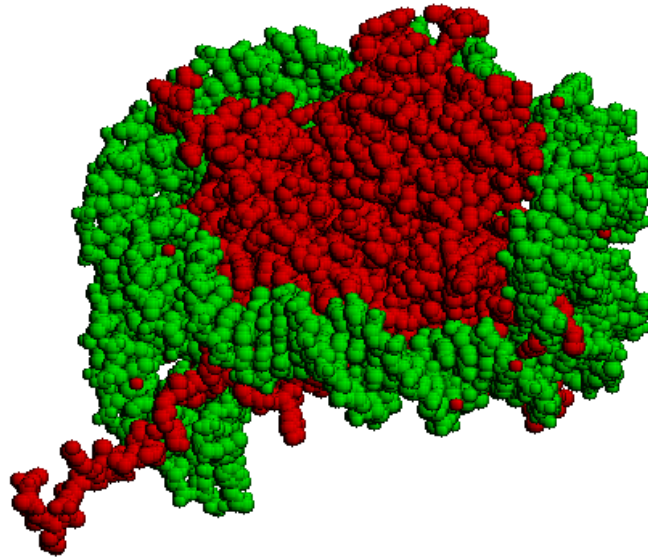


**Figure 1.4.** A (simplistic) overview of keys steps in transcription initiation at a *polIII* promoter.

transcription factors discussed here are unlikely to have unimpeded access to their binding sites. The DNA double-helix is packaged at several levels. Firstly, units of around 150 nucleotides are coiled around octamers of the core histone proteins to form nucleosomes, which can be seen as “beads” on a string when viewed under the electron microscope. The structure of a nucleosome has been determined by X-ray crystallography [Luger *et al.* 1997] and is shown in figure 1.5. The DNA is then packaged further by weaker associations with histone H1 to form the chromatin fibre [Kornberg 1999]. While this system may have evolved primarily to protect the DNA and make it easier to move the chromosomes around during cell division, it must clearly impact on all other processes which involve interaction with the DNA. A number of systems have been identified which alter DNA packaging, including enzymes which add and remove acetyl groups from the histone cores, and remodeling complexes which appear to physically move nucleosomes along the DNA molecule. Many of these enzymes have been found in complexes with *polIII* and transcription factors [Brownell *et al.* 1996]. One conclusion to draw from this is that it is generally wrong to think of transcription initiation as an isolated event.



When a transcription complex is recruited to a promoter, as well as triggering the production of a single RNA copy, it also makes more permanent changes to the chromatin around the promoter, facilitating initiation of subsequent transcripts [Kadonaga 1998].



**Figure 1.5.** X-ray structure of DNA (shown in green) coiled around an octamer of core histone proteins (red).

Another issue with the model of figure 1.4 is that many of the components of the transcription complex appear to exist in complexes – called holoenzymes by analogy to complexes involved in bacterial transcription mechanism – even when not associated with DNA [Carey 1995]. Holoenzymes were first observed in budding yeast, but have since been found in other species as well [Myer and Young, 1998]. Instead of a model where proteins are recruiting one at a time to a transcription complex, it is now thought that large parts of the complex arrive in a single step [Barberis and Gaudreau, 1998]. Presumably there is still some initial binding of individual TFs to the DNA which starts the process and causes recruitment of the holoenzyme.

One final complication, especially in higher eukaryotes, is that covalent modifications can occur to the DNA nucleotide residues themselves, so in effect the alphabet of genomic DNA is actually larger than the four letters normally considered. In mammals, almost all occurrences of the dinucleotide 5'-cytosine-guanine-3' (usually written CpG) are modified to use methylcytosine in place of cytosine. Demethylation of CpG sites is associated with promoter activation [Razin 1998]. When transcription begins at a previously inactive promoter, demethylation seems to accompany chromatin remodeling. In addition to the permanent regulatory signals encoded in the DNA sequence, promoters have a state, and can be switched between inactive (methylated CpG, tightly packaged chromatin) and active (demethylated, with more “open” chromatin structure) states. Information stored as patterns of DNA or histone modifications, rather than directly in the DNA sequence, is often described as epigenetic information [Holliday 1987].

Nevertheless, the naive model of transcription initiation remains a useful view to bear in mind when working with promoters. It suggests that, at a sequence level, a promoter will consist of a cluster of discrete sites at which the transcription factors bind. In the most naive view, sequence between TF binding sites is irrelevant, but in practice there might also be constraints on this: obvious possibilities are signals which control the positioning of nucleosomes to maximize accessibility of the TF binding sites, and compositional biases around the TSS which make it easier for the transcription complex to “open” the DNA helix. Moreover, individual transcription factors must make contact with other proteins in the transcriptional complex. This means that some TFs might have preferred positions in the complex, and their corresponding binding sites in the promoter region will function more effectively if they are in particular positions (relative either to the transcription start site, or to one another).

### **1.3. Resources for studying promoters**

Since I was addressing this project from a computational point of view, I have been

reliant on published experimental work to provide useful sets of data which can be analyzed in order to learn more about promoters. Particularly valuable are large collections of data, either compendiums of individual experimental results, or the output from single high-throughput data collection exercises. The following resources are highly relevant to the study of eukaryotic promoters, and were considered throughout this project.

### 1.3.1. EPD: the eukaryotic promoter database

The Eukaryotic Promoter Database [Périer *et al.* 2000, EPD, <http://www.epd.isb-sib.ch/>] is, as the name suggests, a collection of promoter sequences from a range of organisms in the eukaryotic domain. Sequences are accepted according to a set of fairly demanding criteria, which are set out in the user manual [<http://www.epd.isb-sib.ch/current/usrman.html>]. Notably, transcription start sites must be mapped experimentally, and researchers submitting entries are expected to identify the main TSS with an accuracy of  $\pm 5$  bases. The one exception to this is for promoters which are identified purely on the basis of a strong homology to a previously characterized sequence. However, when using databases such as EPD for training or testing sequence analysis methods, it is normal to work with a non-redundant subset of the sequences, so all sequences which were accepted into the database on the basis of homology should be discarded anyway.

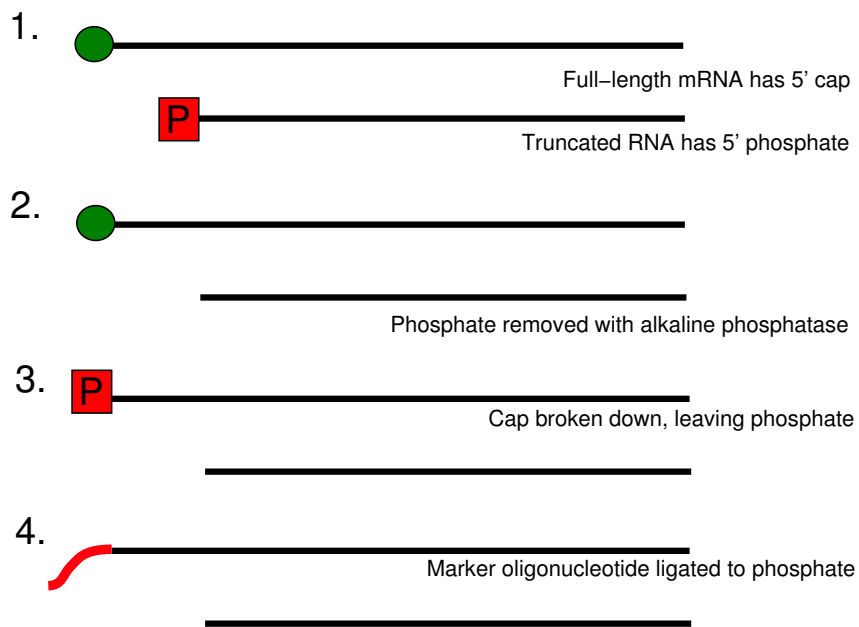
Although EPD is named as a promoter database, it does not necessarily give information about what biologists would consider to be promoter regions. A basic EPD entry just specifies a transcription start site (which is presumed to correspond roughly to the downstream ends of the promoter regions). This is not necessarily a defect: the concept of a transcription start site can be defined fairly rigorously, and experimental techniques for TSS mapping, such as primer extension [Green and Roeder, 1980], are well known and accepted. On the other hand, it is somewhat harder to say where the exact boundaries of an active “promoter region” lie, and even the previously described strategy of making a series of targeting deletions is unlikely to give an absolutely precise definition of the required region.

One point to note about EPD is that its contents are the results of fairly laborious experimentation, and it is therefore quite small. Recently, it has been growing extremely slowly. Release 62, dating from 2000, contained 389 mammalian promoters (after discarding near-duplicates). Two years later, in release 71 of the database, the number of distinct mammalian promoters had only grown to 400. Very recently, the database *has* grown much more rapidly, with over 2000 promoters in release 73, but the balance consists almost entirely of TSS predictions made on the basis of full-length cDNA sequencing projects, as described below. These methods have clear advantages in that they are comparatively easy to implement in high-throughput environments, but it is not yet clear that they offer TSS mapping of the accuracy offered by “traditional” EPD entries. Therefore, researchers may still wish to work with the traditional entries. I made significant use of EPD in this project, but all work on this database pre-dated the addition of the cDNA-based entries.

### 1.3.2. Full-length cDNA sequences

Complementary DNA (cDNA) is a general term for any DNA molecule produced from an RNA template by the enzyme reverse transcriptase – an enzyme which is required in the life-cycle of the retroviruses, but which has also proved to be a very valuable molecular biology tool. In practice, cDNAs of interest are generally DNA copies of messenger RNAs, produced in the laboratory as a method of studying the transcriptome. Sequencing of cDNAs has been taking place for many years. Initially, the emphasis was on sequencing just representative portions of the mRNA molecules, giving expressed sequence tag (EST) sequences [Adams *et al.* 1991]. But several recent projects have attempted to clone and sequence the complete length of messenger RNAs. The issue here is to distinguish between cases where the cDNA represents the full length of mRNA found in the cell, and those where it is truncated for some reason. Likely causes for truncation are the reverse transcriptase dissociating before reaching the end of the mRNA, or simple degradation of the RNA (which is chemically rather unstable *in vitro*). Recent projects such as the FANTOM collection of mouse cDNAs [Kawai *et al.* 2001] and DBTSS, a

similar human project [Suzuki *et al.* 2002], have used an approach called cap-trapping. This takes advantage of the fact that all eukaryotic mRNAs have a specially modified nucleotide residue (the cap) present at their 5' ends. This is added by specialized capping enzymes which are thought to function cotranscriptionally [Proudfoot *et al.* 2002]. A technique exists in molecular biology to attach oligonucleotide markers specifically to RNAs which possess this cap nucleotide, as illustrated in figure 1.6. Once this step has been performed, cDNAs are produced as normal, then cloned into plasmids using a protocol which requires the presence of a particular tag sequence – matching that of the oligonucleotide which was added in the cap-trap stage. In this way, it is possible to build a library consisting primarily of 5'-complete cDNA clones [Maruyama and Sugano, 1994].



**Figure 1.6.** A protocol for oligonucleotide-labeling of mRNAs with intact cap structures, used to preferentially clone full-length mRNA sequences.

If it was effective, full-length cDNA sequencing would be an attractive way to determine transcription start sites. However, it is not a complete solution to the problem: firstly, it may be difficult to isolate cDNAs from genes which are expressed at low levels, or from rare variants transcribed from alternative sites. Second, there remain questions about the effectiveness of the cap-trapping procedure. Cap-trapped sequences were used in this project, but rather than

assuming that the 5' end of one of these clones always reflects a true transcription start site, some caution was taken to use only those clones which appear most likely to be complete. Some results which seem to justify this cautious approach are given in chapter 3.

### 1.3.3. TRANSFAC

TRANSFAC is a database of known transcription factors and their binding sites – both natural sites identified by *in vivo* study of promoters and synthetic oligonucleotides to which the factors have been shown to bind *in vitro* [Matys *et al.* 2003]. The database is made up of several parts:

- A database of known transcription factor proteins (transfac)
- A database of binding sites for many of these factors (tfsite)
- A database of position weight matrices (see section 1.4.1) describing the preferred binding sites of a few factors where a sufficiently large number of distinct binding site sequences are known (tfmatrix)

The database of factors is clearly of interest when studying gene regulatory networks. Using the database of sites is more difficult: this contains a large number of entries (8414 in release 3.4) with binding site sequences varying for 2 to 128 bases in length (strongly biased toward the low end of that range). This is a diverse set of sequences: it includes all 1024 possible 5-letter DNA “words”, over 99% of the 6-letter words, and 84% of 7-letter combinations. Even considering 9-mers, almost 13% of them can be found in, so the probability of any sequence of less than 10 bases being present in tfsite is high. While I noted the existence of TRANSFAC, it was not directly useful in the course of this project.

## 1.4. Existing computational methods for studying promoter

Before this project, a number of attempts had been made to develop methods which could automatically annotate promoters regions or transcription start sites in eukaryotic DNA. Several attempts were made in the pre-genomic era to apply well-known sequence analysis methods such as hidden Markov models [Audic and Béraud-Columb 1997] and neural networks [Knudsen 1999] to the problem of promoter prediction. Work in this era was reviewed in [Fickett and Hatzigeorgiou 1997]. The authors of that review also performed an evaluation of the available methods on a small set of newly-published test sequences. This independent evaluation had the advantage that all the test sequences were obtained after the evaluated methods were published, making that the test entirely impartial since none of the test sequences could have been used when training and developing the prediction methods. However, the test set consisted only of 18 sequences, all of which were rather short (12 were less than 2 kilobases in length), and with some clear problems for any promoter recognition software – in one case, the mapped TSS was only 28 bases from the start of the sequence, giving little information for any method which looks for upstream signals. The evaluated methods all performed better than would be expected for random predictions, but in all cases, specificity was low. In the modern era of sequencing, where researchers often want to look at whole mammalian genomes rather than 2kb regions of interest, specificity is a vital requirement, as discussed in [Scherf *et al.* 2000].

Below, I discuss three specific methods of promoter recognition: one based on a machine learning approach, and two which are more biologically motivated. These methods will appear again in chapter 3 of this report, as benchmarks used when evaluating the EponineTSS method.

#### **1.4.1. Well-known motifs and Position Weight Matrices**

There are a number of short sequence motifs which have been associated with transcription initiation. Most of these are believed to represent preferred binding sites for one particular transcription factor. Few of these are 100% specific – in other words, there is not one single sequence which is recognized, but instead there are several choices of allowable nucleotide for at least some positions in the motif. Given a large number of examples, a good way to represent

knowledge about a motif is a Position Weight Matrix (PWM). This is a matrix where each element represents the probability of a given nucleotide occurring at a particular position in the sequence. In other words, each column of the matrix is a probability distribution over the DNA alphabet. A PWM can be viewed as a probabilistic model of a fixed-length sequence:

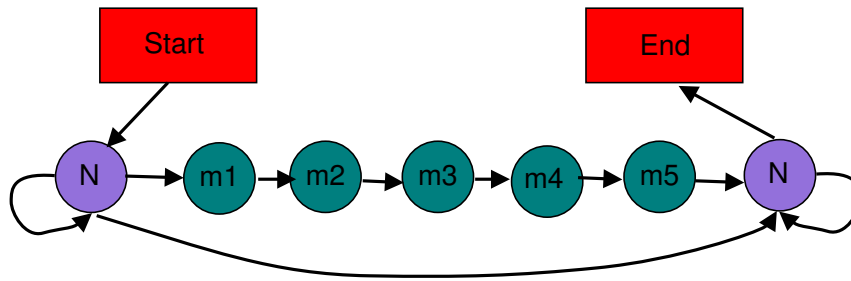
$$\vec{W}(S) = \prod_{i=1}^{|W|} W_i(S_i) \quad (1.4.1)$$

Where  $|W|$  is the length of the PWM, and  $W_i$  is the  $i$ 'th column of the matrix. The higher the probability of a given sequence under this model, the more similar the sequence is to the “ideal” motif, and therefore the higher the probability that it will function as a binding site. This is the zeroth order model: each position in the sequence is assumed to be independent of all the others. Here,  $\vec{W}$  and  $\overleftarrow{W}$  are used to indicate weight-matrix scores for the forward and reverse DNA strands respectively.

Considering a PWM (or other motif description) as a probabilistic model offers an approach for learning optimal PWMs from a set of sequences. A PWM can be seen as a degenerate form of a hidden Markov model [Durbin *et al.* 1998]. HMMs are state machines where each state has an associated emission spectrum over some alphabet of possible observations. In a PWM, the transitions between the states are fixed: after observing the nucleotide at each position, the machine always moves to the next state. But by adding a few additional states to the HMM, it is possible to build a model which emits a variable number of bases of flanking sequence on each side of the motif (figure 1.7). The set of parameters for an HMM which maximize the probability for a set of sequences (often called the maximum likelihood estimate of the parameters) can be found by applying the Baum-Welch algorithm [Durbin *et al.* 1998]. A concrete implementation of motif-learning using this approach is provided by the MEME package [Bailey and Elkan 1994].

It should be pointed out that the assumption of bases being independent in a motif is unlikely to be true in most cases. When proteins bind to DNA, they often significantly deform





**Figure 1.7.** Hidden Markov Model of a motif (states m1 to m5) embedded in a longer sequence.

it, so mechanical properties of the DNA other than the actual base sequence will be significant – in particular, the flexibility of the double helix. This property is determined largely by interactions between neighbouring base pairs. Models which take these, and perhaps also longer range interactions, into account can be expected to better predict the binding of a protein to a given sequence [Barash *et al.* 2003]. However, since non-independent models have many more parameters than simple PWMs, they require more example sequences to learn effectively.

Sequence motifs are commonly displayed in logo form – see, for example, figure 1.8. Here, each column of a PWM is rendered as a stack of symbols, with more height given to the most probable symbols. The total height of the stack is proportional to the Shannon information content of the distribution:

$$S = \log_2(|\mathcal{A}|) + \sum_{n \in \mathcal{A}} P(n) \log_2 \frac{1}{P(n)} \quad (1.4.2)$$

Where  $\mathcal{A}$  is the alphabet we are considering. This quantity is related to entropy, and is measured in bits. For a four-letter alphabet such as DNA, values can range from 0 (for a completely uniform distribution, *i.e.*  $P(n) = 0.25$  for every base) to 2 bits (for the case where only a single base is allowed). Therefore, tall stacks indicate strong constraints on the bases which are acceptable at a given position, while short stacks show positions with only a marginal preference for any particular base. Completely non-informative positions appear blank in the logo.

The TATA box is the best known element of eukaryotic promoter sequences, and was

recognized more than 20 years ago in early experiments with eukaryotic gene expression [Corden *et al.* 1980]. The TATA binding protein (TBP) is known to associate with the TATA box, when it is present. However, this association is weak, and TBP (which plays a central role in transcriptional complexes) is actually present even when no TATA box can be found [Rigby 1993]. The definitive TATA box PWM was learned by applying an maximum likelihood algorithm to sequences from EPD, and was published, along with several other motifs, in [Bucher 1990]. The logo form of this is shown in figure 1.8.



**Figure 1.8.** Logo view of a Position Weight Matrix model of the TATA box (redrawn from data on the EPD website)

The TATA PWM has been used on its own as a promoter prediction method, and was one of the candidate methods included in [Fickett and Hatzigeorgiou 1997]. Further information about the performance of this method appears in chapter 3.

PWMs are a widely used technique in computational biology, and are by no means specific to promoter research: another common application is to model the sequences around splice-junction sites in gene prediction programs. Profile HMMs are close relatives of PWMs, which allow small insertions and deletions relative to the expected consensus sequence, and are used by the Pfam project to identify families of evolutionarily-related proteins [Bateman *et al.* 2002].

#### 1.4.2. CpG islands

As previously mentioned, the the cytosine in the dinucleotide 5'-cytosine-guanine-3' can be covalently modified with a methyl group. Indeed, in mammalian genomes, this is extremely common and the vast majority of CpGs are methylated. CpG methylation is believed to convey

some information, since DNA is methylated in the course of some mechanisms of gene silencing [Razin 1998]. However, storing extra epigenetic information by methylation does not come without a cost. Cytosine bases are prone to deamination. Deamination of normal cytosine yields uracil, which is not normally present in DNA, and can be efficiently recognized and repaired. But the deamination product of methylcytosine is thymine, a normal DNA base. This leaves a thymine-guanine pair in the DNA helix, which may be detected and resolved by mismatch-repair mechanisms. This kind of DNA repair is much less efficient than the rapid detection and elimination of uracil, and a substantial proportion of methylcytosine-to-thymine mutations survive to be replicated. Sequencing of mammalian genomes has shown that CpG dinucleotides are rather rare. If bases in DNA were independent of their neighbours, we would expect the frequency of CpG dinucleotides to be given by:

$$F_{CG} = F_C F_G \quad (1.4.3)$$

In fact, over almost any stretch of mammalian sequence,  $F_{CG}$  is vastly lower than this expected value. But some regions of the genome contain much higher CpG levels than average: these are called CpG islands, and are known to be associated with the 5' ends of genes [Larsen *et al* 1992]. It has been further suggested that genes associated with strong CpG islands have “housekeeping” functions – in other words, they encode proteins which are ubiquitous throughout many cell types and developmental stages [Brandeis *et al.* 1993].

The accepted definition of a CpG island, taken from [Gardiner-Garden and Frommer, 1987], is a region of at least 200 bases where:

- At least 50% of bases are G or C
- The observed content of CpG dinucleotides is at least 60% of the “expected” figure from equation 1.4.3 for a random sequence with the same single-nucleotide composition as the region under consideration

A number of programs exist to detect CpG islands in genomic DNA sequences using the criteria

given above, and these are commonly used by curators annotating gene structures. When the 5' end of a gene structure annotated primarily on the basis of cDNA or EST evidence falls close to a CpG island, this is a good indication that the structure is not badly truncated. However, CpG islands do not provide strong information about the actual position of the transcription start site. The CpG dinucleotide is palindromic: reading the other DNA strand in the 5'-to-3' direction, you will also see C followed by G. This means that for a given region of sequence,  $F_{CG}$  will be the same for both strands, so the predicted CpG islands give no information about the direction of transcription.

### 1.4.3. PromoterInspector

PromoterInspector is a recent method for predicting promoter regions which works on the basis of detecting multiple motifs [Scherf *et al.* 2000]. It was trained based on sequences extracted from EPD, using a brute force method to determine sets of motifs which are over- or under-represented in promoter regions relative to the genome as a whole. In this case, the motifs are not represented as PWMs, but as simple strings, where some of the positions can be wildcard characters. This is equivalent to a degenerate PWM, where all columns have an information content of either 2 bits (requiring an exact match) or 0 bits (non-informative). This is likely to be less sensitive than allowing all possible PWMs, but has the advantage that it is possible to rapidly enumerate the complete set of patterns up to a given length, and count their occurrences in a set of training data. This makes the brute-force training approach practical.

By design, PromoterInspector does not give direct information about transcription start sites. Instead, it marks “promoter regions” on the genome. However, since the positive training set consisted simply of blocks of sequence 500 bases upstream of EPD transcription start sites, rather than mapped, active, promoter regions, there is no particular reason to believe that the boundaries of the predicted promoter regions correspond to the portion of the sequence which is actually biologically significant. A more conservative description would be “regions likely to contain core promoter elements”. Like CpG island predictors, PromoterInspector also gives no

information about the direction of transcription.

## 1.5. Other resources used in this project

The field of bioinformatics is characterized by large, complex data sets. For bioinformaticians concentrating on sequence analysis, the most significant recent developments have been the publication of various genome sequences, with the draft sequences of higher organisms such as human, mouse, and *Fugu* being particularly exciting. The availability of genome data is, of course, vital to this project. But the scale of genome data – plus complications specific to bioinformatics, such as the regularly changing assemblies of draft genome data – mean that they are relatively difficult to manage and work with. The following tools and resources were invaluable in the course of this project.

### 1.5.1. BioJava

BioJava is an open source library of tools and components for developing bioinformatics applications [The BioJava development group, <http://www.biojava.org/>]. It draws some inspiration from earlier projects like Bioperl [Stajich *et al.* 2002], but follows rather different design approaches, and places more emphasis on supporting developers working on new analysis methods, rather than manipulating output from existing tools. Most of the code in current versions is aimed at manipulating and analyzing sequence data, but future versions are expected to improve support for other data types, and integration of data sets from disparate sources. In use, BioJava has proved quite scalable: it is quick and convenient to work with small pieces of sequence data, but exactly the same Application Program Interfaces (APIs) can also be applied to much larger sets of data. With extensions like *bj-ensembl*, described below, it is easy to handle and query databases larger than the computer's memory.

One defining pattern in BioJava development has been the use of query languages in

preference to defining large sets of *getFooByBar()* accessor methods. The strongest example of this is the handling of annotated sequence, which can be queried using the *FeatureFilter* language:

```
// Locate all gene annotations on the positive strand,  
// overlapping the specified region  
  
Sequence seq = loadSequence();  
FeatureHolder features = seq.filter(  
    new FeatureFilter.And(  
        new FeatureFilter.ByType("gene"),  
        new FeatureFilter.And(  
            new FeatureFilter.OverlapsLocation(  
                new RangeLocation(1000, 2000)  
            ),  
            new FeatureFilter.ByStrand(  
                Strand.POSITIVE  
            )  
        )  
    )  
);
```

Simply allowing features to be requested based on a single criterion (type, or location) can be highly inefficient when working on large datasets, since this would require the program above to fetch *all* features in the requested region, then throw away all those which aren't genes on the positive strand. On the other hand, providing accessor methods for all the combinations of criteria which users might wish to access quickly gives unwieldy APIs which are hard to learn, and even harder to maintain and test effectively. By passing in an object which represents a query, users can naturally combine all the available filters operators using the *and*, *or*, and *not* operators, and easily ask complex questions. The underlying implementations can either implement the query system directly, or transform the query into some other language – for example, some BioJava database front ends can directly transform *FeatureFilter* objects into SQL queries. Plans are currently underway for BioJava 2, which will provide a single, more consistent, query language for a wide variety of data types while following a similar spirit to the BioJava 1.x *FeatureFilters*.

I have been involved in BioJava development since the pre-1.0 development which began

in late 1999. During this time, I have worked on many areas of the code base, although I have a specific interest in handling of annotated sequence data, and database interfaces such as the BioSQL [OBDA, <http://obda.open-bio.org/>] and Ensembl (see below) access modules. BioJava code was used extensively in implementing the methods described in this project, and also for the large number of one-off scripts and small programs which were needed to prepare data sets and to analyze and present the results.

### 1.5.2. Ensembl

Ensembl began as a project to carry out a first-pass automatic annotation of the human genome, and the scope has since expanded to cover a wide range of eukaryotic genomes, and provide sophisticated web-based interfaces for accessing and querying these resources [Hubbard *et al.* 2002, Clamp *et al.* 2003]. The most notable aspect of the Ensembl data is the high-quality set of gene predictions, which are produced by a hybrid method using a range of evidence type, aligned to the genome using the Genewise tool [Birney 1999] and *est2genome* [Mott 1997], and also some *ab initio* computational predictions. The gene-builder module, which combines these data and produces final predictions of gene structures, is considered to have a very low rate of overprediction, particularly in comparison with the purely *ab initio* methods. Ensembl predictions are all given ID numbers, starting ENSG for predicted genes, ENST for distinct transcriptions, ENSE for individual exons, and ENSP for protein products. These IDs are stable: if a gene predicted on the current assembly exactly matches one on a previous number, the ID is reused. For computational biologists working with large sets of predicted genes, ENS\* IDs can be a convenient tool, avoiding traditional arguments about gene nomenclature, and allowing newly predicted genes to be uniquely identified even if no nomenclature has yet been defined.

All Ensembl data – both the raw sequence and the results of the gene build and other analyses – is stored in a relational database. In the course of this project, I developed a module called *bj-ensembl*, which plugs into the main BioJava libraries and provides seamless access to

the core Ensembl databases using the standard BioJava interfaces. In the example below, only the first two lines of code are specific to programs working with an Ensembl database. All the querying code is applicable to other types of database, or even (given enough memory to load a whole genome!) objects loaded from normal flat files.

```
// Connect to an Ensembl database

Ensembl ens = new Ensembl(/* database details */);
SequenceDB chromosomes = ens.getChromosomes();

// Find all CpG islands in a particular region

Sequence chr = chromosomes.getSequence(22);
FeatureHolder cpgs = chr.filter(
    new FeatureFilter.And(
        new FeatureFilter.ByType("cpg"),
        new FeatureFilter.OverlapsLocation(
            new RangeLocation(20000000, 21000000)
        )
    )
);

// Find transcripts of a particular gene

FeatureHolder trans = chromosomes.filter(
    new FeatureFilter.ContainsAnnotation(
        Ensembl.TRANSCRIPT_GENEID,
        "ENSG00000135457"
    )
);
```

Ensembl data was used heavily throughout this project. It proved valuable both as a simple database containing genomic sequences in a more convenient form than flat files, and as a source of high-quality gene predictions.

Ensembl data is presented *via* a sophisticated web interface, which includes a graphical sequence viewer (contigview) as well as many different report pages. It is possible to add extra “tracks” of information to contigview displays using by publishing the data using the DAS protocol [Dowell *et al.* 2001]. I developed a BioJava-based DAS server called Dazzle



[Down and Pocock 2001]. I used Dazzle and contigview to visualize many results from this project in their genomic context.

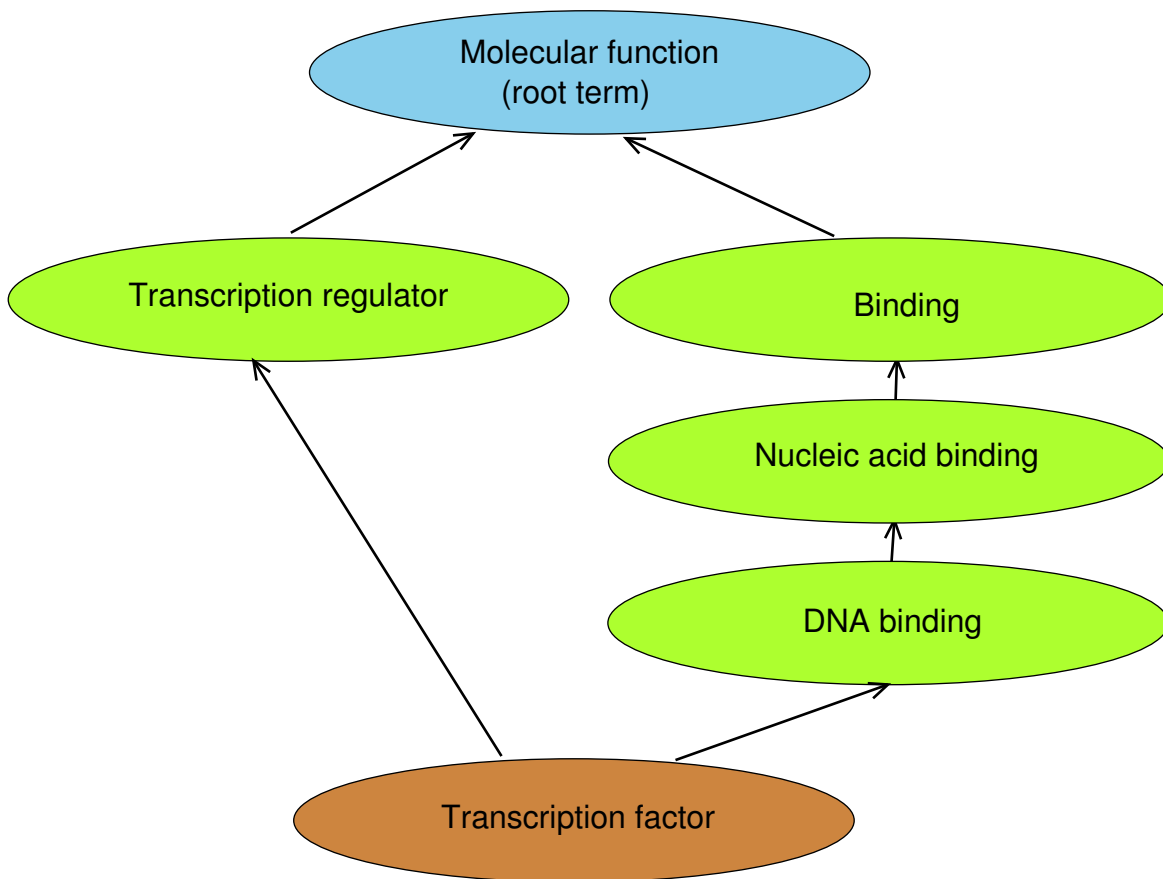
### 1.5.3. The Gene Ontology (GO)

The Gene Ontology is a controlled vocabulary for the annotation of genes [The Gene Ontology Consortium 2000]. It is divided into three portions:

- Molecular functions (e.g. “DNA binding”)
- Biological processes (e.g. “Transcription”)
- Cellular components (e.g. “Nucleus”)

Terms in GO are accompanied by human-readable definitions. These aim for rigour, with the hope that data annotated with GO terms by one group will be directly comparable with results from another group, who might be working on another species and come from a quite different background. For a computational biologist who wishes to ask questions along the lines of “are the genes in this (large) set more likely than average to perform some specific function”, statistics based on GO annotation are expected to be far more robust than alternatives, such as lexical analysis of keywords in human-written gene descriptions.

Rather than just offering a flat set of terms, GO also defines relationships between them. These are what mark out GO as a kind of ontology, rather than merely a controlled vocabulary. The relations in GO are either *is-a* or *part-of*. Both types are transitive, i.e. if A *is-a* B and B *is-a* C then A *is-a* C. This property is useful, since it means that it is reasonable to consider the set of all “descendants” of a given term (the transitive closure). This operation makes it possible to take data annotated with a complex vocabulary such as GO and slice it up at an arbitrary level. So a gene annotated with the molecular function “zinc-mediated transcriptional activator” will be pulled out by a specific query for that particular term, but also for more general queries, such as “transcriptional regulation” or “nucleic acid binding”. Both the relation types are directed,



**Figure 1.9.** Example of a directed acyclic graph of Gene Ontology terms.

so each relationship can be considered to link a parent and a child term. Each term can be in relation to more than one parent, so the ontology forms a directed acyclic graph, as illustrated in figure 1.9.

Of course, a vocabulary is not useful on its own, but becomes relevant when it is used to describe objects of interest. Some genome projects, such as those for *Drosophila* and fission yeast, now use GO terms in all their curated annotation. For mammalian genomes, the current state of the art in GO annotation comes from the GOA project [Camon *et al.* 2003]. The basis of this project is curated annotation of selected Swissprot and Interpro entries with GO terms. Other protein sequences are then annotated on the basis of similarity to entries with curated GO terms. This annotation has been picked up by the Ensembl project, who use it as the basis for GO annotation of their gene predictions. Whenever GO annotation is used in this project, this refers

to GOA results which have been mapped to Ensembl genes in this way.

## Chapter 2. Sparse Bayesian Learning

Supervised machine learning is a generic term for methods which take some set of labeled data – that is, items of data about which we have some special knowledge, such as whether or not a DNA sequence comes from a gene’s promoter region – and build some kind of model which can be used to predict the label values for additional, unseen data. This is quite different to unsupervised machine learning methods, which are used to detect patterns in sets of unlabeled data.

In principle, simply storing away (memorizing) the training data would count as a valid method of machine learning, but this is only likely to be of use if all the unseen data is very similar to pieces of data which have already been seen. Much more interesting are methods which can generalize – build a model which contains less information than the complete training set, but which still match the labeling on the supplied data. Learning methods with good generalization properties are intuitively more likely to correctly interpret unseen data which is significantly different from any of the pieces of training data, and practical experience throughout the history of machine learning has backed up this intuition. Depending on the exact techniques used, the internal model state of a machine learning system may be amenable to inspection or visualization by human users, who can recognize the generalizations made by the method, and perhaps gain some additional understanding of the problem in hand. Returning briefly to the question of classifying sequences (covered in much more detail in chapter 3), a memorization-type solution might simply encode the complete set of training data, but would probably only make trustworthy predictions for sequences which showed strong similarity along their full length to a sequence in the training set. A more general solution would be to identify some common short motifs or compositional biases which occur in promoter regions. This could yield a much more widely applicable model, and also one which would be of direct interest to

researchers wishing to understand transcription initiation.

