Computational gene prediction using generalised hidden Markov models and tiling arrays

Department of Mathematics and Statistics
The University of Melbourne

Bioinformatics division
The Walter and Eliza Hall Institute of Medical Research

Honours thesis

Benjamin Lansdell

Supervisors:
Anthony Papenfuss and Terence Speed

November 2008

Abstract

The genome of a higher organism is a complex entity. It is not merely comprised of the genes it encodes, but also of many other contributing elements. Elucidating the function of these elements is a non-trivial task which lends itself well to both computational and statistical methods. Additionally, it has recently become apparent that non-coding RNA genes play a larger role than previously realised and that the protein-centric view of molecular biology may need revising.

This thesis explores two current methods of identifying these functional elements: computational gene prediction and transcription mapping. In order to do so a generalised hidden Markov model (GHMM) *ab initio* gene predictor is developed which is shown to perform comparably to other published *ab initio* GHMM predictors. We implement a transcription mapping statistic based on correlations that performs comparably to a standard sliding-window approach in the analysis of genomic tiling arrays. We present TileGene, a GHMM gene model which predicts protein-coding genes and non protein-coding gene fragments based on tiling array expression data, thus accommodating a broader and more realistic view of molecular biology.

Contents

Acknowledgments

Many thanks must go first and foremost to both of my supervisors, Tony Papenfuss and Terry Speed, for their guidance through the year. Their time and effort in supporting and providing feedback for this project has been greatly appreciated.

Thanks to Tony for introducing me to the field of bioinformatics; and for providing assistance with some of the illustrations in this thesis.

Also I would like to thank Mark Robinson, WEHI, for his assistance with some of the microarray analysis in `R` and `aroma.affymetrix`.

Thanks to my housemates, Rohan, Tim, Catherine and Sam, for their patience and understanding;

My family, for their help throughout the year;

and Bethany, for always supporting me and for putting up with my obsessive study habits this year.

"Our imagination is stretched to the utmost, not, as in fiction, to imagine things which are not really there, but just to comprehend those things which are there."

Richard Feynman

Chapter 1

Biological overview

In order to understand the project and its aims, some basic biology needs to be introduced. This chapter presents a summary of the relevant biology and experiments used throughout the thesis as well as providing a more detailed description of the project.

1.1 Life from the cellular perspective

A cell can be viewed as the fundamental building block of a multi-cellular organism but also as a large and complex chemical system in its own right. Of key importance to this system are the three macromolecules that will be the focus of this thesis: deoxyribonucleic acid (DNA), ribonucleic acid (RNA) and protein. DNA and RNA work together to synthesize protein and it is the proteins within a cell that largely describe a cell's character, its function and form. Proteins have many functions, including acting as enzymes and acting in a structural or mechanical capacity. Thorough knowledge of these molecules and their interaction is therefore fundamental to the understanding of an organism.

DNA consists of two sugar-phosphate strands wrapped around each other to form a double helix. Attached to each sugar link in the chain is a base molecule, of which there are four: adenine (A), cytosine (C), guanine (G) and thymine (T). Each sugar-phosphate-base unit is referred to as a nucleotide and each nucleotide bonds to its opposing nucleotide on the other strand forming a base-pair (bp). Adenine will only bond with thymine and cytosine will only bond with guanine. This is known as Watson-Crick complementarity and means that each strand of DNA is complementary to the other. Because of this the DNA sequence of only one strand (referred to as the forward strand) needs be described in order to fully specify the molecule. The other strand is known as the reverse strand and its sequence will be the *reverse complement* to the one given. A DNA strand has a preferred direction of synthesis which is in the $5'$ to $3'$ direction;[1] in relation to the forward strand this simply means left to right. See Figure 1.1.

[1] Here, $5'$ and $3'$ refer to the fifth and third carbon of the pentose (five carbon) sugar. The $5'$ end contains a phosphate group that binds to the $5'$ carbon whilst the $3'$ carbon is free for additional nucleotides to bind to it. Nucleotides are more easily added to the $3'$ end, hence the preferred direction of synthesis.

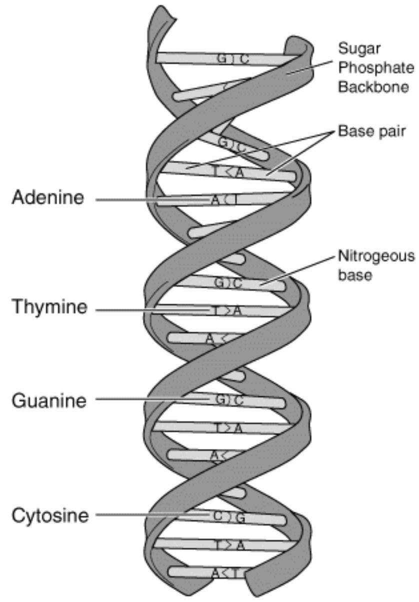


Figure 1.1: Diagram of DNA strands coiled in a double helix demonstrating allowed base pairings according to Watson-Crick complementarity. Source: `http://www.genome.gov/Pages/Hyperion/DIR/VIP/Glossary/Illustration/Pdf/dna.pdf`

DNA exists in almost all cells of an organism[2] and is contained in a larger structure known as a chromosome. The number of chromosomes in each cell depends on the species and can vary to a large degree. Humans, for example, have 46 chromosomes while *Drosophila melanogaster* (the fruit fly, typically referred to simply as drosophila) has only 8. The total DNA contained within an organism's chromosomes is referred to as its genome. A eukaryotic organism houses its chromosomes in a nucleus, seperate to the bulk of the cell, and a prokaryotic organism does not. As such DNA and proteins interact differently in each case. We will be concerned only with eukaryotes. A segment of DNA which is translated into a protein is known as a protein-coding gene or, more simply, a gene. The discrete 'alphabet' of a DNA sequence means that DNA provides a form of information storage. In this context a gene can be thought of as a set of instructions stored on the genome that 'encodes' a protein.

RNA is similar in structure to DNA with the exception that it is generally single stranded and the base thymine is replaced by uracil (U). As with thymine, uracil is complementary to adenine. Being single stranded means that RNA is able to fold itself into complex three dimensional shapes and can perform some catalytic functions similar to protein. RNA acts as an intermediary between genes and protein, providing temporary storage of genetic information as well as some of the functions involved in this translation. RNA exists both inside and outside of the nucleus.

A protein is a chain of amino acids linked by peptide bonds. There are 20 relevant amino acids, each of which has different properties (*e.g.* size, hydrophobicity, charge) that fold the

[2]Red-blood cells in mammals being an exception.

protein into a specific and contorted structure and allow proteins to perform a diverse range of functions. The shape a protein folds into is determined uniquely by the sequence of amino acids it is composed of, which in turn is determined by the instructions in its corresponding RNA sequence. This flow of information from DNA to RNA to proteins is, in general, one way. This is known as the central dogma of molecular biology.[3]

1.2 Genes, genomes and proteins

We describe in more detail the structure of genes and the process by which a protein is created using this genetic information. Of particular interest to us will be the sequences on the genome which are involved. The process is split into two parts, that of transcription and of translation. Transcription is the production of RNA from DNA and translation is the production of protein from RNA.

Not all genes are transcribed at one time; those that are transcribed are said to be expressed. Proteins known as transcription factors control which genes are expressed. Transcription of a gene begins when an enzyme known as RNA polymerase attaches itself to the DNA. RNA polymerase will bind more strongly to certain areas which are known as promoter sites. In eukaryotes the promoter site may contain the *consensus sequence* TATAA or TATA, known as a TATA box. Once the enzyme is attached it splits the DNA into strands and proceeds in the $5'$ to $3'$ direction to make an RNA copy complementary to one strand; nucleotides $\{A,C,G,T\}$ are paired with $\{U,G,C,A\}$. This continues until the polymerase reaches a termination site. The result is known as pre-messenger RNA (pre-mRNA). A eukaryotic gene may contain regions, known as introns, that are removed from pre-mRNA. Regions that are not removed are known as exons. Introns are *spliced* from the RNA by a structure known as a spliceosome. An intron typically begins with a donor site, GT, and typically ends with an acceptor site, AG.[4] In some genes splicing can occur at different points, creating a number of possible transcripts from the one gene. This is known as alternative splicing and can be the result of exon skipping, in which an entire exon is spliced from the RNA. Once introns have been spliced the RNA is then capped on the $5'$ end with a methylated guanosine to prevent it from being degraded by certain enzymes. The $3'$ end undergoes polyadenylation in which a string of adenines is added to the transcript somewhere downstream of a polydenylation signal, typically ATTAAA or AATAAA, to aid in being exported from the nucleus. The result is called a mature mRNA or just mRNA and is exported from the nucleus for translation. This is summarised in Figure 1.2.

A protein is created when a large RNA complex known as a ribosome attaches itself to the mRNA. The ribosome attracts and allows transfer RNA (tRNA) to bind with the mRNA. There are 61 types of tRNAs, each having one of twenty amino acids attached to one end and a correspoding *anti-codon* on the other end. An anti-codon is three nucleotides that are complementary to the *codon* contained on the mRNA transcript. This allows for a correspondence between RNA and amino acids and hence the synthesis of a specified protein. The

[3] As with many aspects of biology there are exceptions. For example some RNA viruses known as retroviruses replicate by invading a host genome with viral DNA, which the host dutifully transcibes back to RNA.

[4] Other splice signals also exist, but account for less than 2% of splice sites. (Burset et al. [2001])

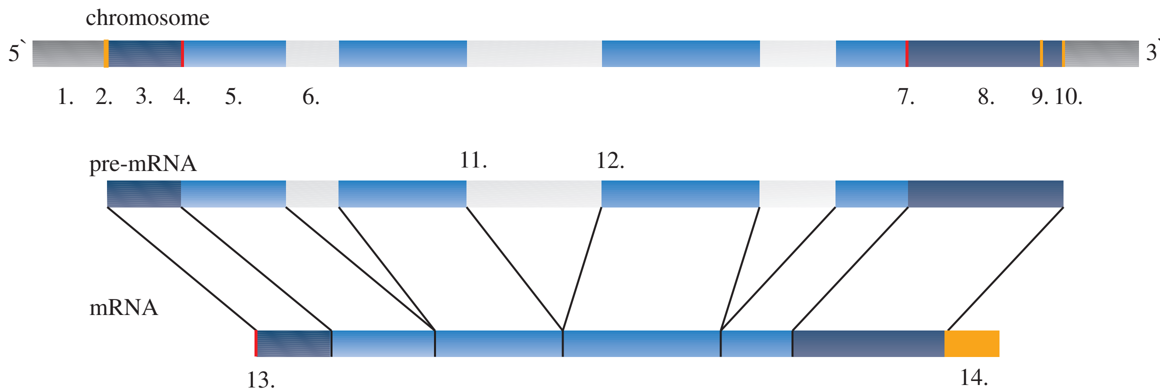


Figure 1.2: How a gene is transcribed from DNA to mRNA, ready to be translated. 1. Intergenic region; 2. Promoter site; 3. 5′ UTR; 4. Start codon; 5. CDS; 6. Intron; 7. Stop codon; 8. 3′ UTR; 9. Polyadenylation signal; 10. Transcription termination site; 11. Splicing occurs between donor site and; 12. Acceptor site; 13. Methylated guanosine cap and; 14. polyadenylation tail added.

ribosome generally begins translation when it recognises a translation start codon, ATG, and continues reading the mRNA until it reaches a stop codon – a triple of nucleotides for which there is no corresponding tRNA – one of TAA, TGA or TAG. As each codon is read its amino acid is attached to the peptide being created. There are $4^3 = 64$ codons and only twenty amino acids that compose protein; the tRNA uses a many-to-one mapping.[5] The region that is translated into protein is called coding sequence (CDS) and regions that are not translated into protein are known as untranslated regions (UTR) and may exist on both the 5′ and 3′ end of the transcript.

It should be noted that a eukaryotic genome is a more complex entity than just a collection of genes. Taking humans as an illustrative example: of the three billion base pairs that compose the human genome an estimated thirty million base pairs (1%) of these are protein coding. A sizable proportion of the human genome is composed of repeats, highly repetitive or low complexity sequence that has no apparent purpose. It is theorised that some repeats are the result of an ancient retrovirus infecting a distant ancestor. Repeats generally occur in intergenic or intronic sequence but there are exceptions. There are also genomic regions that are transcribed but are not translated to proteins. These will be referred to as non-coding RNAs, discussed below.

The preceding two sections are based on Majoros [2007], Ch.1 and Clote and Backofen [2000], Ch.1. A good introduction to these concepts is Gonick and Wheelis [1991].

1.3 Non-coding RNAs

Moving past our basic introduction, we briefly discuss the class of non-coding RNAs (ncRNA, or alternatively non-coding gene). Historically, RNA has been viewed as an intermediary

[5]This degeneracy makes the code more robust to single base mutations.

between DNA and protein – that its primary purpose is the creation of protein. Indeed the RNAs described in the preceding sections (*e.g.* mRNA, tRNA, ribosome) agree with this notion. The central dogma of molecular biology which emphasizes RNA's role in protein synthesis perhaps explains why attention has, until recently, been primarily on protein (Eddy [2001]). More recently, a number of other classes of RNA have been discovered, some of whose function remains unknown. A well known example in human is the ncRNA *XIST* – a 17kb ncRNA which has a role in dosage compensation related to X-chromosome inactiviation.[6] *XIST* is spliced and polyadenylated like an mRNA. An example at a much smaller scale is microRNA. Such RNAs are approximately 22-nucleotides in length and have been shown to have gene regulatory functions. By binding to complementary regions in a corresponding gene the RNA effects the translation of that gene. Other classes have been shown to have roles in RNA modification, transcription and splicing. ncRNAs are found within introns and can overlap genes and other ncRNAs on the same or opposing strand. This creates a complex tapestry of elements which challenges the notion of a gene as a discrete functional unit. For example see Mattick [2003] or Mattick and Makunin [2006].

In addition to the growing number of functions ascribed to ncRNAs, there is mounting evidence to suggest the amount of transcription is much greater than has been documented or expected. This evidence comes from large scale cDNA and genomic tiling array studies (described below). For example, in human it is estimated that approximately 50% of transcriptional activity is not due to protein-coding genes (Kampa et al. [2004], Bertone et al. [2004]). The function of this extra transcription remains largely unknown, although there are reasons to suspect that much of it is biologically meaningful (again, see Mattick).

Ultimately, it is clear that there exists an, at present, hidden network of complex, interacting regulatory and functional ncRNAs. Indeed, Mattick points out that the proteome[7] of the simple worm *Caenorhabditis elegans* and human varies by less than 30%. Yet the difference in developmental complexity is significant, a fact which suggests that in higher organisms ncRNAs play a larger than estimated role. Therefore, the proper characterisation of an organism's transcriptome, in addition to its proteome, is of key importance.

1.4 Tiling microarray experiments

One way of mapping transcription in a large, systematic fashion is a genomic tiling microarray, or tiling array. A microarray is a support on which a collection of known single-strand DNA segments known as probes are attached to features on the array. These supports can be glass slides, beads, silicon chips or nylon membranes. The density of a microarray can be very high, allowing millions of probes to be attached. A microarray experiment aims to quantify DNA or RNA products, known as targets, present in a given sample. Due to their high throughput, such experiments are useful for a number of things, including gene expression analysis, SNP

[6] In female human cells one X chromosome is inactiviated at random so that the expression of X-chromsome genes is equivalent between males and females.

[7] As with the genome, the proteome is the sum of an organism's protein – translated from protein-coding genes. Other 'omes' include the transcriptome and the metabolome.

detection and alternative splicing detection.[8]

While the methods we develop are general, here we are concerned only with tiling microarrays produced by Affymetrix.[9] Affymetrix microarrays consist of probes attached to a coated quartz surface, manufactured at a high density using techniques similar to those of semiconductor device construction. A tiling microarray, or tiling array, aims to *tile* a subset of an organism's chromosomes, or even its whole genome. The array is designed so that the sequence of each probe represents the sequence of a unique position on a chromosome. This is unrealistic, particularly for arrays which use short probes, because some regions will ultimately contain non-unique sequence either through the presence of repeats or simply by chance. Conceptually though, each probe is thought to represent one position. The average genomic distance between neighbouring probes is known as the resolution. Affymetrix' probes are synthesized DNA strands, which are known as oligonucleotides, of length 25bp. See Figure 1.3.

Figure 1.3: Tiling array probes correspond to unique positions along an entire chromosome.

In general, an experiment proceeds as follows: RNA samples from the organism are extracted at times or from cell-types of interest and reverse transcribed into cDNA (complementary DNA). The resulting target is then labeled with a fluorescent dye and is introduced to the array. Probes that are complementary to a target DNA molecule will hybridise, the complementary pairs forming a double helix. The array is then washed to remove un-hybridised target DNA. It is subsequently exposed to a laser, causing the dye on the array to fluoresce, and then imaged. The brightness recorded for each feature on the array is then a measure of the amount of hybridisation a known probe underwent, and therefore a measure of the quantity of the corresponding DNA in the sample. See Figure 1.4.

Since probes on a tiling array correspond to known genomic positions, this intensity gives information about which regions on a chromosome are transcribed under the conditions from which the sample was taken. The benefit of using a tiling array is that it provides this information in an unbiased fashion – other arrays generally restrict probes to represent known genes. Obtaining a reliable transcript level from a set of measured intensities is not straightforward, there are issues both common to all Affymetrix arrays and specific to tiling arrays that we discuss later.

[8] One example might be comparing genes found to be expressed in tumour cells with non-tumour cells. A SNP, or single nucleotide polymorphism, is a variation of a single nucleotide within individuals of the same species and can be the cause of disease.

[9] `www.affymetrix.com`

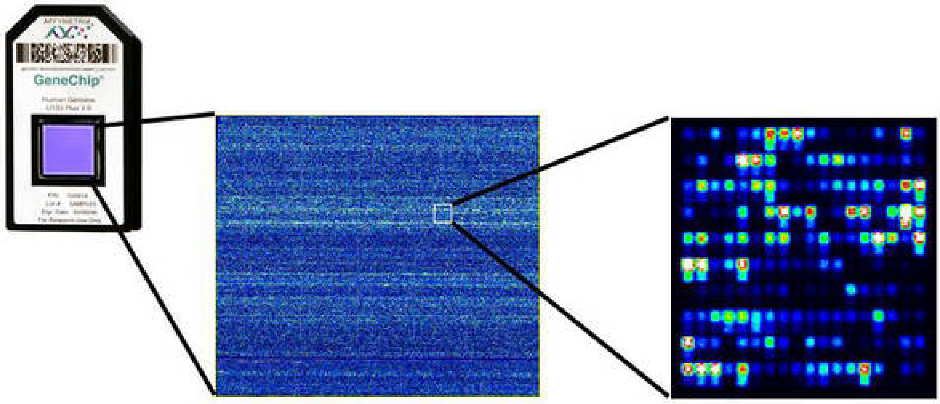


Figure 1.4: An Affymetrix microarray showing recorded intensities after target is hybridised to the array. Each square unit (far right) is a feature. Source: `http://www.dkfz.de/gpcf/24.html`

1.5 Project overview

Having outlined the relevant biology and experimental methods, we can now better examine the project and its goals. Knowing which elements of a genome are transcribed, which are translated and to what protein is a non-trivial task given the size and complexity of a eukaryotic genome. Indeed, the size of the task means that small-scale experimental approaches to such a problem are of limited use. The task is perhaps more efficiently approached using computational methods and high-throughput experiments such as microarrays, leaving smaller experiments to verify subsequent results. Computationally determining which elements of a genome are transcribed is known as transcript mapping and can be performed by analysing tiling array data. Related is the problem of determining which elements are genes and what their underlying structure and resulting protein is. This is known as gene prediction and relies on, among other things, analysing genomic sequence. This thesis will explore both these methods.

Current gene prediction methodologies focus primarily on protein-coding genes. We aim to design a gene predictor that predicts both protein-coding genes and non-coding gene fragments simultaneously, thus incorporating a more realistic view of a genome. Whether the resulting method is a transcript mapper or a gene predictor is arguable but also largely irrelevant, since it is in a sense both. More specifically we identify our goals as:

- Provide a more accurate and finer resolution of transcript mapping by incorporating gene prediction elements and thereby predicting the precise nucleotide, as opposed to the precise probe, which borders a transcribed region. Also provide a characterisation of a transcript map by qualifying elements as either CDS, ncRNA, etc.
- Improve current gene prediction techniques by making use of tiling array data. It is hoped that the tiling data can identify potentially missed exons and genes.

A few problems with these aims jump to mind. An obvious problem is that tiling array data is generally incomplete. Unless samples are taken from every cell type at every possible time there is likely to be transcripts not expressed in the samples, which means tiling array data will only provide support for genes (non-coding and protein-coding) that are active in the provided experiment. Gene prediction improvements using tiling array data is a separate (though related) project in itself and so our focus will primarily be on accurately identifying expressed genes and ncRNAs within tiling array data with the hope that this will provide insights into how gene prediction accuracy can be improved more generally.

The remainder of the thesis proceeds as follows:

Chapter 2 provides the necessary mathematical details for implementing regular and generalised hidden Markov models. This comprises the bulk of the mathematical content of the thesis.

Chapter 3 describes an application of hidden Markov models for use in gene prediction. An implementation of such a model is described and its performance is compared to other published gene predictors on a common dataset.