It should be noted that generalizing learning machines are not automatically “transparent” in terms of being able to extract the rules which have been learned in a form which can be presented to humans. This criticism has been made in particular about machine learning systems based on neural network approaches. While “black box” prediction systems, when suitably validated, can be useful tools, it is always preferable to understand the basis for the predictions that are made. It is therefore worth specifically considering the issue of transparency when developing or choosing new machine learning technologies.

This chapter introduces one particular approach to supervised learning, called Sparse Bayesian Learning, which based on recent theoretical developments by MacKay and Tipping. Later sections discuss a specific “user oriented” implementation of the Sparse Bayesian Learning approach (as opposed to development implementations in simple test harnesses), which is available as a Java library routine, and can be applied to a wide range of real-world machine learning applications.

## 2.1. Generalized Linear Models (GLMs)

Generalized Linear Models are common mathematical devices, by which the value of a real function is represented as the weighted sum of a number of basis functions [McCullagh and Nelder 1983]. The general formulation is:

$$\eta(\mathbf{x}) = \sum_i \beta_i \phi_i(\mathbf{x}) + K \quad (2.1.1)$$

Where  $\beta$  represents a vector of weights,  $\phi$  is the set of basis functions, and  $K$  is a constant. The basis functions can be *any* real-valued function of  $\mathbf{x}$ . In the mathematical literature,  $\mathbf{x}$  is normally a vector, and a common choice of basis function is a hyperspherical Gaussian around some point in the appropriately-dimensioned space (commonly called the radial basis function). However,

since the GLM only refers to  $\phi(\mathbf{x})$ ,  $\mathbf{x}$  could actually be any type of data – including strings or even records of structured data – so long as a suitable family of basis functions can be defined. Analogous to the radial basis function, these functions will typically be simple functions of a distance metric from some point in a hypothetical “data space”.

It is common to refer to the high-dimensional space implied by a list of basis functions as feature space. The common theme of generalized linear modeling is to pick a projection from the natural data space, where the distribution of data may be non-linear (and potentially extremely complex), into a feature space where a linear model is a good fit to the data. Obviously, this means that the choice of feature space is important, and requires either problem-specific knowledge or an appropriate automatic method – this is discussed further in the next section.

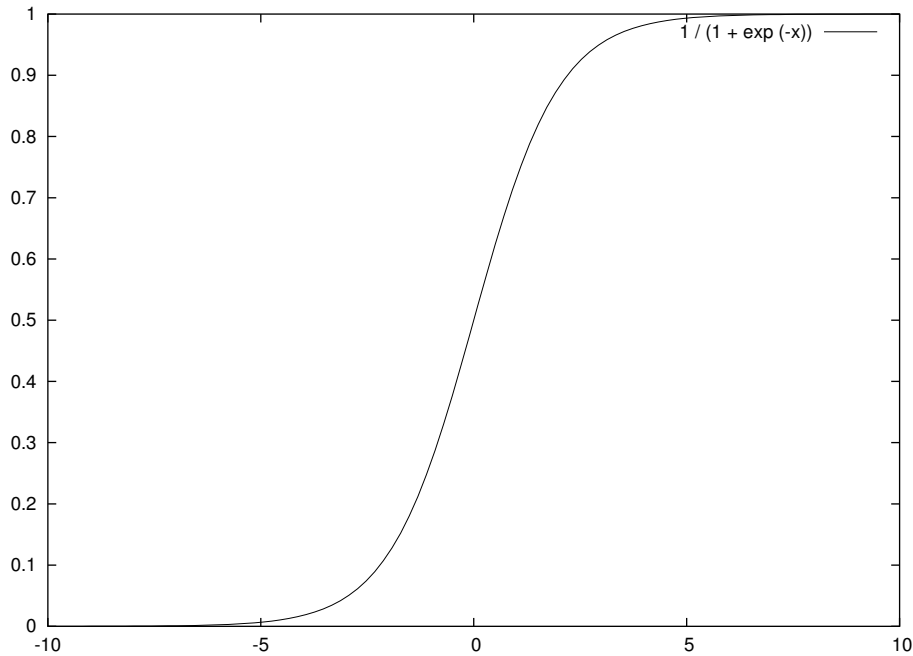
This basic form of a GLM is normally applied as a regression system – to estimate the value of a continuous function. However, for many machine learning tasks, it is more interesting to investigate class membership. As an example relevant to this thesis, a DNA sequence  $\mathbf{x}$  might belong to the class P (promoter) or not-P. GLMs are applied to classification problems using a link function. For binary classification tasks, the logistic function (figure 2.1) is a common choice:

$$\pi = \sigma(\eta) = \frac{1}{1 + e^{-\eta}} \quad (2.1.2)$$

This function has the desirable property that  $0 \leq \pi \leq 1$ . By fitting appropriate values for  $\beta$ ,  $\pi$  is an estimate of  $p(\mathbf{x} \in P)$ , our posterior belief that  $\mathbf{x}$  is a promoter given the learned model.

It is also possible to use multiple GLMs to separate data into more than two classes. In this case, each class is modeled by its own set of weights, and the probability of a data point being a member of class  $c$  is given by the multinomial link function:

$$\pi_c = \frac{e^{\eta_c}}{\sum_{c'=1}^C e^{\eta_{c'}}} \quad (2.1.3)$$



**Figure 2.1.** Plot of the logistic function (equation 2.1.2).

This multi-GLM formulation could, for instance, be used to build a classifier capable of distinguishing between promoter sequences which are active in several different biological environments.

One point to note about GLM classifiers is that while, given suitable values of  $\beta$ , they can serve as probabilistic models, they are not “end-to-end” probabilistic in the sense of some types of hidden Markov model, where every parameter in the model can be interpreted a probability. The values of the individual basis functions need not be probabilities themselves, and it is the responsibility of the training algorithm to fit a model where the output value has a probabilistic interpretation.

In its simplest form, training a GLM simply means picking a set of values for  $\beta$  and  $K$ . For regression questions, the best-known algorithm for this is the least-squares method [Lawson and Hanson 1995], an analytical approach which simply minimizes the square of the error at each point of training data. This method is not directly applicable to classification problems, but comparable methods do exist: for example, iterated reweighted least squares

[Nabney 1999]. However, all these methods are only useful if the data has already been transformed into a suitable feature space.

## 2.2. Feature selection using pruning priors

The utility of machine learning algorithms is severely limited if users have to apply large amounts of domain-specific knowledge to project the data into a small, meaningful, feature space. Therefore, it is interesting to consider methods which can work in very high-dimensional feature spaces and automatically select smaller, informative subspaces. Methods which combine selection of features with learning an optimal set of parameters are called sparse learning algorithms. The classic example of a sparse learning algorithm, and to date the most widely used, is the Support Vector Machine (SVM) [Schölkopf *et al.* (eds.) 1999], which can learn one particular class of generalized linear model in a sparse manner, often reducing a candidate feature space with thousands of basis functions down to just a handful. However, the restrictions on problems which can be solved with support vector machines are rather stringent: instead of supplying an arbitrary collection of basis functions, the user must supply a single kernel function  $K(\mathbf{x}, \mathbf{y})$ , whose value is the inner (dot) product of  $\mathbf{x}$  and  $\mathbf{y}$  when projected into the desired feature space. The basis functions of the learned GLMs are always of the form  $\phi_i(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}_i)$ , where  $\mathbf{x}_i$  is one of the examples from the training data set. An alternative way of looking at this is that the algorithm works on the Gram matrix: a square matrix  $\mathbf{A}$  where:

$$\mathbf{A}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.2.1}$$

In addition, Mercer's condition, which forms part of the derivation of the support vector machine algorithm, states the kernel function – and thus the Gram matrix – must be positive definite, i.e.:



$$\mathbf{z}^\top \mathbf{A} \mathbf{z} \geq 0 \tag{2.2.2}$$

for any vector,  $\mathbf{z}$ , with a length matching the edge size of the square matrix. Proving that an arbitrary function satisfies this condition can be a demanding requirement, and presents a barrier to simply plugging new types of kernel into the SVM framework.

While SVMs have proved to be effective tools for some biological problems (see, for example, [Furey *et al.* 2000]), the constraints on the forms of models which can be learned can be problematic. Most well-known kernel functions are applicable only to numerical data. Some effort has been made to develop kernel functions which can be applied to alternative data types, such as strings [Jaakkola and Haussler 1999] and graphs [Kondor and Lafferty, 2002], but the requirement to prove that the function used is positive definite makes this problematic, and prevents people without strong mathematical backgrounds from developing new SVM applications. Perhaps more seriously, the kernel view of GLMs does not allow models to be learned with basis functions that pick truly arbitrary sub-spaces of the training data space (for example, just considering dimensions 5, 8, and 10 of vector data, or only one region of a long string), since only the set of basis functions implied by the training data are available. Very recently, a “kernel-like” learning system has been developed which adjusts an additional set of scaling variables to select informative dimensions [Krishnapuram *et al.* 2003], but unlike the approach described below this is only applicable to vector data.

For these reasons, it is interesting to consider alternative forms of learning which can be used to train arbitrary GLMs while achieving sparsity comparable or better than that of SVMs. One extremely promising approach which fulfills these requirements is the Relevance Vector Machine (RVM) [Tipping 2000]. This was originally presented as a direct (and competitive, on the basis of performance on standard machine learning test problems) alternative to the SVM, with basis functions implied from a single kernel function and a set of training data. However, unlike the SVM there is no technical reason why this formulation is necessary, and it is entirely

feasible to implement an RVM-like learning method which uses arbitrary basis functions.

Briefly, the RVM is a Bayesian probabilistic view of training a GLM as defined by equation 2.1.1. In other words, the question is, given a list of training data,  $X$ , and a corresponding list of labels or expected outcomes,  $t$ , to find probable values of the weights vector,  $\beta$  which make the model outputs match the supplied labeling.

The basis of all Bayesian statistics is Bayes' theorem:

$$P(a|b) = \frac{P(a)P(b|a)}{P(b)} \quad (2.2.3)$$

This says that, given some prior knowledge of the probability of  $a$  ( $P(a)$ ), and the conditional probability of  $b$  given  $a$  (the likelihood,  $P(b|a)$ ), it is possible to calculate the probability of  $a$  given  $b$ . In cases where there is no prior knowledge whatsoever, a flat (non-informative) prior distribution can, of course, be specified. The evidence term,  $P(b)$ , is generally treated simply as a normalizing constant. Bayes' theorem supports a modeling view of statistics and learning: if  $a$  is the parameters of the model,  $b$  is the training data, and the likelihood function  $P(b|a)$  encapsulates the logic of the model, Bayes' theorem offers us a probability distribution over possible values for the model parameters.

For a two-way classification model, with training data labeled as either positive ( $t_n = 1$ ) or negative ( $t_n = 0$ ), and treating each element of the training set independently, a likelihood function – the probability of that set of labeled data given a particular set of model parameters – can be written as:

$$P(t | X, \beta) = \prod_{n=1}^N \sigma(\eta_n)^{t_n} (1 - \sigma(\eta_n))^{1-t_n} \quad (2.2.4)$$

where  $\eta_n$  is the linear model output for the  $n^{\text{th}}$  example in the training set. It is possible to write comparable probabilistic formulations for regression problems by specifying some probability

distribution for errors relative to the model output, then proceed in an analogous fashion, but regression problems fall outside the scope of this thesis.

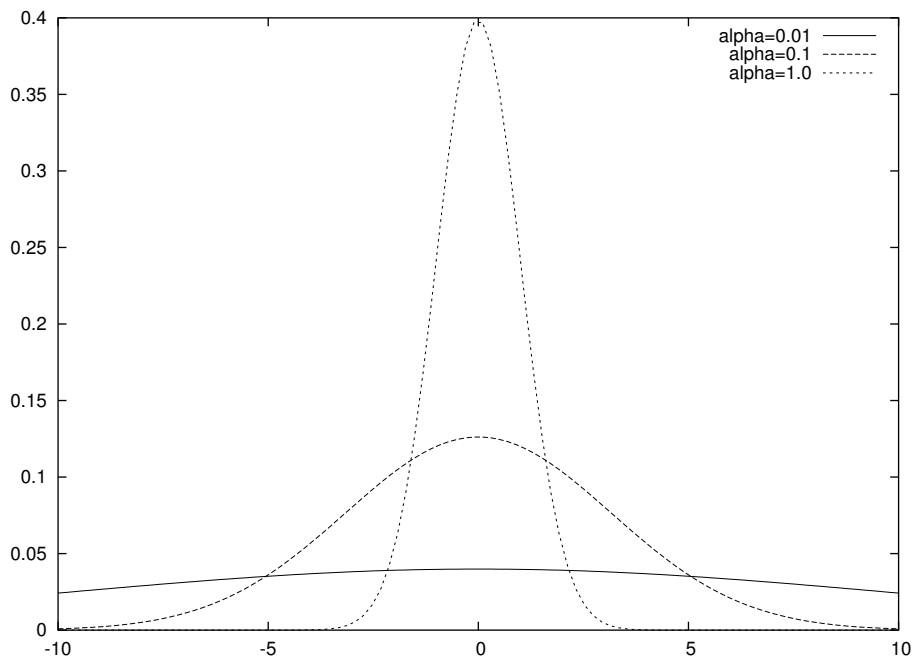
The second part of the Bayesian inference process is the prior distribution over the parameters being inferred. In general, the preferred choice is a non-informative prior, implying that before the inference operation we have no knowledge of what the parameters are likely to be. However, in this case there *is* a prior preference: if possible, we want to learn sparse models. This is encoded in the RVM by using a more sophisticated prior. The basic prior is an independent Gaussian distribution,  $\mathcal{G}$ , over the weight of each basis function:

$$P(\beta) = \prod_i \mathcal{G}(\beta_i | 0, \alpha_i^{-1}) \quad (2.2.5)$$

The “RVM trick” is to define the inverse variances of these Gaussian distributions,  $\alpha$ , as variables, and to infer their values as well. It is therefore, of course, necessary to provide an additional hyperprior over values of these priors. For the hyperprior, a conventional choice of non-informative prior is used: a very broad gamma distribution. Since the parameters of this are chosen such that the distribution is essentially flat over a wide range of “reasonable” values of  $\alpha$ , the exact choice of function is in fact irrelevant except for issues of computational convenience. This form of prior is known as an automatic relevance determination (ARD) prior, and was first proposed (in a somewhat different application, relating to the training of neural networks) in [Mackay 1994].

The inclusion of an ARD prior has been described as an Occam term, since it rewards simplicity: when the  $\alpha$  parameter tends to infinity, the probability of  $\beta$  values close to zero becomes extremely high, as can be seen by extrapolating figure 2.2. So for a basis function which cannot directly contribute to the likelihood of the data, the joint likelihood of the data and the  $\beta$  parameters is maximized by setting  $\alpha$  to a large value and  $\beta$  to zero. As we shall

see, inferring likely parameters for this model requires an iterative process. After a number of cycles, some of the  $\alpha$  values become large. That means that the corresponding weight parameter,  $\beta_i$ , is well-defined with a value extremely close to zero. When  $\alpha$  exceeds some large threshold, we can assume that the corresponding dimension of feature space is irrelevant, and not making a substantial contribution to the calculations. At this point, it is reasonable for practical implementations of the algorithm to simply drop that dimension from further calculations. In this way, sparse models are obtained. In addition, the computational cost of each iteration falls with the number of dimensions under consideration.



**Figure 2.2.** Plots of the Gaussian distribution,  $\mathcal{G}(\beta_i | 0, \alpha_i^{-1})$ , for various values of  $\alpha$ .

Given this probabilistic view of the problem, Bayes' theorem provides us with the distribution of probable values for the weights:

$$P(\beta | X, t) = \frac{\int_{\alpha} P(\beta | \alpha) P(\alpha) P(X, t | \beta)}{P(X, t)} \quad (2.2.6)$$

Note that the  $\alpha$  parameters are hidden variables of the model, and not of direct interest. Therefore, this formula marginalizes  $\alpha$  by integrating over all possible values. This is the

standard Bayesian approach to handling all parameters whose values are not specifically required in the inference.

Unfortunately, like most Bayesian problems, formula 2.2.6 is not directly tractable, due to problems evaluating the evidence term,  $P(X, t)$ . Therefore, it is necessary to take one of a variety of approximate methods in order to obtain likely values for  $\beta$  and  $\alpha$ . The simplest approaches to solve such problems are Monte-Carlo methods, which make it possible to draw samples from the posterior distribution  $P(\beta | X, t)$ . The basic Monte-Carlo method is the Metropolis-Hastings algorithm [MacKay 2003]. The principle here is that if we wish to sample from the probability distribution  $P(x)$ , we perform a random walk within this distribution, then take some of the states which are visited as samples from the distribution. So long as a sufficiently large number of steps are made between samples, they will be independent samples from  $P(x)$ . To perform the random walk, it is necessary to offer a proposal distribution,  $Q(x' | x)$  which suggests states to try next conditioned on the current state,  $x$ , and to be able to easily sample from this distribution. For each step, a new state,  $x'$  is proposed by drawing a sample from  $Q(x' | x)$ , and the following quantity is calculated:

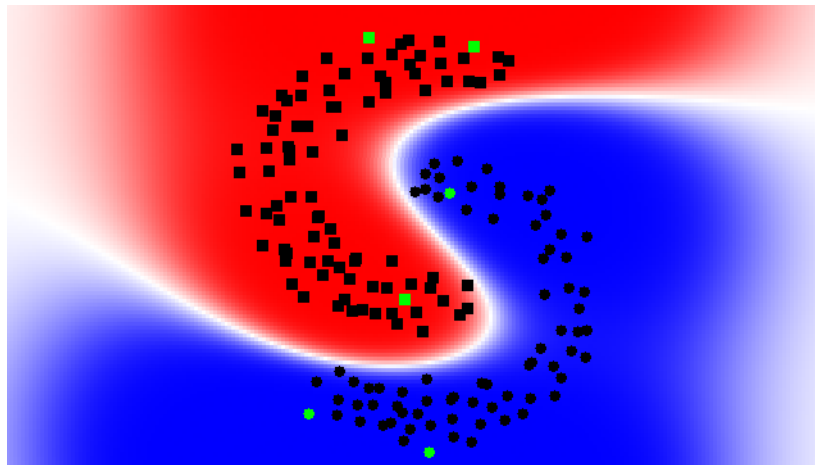
$$\alpha = \frac{P(x')}{P(x)} \frac{Q(x | x')}{Q(x' | x)} \quad (2.2.7)$$

If  $\alpha \geq 1$ , the new state is always accepted. Otherwise, the new state is accepted with probability  $\alpha$ . There are two important points here: firstly, the exact distribution chosen for  $Q$  is not important: the second term of equation 2.2.7 counteracts any bias in the set of states which are proposed. However, a bad proposal distribution may mean that very few proposals are accepted and the random walk proceeds very slowly. Secondly, since it is only necessary to calculate the ratio  $P(x')/P(x)$ , it is not necessary to calculate any constant terms in  $P$ , so the evidence term of distributions such as 2.2.6 can be neglected.

I wrote a naive implementation of the Relevance Vector Machine algorithm which used the basic Metropolis-Hastings algorithm. This was able to solve simple problems, such as that

shown in figure 2.3, but required several minutes of processor time. Figure 2.3 also shows off an attractive side effect of using probabilistic classification methods: each prediction comes with a confidence level. While points close to the training data are shown in deep red or blue, indicating close to 100% confidence in the prediction, points that are some distance from training data of either class appear in paler shades, indicating a much lower confidence. Non-probabilistic methods such as support vector machines cannot provide this information.

Some methods exist for optimizing Monte-Carlo simulations: for example, Skilling's leapfrog [MacKay 2003]. For this project, I did not follow this course further, since the alternative method described below gave good performance and worked well on the problems I considered. However, further investigation of Monte Carlo RVM implementations might be interesting in the future, for cases where the variational solution cannot be applied.



**Figure 2.3.** Example of sparse Bayesian learning in Cartesian 2-space. The data points chosen as centers for the final set of basis functions are highlighted in green.

An alternative approach is to use free-form variational inference [MacKay 1995]. This is a recent approach to Bayesian inference whereby an intractable posterior distribution is approximated by alternative, simple, probability distributions. Analytical formulae can be obtained which minimize the difference between the true and approximating posterior distribution. A variational approximation of the RVM posterior distributions is given in [Bishop and Tipping 2000]. Like many variational inference solutions, this uses a factorisable

posterior distribution (i.e. independent distributions for all parameters). Although the variational approximation yields formulae to analytically determine the moments of one approximating distribution given values for the other, it is still necessary to iterate the process a number of times in order to fit all the parameters to the data. The variational RVM approach can be applied effectively to both regression and binary classification problems. However, for classification problems it relies on transforming the logistic function to a convex form, which allows a linear approximation to be made using Jensen's inequality. This appears not to be practical in the case of the multinomial function, therefore multi-class problems must still be solved using an alternative method such as Metropolis-Hastings.

Implementations of the RVM methodology which iteratively apply the variational estimators from [Bishop and Tipping 2000] are able to solve problems of the scale of figure 2.3 in a timescale of around 10 seconds on a typical desktop computer. Unlike Monte-Carlo models, it is possible to specify a clearly objective stopping criterion, halting iterations once the training process has “converged” (i.e. the moments of the approximating distributions no longer change significantly from one cycle to the next).

### **2.3. A pragmatic approach to handling large spaces**

In principle, we would like to provide as little prior knowledge as possible about which feature spaces to consider when solving a problem, instead allowing a sparse trainer such as the RVM to pick freely from a wide range of possible basis functions. Unfortunately, even when using the variational approximations to avoid the inefficiencies of a sampling strategy, computational costs become significant. The progressive simplification of the problem means that it is difficult to quantify the full computational complexity since the size of the matrices varies from cycle to cycle. However, scaling is quite substantially worse than linear, as can be seen in figure 2.5.

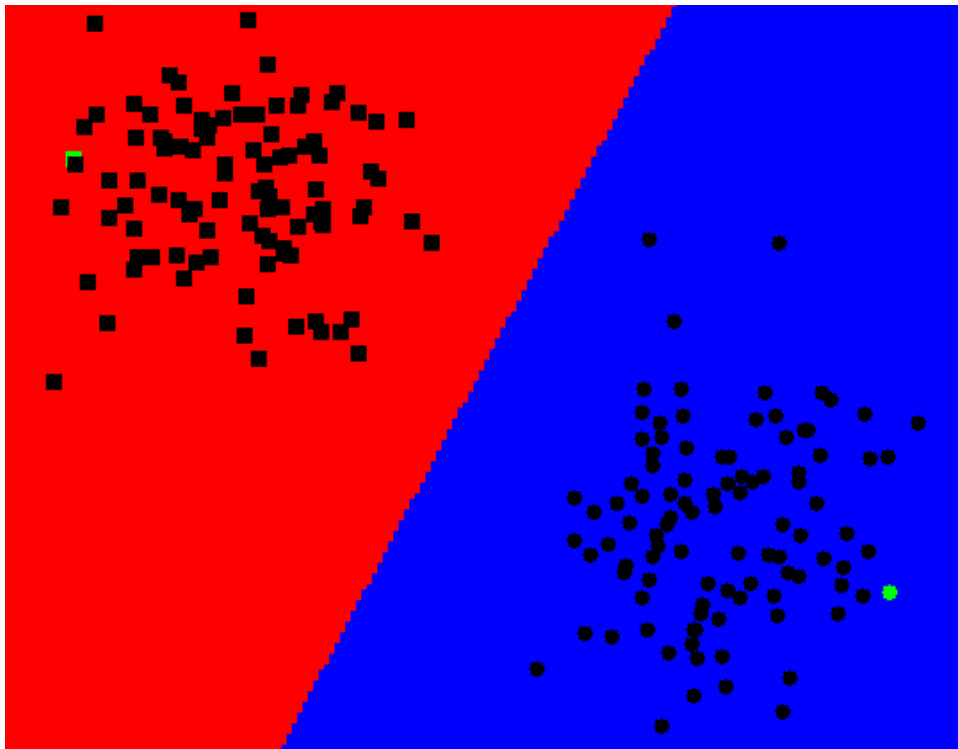
Here, a pragmatic approach to handling large sets of candidate basis functions is suggested: a working set is first initialized with a subset of basis functions, picked at random from the pool of candidates. The trainer runs as previously described, and as before some  $\alpha$  values increase to a level such that it is possible to remove the associated basis functions from further consideration. Once the size of the working set drops below a designated low water mark, additional basis functions are added from the pool. At this point, all  $\alpha$  and  $\beta$  values are reinitialized and training continues. In this way, the working set fluctuates between high and low water marks until the pool is exhausted, at which point the trainer continues to run until the weights and priors no longer change significantly between cycles (*i.e.* convergence) to give a complete model.

To evaluate the performance benefits of this method, I considered a simple classification problem in Cartesian 2-space. This involved two equal-sized classes of points sampled from two Gaussians, with a substantial space between them. Basis functions were generated in an “SVM-like” manner, with a radial basis function centred on each point in the training data. Thus, the size of the basis-value matrix of the sparse Bayesian trainer increases with the square of dataset size. As shown in figure 2.4, an optimal model requires only two basis functions, and gives 100% classification accuracy on the training data.

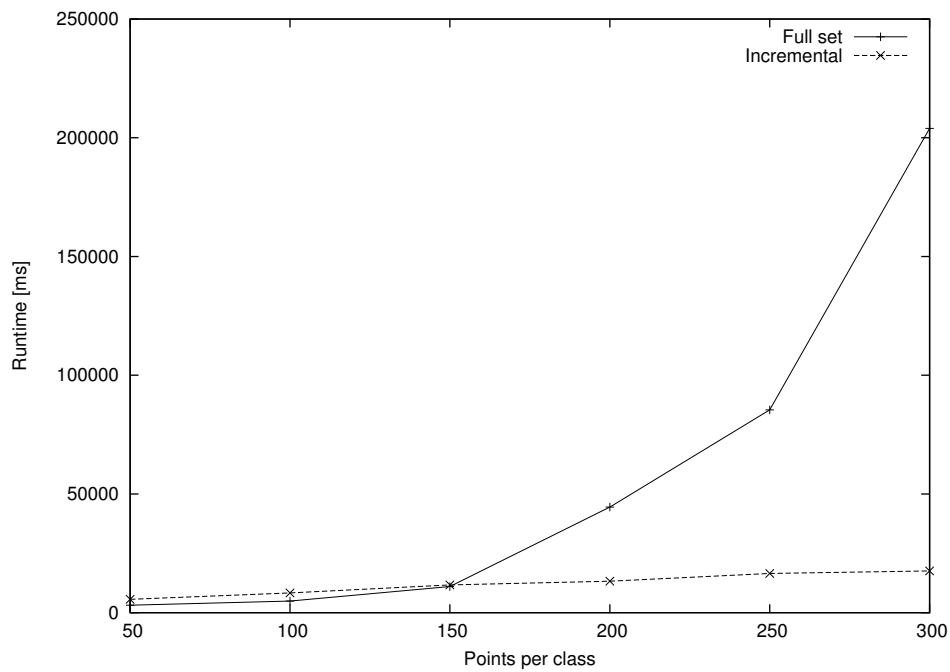
A range of problem sizes were tested, from 50 points in each class (*i.e.* 100 basis functions) to 300 points. I compared the basic full-set training method with the incremental method, picking a high water mark of 20 basis functions and a low water mark of 15. Timings are shown in figure 2.5. For small problems, the incremental training approach is in fact significantly slower, due to the increased number of cycles and the need to periodically restart the training process as more basis functions are added. However, the scalability of the method to large problems is substantially better.

It is interesting to note that, while in all cases all the training data was correctly classified, the full-set training runs with 300 points did not learn the simple 2-basis function model seen with smaller training sets, but instead converged before all unnecessary basis functions had





**Figure 2.4.** Example of a data set used for testing of training speed.



**Figure 2.5.** Training time vs. problem size for full-set and incremental sparse Bayesian learning.

been removed. However, the incremental trainer consistently learned a sparse model for the

same dataset with no complications. I believe that this indicates the limits of either the linear algebra routines used by the trainer, or the numerical precision of the hardware on which it was running. Errors caused by lack of precision (often described as numerical instability) are a common problem in numerical computation, and large linear algebra computations are a frequent source of trouble. By keeping the size of individual computations down, using the small-working-set variant of the RVM neatly sidesteps these problems and makes it possible to tackle large problems.

Incremental training does have one potential drawback: if the chosen working set sizes are insufficient to cover the actual complexity of the problem in hand, it is possible for the training process to become “stuck” in a state where adding extra basis functions could increase classification accuracy, but it is not possible to remove any functions from the current set. The implementation described here applies some simple checks to detect the stuck state and can, when appropriately configured, automatically increase the user-specified high and low watermarks, but this mechanism has not been intensively tested, so it is suggested that some care is taken in picking appropriate watermark values.

Having made the step toward the approach of gradually sifting through a large set of basis functions, it is only a small additional step to consider an implementation which generates new basis functions on the fly during the training process. In principle, this can allow exploration of infinite sets of basis functions. This is particularly feasible in cases where there are clear correlations between basis functions. For instance, a radial basis function with centre  $(10.0, 10.2)$  and variance 10 will have outputs that are highly correlated to a second radial basis function with centre  $(10.0, 10.0)$  and equal variance. This means that if the first function is found to be informative for modeling a particular problem, it is likely that the second will also be effective. Therefore, after some period of training, it is likely to be worthwhile proposing more candidate basis functions which are correlated to those currently in the working set.

## 2.4. A general-purpose Sparse Bayesian trainer

During this project, and especially the work described in chapter 3, the most useful form of sparse learning was the variational implementation of binary classification. I implemented this, and several other algorithms, for the Java 2 platform, as described in chapter 1. I used a number of APIs from the COLT library [Hoschek *et al.* 2000], which offer high-performance implementations of common linear algebra operations for Java programmers. Java code using COLT is often little more verbose than dedicated mathematical programming environments such as Matlab. Source code for the RVM library is available under the terms of the GNU lesser GPL on request from Thomas Down.

It is possible for programmers to make use of the trainer without detailed knowledge of the inference process and how it is implemented internally. Basis functions are provided by writing one or more implementations of the *BasisFunction* interface

```
package stats.glm;

public interface BasisFunction {
    /**
     * Return a feature value for a particular object.
     *
     * @throws ClassCastException if this BasisFunction cannot
     * be evaluated for an object of
     * this type.
     */
    public double evaluate(Object o);
}
```

These *BasisFunction* objects can in principle work with *any* object in the Java virtual machine. Obviously, it is the user's responsibility to ensure that the basis functions match the supplied training data.

Training data is supplied using the *SVMTarget* class, part of a support vector machine toolkit which was previously developed by the author, and is included in the BioJava library.

Reusing code in this way saved duplicated development effort, and makes the Sparse Bayesian trainer more accessible for people who already use BioJava.

The final part of the system is the *BasisSource* interface, which is, as its name suggests, a source of *BasisFunction* objects. For conventional use, the supplied *ListBasisSource* simply provides basis functions from a pre-defined list, until it is exhausted. For training systems which use sampling approaches, developers will have to write their own implementation of *BasisSource*, providing the desired sampling moves.

All these elements are presented to the training code, which returns a model object including the chosen set of basis functions and their weights:

```
// Initialize

SVMTarget target = loadTrainingData();
BasisSource basisSource = new ListBasisSource(basisFunctions);
VRVMTrainer trainer = new VRVMTrainer();

// Train model

GLMClassificationModel model;
model = trainer.trainClassification(
    target,
    basisSource,
    new SimpleTrainingListener()
);

// Print results for test data

for (Iterator i = testData.iterator(); i.hasNext(); ) {
    Object testDataPoint = i.next();
    System.out.println(
        testDataPoint.toString() +
        " -> " +
        model.positiveProbability(testDataPoint)
    );
}
```

The final parameter in the call to the trainer is an implementation of a simple callback interface which receives status notifications during the training process. This is especially useful if the

trainer is to be embedded in a user-oriented application, since it makes it easy to hook up some graphical feedback of training progress.

## 2.5. Sparse Bayesian Learning Discussion

Sparse Bayesian Learning using the automatic relevance determination prior offers a convenient and principled approach towards developing machine learning systems which actively penalize unnecessary complexity in the model, and consequently build the simplest model which can still give a good fit the training data. While the connection is not absolutely straightforward, both intuition and experience tell us that sparsity usually translates to making generalizations about the supplied data (rather than overfitting or memorizing it) and consequently making good predictions from unseen pieces of data – a valuable property in any learning system. Moreover, since the model output is a probability value, it is possible to distinguish between cases where enough information is available to make a confident prediction and cases where a piece of data is only marginally more likely to fall in one class than the other. It is principled to apply an arbitrary threshold to these scores and, for example, consider only those predictions with 99% confidence.

Throughout this chapter I have concentrated on one specific form of Sparse Bayesian Learning – binary classification GLMs. This was the form which I found directly useful in the course of this project. It is possible to apply the Automatic Relevance Determination principle of sparse learning to other problems – for example multi-way classifiers, but in some cases it is no longer possible to use the elegant analytical approximations obtained by variational inference. Developing these forms of learning into practical methods suitable for widespread application will mean investigating alternative approaches to the inference problem. Optimizations which allow independent samples to be drawn from a Monte Carlo simulation with less computation than is needed for the basic Metropolis-Hastings approach would seem to be a profitable direction to explore.

I have implemented and tested a practical implementation of Sparse Bayesian classification. This is freely available as a Java library. Using the simple interfaces described in this chapter, it is possible to write new types of basis function, and thus apply the trainer to new types of data, without requiring substantial knowledge of variational inference or the internal implementation details of the method. This library is used without further description as the “learning engine” driving the sequence analysis methods described in chapters 3 and 4. The results in these chapter additionally provide validation of the method, and show that it is extremely effective when applied to non-vector data (genomic contexts and genomic sequence fragments respectively). They also show that, subject to the choice of basis functions, RVM-based learning methods can be quite transparent, with models which can be viewed and related back to biological processes. Other researchers have applied exactly the same library to quite different problems. For example in [Pocock 2001], the RVM trainer was used to classify microarray gene expression data from samples taken before and after treatment of a tumour with doxyrubicin. The basis functions chosen indicated a small set of representative genes whose expression levels changed dramatically and consistently when cells were treated with this drug. This application does, however, also show off one limitation of sparse learning systems: to discover the complete set of genes implicated in the process, rather than a minimal informative subset, it is necessary to post-process the results using a clustering algorithm.

# Chapter 3. Modeling of transcription start sites

My initial objective here was to build a system which can predict transcription start sites in bulk genomic DNA sequences. For this problem, selectivity is vitally important: of all the possible positions in the human genome – around 3 billion in total – only a tiny proportion are expected to be actual transcription start sites, so even a low rate of overprediction, taken on a position-by-position basis, could still lead to a extremely large number of false positives across the genome as a whole, giving results which are of little value either to laboratory researchers wishing to perform directed promoter-mapping experiments, or as a starting point for performing other computational analyses.

There have been a number of attempts to develop computational methods of promoter prediction (see page 17). Older methods, predating the availability of large volumes of vertebrate genome sequence, tended to suffer from limited selectivity [Fickett and Hatzigeorgiou 1997, Scherf *et al.* 2000]. A more recent method, PromoterInspector [Scherf *et al.* 2000], aimed to provide a rather higher degree of selectivity, but it makes predictions for regions of the genome, giving an approximate area for the promoter but little specific information about the actual transcription start site. This program also has rather limited use for genome annotators, since it is a proprietary product. While a free web-based interface exist, users are only allowed to run a small number of analyses each month.

In order to develop a new prediction method, I wished to build a realistic model, preferably probabilistic in nature, of the sequence around transcription start sites. As well as providing a valuable predictive tool, using suitable machine learning methods to build such a model can also provide additional insights into the structures being investigated. Modeling approaches

are popular in sequence bioinformatics: in particular, Hidden Markov Models (HMMs) are widely used [Durbin *et al.* 1998], for example in the field of gene prediction where models such as Genscan represent the state of the art [Burge and Karlin, 1997]. In simple terms, HMMs can be viewed as two components. Firstly, the model architecture consists of a set of states and valid transitions between them. Secondly, a parameterization gives actual transition and emission probabilities. Given a specific architecture, the Baum-Welch optimization method offers a straightforward algorithm for finding a parameterization to optimally fit a given dataset, but learning architectures is much harder. HMMs such as the Genscan gene model have rather complex architectures, dedicated to a particular task and consciously designed and tested for this purpose by the method's developers. A less specialized approach is offered by profile HMMs, such as those used to build the Pfam protein family database [Bateman *et al.* 2002]. These have architectures built from simple repeating units. The Pfam models are built from multiple sequence alignments, and the “backbone” of states in the profile HMM represent the consensus of this alignment. This approach works very well for protein families, where all the members are evolutionarily related to one another, generally *via* point mutations and relatively small insertions and deletions, but profile models cannot be considered as a truly general approach to sequence analysis.

In this case, I decided that basic HMM methods were not an optimal approach for modeling promoters. Based on existing knowledge of transcription initiation (see chapter 1), promoters appear to be loosely-connected sets of motifs, rather than evolutionarily-related variants on a single theme, which suggests that Pfam-like profile models are not applicable. Other forms of HMM could potentially be designed which modeled motifs individually, and left some flexibility in their positions, and indeed steps have been taken in this direction by the Meta-MEME program [Grundy *et al.* 1997]. However, it is difficult, for example, to model the case of two motifs which must occur together, but in either order (and might perhaps even overlap). Moreover, it is likely that any non-trivial model architectures would have to be built either by hand or using heuristic methods, which potentially constrains the range of architectures which could viably



be explored.

I wished to search for interesting signals in as unconstrained a fashion as possible. I therefore set the following requirements for a new method, to be used for modeling transcription start sites (and hopefully extensible to other significant sites in biological sequence data):

- The model should be applicable to individual points (in their context) in a sequence, making accurate location of point features such as transcription start sites possible.
- The model should be modular, and built up from specific signals, rather than simply treating a whole sequence as a monolithic entity (as a profile HMM does).
- Any “architectural” aspects of the model must be learned automatically from supplied training data, preferably in the same process as the learning of individual signals, rather than a two-step process like Meta-MEME.

This set of requirements was not directly met by any existing methods, so I began the development of the Eponine Anchored Sequence method, described below.

### **3.1. The Eponine Anchored Sequence (EAS) model**

The Eponine Anchored Sequence model is a new approach to probabilistic sequence analysis which is capable of learning a complex overall model architecture and a set of small-scale sequence features in a single process. Mathematically, it can be represented as a generalized linear model, as described in the introduction to chapter 2.

EAS is a classification model which is designed to be applied to genomic contexts – that is, individual points within a large genome. In practice, genomic contexts are presented to the model as a large piece of sequence data and an integer defining the anchor point under consideration. When searching for features in bulk sequence, the same sequence is presented many times while scanning the anchor point along its length. The basic element of an EAS model is the positioned

constraint (hereafter, PC). This consists of:

- A preferred sequence motif, defined as a DNA position-weight matrix (page 1.4.1). Briefly, this is a list of columns, each defining one base of the motif, represented by a probability distribution over the DNA alphabet.
- A probability distribution over integer offsets relative to the anchor point, which defines the expected localization of the motif. In the work presented here, these distributions were always discretized Gaussians (*i.e.* the result of integrating the Gaussian probability density function over unit intervals). Gaussians were chosen because of their familiarity, and a smooth shape that made Gaussian-based models less prone to overfitting than functions with abrupt changes, such as square waves. However, any distribution over integers could in principle be used here.

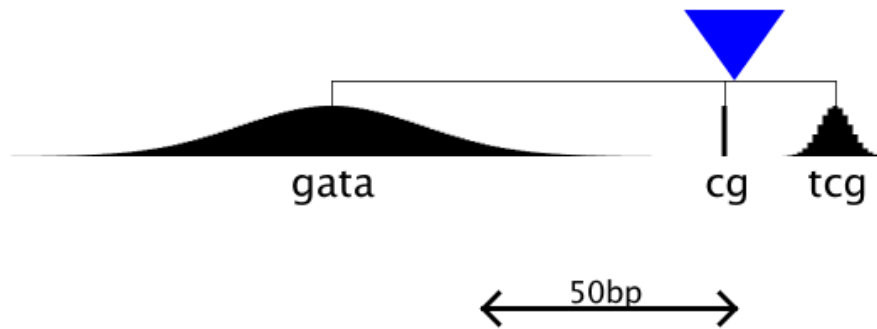
To obtain a score for a PC on a given genomic context, the program scans over all positions in the sequence which are assigned a non-infinitesimal probability by the chosen position distribution. For each position, the probability of the sequence motif starting at that position being emitted by the chosen weight matrix is evaluated. The final score is given by:

$$\phi(C) = \frac{\log\left(\sum_{i=-\infty}^{\infty} P(i) \vec{W}(C, i)\right)}{|W|} \quad (3.1.1)$$

where  $C$  is a genomic context,  $P$  is a position distribution, and  $\vec{W}(C, i)$  is a DNA weight matrix probability for offset  $i$  relative to the anchor point of  $C$ . Note the division by  $|W|$ , the number of columns in the weight matrix (*i.e.* the length of the sequence motif which it defines). This is important, since this method allow motifs with a wide range of lengths – with the trainer implementation described here, the length varies between 2 and 20 columns. The RVM trainer has a weak bias towards selecting basis function with higher absolute magnitudes. Normalizing the scores allows unbiased selection between motifs of different lengths. It is somewhat analogous to the whitening process often used to pre-process data for SVM classifiers, where all

the training vectors are normalized to constant length [Schölkopf *et al.* (eds.) 1999].

A single PC describes an individual sequence motif and its relationship to a point in a sequence, but a set of them can be combined to describe more complex structures. Figure 3.1 shows a schematic of a model combining three positioned constraints (note that in this schematic form, which is used throughout this chapter, the weight matrices are represented by single consensus sequences, which show the most likely symbol at each position in the motif). If the final output score is defined as a weighted sum of individual PC scores, the combined model is a generalized linear model over genomic contexts, with the PCs as basis functions. Therefore, it is possible use the sparse Bayesian learning methods from chapter 2 to reduce a large set of candidate PCs down to a sparse model containing a small, informative subset.



**Figure 3.1.** Example schematic architecture of an Eponine Anchored Sequence model.

There are several points to note about this approach to sequence modeling:

- While this description of the model architecture emphasizes detection of single, well-defined “words” in the sequence, since the overall PC score is based on a sum of weight matrix scores across a region, it is also possible to represent general compositional biases of a region by picking a PC with a short weight matrix and a very broad position distribution.
- It is quite acceptable for the distributions of two or more PCs to overlap. Since arbitrary weights are assigned in the training process to fit the model to the training data, any issues

with double-counting of a particular piece of information are corrected automatically

- While in this work, the models consist purely of PCs of the form defined above, it is possible to include completely different forms of basis function in the model. Once again, the training process will weight different types of evidence appropriately.

One limitation of the basic EAS model is that it is not able to capture interactions between pairs of motifs. Consider two motifs: A is found in the range [50:100] relative to an anchor point, while B is found in the range [30:80]. However, the spacing between the two motifs is much more conserved: always in the range [18:22]. An EAS model can capture the two motifs, and their broad position distributions, but misses the additional information in the covariance of the two motifs' positions. Including this information while maintaining the useful property of each motif being a single basis function would break the restrictions on generalized linear models, and therefore prohibit the use of the closed-form training algorithm from chapter 2. A possible solution would be to use more complex basis functions, each of which represented a small “scaffold” of several motifs, with particular spacings between them. This makes the space of possible basis functions far more complex. This makes the training procedure substantially more complex, and was not found to be helpful for the problem considered. Scaffolds *are* used in the alternative model described in chapter 4.

### 3.1.1. Learning EAS models

The space of potentially interesting PCs is extremely large. Even taking a highly simplified view, restricting the constraints to simple motifs rather than weight matrices and the position distributions to Gaussians of a constant width, there are over one million PCs representing six-base motifs with distributions centred at positions in the range [-250:50] relative to the anchor point. This space is too large to search exhaustively. Of course, in practice the EAS framework allows for an infinite (limited only by numerical precision on the probability values) number of PCs. Fortunately, as in the case of radial basis functions discussed in the chapter 2,

the space of possible basis functions is highly correlated. Making a small change to, say, one of the probabilities in a weight matrix will give a second PC whose output on a given sequence is correlated with the first. For this reason, exploring regions of a conceptual “PC-space” in the neighborhood of constraints which have already proved to be informative is likely to reveal even more informative constraints.

Taking advantage of these correlations, EAS models were trained using the two-class sparse Bayesian GLM trainer described in chapter 2, with a sampling strategy to create new basis functions. When the size of the working set fell below the low water mark, the trainer selected sampling strategies at random from the following set:

- Constructing a new PC, not based on the current set. This is performed by the following algorithm:
  - i. First, select a context at random from the training set (either positive or negative, without bias).
  - ii. Pick some point relative to that context’s anchor point.
  - iii. From that point, take a sequence motif of between 3 and 6 bases in length, and construct a weight matrix which optimally matches that consensus sequence, but includes some degree of uncertainty.
  - iv. Construct a PC using the newly selected weight matrix with a Gaussian position distribution of random width, centred at the position at which the motif was originally found.

Obviously, PCs selected in this way will strongly match the training example from which they were originally derived. This is closely analogous to selecting radial basis functions centred on points in the training data set, as used in the examples in chapter 2.

- Selecting an existing PC and adjusting the emission spectrum of one column of its

weight matrix, by sampling from a tightly-focused Dirichlet distribution centered on the current values.

- Adding an extra column to either the start or the end of an existing weight matrix, up to a maximum number of columns (in this case 20).
- Removing the start or end column from an existing weight matrix, down to a minimum of two columns
- Adjusting the width parameter of a Gaussian position distribution
- Adjust the centre position for a Gaussian position distribution

For an initial period of 200 cycles, only the first sampling rule (creation of novel basis functions) was used. After this point, the full range of sampling strategies were available, and training proceeded by a mixture of sampling and introduction of novel PCs.

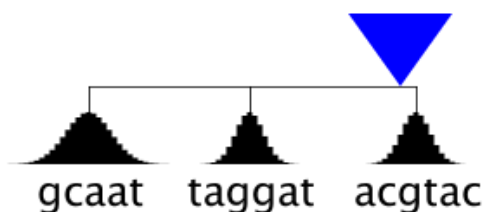
### **3.1.2. Implementation and validation of EAS**

The EAS model system, and the various sampling rules described above, were implemented running on the Java 2 platform, using components from the BioJava library [The BioJava development group, <http://www.biojava.org/>] to load and manipulate sequence data and probability distributions. Training was performed using the Java variational RVM training library described in chapter 2.

Before commencing work on real datasets, I wished to validate the training mechanism to ensure that it could build viable classifiers, and that it would correctly recover known information from the training set. Therefore I constructed a synthetic dataset consisting of unbiased random sequence (i.e. a list of samples from a uniform distribution over the alphabet of DNA symbols) into which were inserted three motifs: “GCAAT”, “TAGGAT”, and “ACGTAC” with some variability (for each motif-instance, either zero, one, or two bases were changed relative to the consensus pattern), and some variability (5 bases in either direction) in the position

with respect to the anchor point. Clearly, this dataset provides a good target for modeling with the EAS framework, since its construction closely matches the principles which were assumed when designing the model. Therefore, this test is simply a validation of the implementation and training method, rather than confirming that EAS will be able to answer real biological questions. The training data consisted of 100 of these synthetic sequences as positive examples, and 100 unspiked unbiased random sequences as negative examples.

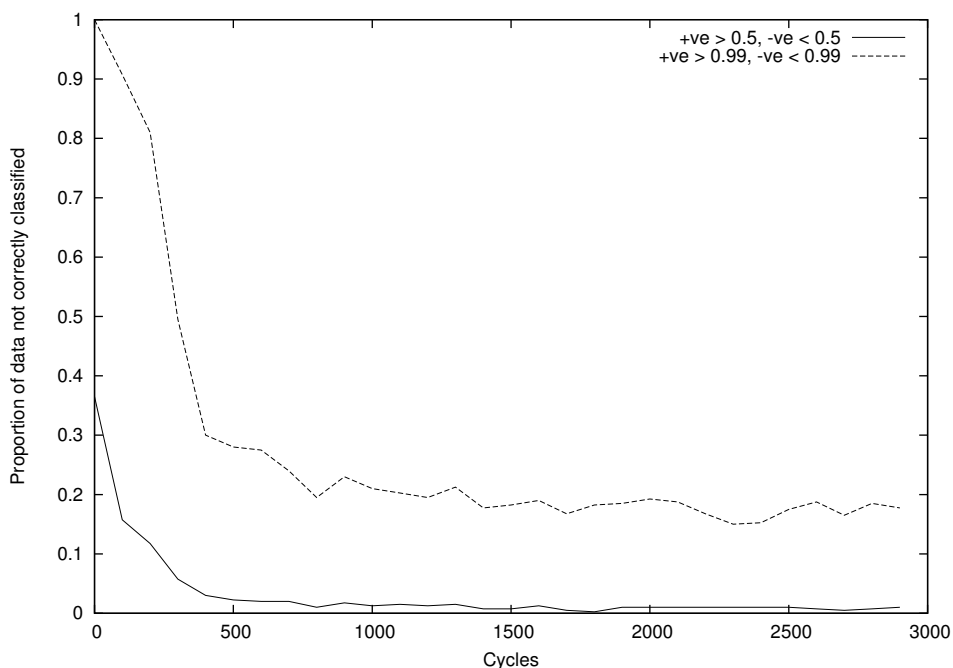
Models were trained for 3000 cycles using the VRVM training module described in chapter 2, in small working set model with a high water mark of 27 basis functions and a low water mark of 24. Monitoring of the trainer showed that the working set reached low water mark and was topped up every 5 to 10 cycles of training, with no cases of ‘stuck’ training (where no further basis functions can be removed). Checkpoints of the trainer state were stored every 100 cycles for later evaluation. The final model is shown in figure 3.2. After 3000 cycles, the three motifs used to build the synthetic dataset were recovered perfectly. This was generally quite reproducible, although in some cases the first or last base of one of the motifs would be missed. However, recovery of information from this training set was generally excellent, indicating that the RVM-based training approach can be applied to sequence data, and that the pragmatic sampling strategy is able to successfully train models of this complexity.



**Figure 3.2.** Schematic of an EAS model learned from the synthetic dataset, showing the three spiked motifs.

To monitor the progress of the training, and also to verify that overtraining did not reduce the predictive power of the model, I tested the checkpoint models produced while training the model in figure 3.2. Figure 3.3 shows two learning curves indicating the proportion of data which was correctly classified by the various checkpoint models. In the first curve, a threshold

probability of 0.5 is used – in other words, all sequences are assigned to either the positive or the negative class. By the end of the training process, almost all of the data is assigned to the correct class by this criterion. The second curve only counts positive test sequences as correctly classified if the model  $P(\text{positive}) > 0.99$ , and similarly  $P(\text{positive}) < 0.01$  for negatives. Any example with an intermediate model output is counted as unclassified. Early in the training process, few sequences receive such a high-confidence classification, but after 1000 cycles, around 80% of the data is correctly classified with this level of confidence. The proportion increases only very slightly after this. The sequences which are never classified with high-confidence predictions are mainly examples from the positive class with large numbers of mismatches in the spiked motifs. While both learning curves fluctuate slightly, there do not appear any substantial increases in the proportion not correctly classified. This suggests that overtraining is not a serious problem with this type of model.

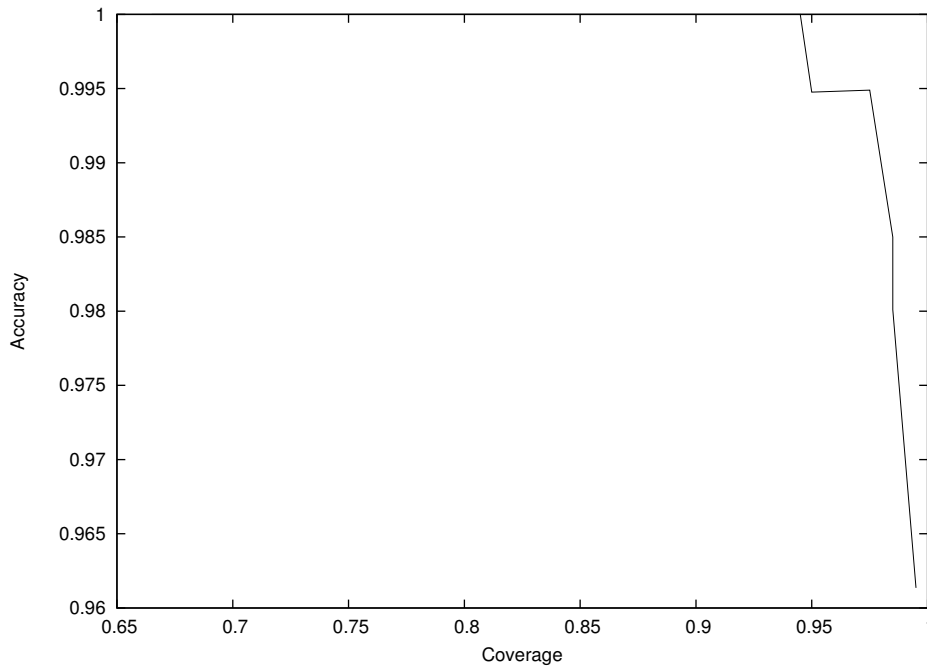


**Figure 3.3.** Learning curves for the training of an EAS model on the synthetic dataset.

Finally, I tested the predictive power of the learned model by plotting a receiver operating characteristic (accuracy vs. coverage) curve, shown in figure 3.4. Accuracy remains at 100% up to a coverage of 94% showing that, with a suitably chosen threshold, the model is a powerful



predictive tool when working with this kind of data.



**Figure 3.4.** Accuracy vs. coverage (ROC) curve for a model trained on the synthetic dataset.

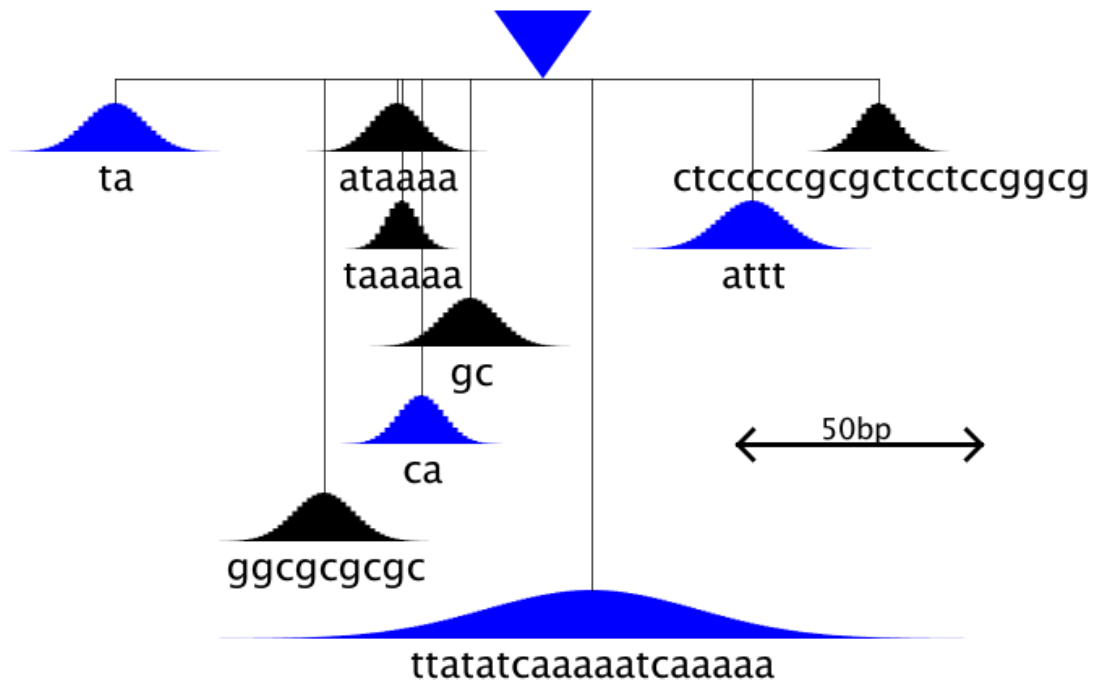
### 3.2. Training a transcription start site model

As discussed in chapter 1, sources of data about the precise location of vertebrate transcription start sites are relatively limited. For initial training, 389 mammalian promoters were extracted from release 62 of the EPD database [Périer *et al.* 2000]. I used the standard EPD web tools to download a non-redundant (defined by EPD as <50% sequence identity) set of sequences, extracting sequence data in the interval [-499:100] relative to the experimentally-mapped transcription start sites. Of these sequences, 50 were held aside for later use as an independent test set. This left 339 sequences which were used for training purposes. Around half of the EPD sequences were of human origin, but none used in the training set could be mapped to human chromosome 22 – this is important since chromosome 22 sequence was used as test data for some of the evaluation procedures used later in this chapter.

The trainer also requires a set of examples which are representative of the negative (non-promoter) class. For this purpose, I used fragments selected at random from the final introns of multi-intron genes on human chromosome 20. Introns early in a gene may often contain alternate transcription start sites [Laurinn *et al* 2000], but it seems unlikely that this will be the case for final introns, making them a good choice as representative non-promoter sequences. All negative and positive sequences were of the same length, and I picked an equal number of negative and positive sequences.

Models were trained using the procedure described previously, except that training was continued for 5000 cycles. Training typically took less than three hours on a typical personal computer (300MHz Pentium II CPU). Once again, there were no cases where the training process got stuck. A typical model from this process was selected as the EponineTSS\_1 model, and is shown in figure 3.5. Unlike the models trained from the synthetic data, the learned models contained PCs which were assigned negative weights in the generalized linear model. These are shown in blue in the schematic diagrams. The obvious interpretation of these is that the presence of a particular motif at a particular position actually makes it less likely that this is a promoter sequence. Whether these are genuine negative signals, or whether they are simply artifacts of the training process remains to be seen, although results later in the chapter appear to favor the latter possibility.

Inspecting this model, it can be seen that all learned PCs target sequences less than 150 bases upstream of the anchor point, despite the inclusion of 500 bases of upstream sequences in the training set: this model is targeting a rather compact area of sequence. This result also confirms that the sparsity properties of RVM learning can indeed be extended to the field of sequence analysis. There are two motifs with distribution means at -29 and -30 which are both A/T rich. I believe that these are related to the TATA box described in classical promoter literature [Bucher 1990]. These, or similar, motifs were detected at around this position in all models inspected, with the presence of two A/T-rich PCs with strongly overlapping distributions occurring very frequently. Most of the other PCs in the model are C/G-rich (note several



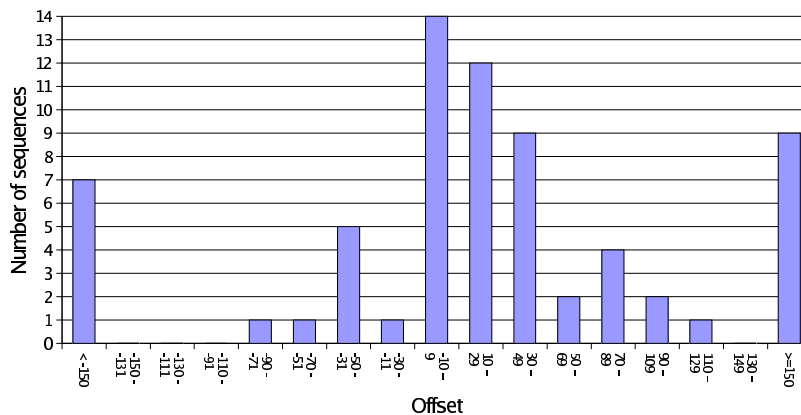
**Figure 3.5.** The EponineTSS\_1 model, trained from 389 mammalian sequences from EPD.

A/T-rich motifs which have been assigned negative weights in the GLM, signaling a preference for “not A/T” in that region). However, the exact sequences and positions of these PCs varied substantially between training runs. I believe that this instability was due to lack of training data, or possibly the inclusion of some strongly atypical promoters in the positive set. In order to improve the quality of the learned models, I therefore decided to consider alternate sources of training data.

### 3.3. Model refinement using mouse cDNA sequence data

An alternative source of data, which became available during the course of this project, is the results from high throughput cap-trapped cDNA sequencing projects, such as the mouse FANTOM project [Kawai *et al.* 2001]. These projects use a molecular biology trick, outlined on page 16, to preferentially clone cDNA copies of full-length messenger RNA molecules. In principle, full-length cDNA sequencing seems an attractive way to discover transcription start

sites. However, there is still some uncertainty remaining as to whether cap-trapped cDNA clones really tell us about the true transcription start site, or if there is still a degree of truncation despite the cap-trap method. To demonstrate the causes for concern, I considered 68 human genes which had transcription start sites annotated in both EPD release 71 and DBTSS, a human full-length cDNA project based on similar technology to FANTOM. The sequences were mapped onto the current human genome assembly using the SSAHA fast sequence-matching package [Ning *et al.* 2001]. In many cases, the expected transcription start sites from the two methods differed significantly: a histogram of offsets is shown in figure 3.6. While the fact that differences can be seen is not entirely surprising – many cases of alternate transcription start sites have been observed in eukaryotes, and I believe that this is probably a common phenomenon – it is surprising, and a possible cause for concern, that the transcription start site according to DBTSS is more likely to be downstream of EPD than *vice versa*.



**Figure 3.6.** Histogram showing offsets of DBTSS start sites relative to those of corresponding EPD entries.

While the number of genes considered here is unfortunately rather too small to draw particularly strong conclusions, it appears that there seem to be approximately equal number of cases where DBTSS and EPD evidence for transcription start sites differs by a significant amount (>150 bases). These are likely to reflect *bona fide* alternate transcription start sites. However, when a small difference occurs, it appears that the DBTSS evidence is likely to be truncated. There are several plausible ways in which truncated mRNAs could be accidentally tagged by the cap-trapping process. Firstly, it is possible that the phosphatase treatment may not go

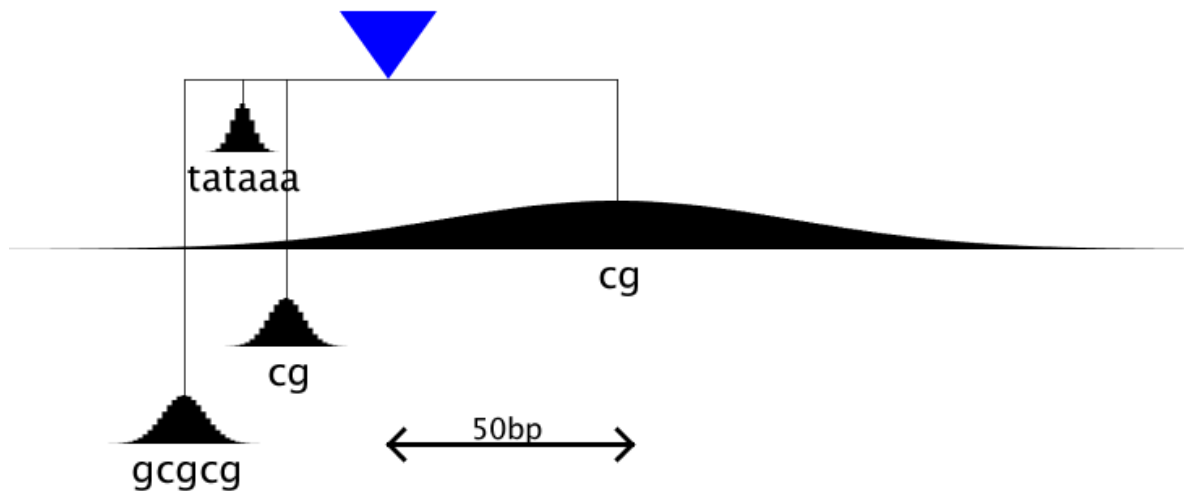
completion, leaving some capless mRNAs which retain their 5' phosphate groups. Second, it is possible that some RNA degradation could occur between the removal of the cap and the ligation of the tag – RNA is, after all, a rather unstable molecule and can undergo spontaneous hydrolysis. Despite these concerns, cap-trapped cDNAs still offer an interesting window onto the cell's transcriptome, but some care should be taken when using them as evidence of TSS position.

Simply sequencing cDNAs does not, by itself, give promoter sequences. However, in addition I had access to a repository of raw trace sequences [<http://trace.ensembl.org/>] which, at that time, contained around 1x coverage of the mouse genome, produced in the early phases of the the whole-genome shotgun sequencing effort [MGSC 2002]. Each trace was simply is simply the result of a single sequencing run, so the error rate is likely to be higher than that for assembled sequence. But modern sequencing methods are fairly accurate, and I considered trace data to be good enough for training this kind of model. To identify promoters in the trace repository, I searched 100 base fragments from the 5' ends of 19168 FANTOM cDNA sequences against the full set of sequences, again using SSAHA.

Using this procedure, I was able to retrieve trace sequences containing an exact match for 9958 cDNA ends. Of these, 3813 traces had at least 150 bases upstream of the mapped cDNA end. These were used as the basis for the second-round training set. But as shown above, it seems likely that many of these cDNAs may be slightly truncated. Therefore, I used the EponineTSS\_1 model to scan the sequence 20 bases upstream of the mapping point for the cDNA 5' end, to filter this set and create a dataset whose entries are very likely to represent true transcription start sites. When an EponineTSS\_1 prediction with a score of at least 0.999 occurred in this region, the sequence was accepted and the TSS annotation was adjusted to the point with the highest EponineTSS\_1 score.

Finally, this process left a set of 599 mouse sequences with at least 130 bases of upstream sequence, anchored with a high degree of confidence at the true transcription start site. As before, a negative training set of equal size was built using final-intron sequences, and the

VRVM trainer was run using the same configuration as before. The learned model, known as EponineTSS\_2, is shown in figure 3.7.

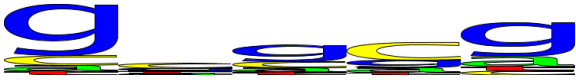





**Figure 3.7.** The EponineTSS\_2 model, trained from 599 mouse sequences.

This model is dramatically simpler than the previous versions from figure 3.5. I note a single clear motif (position -30) which would appear to represent the TATA box. This is closely flanked by two C/G-rich motifs. Finally, there is a preference for C/G enrichment, primarily in the region immediately downstream of the transcription start site. I presume that this part of the model is detecting a signal related to the previously reported CpG islands in promoter regions – however, this model taken as a whole is clearly more than a straightforward CpG island detector. In this case, all learned PCs were given positive weights in the GLM. This suggests that the PCs with negative weights seen previously may have been artifacts, or reflect problems with the training data.

### 3.4. Validation and testing of EponineTSS

I was not able to identify a single, self-contained, dataset which tested every aspect of the EponineTSS model. The EPD database contained accurately mapped transcription start sites, but did not cover any single large region of genomic sequence. On the other hand, while

Motif	Centre	Weight
	-42	6.23
	-30	8.64
	-21	3.52
	47	12.3

**Table 3.1.** The four position weight matrices used in the EponineTSS\_2 model.

curated annotation of finished human genomic sequence was underway, this did not include such accurate localization of transcription start sites. I therefore used a two-stage evaluation procedure to consider accuracy both in terms of detecting the exact position of transcription initiation and the rate of overprediction in bulk genomic sequence.

### 3.4.1. Testing on human chromosome 22: bulk genomic performance

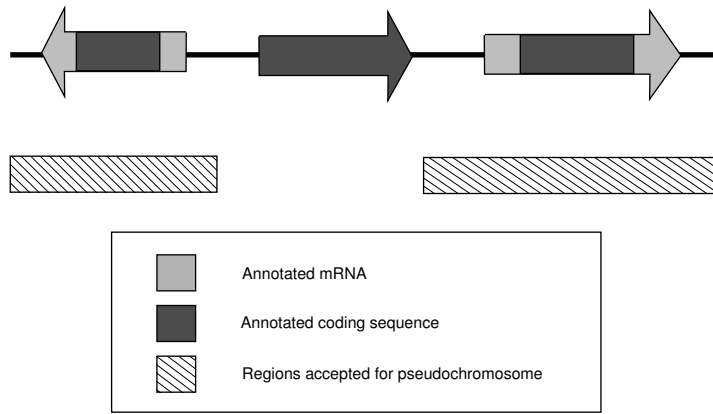
Since genome sequencing is now a large-scale operation, it is important that analysis tools perform well across large stretches of sequence, rather than just small regions being subjected to individual detailed analysis. Therefore, I assessed the performance of the EponineTSS model on human chromosome 22. This was an early milestone in the human genome finishing process, and at the time of its publication was the largest continuous piece of finished sequence

[Dunham *et al.* 1999]. When I was developing the EponineTSS models, version 2.3 of the chromosome 22 annotation [Dunham, personal communication] was the best-annotated large region available. Unlike some other regions, where annotators have concentrated entirely on annotating the coding regions of genes, chromosome 22 includes many annotations of complete transcribed regions, including full UTRs, based on EST and cDNA evidence and experimentally validated using the Rapid Amplification of cDNA ends technique [Schramm *et al.* 2000]. While this still is not guaranteed to pinpoint the actual TSS, it at least increases the chance of finding the correct first exon in genes with interrupted 5' UTRs.

Unfortunately, not all genes were validated to an equal degree. Out of 618 annotated genes, 284 were marked “GD\_mRNA”, indicating that they are presumed complete gene structures with supporting experimental evidence. I therefore believed that the true TSS is likely to lie close to the annotated 5' end of these structures. The remaining annotations are not necessarily complete, and I wished to treat them with a degree of caution. In particular, it is quite possible that a prediction made some distance upstream of one of these structures could be the true transcription start site, if the true first exon is missing from the gene structure. Therefore, I decided to construct a large synthetic DNA sequence – a pseudochromosome – consisting exclusively of those portions of chromosome 22 where the annotation which could be treated with reasonable confidence.

The pseudochromosome was constructed by selecting all regions containing GD\_mRNA genes (including as much flanking sequence as possible), while rejecting regions containing partial and unverified genes (figure 3.8). The problematic case occurred when a GD\_mRNA gene appeared adjacent to a partial gene in the opposite orientation, with both genes transcribed outwards from a common intergenic region: a pair of divergent genes. In most cases, these are assumed to be entirely separate transcription units, although there are thought to be some closely spaced cases where two genes are transcribed from a single regulatory region [Asakawa *et al.* 2001]. When preparing the pseudochromosome, the intergenic region of divergent pairs was split midway between the two gene structure annotations.