Chapter 4 describes current methods of transcription mapping. We investigate one such method which is suitable for our purposes and compare its performance to one other published method.

In Chapter 5 we describe modifications to our gene model that allow it to predict protein-coding genes and non–coding gene fragments. Its performance is compared to both our original gene predictor and transcription mapper on a set of expressed genes and known ncRNAs.

Chapter 6 contains discussion about our results and future directions in which the project can continue.

Chapter 2

Hidden Markov models

This chapter will develop the mathematics necessary for designing and implementing a hidden Markov model. This will provide the framework for our gene predictor and therefore important algorithms involved in its use are also described.

2.1 Markov models

A Markov model is a stochastic process.

Formally, we can define a discrete-time stochastic process as a set of indexed random variables $\{X_t\}_{t\geq 1}$ on a common probability space $(\Omega, \mathcal{F}, \mathbf{P})$, where Ω is the outcome space, $\mathcal{F}$ is a σ-algebra and $\mathbf{P}$ is a probability measure. For now we will be limiting ourselves to finite time, discrete processes – the case when each X_t is a discrete random variable and $1 \leq t \leq T$. All of our models will use a finite state space, $\mathcal{S} = \{S_i\}_{1\leq i\leq N}$, which is simply defined as the set of possible values X_t can assume. The index is often interpreted as time, though this is not necessary. If such a process satisfies the *Markov property*:

$$\Pr(X_t = S_j | X_{t-1} = S_i, X_{t-2} = S_k, \ldots, X_0 = S_l) = \Pr(X_t = S_j | X_{t-1} = S_i)$$

then it is said to be a Markov model, or Markov chain.

This can be thought of as a process in which the 'history' of the process is irrelevant to its future state, given the current state – the only determining factor in a process's next state is its current state. This is a useful assumption to make as it allows for some past dependence to be modeled, without having to include all possible dependencies on history. Markov chains can be used, for example, to model a board game played with dice in which potential outcomes are dependent only on a dice roll and the current position on the board, *e.g.* Snakes and Ladders. Markov chains could not be used, for example, to model a card game in which potential outcomes are dependent not only on cards currently played but on cards previously played, *e.g.* 500.[1]

[1] Strictly speaking, this is true of first order Markov chains. However higher orders could possibly be used.

The simplest discrete Markov model is a homogeneous Markov model in which we assume that $\Pr(X_t = S_j|X_{t-1} = S_i)$ is independent of index t. Then it is useful to define:

$$a_{ij} = \Pr(X_t = S_j|X_{t-1} = S_i), \qquad 1 \leq i, j, \leq N$$

The set of $\{a_{ij}\}$ are known as transition probabilities and describe the probability of the process transitioning from state S_i to state S_j in one time-step. These are often contained in a transition matrix, A, or are labeled in a state diagram – a directed graph in which vertices represent states and edges represent allowed transitions. If we also define the initial state probabilities as:

$$\pi_i = \Pr(X_1 = S_i), \qquad\qquad 1 \leq i \leq N$$

then these two sets of parameters are sufficient to compute the probability of any observed sequence. The Markov model is entirely described by the set $\lambda = \{\mathcal{S}, A, \pi\}$. Using the notation $x_1^T = \{x_1, x_2, x_3, \ldots, x_T\}, x_i \in \mathcal{S}$ as shorthand for the event $\{X_1 = x_1, X_2 = x_2, X_3 = x_3, \ldots, X_T = x_T\}$ (more loosely $\mathbf{x}$ may also be used) and applying the Markov property where needed, we compute the probability of a sequence of observations as:

$$\begin{aligned}\Pr(x_1^T|\lambda) =& \Pr(x_1^{T-1}|\lambda)\Pr(X_T = x_T|x_1^{T-1}, \lambda)\\ =& \Pr(x_1^{T-2}|\lambda)\Pr(X_{T-1} = x_{T-1}|x_1^{T-2}, \lambda)\Pr(X_T = x_T|x_1^{T-1}, \lambda)\\ &\vdots\\ =&\pi_{x_1} a_{x_1x_2} \ldots a_{x_{T-2}x_{T-1}} a_{x_{T-1}x_T}\end{aligned} \tag{2.1}$$

Note that the probability of remaining in a given state S_i over d time-points has a geometric distribution:

$$\Pr(X_1 = S_i, X_2 = S_i, \ldots, X_d = S_i, X_{d+1} \neq S_i|X_1 = S_i, \lambda) = (1 - a_{ii})a_{ii}^{(d-1)} \tag{2.2}$$

In most cases it is likely that dependencies exist in a process that extend beyond the current state. If this is the case then a higher order Markov model may be appropriate. An nth order Markov chain satisfies:

$$\begin{aligned}\Pr(X_t = x_t|X_{t-1} = x_{t-1}, &X_{t-2} = x_{t-2}, \ldots, X_0 = x_0) =\\ &\Pr(X_t = x_t|X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, \ldots, X_{t-n} = x_{t-n})\end{aligned}$$

with $n > 0$. If we denote:

$$a_{ij\ldots kn} = \Pr(X_t = S_n|X_{t-1} = S_k, \ldots, X_{t-n+1} = S_j, X_{t-n} = S_i)$$

then our set of parameters now allows an amount of the past to affect future outcomes. Computing the probability of a sequence lends itself to the same decomposition as above:

$$\Pr(x_1^T|\lambda) = \pi_{x_1} a_{x_1x_2} a_{x_1x_2x_3} \cdots \prod_{t=n+1}^{T} a_{x_{t-n}\cdots x_t}$$

where care needs to be taken in dealing with the observations near the start of the sequence, for which there have not been n preceeding observations.

An inhomogeneous Markov model drops the assumption that our set of parameters $\{a_{ij}\}$ describe transition parameters for any t. If this was done in general, it would greatly increase the number of parameters needed to describe a model. We make use of periodic Markov models in which the transition probabilities are a function of time, but only in a regular fashion. Here transition probabilities depend on a cycling index of period p. We would evaluate the probability of a given sequence as:

$$\Pr(x_1^T|\lambda) = \pi_{x_1} \prod_{t=2}^{T} a_{x_{t-1}x_t}^{t_p}$$

where $t_p = (t \mod p)$ and a^{t_p} is the transition matrix to be used during period t_p. We can picture that such a model would be useful in animal populations throughout months of the year. A present population depends not only on the month's previous population but additionally on the season of the year, which varies periodically.

2.2 Hidden Markov models

Markov models provide a simple probabilistic framework for modeling real-life processes, but for many applications they are either not directly useful or interesting models. It is easy to imagine a process in which the precise sequence of states of interest is not immediately observable but would still be reasonable to model because of some related variable that *is* observable. An example might be the spoken word. Sounds waves can be distorted as they travel from source to destination, such that the word heard (observable) might be quite different to the word spoken (hidden) and whose determination would then involve guessing the most *likely* actuality.

Hidden Markov models (HMMs) provide a framework for modeling such situations and since their development in the late 60s (Baum and Petrie [1966]) have been popular models for speech recognition. Lately they have become popular in other areas, including bioinformatics. They are flexible and built on a solid probability theory.

We can define an HMM as a stochastic process which at any time is described by a tuple $\{(X_t, Y_t)\}_{t\geq 1}$. Here $\{X_t\}$ is a Markov chain with state-space $\mathcal{S}$, transition matrix A, initial distribution π and x_1^T defined as above. We again assume that $1 \leq t \leq T$. $\{X_t\}$ is the unobservable or hidden process. $\{Y_t\}_{1\leq t\leq T}$ is the observed process and is conditional on $\{X_t\}$. For the moment we assume the Y_t are finite, discrete random variables. We denote the state space of Y_t as $\mathcal{O} = \{O_1, \ldots, O_K\}$ and $y_1^T = \{Y_1 = y_1, \ldots, Y_T = y_T\}, y_i \in \mathcal{O}$. The conditional dependencies can be stated as:

$$\Pr(X_t = x_t | x_1^{t-1}, y_1^{t-1}) = \Pr(X_t = x_t | X_{t-1} = x_{t-1})$$

and

$$\Pr(Y_t = y_t | X_t = x_t, x_1^{t-1}, y_1^{t-1}) = \Pr(Y_t = y_t | X_t = x_t)$$

which is represented in Figure 2.1. We further denote:

$$b_j(k) = \Pr(Y_t = O_k | X_t = S_j)$$

for $1 \leq k \leq K$ and $1 \leq j \leq N$. This is known as the emission distribution, $B = \{b_j(k)\}$. An HMM is then fully described by the set $\lambda = \{\mathcal{S}, \mathcal{O}, \pi, A, B\}$. We will denote our parameter space as Λ.

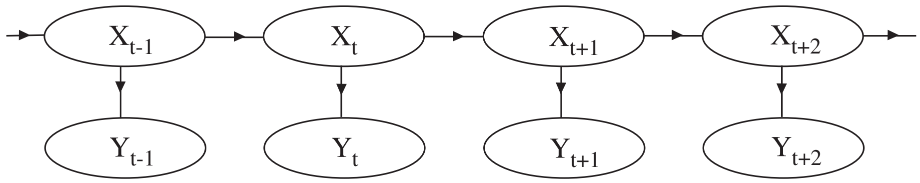


Figure 2.1: Representation of an HMM. Random variables X_i are hidden and have the serial dependence of a Markov chain. Random variables Y_i are observed and have distribution conditional on X_i.

For an example that demonstrates the above formulation consider a dice game at a casino in which you suspect that at least some of the time the game is being played with a weighted die. There are a number of HMMs that could model this situation and we will describe just one. Let there be two hidden states, biased die (Weighted) and unbiased die (Natural) which are of interest, so that $\mathcal{S} = \{W, N\}$. A sequence of observations would consist of a sequence of dice rolls and a result between 1 and 6, so that $\mathcal{O} = \{1, \ldots, 6\}$. The emission distribution B would describe the distribution of rolls results for either $X = W$ or $X = N$. Since we don't know any better we assume that the die used (W or N) transitions stochastically according to A and starts in some state according to π. It can be seen formulating this as an HMM allows us to compute a number of things of interest such as how long, on average, the game is biased and how the two die are weighted.

An HMM is a generative model. This means it can be used to stochastically generate observation sequences as follows:

---

1. Choose an initial state $X_1 = S_i$ according to initial distribution π.
2. Set $t = 1$.
3. Choose observation $Y_1 = O_k$ according to emission probability distribution B.
4. Transition to a new state $X_{t+1} = S_j$ according to transition probability matrix A.
5. Set $t = t + 1$; If $t > T$ terminate otherwise go to step 3.

Following Rabiner [1989] there are three key problems in the application of HMMs. These are:

1. Given an observation sequence, $\mathbf{y}$, and a model, λ, how can $\Pr(\mathbf{y}|\lambda)$ be efficiently computed.
2. Given $\mathbf{y}$ and λ how do we choose a state sequence $\mathbf{x}$ that best explains the given observation, that is in some sense the most likely.

3. How do we find model parameters λ that maximises $\Pr(\mathbf{y}|\lambda)$.

Each will be discussed in order. The algorithms solving these problems are sufficient for many HMMs. The main task when using a hidden Markov model is the choice of a realistic or useful state architecture and set of model parameters. This will be discussed later in the context of gene prediction.

2.3 The forward procedure

Addressing problem 1, our aim is to compute $\Pr(\mathbf{y}|\lambda)$. The obvious way is to use the total probability formula:

$$\Pr(\mathbf{y}|\lambda) = \sum_{\mathbf{x}} \Pr(\mathbf{y}|\mathbf{x}, \lambda) \Pr(\mathbf{x}|\lambda)$$

which using equation (2.1) becomes:

$$\Pr(\mathbf{y}|\lambda) = \sum_{\mathbf{x}} \Pr(\mathbf{y}|\mathbf{x}, \lambda)\pi_{x_1} \prod_{t=2}^{T} a_{x_{t-1}x_t}$$

and since we assume observations are conditionally independent of each other given the hidden state then:

$$\begin{aligned} \Pr(\mathbf{y}|\lambda) &= \sum_{\mathbf{x}} \prod_{t=1}^{T} b_{x_t}(y_t)\pi_{x_1} \prod_{t=2}^{T} a_{x_{t-1}x_t} \\ &= \sum_{\mathbf{x}} \pi_{x_1} b_{x_1}(y_1) \prod_{t=2}^{T} b_{x_t}(y_t) a_{x_{t-1}x_t} \qquad (2.3) \end{aligned}$$

This computation requires $O(T(N^T))$ operations, which is unfeasible for even modest values of either T or N. We must turn to dynamic programming for something more reasonable, we define the forward variable to be:

$$\alpha_t(i) = \Pr(y_1^t, X_t = S_i|\lambda) \qquad (2.4)$$

which is the probability of observing the sub-sequence y_1^t and being in state S_i at time t. For each t this can be computed inductively via the forward procedure:

---

1. Initialisation:

$$\alpha_i(1) = \pi_i b_i(y_1)$$

for $1 \leq i \leq N$.

2. Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(y_{t+1}),$$

for $1 \leq j \leq N$, $1 < t \leq T - 1$.

3. Termination:

$$\Pr(\mathbf{y}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

The forward procedure involves $O(N^2T)$ operations so is far more acceptable for large HMMs. A useful interpretation of the value $\Pr(\mathbf{y}|\lambda)$ is as a score of how well a model λ fits a set of observations.

2.4 The forward-backward procedure

Addressing problem 2, our aim is to find the most likely sequence of states, later to be referred to as a parse, given a sequence of observations. This is known as decoding. One obvious solution to the problem is to consider the optimal parse, $\mathbf{x}^*$, as being constructed by the states that in each position maximises the probability of the observed variable. To compute this efficiently we define the backward variable as:

$$\beta_t(i) = \Pr(y_{t+1}^T | x_t = S_i, \lambda)$$

which is the probability of observing the remaining sequence y_{t+1}^T given the process is in state S_i at time t. We compute $\beta_t(i)$ inductively as:

---

1. Initialisation:

$$\beta_T(i) = 1$$

   for $1 \leq i \leq N$.

2. Induction:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(y_{t+1}) \beta_{t+1}(j)$$

   for $t = T-1, T-2, \ldots, 1$, $1 \leq i \leq N$.

For each position we then define:

$$\gamma_t(i) = \Pr(X_t = S_i | \mathbf{y}, \lambda)$$

which is easily expressed in terms of the forward and backward variables:

$$\begin{aligned}\gamma_t(i) &= \frac{\alpha_t(i)\beta_t(i)}{\Pr(\mathbf{y}|\lambda)} \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}\end{aligned}$$

$\gamma_t(i)$ is the probability of being in state S_i at time t. Given a parse and a set of observations γ_t provides a measure of how 'consistent' the proposition the system is in state S_i at time t is with the rest of the observations. Then we set:

$$x_t^* = \operatorname*{argmax}_{1\le i\le N} \gamma_t(i), \qquad 1 \le t \le T$$

2.5 The Viterbi algorithm

The problem with the forward-backward proceduce in determining the 'optimal' parse is that it does not consider the likelihood of the underlying state transtions that must occur between times. It may be that the optimal parse generated is not even one allowed by the state transition matrix. For this reason we now define $\mathbf{x}^*$ as:

$$\begin{aligned}\mathbf{x}^* &= \operatorname*{argmax}_{\mathbf{x}} \Pr(\mathbf{x}|\mathbf{y},\lambda)\\ &= \operatorname*{argmax}_{\mathbf{x}} \frac{\Pr(\mathbf{x},\mathbf{y}|\lambda)}{\Pr(\mathbf{y}|\lambda)}\\ &= \operatorname*{argmax}_{\mathbf{x}} \Pr(\mathbf{x},\mathbf{y}|\lambda)\end{aligned}$$

By maximising the joint probability of state and observation sequence we take into account state transition probabilities and also maximise the conditional probability of the state sequence. This problem is effectively solved using the Viterbi algorithm (Viterbi [1967]). Using the notation:

$$\delta_t(i) = \max_{x_1^{t-1}} \Pr(x_1^{t-1}, X_t = S_i, y_1^t|\lambda)$$

$\delta_t(i)$ is the highest probability of a single parse up until time t, ending in state S_i. Then by induction we can infer:

$$\delta_{t+1}(j) = [\max_{1\le i\le N} \delta_t(i)a_{ij}]b_j(y_{t+1})$$

To keep track of this optimal state sequence we also introduce array $\psi_t(j)$ and proceed:

---

1. Initialisation:

$$\begin{aligned}\delta_1(i) &= \pi_i b_i(y_1)\\ \psi_1(i) &= 0\end{aligned}$$

   for $1 \le i \le N$.

2. Recursion:

$$\begin{aligned}\delta_t(j) &= [\max_{1\le i\le N} \delta_{t-1}(i)a_{ij}]b_j(y_t)\\ \psi_t(j) &= \operatorname*{argmax}_{1\le i\le N} \delta_{t-1}(i)a_{ij}\end{aligned}$$

   for $1 \le j \le N$, $2 \le t \le T$.

3. Termination:

$$P^* = \max_{1 \leq i \leq N} \delta_T(i)$$
$$x_T^* = \operatorname*{argmax}_{1 \leq i \leq N} \delta_T(i)$$

4. Backtracking:

$$x_t^* = \psi_{t+1}(x_{t+1}^*)$$

for $t = T-1, T-2, \ldots, 1$.

Viterbi is quite similar in form to the forward algorithm, the main difference being the use of max instead of the summing of probabilities. The algorithm takes $O(N^2T)$ operations and so is quite acceptable for longer sequences.

2.6 Parameter estimation

We now turn our attention to problem 3. In some cases we may know the set of parameters λ based on prior experience or perhaps a known physical interpretation. In other cases we may not know λ or discovering certain parameters may in fact be the aim of using an HMM. If this is the case then we need methods for estimating λ, known as training methods. If a number, S, of labeled training sequences, with observations $\mathcal{Y}$ and labels $\mathcal{X}$, are available then we aim to find parameters such that $\Pr(\mathcal{Y}, \mathcal{X}|\lambda)$ is maximised. If labels, $\mathcal{X}$, are not available then we aim to maximise $\Pr(\mathcal{Y}|\lambda)$. Obviously training will be most effective if the training set is representative of the process being modeled so thought and perhaps effort is required in choosing or finding $\mathcal{Y}$. We assume individual training sequences are independent such that $\Pr(\mathcal{Y}, \mathcal{X}|\lambda) = \prod_{s=1}^{S} \Pr(\mathcal{Y}_s, \mathcal{X}_s|\lambda)$.

2.6.1 Estimation from labeled training sequences

When $\mathcal{X}$ is available we find parameters that maximise the likelihood of $\Pr(\mathcal{X}, \mathcal{Y}|\lambda)$. The method is simple and intuitive. Parameters λ' are estimated from observed frequencies of emissions and transitions between states in the training set. If $C_a(i,j)$ is the observed number of transitions from state S_i to S_j and $C_b(i,j)$ is the observed number of emissions of symbol O_j from state S_i, $C_\pi(i)$ is the observed number of times state S_i is observed at $t = 1$ and n_i is the number of visits to state S_i then our estimates are:

$$a'_{ij} = \frac{C_a(i,j)}{n_i}$$

and

$$b'_i(j) = \frac{C_b(i,j)}{n_i}$$

Similarly π' could be defined as:

$$\pi'_i = \frac{C_\pi(i)}{S}$$

but this is generally too few observations to be useful. Instead π' may be estimated from the proportion of time observed in each state, considering the whole sequence, not just the beginning:

$$\pi'_i = \frac{n_i}{ST}$$

We can show that λ' maximises $\Pr(\mathcal{X}, \mathcal{Y}|\lambda)$ and thus are the desired maximum likelihood estimates. First we need to define the relative entropy(also known as the 'Kullback-Leibler distance') of two discrete probability distributions p and q as:

$$H(p||q) = \sum_i p(x_i) \log \left(\frac{p(x_i)}{q(x_i)} \right)$$

then:

Lemma 1. *Gibbs' inequality.* $H(p||q) \geq 0$ *with equality if and only if* $p(x_i) = q(x_i)\,, \forall x_i$.

Proof. It is true that $\log(x) \leq x - 1$ with equality iff $x = 1$. Then:

$$\begin{aligned} -H(p||q) &= \sum_i p(x_i) \log \left(\frac{q(x_i)}{p(x_i)} \right) \\ &\leq \sum_i p(x_i) \left(\frac{q(x_i)}{p(x_i)} - 1 \right) \\ &= \sum_i q(x_i) - p(x_i) = 0 \end{aligned}$$

Hence $H(p||q) \geq 0$. For equality we require $\frac{q(x_i)}{p(x_i)} = 1, \forall i$ so that the inequality becomes exact. In other words we require $p(x_i) = q(x_i)$. $\square$

Theorem 1. *Given training sequences and parameters* λ' *then* $\Pr(\mathcal{X}, \mathcal{Y}|\lambda') \geq \Pr(\mathcal{X}, \mathcal{Y}|\lambda)$ *for all* $\lambda \in \Lambda$. *Further,* $\Pr(\mathcal{X}, \mathcal{Y}|\lambda') = \Pr(\mathcal{X}, \mathcal{Y}|\lambda) \iff \lambda' = \lambda$.