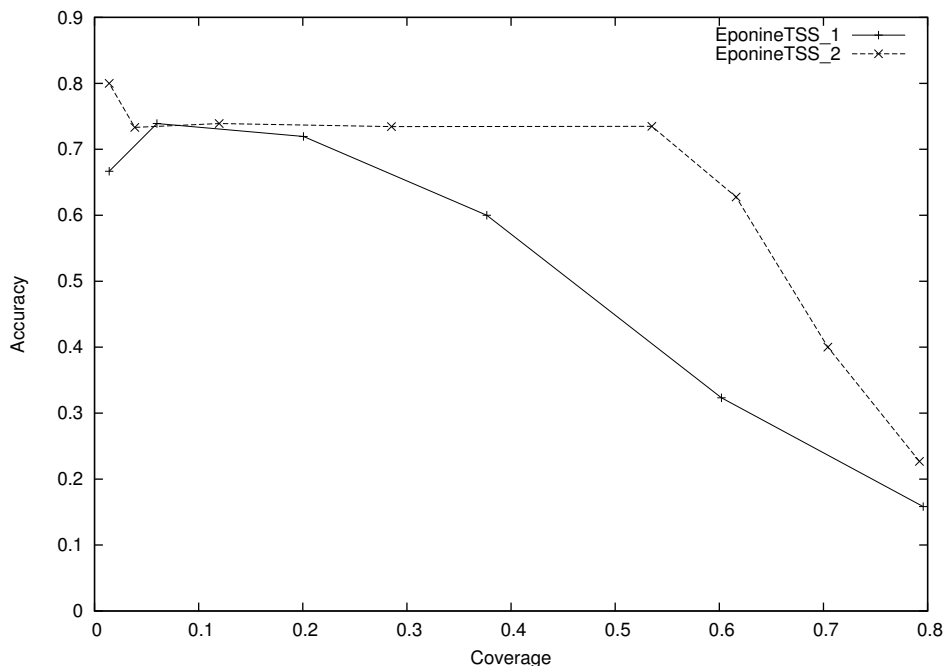


**Figure 3.8.** Selection of regions to include when the pseudochromosome sequence.

A pseudochromosome constructed in this way should be reasonably representative of chromosome 22 as a whole. This is, however, considered to be a relatively gene-dense chromosome by the standards of the human genome as a whole [Dunham *et al.* 1999], so some care should be taken when extrapolating results to other systems.

Both the EponineTSS\_1 and EponineTSS\_2 models were run across the pseudochromosome at a range of thresholds, from 0.97 (extremely low stringency) to 0.99999 (only a very small number of predictions). Most predictions fell into small clusters, with sizes generally in the range of 50-1000 bases. These clusters seem likely to reflect alternate transcription start sites of a single, but since there is some uncertainty in the actual TSSs of the chromosome 22 genes, and no annotation of alternate TSSs, it was not possible to directly validate this hypothesis. Instead, all predictions were subjected to single-linkage clustering, joining all predictions at a distance of 1000 bases or less. Around 60% of clusters included predictions on both positive and negative DNA strands, so the strandedness of predictions was ignored at this point – this is discussed later. I counted coverage as the proportion of genes which had a prediction cluster overlapping a window stretching 2kb upstream of the annotated start, and accuracy as the proportion of prediction clusters which overlapped one of these windows. Receiver operating characteristic curves for both models are shown in figure 3.9. Note that at thresholds giving coverages of less than about 0.1, the total number of predictions considered

was rather low, therefore the accuracy figures for the points on the extreme left of the curves should be treated as relatively poor estimates.

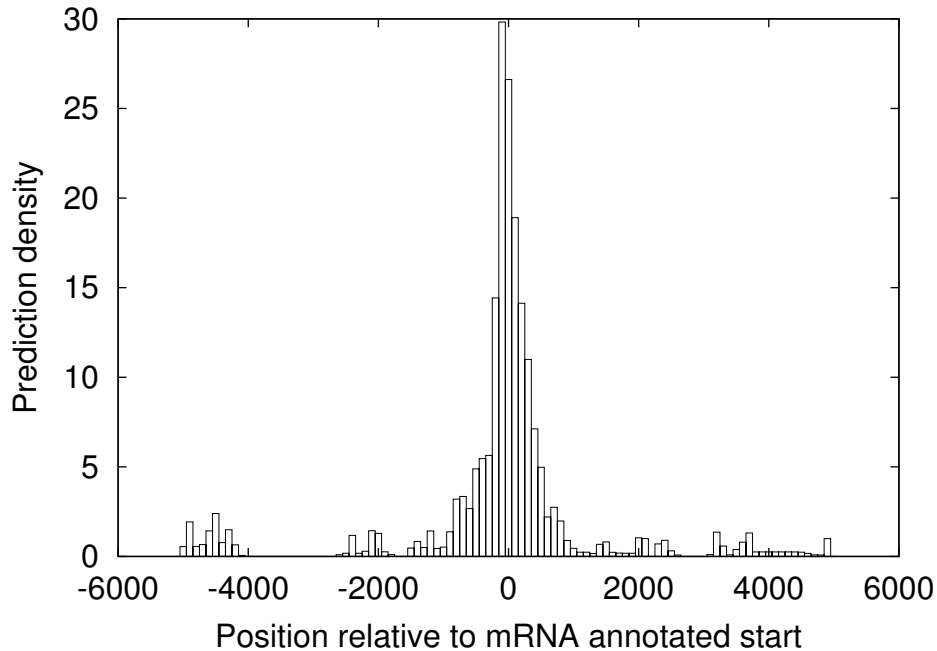


**Figure 3.9.** Accuracy vs. coverage for two EponineTSS models on the pseudochromosome.

From figure 3.9, it is clear that the EponineTSS\_2 model is significantly and unambiguously the better of the two predictive models, in terms of giving a substantially higher accuracy for a given coverage across a wide range of thresholds. Moreover, the accuracy of the EponineTSS\_2 model reaches a plateau of around 73% at a threshold of 0.999, and does not vary significantly from this when used at higher thresholds. The maximum accuracy of the EponineTSS\_1 model is very similar, but is only achieved at a much lower coverage. I therefore chose to use the EponineTSS\_2 model at a threshold of 0.999 as the final “product” of this approach.

While I do not consider this dataset to be an ideal test of positional accuracy, it remains interesting to see how well the predictions correspond to the annotated gene starts. Therefore, predictions from the EponineTSS model were placed into 50bp-wide bins according to their position relative to the annotated start of the nearest GD\_mRNA gene, giving the density histogram in figure 3.10. While there is some variation in position, the vast majority of

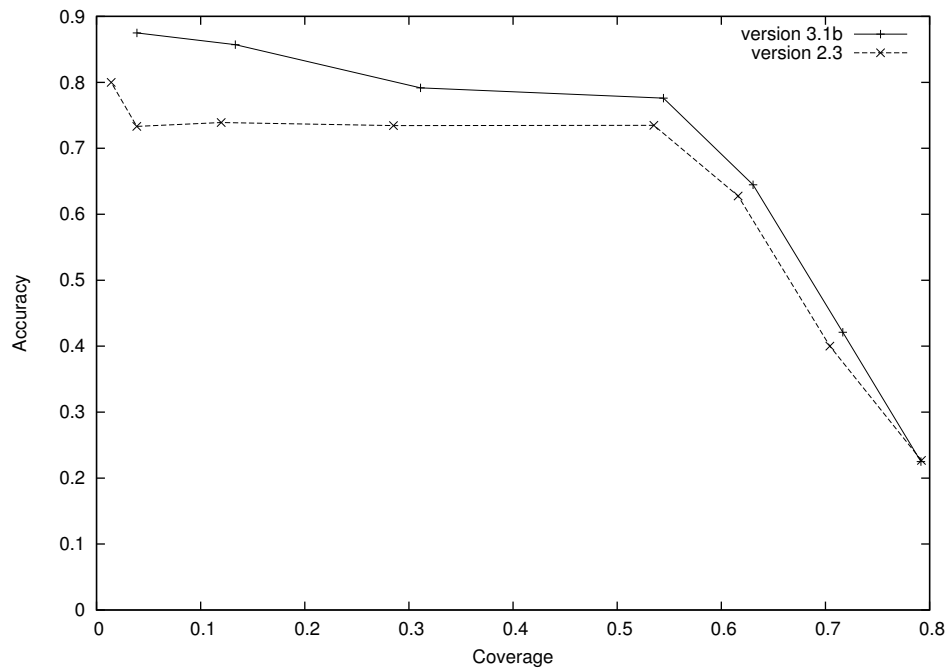
predictions are within 1000 bases of the annotated start, and the distribution is sharply peaked around 0. In a large proportion of cases, the predictor and the curated annotation agree quite well.



**Figure 3.10.** Density of prediction from the EponineTSS\_2 model relative to annotated gene starts.

More recently, a new (version 3) assembly of chromosome 22 has been published, and I obtained an updated version of the gene annotation, described as version 3.1b [Dunham *et al.*, personal communication]. This included more GD\_mRNA confirmed gene structures, and also improved annotation of pseudogenes and non-coding RNA genes. A new pseudochromosome was prepared using the same protocol as before, excluding all genes except those designated GD\_mRNA. The resulting sequence was similar to the original pseudochromosome, but was slightly longer (19.3Mb), and included an additional 76 GD\_mRNA genes, bringing the total to 360. Running the EponineTSS\_2 model with a threshold of 0.999 gave a coverage of 54.4% – not a significant difference from the 53.5% obtained with the earlier annotation – but accuracy had increased slightly from 73.5% to 77.6%. Plotting the full ROC curve (figure 3.11) shows that the accuracy was consistently a little higher over a wide range of coverage values. This suggests that at least some of the false positives are real transcription start sites which will no longer

appear as false positives as the standard of annotation improves.

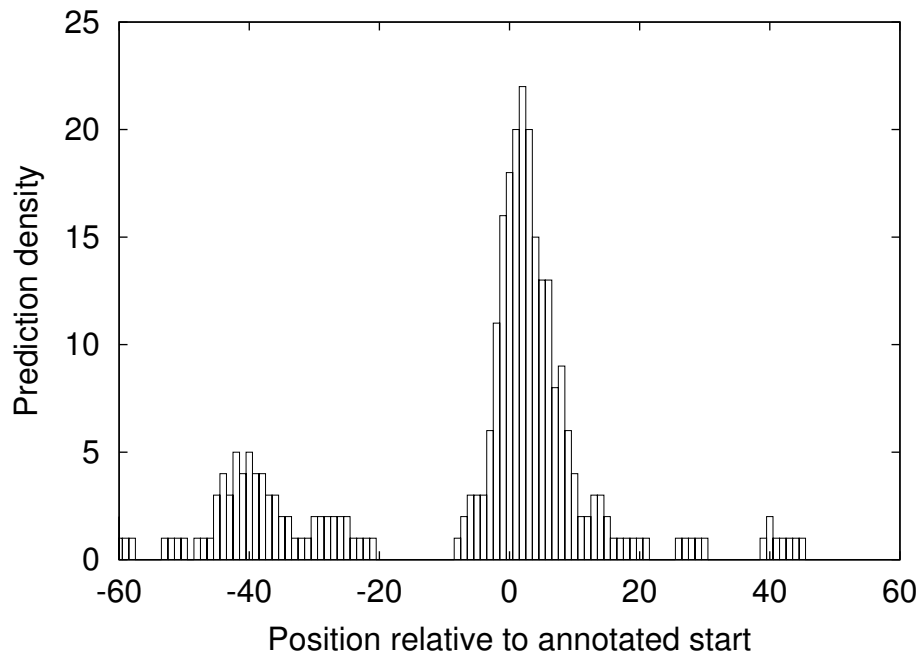


**Figure 3.11.** Accuracy vs. coverage for the EponineTSS\_2 model on pseudochromosomes based on old (2.3) and new (3.1b) curated annotation of chromosome 22.

### 3.4.2. Testing on EPD: calibration of positional accuracy

The specifications for the EPD database require transcription start sites to be mapped experimentally to an accuracy of  $\pm 5$  bp, making it the most precise resource available for evaluating the positional accuracy of the model's predictions. I used the EponineTSS\_2 model to scan the 50 EPD entries which were not used in the phase-one training process. A histogram of the prediction positions relative to mapped TSSs is shown in figure 3.12.

The results show a clear peak, with predictions clustered in the interval  $[-10:20]$  relative to the EPD-specified transcription start site – only a little more than the  $\pm 5$  base pair tolerance specified for EPD mapping. Since EPD is a resource made up from data submitted or published by a large community of researchers, it is not known for certain if the  $\pm 5$  criterion is actually true. However, this result suggests that EponineTSS predictions are very likely to be within 10 bases of the true TSS. If EPD mapping is slightly less accurate than stated, it is possible that EponineTSS



**Figure 3.12.** Density of EponineTSS\_2 predictions relative to the annotated TSS of EPD entries.

accuracy may be higher than this. A second, much smaller, peak occurs around -40 relative to the true TSS: this might represent a common spacing for alternate transcription start sites, but it could also be noise in the results – especially since 50 sequences represents a relatively small test set.

### 3.4.3. Comparison with other methods

I compared EponineTSS specificity and coverage on the human pseudochromosome sequence. Three other methods were considered:

- A generic CpG island detection program, as described on page 22, which was run by the chromosome 22 analysis group [Dunham *et al.* 1999].
- The PromoterInspector program [Scherf *et al.* 2000], a dedicated method for prediction of promoter regions.
- A DNA weight matrix for the TATA box, derived by alignment of EPD sequences

[Bucher 1990].

Due to availability restrictions, I could not download PromoterInspector and directly scan the chromosome 22 sequence, and limitations on the web-based version of the program prevented scanning large pieces of sequence. However, the authors had previously published a set of predictions on chromosome 22 [Scherf *et al.* 2001], and I was able to extract the complete set of predictions from their web pages. Unfortunately, these predictions were published in full-chromosome coordinates for an earlier assembly of chromosome 22, while I was keen to use the more recent version 2 assembly, which had better gene annotation with many more GD\_mRNA confirmed gene structures. Without the ability to rerun PromoterInspector, it was not possible to make a direct comparison. Therefore, I extracted DNA sequences for the region of each prediction from the original assembly and used SSAHA [Ning *et al.* 2001] to find perfect matches on the version 2 assembly. In this way, I mapped 99.4% of the original PromoterInspector predictions. Therefore I believe that results given here for PromoterInspector should be representative.

The TATA box weight matrix (shown graphically in figure 1.8) was downloaded from the EPD website [EPD, <http://www.epd.isb-sib.ch/>]. From this website, I also obtained a recommended log-odds score threshold of -6.5. However, scanning the full pseudochromosome with this threshold gave 39869 predictions – far more than any of the other methods considered here. Moreover, these were distributed quite uniformly across the chromosome and I was not able to reduce them to a more reasonable number by clustering. Therefore, I experimented with alternative thresholds, and found that a log-odds score of -2.6 gave 540 predictions, a number more in line with the other methods considered here.

Accuracy and coverage figures on the pseudochromosome were assessed as before, and results for all four methods are shown in table 3.2. I note that, at either of the thresholds tested, the accuracy of the TATA weight matrix is extremely low – this is clearly not an acceptable methods for finding promoters in a genomic context

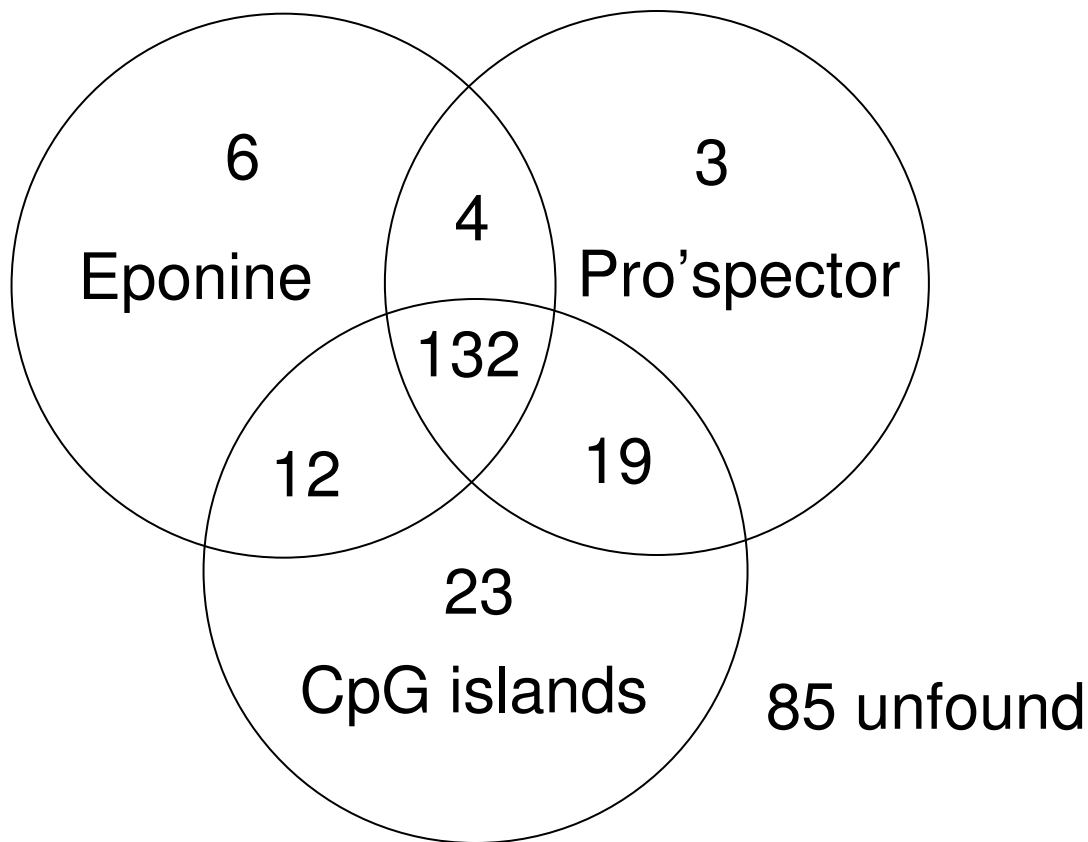
Method	Predictions	True positives	False positives	Coverage (%)	Accuracy (%)
<b>Eponine</b>	215	152	57	53.5	73.5
<b>Pro'spector</b>	278	157	100	55.3	64.0
<b>CpG</b>	306	187	116	65.8	62.1
<b>TATA -2.6</b>	540	37	500	13.0	7.4
<b>TATA -6.5</b>	39869	283	37581	99.6	5.7

**Table 3.2.** Sensitivity and selectivity of various promoter-prediction mechanisms on the human pseudochromosome.

The three remaining methods all offer much higher levels of accuracy, indicating that they can distinguish promoters from bulk genomic DNA. The coverage of all three methods is relatively similar. Moreover, the sets of genes detected by these methods are strongly overlapping, as shown in figure 3.13. This seems significant, since the three methods under consideration are technically quite different (although EponineTSS and PromoterInspector both used the EPD database during training). When this is taken in the context of the prior observation that further increasing the coverage for the EponineTSS\_2 model means a severe loss of accuracy (figure 3.9, I believe that this indicates some significant difference between promoters which are detected by the methods considered here and those which are not.

### 3.5. Analysis of cases where promoters were not detected by EponineTSS

At least in this test region, the EponineTSS model was able to detect just over 50% of promoters. Noting that this set seemed to be largely common with other *ab initio* promoter-prediction method, I suspected that promoters could be subdivided into several classes, only one of which was being detected here – and which was also correlated with the previously noted phenomenon of CpG islands. I was therefore interested to see if I could learn anything else about this subset of genes, and also about the promoters which cannot currently be detected.



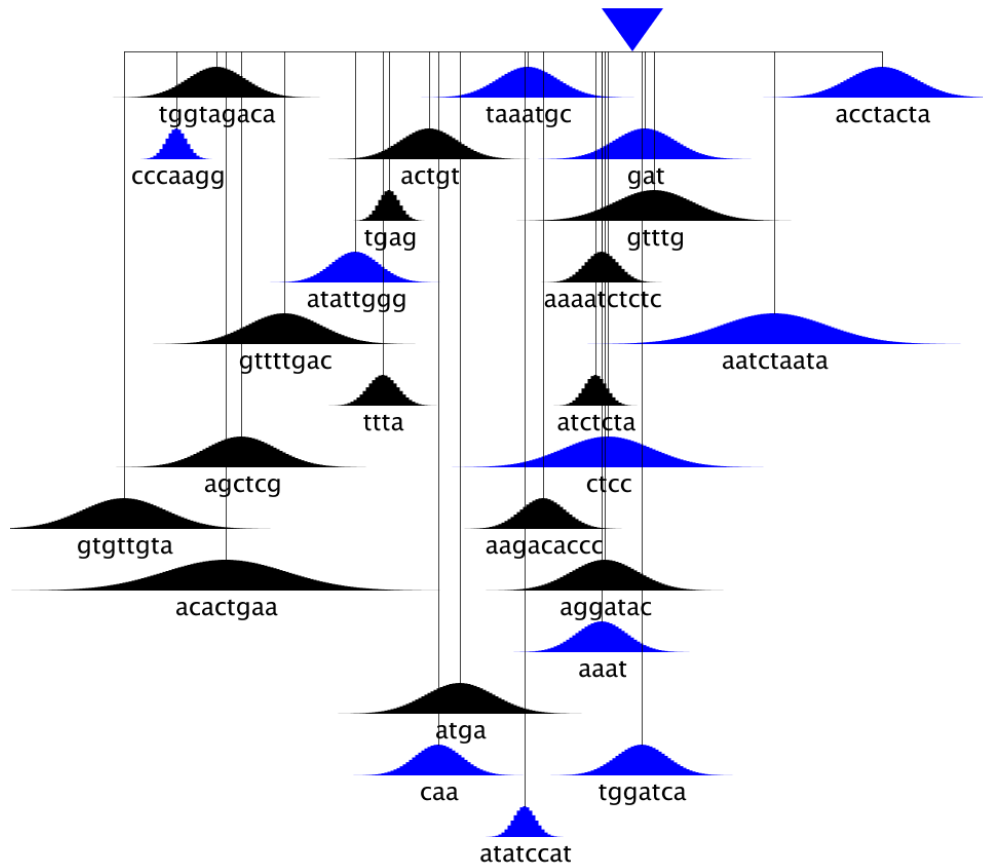
**Figure 3.13.** Intersection of “correct” predictions of promoters by EponineTSS, PromoterInspector, and CpG islands.

### 3.5.1. Modeling of non-detectable promoters

An obvious first step in analyzing the non-detectable promoters is to attempt to model them using the same approach as previously. At the time, the best candidate training set for this task was the set of mouse promoters previously derived from the FANTOM cDNA set. While previously, the EponineTSS\_1 model was used to positively select a subset of these with detectable promoters, here a model was trained on sequences which do not receive any prediction. Unfortunately, in this case there was no obvious approach to align the training sequence at the transcription start site, so any model learned was unlikely to give the same positional accuracy as those considered so far. Once again, human final intron sequences were used as a negative set. When this process was carried out, a non-empty model was learned, suggesting that there *is* some information in this training set. However, the models are rather

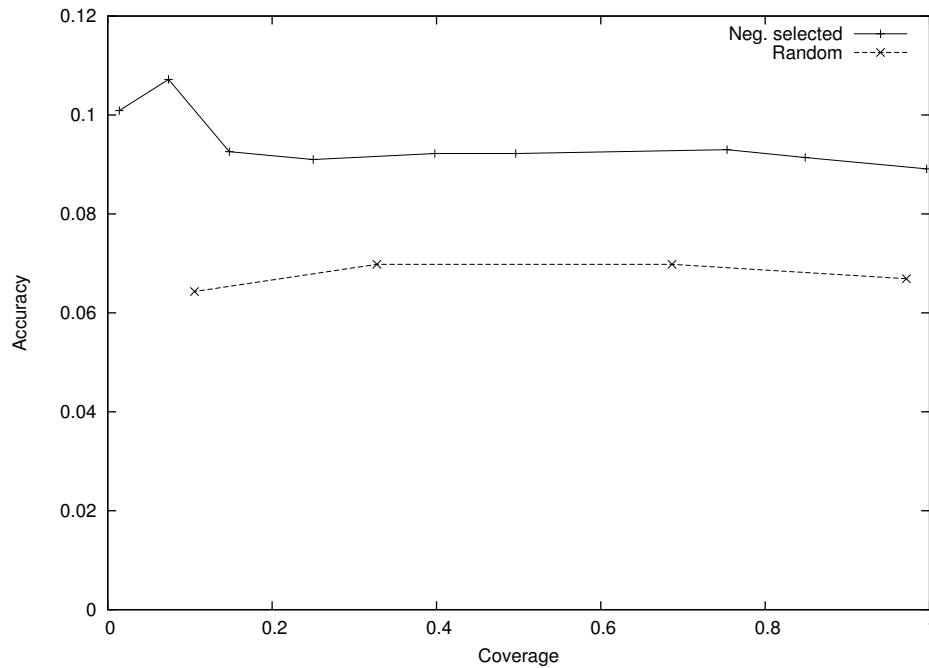


complex, and had few consistent features between training runs. An example is shown in figure 3.14.



**Figure 3.14.** Model trained on negative-selected FANTOM data.

This model was tested on the pseudochromosome, using the same approach as before: see accuracy vs. coverage in figure 3.15. Regardless of threshold, this model gives a far lower accuracy than those considered previously – certainly too low to be a useful predictive tool on a genomic scale. To determine whether the model has captured any information about promoters at all, I compared it with the results from an entirely random predictor (i.e. assigning each point on the pseudochromosome a random score sampled from a uniform distribution over the interval [0.0:1.0]). This is the second trace in figure 3.15. This comparison shows that the negatively selected model does capture some information, but has very low predictive power compared to the previously described models.



**Figure 3.15.** Accuracy vs. coverage for model trained on negative-selected FANTOM data, with random predictions for comparison.

I attempted to amplify the information contained in this preliminary model by using it to positively select a subset from the FANTOM dataset. However, the second-stage models produced in this way were not substantially simpler than the first-stage model, nor did they give a better prediction accuracy.

### 3.5.2. Correlation of promoter-detectability with gene type and function

I wished to determine if the distribution of promoters detected by the EponineTSS model was entirely random, or whether some classes of genes were much more likely to be regulated by detectable promoters than others. This question is particularly significant in the light of previous reports that CpG islands are associated primarily with “housekeeping” genes [Larsen *et al* 1992].

I concluded that the set of 284 chromosome 22 genes was too small to get a reliable impression of any correlation. Therefore, the complete set of Ensembl gene predictions for the

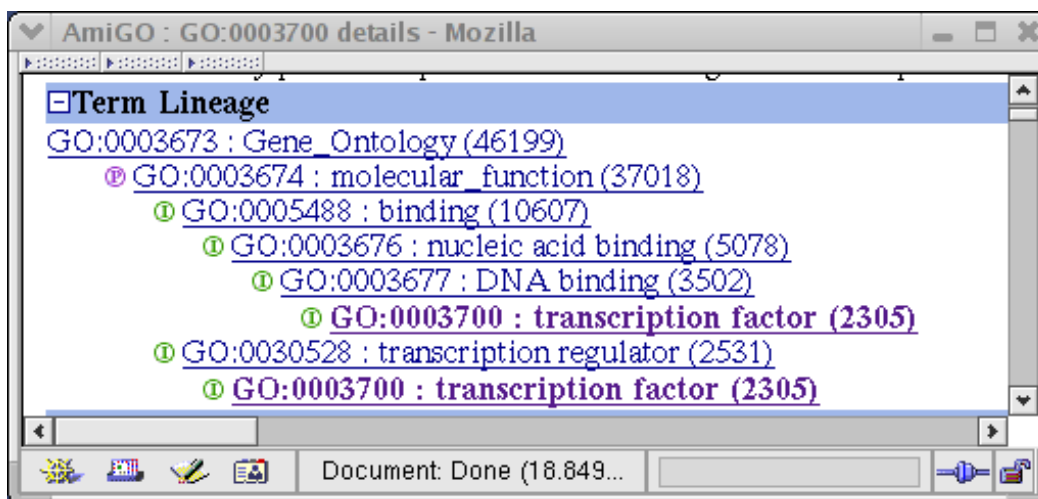
NCBI\_30 assembly of the human genome was used – a total of 27,628 predicted transcripts from 22,980 predicted genes. The EponineTSS\_2 model was used with a threshold of 0.999 to scan the entire genome. Transcripts were classified as:

- **Found** if a TSS prediction occurred within 2000 bases of the predicted gene start. This set consisted of 3765 transcripts.
- **Uncertain** if a TSS prediction occurred within 25,000 bases, but not within 2000 bases. This set consisted of 10948 transcripts.
- **Unfound** if no predictions occur within 25,000 bases. This remaining set consisted of 12915 transcripts.

The existence of the **uncertain** category aims to cover at least the majority of cases where the first (perhaps entirely non-coding) exon is missing from the gene prediction. Here, the uncertain cases are ignored and the emphasis is on comparing the found and unfound categories. It seems very likely that in many cases the predictions for the **uncertain** transcripts were, in fact, correct. However, the **found** category was sufficiently large that I did not consider it necessary to count the **uncertain** cases.

I wished to avoid lexical processing of gene description strings if possible. Instead, I relied on terms from the controlled vocabularies for molecular functions and biological processes provided by the GO project [The Gene Ontology Consortium 2000]. Automatic GO annotation is available for the bulk of Ensembl predicted transcripts, *via*. the GOA annotation of the Swissprot database [Camon *et al.* 2003].

I counted the proportion of both the found and the unfound transcript sets which was covered by each term in the GO ontologies. As well as counting terms applied by the GOA annotation, I counted all ancestor terms through the *is-a* and *part-of* relationships recorded in the GO database. For example, a gene annotated with the term GO:0003700 (“transcription factor”), as shown in figure 3.16 would also be counted as “transcription regulator”, “DNA binding”,



**Figure 3.16.** Example GO term lineage.

“nucleic acid binding”, and “binding”. Counting all levels of the ontology in this way avoids questions about which levels of the ontology would be most appropriate for comparing the two sets of genes.

This process resulted in a list of 3746 GO terms. These were ranked by the difference between the proportions of found and unfound transcripts labeled with the term (see tables 3.3 and 3.4). This means that a term which labels around 10% of both the found and unfound sets will appear near the middle of the listing. But conversely, a term which labels one found gene and no unfound genes will also appear close to the middle – a useful property, since a difference of a single gene is not statistically significant. The two tables show the sets of terms with the largest positive and negative differences respectively, and show terms which exhibit a clear and indisputable correlation with EponineTSS detectability. In the **found** column, a frequency of 0.01 corresponds to an actual count of 38 transcripts, while in the **unfound** column, a frequency of 0.01 means 129 transcripts. So most of the entries shown here are based on hundreds of transcripts, and are clearly significant.

Looking first at the list of terms overrepresented in the **found** category (i.e. large positive difference), near the top of the list we find “cell growth and/or maintenance”, “metabolism”, and “enzyme”. These are classically considered to be housekeeping functions, and seem to support

GO term name	Freq. (unfound)	Freq. (found)	Diff.
cell growth and/or maintenance	0.581	0.674	0.092
metabolism	0.400	0.488	0.087
ligand binding or carrier	0.439	0.507	0.067
DNA binding	0.118	0.176	0.057
enzyme	0.307	0.360	0.052
nucleic acid binding	0.193	0.243	0.050
nucleotide binding	0.101	0.148	0.046
purine nucleotide binding	0.101	0.147	0.046
nucleobase, nucleoside, nucleotide and nucleic acid metabolism	0.152	0.192	0.040
transferase	0.103	0.141	0.038
transcription	0.089	0.126	0.036
transcription regulator	0.052	0.088	0.036
transcription, DNA-dependent	0.088	0.123	0.035
transcription factor	0.046	0.080	0.034
protein metabolism and modification	0.147	0.182	0.034
protein modification	0.063	0.096	0.032
transcription regulation	0.083	0.115	0.032
kinase	0.049	0.081	0.032
phosphotransferase, alcohol group as acceptor	0.044	0.075	0.030
intracellular signaling cascade	0.065	0.091	0.026
transcription, from Pol II promoter	0.016	0.042	0.026
protein binding	0.070	0.096	0.025
protein kinase	0.035	0.061	0.025
ATP binding	0.078	0.103	0.024
cell organization and biogenesis	0.030	0.054	0.024
phosphate metabolism	0.046	0.070	0.024
adenyl nucleotide binding	0.079	0.103	0.023
guanyl nucleotide binding	0.022	0.046	0.023
protein serine/threonine kinase	0.022	0.045	0.022
developmental processes	0.086	0.106	0.020
cell cycle	0.046	0.065	0.019
GTP binding	0.017	0.036	0.018
transport	0.107	0.126	0.018
cell cycle control	0.014	0.031	0.017
transcription regulation, from Pol II promoter	0.008	0.025	0.017

Table 3.3. GO terms applied preferentially to **found** genes.

GO term name	Freq. (unfound)	Freq. (found)	Diff.
receptor	0.158	0.073	-0.085
response to external stimulus	0.135	0.058	-0.076
transmembrane receptor	0.122	0.046	-0.076
signal transducer	0.225	0.149	-0.075
defense response	0.090	0.025	-0.064
cell communication	0.325	0.265	-0.060
immune response	0.082	0.023	-0.058
response to biotic stimulus	0.095	0.037	-0.058
integral membrane protein	0.227	0.169	-0.058
G-protein coupled receptor	0.069	0.014	-0.055
rhodopsin-like receptor	0.060	0.009	-0.051
G-protein coupled receptor protein signaling pathway	0.081	0.033	-0.048
cell surface receptor linked signal transduction	0.105	0.061	-0.044
response to pest/pathogen/parasite	0.047	0.016	-0.031
defense/immunity protein	0.027	0.003	-0.024
perception of abiotic stimulus	0.032	0.009	-0.023
sensory perception	0.031	0.009	-0.022
integral plasma membrane protein	0.097	0.075	-0.021
response to wounding	0.028	0.007	-0.021
perception of external stimulus	0.043	0.022	-0.021
response to abiotic stimulus	0.042	0.021	-0.020
biological_process unknown	0.056	0.036	-0.019
RNA-directed DNA polymerase	0.018	3.969	-0.018
RNA dependent DNA replication	0.018	3.969	-0.017
cell adhesion	0.043	0.025	-0.017
signal transduction	0.206	0.188	-0.017
ligand	0.039	0.023	-0.016
stress response	0.055	0.039	-0.015
innate immune response	0.020	0.005	-0.014
perception of chemical substance	0.015	7.939	-0.014
chemosensory perception	0.015	7.939	-0.014
cytokine	0.023	0.008	-0.014
cellular_component unknown	0.057	0.042	-0.014
inflammatory response	0.019	0.005	-0.013
olfactory receptor	0.013	0.0	-0.013

Table 3.4. GO terms applied preferentially to **unfound** genes.

some connection between housekeeping and CpG-island-related promoters. However, the list also includes “transcription factor” and “protein kinase” – in fact, proportionately these terms are overrepresented by a much greater factor than “metabolism”. Yet expression of these genes is likely to be more restricted than that of metabolic enzymes.

Amongst the underrepresented terms, perhaps the most dramatic example is the “olfactory receptor” term, which makes up 1.3% of the unfound set, yet does not occur *at all* in the found set. Many other types of receptor also appear in this list. “G-protein coupled receptor” makes up 6.9% of the unfound set yet only 1.4% of the found set. Part of this is explained by the olfactory receptors (which are G-protein coupled), but other sub-types are also dramatically underrepresented. Other terms in this list include “immune response”.

The term showing the most dramatic difference – olfactory receptors – is perhaps the easiest to explain. The human genome is thought to contain well over 300 functional olfactory receptor genes [Crasto *et al* 2001], although not all of these are predicted by Ensembl. There are also many pseudogenes. But only one allele of one receptor locus is expressed in any particular olfactory neuron [Chess *et al* 1994]. The choice of receptor is presumed to remain constant throughout the neuron’s life. This seems to suggest that there is some specialized mechanism to select one particular olfactory receptor allele and promote the transcription of that – and no other – allele. Logic suggests that there might be advantages in using an atypical promoter system for these genes, since it lowers the probability of accidental transcription, compared to a common type of promoter sequence. Similar arguments could also apply to some components of the immune system. In addition, Ensembl identifies at least some of the fragments of the immunoglobulin light and heavy chains as genes, yet these would not be expected to have promoters at all, since they will only be expressed in cells where recombination events have occurred to create a complete immunoglobulin gene.

Some of the overrepresented terms are rather more surprising. Genes annotated as “metabolism” or “enzyme” fit well with the intuitive notion of housekeeping genes. Many

of these are required by most or all cell types for day-to-day survival. But the presence of transcription factors and other genes that play predominantly regulatory functions in this set does not fit this model and shows that the notion of a “housekeeping” gene is, at least, somewhat simplistic.