Proof. Showing that $\Pr(\mathcal{X}, \mathcal{Y}|\lambda') \geq \Pr(\mathcal{X}, \mathcal{Y}|\lambda)$ is equivalent to showing that $\log \left(\frac{\Pr(\mathcal{X}, \mathcal{Y}|\lambda')}{\Pr(\mathcal{X}, \mathcal{Y}|\lambda)} \right) \geq 0$.

Let C_a^s and C_b^s be the transition counts and emission counts for training sequence s only. Then:

$$\begin{aligned}
\log\left(\frac{\Pr(\mathcal{X},\mathcal{Y}|\lambda')}{\Pr(\mathcal{X},\mathcal{Y}|\lambda)}\right) &= \sum_{s=1}^{S}\log\left(\frac{\Pr(\mathcal{Y}_s|\mathcal{X}_s,\lambda')\Pr(\mathcal{X}_s|\lambda')}{\Pr(\mathcal{Y}_s|\mathcal{X}_s,\lambda)\Pr(\mathcal{X}_s|\lambda)}\right)\\
&= \sum_{s=1}^{S}\left[\log\left(\frac{\Pr(\mathcal{Y}_s|\mathcal{X}_s,\lambda')}{\Pr(\mathcal{Y}_s|\mathcal{X}_s,\lambda)}\right)+\log\left(\frac{\Pr(\mathcal{X}_s|\lambda')}{\Pr(\mathcal{X}_s|\lambda)}\right)\right]\\
&= \sum_{s=1}^{S}\left[\sum_{i=1}^{N}\left[\sum_{k=1}^{K}C_b^s(i,k)\log\left(\frac{b_i'(k)}{b_i(k)}\right)+\sum_{j=1}^{N}C_a^s(i,j)\log\left(\frac{a_{ij}'}{a_{ij}}\right)\right]\right]\\
&\quad+\sum_{i=1}^{N}C_\pi(i)\log\left(\frac{\pi_i'}{\pi_i}\right)\\
&= \sum_{i=1}^{N}\left[\sum_{k=1}^{K}C_b(i,k)\log\left(\frac{b_i'(k)}{b_i(k)}\right)+\sum_{j=1}^{N}C_a(i,j)\log\left(\frac{a_{ij}'}{a_{ij}}\right)\right]\\
&\quad+\sum_{i=1}^{N}C_\pi(i)\log\left(\frac{\pi_i'}{\pi_i}\right)\\
&= \sum_{i=1}^{N}\left[n_i\sum_{k=1}^{K}b_i'(k)\log\left(\frac{b_i'(k)}{b_i(k)}\right)+n_i\sum_{j=1}^{N}a_{ij}'\log\left(\frac{a_{ij}'}{a_{ij}}\right)\right]\\
&\quad+S\sum_{i=1}^{N}\pi_i'\log\left(\frac{\pi_i'}{\pi_i}\right)\\
&\geq 0
\end{aligned}$$

The last line follows since each inner sum is a measure of the relative entropy between the respective distributions and is therefore greater than or equal to 0. It is equal to 0 iff $\lambda = \lambda'$. $\square$

Note that this is not equivalent to maximising $\Pr(\mathcal{Y}|\lambda)$ since:

$$\Pr(\mathcal{Y}|\lambda) = \sum_{\mathbf{x}}\Pr(\mathcal{Y},\mathbf{x}|\lambda)$$

and we have only maximised the expression $\mathbf{x} = \mathcal{X}$ in the sum. Also the fact that λ' maximises $\Pr(\mathcal{X},\mathcal{Y}|\lambda)$ does not mean necessarily that $\mathcal{X} = \mathbf{x}^*$ as found by the Viterbi algorithm. Discriminative training methods that do find λ such that $\mathcal{X} = \mathbf{x}^*$ will not be discussed in this thesis.

2.6.2 Viterbi training

If we don't know the underlying state sequences then our aim becomes to maximises $\Pr(\mathcal{Y}|\lambda)$. For simplicity we will now assume we are training from just one training sequence. One simple method, as outlined in Durbin et al. [1998], is known as Viterbi training. This method proceeds as:

---

1. Guess initial parameters, λ^0, set $t = 1$.
2. Compute optimal parse, $\mathbf{x}^{*t}$, using Viterbi algorithm and previous parameters, λ^{t-1}.
3. Estimate a new set of parameters, λ^t, using maximum likelihood estimates based on observations $\mathcal{Y}$ and labels $\mathbf{x}^{*t}$.
4. Set $t = t + 1$. If $\mathbf{x}^{*t} = \mathbf{x}^{*t+1}$ terminate, otherwise go to step 2.

Since $\mathbf{x}^{*t}$ is discrete this will converge after finite iterations, most likely very quickly. Viterbi training is very simple to implement but is not guaranteed to maximise $\Pr(\mathcal{Y}|\lambda)$. It is likely there exist many local maxima and Viterbi training will only converge locally. More specifically, rather than maximising the likelhood $\Pr(\mathcal{Y}|\lambda)$ the method maximises the conditional likelihood $\Pr(\mathcal{Y}|\lambda, \mathbf{x}^*)$.

2.6.3 Baum-Welch estimation

A more rigorous estimation method when $\mathcal{X}$ is unknown is an implementation of the expectation-maximisation algorithm (EM) (Dempster et al. [1977]) known as Baum-Welch estimation (BW)(Baum et al. [1970]). Following Rabiner we define:

$$\xi_t(i, j) = \Pr(X_t = S_i, X_{t+1} = S_j|\mathcal{Y}, \lambda)$$

We can then compute ξ_t from the forward and backward variables as:

$$\begin{aligned} \xi_t(i, j) &= \frac{\alpha_t(i)a_{ij}b_j(y_t)\beta_{t+1}(j)}{\Pr(\mathcal{Y}|\lambda)} \\ &= \frac{\alpha_t(i)a_{ij}b_j(y_t)\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{n}\alpha_t(i)a_{ij}b_j(y_t)\beta_{t+1}(j)} \end{aligned}$$

Now let:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j)$$

Given the sequence and the current parameters we can interpret $\gamma_t(i)$ as the probability of being in state S_i at time t. Summing each over t we get:

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from state } S_i$$

and

$$\sum_{t=1}^{T-1} \xi_t(i, j) = \text{expected number of transitions from state } S_i \text{ to state } S_j$$

Baum-Welch estimation uses these quantities as a new set of parameters, λ^*, as follows:

$$\pi_i^* = \gamma_1(i) \tag{2.5}$$
$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{2.6}$$
$$b_j^*(k) = \frac{\sum_{t:y_t=O_k}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)} \tag{2.7}$$

It can be shown that either $\lambda = \lambda^*$ in which case λ represents a (local) maximum likelihood parameter set or $\Pr(\mathcal{Y}|\lambda^*) > \Pr(\mathcal{Y}|\lambda)$. The process can then be iterated until a sufficient degree of convergence is attained. As with Viterbi training BW is does not find the global optimum and so is sensitive to initial parameter choices. Baum-Welch generally performs better than Viterbi training although if the main purpose of an HMM is the use of the Viterbi algorithm then it can be argued Viterbi training is sufficient.

To prove that Baum-Welch estimates converge to a (local) maximum likelihood parameter let $\mathbf{y}$ be fixed and set:

$$p(\lambda) = \Pr(\mathbf{y}|\lambda)$$
$$p(\mathbf{x}, \lambda) = \Pr(\mathbf{x}, \mathbf{y}|\lambda)$$

Then it is true that:

$$p(\lambda) = \sum_{\mathbf{x}} p(\mathbf{x}, \lambda))$$

Also let:

$$\begin{aligned} Q(\lambda, \bar{\lambda}) &= \sum_{\mathbf{x}} \Pr(\mathbf{x}|\mathbf{y}, \lambda) \log[\Pr(\mathbf{x}, \mathbf{y}|\bar{\lambda})] \\ &= \frac{1}{p(\lambda)} \sum_{\mathbf{x}} p(\mathbf{x}, \lambda) \log[p(\mathbf{x}, \bar{\lambda})] \end{aligned}$$

Lemma 2. *If $Q(\lambda, \bar{\lambda}) \geq Q(\lambda, \lambda)$ then $p(\lambda) \geq p(\lambda)$ with equality iff $Q(\lambda, \bar{\lambda}) = Q(\lambda, \lambda)$ and $p(\mathbf{x}, \lambda) = p(\mathbf{x}, \bar{\lambda}), \forall \mathbf{x}$.*

Proof. We show that $\log\left(\frac{p(\bar{\lambda})}{p(\lambda)}\right) \geq 0$:

$$\begin{aligned} \log\left[\frac{p(\bar{\lambda})}{p(\lambda)}\right] &= \log\left[\sum_{\mathbf{x}} \left(\frac{p(\mathbf{x}, \bar{\lambda})}{p(\lambda)}\right)\right] \\ &= \log\left[\sum_{\mathbf{x}} \left(\frac{p(\mathbf{x}, \lambda)}{p(\lambda)}\right)\left(\frac{p(\mathbf{x}, \bar{\lambda})}{p(\mathbf{x}, \lambda)}\right)\right] \\ &\geq \sum_{\mathbf{x}} \left(\frac{p(\mathbf{x}, \lambda)}{p(\lambda)}\right) \log\left[\frac{p(\mathbf{x}, \bar{\lambda})}{p(\mathbf{x}, \lambda)}\right] \qquad (2.8) \\ &= \frac{1}{p(\lambda)} \sum_{\mathbf{x}} p(\mathbf{x}, \lambda) \left(\log[p(\mathbf{x}, \bar{\lambda})] - \log[p(\mathbf{x}, \lambda)]\right) \\ &= Q(\lambda, \bar{\lambda}) - Q(\lambda, \lambda) \geq 0 \end{aligned}$$

In which the inequality at (2.8) follows from Jensen's inequality for concave functions. The equality holds iff $p(\mathbf{x}, \bar{\lambda})/p(\mathbf{x}, \lambda)$ is constant for all $\mathbf{x}$. $\square$

Thus if we can find λ^* that maximises $Q(\lambda, \bar{\lambda})$ then we have found parameters that increase $p(\lambda)$. Here we write:

$$Q(\lambda, \bar{\lambda}) = \sum_{\mathbf{x}} \Pr(\mathbf{x}|\mathbf{y}, \lambda) \left(\log(\bar{\pi}_1) + \sum_{t=1}^{T-1} \log(\bar{a}_{x_{t-1}, x_t}) + \sum_{t=1}^{T} \log(\bar{b}_{x_t}(y_t)) \right)$$

Theorem 2. *The parameter set* λ^* *given in* (2.5), (2.6) *and* (2.7) *maximises* $Q(\lambda, \bar{\lambda})$.

Proof.

$$\begin{aligned}
\frac{\partial Q}{\partial \pi_i} &= \sum_{\mathbf{x}, x_1 = S_i} \Pr(\mathbf{x}|\mathbf{y}, \lambda) \frac{1}{\pi_i} - \sum_{\mathbf{x}, x_1 \neq S_i} \Pr(\mathbf{x}|\mathbf{y}, \lambda) \frac{1}{1 - \pi_i} \\
&= \sum_{\mathbf{x}} \Pr(\mathbf{x}|\mathbf{y}, \lambda) \left[\frac{I_{x_1 = S_i}}{\pi_i} - \frac{I_{x_1 \neq S_i}}{1 - \pi_i} \right] \\
&= \frac{\gamma_1(i)}{\pi_i} - \frac{1 - \gamma_1(i)}{1 - \pi_i}
\end{aligned}$$

Where I is the indicator function. To maximise Q we set $\frac{\partial Q}{\partial \pi_i} = 0$ and find that:

$$\frac{1 - \pi_i}{\pi_i} = \frac{1 - \gamma_1(i)}{\gamma_1(i)}$$

Hence $\pi_i = \gamma_1(i)$. Similarly for $b_j(k)$:

$$\begin{aligned}
\frac{\partial Q}{\partial b_j(k)} &= \sum_{\mathbf{x}} \Pr(\mathbf{x}|\mathbf{y}, \lambda) \sum_{t=1}^{T} \left[\frac{I_{x_t = S_j, y_t = O_k}}{b_j(k)} - \frac{I_{x_t = S_j, y_t \neq O_k}}{1 - b_j(k)} \right] \\
&= \frac{1}{b_j(k)} \sum_{t=1, y_t = O_k}^{T} \gamma_t(j) - \frac{1}{1 - b_j(k)} \sum_{t=1, y_t \neq O_k}^{T} \gamma_t(j) \\
&= \frac{(1 - b_j(k)) \sum_{t=1, y_t = O_k}^{T} \gamma_t(j) - b_j(k) \sum_{t=1, y_t \neq O_k}^{T} \gamma_t(j)}{b_j(k)(1 - b_j(k))} \\
&= \frac{\sum_{t=1, y_t = O_k}^{T} \gamma_t(j) - b_j(k) \sum_{t=1}^{T} \gamma_t(j)}{b_j(k)(1 - b_j(k))}
\end{aligned}$$

Again setting to zero we find:

$$\sum_{t=1, y_t = O_k}^{T} \gamma_t(j) - b_j(k) \sum_{t=1}^{T} \gamma_t(j) = 0$$

Hence:

$$b_j(k) = \frac{\sum_{t=1, y_t = O_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Finally for state transitions, a_{ij}:

$$\begin{aligned}
\frac{\partial Q}{\partial a_{ij}} &= \sum_{\mathbf{x}} \Pr(\mathbf{x}|\mathbf{y},\lambda) \sum_{t=1}^{T-1} \left[\frac{I_{x_t=S_i, x_{t=1}=S_j}}{a_{ij}} - \frac{I_{x_t=S_i, x_{t+1}\neq S_j}}{1-a_{ij}} \right] \\
&= \frac{1}{a_{ij}} \sum_{t=1}^{T-1} \xi_t(i,j) - \frac{1}{1-a_{ij}} \sum_{t=1}^{T-1} [\gamma_t(i) - \xi_t(i,j)] \\
&= \frac{(1-a_{ij}) \sum_{t=1}^{T-1} \xi_t(i,j) - a_{ij} \sum_{t=1}^{T-1} [\gamma_t(i) - \xi_t(i,j)]}{a_{ij}(1-a_{ij})} \\
&= \frac{\sum_{t=1}^{T-1} \xi_t(i,j) - a_{ij} \sum_{t=1}^{T-1} \gamma_t(j)}{a_{ij}(1-a_{ij})}
\end{aligned}$$

Setting to zero we find:

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

These match exactly the formulae of (2.5), (2.6) and (2.7). □

2.7 Generalised hidden Markov models

As noted in Eq. (2.2) the implicit state length distribution of an HMM is geometric. This is a result of the Markovian assumption made in linking consecutive states. Under this assumption a geometric distribution may be the most reasonable but not always. A generalised hidden Markov model (GHMM) allows state length distributions to be modeled explicitly. We include in out set of parameters:

$$d_i(k) = \Pr(X_1 = S_i, \ldots, X_k = S_i, X_{k+1} \neq S_i | X_1 = S_i, \lambda)$$

$d_i(k)$ is the probability of remaining in state S_i for k consecutive steps. Let $D = \{d_i(k)\}$. It is convenient to consider the GHMM as emitting more than one observation (according to d_i) per time step and disallowing self-transitions. The following procedure generates observations using such a model:

---

1. Choose an initial state x_1 according to distribution π.
2. Set $t = 1$.
3. Choose a duration, l_t, according to distribution d_{x_t}.
4. Generate l_t observations according to joint emission distribution B.
5. Transition to state x_{t+1}, according to A, where self-transitions are not allowed: $a_{ii} = 0$.
6. Set $t = t + 1$. If $t > T$ terminate otherwise go to step 3.

Note that in step 4 there is now some freedom in our choice of joint emission distribution B. If we assume each emission is independent, identically distributed (i.i.d.) then, as for an HMM, we obtain:

$$b_i(y_1, \ldots, y_{l_1}) = \prod_{j=1}^{l_1} b_i(y_j)$$

but this assumption is not necessary. This ability to incorporate more information into individual states makes a GHMM more modular. Individual states can generate observations by completely different mechanisms and still be incorporated into the one model. This, coupled with the explicit state length modeling, is the main benefit of using a GHMM. We must now distinguish between the number of observations and the number of states to have emitted those observations. Let T now be the number of states and L be the length of the observation sequence, such that $\sum_{t=1}^{T} l_t = L$. Further let $\mathbf{y}_t$ be the emission at time t and let L_t the the total length of emissions upto and including time t. Then $\Pr(\mathbf{x}, \mathbf{y}|\lambda)$ is now:

$$\Pr(\mathbf{x}, \mathbf{y}|\lambda) = \pi_{x_1} d_{x_1}(l_1) b_{x_1}(\mathbf{y}_1) \prod_{t=2}^{T} a_{x_{t-1}x_t} d_{x_t}(l_t) b_{x_t}(\mathbf{y_t})$$

Preceding sections described algorithms for practical use of an HMM. Decoding and training a GHMM will be discussed in the context of gene prediction.

2.8 HMMs in context

Hidden Markov models fall under a broader class of models known as graphical models. Another example of which, the conditional random field, we will encounter later. Graphical models are a combination of probability theory and graph theory. A complex, uncertain system is represented graphically as a set of interacting random variables. Probability theory ensures the model is consistent and provides an interface for the model to the data. Graph theory provides a natural and intuitive data structure with which efficient algorithms can be designed. Graphical models have found application in areas of statistics, machine learning and statistical mechanics.

Graphical models are generally either directed or undirected. Vertices in the graph represent random variables and the lack of edges in the graph represent conditional independence assumptions. In an undirected model two sets of nodes A and B are conditionally independent given a set C if all nodes in A and B are separated by a node in C. This is written:

$$A \perp\!\!\!\perp B \,|\, C$$

If random variables X, Y, Z are three such nodes and are discrete then we have:

$$X \perp\!\!\!\perp Y \,|\, Z \iff \Pr(X = x, Y = y|Z = z) = \Pr(X = x|Z = z) \Pr(Y = y|Z = z)$$

for $\Pr(Z = z) > 0$. One of the original inspirations for such a model comes from statistical mechanics, in which Gibbs [1902] considered a large system of interacting particles, each particle being in a different energy state. Typically it is assumed that only neighbouring particles exert some significant influence on a given particle, meaning the system satisfies a set of conditional independence criteria as above. Undirected graphical models are also known as Markov random fields, due to this 'local-only' conditional dependence.

Directed models can be defined more loosely as a graph of random variables in which directed edges represent some degree of causality. Both deterministic and probabilistic causal relationships can be modeled. Each node is conditionally independent of its ancestors given its parents. Specifying parameters for a directed model involves specifying the conditional probability distribution at each node. The hidden Markov model is the obvious example to mention here though other popular machine learning models, such as neural networks, are also classed as directed graphical models. As we have seen with HMMs, one of the main uses of a graphical model is inference. Given a set of observations we can ask what was the most likely underlying cause? This cause and effect reversal makes use of Bayes's rule for probabilistic inference so directed graphical models are also known as Bayesian models.

For more information see Castillo et al. [1997] or Lauritzen [1996].

Chapter 3

Computational gene prediction

This chapter develops a GHMM *ab-initio*[1] gene predictor. Gene predictors attempt to infer the precise location of genes and their exon/intron structure based on an input DNA sequence and perhaps additional information. As a brief overview, one of the first gene finders considered accurate enough for use in annotation projects, GeneMark (Borodovsky and McIninch [1993]), used various Markov chains to model coding and non-coding sequence in prokaryotes. The first to use an HMM for finding genes in E. coli was ECOPARSE (Krogh et al. [1994]) and the first to use a GHMM was Genie (Kulp et al. [1996]). Many HMM and GHMM gene finders have since been created, among which Genscan (Burge and Karlin [1997]) is one of the most influential. These *ab initio* gene predictors base their predictions entirely on sequence content. Improvements can be made if other forms of evidence are considered alongside sequence information, such as protein alignments and sequence conservation with a related species. An example is GenomeScan (Yeh et al. [2001]), which takes a set of proteins (amino acid sequences) and aligns them to the sequence to identify regions with homology (similarity) to known genes. Recently, attempts have been made to consider multiple evidence tracks in a less *ad hoc* fashion either through use of more complex probabilistic (G)HMM models (ExonHunter, Brejová [2005]) or through related but more discriminative models such as conditional random fields (*e.g.* Conrad, DeCaprio et al. [2007]; CRAIG, Bernal et al. [2007]). Genscan will be the primary inspiration for our model.

3.1 A generative model of a gene

The GHMM developed here, and HMMs more generally, are generative models. Such a model can be used to generate (or simulate) genomic sequence that contains one or more genes. The goal is to build as accurate a model as possible, such that sequence generated by the model is most realistic. Once the model is constructed it is used in reverse; not to generate a sequence but to take a sequence and predict the most likely set of to states to have generated it. But what constitutes a 'realistic' gene? Chapter 1 outlined the features that are typically present within a protein-coding gene. These features are a result of a number of complex interactions

[1] *ab initio* – from first principles.

between DNA and various RNA and protein molecules. The biology we incorporate into our model is a significant simplification of these interactions.

An example is the mechanism of intron splicing, which Burge describes briefly in his thesis (Burge [1997], ch 4). As mentioned in Chapter 1, splicing is achieved by the splicesome. The spliceosome is composed of five small nuclear RNAs (snRNA) named U1, U2, U4, U5 and U6 as well as numerous proteins. Firstly, the donor site is recognised by U1 which binds with a region stretching six nucleotides downstream of the exon, into the intron, and three nucleotides upstream, into the exon. The branch point/acceptor site is then recognised by the binding of U2 auxiliary factor to a region upstream of the acceptor site, which prompts the binding of U2 to the branch point sequence which lies somewhere between 20 and 40bp upstream of the exon. A mature spliceosome comprised of the remaining snRNAs then forms and splicing can begin. The guanosine (G) at the 5' end of the intron bonds to an adenosine (A) near the 3' end of the intron, causing the donor site to be cleaved. The acceptor site is then cleaved, the exons are ligated together and the intron is released. How this process, in particular splice site recognition, occurs exactly is not well enough understood to model. Instead of modeling these interactions, probabilistic models such as Markov chains are typically employed, in which the whole process is 'flattened' onto the DNA strand and the model is applied in a left to right fashion.

Despite these simplifications gene predictors have proven a successful annotation tool. Indeed, most of this knowledge is not used in gene prediction, we are interested only in patterns found in DNA sequence that are indicative of the relevant interactions. These include changes in nucleotide composition within coding sequence, presence of consensus sequence and changes in nucleotide composition near signals such as splice sites, having exons of typical length, an absence of stop codons in coding sequence. The more of these we include in our GHMM the more 'realistic' our generative model.

In addition to the reduction of complex molecular interactions to the presence or absence of patterns in DNA, it is important to state a number of other simplifications made. We make the following assumptions (none of which are in general true but are good first approximations): genes do not overlap, on either strand; genes do not have alternative splicing; there are no pseudo-genes;[2] apart from splicing, transcripts undergo no other modification affecting translation; genes do not occur in identified repeat sequence; all biological signals contain canonical consensus sequences (as identified in Chapter 1); provided DNA sequences begin and end in intergenic or intronic regions; there exist no sequencing errors in provided sequence – such errors, though rare, can have a large effect if inside a gene. These are necessary for a simple and tractable model and are common assumptions in gene prediction (Majoros [2007], ch. 3).

We now limit ourselves to a genetic alphabet of obvserations, $\mathcal{O} = \{A, C, G, T\}$, and our state space becomes the set of features we wish to infer from the DNA sequence – a simple example might be $\mathcal{S} = \{exon, intron, intergenic\}$. It should be clear how, using the procedure of Section 2.6, we generate sequence containing genes. We need to address our choice of joint emission distribution, B, state transition parameters, A, length distribution parameters, D, and modifications required for Viterbi decoding.

[2] A pseudo-gene is a gene that has lost its protein-coding capability or is otherwise not expressed in the cell.

3.2 Modeling genomic sequence

Coding regions of DNA consist of a sequence of codons. These codons create a protein through a degenerate conversion table. This fixed protein-alphabet and the fact that genes show strong evolutionary conservation means that coding regions are generally more constrained than non-coding regions. These constraints can create recognisable compositional differences between sequence found within exons compared to sequence found within intergenic regions. A good example of this is the human malaria parasite, *Plasmodium falciparum* (plasmodium), whose genome is composed of approximately 14% GC nucleotides in intronic and intergenic regions and approximately 24% in exons (Gardner et al. [2002]).[3]

One way we could generate sequence within exons or within introns is to use a zeroth order Markov chain in which there is no conditional dependence on neighbouring observations. Taking plasmodium as an example we might parameterise as in Table 3.1. Of course such a simple model, especially with the assumed AT and GC symmetry, amounts to nothing more than counting GC% content.

$b_i(k)$	*exon*	*intron*	*intergenic*
A	0.38	0.43	0.43
C	0.12	0.07	0.07
G	0.12	0.07	0.07
T	0.38	0.43	0.43

Table 3.1: Parameters used to generate sequence in plasmodium for use in a 3-state GHMM.

In addition to simple compositional differences, we can model higher order dependencies that may exist between or within codons. An important concept that complicates and improves models is that of codon phase. Starting at zero we number nucleotides in the coding regions of a gene and give nucleotide i a phase of $(i \mod 3)$. The codon position of a nucleotide can create strong dependencies on previous nucleotides. Of the various homogeneous and inhomogeneous models that are possible, a study by Fickett and Tung [1992] finds that models based on hexamer, phase-specific composition are most discriminative.[4] As such, a 3-period inhomogeneous 5th order Markov chain is typically used in gene predictors. Within non-coding states phase is no longer sensible, although dependencies may still exist between adjacent nucleotides. A homogeneous 5th order Markov chain is typically used to model non-coding regions. We will call these models for sequence within coding and non-coding regions content sensors. Since no attempt is made in this chapter to model UTR, we use the term exon to refer to CDS only.

Training a model is a matter of collecting a set of known gene sequences with corresponding annotation and calculating frequencies of observed hexamer emissions in each annotated state, keeping track of phase and using MLE as in Section 2.6.1. It may be the case that there is insufficient training data to observe the occurrence of every possible hexamer ($6^4 = 1296$) in

[3] *Plasmodium falciparum* has the highest AT content of any sequenced genome to date. More typical values (*e.g.* for human) are approximately 60% AT overall.

[4] *Ab initio* gene finder Augustus (Stanke and Waack [2003]) found that using 5th order Markov chains was only more effective than 4th order during development, when their model was less complicated.

which case pseudo-counts may be used to stop the occurrence of such a hexamer yielding zero probability of emission. Later, our model adds a count of 1 to every possible hexamer. To show that such models can be used to identify exons we take DNA sequence from plasmodium and split it into 50bp blocks. We then compute the probability of emission assuming the block is either an exon beginning in phase 0, 1 or 2 or the block is intergenic sequence:

$$\Pr(\mathbf{y}_t|exon^p,\lambda) = \prod_{i=L_{t-1}+1}^{L_t} b^q_{exon}(y_i)$$

$$\Pr(\mathbf{y}_t|intergenic,\lambda) = \prod_{i=L_{t-1}+1}^{L_t} b_{intergenic}(y_i)$$

for $p = 0, 1, 2$ and $q = (p + i - L_{t-1} - 1 \mod 3)$. We can then take the log-ratio for each block:

$$\mathcal{L}(t) = \max_p \log\left(\frac{\Pr(\mathbf{y}_t|exon^p,\lambda)}{\Pr(\mathbf{y}_t|intergenic,\lambda)}\right)$$

Log-ratios above zero indicate that a particular block is more likely to have been generated by an exon. Figure 3.1 shows the results along a portion of chromosome 1, trained on chromosome 9 sequence. The content sensor does a fairly good job of distinguishing coding from non-coding sequence. Short exons are more likely to be missed by the model.

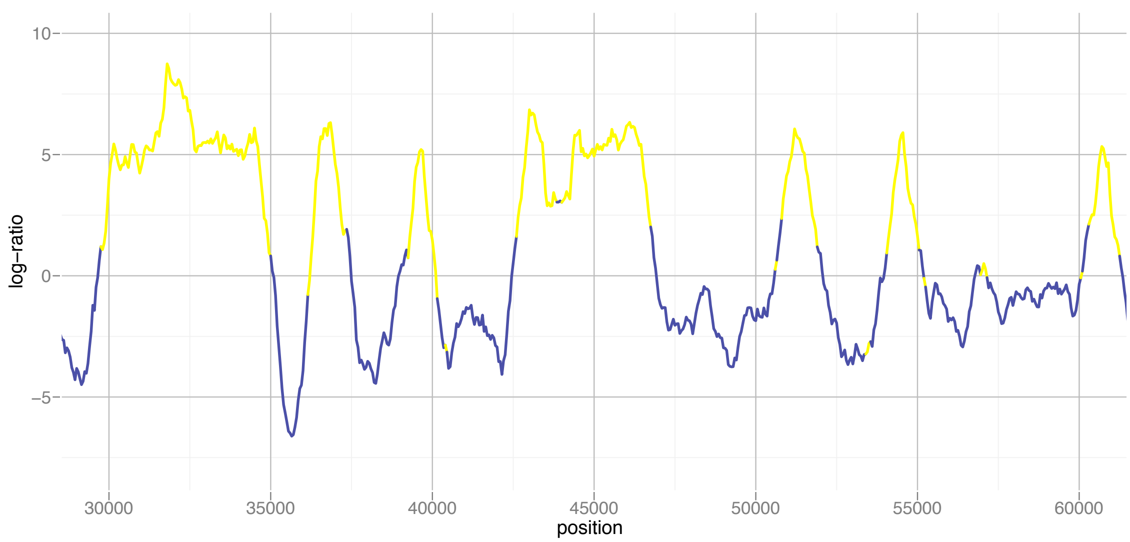


Figure 3.1: Log-ratio scores as a function of position on plasmodium (3D7) chromosome 1. The model is applied to both strands and the maximum score at each position is taken. Sequence from chromosome 9 was used for training. Exon content sensors were trained on all genes identified on the chromosome and the intergenic sensor was trained on both intergenic and intronic sequence. Annotation and sequence was taken from PlasmoDB, `http://plasmodb.org/common/downloads/release-5.4/Pfalciparum/`. Points in yellow indicate protein-coding regions, points in blue indicate non-coding regions.

3.3 Modeling biological signals

Probabilistic modeling of coding and non-coding regions of DNA is not new (*e.g.* Blaisdell [1984]). A key benefit of using HMMs is their ability to incorporate signals sensors in addition to content sensors into the one model. A signal sensor is a probabilistic model of biological signals, such as splice sites, that are present in DNA. Consensus sequences are too short to be used as the sole identifier of a signal. Such an approach would result in many, many false positives meaning longer sequences need to be modeled. The GHMM's modularity is particularly useful as it allows a diverse range of models to be used for this purpose. We mention two possibilities: the weight matrix method (WWM) and the weight array model (WAM).

The weight matrix method models a sequence of length I as a 0th order inhomogeneous Markov chain – each nucleotide is generated according to a position specific distribution. As with content sensors, parameters are maximum likelihood estimates from labeled training sequences. Again taking plasmodium as an example, Table 3.2 shows the WMM for the donor splice site.

position	-3	-2	-1	0	1	2	3	4	5
A	0.43	0.63	0.23	0	0	0.85	0.64	0.37	0.53
C	0.20	0.07	0.05	0	0	0.01	0.08	0.07	0.09
G	0.14	0.09	0.5	1	0	0.04	0.06	0.29	0.09
T	0.22	0.20	0.21	0	1	0.11	0.22	0.27	0.28

Table 3.2: WMM for donor splice sites in plasmodium. Here length, $I = 9$. As for the content sensor of Figure 3.1, the model is trained on all donor sites identified in chromosome 9.

The weight array model allows for higher order dependencies to be incorporated, simply by increasing the order of the WMM. This, in some cases, increases predictive accuracy but additionally it increases the number of parameters that need to be estimated. In practice a limited amount of training data is available and so there exists a compromise between higher emission order and fewer parameters for training. In order to demonstrate that such a model is useful we will identify the set of true donor splice site signals and a disjoint set of 'pseudo' signals that are present on chromosome 1 of plasmodium. We will then train, using MLE, a 2nd order 'true' WAM and a 'pseudo' WAM on all true and some (randomly chosen) pseudo donor splice sites found on chromosome 9. Each model is 11 nucleotides long and the consensus nucleotides, GT, occur at position $i = 4$. For each signal, $\mathbf{s}$, in our test sets we compute:

$$\mathcal{L}(\mathbf{s}) = \log\left(\frac{\Pr(\mathbf{s}|\lambda^{true})}{\Pr(\mathbf{s}|\lambda^{pseudo})}\right)$$

Recall that scores above zero indicate a higher emission probability using the true WAM. Figure 3.2 shows that true signals are not scored below zero often and that pseudo signals are scored above zero more often – true sites are rarely confused for pseudo sites but pseudo sites are confused for true sites.

Neither of these models capture all possible dependencies found near splice sites but are used

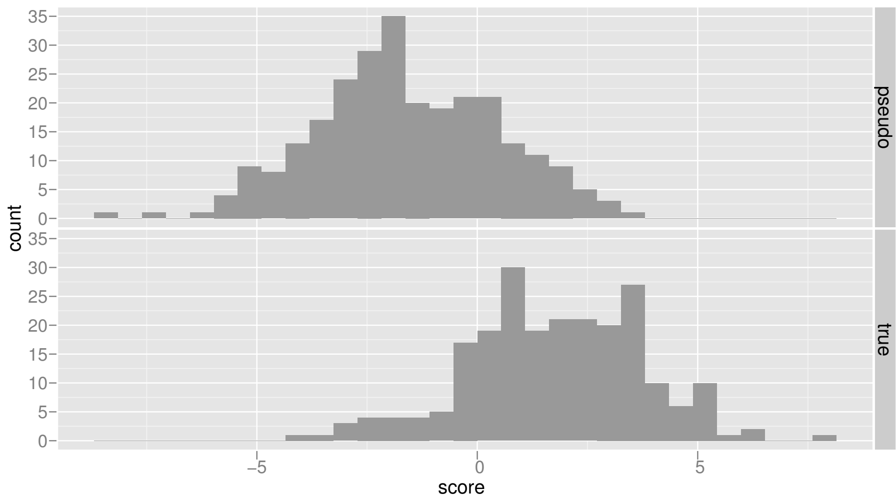


Figure 3.2: Weight array model signal sensor log-ratio scores for the pseudo and true donor splice sites on chromosome 1 of plasmodium. Data was obtained as described in Figure 3.1.

for their simplicity. In particular the above models assume dependencies occur in a left to right linear fashion which is not necessarily how splice site recognition occurs biologically. Genscan's maximal dependence decomposition model is an example of a method that addresses this problem.

3.4 Modeling exon and intron lengths

We make the assumption that intergenic regions are homogeneous and that the chance of a gene beginning at one position is independent of its position in relation to other genes – we assume intergenic regions have a geometric length distribution. Figure 3.3 shows a set of observed exon and intron lengths found in plasmodium. The figure is a simplification since different types of exons have different length distributions. Single exon genes tend to be longer than exons in multi-exon genes. Initial and terminal exons tend to be longer than internal exons. Empirical length distributions can be modeled using a normal distribution kernal density estimate, such that if $\{l_1, \ldots, l_J\}$ is our set of observed lengths then:

$$\Pr(L_t = l) = \frac{1}{\beta} \sum_{j=1}^{J} \psi^+ \left(\frac{l - l_j}{h} \right)$$

where $\psi^+(z)$ is the standard, non-negative, discrete normal density function and h is the bandwidth, chosen by inspection to produce the 'best' [5] (somewhere between over and under-

[5] The optimal bandwidth can be defined rigorously as the bandwidth which gives the optimal rate of convergence in probability to zero of the integrated squared error $\int (\hat{f}(x) - f(x))^2 \, dx$ where $\hat{f}(x)$ is the density estimator of density $f(x)$. For our purposes this degree of optimality is not required.

smoothed) looking distribution. β is a normalising scale factor chosen so that:

$$\sum_{l=0}^{\infty} \Pr(L_t = l) = 1$$

An HMM implicitly uses geometric distributions for both exon and intron lengths. Long features are deemed most unlikely under this assumption, which hinders the modeling of exons, particularly those found in single exon genes. Introns, in comparison, show a more geometric decay past a certain length, making the assumption more valid for non-coding states. A geometric distribution deems short exons and introns the *most* likely which due to biological constraints is unrealistic. Our model will use empirical density estimates for coding states and a geometric distribution for non-coding states. This assumption leaves room for improvement though is common in GHMM gene predictors (*e.g.* Burge and Karlin [1997], Korf et al. [2001], Kulp et al. [1996]). Indeed, the feature of Augustus that most improved its performance in drosophila was the addition of a semi-empirical distribution for intron states.

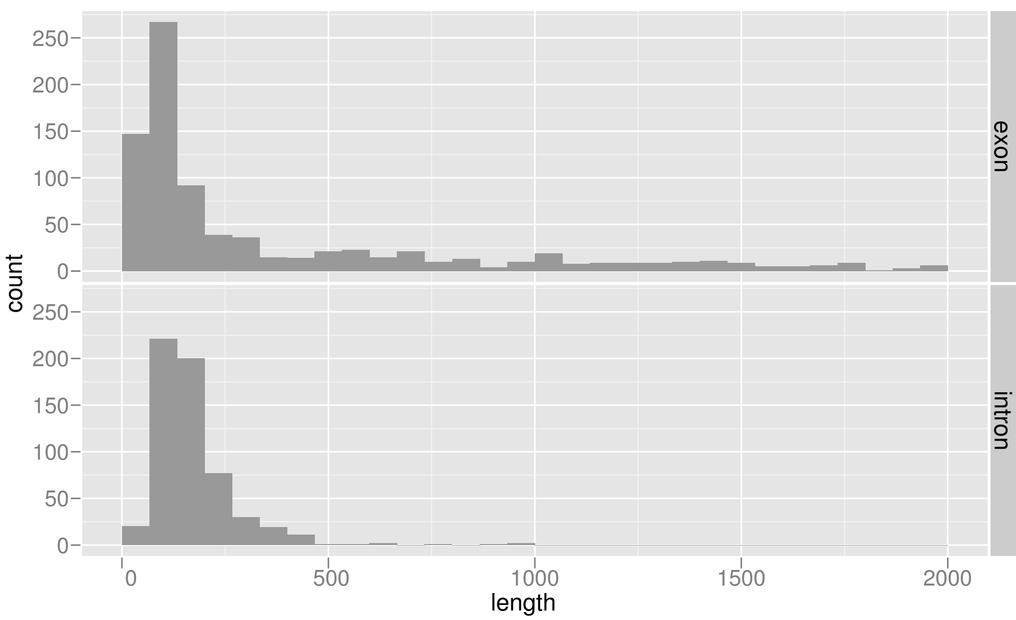


Figure 3.3: Length of exons and introns in genes from chromosome 9 of plasmodium. The x-axis has been truncated to allow for a comparison between the two distributions. Data was obtained as described in Figure 3.1.

3.5 Putting it together

The state space and state transition matrix of a model allows us to include what is known about the structure of a gene. One of the simplest models might be $\mathcal{S} = \{intergenic, exon, intron\}$. An intron is assumed to occur internally and so we accordingly disallow transitions between *intergenic* and *intron* (Figure 3.4). A more realistic state space, and the one we implement, is illustrated in Figure 3.5. Each state within the GHMM is classified as either coding ($\mathcal{C}$) or non-coding ($\mathcal{N}$). To the non-coding states we assign a content sensor and to the coding states we assign a content sensor and two signal sensors – one for each end of the state.

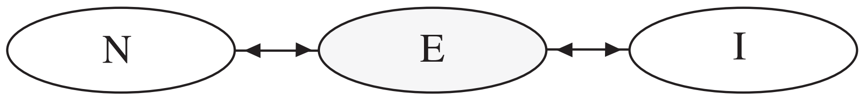


Figure 3.4: State transitions allowed for a simple GHMM in gene prediction. Here N = *intergenic*, E = *exon*, I = *intron*. Gray indicates protein-coding states.

The reason we differentiate between different types of exons is so we can use the correct signal and content sensor for that state. To the initial exon state we assign a start codon signal sensor on the $5'$ end and a donor splice site sensor on the $3'$ end; to the internal exon we assign and donor and acceptor site sensor respectively; to the terminal exon we assign an acceptor site and a stop codon sensor and to the single exon we assign a start and stop codon sensor. The content sensor for the initial exon is trained on initial exon coding sequence only. Similarly for internal exons and the terminal exon content sensor. A (reverse) mirror image of each state on the forward strand exists on the reverse strand, so genes are generated on each strand equally.

Since introns are spliced out before translation they don't affect the phase of coding nucleotides – codons can span introns – and so phase must be tracked across introns. This is accomplished by having three internal exon states and three intron states. On the forward strand denote them as E0F, E1F, E2F and I0F, I1F and I2F respectively. E0F indicates an exon which starts in phase 0, I0F indicates an intron whose preceding exon ended in phase 2 and therefore whose succeeding exon will begin in phase 0. This means the only allowed transitions from I0F are to E0F or ETF. The intergenic state can be revisited indefinitely meaning multiple genes can be generated using such a model.

As an example, consider an emission $\mathbf{y}$ from a single exon gene, on the forward strand, of length L, in which start and stop codon signal sensors of length I_{ATG} and I_{TGA}, respectively, are used. We evaluate as follows:

$$\Pr(\mathbf{y}|\lambda, X = ESF) = \prod_{l=1}^{I_{ATG}} b_{ATG}(y_l) \prod_{l=I_{ATG}+1}^{L-I_{TGA}} b_{ESF}^{p}(y_l) \prod_{l=L-l_{TGA}+1}^{L} b_{TGA}(y_l)$$

p denotes the phase of the current position which is not necessarily aligned with the start of the content sensor. p must be counted from the start of the coding sequence and so can depend on the signal sensor and the state itself. For simplicity these considerations are not included in this mathematical formalism, but only considered in the decoding algorithm. When dealing with states on the reverse strand the probability is evaluated on the reverse complement sequence of the emission, $\mathbf{y^c}$. Note that due to the signal sensors, a coding state will extend beyond the signals that biologically define the region. Provided this is taken into account later when forming a gene prediction, this is not an issue. Figure 3.6 shows sequence containing a gene and its corresponding state space.