Another possible explanation for the difference is that gene products are required by the cell in radically different amounts. Many enzymes which catalyze key metabolic reactions are needed in significant amounts, simply to provide adequate throughput on a particular metabolic pathway. On the other hand a membrane-bound receptor can adequately perform its function with only a few copies present per cell. But once again, this fails to explain the presence of transcription factors on the overrepresented list, since they are not generally expected to be present in large quantities.

### **3.6. EponineTSS discussion**

The results shown here demonstrate that a novel sequence analysis model, based on some previous suggestions of promoter structure, is able to capture information from a suitable training set. Representing the model within the GLM framework allowed a standard “learning engine”, which did not itself contain any domain-specific knowledge, to drive the training procedure, selecting and weighting the final set of model elements. While the emphasis here has been on modeling the sequence around transcription start sites, similar approaches might be suitable for other point features in the genome, and indeed the code is currently being tested for the prediction of transcription termination sites [A. Ramadass, personal communication].

The EponineTSS model is able to predict a substantial proportion of transcription start sites (over 50% of the pseudochromosome test set) with a comparatively low level of apparent overprediction (over 70% accuracy on a representative subset of human chromosome 22). This combines better accuracy than existing methods with a comparable level of coverage. The



EponineTSS scanner is fast and efficient, making it a convenient method for first-pass promoter annotation on bulk genomic sequence – and as such, it has been integrated into the Ensembl project's standard vertebrate analysis pipeline.

I find it striking that accuracy on the original pseudochromosome seems to peak at 73%, and neither re-trained versions of the EponineTSS model, nor any other method, were able to substantially exceed this. I believe that as our understanding of the genome improves, at least some of the false-positive prediction clusters will turn out to correspond with real promoters. Certainly, the improvement in accuracy when evaluating against the new version 3.1b chromosome 22 annotation seems to support this idea. On the other hand, it seems unlikely that all the remaining false positives correspond to protein coding genes: one possible explanation would be these promoters drive the expression of functional but noncoding RNA genes, such as the regulatory micro-RNAs [Grosshans and Slack, 2002], and it will be very interesting to watch as methods are developed which can more sensitively detect noncoding RNA genes, to see if some of EponineTSS' false positives turn out to be correct predictions.

Similarly, I note that this method, and many others, fail to predict more than 50-60% of promoters, at least without greatly increased rates of false-positive predictions, and that all current *ab initio* promoter-finding methods seem to detect similar subsets of promoters. I can show that the promoters correctly detected by this method are associated with a rather biased selection of genes. Olfactory receptor genes, whose expression is confined to small sub-populations of olfactory neurons, never have detectable promoters. This bias seems to extend to other types of receptor genes (I presume that many of these also have rather restricted expression patterns), and to components of the immune system. At the other extreme, enzymes are more likely than average to have a detectable promoter. This fits well with the existing suggestion that housekeeping genes have CpG-island promoters. But the same results also show detectable promoters associated with some of the key regulatory mechanisms of molecular biology: protein kinases and transcription factors. I suggest that, while most of these are not ubiquitously expressed, they are likely to be more widely expressed, in general, than the receptor

genes. Firstly, gene regulation is largely combinatorial, and it is a collection of transcription factors, rather than one single switch, which promotes gene expression. Secondly, regulators and signal-transducers are often re-used for multiple tasks in different cell types – for example, the MAP-kinase cascade [Nishida and Gotoh, 1993]. So while the idea that standard promoters are associated with housekeeping genes would certainly seem to be an oversimplification, it may well be true that promoters which include some standard core elements – and are detected by the methods considered here – tend to have more widespread expression patterns than those which do not.

One explanation for this is that the sets of training data used here (EPD and FANTOM) are significantly biased towards particular types of genes. This is an interesting possibility since PromoterInspector was also trained from EPD and exhibits the same bias as EponineTSS. There is more discussion of this in chapter 4. Assuming that the training set is not biased, another possibility is that there is some kind of functional bias in the correctness of Ensembl gene predictions, with certain types of gene more likely to have complete or near-complete UTR annotations, which are consequently more likely to match the TSS predictions. Alternatively, some types of gene may be particularly prone to duplication as pseudogenes, some of which are being predicted as false-positives by Ensembl but which lack functional promoters. Both of these hypotheses will become easier to test once curated annotation is available for a large fraction of the genome. The final possibility is that there is a genuine distinction between a “detectable” class of promoters, located by EponineTSS and PromoterInspector and generally associated with CpG islands, and the remaining promoters, which look quite different in sequence terms.

During the development of this method, I briefly experimented with two extensions to the basic EAS model. The first variant allowed PCs containing ‘scaffolds’ of two or more weight matrices, and was similar in implementation to the scaffold-based models in chapter 4. A second variant replaced simple position weight matrices with first order weight matrices, where the emission distribution at each position was conditional upon the symbol observed in the previous

position. This allowed dependencies between neighbouring bases to be modeled, as discussed on page 19. Both these variants allowed models to capture substantially more information than the basic EAS model, but did not give better performance for this particular task. This does not, however, rule out the possibility that they might be useful for some other sequence-analysis tasks.

So far, I have not been able to train a model to effectively detect the remainder of promoters. This may be due to limitations in the available training data (in particular, the fact that I could not precisely localize the true transcription start sites for the FANTOM promoters). However, it is also probable that the group of hard-to-detect promoters might be subdivided into several distinct types, without any strong common features. This could explain the complexity of the model in figure 3.14: the model includes elements from a number of unrelated different promoter types. In the future, this question could be addressed by taking a mixture-modeling approach which builds separate models for clusters of data in the training set, rather than forcibly fitting it to a single model.

One additional limitation is that the current models do not seem to effectively predict the direction of transcription from a given promoter, since the majority of prediction clusters include predictions on both strands. In some cases, such clusters can be explained by considering divergent genes, transcribed outwards from a single compact promoter region. A number of cases like this have been described, primarily in bacteria but also in eukaryotes, but there certainly are not enough divergent protein-coding genes to explain the large number of bidirectional clusters. One explanation would be that the “core” promoter signal seen here really does not provide substantial information about the direction of transcription, and that directionality is conferred by additional signals which are specific to individual promoters rather than being shared among large numbers of different promoters. Another, more radical, view is that divergent promoters such as that described in [Asakawa *et al.* 2001] – cases where a pair of closely-spaced genes are transcribed outwards from a single regulatory region – are more common than has so far been realized. While it is very unlikely that a large number of

genes have a so-far undiscovered protein-coding partner, it is much more plausible that there are many additional regulatory micro-RNAs, which could form divergent pairs with coding genes. Once again, until micro-RNAs can be accurately predicted, or experimentally detected in a high-throughput fashion, this will have to remain a tentative suggestion. In the mean time, it may still be useful to consider EponineTSS results when searching for novel RNA genes.

# Chapter 4. Learning from comparative genomics

Comparative genomics is the study of similarities and differences between two or more genome sequences. In its simplest form, it is based on the assumption that regions of the genome which perform important biological functions are more likely to be conserved between species than non-functional “junk”. This is intuitively quite logical, since at least some proportion of mutations in functional regions would be expected to have deleterious phenotypes, and be selected against, while mutations in non-functional areas should have no phenotype, positive or negative. The availability of the mouse genome, with close similarities to human, has caused much excitement in this field. In fact, there are a variety of genomes which make attractive targets for comparison – for example, the pufferfish *Tetraodon nigroviridis* has been proposed as a useful target for vertebrate comparative genomics, especially because of its small genome size, presumably containing little non-functional DNA [Roest *et al.* 2000].

To date, the main application of comparative genomics, at least in vertebrates, has been in the prediction of coding genes, with approaches such as Exofish [Roest *et al.* 2000] and Twinscan [Korf *et al.* 2001], and DoubleScan [Meyer and Durbin, 2002] utilizing similarity information from a variety of distances to predict genes with a greater confidence than purely *ab initio* methods, while still not requiring direct experimental data. Protein coding regions are among the best conserved regions, and sensitivity can be increased further by using an alignment model such as WABA [Kent and Zahler, 2000] which recognizes that certain nucleotide changes, mainly in the last position of a codon, are synonymous and have no effect on the final protein sequence. This sensitivity allows distant comparisons such as human-fish to yield interesting information. But other types of functional region also show conservation, and other types of comparative genomic application are being developed, for example in the

prediction of functional non-coding RNA genes [Rivas and Eddy 2001, di Bernado *et al.* 2003], where the expectation is that the secondary structure of the RNA will be more conserved than its actual sequence.

The human and mouse genomes show quite substantial similarity. An initial survey of these was published along with the draft mouse genome [MGSC 2002]. Briefly, coding genes are – as expected – among the most strongly conserved regions, but homologous regions can be observed throughout the genome. In total, it is possible to align up to 40% of the mouse genome to human [Schwartz *et al.* 2003], but it seems likely that at least some of this is just random “comparative noise” – regions of sequence which serve no particular purpose but which, purely by chance, have not yet accumulated enough mutations to make their relationship unrecognizable. However, it seems clear that some of the noncoding-but-similar regions, especially those with the highest levels of sequence identity between the two species, must have biological explanations.

Here, I suggest an alternative approach to comparative genomics, and present an example of its application in the analysis of mouse-human homologies. I chose to take a set of so-far-unexplained regions of strong similarity between two species, and try to identify what they have in common. This could be described as an ignorance-driven approach to scientific research, and is significantly different from the traditional approach of proposing hypotheses then testing them. However, it is an attractive way to explore large data sets. To make this strategy feasible, I used a machine learning approach in order to detect significant patterns in the chosen set of sequences. There have been some prior indications that this might prove to be a useful strategy. In the malarial parasite, *Plasmodium falciparum*, unsupervised training of a rather simple hidden markov model can partition the genome into several portions, each modeled by one state of the HMM. Moreover, some of the learned model states effectively identify expressed regions – and the direction of transcription – without having to supply any prior knowledge of what a *Plasmodium* gene might look like [Pocock 2001].

The HMMs of [Pocock 2001] just detect fairly simple biases in the frequency of either

single nucleotides or pairs of nucleotides in particular regions of the genome. This was effective in one case, but I wished to develop a method which could capture more complex information about regions of the genome, and this motivated the development of the Eponine Windowed Sequence model family.

#### 4.1. The Eponine Windowed Sequence (EWS) model family

Previously, I introduced the Eponine Anchored Sequence (EAS) model, which was a generalized linear model classifier for points within a large sequence. From a biological point of view, this kind of point classifier is a principled way to look at questions like transcriptional activation, since while the signals may be spread over a considerable area, their activity is focused on a single point or small set of points. However, it is often not possible to exactly define the points that are interesting. This is particularly true in this case, where a training set of sequences is being proposed as “interesting” purely on the basis of similarity to another species, with no prior knowledge of function. As well as having no indication of which point (if any) should be used as an anchor point, there will not even be any indication of the orientation in which the sequence should be considered.

The Eponine Windowed Sequence model is a generalized linear model approach to sequence analysis, following many of the same principles as EAS, and sharing many details of practical implementation, but it is directly applicable to regions of sequence. EWS models were designed to be trained with the previously described variational RVM library, using the same sampling-based strategy as EAS. The EWS model as used in this chapter is adirectional by design, but if it is to be applied to data from alternative sources, where the functional orientation of the training sequences *is* known, directionality could be reintroduced with only the most trivial changes in the computation.

In the first version of EWS, each basis function consists of a single position-weight matrix

(see definition, page 19). The basis function score is a normalized sum of the position-weight matrix probability (see page 19) at every position along the window of sequence under consideration:

$$\phi(S) = \sum_{i=1}^{|S|-|W|+1} (\vec{W}(S, i) + \overleftarrow{W}(S, i)) \frac{4^{|W|}}{|S|-|W|+1} \quad (4.1.1)$$

Where  $|S|$  is the length of sequence  $S$  and  $|W|$  is the length of PWM  $W$ . Dividing the scores by  $|S| - |W| + 1$ , the number of positions at which a motif of length  $|W|$  could begin, makes this scoring function independent of the exact size of the windows under consideration, making training on examples with a range of lengths possible. Like the basis functions of the EAS model, scores are also normalized for motif length: the  $4^{|W|}$  term avoids the basis function outputs taking very small values, which could cause numerical precision issues in the training algorithm.

Since the basis function space for this model was somewhat smaller than for EAS, a simpler strategy could be adopted for making initial choices of basis functions. Rather than starting with fragments picked from the training sequence, it was practical to enumerate all possible words of a specified starting length. These were shuffled onto a queue in a random order. When the trainer required a new basis function, a word was taken from the head of the queue and a new PWM proposed which preferentially matches this word. Once the queue is empty, the words are re-shuffled for another cycle through the pack. Once motifs have been proposed, the remaining sampling moves in the training procedure were similar to those used for EWS: weight matrices can be altered by re-sampling the columns from Dirichlet distributions, or can be shortened by dropping a column from either end. In this case, extensions of existing weight matrices are not permitted, so all the PWMs in the final model will have a length less than or equal to the starting length.

The second version extends EWS to capture larger-scale patterns in sequences. In this case, each basis function is a scaffold consisting of one or more PWMs, each with an associated position distribution relative to a scaffold anchor. In principle, distributions such



as the discretized Gaussian extend to infinity, but in practice it is reasonable to apply some cut-off: for instance, only considering the portion of the distribution which includes 99% of the total probability mass. The probabilities of all points outside this region are assumed to be infinitesimal and ignored. Now that the distributions have finite size, for a given scaffold there is a pair of integers,  $n$  and  $m$ , such that when the scaffold anchor is placed in the interval  $[n : m]$ , the non-infinitesimal parts of all the position distributions fall entirely within the length of a particular target sequence. A score for scanning this scaffold across the sequence can be given by

$$\phi(S) = Z \sum_{i=n}^m \left( \prod_{k=1}^K \left( \sum_{j=-\infty}^{\infty} P_k(j) (\vec{W}_k(S, i+j) + \overleftarrow{W}_k(S, i+j)) \right) \right) \quad (4.1.2)$$

where  $P_k$  is the  $k$ 'th position distribution and  $W_k$  is the  $k$ 'th weight matrix in the scaffold.  $Z$  is the normalizing constant:

$$Z = \frac{4^{\sum_k |W_k|}}{m - n + 1} \quad (4.1.3)$$

For the case when the scaffold only contains one PWM with a narrow distribution, the results will be the same as those from equation 4.1.1. So this can be considered a direct extension to the basic EWS model that can capture information about sets of motifs with correlated positions. Since the scaffold scores are only evaluated in the regions where the whole scaffold fits onto the sequence window, there is a risk of introducing edge effects. A possible future solution to this would be to use windows that are a little larger than the actual region, of interest, and use “soft boundaries” where scores from the edges of the window are given less weight than those at the centre. To train scaffold-based models, some additional sampling rules are needed:

- Combine the sets of motifs from two scaffolds, with randomly chosen offsets between the two
- Take a scaffold with two or more PWMs and return the scaffold with one of those PWMs

(picked at random) removed

- Alter the position or width of one of the relative position distributions in a scaffold.

The inclusion of scaffolds which could, for example, model two transcription factor binding sites with some preferred spacing between them, makes this second variant look more similar to EAS. In the implementation used here, scaffolds were limited to a maximum of three motifs. An example EWS-scaffold model is illustrated in figure 4.6.

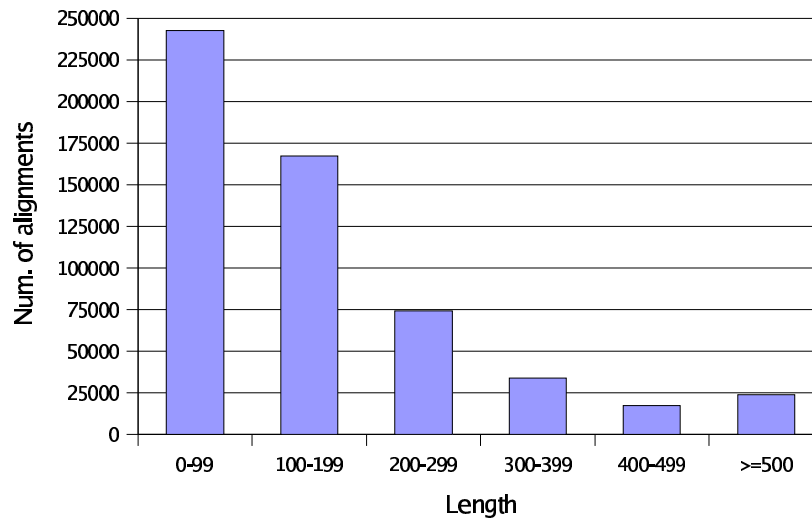
## 4.2. Training from non-coding homologies between human and mouse

Here, I used the EWS model to investigate a set of strong non-coding homologies between the human and mouse genomes. A number of methods have been developed for aligning two genomes [Schwartz *et al.* 2003]. These methods all use optimized sequence-search algorithms which trade some sensitivity for better computational performance. Nevertheless, whole-genome alignment is a computationally very demanding task, so I was keen to use an existing publicly-available set of results rather than running a new set of analyses.

At the time, the main set of publicly downloadable mouse-human alignments was provided by the Ensembl project in their *ensembl-compara* database [Clamp *et al.* 2003, A. Ureta-Vidal, personal communication]. This data was produced by first using the Exonerate program [G. Slater, unpublished] to perform a very rapid search for strong matches between the two genomes, which were used as “seeds” for the alignment process. When two seed hits occurred close to one another on both genomes, attempts were made to extend the alignments further by running *bl2seq* [Tutsova and Madden, 1999] (an implementation of the blast algorithm specialized for aligning two sequences rather than searching a database) on the regions of sequence lying between pairs of adjacent seed hits on the respective assemblies.

The *compara* database from Ensembl release 5 contained results from this protocol on release 5.28 of the draft human genome and release 5.3 of the mouse whole-genome shotgun

assembly. This gave 559,670 regions of similarity, covering a total of almost 93 megabases. While this is a substantial amount of sequence in absolute terms, it represents only 3.4% of the sequenced bases in the mouse genome, and somewhat less than that for the larger, more completely sequenced, human genome. The individual regions ranged from 20 to 8581 bases in length, and from 71% to 100% nucleotide identity. The bulk of the sequences are towards the lower end of this length range, as shown in figure 4.1



**Figure 4.1.** Length distribution of sequence regions aligned between human and mouse by ensembl-compara version 5.

One of the strongest contributors to the fraction of sequence in the compara data set was protein-coding genes. These represent a reasonably well-understood fraction of the genome, and were already known to be well conserved, so for the purposes of this particular study they were considered as “uninteresting”. Therefore, all similarity regions which overlapped an Ensembl gene prediction in either species were excluded. Note that the UTRs of predicted transcripts were also excluded at this point, but these might in fact make an interesting target for detailed analysis in the future using similar techniques to those presented here. Compara sequences overlapping repetitive elements, as detected by the RepeatMasker program [Smit and Green, unpublished] were also removed at this point, but these represented a very small fraction of the total. This is partially due to the use of repeat masked sequence in the seeding stage of the alignment process, but also since some of the largest families of repeats,

such as the Alu elements of the human genome, proliferated fairly recently in evolutionary time, and therefore only occur in one lineage.

The final restriction which must be applied is a minimum length threshold, since short sequences contain only a small number of sequence words, and are thus unlikely to give interesting results when analysed with the EWS model. For this experiment, I arbitrarily picked a sequence window size of 300 bases. Sequences which were shorter than this were discarded, while longer sequences were trimmed, and only a (randomly selected) 300-base portion from each sequence was used. This clearly removed a large number of sequences from consideration. However, this still left 75332 sequences, so there was little incentive in this case to analyze the shorter regions of similarity, or attempt to extend them with more sensitive alignment methods. In fact, training an EWS model on a dataset of that size would have been prohibitively slow (the limiting factor actually being the vast number of PWM score evaluations, rather than the actual RVM learning algorithm), so for this experiment I picked random subsets of size 2000, 4000, and 6000 to use as training datasets. Since EWS is still a binary classification model, a negative dataset was also required. In this case, randomly chosen non-coding, non-repetitive fragments from human chromosome 20 were used. This gave a roughly equal mixture of intron and intergenic sequence, with a small number of UTR sequences.

Single motif EWS models were trained from each of these sets with proposed words of length 5. The final motif sets from three runs are shown in table 4.1. These sets vary significantly from run to run, raising the serious possibility that the training process was not actually detecting any specific signal. The models were initially tested by scanning a two megabase region of human chromosome 22, considering non-overlapping 300 base windows. The scores for models 1 and 2 from table 4.1 are plotted in figure 4.2. While the two models do not agree precisely, a strong correlation can be seen. This can be quantified by calculating the Pearson correlation coefficient, in this case  $r = 0.89$ , which confirms the visual impression of a strong, but not perfect, correlation between the models. This correlation value was typical for other pairs of models tested.

Positive forward	Positive reverse	Negative forward	Negative reverse
aaagc	gcttt	aaaaa	ttttt
aatta	taatt	atttg	caaat
agc	gct	ccacc	ggtgg
catac	gtatg	cgtgc	gcacg
cccac	gtggg	ctagc	gctag
ccgc	gcgg	ctca	tgag
cggtta	taccg	cttac	gtaag
gacga	tcgtc	gatgc	gcatc
gcga	tcgc	gcgac	gtcgc
tcaca	tgtga	gggaa	ttccc
tccaa	ttgga	tagga	tccta
		tata	tata

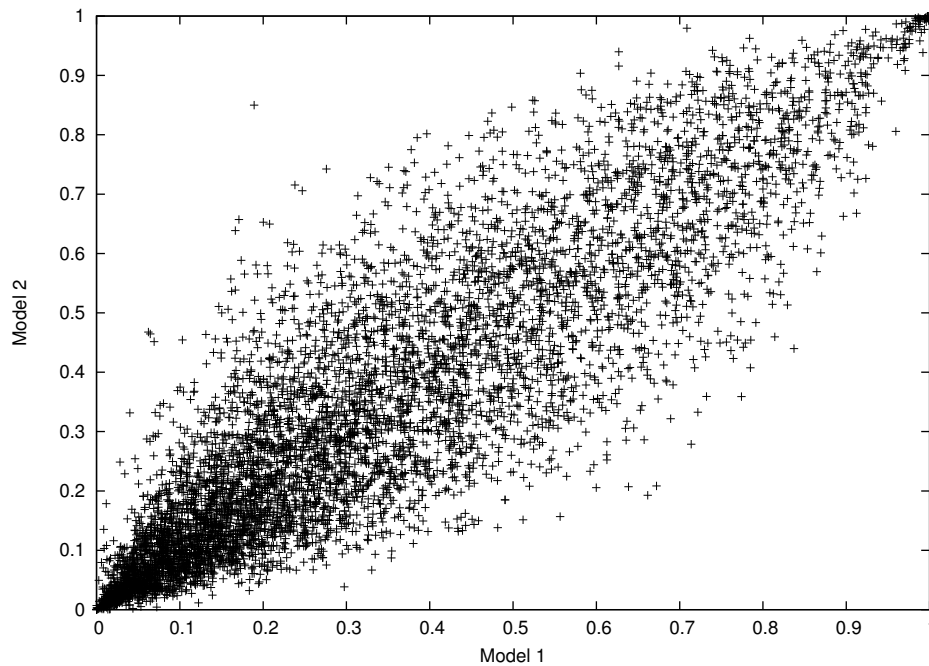
  

Positive forward	Positive reverse	Negative forward	Negative reverse
aacac	gtgtt	aaaaa	ttttt
atgga	tccat	aaata	tattt
attag	ctaata	attc	gaat
attg	caat	caccc	gggtg
cagc	gctg	ccca	tggg
cagta	tactg	ccga	tcgg
ccgac	gtcgg	ctagc	gctag
cgaaa	tttcg	cttac	gtaag
cgac	gtcg	gacca	tggtc
cgcca	tggcg	gtga	tcac
ctccc	gggag	gttga	tcaac
cttta	taaag	tagca	tgcta
tcaca	tgtga		

Positive forward	Positive reverse	Negative forward	Negative reverse
aacgc	gcgtt	aataa	ttatt
aactt	aagtt	accca	tgggt
aataa	ttatt	ag	ct
agccg	cggct	caac	gttg
agcg	cgct	cctga	tcagg
atatc	gatat	ctagc	gctag
atgac	gtcat	cttcc	gaaag
attag	ctaata	gcgc	gcgc
catca	tgatg	ggaca	tgtcc
cga	tcg		
cgaca	tgtcg		
ctgtc	gacag		
cttta	taaag		
gctga	tcagc		

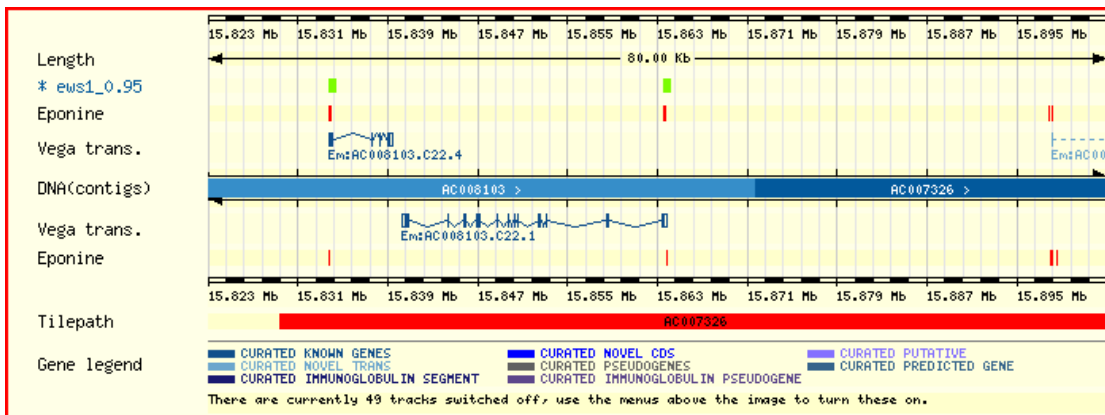
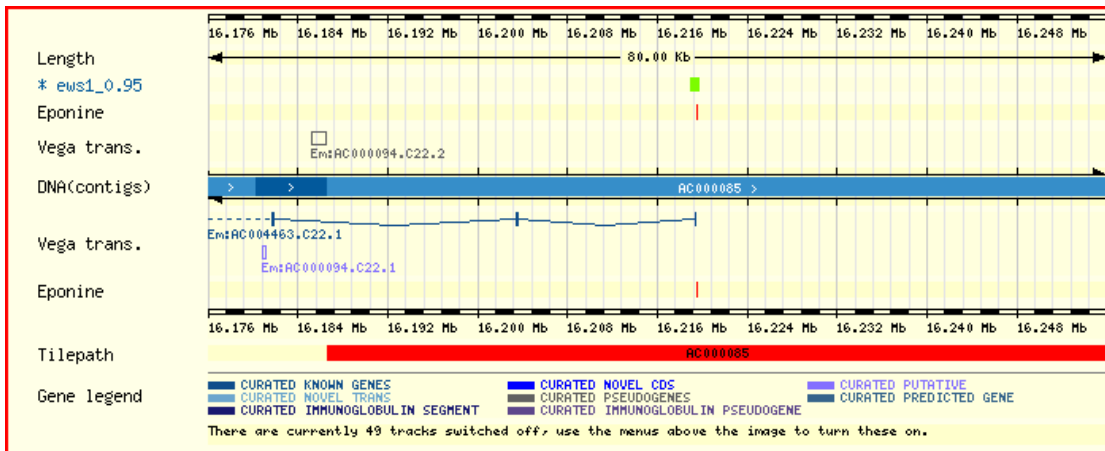
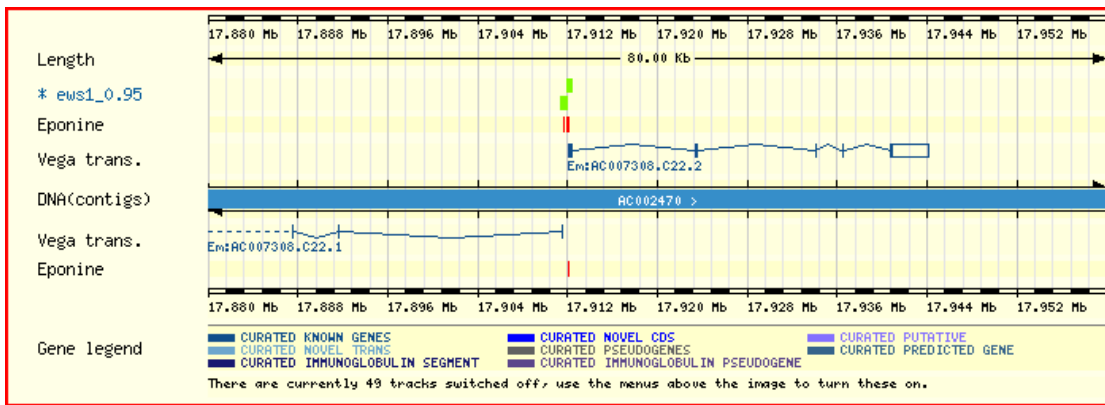
**Table 4.1.** Words learned from three EWS models trained from mouse-human homologies.



**Figure 4.2.** Scatter of scores from two different EWS models.

Given these strong correlations, it seems reasonable to assume that the training process locked on to some specific signal in the homology dataset. In order to get some impression of what this might be, I used one of the models to scan 300 base windows, with 200 base overlaps between adjacent windows, across the whole of chromosome 22, and set a threshold of 0.95, giving 985 high-scoring windows – a reasonable number for visual inspection. These were published using a DAS protocol server [Dowell *et al.* 2001] and viewed, together with other types of annotation, using the Ensembl genome browser. Some typical displays are shown in figure 4.3. Note that there is only one track of EWS predictions in each image, since the EWS model is adirectional, and its predictions cannot be placed on one or other DNA strand. Since these results were just taken from human chromosome 22, which is well covered with manually-curated gene annotations, the figures show the curated gene structures (available in Ensembl as “Vega transcripts” [<http://vega.sanger.ac.uk/>]) rather than Ensembl predictions.

This examination gave some indication of what the EWS models might be learning. Firstly, it is clear that the model has not learned a signal characteristic of coding genes – always a concern, since while homologies overlapping existing Ensembl genes were removed from the



**Figure 4.3.** Ensembl contigview displays for selected portions of human chromosome 22, showing windows with high scores for one of the homology models (labeled “ews1\_0.95”), and predictions from the EponineTSS\_2 model (labeled “Eponine”).

training set, it seems likely that at least a small number of coding sequences will have remained, reflecting false negatives from the Ensembl gene prediction pipeline. The bulk of predictions do not significantly overlap annotated exons, and for those which do it is almost always the first exon of a gene, which often consists largely or entirely of 5' untranslated sequence. The fact that this experiment did not pick up a protein-coding signal can be seen as evidence that there is probably not a large amount of undiscovered coding sequence in the genome. It is, however, clear that the predictions are concentrated near the start of genes. Moreover, they are strongly correlated with predictions from the EponineTSS\_2 model.

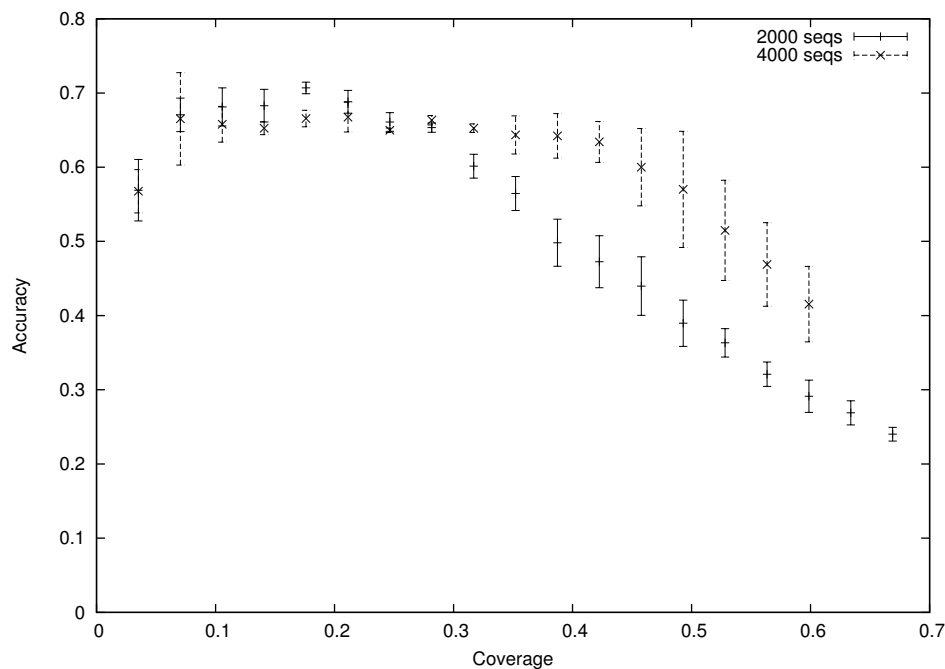
Based on this visual inspection, I provisionally concluded that the learned models were acting as promoter predictors, a result which was later backed up by the evaluation in the next section. It is not initially obvious exactly what aspect of the promoter signal is being detected by these models. Considering table 4.1, there are a number of words in this set which include the 5'-cytosine-guanine-3' dinucleotide: 11 such words with positive weights, against just 3 with negative weights. CpG dinucleotides are known to be significantly in promoters, both by inspection of the EponineTSS models, and from previous knowledge of CpG islands. However, given the number of motifs which were learned, it seems unlikely that this method is relying solely on this signal. It seems more plausible to think that the learned set of motifs consists of some CpG island signals, plus additional motifs reflecting common transcription factor binding sites.

One approach to check for transcription factor binding sites in the model is to compare the learned motifs with the TRANSFAC database [Matys *et al.* 2003]. Unfortunately, this proved not to be practical. The set of binding sites in the TFSITE database actually includes all 1024 possible 5-base nucleotide sequences. Obviously, most of these are embedded in longer sequences, but it means that there is no way to rigorously compare TRANSFAC with the set of 5-base motifs listed here. Moreover, moving to slightly longer motifs would not improve the situation significantly: of the 4096 possible hexamers, only 15 cannot be found in TFSITE.



### 4.3. Evaluating the function of mouse-similarity models as promoter predictors

To test the EWS models as a practical mechanism for predicting promoters in genomic DNA, I took the same approach previously used to evaluate the EponineTSS predictor. The models were used to scan the pseudochromosome described on page 67, again using overlapping 300 base windows, and the set of high-scoring windows was compared with the curated annotation using the same criterion of accepting predictions within 2kb of the annotated transcript start. Once again, receiver operating characteristic curves (accuracy vs. coverage) were plotted. As already shown, the learned models varied somewhat between training runs. I therefore trained three models from each dataset and calculated the mean and standard deviation statistics of the accuracy score at a range of coverage figures. Results of this analysis for models trained on two of the datasets are shown in figure 4.4. For higher levels of coverage the models from the 4000 sequence set gave significantly higher accuracy. However, training from the 6000 sequence set did not give any further improvement (data not shown for clarity reasons, since the results closely overlapped those for 4000 sequence models).