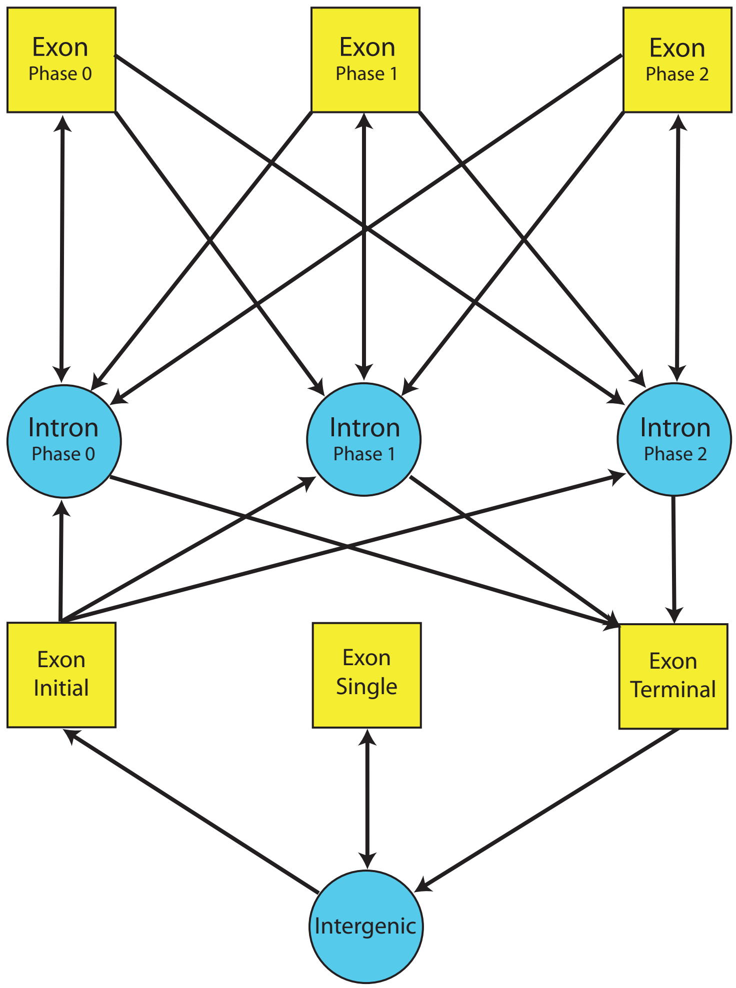


Figure 3.5: Common state transitions allowed for a GHMM in gene prediction. Here only states on the forward strand are shown. The square states are generalised, empirical length distribution states and the circle states are geometric length distribution states.

UTR is not modelled in *ab initio* gene predictors due to its lack of strong composition bias. Comparative methods, for example those including EST alignments, can successfully model UTR (*e.g.* NSCAN_EST, Wei and Brent [2006]).

MLE from labeled training sequences can be used to estimate our transition matrix, A. As previously this merely involves counting transitions between states and normalising them to probabilities. Some transitions are degenerate and don't need to be estimated. As with HMMs it can be shown that this counting of observed transitions and emissions maximises $\Pr(\mathcal{X}, \mathcal{Y}|\lambda)$ and are hence maximum likelihood estimates. We omit the proof.

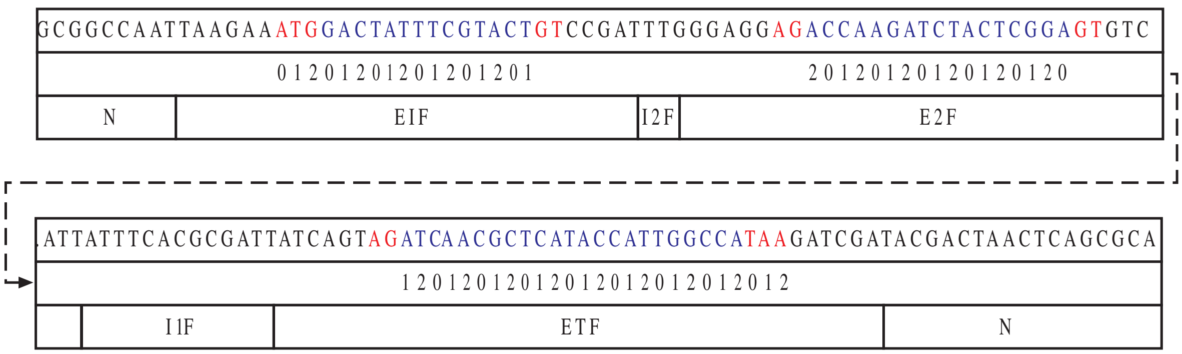


Figure 3.6: States underlying DNA sequence containing a three-exon gene on the forward strand. The top window shows the sequence, the middle window shows the phase of the nucleotide (where applicable) and the bottom window shows the state. Red nucleotides are signals delimiting exon boundaries. Blue nucleotides are CDS.

3.6 Gene prediction as decoding

Now we have described our GHMM in detail, we turn to the task of using it to find genes. For HMMs the Viterbi algorithm can be used to find the optimal parse. For our application the parse corresponds to an optimal set of genes and their constituent exons and introns to have generated the provided DNA sequence. That is, it is a gene prediction. Recall that our goal was to find $\mathbf{x}^*$ as:

$$\begin{aligned}
\mathbf{x}^* &= \operatorname*{argmax}_{\mathbf{x}} \Pr(\mathbf{x}, \mathbf{y}|\lambda) \\
&= \operatorname*{argmax}_{\mathbf{x}} \pi_{x_1} b_{x_1}(\mathbf{y}_1) d_{x_1}(l_1) \prod_{t=2}^{T} a_{x_{t-1}x_t} b_{x_t}(\mathbf{y}_t) d_{x_t}(l_t)
\end{aligned}$$

A direct analogue of the Viterbi algorithm exists for GHMMs but exhibits unacceptable behaviour as a function of sequence length. The problem in decoding as previously is that the variable length state durations mean that not only the immediate position needs to be considered to find the optimal parse thus far, but all preceding positions. This creates $O(L^2)$ behaviour in operation counts.

Therefore, in order to use Viterbi for GHMMs we will make the following assumptions:

1. Non-coding states can only transition to coding states and vice versa, creating a parse of the form $\{N, C, N, C, N, \dots\}$. This is not too large a restriction and it fits already with the model outlined in Figure 3.5.

2. Non-coding states have a geometric length distribution:

$$d_i(l) = a_{ii}^l(1 - a_{ii})$$

   This is unrealistic particularly for short introns, as illustrated in Figure 3.3, but as a first approximation is sufficient.

3. Non-coding states have factorable emission distributions, such that:

$$\Pr(\mathbf{y}_t|x_t, \lambda) = \prod_{i=L_{t-1}+1}^{L_t} \Pr(y_i|x_t, \lambda)$$

   Using Markov chains this presents no problem.

These are assumptions made so as to use the decoding method of Genscan. They are made so that we can, in some sense, treat our GHMM as an ordinary HMM. We consider the model as containing non-coding states only, which by assumption behave like an HMM. The difference is in the transitions – when the model transitions to another non-coding state it does so *through* a coding state. With each transition is associated an emission of length and content described by the coding state's distributions. This simplifies the model and allows us to keep explicit length modeling of coding states.

It is sensible to only consider transitions through coding states that we know will have non-zero probability. For this purpose we need to create a structure of valid transitions. Only transitions occurring through sequence containing consensus signals in the right position, according to the specified coding state, will be non-zero.

One additional and important feature of protein-coding genes yet to be included is the lack of stop codons within coding sequence. No exon can contain an in-phase stop codon. In fact, in prokaryotes, one method of gene prediction (though perhaps flawed: Skovgaard et al. [2001]) is simply through identifying long sequence of DNA that contain no stop codons in a given phase, so-called open reading frames.[6] We will thus identify in-frame stop codons and remove transitions from our structure that contain any.

Let $\Theta_{k,l}$ be a list of transitions, or coding emissions. Each one ending at position l and transitioning into non-coding state S_k. Each item in the list is a tuple: (μ, i, j). Where μ is the start position of the coding emission, S_i is the previous non-coding state and S_j is the coding state. Computing Θ requires some work, Appendix A contains pseudocode for doing so.

Using our created list we now decode. As for Viterbi we will denote:

$$\delta_l(k) = \max_{x_1^{t-1}} \Pr(x_1^{t-1}, X_l = S_k, y_1^l|\lambda)$$

[6]Prokaryotic genes seldom contain introns and so an entire gene will be one contiguous open reading frame. Long open reading frames are not expected to occur by chance.

and we keep track of the optimal parse via necessary structures as before. This time we need to keep track of both the preceding optimal non-coding state and the preceding optimal position. Let:

$$\psi_l(k) = \underset{\substack{(m,i);\\ 1 \le m < l, S_i \in \mathcal{N}, S_j \in \mathcal{C}}}{\operatorname{argmax}} \delta_m(i) \Pr(X_{m+1} = S_j, \ldots, X_l = S_k, y_{m+1}^l | \lambda, X_m = S_i)$$

A slight abuse of notation follows though hopefully it remains clear that $\psi_l(k)$ is the pair containing the optimal preceding non-coding position and state given state S_k at position l. Let $\gamma_i = \Pr(X_t = S_i | X_{t-1} = S_i, \lambda)$ be the probability of not changing state between successive positions. From our previous set of parameters $\gamma_i = 1 - d_i(0), S_i \in \mathcal{N}$.

---

1. Intialisation:

$$\delta_1(i) = \pi_i b_i(y_1)\gamma_i$$

   for $i : S_i \in \mathcal{N}$.

2. Recursion:

$$\begin{aligned}
\delta_{l+1}(k) = &\max \Big\{ \delta_l(k)\gamma_k b_k(y_{l+1}), \\
&\max_{(\mu,i,j)\in\Theta_{k,l}} \left\{ \delta_{\mu-1}(i)[1-\gamma_i] a_{ij} d_j(l-\mu+1) b_j(y_\mu^l) a_{jk} \gamma_k b_k(y_{l+1}) \right\} \Big\} \\
\psi_{l+1}(k) = &\underset{\substack{(l,k);\\ (\mu-1,i):(\mu,i,j)\in\Theta_{k,l}}}{\operatorname{argmax}} \Big\{ \delta_l(k)\gamma_k b_k(y_{l+1}); \\
&\max \left\{ \delta_{\mu-1}(i)[1-\gamma_i] a_{ij} d_j(l-\mu+1) b_j(y_\mu^l) a_{jk} \gamma_k b_k(y_{l+1}) \right\} \Big\}
\end{aligned}$$

   for $k : S_k \in \mathcal{N}, 1 < l < L$.

3. Termination:

$$\begin{aligned}
\delta_{L+1}(k) &= \max \Big\{ \delta_L(k), \max_{(\mu,i,j)\in\Theta_{k,L}} \left\{ \delta_{\mu-1}(i)[1-\gamma_i] a_{ij} d_j(l-\mu+1) b_j(y_\mu^l) a_{jk} \right\} \Big\} \\
\psi_{L+1}(k) &= \underset{\substack{(L,k);\\ (\mu-1,i):(\mu,i,j)\in\Theta_{k,L}}}{\operatorname{argmax}} \Big\{ \delta_L(k); \max \left\{ \delta_{\mu-1}(i)[1-\gamma_i] a_{ij} d_j(l-\mu+1) b_j(y_\mu^l) a_{jk} \right\} \Big\}
\end{aligned}$$

   for $k : S_k \in \mathcal{N}$.

The decomposition of $\delta_{l+1}(k)$ is perhaps less obvious than for a regular HMM so we outline it explicitly. Parameters λ are assumed in the following:

$$\begin{aligned}
\delta_{l+1}(k) &= \max_{x_1^l} \Pr(x_1^l, X_{l+1} = S_k, y_1^{l+1}) \\
&= \max\{ \max_{x_1^{l-1}, X_l = S_k} \Pr(x_1^l, X_{l+1} = S_k, y_1^{l+1}), \max_{x_1^{l-1}, X_l \ne S_k} \Pr(x_1^l, X_{l+1} = S_k, y_1^{l+1}) \}
\end{aligned}$$

where:

$$\begin{aligned}
\max_{x_1^{l-1}, X_l = S_k} \Pr(x_1^l, X_{l+1} = S_k, y_1^{l+1}) &= \max_{x_1^{l-1}} \Pr(x_1^{l-1}, X_{l+1} = S_k, X_l = S_k, y_1^{l+1}) \\
&= \max_{x_1^{l-1}} \Pr(x_1^{l-1}, y_1^l, X_l = S_k) \\
&\quad \times \Pr(X_{l+1} = S_k, Y_{l+1} = y_{l+1} | X_l = S_k) \\
&= \max_{x_1^{l-1}} \Pr(x_1^{l-1}, y_1^l, X_l = S_k) \\
&\quad \times \Pr(X_{l+1} = S_k, | X_l = S_k) \Pr(Y_{l+1} = y_{l+1} | X_{l+1} = S_k) \\
&= \delta_l(k) \gamma_k b_k(y_{l+1})
\end{aligned}$$

and:

$$\begin{aligned}
\max_{x_1^{l-1}, X_l \neq S_k} \Pr(x_1^l, X_{l+1} = S_k, y_1^{l+1}) &= \max_{x_1^{l-1}} \Pr(x_1^{l-1}, X_l \neq S_k, X_{l+1} = S_k, y_1^{l+1}) \\
&= \max_{x_1^{\mu-2}, (\mu,i,j) \in \Theta_{k,l}} \Pr(x_1^{\mu-2}, X_{\mu-1} = S_i, X_\mu = S_j, \ldots, \\
&\quad X_l = S_j, X_{l+1} = S_k, y_1^{l+1}) \\
&= \max_{x_1^{\mu-2}, (\mu,i,j) \in \Theta_{k,l}} \Pr(x_1^{\mu-2}, X_{\mu-1} = S_i, y_1^{\mu-1}) \\
&\quad \times \Pr(X_\mu = S_j, \ldots, X_l = S_j, X_{l+1} = S_k, y_\mu^{l+1} | X_{\mu-1} = S_i) \\
&= \max_{(\mu,i,j) \in \Theta_{k,l}} \delta_{\mu-1}(i) \\
&\quad \times \Pr(X_\mu = S_j, \ldots, X_l = S_j, X_{l+1} = S_k, y_\mu^{l+1} | X_{\mu-1} = S_i) \\
&= \max_{(\mu,i,j) \in \Theta_{k,l}} \delta_{\mu-1}(i) \\
&\quad \times \Pr(X_\mu = S_j, \ldots, X_l = S_j, X_{l+1} = S_k, y_\mu^{l+1} | X_{\mu-1} = S_i, X_\mu \neq S_i) \\
&\quad \times \Pr(X_\mu \neq S_i | X_{\mu-1} = S_i) \\
&= \max_{(\mu,i,j) \in \Theta_{k,l}} \delta_{\mu-1}(i)[1-\gamma_i] a_{ij} d_j(l-\mu+1) b_j(y_\mu^l) \\
&\quad \times \Pr(X_{l+1} = S_k, Y_{l+1} = y_{l+1} | X_l = S_j) \\
&= \max_{(\mu,i,j) \in \Theta_{k,l}} \delta_{\mu-1}(i)[1-\gamma_i] a_{ij} d_j(l-\mu+1) b_j(y_\mu^l) a_{jk} \gamma_k b_k(y_{l+1})
\end{aligned}$$

Since we are working with a GHMM we need to take special care with the state transitions. Note when a non-coding state is transitioned from we include a $[1-\gamma_i]$ term to complete the $d_i(l)$ expression. Whenever we transition into a non-coding state we begin the $d_k(l)$ expression by including a γ_k term. This is not required for transitions to and from the coding state because we model the length distribution explicitly with $d_j(l)$. This gives the result that:

$$\delta_{l+1}(k) = \max\{\delta_l(k)\gamma_i b_i(y_{l+1}), \max_{(\mu,i,j) \in \Theta_{k,l}} \delta_{\mu-1}(i)[1-\gamma_i] a_{ij} d_j(l-\mu) b_j(y_\mu^t) a_{jk} \gamma_k b_j(y_{t+1})\}$$

Decoding involves exclusively multiplications, meaning we can easily use log-probabilities. This stops numerical underflow, since probabilities of long sequences will be very small, and can improve performance since decoding will involve exclusively additions. Coding states can theoretically be of arbitrary length meaning that decoding takes, in theory, $O(L^2)$ operations. However, since we eliminate all coding states with stop codons in-frame and these occur with increasing likelihood as state length increases, in practice coding length is bounded and we decode in $O(L)$ operations.

It is worth noting here that the forward-backward procedure can also be implemented for a GHMM gene predictor and is of some use. Recall the procedure allowed efficient calculation of:

$$\gamma_i(l) = \Pr(X_l = S_i|\mathbf{y}, \lambda)$$

that is, the probability that $X_l = S_i$ given the entire sequence. In a gene prediction context we can adapt this to compute the probability that the model is in an exon at a given position, given the entire sequence. Given a set of predicted exons using the Viterbi algorithm, the forward-backward procedure can be applied to measure the probability that a particular predicted exon is correct. In terms of the forward and backward variables, considering an exon, ϵ, that starts at u and ends at v then:

$$\Pr(\epsilon|\mathbf{y}, \lambda) = \frac{\alpha_{u-1}(i)(1-\gamma_i)a_{ij}b_j(y_u^v)d_j(v-u+1)a_{jk}\gamma_k\beta_{v+1}(k)}{\Pr(\mathbf{y}|\lambda)}$$

Modifications are required to compute the forward and backward variables for a GHMM. Since this was not implemented we do not describe these.

3.6.1 Implementation

The above model was implemented in `Python` 2.5.2 (van Rossum [1995]) and made use of the scientific computing packages `Scipy` (Jones et al. [2001–]) and `Numpy` (Oliphant [2007]) in order to improve speed. The program takes as input a fasta file having the format:

```
>Adh0:13506243:13903243
ATGACTGATCGAGACTACGTAGATATATAAAAAGCGCGCCCCCCC
GTACGTGGCGGCGCCGAAATCGTCAGCTACGATACGCTAGCAAAT
...
```

in which `Adh0` is a reference name and the proceeding two numbers are the start and end positions of the provided sequence. Each file can contain more than one sequence formatted as described. In order to allow for simple modifications most of the model parameters are detailed in a configuration file describing the GHMM, states, signal sensors and content sensors. The program outputs a set of gene predictions taken from the optimal parse as decoded using the above Viterbi algorithm. The output file is in GFF (gene feature format):

```
Adh3 GHMM exon 410985 411401 . + . Gene 1
Adh3 GHMM exon 412747 412767 . + . Gene 1
Adh3 GHMM exon 415576 416211 . -. Gene 2
Adh3 GHMM exon 416276 416749 . -. Gene 2
Adh3 GHMM exon 417100 417111 . -. Gene 2
...
```

in which `Adh3` is a reference matching the sequence fasta file, `GHMM` is the source of the feature, `exon` is the type of feature. The remaining used fields are the start and end position, feature strand and feature name. The dots are unused fields but can represent the frame and a score of some type. Notice that only the exon coordinates are specified. The start and end of the whole gene can be taken as the minimum and maximum of the exon coordinates respectively and the introns, by definition, are the intervening sequence between the exons – thus the program specifies the entire gene structure.

On the 3MB sequence analysed the program takes approximately an hour to run on a 2.33 GHz Intel Core 2 Duo machine with 2GB of memory, running `OS X`. `Python` enables quick application development but is slower to run than an equivalent program written in `C` or similar. Some approaches to remedy this slow performance include: compiling parts of the code using `pyrex` or similar python compiling tools, re-writing routines in `C`, or re-writing the entire application in `C`.

3.7 Prediction results

3.7.1 The dataset

We test the performance of our gene model predictions on a test set taken from drosophila. The region chosen for testing is the *Adh* region of chromosome 2L. This region (see Ashburner et al. [1999]), and drosophila in general are well-studied and are thus suitable for gene prediction evaluation in which a good idea of the 'true' gene annotation is required. It is the same region used by the Genome Annotation aSsessment Project (GASP, Reese et al. [2000]) which allows for a simple comparison to other published gene predictors. The region is approximately 3MB in length and contains a variety of gene structures and varying gene densities. Some gene predictors (for example Genscan) publish performance results based on sequences known to contain only one gene which is unrealistic and tends to overestimate performance – a gene predictor is typically used to annotate an entire region and so it makes most sense to test it accordingly.

The GASP project provides DNA sequence of the region and two annotations, named *std3* and *std1*. *Std1* is a carefully annotated set entirely based on experimentally verified genes while *std3* is a larger set, some of which is not experimentally verified but based on computational approaches. *Std1* and *std3* contain 43 and 222 protein-coding genes respectively. Ideally, a comprehensive *and* experimentally verified annotation is desirable for performance evaluation but this is unreasonable. Splitting the annotation into two sets enables these two criteria to be met individually, so in some sense addresses this issue. Neither set is likely to be the 'truth' but together they provide a reliable set of annotations on which to assess performance. We will subsequently use the term true annotation but recognise this fact.

3.7.2 Performance metrics

Gene prediction can be thought of as a classification problem in which the goal is to correctly classify elements of the provided DNA sequence as either intergenic, intronic or exonic. In measuring the performance we compare the set of predicted elements to the set of true elements. Loosely, the more overlap there is between these two sets the better the predictions. We can define an element to be an individual nucleotide, an exon or a gene. All three are used to judge a gene predictor's performance. Taking our elements to be nucleotides as an example, in comparing the predicted set to the true set we classify each element as a:

- True positive(TP): correctly predicted to be within an exon
- True negative(TN): correctly predicted to not be within an exon
- False positive(FP): incorrectly predicted to be within an exon
- False negative(FN): incorrectly predicted to be not within an exon

Figure 3.7 shows a short set of predictions and true annotation and corresponding nucleotide classification. The fewer false negatives and false positives, the better the prediction. This

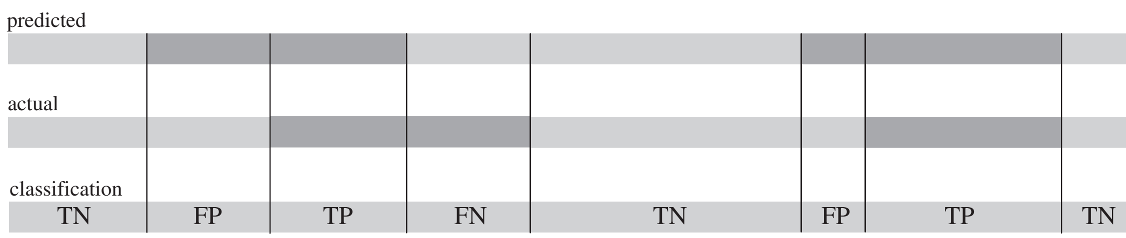


Figure 3.7: Classification of nucleotides when a predicted set is compared to a true set. Nucleotides within exons are shown in dark gray.

motivates the two scores known as sensitivity (Sn) and specificity (Sp):

$$Sn = 100 \times \frac{TP}{TP + FN}$$
$$Sp = 100 \times \frac{TP}{TP + FP}$$

In words sensitivity is the percentage of actual nucleotides lying within exons correctly predicted as such and specificity is the percentage of predicted nucleotides lying within a true exon. These scores are related to type I and type II errors. A more common definition of specificity is in fact:

$$Sp = 100 \times \frac{TN}{TN + FP}$$

but since the number of true negatives at the nucleotide level is typically much greater than the number of false positives this creates an artificially high specificity which is less informative. Therefore, typical in gene prediction is the first definition. A good prediction requires high sensitivity and specificity. Indeed, if a high score in just one of these measures were desired then by predicting every nucleotide or predicting no nucleotide to be lying within an exon we could achieve perfect sensitivity or specificity respectively. Clearly neither of these sets of predictions are useful.

We can similarly define sensitivity and specificity for the sets of predicted and actual exons. We require a true positive to correctly predict the exact start and end positions of the exon.

Then we define the false negatives to be everything else within the true set and the false positives to be everything else within the predicted set. The same approach is adopted at the gene level. The latter definition of specificity is less appropriate here also because it is unclear exactly what a true negative is at either the exon or gene level. Since exon and gene predictions can overlap significantly with the true annotation, but not entirely, it also helps to keep track of the number of missed exons (ME),wrong exons (WE), partial exons (PE) and correct exons (CE) which are summarised in Figure 3.8. We can then measure:

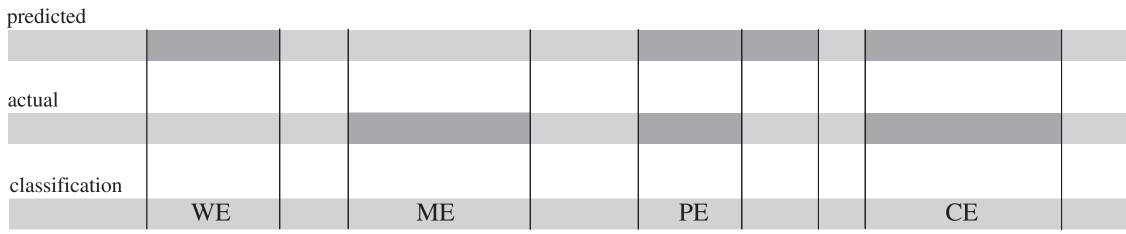


Figure 3.8: Classification of exons when a predicted set is compared to a true set. Exons are shown in dark gray. WE = wrong exon, ME = missing exon, PE = partial exon, CE = correct exon.

$$\text{ME\%} = 100 \times \frac{ME}{TP + FN}$$
$$\text{WE\%} = 100 \times \frac{WE}{TP + FP}$$

A number of other statistics have been suggested for summarising gene prediction performance (see for example those in Burset and Guigó [1996]) but we will content ourselves with sensitivity and specificity. These scores are typically enough to judge performance of one predictor compared to another, however ambiguity can arise when the results are close. Deciding which is 'best' when there is not one consistently better performer is ultimately a choice the user must make. As a general guide, we will judge superior gene level performance over nucleotide and exon level performance.

3.7.3 Results

We experiment with a number of different models in order to determine the best performing. Individual parameters can be trained using maximum likelihood or discriminative methods but the overall model architecture and signal and content sensors used are at the discretion of the author and must often be determined through experimentation – there is no immediately obvious choice or systemic method of determining the optimal model. Given this we will experiment with a few individual aspects of the GHMM independently and take the final model to be the one comprised of the individual models found to be optimally performing. These models are described, with their results, and are compared to a selection of published gene predictors.

Our first set of models learn from the training dataset used by Augustus – the motivation being that it would provide a fair comparison between our model and another published gene predictor (the genes are listed in Appendix B).[7] The training set consists of sequence (including intronic and flanking intergenic) and annotation of 400 genes taken from the drosophila database, FlyBase (Wilson et al. [2008])[8]. The set consists of 76 single exon genes and 324 multiple exon genes. The set was chosen to be independent of the test set being used from the *Adh* region. The base model we begin with is the one described in the preceding sections. Following Augustus, 4th order 3-periodic inhomogenous Markov chains are used for the content sensors within coding states, 4th order homogenous Markov chains are used for the content sensors within non-coding states. 2nd order inhomogenous Markov chains are used for the signal sensors. The WAMs used include 6 nucleotides upstream of the consensus signal and 3 nucleotides downstream of the signal; this means the start/stop sensors are 9 nucleotides in length and splice site sensors are 8 nucleotides. Length distributions for non-coding states are geometric. All parameters are maximum likelihood estimates taken from the training set.

		Length distribution	
		Geometric	Empirical
Nucleotide	Sn (*std1*)	94	94
	Sn (*std3*)	76	76
	Sp	88	88
Exon	Sn (*std1*)	73	76
	Sn (*std3*)	47	48
	Sp	43	46
	ME%	9	9
	WE%	33	29
Gene	Sn (*std1*)	43	46
	Sn (*std3*)	17	18
	Sp	16	17

Table 3.3: Performance of gene model using geometric or empirical length distributions for coding states. Sensitivity is measured against both *std1* and *std3*. Missing exons are measured against *std1*, wrong exons and specificity are measured against *std3*.

Table 3.3 shows the performance of two length distributions used for the coding states. It is worth noting that some scores are computed using *std1* and some are computed using *std3*. It makes most sense to measure specificity against the comprehensive annotation, *std3*, and to measure sensitivity against the reliable annotation, *std1*. Similarly for WE% and ME%. The higher sensitivity and specificity shows that, with everything else constant, modeling coding state lengths using empirical distributions outperforms using geometric distributions. This performance increase is one of the benefits in using generalised hidden Markov models. While strictly this conclusion only applies for the model we have experimented with, we will make the assumption that, in general, empirical is better than geometric and will use it for all subsequent experiments.

Table 3.4 shows the results for a variety of signal sensor and content sensor orders, modified

[7] The training set can be downloaded from `http://augustus.gobics.de/datasets/`

[8] `http://flybase.org/`

independently of each other. While higher order models may be expected to perform better, indeed 5th order Markov chains are typically used for content sensors, the performance really depends on the amount of training data available and the extent to which the order of the Markov chain reflects the extent of the meaningful dependencies that exist in the sequence – both of these can vary between gene predictors and between species – bigger is not always better. Indeed, as the results attest, the best performing models (at least at the gene level) are those with the lowest orders. It is interesting to note that the worst performing model is the 5th order Markov chain content sensor.

		Signal sensor order			Content sensor order			
		0	1	2	2	3	4	5
Nucleotide	Sn (*std1*)	95	95	94	94	95	94	89
	Sn (*std3*)	76	76	76	75	76	76	80
	Sp	89	88	88	88	88	88	90
Exon	Sn (*std1*)	77	75	76	74	77	76	68
	Sn (*std3*)	47	48	48	48	48	48	41
	Sp	46	46	46	48	47	46	40
	ME%	9	8	9	11	8	9	12
	WE%	29	30	29	29	29	29	29
Gene	Sn (*std1*)	49	46	46	49	49	46	31
	Sn (*std3*)	19	19	18	23	21	18	12
	Sp	18	18	17	22	21	17	13

Table 3.4: Performance of gene model using with a range of signal and content sensor orders.

Table 3.5 shows results for a few experiments with signal sensor lengths, still utilising weight array models. The number of combinations to experiment with is huge so we attempt no systematic experiment to determine the best. Instead we experiment with a longer start site signal sensor that extends 33bp upstream of ATG and 3bp downstream and a longer acceptor splice site that extends 33bp upstream and 3bp downstream of AG. The longer acceptor site being an attempt to model the branch point described in Section 3.1 and the longer start site being an attempt to model the Kozak signal present shortly upstream of the start codon (*e.g.* see Kozak [1987], Kozak [1991]). At the exon and nucleotide level there seems negligible different between the models. At the gene level the longer acceptor model appears to decrease performance.

The observation that lower order Markov chains perform better begs the question: is there sufficient training data for the signal and content sensors to learn from? To investigate this question we collect a separate, larger training set for which to train our Markov chains and perform exactly the same set of experiments with length distribution, signal and content order and signal sensor length starting from the same base model. We select 977 genes at random from chromosome 2R and obtain their sequence and annotation using FlyBase's known genes annotation, from the April 2006 assembly. These were downloaded via UCSC Genome Browser's table browser (Karolchik et al. [2008], Karolchik et al. [2004]).[9] Only one

[9] `http://genome.ucsc.edu/`

		Signal sensor length			
		Default	Long ATG	Long AG	Long ATG & long AG
Nucleotide	Sn (*std1*)	94	94	94	94
	Sn (*std3*)	76	76	76	76
	Sp	88	89	89	89
Exon	Sn (*std1*)	76	76	73	73
	Sn (*std3*)	48	48	49	49
	Sp	46	46	47	47
	ME%	9	9	10	10
	WE%	29	30	29	30
Gene	Sn (*std1*)	46	46	40	40
	Sn (*std3*)	18	17	17	17
	Sp	17	17	17	17

Table 3.5: Performance of gene model when length of signal sensors are changed.

transcript from genes with alternative splicing were selected. The genes are listed in Appendix B. Such a large training set is easily obtained for drosophila but for less well characterised species, collecting approximately 1000 known protein-coding genes may present a formidable task. As a comparison, the training datasets provided for the use of GASP participants included annotation and sequence for approximately 400 genes and the coding sequence of approximately 2000 genes. This is comparable in size to our larger training set. Only the Markov chains were retrained; length distribution and state transition parameters were left unchanged.

Tables 3.6-3.8 contain the results. Comparison of the two sets of tables shows that, indeed, a larger training set does improve performance. At the gene level there is a 7-9% increase in both sensitivity and specificity between the two base models which is significant. Empirical state length distributions outperform geometric distributions for coding states as found previously. The improvement is not as significant as might be expected. This may be to do with the kernel density implementation, which used a large bandwidth to prevent large exon lengths being scored with zero probability. This may have, as a result, over-smoothed the distribution, which has a sharp peak at approximately 100bp (as in Figure 3.3). It is less clear which signal sensor and content sensor order is optimal, though (again looking at the gene level statistics) it no longer appears that the lowest order is best. It is interesting that the 5th order Markov chain model continues to show a dramatic performance decrease. Perhaps a larger number of pseudo-counts need to be added to the observed hexamer frequencies. It is unclear whether the longer signal sensor additions provide any performance benefits. The longer AG model arguably shows most improvement over the default model, which contrasts with the previous results. It seems clear that sufficient training data is required to draw any conclusions about the merit of any particular sub-model.

Ultimately, these results only scratch the surface of possible experiments that can be tried but due to available time we are forced to draw a line and continue with the project. For the two sets of experiments an 'optimal' model was selected; at times the close performance of various options meant the decision was made somewhat arbitrarily. The optimal model

		Length distribution	
		Geometric	Empirical
Nucleotide	Sn (*std1*)	98	98
	Sn (*std3*)	79	79
	Sp	89	89
Exon	Sn (*std1*)	77	79
	Sn (*std3*)	51	51
	Sp	52	53
	ME%	6	6
	WE%	24	22
Gene	Sn (*std1*)	54	54
	Sn (*std3*)	25	25
	Sp	23	24

Table 3.6: Performance of gene model using geometric or empirical length distirbutions for coding states. Trained on secondary, larger training set.

		Signal sensor order			Content sensor order			
		0	1	2	2	3	4	5
Nucleotide	Sn (*std1*)	97	98	98	98	98	98	96
	Sn (*std3*)	80	80	79	79	79	79	76
	Sp	89	89	89	88	89	89	89
Exon	Sn (*std1*)	78	78	79	79	79	79	69
	Sn (*std3*)	50	52	51	51	52	51	45
	Sp	53	54	53	53	56	53	45
	ME%	7	6	6	7	6	6	6
	WE%	23	23	23	25	22	23	26
Gene	Sn (*std1*)	51	54	54	51	51	54	29
	Sn (*std3*)	25	26	25	29	29	25	11
	Sp	25	26	24	27	28	24	12

Table 3.7: Performance of model using with a range of signal and content sensor orders. Trained on secondary, larger training set.

		Signal sensor length			
		Default	Long ATG	Long AG	Long ATG & Long AG
Nucleotide	Sn (*std1*)	98	97	98	98
	Sn (*std3*)	79	79	80	79
	Sp	89	90	89	90
Exon	Sn (*std1*)	79	74	78	74
	Sn (*std3*)	51	49	53	51
	Sp	53	53	57	55
	ME%	6	5	5	5
	WE%	23	22	22	21
Gene	Sn (*std1*)	54	46	51	46
	Sn (*std3*)	25	23	29	25
	Sp	24	24	28	25

Table 3.8: Performance of model when length of signal sensors are changed. Trained on secondary, larger training set.

from the first set (denoted GHMM1) uses empirical length distributions for coding states, 2nd order and default length signal sensors and 2nd order content sensors. The optimal model from the second set (GHMM2) uses empirical length distributions for coding states, 2nd order signal sensors with a longer AG model and 2nd order content sensors. These results are shown in Table 3.9 and are compared to other published gene predictors tested on the same region and against the same annotation. Both GHMM1 and GHMM2 perform comparably to the older gene predictors Genscan, Genie and HMMGene but are outperformed by Augustus. In general, the training set used for GHMM1 is of a more realistic size than that of GHMM2, so its results are in some sense more important. Considering that GHMM1 and Augustus are trained on the same dataset there is a significant difference in performance. The model implemented in Augustus is significantly more detailed, including a semi-empirical intron length distribution model and interpolated Markov chain content sensors. The results of GHMM2 indicate that, by training on a larger dataset, we are able to approach the performance of a sophisticated, state-of-the-art gene predictor. The Genscan webserver was used for the Genscan predictions, which unfortunately only provides a Human trained vertebrate parameter set. This no doubt adversely effected the results. Figure 3.9 shows an example annotation taken from the UCSC genome browser (Kent et al. [2002]).

		Gene predictor					
		GHMM1	GHMM2	Augustus	Genscan	Genie	HMMGene
Nucleotide	Sn (*std1*)	96	98	98	96	97	97
	Sn (*std3*)	74	79	-	97	-	-
	Sp	88	88	93	59	92	91
Exon	Sn (*std1*)	74	77	86	60	70	68
	Sn (*std3*)	48	52	-	75	-	-
	Sp	48	55	66	38	57	53
	ME%	11	6	-	14	8	5
	WE%	29	24	-	55	17	20
Gene	Sn (*std1*)	49	51	71	34	40	35
	Sn (*std3*)	23	32	-	49	-	-
	Sp	22	29	39	22	29	30

Table 3.9: Performance of the two optimal GHMM models compared to other published gene predictors. Augustus results taken from Stanke and Waack [2003]. Genie and HMMGene results taken from Reese et al. [2000]. Genscan results taken from webserver, `http://genes.mit.edu/GENSCAN.html`

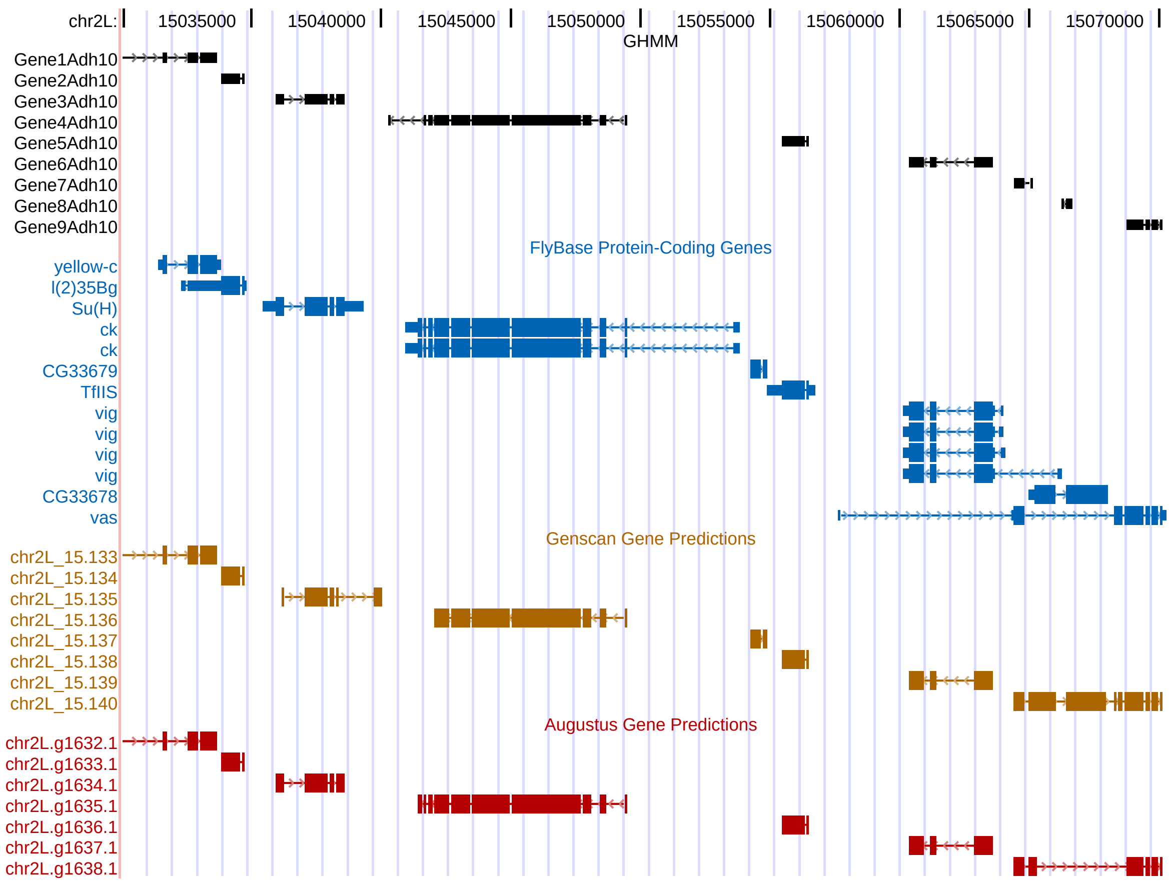


Figure 3.9: Example of GHMM2 gene predictor's performance (top) alongside the FlyBase annotation of known protein-coding genes, Genscan and Augustus gene predictions. Taken from UCSC genome browser, drosophila, April 2004 assembly.

Chapter 4

Transcript mapping

Before delving into transcript mapping and tiling array analysis it is worth asking why similar methods to those used in protein-coding gene prediction can't also be used to find ncRNAs. Protein-coding genes have a well defined structure enforced by the mechanisms of splicing and translation. In contrast, non-coding RNAs perform a wide range of functions and therefore assume a wide range of structures. Since not all of these functions are understood the task of identifying these genes is much harder. Indeed, Section 1.3 highlighted the extreme difference in size ncRNAs can exhibit. Rivas and Eddy [2000] finds little to no statistical signal present in ncRNAs that would aid an *ab initio* gene predictor such as the one implemented in this text (though some alternative methods do exist for some classes of RNA, see Meyer [2007]). Given this, large-scale experiments such as tiling array experiments are one way of finding ncRNAs. It is important to make the distinction between ncRNA prediction and transcript mapping. Transcript mapping identifies the location of potential genes or ncRNAs but makes no attempt to form a discrete prediction in the way a gene predictor does. This chapter will look at current methods of tiling array analysis and investigate a scoring statistic appropriate for the task of transcript mapping.