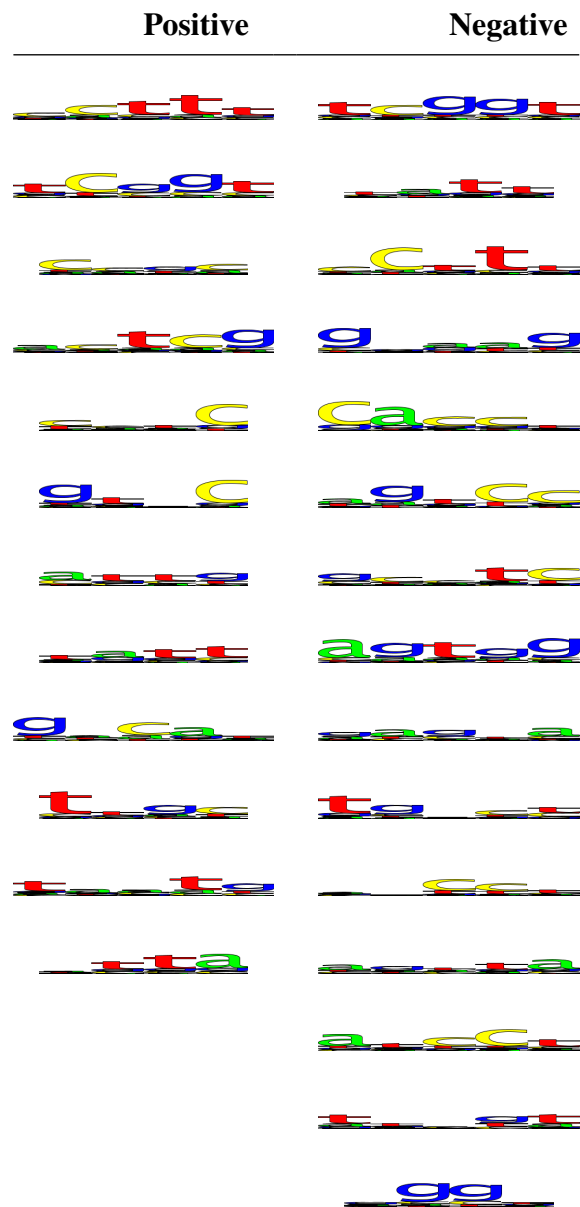


**Figure 4.4.** ROCs for EWS models trained from sets of 2000 or 4000 human-mouse homologies.

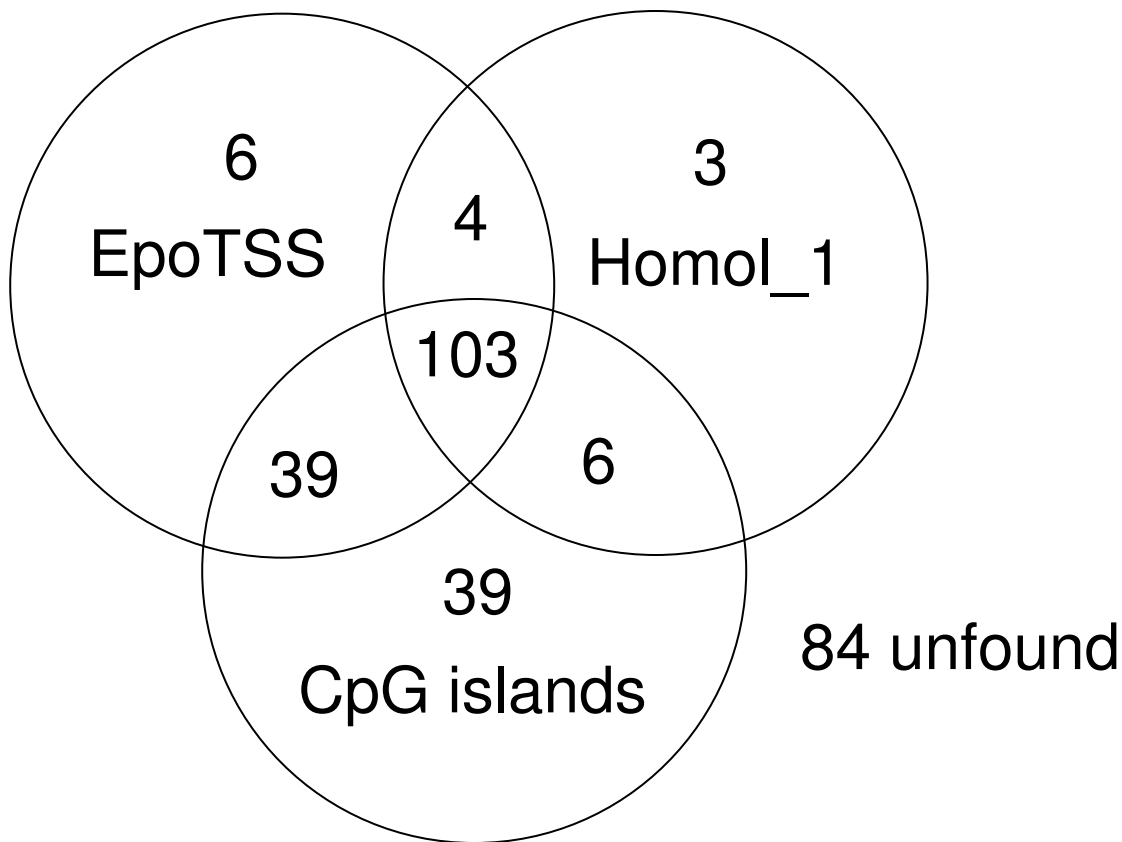
To optimize the initial word-length, a second set of models were trained using proposed words 6 bases in length. These all offered lower accuracy than the 5-base models, but the difference decreased with larger sets of training data. For models trained with 6000 sequences, the differences were minimal. Thus, it seems that models including longer motifs require larger amounts of data to train effectively. But it was not possible to improve the prediction performance significantly above what could be achieved using 5 base words.

One of the 4000-sequence, 5 base word models was selected for further study: from now on, this is labeled as EponineHomol\_1. The full set of motifs in this model are shown in table 4.2. At a threshold of 0.97, this gave a coverage on the pseudochromosome of 41% and an accuracy of 68%. These figures are both a little lower than those for the EponineTSS\_2 model, but still indicate significant predictive power. In fact, the accuracy is slightly higher than that for the EponineTSS\_1 model at a comparable coverage level. Of course, the two are not directly comparable since the EponineTSS models also give information on the actual position of transcription initiation, while the models discussed here simply indicate regions of 300 bases or more which are likely to lie in the vicinity of transcription start sites.

Next, I investigated which promoters were detected by the EponineHomol\_1 model. Of the 284 gene starts on the pseudochromosome, 116 were correctly predicted by the EponineHomol\_1 model at the chosen threshold. 107 of these were also detected by EponineTSS\_2 (see figure 4.5). If the two methods made predictions entirely independently, the expectation would be that only 62 out of 116 would coincide by chance, so this would seem to be evidence for a significant correlation. As discussed previously, the set of promoters found by EponineTSS\_2 could quite plausibly be biased either by the set of promoters which had been submitted to EPD, or by the set of mRNAs which were successfully cloned in the FANTOM project. Neither of these dataset-related biases could have had any effect on the training of the EponineHomol\_1 model. Thus, this coincidence suggests that the distinction between the “found” and “unfound” sets of genes discussed in chapter 3 must have some deeper biological significance.



**Table 4.2.** Logo view of the motifs used in the EponineHomol\_1 model.



**Figure 4.5.** Intersection of transcription start sites correctly predicted by the EponineTSS, EponineHomol, and CpG methods.

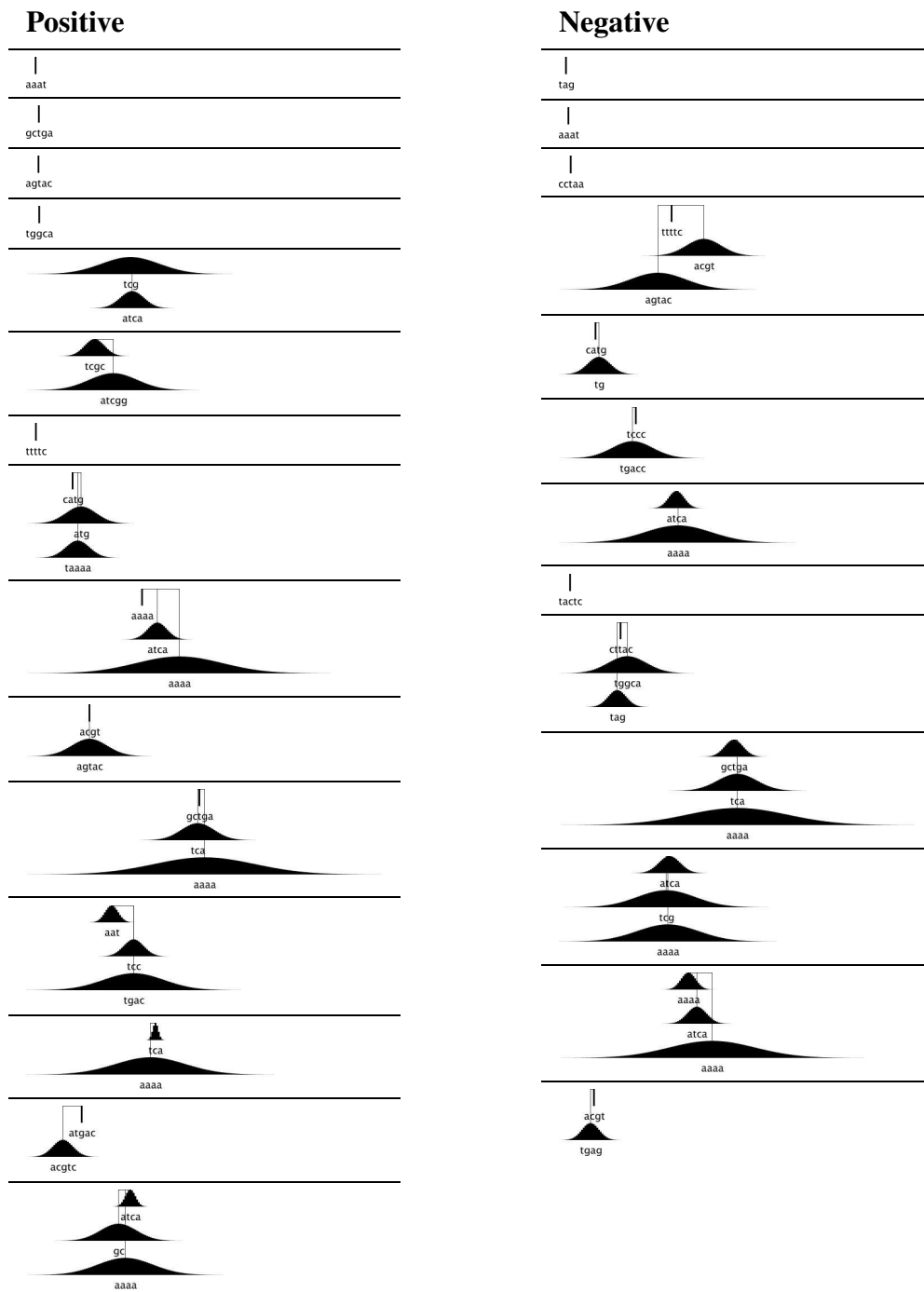
Given a functional predictive system, it is always interesting to investigate methods to increase the system's predictive accuracy. Two different approaches were taken with the hope of generating better EWS promoter models. The first was to use the extended version of EWS, with extra sampling rules to construct “scaffolds” of several PWMs. Potentially, these scaffolds should be able to capture much more information than simple motifs could: two or more PWMs with very narrow position distributions, close to one another, could effectively model a large transcription factor binding site, while slightly broader and more distant distributions might reflect sites for factors which bind cooperatively. Training on the same datasets as before, scaffold-EWS learns significantly more complex models than those seen previously: the total number of basis functions used in the model was typically slightly higher than for motif-EWS, and around half of these basis functions were scaffolds of two or more motifs. One example is shown in figure 4.6. Unfortunately, the additional information did not appear to improve the

predictive power of the models in the chromosome 22 test system.

The second approach takes the view that the training data might be the limiting factor in performance of these models. The set of mouse-human homologous sequences is likely to be a relatively complex data set, and while it clearly contains substantial amounts of promoter sequence, it is also likely to include fragments of coding genes which were missed by Ensembl predictions, RNA genes, UTRs, splicing regulators, and probably other types of functional sequence, as well as some regions which are included purely by chance conservation. In an attempt to make a cleaner data set, the EponineHomol\_1 model was used to select a training set of 4000 sequences with scores over 0.95. This was used as before to train both EWS-motif and EWS-scaffold models, with initial words of length either 5 or 6, which were then tested on chromosome 22. Once again, the ROC curves were very similar to those seen previously. Despite the enrichment of the training data, it was not possible to increase the classification power of this method. This suggests that performance may be limited by a fundamental problem with classification based on fixed-sized windows of sequence. This is considered further in the discussion section.

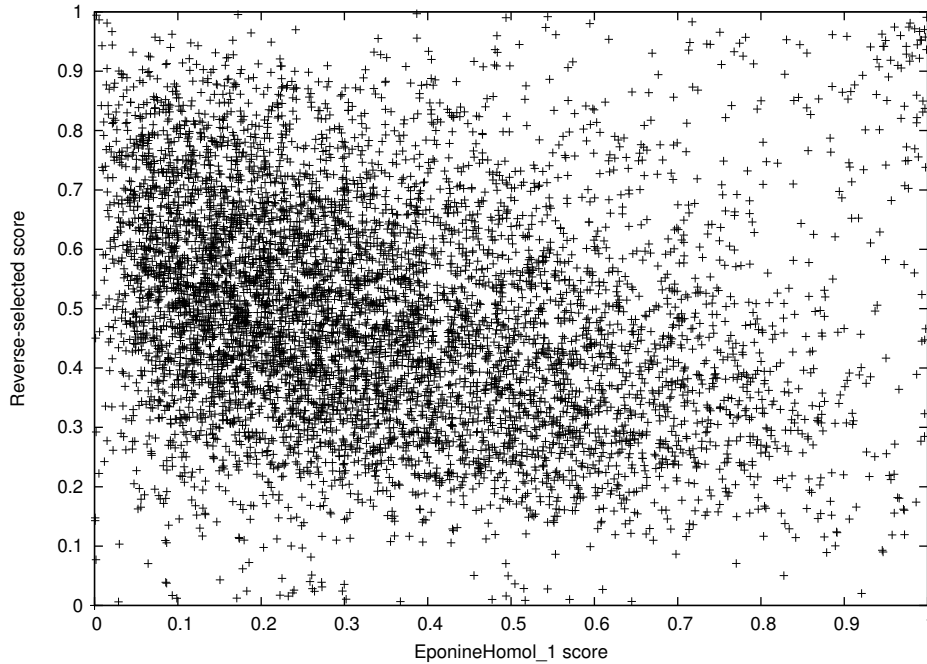
#### **4.4. An attempt to discover a second signal in mouse-human homologies**

Following the success of discovering one signal in conserved noncoding sequences, I attempted to use the same approach to discover more, distinct, signals. To do this, a new training set was prepared, following the same principles as previously, but including only sequences which received low scores ( $<0.25$ ) with the EponineHomol\_1 model. A set of 2000 sequences was selected, henceforth referred to as the reverse-selected set. Models were trained from this set using the same procedure as before. To confirm that this reverse-selection protocol gives models which do detect novel sequences, the models were tested on the same two megabase region as previously. Figure 4.7 shows the scores for a typical reverse-selected model against those for EponineHomol\_1. There is a slight negative correlation between the two axes, so



**Figure 4.6.** A set of basis functions learned by training the EWS-scaffold system on human-mouse homologous sequences. Each cell of this table shows an individual basis function, made up of between one and three sequence motifs.

clearly this model is substantially different from EponineHomol\_1, and indeed any of the models which were trained without using the reverse-selection protocol. One of these models, EponineHomol\_2, was selected for further characterization. A logo view of the motifs in this model can be seen in table 4.3.



**Figure 4.7.** Scores for a model trained from the reverse-selected dataset vs. EponineHomol\_1.

Predicting high scores for different localized sequence windows does not necessarily mean that the model makes very different predictions when considered on a genomic scale. In fact, once again this model appears to localize in promoter regions (see some examples in figure 4.8). The predictions shown in the third panel are somewhat anomalous: the majority of EponineHomol predictions were accompanied by an EponineTSS prediction. When evaluating the EponineHomol\_2 model as a promoter predictor on the pseudochromosome, it gave an accuracy of 42% at a coverage of 50%, less effective than EponineHomol\_1 but still useful, and comparable to EponineTSS\_1.

Perhaps more surprisingly, the set of promoters detected by EponineHomol\_2 is correlated with that found by EponineHomol\_1. As figure 4.8 shows, predictions from the two models

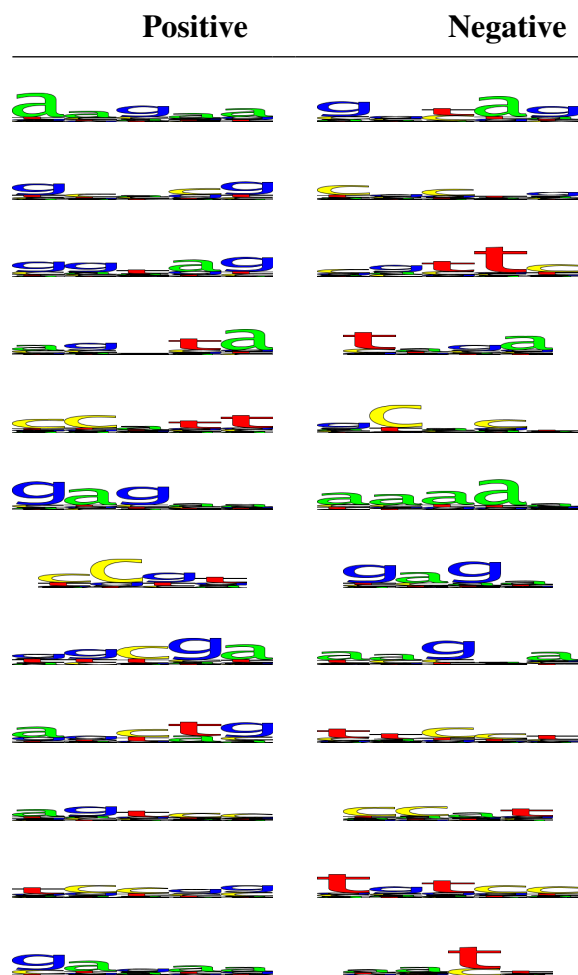


Table 4.3. Logo view of the motifs in the EponineHomol\_2 model.



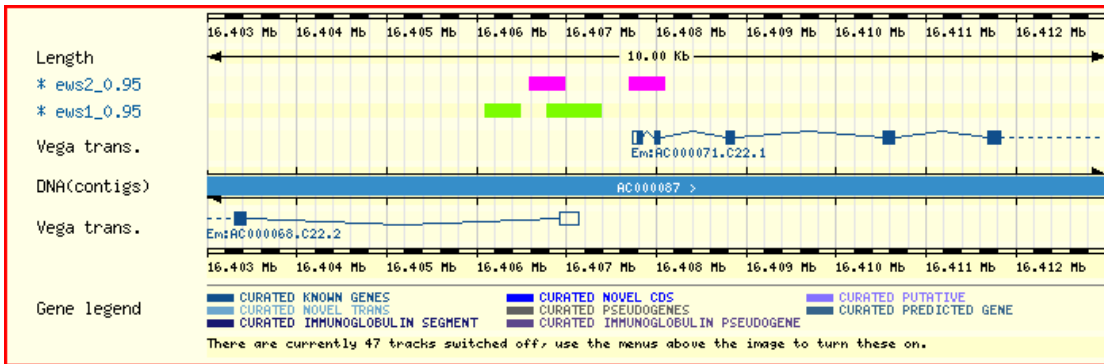
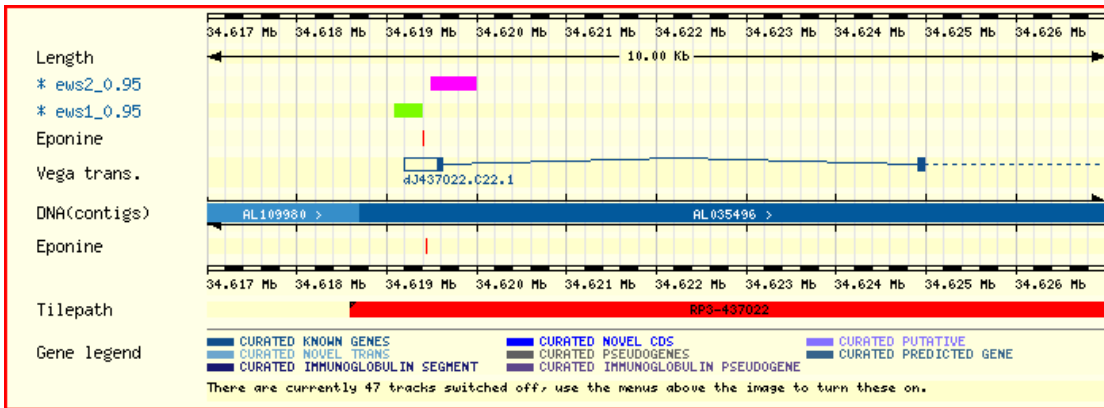
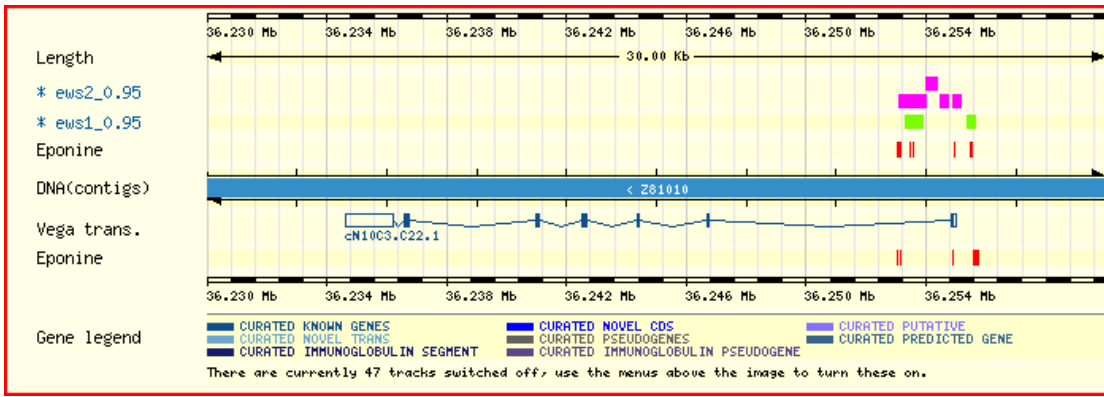
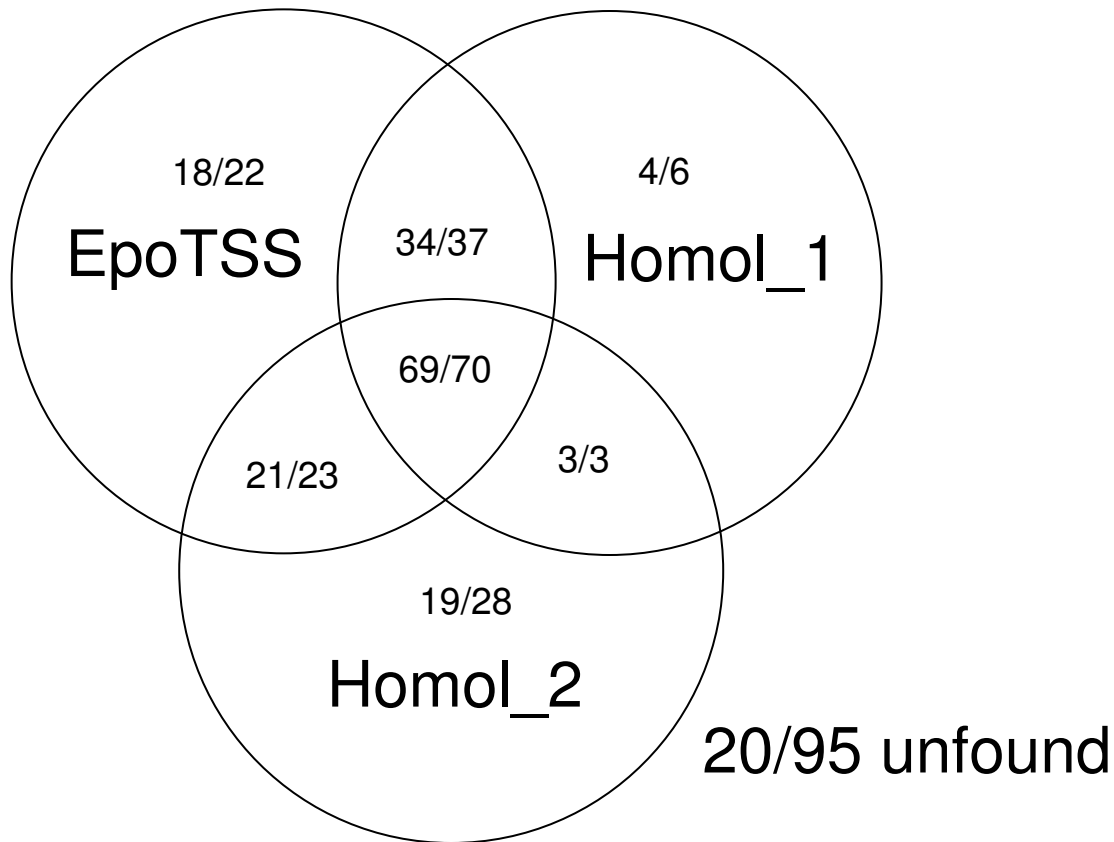


Figure 4.8. Contigview displays showing both EponineHomol\_1 and EponineHomol\_2 predictions.

are often very close together (although generally not precisely co-localized). The strong intersection between the sets of genes found by the two models (and also EponineTSS\_2, and CpG islands) can be seen in figure 4.9. So while it is interesting to find that it is possible to learn a second promoter model, it appears not to be a great help in completing the task of predicting transcription start sites for every gene in the genome.



**Figure 4.9.** Intersection of “correct” promoter predictions from EponineTSS, EponineHomol\_1, and EponineHomol\_2. In each compartment, the first figure indicates the number of start sites which were also detected by a CpG island predictor, while the second figure gives the total number.

#### 4.5. Comparative Genomics Discussion

This method began as an ignorance-driven approach to find interesting signals in pieces of sequence conserved between the human and mouse genomes. But visual inspection of results, followed by comparison with annotated gene starts on the pseudochromosome, makes it clear that the learned models are effective promoter predictors. Based on the criteria of accuracy

and coverage on the pseudochromosome, the models described in this chapter are slightly less effective than the EponineTSS\_2 model from chapter 3. Moreover, they do not give predictions of the actual transcription start site, one of the defining features of EponineTSS. Nevertheless, the results in this chapter are impressive, and potentially significant for the future of genomics. The EponineTSS model represents a “shoulders of giants” approach to promoter prediction: the model is, in effect, distilled information from the 339 EPD sequence entries which were used in the training process. Each of these represents a considerable amount of time, effort, and expense in laboratory work.

The models described in this chapter, however, are based on purely computational analysis of genome sequences. While the difficulty of sequencing a genome should, of course, not be underestimated, a number of genomes have now been completed, and techniques such as preparation of shotgun libraries and assembling the resulting data are now quite well developed. In just a few years, whole-genome shotgun sequencing in bacteria has become an almost routine operation, and it seems hard to imagine that this will not eventually happen for higher organisms, too. To date, most eukaryotic genome sequencing (at least in terms of public and academic projects) has focused on the so-called model organisms, species which have a well-established tradition of experimentation – often (but not always) including laboratory molecular biology work. But once genomes are sequenced in organisms which do not have this tradition and background, it becomes more important to be able to “bootstrap” biological knowledge on the basis of genome sequences. The results in this chapter show this kind of bootstrapping in action, learning information about an organism’s promoters with rather limited *a priori* information. Such techniques will be particularly applicable if sequences are produced for several different species in a previously uninvestigated clade.

This experiment was also interesting in terms of the surprisingly close agreement of promoter-detection results on chromosome 22 between the EponineTSS\_2 and EponineHomol\_1 prediction systems. Since the training of the EponineHomol\_1 model was entirely independent of the databases used to train EponineTSS, it is now possible to discount

the possibility that the set of promoters found by these models is biased by the contents of the training set. The remaining explanations are either a number of the chromosome 22 annotations being significantly truncated at the 5' end, or that many of them are really pseudogenes with no functional promoter, or that promoters can be split into two classes, only one of which is detected by these prediction methods. If the latter hypothesis is true, it seems likely that promoters in the second class have far less in common with one another than those in the first, hence the difficulty training models to detect them.

Given that a rather simple model gave good specificity as a promoter predictor – perhaps surprisingly good, given the rather *ad hoc* nature of the training, and especially the fact that a substantial fraction of the training data was probably non-promoter sequence – it is interesting that various attempts to train better models using longer motifs, sets of motifs on scaffolds, and enriched training sets did not give a further improvement. This suggests that the training may not, in fact, be the limiting factor defining this method's performance. An alternative view is that there are inherent limitations in the use of fixed-size windows. Some promoters might have all their signals focused into a very small region of sequence – perhaps significantly less than 300 bases – while others are likely to be more diffuse. Therefore, there is no ideal window size to catch all the signals of all promoters while not diluting the signals of the smaller promoters. The ideal solution to this issue is to use windows of variable width. This has the added benefit that, given a suitably comprehensive model, it could offer at least some information about the boundaries of promoter regions. One algorithm for scanning a sequence with variable sized windows was proposed in [Byng 2001] (although this used a rather simpler sequence model, just counting occurrences of specific nucleotide words). It would be interesting in the future to combine a variable-window approach to scanning sequences with a sparse Bayesian training method.

The final part of this project was an attempt to peel away the first signal detected by this approach, and retrain the same type of model on a new dataset from which that first signal had been removed. This exercise was successful to the extent that it produced a new model whose

scores were not substantially correlated with the first signal. It is interesting that the second signal to be extracted from this data also appeared to come from near the transcription start site. It is also significant that this second promoter-area model includes motifs containing the CpG dinucleotide, and appears to be making predictions for the same class of CpG-associated promoters. Predictions from the second model often lie very close to, but not necessarily overlapping, the predictions from the first model, suggesting that they are modeling different parts of large promoter regions. This failure to detect large numbers of non-CpG promoters adds more weight to the hypothesis that non-CpG promoters do not form a single group with some set of strong motifs in common with one another. It may, in fact, be necessary to treat each of these individually, rather than identifying stereotypical signals which are found in all promoters.

# Chapter 5. Evolutionary conservation of promoter regions

Like the previous chapter, this part of the project relies on the availability of fairly complete sequences for more than one genome, and also on the basic tools which make it possible to compare them. However, while chapter 4 made a global examination of non-coding regions which were conserved between the mouse and human genome, using a machine learning approach to extract characteristic patterns from them, the analyses in this chapter are far more focused, and concentrate specifically on the conservation of sequence in the promoter regions of already-known genes.

With the availability of high quality sequence data for both the human [IHGSC 2001] and mouse [MGSC 2002] genomes, a lot of interest has turned to the comparison of the two. Automatic gene predictions are available across both genomes – for example, those from the Ensembl project [Hubbard *et al.* 2002]. There is also curated annotation for substantial parts of the human genome [<http://vega.sanger.ac.uk>], but not yet for much of the mouse sequence, so this was not used here. By comparing the predicted protein sequences from these gene sets and identifying reciprocal best hits, likely pairs of orthologous genes – genes which are evolutionarily related and perform the equivalent roles in the two species – can be found. However, it is clear that protein coding sequences are not the only important pieces of information in the genome, so here I consider the evolution and conservation of the promoter sequences which regulate the expression of those genes.

Initial comparative analyses included in the publication of the mouse genome [MGSC 2002] have shown that there is a degree of sequence conservation between human promoters and those of the corresponding mouse gene. The results in chapter 4 offer further

evidence that at least some parts of promoter regions must be conserved between species. Here, the aim was specifically to investigate the conservation of these promoter sequences. There are a number of interesting questions about promoter evolution: to what extent do the functional elements of promoters remain conserved between species? Is conservation between “orthologous” promoters more or less significant than the appearance of shared motifs between promoters for two unrelated genes which happen to require similar expression patterns. And finally, can any clear examples be found where a promoter which drove the expression of one gene in the first species is associated with an entirely different gene in the second.

To investigate this, I first identified a set of promoters pairs, based on orthology of their associated protein-coding regions, then performed an all-against-all promoter sequence comparison and determined how effectively the expected pairs are recovered. In execution, this strategy is quite similar to protocols used to assess protein-alignment methods [Brenner *et al.* 1998]. In this case, the results from an all-against-all comparison will indicate whether the conserved signals are specific to particular promoters, or reflect more general themes which appear in many promoter regions.

In the short history of bioinformatics, a wide range of software has been developed for the comparison and alignment of biological sequences. Most of them have, at their core, some kind of dynamic programming methodology [for review, see Birney 1999], but there are many different implementations, designed and tuned for different purposes. The clearest distinction is between global alignment algorithms (the classic example being the method from [Needleman and Wunsch, 1970]) which attempt to find the best match along the full length of two sequences, and local aligners (e.g. [Smith and Waterman, 1981]) which detect the best-matching portions of two sequences, even if the remainder of the sequence shows no similarity whatsoever. Additionally, some methods, for example the **ssearch** program, perform an exhaustive search of alignment space using a fairly pure implementation of the dynamic programming algorithm, while others such as Blast [Altshul *et al.* 1997] use optimizations – specifically, an initial seeding stage which searches for words which match exactly between

the query and subject sequences – to detect the majority of matches quickly, at the expense of some sensitivity. In general, there is a trade-off of speed for sensitivity, with full dynamic programming at the extreme of good sensitivity, and methods like SSAHA [Ning *et al.* 2001] concentrating entirely on speed. Blast falls between these extremes. These optimizations are normally achieved by first searching for a small cluster of symbols which all match exactly, then extending this in both directions using more sensitive methods. A variety of sequence search methods have been evaluated by the MaxBench system [Leplae and Hubbard, 2002], but this is based on performance when aligning protein sequences, which is not necessarily equivalent to aligning regulatory DNA.

In this case, I chose to use the well-known **blastn** nucleotide search and alignment tool [Altschul *et al.* 1997]. Since this is a local alignment method, and is seeded by words which match exactly between the two sequences, it specifically detects well-conserved blocks, rather than attempting to align large pieces of sequence with only marginal similarity. Its default parameter set, which was used here, further emphasizes the detection of highly conserved regions.

### 5.1. Alignment of promoter regions

Based on human release 8.30a and mouse release 8.3c, the compara databases from Ensembl release 8 [Clamp *et al.* 2003] identified 19914 orthologous gene pairs, based on reciprocal best hits in an all-against-all **blastp** (protein sequence) search. Note that while these results are stored in the same compara database as the nucleotide alignments discussed in chapter 4, the protein comparisons are performed quite independently.

Since the aim was to specifically investigate conservation of promoter regions, it was important to work only with those sequences where it was possible to extract the true promoter region with a good level of confidence, and to avoid contamination of the set with other types of conserved sequence – in particular, unannotated coding regions. I therefore applied a rigorous



procedure to select a high-confidence subset of the data. From the initial comparison set of 19914 orthologous pairs, I selected those cases where both the mouse and the human gene had only a single predicted transcript. This avoided cases with alternate transcripts, which might have additional upstream exons, and also simplified the evaluation schemes applied later in this chapter by removing scope for double-counting of alternate transcripts with starts close to one another. This left 8989 pairs – still a good-sized dataset for large scale evaluation. I then picked the subset where the human sequence had an EponineTSS [Down and Hubbard, 2002 and chapter 3] transcription start site prediction in the interval [-200:+50] relative to the start of the Ensembl-predicted transcript. This gave a set of 2442 pairs. The agreement of Ensembl's (evidence-based) UTR predictions with the computational results from EponineTSS means that these positions should have a very high probability of reflecting true transcription start sites, which in turn means that sequence upstream of this point is likely to primarily have promoter functionality.