4.1 The dataset

The data we will be analysing for this and proceeding chapters is was obtained from *Drosophila melanogaster* (Manak et al. [2006]). Total RNA samples longer than 200bp were taken every 2 hours for the first 24 hours of embryonic development and hyrbidised to a single array. The array tiles the non-repeat portion of the genome on average every 35bp using probes of 25 nucleotides in length. For each time point 3 replicate hybridisations were performed, taken from the same biological sample. The raw data and processed 'graph' data can both be obtained online.[1][2] We will perform some analysis on the raw data and some analysis on the processed graph data. Of the number of tiling datasets available this one was chosen due

[1]Raw data is available on Gene Expression Omnibus(GEO): `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE5514`

[2]Graph data are available at `http://transcriptome.affymetrix.com/publication/drosophila_development`

to the large number of time points available. Taking samples at a number of points provides a larger range of gene expression profiles and therefore evidence supporting of a larger set of genes.

We will continue to analyse the *Adh* region of chromosome 2L. The sequence and annotation are required to match the corresponding probe positions from the tiling array. The April 2004 assembly downloaded from the UCSC genome browser was found to align with the probe positions. Sequence was downloaded between positions 13506243 and 16425263. The FlyBase annotation of known protein-coding genes annotation for the assembly were downloaded from the UCSC table browser. Similarly the FlyBase annotation of known non-coding genes were also downloaded, although only 13 were found without the region of interest. All of these ncRNAs are shorter than 200bp in length so may not be represented in the current dataset. A curated annotation of expressed genes was created. Genes showing evidence of expression in the dataset of Arbeitman et al. [2002] (viewable on FlyBase) and showing overlap with the transfrags identified by Manak et al. [2006] were selected. This selection was done by eye and so likely biases the annotation towards highly expressed genes, more easily identified. In addition to a true positive set of probes that show evidence of expression we require a set of known true negatives. The set of probes outside of transfrags and outside of known genes was taken for this purpose.

4.2 Microarray preprocessing

Before we can analyse the data, a number of preprocessing steps are required. Recall that the goal of a microarray experiment is to quantify amounts of RNA transcripts present in one or more samples. There are, however, many factors beside RNA abundance that contribute to an observed intensity. The process of normalisation involves correcting for both intra- and inter-array effects of non-biological origin.

Non-specific binding of the sample to the array and optical noise introduced from the scanner contribute to background noise. A number of methods have been proposed to address this intra-array variability. Affymetrix arrays pair each biologically meaningful probe (perfect match, PM) with an adjacent mismatch probe (MM). The MM probe sequence is identical to the PM probe except for the middle (13th) base. The intention is to measure non-specific binding, which in theory will hybridise equally to both the PM and MM probes. The simplest form of background correction is then simply to use the difference as an estimate of abundance: $PM - MM$. Irizarry et al. [2003] finds this method variance increasing.

An alternative is known as GC-RMA background correction (Wu et al. [2004]) and separates the optical correction and non-specific binding correction into two processes.[3] Firstly, optical background correction is performed in which the observed intensity (PM) for each point is assumed to be the convolution of a normally distributed background noise random variable ($bg \sim \mathrm{N}(\mu, \sigma^2)$) and an exponentially distributed signal random variable ($s \sim \mathrm{Exp}(\alpha)$). Using

[3] The full GC-RMA method adds a non-specific binding correction to the RMA procedure, in which norm-exp background correction, quantile normalisation and a probe-level summarisation model is applied. (Irizarry et al. [2003])

this model the signal is estimated by the background correction:

$$\begin{aligned} B(PM) &= \mathrm{E}(s|PM) \\ &= a + b\frac{\psi(\frac{a}{b}) - \psi\left(\frac{PM-a}{b}\right)}{\Psi(\frac{a}{b}) - \Psi\left(\frac{PM-a}{b}\right) - 1} \end{aligned}$$

where $a = PM - \mu - \sigma^2\alpha, b = \sigma$ and where ψ and Ψ are respectively the normal density function and normal cumulative density function.

Non-specific binding is then corrected for by fitting a model which removes the effect of probes' binding affinities. Binding affinity is dependent on the probe sequence and is an important factor to correct for in tiling arrays (Royce et al. [2007]). For example there exist three hydrogen bonds between complementary G-C nucleotides while only two between complementary A-T nucleotides and as a result high G/C content in probe sequence increases its affinity. Naef and Magnasco [2003] develop a model for estimating this probe sequence dependent affinity, α, as:

$$\alpha = \sum_{k=1}^{25} \sum_{j\in\{A,T,G,C\}} \mu_{j,k} I_{b_k=j}$$

where $\mu_{j,k} = \sum_{l=0}^{3} \beta_{j,l} k^l$ and k indicates position along a given probe, j indicates the base letter, b_k represents the base at k, I is the indicator function and $\mu_{j,k}$ is the contribution to the affinity of base j and k. The model is fitted to the log intensities of many arrays (*i.e.* coefficients $\beta_{j,l}$ are determined) using least squares. These estimates are used in the model for PM and MM probe pairs:

$$\begin{aligned} PM &= O_{PM} + N_{PM} + S \\ MM &= O_{MM} + N_{MM} \end{aligned}$$

where O is optical noise assumed to be an array-dependent constant, $\log(N_{PM}), \log(N_{MM})$ are bivariate-normal distributed non-specific binding with means μ_{PM}, μ_{MM}, each of variance σ^2 and correlation ρ across probes and S is signal proportional to RNA abundance. μ_{PM} and μ_{MM} are assumed to be proportional to α_{PM} and α_{MM}. The maximum likelihood estimate of S then is:

$$\hat{S} = \begin{cases} PM - O - \hat{N}_{PM}, & PM - \hat{N}_{PM} > m \\ m, & \text{otherwise} \end{cases}$$

where m is the minimum allowed signal, typically $m = 0$, and $\hat{N}_{PM}$ is an estimator related to the probe's binding affinity:

$$\hat{N}_{PM} = \exp\left(\rho \log(MM - O) + \mu_{PM} - \rho\mu_{MM} - (1-\rho^2)\sigma^2\right)$$

This method of background correction can be thought of as a sophisticated version of the $PM - MM$ transformation mentioned above. Refer to Wu et al. [2004] for more detail.

Inter-array effects can be caused by a number of factors including different scanner setting or quantities of dye. Figure 4.1 shows the intensity distributions for each array in the dataset. Quantile normalisation (Bolstad et al. [2003]) corrects for these differences by transforming each distribution to a common, or total, distribution. Silver [2007] summarises the procedure as follows: let G be the empirical cumulative distribution function of the data from one array, $\{x_i\}$, and F be the empirical cumulative distribution function of the entire data set(from all arrays) having inverse F^{-1} then we apply the transformation $x_i' = F^{-1}(G(x_i))$. In practice, the intensities for each array are sorted and each intensity is replaced by the mean of intensities across all arrays in the same rank.

To implement the above methods the `R` (R Development Core Team [2008]) package `aroma.affymetrix` (Bengtsson et al. [2008]) provides an implementation of both GC-RMA background correction and quantile normalisation.

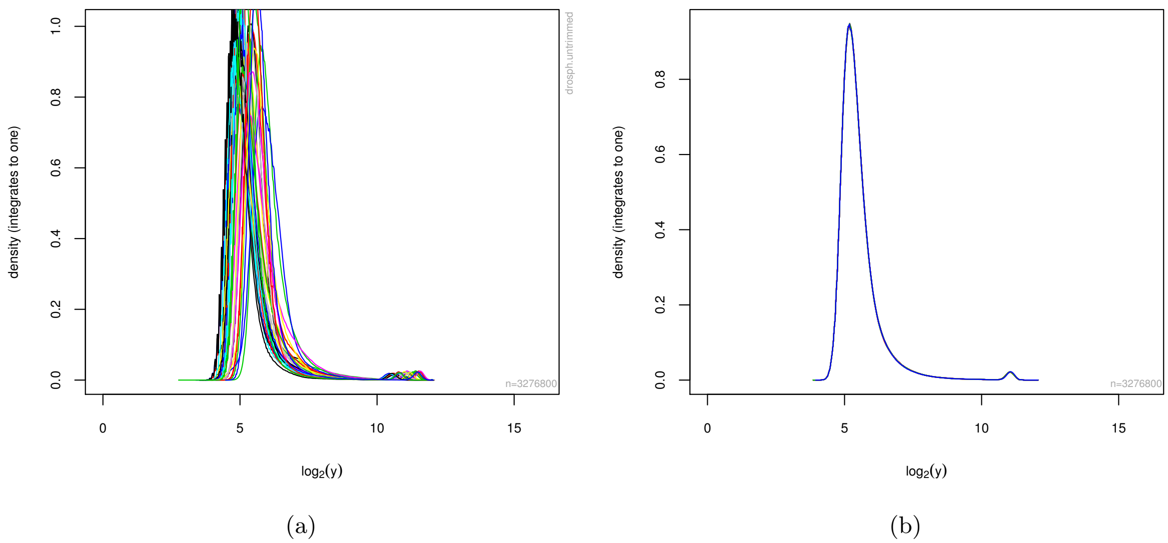


Figure 4.1: Quantile normalisation of raw intensity data forces a common distribution on each array. (a) shows the empirical distribution of the log of raw intensity data for each of the 36 arrays, (b) shows the same densities after quantile normalisation.

4.3 Current methodolgy

Following normalisation is the task of transcript mapping, or determining which probes or set of adjacent probes show evidence of transcription in the experiment. A number of methods have been proposed for converting an analogue signal into a discrete set of predictions. One of the simplest and perhaps most widely used is a sliding window and thresholding approach. For example: He et al. [2007], Kampa et al. [2004], Bertone et al. [2004]. This is the method adopted by Manak et al. [2006] for our current dataset which we now describe. At each time point and for each genomic position the median of all pairwise average PM-MM values for all probe pairs within a 101bp window is taken, using quantile normalised data with a scaled median of 25. Consecutive probes significantly above background levels are deemed transcribed. More specifically, a cutoff is determined which is associated with a 5% false positive rate in negative bacterial controls present on the array. Consecutive probes greater than the cutoff threshold, separated by less than 50bp and stretching longer than 90bp are called transcribed. These data and the resulting 'transfrags', or transcribed fragments, are available online.[2] Sliding window approaches smooth data which reduces resolution. Such methods thus are less likely to identify small RNAs.

The task has also been tackled with HMMs (Zeller et al. [2008], Du et al. [2006], Li et al. [2005], Ji and Wong [2005]). The most simple examples consist of two states – expressed and non-expressed. Transfrag regions are defined as self-transitions in the expressed state. Unlike gene prediction HMMs which emit sequence, these emit scores related to intensity. The scores could be signal scores, smoothed signal scores as in a sliding window or t-statistics, for example. Once appropriate parameters have been estimated, Viterbi decoding can then be used to obtain a discrete set of predicted transcribed elements. By treating neighbouring emissions as independent, HMMs ignore the fact that adjacent probe scores are likely to be correlated, which can be useful in identifying transcribed elements.

Segmentation models have been used successfully in Huber et al. [2006], David et al. [2006], Piccolboni [2007]. For example, Huber fits a piece-wise constant function to segments of expression scores. The segments are chosen to maximise the fit while minimising the model complexity, or number of segments. The assumption is that once properly normalised expression signal will be roughly constant in transcribed regions. Segmentation methods remove the required independence assumption between neighbouring emissions but are otherwise similar to HMMs.

4.4 Correlated intensities

We will explore a summarisation score suitable for our gene prediction task. Effectively combining tiling array data from multiple time-points or cell-types has not sufficiently been addressed. TileMap (Ji and Wong [2005]) and TranscriptionDetector (Halasz et al. [2006]) are two such examples but require control array and information on control probes respectively. The goal of gene prediction and the goal of our project is the accurate identification of functional elements supported by tiling array data. It is then desirable to have a summarisation statistic that can effectively combine information across different time points or cell-types. Although determining expression profiles using tiling data for single time points is informative for the present chapter we will ignore this task; the task already having being addressed by Manak et al. [2006].

The statistics we investigate are based on the following observation: intensities of neighbouring probes within a transcribed region will be correlated. Figure 4.2 contains three examples. In general the level of expression will vary between time points meaning within expressed regions a linear relationship will exist between intensities of neighbouring probes. No such relationship will exist between regions not expressed at any sampled time point. More formally we can write a model as follows: let x_{ij} and y_{ij} be the intensities of neighbouring probes where i indicates time and j indicates replicate. We have 12 time points and 3 replicates. Then let:

$$x_{ij} = \xi_i + \epsilon_{ij} \tag{4.1}$$
$$y_{ij} = \eta_i + \delta_{ij} \tag{4.2}$$

We will further assume that:

$$\xi_i = \alpha\tau_i \tag{4.3}$$
$$\eta_i = \beta\tau_i \tag{4.4}$$

α and β can be interpreted as probe affinities for probes x and y. τ_i is the true expression at time i. ϵ and δ are zero-mean additive noise. $\xi, \eta, \tau, \epsilon, \delta$ have variances $\sigma_\xi^2, \sigma_\eta^2, \sigma_\tau^2, \sigma_\epsilon^2, \sigma_\delta^2$ respectively. We expect σ_τ^2 to be greater when the region is differentially expressed. A number of methods were implemented to utilise this fact, which we outline briefly.

4.4.1 Method 1

Our first attempt assumes genes show some degree of differential expression. Under this assumption we have:

$$\mathrm{corr}(\xi, \eta) = \frac{\mathrm{cov}(\xi, \eta)}{\sqrt{\mathrm{var}(\xi)\,\mathrm{var}(\eta)}} = \frac{\alpha\beta\,\mathrm{var}(\tau)}{\alpha\beta\sqrt{\mathrm{var}(\tau)\,\mathrm{var}(\tau)}} = 1 \tag{4.5}$$

Estimating $\mathrm{corr}(\xi, \eta)$ will reflect how well the data fits this model. Both x_{ij} and y_{ij} are random effects models (see, for example, Snedecor and Cochran [1989]) so we can estimate each requisite component of (4.5) using $E(SSB_x)$ and $E(SSW_x)$, the expected sum of squares between and expected sum of squares within, respectively given by:

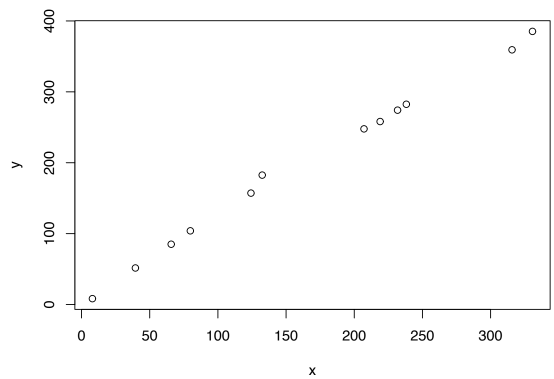
$$E(SSB_x) = \frac{1}{11}\sum_{i=1}^{12}\sum_{j=1}^{3}(x_{i.} - x_{..})^2$$
$$E(SSW_x) = \frac{1}{12(2)}\sum_{i=1}^{12}\sum_{j=1}^{3}(x_{ij} - x_{i.})^2$$

where $x_{i.} = \frac{1}{3}\sum_{j=1}^{3} x_{ij}$ and $x_{..} = \frac{1}{36}\sum_{i=1}^{12}\sum_{j=1}^{3} x_{ij}$. It can be shown that:

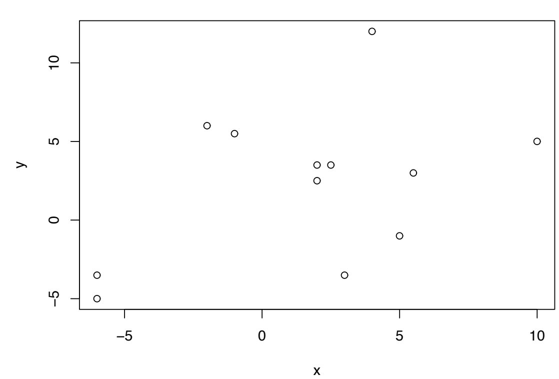
$$E(SSB_x) = 3\sigma_\xi^2 + \sigma_\epsilon^2$$
$$E(SSW_x) = \sigma_\epsilon^2$$

Hence:

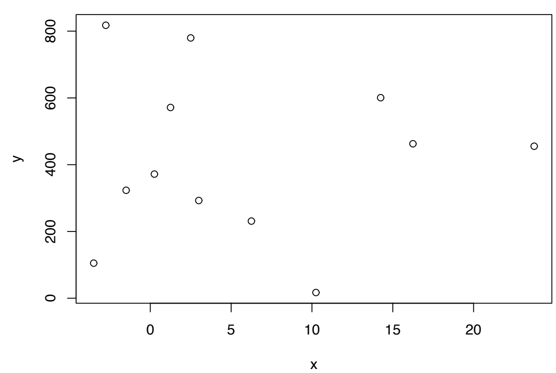
$$\sigma_\xi^2 = \frac{E(SSB_x) - E(SSW_x)}{3}$$
$$\sigma_\eta^2 = \frac{E(SSB_y) - E(SSW_y)}{3}$$



(a) $\rho = 1.00$



(b) $\rho = 0.43$



(c) $\rho = -0.01$

Figure 4.2: Neighbouring probes can be (a) highly correlated in transcribed regions, (b) moderately correlated in non-transcribed regions or (c) uncorrelated between probes bordering transcribed regions. Here corresponding intensities from two adjacent probes are plotted for each time point. One probe is plotted on the x-axis and one probe on the y-axis. Intensities are averaged across replicates.

The same estimates are used for y. For covariance a similar line of reasoning gives:

$$\mathrm{cov}(\xi, \eta) = \frac{E(SPB_{xy}) - E(SPW_{xy})}{3}$$

where $E(SPB_{xy})$ and $E(SPW_{xy})$ are the expected sum products between and within respectively, given by:

$$E(SPB_{xy}) = \frac{1}{11}\sum_{i=1}^{12}\sum_{j=1}^{3}(x_{i.} - x_{..})(y_{i.} - y_{..})$$
$$E(SPW_{xy}) = \frac{1}{12(2)}\sum_{i=1}^{12}\sum_{j=1}^{3}(x_{ij} - x_{i.})(y_{ij} - y_{i.})$$

ANOVA-type estimates of variance are sometimes negative which is nonsense and particularly not useful in this context. Variance estimates too small mean that either the estimated correlation is greater than 1, less than -1 or the correlation is not defined. Figure 4.3 illustrates that the estimates (when defined) vary well beyond what is sensible. For these reasons method 1 was not implemented.

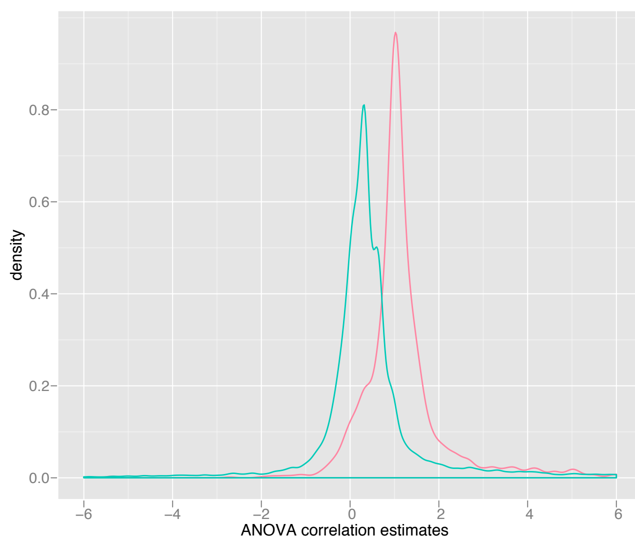


Figure 4.3: Distribution of correlation estimates using method 1 from GC-RMA corrected, quantile normalised dataset. The red curve is the distribution for probes within the expressed annotation and blue curve is the distribution for probes within the non-expressed annotation.