For each selected pair, mouse sequence was extracted from -5500 to +500 relative to the Ensembl-predicted gene start, and human sequence was extracted for the 5000 bases upstream from the predicted TSS. Note that the mouse sequence was longer at each end than the human sequence. This means that, when aligning human sequences to mouse sequences, cases where the mouse sequence is slightly longer (perhaps due to a repeat insertion) should still give good alignments, and possible edge effects from the alignment algorithm will be reduced. In the case of two closely-spaced pairs of divergent genes, the extracted windows overlapped. These cases were recorded to ensure that matches to the overlapping regions were not counted as false positives. The human sequences were masked for known repeat sequences using the standard RepeatMasker method [Smit and Green, unpublished], and also for possible extra regions of coding sequence as predicted by the Genscan algorithm [Burge and Karlin, 1997]. These additional coding regions could be alternate first exons from genes with alternate transcription start sites (which may not have been recognized in the Ensembl gene build), or they may be pseudogenes. In either case, it is not appropriate to include them in the sequence comparison

when searching for promoter regions.

Finally, each human sequence was searched against the full set of mouse sequences using the **blastn** [Altshul *et al.* 1997] program (release 2.2.2) with its default alignment parameters:

<b>Word size</b>	11
<b>Score threshold</b>	30
<b>Reward for match (N)</b>	+1
<b>Penalty for mismatch (M)</b>	-3

**Table 5.1.** Default **blastn** parameters used

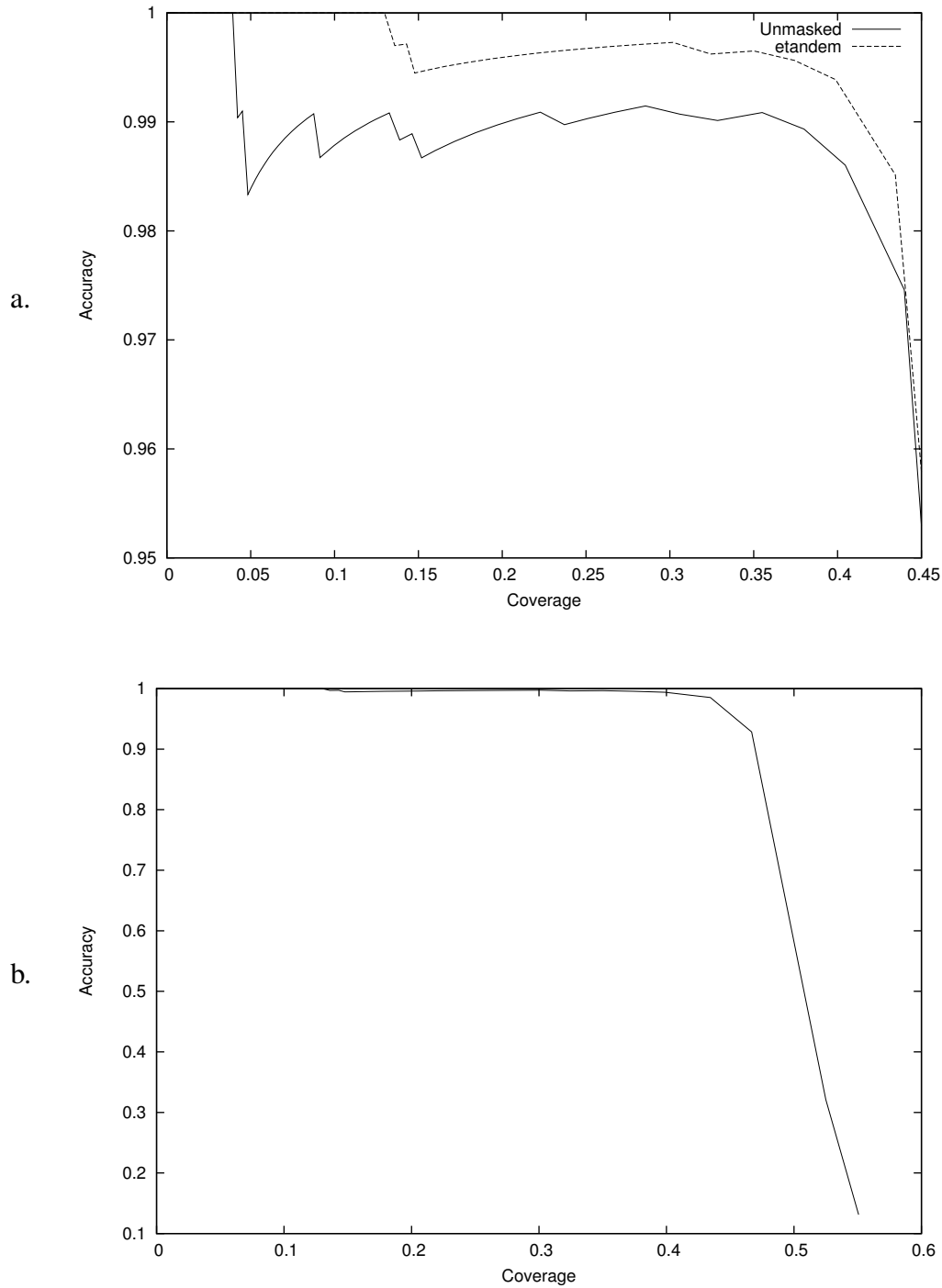
It should be noted that with this parameterization, the magnitude of the penalty for a single-base mismatch is much greater than the reward for a correct match. This means that alignments will not be extended through regions with a low percentage identity. This implies that the algorithm is running with less than maximal sensitivity, but is desirable in this case: the mouse and human genomes are relatively similar to one another, and a degree of background synteny can be seen across many regions, which does not necessarily indicate any functional reason for conservation but simply the fact that the species diverged recently enough that there have not been enough mutations to make related sequences unrecognizable, even where there is little or no evolutionary pressure to maintain a particular sequences. The parameters used here will highlight the most conserved portions of sequence between the two species.

To investigate the efficiency of gene pairing based on promoter sequences, all pairs of human and mouse sequences for which a **blastn** alignment was produced were scored by the total number of bases in aligned regions in the interval [-2000:0] relative to the human TSS, counting alignments to any part of the 6kb mouse sequences. This somewhat unconventional scoring strategy was motivated by the observation that, in many cases, there were several distinct blocks of aligned sequence. There is no justification in the blast scoring scheme for combining the blast scores for these individual blocks by addition, so in order to count the contributions of all the blocks, it was necessary to use an alternative scheme. This approach also made it easy

to calculate scores which only considered alignments of some portion of the sequence, without actually having to re-run the alignment method. The highest-scoring pair in this set had 1308 out of 2001 aligned bases – over 65% of its total length.

Those pairs which matched the previously-defined protein orthology were considered to be “correct” pairings, while all others were considered “incorrect”. In two cases, apparently incorrect pairings were seen because the upstream regions of divergent genes overlapped. These cases were ignored. The relationship between number of correct and incorrect pairs as a score threshold was varied is shown in figure 5.1. It can be seen that promoter-based and protein-based pairings agree extremely strongly up to a coverage of just over 40% (which occurs at a threshold of 50 total aligned bases), beyond which correspondence falls rapidly. This indicates the point at which blocks of detectable similarity become comparable in size with the blocks of similarity which occur either entirely by chance, or because of small, common, functional elements – either individual transcription-factor binding sites, or perhaps small clusters of sites. There are, however, a small number of cases where an incorrect pair is detected with a score much higher than noise. Examination of these sequences revealed a number of low complexity regions (in one example, “gaatgaatgcaggatgcagtgag”) that were not being masked by the **dust** filter built into the Blast software. A more detailed scan for low complexity sequence was made using the **etandem** program from EMBOSS [Rice *et al.* 2000], searching for tandem repeats between 4 and 24 bases long. Masking these regions and realigning the sequences eliminated a number of probable false positive matches, giving the second trace of figure 5.1.

This process of building an all-against-all score matrix then counting those pairs where the score is greater than a specified threshold is closely analogous to the method of single-linkage clustering. In the case of protein sequences, a number of projects have studied clustering of sequences based on pairwise alignment scores. Clusters are often observed, and have been well studied. In this context they are generally called families, and are presumed to reflect evolutionarily related genes. Ensembl gene predictions are allocated to families using the TRIBE-MCL clustering algorithm [Enright *et al.* 2002]. In mouse release 8.3c, the 22,444

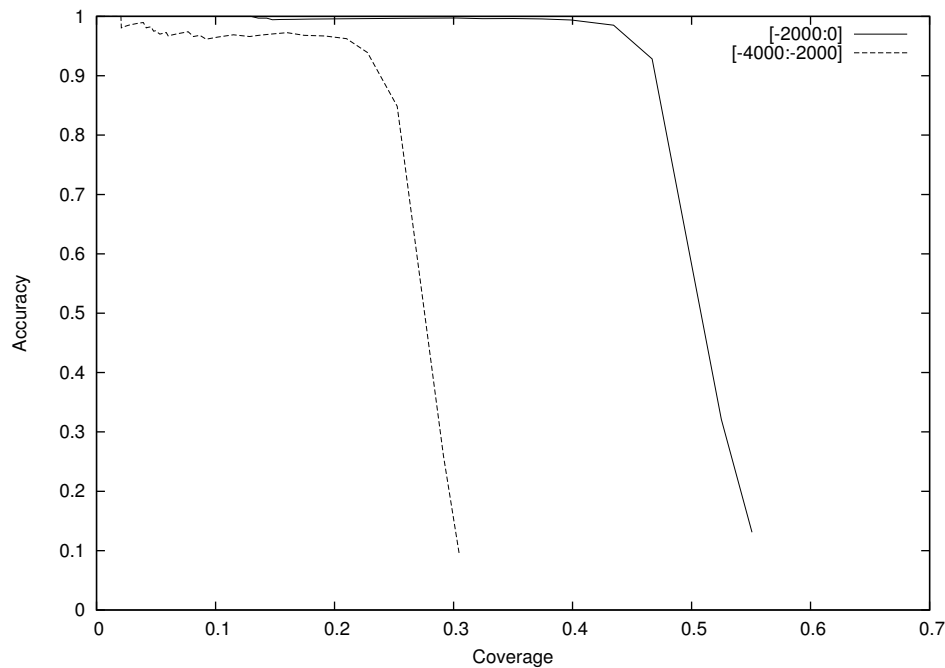


**Figure 5.1.** Agreement of promoter orthology with protein orthology. Panel a shows coverage (proportion of sequences correctly paired) at high levels of accuracy, with and without additional repeat-masking using **etandem**. Panel b shows a wider range of coverages, and only includes the results with **etandem** masking.

predicted genes were assigned to 11,579 families. If strong clusters of promoter sequences could be observed, these would appear as a gradual fall-off of the accuracy value in figure 5.1 as more distantly related clusters join together. Instead, this plot shows very few strong pairs which do not agree with the protein best-hit relationships, right up to the point where the accuracy collapses. The promoter-pairs which do not agree with the protein-defined orthology were compared with the Ensembl family data for the mouse protein set. At a low threshold of 20 aligned bases, 1283 pairs were detected which were not in the protein orthology set, but just one of these was a match to another mouse promoter for a gene in the same family. Based on this result, I suggest that there is little significant large-scale similarity between non-orthologous promoters, even when they are driving the expression of closely related genes.

Without detailed laboratory experimentation, there is currently no available method to accurately determine the boundaries of promoter regions on bulk genomic sequence, therefore these results are open to the criticism that this strategy might be detecting similarities which remain by chance in regions of sequence which are, after all, believed to be related to one another. For the same reasons, it was not possible to build a corresponding control set of sequences which are close to genes but provably not involved in gene regulation: as well as the extent of promoter regions being uncertain, enhancer regions – which are thought to be similar to promoters in terms of their general architecture, and are certainly likely to be conserved between species – can be found throughout the genome. However, making the assumption that the bulk of regulatory elements, particularly in genes which have simple regulation mechanisms, are situated quite close to the transcription start site, it is reasonable to compare pairing results based on homologies in the previously considered window of [-2000:0] (relative to the TSS) against those further upstream in the range [-4000:-2000]. While the second set is very likely to contain some promoter elements, the number is expected to be rather smaller than in the region immediately upstream. The results of this comparison are shown in figure 5.2. This gives the same basic shape of a plateau followed by a rapid collapse in accuracy, but note that the collapse occurs at around half the coverage seen with a window closer to the TSS. It is, of course, believed that promoter

elements still occur (albeit somewhat less frequently) 2kb and more from the TSS, and that this explains the bulk of the 20% of genes which are paired correctly based on the [-4000:2000] window, but this result gives a lower bound on the information contributed by promoter elements. Finally, I note that, although the plateau region shows a generally very good agreement between pairings from protein sequences and [-4000:2000] DNA sequences, the agreement is not quite so strong as that seen for the [-2000:0] region, suggesting that blocks further upstream might be more frequently shared between separate promoters.



**Figure 5.2.** Comparison of orthology in the windows [-2000:0] and [-4000:-2000]

## 5.2. Relationship of promoter alignments to regulatory roles

The extent of alignments between orthologous promoters varies substantially, with many cases having no significant alignment (above **blastn**'s default threshold of 30 bits, which corresponds to a 15-base exact alignment), but others having several large sections of aligning sequence, in some cases covering over 50% of the 5kb region. The set of 2442 alignments include 1312 individual alignment blocks of 100 bases or more. Across this set, there is an average 88% nucleotide identity, with 414 blocks over 90% identity and 81 over 95% identity.

These are extremely well conserved pieces of sequence.

To investigate the significance of this variation, I made use of Gene Ontology (GO) terms [The Gene Ontology Consortium 2000], which are applied to the majority of Ensembl gene predictions on the basis of annotations by the GOA project [Camon *et al.* 2003]. Dividing promoters into groups with high (200 bases or more) and low total numbers of aligning bases, I counted GO terms which were overrepresented in the annotation of high-scoring sequences (tables 5.2 and 5.3). This approach is closely related to the comparison of found and unfound promoters in chapter 3. Some clear correlations can be observed in these tables: at the top of the high-alignment list, and overrepresented by factors of around 2, are genes involved in transcription and developmental processes. At the other extreme, genes taking part in the cell's basic metabolic activities are found predominantly in the short-aligning set. Genes annotated with the GO biological process ontology term for 'cell cycle' are also found predominantly in this set.

Focusing on genes annotated with the process term "transcription, DNA dependent" (figure 5.3) it can be seen that there is still a wide variation in number of aligning bases, but the proportions with larger amounts are consistently higher than that for the gene set as a whole. In particular, 8.0% of these promoters have 1000 bases or more of aligning sequence, compared to 2.4% for the set as a whole.

An more detailed way to view the variation is to look at the parts of the sequence which are actually aligning. A compact representation of this for the "transcription, DNA dependent" genes appears in figure 5.4, with each line of the figure representing one of the 250 upstream regions which matched the this term, and the red blocks indicating regions which show strong similarity to the orthologous mouse sequence. This figure also clearly shows that many genes have multiple aligning blocks, sometimes spaced quite widely apart in the 5kb region. For the cases where a moderate amount of sequence is aligning, this is generally, but not always, concentrated close to the transcription start site, fitting in with the view that promoter elements

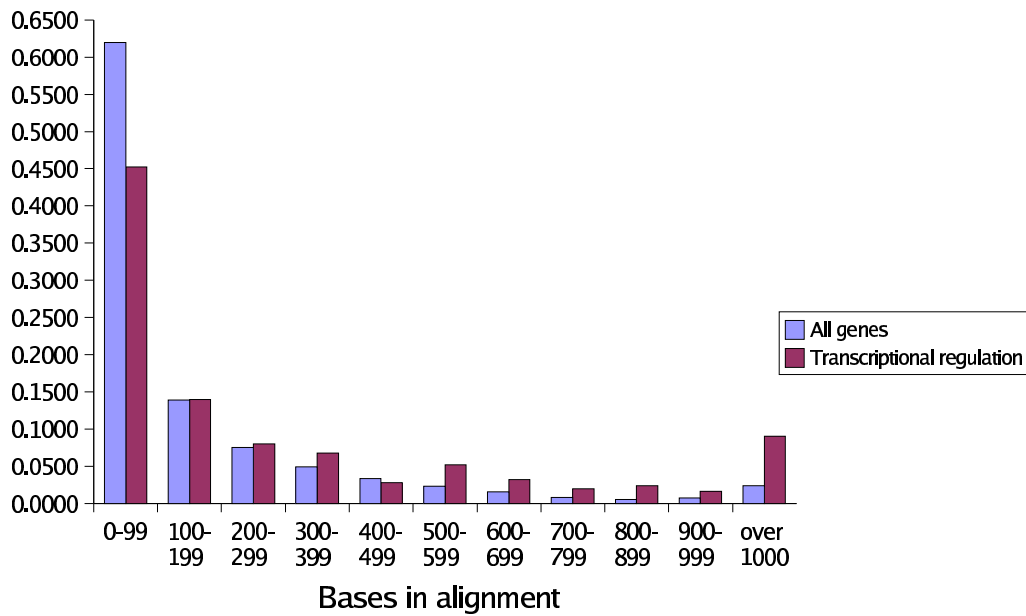
GO term name	Freq. (short)	Freq. (long)	Diff.
transcription, DNA-dependent	0.088	0.173	0.085
transcription	0.088	0.173	0.085
transcription regulator	0.069	0.142	0.073
developmental processes	0.059	0.127	0.067
embryogenesis and morphogenesis	0.044	0.105	0.060
DNA binding	0.080	0.137	0.057
nucleobase, nucleoside, nucleotide and nucleic acid metabolism	0.145	0.195	0.049
histogenesis and organogenesis	0.037	0.086	0.048
cell communication	0.218	0.258	0.039
ligand binding or carrier	0.366	0.404	0.038
nucleic acid binding	0.150	0.188	0.038
ectoderm development	0.017	0.054	0.036
cellular_component	0.558	0.590	0.031
biological_process	0.638	0.663	0.025
transcription factor	0.021	0.045	0.024
cell	0.505	0.528	0.023
Gene_Ontology	0.741	0.763	0.022
molecular_function	0.662	0.683	0.020
receptor	0.040	0.061	0.020
transcription regulation	0.022	0.042	0.019
integral membrane protein	0.063	0.079	0.016
protein kinase	0.036	0.052	0.015
defense response	0.013	0.028	0.015
mesoderm development	0.017	0.032	0.015
membrane	0.180	0.195	0.014
cation channel	0.009	0.023	0.014
metal ion transport	0.017	0.032	0.014
chromosome organization and biogenesis (sensu Eukarya)	0.004	0.018	0.013
transcription, from Pol II promoter	0.023	0.037	0.013
voltage-gated ion channel	0.005	0.018	0.013
response to biotic stimulus	0.026	0.039	0.012
DNA packaging	0.004	0.017	0.012
ion channel	0.013	0.025	0.012
protein modification	0.055	0.068	0.012
nuclear organization and biogenesis	0.006	0.018	0.012

Table 5.2. GO terms which are overrepresented in the long-aligning promoter set.



GO term name	Freq. (short)	Freq. (long)	Diff.
enzyme	0.267	0.209	-0.057
cytoplasm	0.153	0.105	-0.047
biosynthesis	0.064	0.028	-0.035
cell cycle	0.055	0.027	-0.028
oxidoreductase	0.033	0.006	-0.026
hydrolase	0.109	0.085	-0.024
catabolism	0.055	0.034	-0.021
nucleus	0.037	0.018	-0.019
transporter	0.089	0.071	-0.017
RNA metabolism	0.025	0.008	-0.017
mitotic cell cycle	0.027	0.011	-0.015
nucleotide binding	0.107	0.091	-0.015
purine nucleotide binding	0.107	0.091	-0.015
protein transport	0.041	0.027	-0.014
DNA replication and chromosome cycle	0.015	0.001	-0.013
macromolecule catabolism	0.038	0.025	-0.013
amino acid and derivative metabolism	0.017	0.005	-0.012
intracellular	0.432	0.420	-0.012
G-protein coupled receptor protein signaling pathway	0.022	0.010	-0.012
RNA binding	0.018	0.006	-0.012
S phase of mitotic cell cycle	0.013	0.001	-0.011
monovalent inorganic cation transporter	0.013	0.001	-0.011
hydrolase, acting on acid anhydrides	0.045	0.034	-0.011
hydrolase, acting on acid anhydrides, in phosphorus-containing anhydrides	0.045	0.034	-0.011
protein degradation	0.036	0.025	-0.011
cell fraction	0.052	0.040	-0.011
RNA processing	0.017	0.006	-0.011
protein binding	0.059	0.049	-0.010
macromolecule biosynthesis	0.025	0.015	-0.010
inner membrane	0.015	0.005	-0.010
ion transporter	0.019	0.010	-0.009
amino acid metabolism	0.011	0.001	-0.009
RNA splicing	0.011	0.001	-0.009
lipid metabolism	0.026	0.017	-0.009
cation transporter	0.017	0.008	-0.009

**Table 5.3.** GO terms which are overrepresented in the short-aligning promoter set.

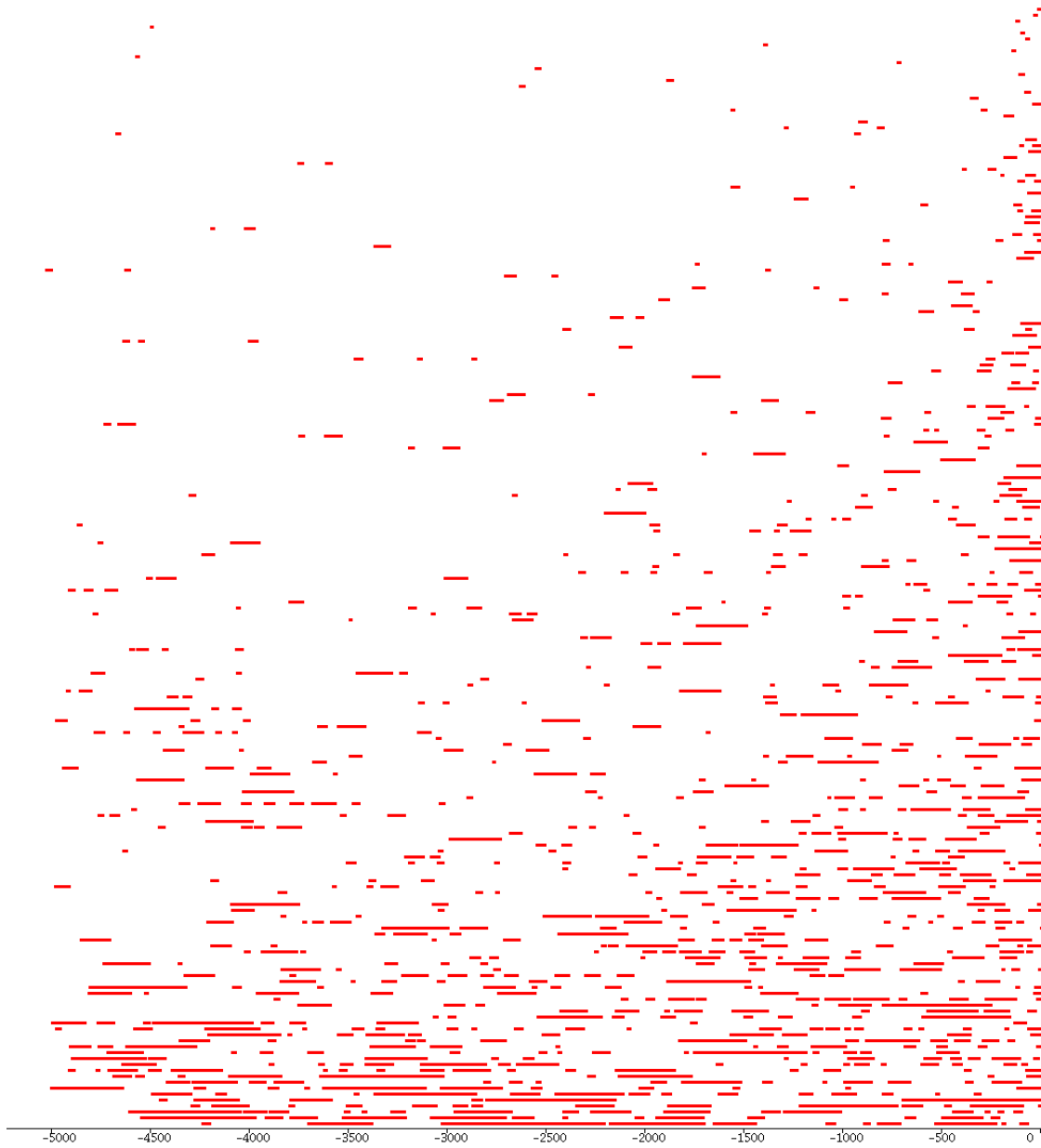


**Figure 5.3.** Histograms of sequences binned by number of bases of promoter sequence included in blastn alignments to the orthologous promoter.

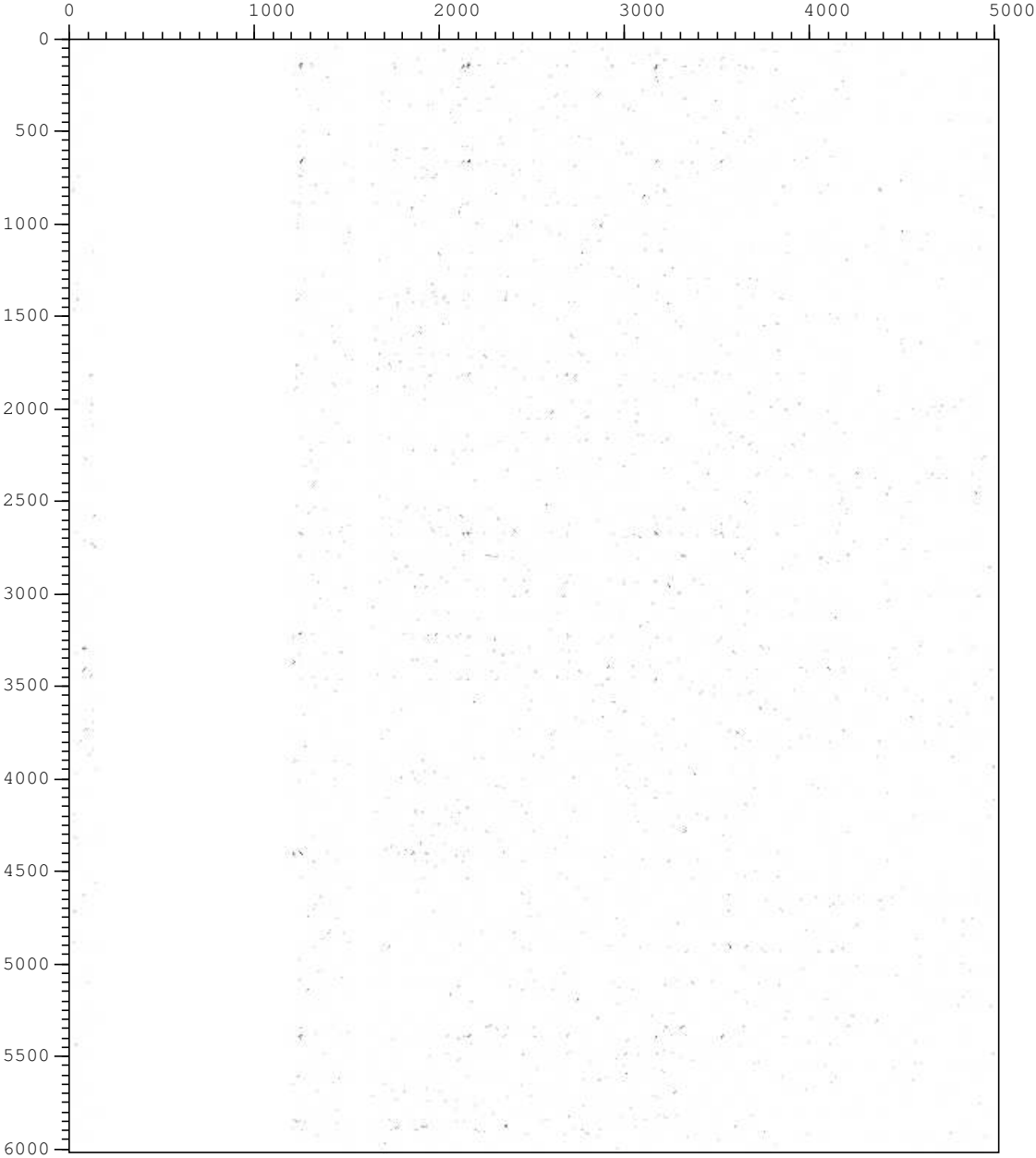
are close to the transcription start site. It is possible that some exceptions to this rule reflect alternative promoters.

Dot plots provide a detailed view of the similarities between two sequences. A number of mouse-human pairs were examined in this way, using the **dotter** tool [Sonnhammer and Durbin, 1995]. Representative examples from the top, middle, and bottom of figure 5.4 appear in figures 5.5, 5.6, and 5.7 respectively. In all cases, the 5kb region of human sequence is represented on the horizontal axis, while 6kb of orthologous mouse sequence is shown on the vertical axis, and dark dots indicate sequence similarity. The most interesting of these plots is figure 5.7, which shows that similarity does, indeed, continue right across the 5kb upstream region. The line of dark points does not quite follow a perfect diagonal, indicating that there have been some minor insertion or deletion events. However, there is no evidence of either major rearrangement or local inversions.

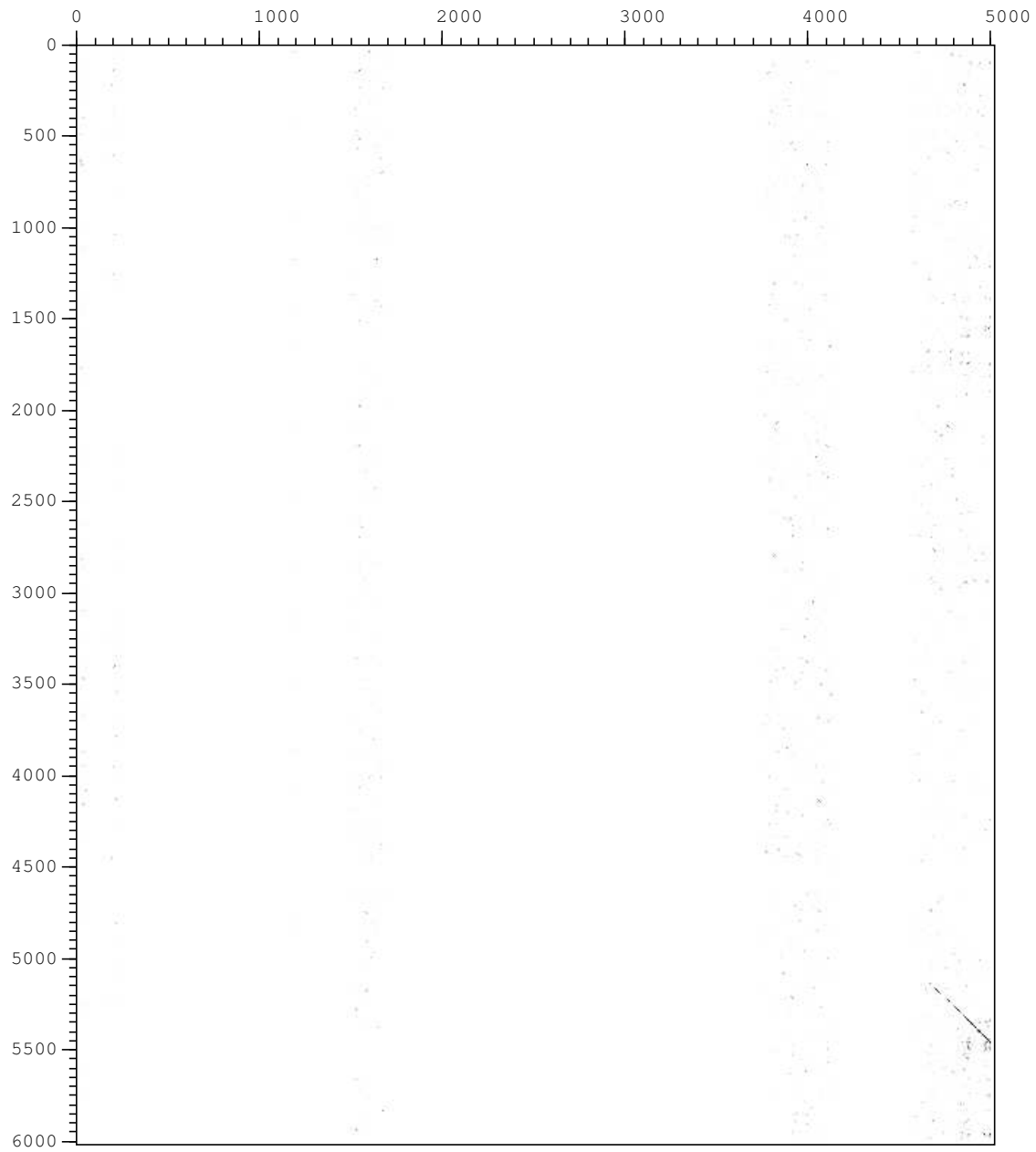
Of course, “transcription, DNA dependent” is a rather broad category (including over 10% of the gene set considered here). The long-aligning subset is dominated by transcription factors, with four homeobox genes in the top ten. At the other extreme, the group of promoters with no



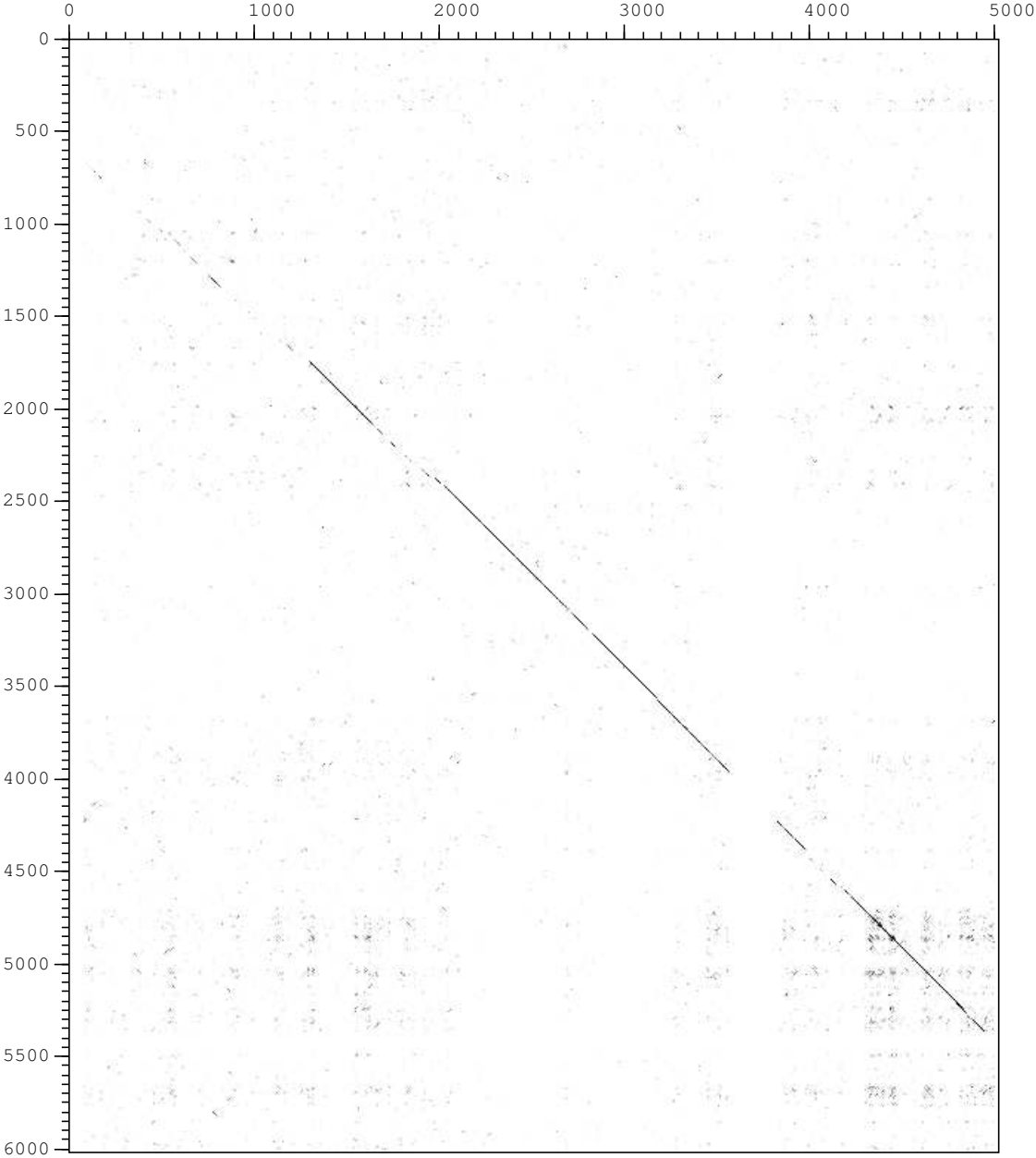
**Figure 5.4.** Aligned regions from 250 transcription-associated genes, sorted by number of aligning bases.



**Figure 5.5.** Dot plot between human and mouse upstream regions with low similarity.



**Figure 5.6.** Dot plot between human and mouse upstream regions with moderate similarity.



**Figure 5.7.** Dot plot between human and mouse upstream regions with strong similarity.

detectable alignment at all includes subunits of *polIII* RNA polymerase.

### 5.3. Discussion of promoter conservation

In this chapter, I have shown that 40% of human promoter sequences have an equivalent with at least one strongly conserved block in the mouse genome which can be detected and distinguished from background noise using the common **blastn** method. Once low-complexity sequences are masked, there are very few cases where a comparably strong match is observed to a sequence other than the promoter of the orthologous gene. I believe that promoter sequences evolve in a manner which is closely linked to the genes they control, and do not see evidence to suggest that a widespread exchange of promoters between genes has occurred, at least in the timescale of the divergence between human and mouse. However, between more distantly related genes, as identified by the TRIBE-MCL clustering, there is little or no identifiable promoter similarity. This suggests that gene duplication (presumably followed by a change in either function or expression pattern of one copy) is likely to be accompanied by fairly radical changes in the promoters. This is interesting but not entirely surprising, since, following a gene duplication, if both copies are to remain active, one must rapidly change to fill some alternative role in the organism, otherwise there is unlikely to be any selection pressure against an accumulation of mutations in one of the copies, eventually leaving a pseudogene. There do not appear to be any large contiguous conserved elements – for example, very complex regulatory modules – which appear in multiple promoters, since these would appear as non-orthologous pairings in this analysis. This is not, of course, inconsistent with the existence of short conserved elements (such as individual transcription factor binding sites, or perhaps even small modules consisting of two or three such sites).

Given the fall-off of similarity when looking further from the transcription start site, I am confident that the bulk of the alignments produced by this method reflect *bona fide* functional sequence, rather than background synteny. The observed correlation between alignment length

and gene function adds further support to this view.

As previously mentioned, there is currently no purely computational method to delineate promoter regions (as opposed to transcription start sites) – while PromoterInspector and the EponineHomol models from chapter 4 do predict regions rather than specific TSS points, given the methods used for training these models there is no reason to think that the predicted region accurately matches the region of sequence which has biological regulatory significance. Similarly, experimental delineation of full promoter regions cannot yet be performed in a high-throughput way across the genome. However, the extremely high level of nucleotide identity in many of the blocks detected by this approach, combined with the increased number of conserved blocks closely upstream of the transcription start site, gives a strong indication that these blocks represent functionally important elements – presumably promoter signals. This seems to be further reinforced by the results from chapter 4 which show that a promoter signal can be learned from a random selection of blocks of mouse-human similarity. This means that these high-similarity blocks would seem to be good initial targets for further computational work to discover individual regulatory motifs and modules and learning more about regulatory mechanisms.

The utility of having an approach to highlight promoter regions is not confined to computational methods. One currently promising experimental method of investigating DNA-protein interactions in a relatively high-throughput manner is ChIP-on-chip: chromatin immunoprecipitation followed by hybridization with probes on a DNA microarray [Ren *et al.* 2000]. While it is possible to place a large number of probes on a microarray, reasons of cost and convenience mean it is still important to be selective. From figure 5.4, it is clear that even among the promoters for one family of genes, some only have large conserved blocks close to the transcription start site, while others show conservation right across the 5kb region studies here. In fact, re-aligning longer sequences for a few individual promoters suggested that some of these might extend 8kb or more upstream of the TSS. Clearly, it is worth including probes much further upstream for these genes.



And additional interesting observation from this study is that there is a fairly clear correlation between the length of aligning sequence in a promoter region and the function of the gene regulated by that promoter. I can envisage two possible causes for this. Firstly, in order to fulfill their roles, some genes require many more regulatory “inputs” than others – in a graphical view of a regulatory network [Pilpel *et al.* 2001], some genes will appear as hubs with many connections, while others will appear as leaf nodes. Large numbers of inputs seem likely to correspond with large regions of functional promoter sequence, containing many different transcription factor binding sites. Secondly, even within the functional regions, some promoters may be more labile than others – radical changes can occur without compromising the basic viability of the organism. Both these factors may be significant. I note that cell-cycle genes tend to have short-aligning promoters, yet they represent a well-conserved biological process, where correct regulation is important for survival. However, the regulation of these genes may be quite simple: they need only be expressed in response to very specific signals at one particular point in the cell cycle. In budding yeast, a set of nine cell-cycle regulators have been identified, and most cell-cycle-regulated genes (including these regulators themselves) only respond to a small subset of these regulators [Simon *et al.* 2001]. If the situation in vertebrates is similar, then it seems likely that a relatively compact promoter, with a small number of conserved transcription factor binding sites, could be sufficient to express the required regulatory logic. In conclusion, I suggest that promoter size gives a good indication of a gene’s “regulatory complexity”, and gives some prediction of the gene’s function.

Overall, I believe comparative genomics presents interesting opportunities for research into promoters and regulatory regions, especially in terms of defining the boundaries of functional regions so that they can be studied in more detail with other methods, such as the learning techniques from previous chapters. In the future, it will be interesting to consider more sensitive methods for aligning promoters. However, simply increasing the sensitivity is not a panacea, since it increases the risk of detecting similarities between sequences where there is not actually any functional reason for conservation. There has been some interest in methods

to distinguish between alignments covering regulatory regions and alignments between regions of change conservation. In [Elnitski *et al.* 2003], a number of classification methods are used to distinguish between a training set of known regulatory elements and some conserved neutral sites (actually ancient transposons). Some of these methods are effective, but it seems likely that they will only be applicable to relatively long aligned regions – not the short, very strongly conserved blocks discovered by the **blastn** alignments here. The biggest benefits from this approach may be for identifying regulatory possible regulatory regions – such as enhancer elements – far from known genes and transcription start sites, rather than in the analysis is proximal promoters.

Another possible direction of research is to investigate promoter-specific sequence comparison methods, which might give superior performance to generic alignment techniques: for instance, conventional aligners would not give good a good score for a pair of sequences where localized rearrangements or small inversions had taken place, yet these two sequences might actually include the same repertoire of transcription factory binding sites, and have similar effectiveness and specificity as promoters. In defense of “simple” alignment algorithms, however, inspection of the dot plots accompanying this chapter, and several similar plots not shown here, did not show strong evidence for rearrangements or inversions. Finally, it may be that mouse-human comparisons do not represent the optimal evolutionary “distance” for detecting promoter regions. As more genomes are completed, other species may take over as targets for regulatory comparative genomics, or perhaps comparisons of more than two species will be used to improve the confidence of the results. Data from the ENCODE project, which will select various regions from the genome then sequence their equivalents in a large number of different vertebrates [ENCODE, <http://www.genome.gov/Pages/Research/ENCODE/>], should provide a good testbed to determine the most informative pairs of genomes to compare, and the added value of considering more than two species at once.

## Chapter 6. Conclusions

In this project, I developed a number of methods which give information about the location of promoter regions and transcription start sites in mammalian genomes. The EponineTSS method from chapter 3, based on a novel machine learning approach, offers state-of-the-art performance in predicting promoters, including information about the transcription start site, and is now used as part of the Ensembl genome annotation pipeline. Making use of a rather different source of information, in chapter 5 I have shown that simple methods of comparative genomics between the human and mouse genomes can reveal highly conserved blocks of sequence – including blocks of 100 bases or more with 95% nucleotide identity – which seem likely to correspond to functional promoter regions. These two methods are strongly complementary: EponineTSS can be used to predict transcription start sites, which give an indication of the 3' end of the promoter, and then information from comparative genomics can be used in conjunction with this to indicate how far upstream the functional region is likely to stretch. This combined approach meets my objective of providing promoter detection methods which can be used to support more detailed future analyses of promoter signals.

A third strand of this project, covered in chapter 4, was also related to human-mouse comparative genomics but took a very different approach, considering the complete set of non-coding similarities between the two species. Interestingly, but not entirely surprisingly, this also led me to a promoter signal, and provided a second predictive method which could be used for scanning genomes, albeit with somewhat lower accuracy than EponineTSS, and no direct predictions of the transcription start sites. While the advantages of EponineTSS mean that the models trained from homologies will probably not be useful in themselves as predictive tools, the principle that it is possible to learn such a predictor from raw comparative data is intriguing, and might prove helpful in “bootstrapping” genomic knowledge if sequencing is carried out in

clades of organisms which don't have a prior tradition of genetic or molecular biology research, and consequently no body of knowledge to draw on when annotating their genomes. In addition, I still believe that with improvements – perhaps including automatic choice of windows rather than using a fixed window of arbitrary size – the sensitivity of this method might be increased sufficiently that, once all the promoter regions have been stripped out of the training set, it might identify other types of functional non-coding region in the genome.

When evaluating the EponineTSS method, I found that it detected primarily CpG-enriched promoters. Moreover, the set of promoters detected overlapped strongly with the set detected by another computational method, PromoterInspector. This could be explained away by the fact that both EponineTSS and PromoterInspector were trained on promoters from the EPD database, and there might be some bias in the entries of this database. However, it was subsequently found that the EponineHomol 1 and 2 predictors, which were not trained using any information from EPD, also preferentially detected the same subset of promoters. This suggests that there is one group of promoters which are based around a common set of core signals (captured in the EponineTSS model). There also appear to be other types of promoter which do not follow this pattern. Attempts were made with both the EAS and EWS model families to train models on a dataset which had these core promoter sequences removed, in the hope that this might reveal a second type of core promoter. Neither of these attempts produced a model which could be usefully applied to detecting a distinct set of promoters, which suggests that there may not be an single alternative core promoter, but that each of these atypical promoters is different. As discussed in section 3.5.2, the distribution of the core promoters is not entirely uniform: metabolic enzymes, protein kinases, and transcription factors are all likely to have common core promoters, while the majority of receptors and immune system components are apparently transcribed from atypical promoters. This distinction illustrates one more level of complexity in the regulatory story.

To support the sequence analysis requirements of this project, I investigated the recently developed field of Sparse Bayesian Learning algorithms, and developed a practical implementation of a Sparse Bayesian binary classification system. This can be applied either as a

one-step method similar to the previously described Relevance Vector Machine [Tipping 2000], or using a novel incremental approach which offers a pragmatic method of sparse learning from very large sets of basis functions and even, given suitably correlated basis function, a possibly infinite family of basis functions. This library code proved to be a good basis for the learning applications described in chapters 3 and 4, but it is a general-purpose implementation and has been applied to other tasks, including analysis of microarray gene expression data. In addition, the Eponine Anchored Sequence system from chapter 3 has been applied as-is to other problems such as prediction of transcription termination sites [A. Ramadass, personal communication], and I hope that it will find many further applications in the future.

Finally, during the course of this project I had many opportunities to work on the infrastructure of genomics and sequence analysis. Throughout the time, I was involved in the development of the BioJava toolkit (section 1.5.1), which provided a foundation for all the applications and experimental programs which I developed here. I was also involved in the early development of the DAS protocol [Dowell *et al.* 2001], and wrote a BioJava-based server for this protocol [Down and Pocock 2001]. I used DAS on a number of occasions to view results from my analysis methods in a genomic context. For some parts of this work, I made direct use of Ensembl's relational databases, either by direct SQL queries or using the `bj-ensembl` library.

At the conclusion this project, I remain interested in the mechanisms of transcriptional regulation, and how regulatory sequences can be decoded *in silico*. Understanding transcriptional regulation would be relatively easy if we could firstly accurately determine the structure of every protein produced by the genome, and then predict interactions between those proteins and the nucleic acids. Realistically, though, the current state of the art in structure prediction still leaves much to be desired [Moult *et al.* 2001], and while Richard Lavery's group have had some successes in physical modeling of DNA-protein complexes (see, for example, [Harvey *et al* 2003]), their methods currently require a high-resolution structure of the full complex before predictions of binding specificities can be made. Therefore, to further our understanding of promoters and gene regulation, it seems important to continue direct study of

the sequences themselves, and also take advantage of any experimental techniques which can give better information about gene expression patterns and the factors which influence them.

One such experimental approach is direct study of which proteins bind to which regions of DNA sequence. Ideally, the results from high throughput experiments of this kind could give the same kind of information as simulation of DNA-protein interactions, and have the advantage that they can be performed with real chromatin, rather than the idealized situation of a single protein interacting with a “naked” DNA helix. The results of my comparative promoter analysis are currently being considered in planning a new experimental study, using a combination of chromatin immunoprecipitation and microarray techniques [Ren *et al.* 2000] to localize the *in vivo* binding locations of proteins to DNA. A comparison approach based on that described here will be used as an indicator of how much sequence upstream of each gene under consideration should be tiled onto the array. Since each probe on the array requires an individual PCR reaction with custom primers [D. Vetric, personal communication], optimizing the choice of probes could help to maximize the number of genes which can be studied for a given budget.

Returning to computational methods, this project has introduced some powerful tools for classifying sequence data. Both the EWS and EAS models are flexible sequence analysis techniques which have the advantage that the motifs learned by the model are immediately visible for user examination. It may prove possible to apply these, or similar, methods to classification of promoters based on expression patterns, thus learning the signals which confer those particular patterns. An obvious extension for this purpose is to use a multi-class variant of the training algorithm, so that a number of different patterns can be considered in a single training run. A more radical variant of this approach, and one which I am interested in developing, is to extend this approach to unsupervised machine learning, where patterns are discovered in previously unlabeled data. In this case, the aim is to learn both the “labeling” (which groups of genes share either complete or – more likely – partial expression patterns) and the sequence-based signals which regulate this. This is a demanding problem, with no off-the-shelf solution. However, if promoter sequences are treated as a “mixture” of sequence

motifs or regulatory modules, identifying the set of modules present in a large set of promoters shows some similarities to the well-known problem of Independent Component Analysis (see, for example, [Miskin 2000]). I believe it will be possible to adapt an approach analogous to ICA to conceptually de-mix promoter regions.

# References

M. D. Adams, J. M. Kelley, J. D. Gocayne, M. Dubnick, M. H. Polymeropoulos, H. Xiao, C. R. Merrill, A. Wu, B. Olde, R. F. Moreno, et al.. Complementary DNA sequencing: expressed sequence tags and human genome project. *Science* **252**, 1651–1656 (1991).

S. F. Altschul, T. L. Madden, A. S. Alejandro, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* **25**, 3389–3402 (1997).

S. Asakawa, K. Tsunematsu, A. Takayanagi, T. Sasaki, A. Shimizu, K. Kawasaki, A. J. Mungall, S. Beck, S. Minoshima, and N. Shimizu. The Genomic Structure and Promoter Region of the Human Parkin Gene. *Biochemical and Biophysical Research Communications* **286**, 863–868 (2001).

S. Audic and E. Béraud-Columb. Detection of eukaryotic promoters using Markov transition matrices. *Computation and Chemistry* **21**, 223-228 (1997).

T. L. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Seconding International Conference on Intelligent Systems for Molecular Biology*, pages 28–36, 1994.

Y. Barash, G. Elidan, N. Friedman, and T. Kaplan. Modeling dependencies in protein-DNA binding sites. In *Proceedings of RECOMB*, 2003.

A. Barberis and L. Gaudreau. Recruitment of the RNA polymerase II holoenzyme and its implications in gene regulation. *Biological Chemistry* **379**, 1397–1405 (1998).

A. Bateman, E. Birney, L. Cerutti, R. Durbin, L. Ewinger, S. R. Eddy, S. Griffiths-Jones, K. L. Howe, M. Marshall, and E. L. Sonnhammer. The Pfam Protein Families Database. *Nucleic Acids Research* **30**, 276–280 (2002).



- E. Birney. *Sequence alignment in bioinformatics*. Ph.D. thesis, Sanger Centre, 1999.
- C. M. Bishop, M. E. Tipping. Variational Relevance Vector Machines. In *Proceedings on the 16<sup>th</sup> Conference in Uncertainty in Artificial Intelligence*, pages 46–53, 2000.
- B. J. Blencowe. Exonic splicing enhancers: mechanism of action, diversity and role in human genetic diseases.. *Trends in Biochemical Science* **25**, 106–110 (2000).
- M. Brandeis, M. Ariel, and H. Cedar. Dynamics of DNA methylation during development. *Bioessays* **15**, 709–713 (1993).
- S. E. Brenner, C. Chothia, and T. J. P. Hubbard. Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. *Proceedings of the National Academy of Science* **95**, 6073–6078 (1998).
- J. E. Brownell, J. Zhou., T. Ranalli., R. Kobayashi, D. G. Edmondson, S. Y. Roth, C. D. Allis. *Tetrahymena* histone acetyltransferase A: a homolog to yeast Gcn5p linking histone acetylation to gene activation.. *Cell* **84**, 843–851 (1996).
- P. Bucher. Weight matrix descriptions of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences.. *Journal of Molecular Biology* **212**, 563–578 (1990).
- C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology* **268**, 78–94 (1997).
- Martyn Byng. *A statistical model for locating regulatory regions in novel DNA sequences*. Ph.D. thesis, University of Reading, 2001.
- E. Camon, M. Magrane, D. Barrel, D. Binns, W. Fleishmann, P. Kersey, N. Mulder, T. Oinn, and R. Apweiler. The Gene Ontology Annotation (GOA) project: implementation of GO in SWISS-PROT, TrEMBL, and InterPro. *Genome Research* **13**, 662–672 (2003).

M. F. Carey. Transcriptional activation. A holistic view of the complex. *Current Biology* **5**, 1003-1005 (1995).

The *C. elegans* Sequencing Consortium. Genome Sequence of the Nematode *C. elegans*: A Platform for Investigating Biology. *Science* **282**, 2012–2018 (1998).

A. Chess, I. Simon, H. Cedar, and R. Axel. Allelic Inactivation Regulates Olfactory Receptor Gene Expression. *Cell* **78**, 823–834 (1994).

M. Clamp, D. Andrews, D. Barker, P. Bevan, G. Cameron, Y. Chen, L. Clark, T. Cox, J. Cuff, V. Curwen, T. Down, R. Durbin, E. Eyraas, J. Gilbert, M. Hammond, T. Hubbard, A. Kasprzyk, D. Keefe, H. Lehvaslaiho, V. Iyer, C. Melsopp, E. Mongin, R. Pettett, S. Potter, A. Rust, E. Schmidt, S. Searle, G. Slater, J. Smith, W. Spooner, A. Stabenau, J. Stalker, E. Stupka, A. Ureta-Vidal, I. Vastrik, and E. Birney. Ensembl 2002: accommodating comparative genomics. *Nucleic Acids Research* **31**, 38–42 (2003).

J. Corden, B. Wasylyk, A. Buchwalder, P. Sassone-Corsi, C. Keding, P. Chambon. Promoter sequences of eukaryotic protein-coding genes. *Science* **209**, 1406–1414 (1980).

C. Crasto, M. S. Singer, and G. M. Shepherd. The olfactory receptor family album. *Genome Biology* **2**, 1027.1–1027.4 (2001).

D. di Bernardo, T. A. Down, and T. J. P. Hubbard. aReNA: Detection of conserved secondary structures in multiple alignments. *Bioinformatics* (Submitted).

R. D. Dowell, R. M. Jokerst, A. Day, S. R. Eddy, L. Stein. The Distributed Annotation System. *BMC Bioinformatics* **2**, 7 (2001).

T. A. Down, and T. J. P. Hubbard. Computational Detection and Location of Transcription Start Sites in Mammalian Genomic DNA. *Genome Research* **12**, 458–461 (2002).

T. A. Down and M. R. Pockock. Building a Distributed Annotation System. In *Proceedings of NETTAB*, 2001.

- I. Dunham, N. Shimizu, B. A. Roe, S. Chissoe, A. R. Hunt, J. E. Collins, R. Bruskiewich, D. M. Beare, M. Clamp, L. J. Smink, R. Ainscough, J. P. Almeida, A. Babbage, C. Bagguley, J. Bailey, K. Barlow, K. N. Bates, O. Beasley, C. P. Bird, S. Blakey, A. M. Bridgeman, D. Buck, J. Burgess, W. D. Burrill, K. P. O'Brien, *et al.*. The DNA sequence of human chromosome 22. *Nature* **402**, 489–495 (1999).
- R. Durbin, S. Eddy, A. Krogh, G. Mitchison. *Biological Sequence Analysis*. Cambridge University Press, 1998.
- S. R. Eddy. Non-coding RNA genes and the modern RNA world. *Nature Reviews Genetics* **2**, 919–929 (2001).
- L. Elnetski, R. C. Hardison, J. Li, S. Yang, D. Kolbe, P. Eswara, M. J. O'Connor, S. Schwartz, W. Miller, and F. Chiaromonte. Distinguishing Regulatory DNA From Neutral Sites. *Genome Research* **13**, 64–72 (2003).
- The ENCODE project. Encyclopedia of DNA elements (ENCODE). URL <http://www.genome.gov/Pages/Research/ENCODE/>.
- A. J. Enright, S. Van Dongen, C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research* **30**, 1575–1584 (2002).
- R. C. Périer, V. Praz, T. Junier, C. Bonnard, P. Bucher. The Eukaryotic Promoter Database. URL <http://www.epd.isb-sib.ch/>.
- J.W. Fickett and A.C. Hatzigeorgiou. Eukaryotic Promoter Recognition. *Genome Research* **7**, 861–878 (1997).
- C. Fields, M. D. Adams, O. White and J. C. Venter. How many genes in the human genome?. *Nature Genetics* **7**, 345–346 (1994).
- T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray

expression data. *Bioinformatics* **16**, 906–914 (2000).

D. J. Galas and A. Schmitz. DNase footprinting: a simple method for the detection of protein-DNA binding specificity. *Nucleic Acids Research* **5**, 3157–3170 (1978).

M. Gardiner-Garden and M. Frommer. CpG islands in vertebrate genomes. *Journal of Molecular Biology* **196**, 261–282 (1987).

M. R. Green and R. G. Roeder. Definition of a Novel Promoter For the Major Adenovirus-associated Virus mRNA. *Cell* **22**, 231–242 (1980).

H. Grosshans and F. J. Slack. Micro-RNAs: small is plentiful. *Journal of Cell Biology* **156**, 17–21 (2002).

W. N. Grundy, T. L. Bailey, C. P. Elkan, M. E. Baker. Meta-MEME: Motif-based Hidden Markov Models of Biological Sequences. *Computer Applications in the Biosciences* **13**, 397–406 (1997).

G. Slater. Exonerate. URL <http://www.ebi.ac.uk/~guy/exonerate/>.

S. C. Harvey, C. Wang, S. Teltchea, and R. Lavery. Motifs in Nucleic Acids: Molecular Mechanics Restraints for Base Pairing and Base Stacking. *Journal of Computational Chemistry* **24**, 1–9 (2003).

M. Hayashi. A DNA-RNA complex as an intermediate of in vitro genetic transcription. *Proceedings of the National Academy of Sciences* **54**, 1736–1743 (1965).

R. Holliday. The inheritance of epigenetic defects. *Science* **238**, 163–170 (1987).

W. Hoschek *et al.*. The Colt Distribution – Open Source Libraries for High Performance Scientific and Technical Computing in Java. URL <http://tilde-hoschek.home.cern.ch/~hoschek/colt/index.htm>.

The Ensembl Project. The Ensembl Trace Repository. URL <http://trace.ensembl.org/>.

T. J. P. Hubbard, D. Barker, E. Birney, G. Cameron, Y. Chen, L. Clark, T. Cox, V. Curwen, T. A. Down, R. Durbin, E. Eyras, J. Gilbert, M. Hammond, L. Huminiecki, A. Kasprzyk, H. Lehvaslaiho, P. Lijnzaad, C. Melsopp, E. Mongin, R. Pettett, M. Pockock, S. Potter, A. Rust, E. Schmidt, S. Searle, G. Slater, J. Smith, W. Spooner, A. Stabenau, J. Stalker, E. Stupka, A. Ureta-Vidal, I. Vastrik, and M. Clamp. The Ensembl genome database project. *Nucleic Acids Research* **30**, 38–41 (2002).

The Genome International Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature* **409**, 860–921 (2001).

T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems*, 1999.

J. T. Kadonaga. Eukaryotic Transcription: An Interlaced Network of Transcription Factors and Chromatin-Modifying Machines. *Cell* **92**, 307–313 (1998).

J. Kawai, A. Singagawa, K. Shibata, M. Yoshino, M. Itoh, Y. Ishii, T. Arakawa, A. Hara, Y. Fukunishi, H. Konno *et al.*. Functional annotation of a full-length mouse cDNA collection. *Nature* **409**, 685–690 (2001).

W. J. Kent and A. M. Zahler. Conservation, regulation, synteny and introns in a large-scale *C. briggsae* – *C. elegans* genomic alignment. *Genome Research* **10**, 1115–1125 (2000).

S. Knudsen. Promoter2.0: for the recognition of polII promoter sequences. *Bioinformatics* **15**, 356–361 (1999).

R. Kodandapani, F. Pio, C. Z. Ni, G. Piccalli, M. Klemsz, S. McKercher, R. A. Maki, and K. R. Ely. A new pattern for helix-turn-helix recognition revealed by the PU.1 ETS-domain-DNA complex. *Nature* **380**, 456–460 (1996).

R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Machine Learning: Proceedings of the Nineteenth International Conference*, 2002.

- I. Korf, P. Flicek, D. Duan, and M. R. Brent. Integrating genomic homology into gene structure prediction. *Bioinformatics*, S140-148 (2001).
- R. D. Kornberg. Eukaryotic Transcription Control. *Trends in Genetics* **15**, M46–49 (1999).
- B. Krishnapuram, L. Carin, and A. Hartemink. Joint classifier and feature optimization for cancer diagnosis using gene expression data. In *Proceedings of RECOMB*, 2003.
- A. N. Ladd and T. A. Cooper. Finding signals that regulate alternate splicing in the post-genomic era. *Genome Biology* **3**, reviews0008 (2002).
- F. Larsen, G. Gundersen, R. Lopez, H. Prydz. CpG islands are gene markers in the human genome. *Genomics* **13**, 1095–1107 (1992).
- N. N. Laurinn, S. P. Wang, and G. A. Mitchell. The hormone-sensitive lipase gene is transcribed from at least five alternative first exons in mouse adipose tissue. *Mammalian Genome* **11**, 972–978 (2000).
- C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1995.
- R. Leplae and T. J. P. Hubbard. MaxBench: evaluation of sequence and structure comparison methods. *Bioinformatics* **18**, 494–495 (2002).
- B. Lewin. *Genes VII*. Oxford University Press, 2000.
- K. Luger, A. W. Mader, R. K. Richmond, D. F Sargent, and T. J. Richmond. Crystal structure of the nucleosome core particle at 2.8Å resolution.. *Nature* **389**, 251–260 (1997).
- D. J. C. Mackay. Bayesian non-linear modelling for the prediction competition. *ASHRAE Transactions* **100**, 1053–1062 (1994).
- D. J. C. MacKay. *Ensemble Learning and Evidence Maximization*, 1995.
- D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. To be published by

Cambridge University Press, 2003. URL <http://www.inference.phy.cam.ac.uk/mackay/itprnn/>.

K. Maruyama and S. Sugano. Oligo-capping: a simple method to replace the cap structure of eukaryotic mRNAs with oligoribonucleotides. *Gene* **138**, 171–174 (1994).

C. Mathé, M. Sagot, T. Shiex, and P. Rouzé. Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Research* **30**, 4103–4117 (2002).

V. Matys, E. Fricke, R. Geffers, E. Gossling, M. Haubrock, R. Hehl, K. Hornischer, D. Karas, A. E. Kel, O. V. Kel-Margoulis, D. U. Kloos, S. Land, B. Lewicki-Potapov, H. Michael, R. Munch, I. Reuter, S. Rotert, H. Saxel, M. Scheer, S. Thiele, E. Wingender. TRANSFAC: transcriptional regulation, from patterns to profiles. *Nucleic Acids Research* **31**, 374–378 (2003).

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, London, 1983.

I. M. Meyer and R. Durbin. Comparative *ab initio* prediction of gene structures using pair HMMs. *Bioinformatics* **18**, 1309–1318 (2002).

The Mouse Genome Sequencing Consortium. Initial sequencing and comparative analysis of the mouse genome. *Nature* **420**, 520–562 (2002).

J. W. Miskin. *Ensemble Learning for Independent Component Analysis*. Ph.D. thesis, University of Cambridge, 2000.

R. Mott. EST\_GENOME: a program to align spliced DNA sequence to unspliced genomic DNA. *Computer Applications in the Biosciences* **13**, 477–478 (1997).

J. Moulton, K. Fidelis, A. Zemla, T. Hubbard. Critical assessment of methods of protein structure prediction (CASP): round IV. *Proteins Suppl* **5**, 2–7 (2001).

V. E. Myer and R. A. Young. RNA Polymerase II Holoenzymes and Subcomplexes. *Journal of Biological Chemistry* **273**, 27757–27760 (1998).

I. T. Nabney. Efficient training of RBF networks for classification. In *Proceedings of ICANN99*,

pages 210–215, 1999.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* **48**, 443–453 (1970).

Z. Ning, A. J. Cox, and J. C. Mullikin. SSAHA: A fast search method for large DNA databases. *Genome Research* **11**, 1725–1729 (2001).

E. Nishida and Y. Gotoh. The MAP kinase cascade is essential for diverse signal transduction pathways. *Trends in Biochemical Sciences* **18**, 128–131 (1993).

Open-bio database access standards. URL <http://obda.open-bio.org/>.

A. B. Pardee, F. Jacob, J. Monod. The genetic control and cytoplasmic expression of “inducibility” in the synthesis of B-galactosidase by *E. coli*. *Journal of Molecular Biology* **1**, 165–178 (1959).

G. Pesole, F. Mignone, C. Gissi, G. Grillo, F. Licciulli, and S. Liuni. Structural and functional features of eukaryotic mRNA untranslated regions. *Gene* **276**, 73–81 (2001).

Y. Pilpel, P. Sudarsanam, G. M. Church. Identifying regulatory networks by combinatorial analysis of promoter elements. *Nature Genetics* **29**, 153–159 (2001).

M. R. Pocock. *Computational Analysis of Genomes*. Ph.D. thesis, Wellcome Trust Sanger Institute, 2001.

R. C. Périer, V. Praz, T. Junier, C. Bonnard, P. Bucher. The Eukaryotic Promoter Database (EPD). *Nucleic Acids Research* **28**, 307–309 (2000).

N. J. Proudfoot, A. Furger, and M. J. Dye. Integrating mRNA processing with transcription. *Cell* **108**, 501–512 (2002).

A. Razin. CpG methylation, chromatin structure and gene silencing – a three-way connection. *EMBO Journal* **17**, 4905–4908 (1998).



B. Ren, F. Robert, J. J. Wyrick, O. Aparicio, E. G. Jennings, I. Simon, J. Zeitlinger, J. Schreiber, N. Hannet, E. Kanin, T. L. Volkert, C. J. Wilson, S. P. Bell, R. A. Young. Genome-wide location and function of DNA binding proteins. *Science* **290**, 2306–9 (2000).

P. Rice, I. Longden, and A. Bleasby. EMBOSS: The European Molecular Biology Open Software Suite. *Trends in Genetics* **16**, 276–277 (2000).

P. W. Rigby. Three in one and one in three: it all depends on TBP. *Cell* **72**, 7–10 (1993).

Tissue engineering: implications in the treatment of organ and tissue defects. Tissue engineering: implications in the treatment of organ and tissue defects. *Biogerontology* **2**, 118–125 (2001).

E. Rivas and S. R. Eddy. Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics* **2**, 8 (2001).

H. C. Roest, O. Jaillon, A. Bernot, C. Dasilva, L. Bouneau, C. Fischer, C. Fizames, P. Wincker, P. Brottier, F. Quetier, W. Saurin, J. Weissenbach. Estimate of human gene number provided by genome-wide analysis using *Tetraodon nigroviridis* DNA sequence. *Nature Genetics* **25**, 235–238 (2000).

M. Scherf, A. Klingenhoff, and T. Werner. Highly Specific Localization of Promoter Regions in Large Genomic Sequences by PromoterInspector: A Novel Context Analysis Approach. *Journal of Molecular Biology* **297**, 599–606 (2000).

M. Scherf, A. Kingenhoff, K. Frech, K. Quandt, R. Schneider, K. Grote, M. Frisch, V. Gailus-Durner, A. Seidel, R. Brack-Werner and T. Werner. First pass annotation of promoters on human chromosome 22. *Genome Research* **11**, 333–340 (2001).

B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors. *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1999.

G. Schramm, I. Bruchhaus, and T. Roeder. A simple and reliable 5'-RACE approach. *Nucleic Acids Research* **28**, e96 (2000).

S. Schwartz, W. J. Kent, A. Smit, Z. Zhang, R. Baertsch, R. C. Hardison, D. Haussler and W. Miller. Human-Mouse Alignments with BLASTZ. *Genome Research* **13**, 103–107 (2003).

I. Simon, J. Barnett, N. Hannett, C. T. Harbison, N. J. Rinaldi, T. L. Volkert, J. J. Wyrick, J. Zeitlinger, D. K. Gifford, T. S. Jaakola, R. A. Young. Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell* **106**, 697–708 (2001).

A. F. A. Smit and P. Green. RepeatMasker. URL <http://ftp.genome.washington.edu/RM/RepeatMasker.html>.

T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology* **147**, 195–197 (1981).

E. L. L. Sonnhammer and R. Durbin. A dot-matrix program with dynamic threshold control suited for genomic DNA and protein sequence analysis. *Gene* **167**, GC1–10 (1995).

J. E. Stajich, D. Block, K. Boulez, S. E. Brenner, S. A. Chervitz, C. Dagdigian, G. Fuellen, J. G. Gilbert, I. Korf, H. Lapp, H. Lehvaslaiho, C. Matsalla, C. J. Mungall, B. I. Osborne, M. R. Pocock, P. Schattner, M. Senger, L. D. Stein, E. Stupka, M. D. Wilkinson, E. Birney. The Bioperl toolkit: Perl modules for the life sciences. *Genome Research* **12**, 1611–1618 (2002).

Y. Suzuki, R. Yamashita, K. Nakai, and S. Sugano. DBTSS: DataBase of human Transcriptional Start Sites and full-length cDNAs. *Nucleic Acids Research* **30**, 328–331 (2002).

M. Pocock, T. Down, M. Schreiber, K. James, D. Huen, *et al.*. BioJava: Open Source components for Biological Computation.. URL <http://www.biojava.org/>.

The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics* **25**, 25–29 (2000).

M. E. Tipping. The Relevance Vector Machine. *Advances in Neural Information Processing Systems* **12**, 652–658 (2000).

T. A. Tatusova and T. L. Madden. Blast 2 sequences – a new tool for comparing protein and nucleotide sequences. *FEMS Microbiology Letters* **174**, 247–250 (1999).

A. Wright, B. Charlesworth, I. Rudan, A. Carothers, and H. Campbell. A polygenic basis for late-onset disease. *Trends in Genetics* **19**, 97–106 (2003).