4.4.2 Method 2

The following quantity is then our next attempt:

$$
\begin{aligned}
\operatorname{corr}(X,Y) &= \frac{\operatorname{cov}(x,y)}{\operatorname{var}(x)\operatorname{var}(y)} \\
&= \frac{\operatorname{cov}(\xi,\eta)}{\sqrt{\operatorname{var}(\sigma_\xi^2+\sigma_\epsilon^2)\operatorname{var}(\sigma_\eta^2+\sigma_\delta^2)}} \\
&= \frac{\alpha\beta\sigma_\tau^2}{\sqrt{(\sigma_\xi^2+\sigma_\epsilon^2)(\sigma_\eta^2+\sigma_\delta^2)}} \\
&= \frac{\alpha\beta\sigma_\tau^2}{(\alpha^2\sigma_\tau^2+\sigma_\epsilon^2)(\beta^2\sigma_\tau^2+\sigma_\delta^2)}
\end{aligned}
$$

We obtain a function that is monotonically increasing with σ_τ^2 and so perhaps will make a statistic suitable for the detection of differential expression. It is not clear how to measure $\operatorname{corr}(X,Y)$, however. Naïvely using a sample estimate of all 36 observations is equivalent to treating each observation as independent. Data within replicates is not independent but highly correlated. If we modify our model slightly by averaging the replicate intensities to obtain a single expression value per time point then we have:

$$
\begin{aligned}
x_i &= \xi_i + \bar{\epsilon}_i \\
y_i &= \eta_i + \bar{\delta}_i
\end{aligned}
$$

where $\bar{\epsilon}_i = \sum_{j=1}^3 \epsilon_{ij}$ and $\bar{\delta}_i = \sum_{j=1}^3 \delta_{ij}$. Maintaining the assumption that $\xi_i = \alpha\tau_i$ and $\eta_i = \beta\tau_i$ then:

$$
\begin{aligned}
\operatorname{corr}(X,Y) &= \frac{\operatorname{cov}(x,y)}{\operatorname{var}(x)\operatorname{var}(y)} \\
&= \frac{\operatorname{cov}(\xi,\eta)}{\sqrt{\operatorname{var}(\sigma_\xi^2+\sigma_{\bar{\epsilon}}^2)\operatorname{var}(\sigma_\eta^2+\sigma_{\bar{\delta}}^2)}} \\
&= \frac{\alpha\beta\sigma_\tau^2}{\sqrt{(\sigma_\xi^2+\sigma_{\bar{\epsilon}}^2)(\sigma_\eta^2+\sigma_{\bar{\delta}}^2)}} \\
&= \frac{\alpha\beta\sigma_\tau^2}{(\alpha^2\sigma_\tau^2+\sigma_{\bar{\epsilon}}^2)(\beta^2\sigma_\tau^2+\sigma_{\bar{\delta}}^2)}
\end{aligned}
$$

in which $\sigma_{\bar{\delta}}^2 = \frac{\sigma_\delta^2}{3}$ and $\sigma_{\bar{\epsilon}}^2 = \frac{\sigma_\epsilon^2}{3}$. Now take the sample estimate:

$$
\operatorname{corr}(X,Y) = \frac{12\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{12\sum x_i^2 - (\sum x_i)^2}\sqrt{12\sum y_i^2 - (\sum y_i)^2}}.
$$

To incorporate information from additional neighbouring probes the following side mean correlation statistic (both this statistic and the following MISMC statistic are from Xu [2008]) is adopted:

$$SMC_i = 100 \times \max \left\{ \frac{1}{3} \sum_{j=1}^{3} \text{corr}(x_i, x_{i+j}), \frac{1}{3} \sum_{j=1}^{3} \text{corr}(x_i, x_{i-j}) \right\}$$

The score is simply the maximum of the mean of the correlation between the current probe and three neighbouring probes upstream or the three neighbouring probes downstream. Probes that border transcribed regions will show correlation with only upstream or downstream probes. Larger window sizes reduce noise from chance, highly correlated probes but risk oversmoothing and missing short transcribed fragments. A range of sizes were tried and $w = 3$ was decided as the compromise.

Figure 4.4 shows the correlation scores on three sets of probe intensities. The first set is taken from the GC-RMA corrected, quantile normalised data, the second and third sets are taken from the PM-MM data available online. The second set uses quantile normalised PM-MM values and the third set uses a sliding window method applied to PM-MM values. Each are described in Sections 4.2 and 4.3. The first two sets allow for a comparison of the score between different normalisation methods. The third set was chosen to investigate whether a 'smoothed' average intensity was more discriminative than a non-smoothed intensity.[4] It is clear that different normalisation or summarisation methods effect the behaviour of the SMC statistic. There is a small, highly correlated peak in the non-expressed probes of the GC-RMA dataset that is not present in the PM-MM methods. These non-expressed probes that are highly correlated may adversely affect the statistics' discriminative performance.

Figure 4.5 illustrates how SMC is related to intensity. Intensity is expected to be positively related to correlation in some way. The plots show a number of high intensity probes are not highly correlated with neighbouring probes. This is likely caused by short spikes in intensity. The GC-RMA dataset appears to contain more high intensity, yet lowly correlated probes.

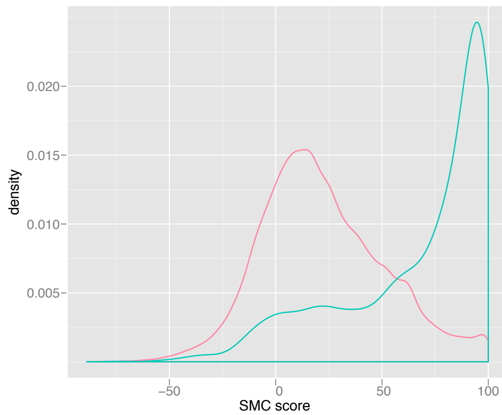
4.4.3 Method 3

To address the problem of short, isolated spikes in expression being missed by the SMC procedure the following addition is made:

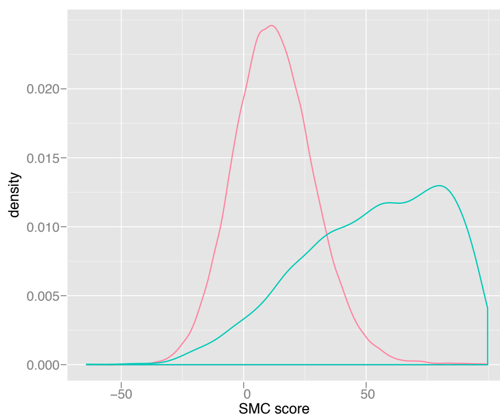
$$MISMC_i = \max \left\{ \text{per}\left(SMC_i\right), \text{per}\left(\sum x_i\right) \right\}$$

in which per(x) indicates the percentile function. Either high expression or high correlation is recognised now. Although the current dataset has been generated by selecting total RNA greater than 200nt in length it is desirable to have a method that can also identify RNAs shorter in length. The main issue with using percentiles as a score is they increase the spread of the majority of probes. In this case the majority of probes are likely to be low intensity or lowly correlated and hence not of interest. Figure 4.6 shows the non-expressed probes being spread across a wide range of MISMC scores.

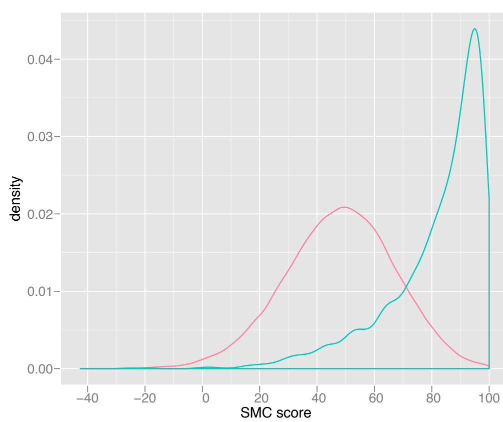
[4] Initial analysis was performed on the PM-MM data due to issues with the GC-RMA implementation. Given time the same sliding window method would be applied to the GC-RMA dataset also.



(a)

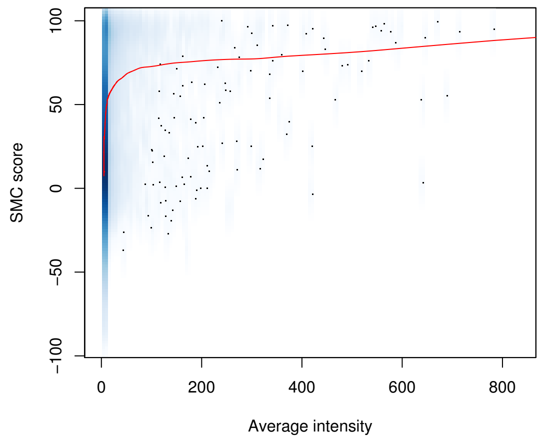


(b)

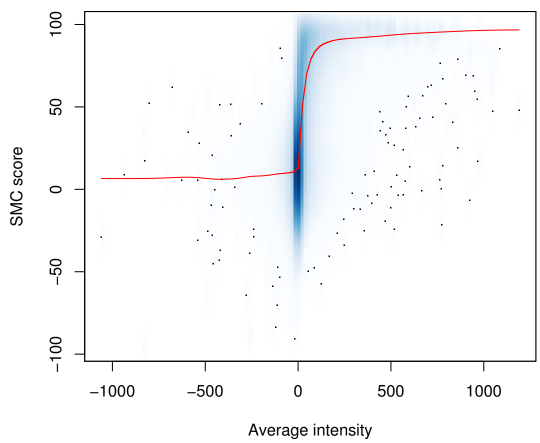


(c)

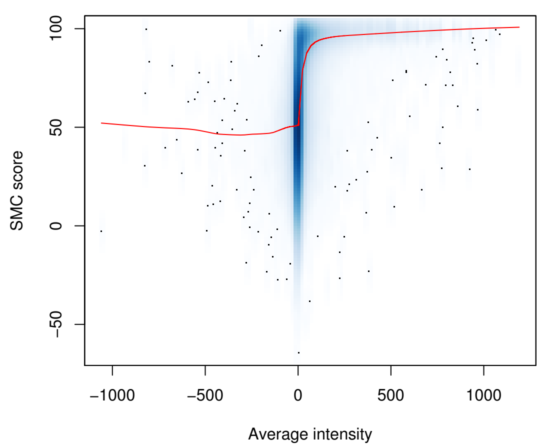
Figure 4.4: Distribution of SMC scores for probes annotated as expressed (blue) and not expressed (red). Intensities have been quantile normalised and (a) GC-RMA background corrected, (b) PM-MM background corrected (c) PM-MM corrected then smoothed via a sliding window approach.



(a) GC-RMA



(b) PM-MM average



(c) PM-MM sliding window

Figure 4.5: Scatter plots of SMC scores versus average of intensity across all time points. A LOESS curve (red) has been fitted to each plot.

Figure 4.7 illustrates how MISMC is related to intensity. Taking the maximum of the percentiles of intensity and correlation has, in some sense, had the desired effect – probes of high intensity are highly scored by MISMC. Probes of a baseline intensity are roughly uniformly distributed as a function of MISMC score. Inspection of both sets of figures shows a difference between the GC-RMA corrected set and the PM-MM corrected sets. By rounding baseline intensities to a common value GC-RMA correlations may be distorted. The PM-MM datasets do not suffer this problem. In fact, one of the drawbacks of the PM-MM correction method, negative expression measures, in this case, is not a drawback at all since negative intensities still contain information about correlation.

Ultimately both SMC and MISMC have issues as statistics for transcript mapping and it is unclear whether they are appropriate for the task. The key benefit is that even low intensity probes can exhibit a strong correlation with neighbours allowing the detection of lowly expressed probes. This is also a drawback however since, by chance, very low intensity probes can be highly correlated obscuring any meaningful correlation that may be present nearby. Transcript mapping techniques based purely on intensities do not suffer this problem. Figure 4.8 highlights this issue.

4.5 A transcript mapping HMM

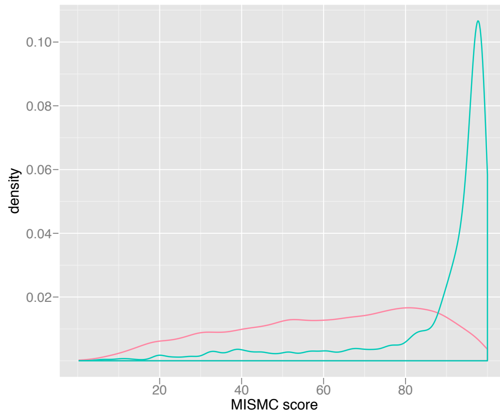
In order to better judge the value of both correlation statistics it is necessary to perform transcript mapping with them. To this end a two-state HMM is constructed consisting of an expressed state (E) and a non-expressed state (N). Since the scores take intensities from a number of times it is worth noting that the 'expressed' state is used to represent probes showing evidence of expression at any of the time points. The HMM operates at the probe-level. In order to train the emission distributions an independent set of approximately 100 expressed genes is identified in the 3MB downstream of the *Adh* region in a similar fashion to that described in Section 4.1. The genes are listed in Appendix B. A true negative set of non-expressed probes was identified in the same region, again as in Section 4.1.

It is appropriate to consider the emissions made by the HMM as continuous. It is then necessary to use a probability density function (pdf) in evaluation. The typical approach for a continuous observation HMM is to use a pdf of the form:

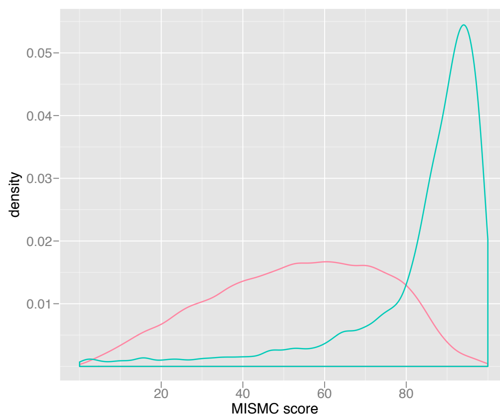
$$f(s|X = S_j) = \sum_{m=1}^{M} c_{jm}\psi\left(\frac{s - \mu_{jm}}{h_{jm}}\right)$$

in which ψ is the gaussian density function and M is the number of mixture components. This form is well behaved and allows the set of parameters $(c_{jm}, \mu_{jm}, h_{jm})$ to be estimated using Baum-Welch estimation. We instead use a kernel density estimate based on a training set of observed emissions within expressed and non-expressed regions. That is, if $\{s_{jm}\}_{m=1}^{M}$ is a set of observed emissions within state j then:

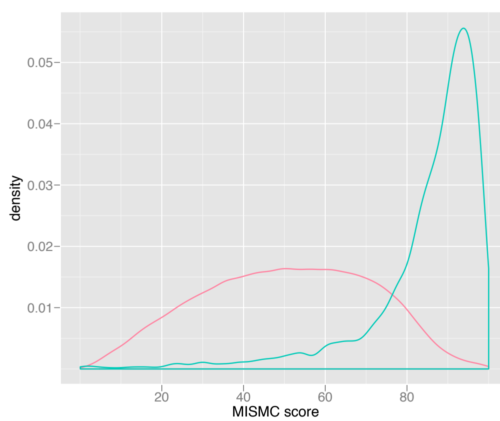
$$f(s|X = S_j) = \sum_{m=1}^{M} \frac{1}{\beta}\frac{1}{\sqrt{2\pi h}} \exp\left[\frac{-(s - s_{jm})^2}{2h}\right]$$



(a)

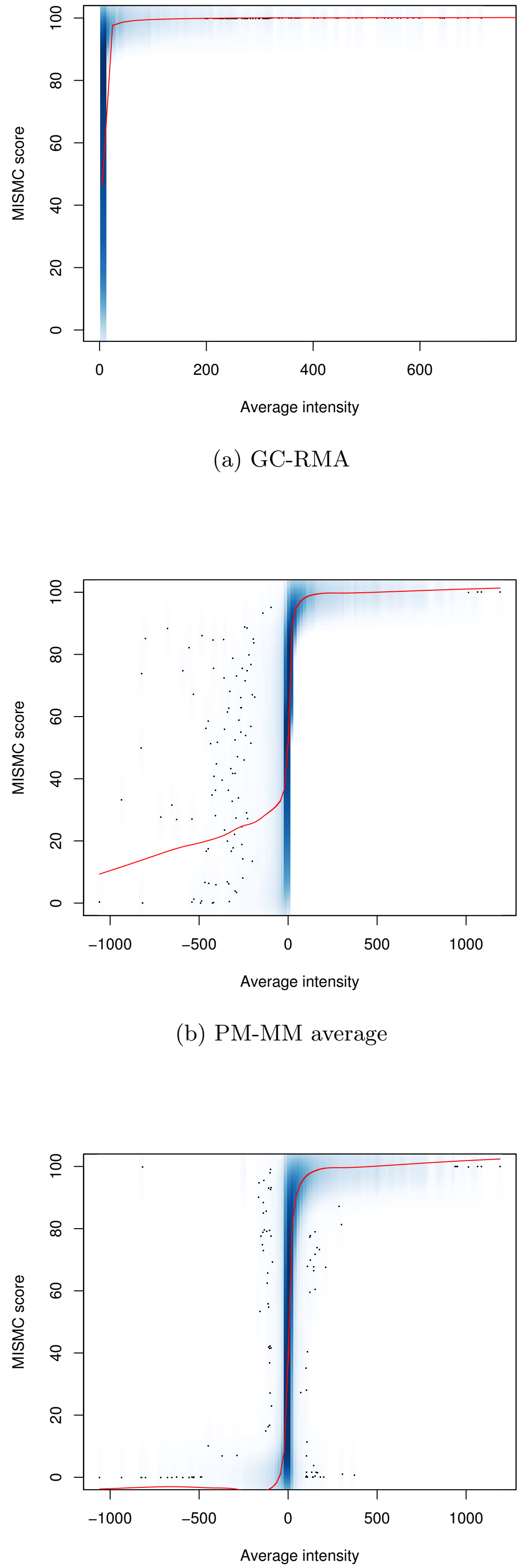


(b)



(c)

Figure 4.6: Distribution of MISMC scores for probes annotated as expressed(blue) and not expressed(red). Intensities have been quantile normalised and (a) GC-RMA background corrected, (b) PM-MM background corrected or (c) PM-MM corrected then smoothed via a sliding window approach.



(a) GC-RMA

(b) PM-MM average

(c) PM-MM sliding window

Figure 4.7: Scatter plots of MISMC scores versus average of intensity across all time points. A LOESS curve (red) has been fitted to each plot.

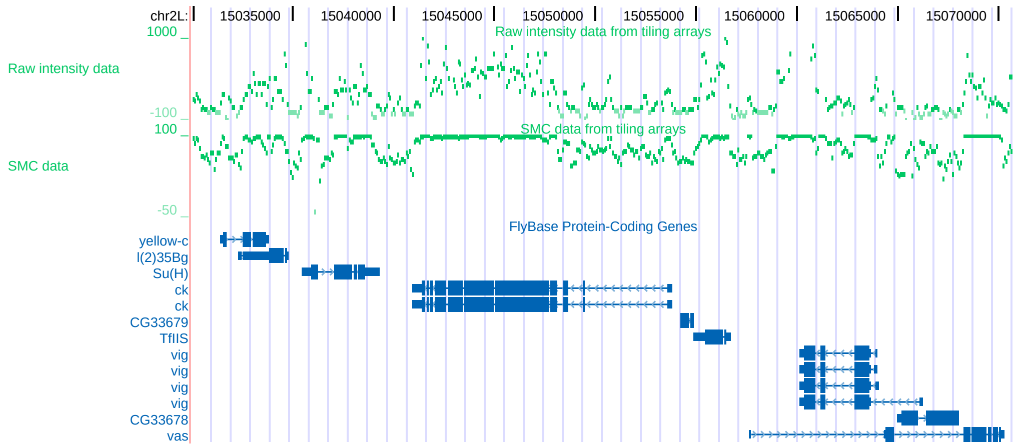


Figure 4.8: Example of PM-MM sliding window intensity scores and corresponding SMC score alongside FlyBase known protein-coding genes illustrating the noise present in both the smoothed intensities and the SMC score.

in which h is the bandwidth and β is chosen so that:

$$\int_a^b f(s|X = S_j)\, ds = 1$$

for minimum and maximum emission scores a and b respectively. We find:

$$\beta = \sum_{m=1}^{M} \Psi\left(\frac{b - s_{jm}}{h}\right) - \Psi\left(\frac{a - s_{jm}}{h}\right)$$

For standard normal distribution function Ψ.

Once this choice has been made it is simple to incorporate scores from the tiling array into the two-state HMM. Empirical kernel densities were trained on the corresponding expressed and non-expressed probesets. Transition parameters need to be determined for each state. For a two state HMM these parameters correspond to the geometric parameters used to model their lengths. The parameters chosen correspond to a mean transfrag length of 160bp and a mean 'intergenic' length of 1000bp. Given the unknown character and frequency of non-coding RNAs is it unclear whether these are reasonable estimates. Viterbi decoding is then used to determine which elements are expressed and which are not expressed. The model was implemented in `Python`.

Later, the models developed here will be incorporated into a larger HMM used for gene finding. Gene finders operate on the nucleotide level so the above transcript mapper needs to be adapted to somehow emit symbols per nucleotide. To model the tiling data at the nucleotide level we propose the following process: let $\{Z_t\}_{t\geq 1}$ be an observable stochastic process with emits a symbol every nucleotide. At each position, t, the process emits a no emission symbol, ω, with probability (w.p.) $\frac{34}{35}$ and emits a tiling array intensity or score with probability $\frac{1}{35}$. The length between consecutive tiling array emission is therefore a geometric random variable with parameter $p = \frac{1}{35}$. The expected length between tiling array emissions is 35 – the resolution of the array. Here, to avoid the complications of using a mixed-type distribution, each tiling array score, ζ_t, is a discrete random variable emitted according to a distribution conditional on the state, $P_{\zeta_t|X_t}(s_t|x_t)$. We have:

$$Z_t = \begin{cases} \omega, & \text{w.p. } \frac{34}{35} \\ \zeta_t, & \text{w.p. } \frac{1}{35} \end{cases}$$

Then:

$$\Pr(Z_t = \omega | X_t = x_t) = \frac{34}{35}$$

and

$$\Pr(Z_t = z_t | X_t = x_t) = \frac{1}{35} P_{\zeta_t|X_t}(z_t|x_t), \qquad z_t \neq \omega$$

If, as before, we denote $\mathbf{z} = z_1^T = \{Z_1 = z_1, Z_2 = z_2, \ldots, Z_T = z_T\}$ then we find:

$$\Pr(\mathbf{z}|\mathbf{x}) = \left(\frac{34}{35}\right)^{T-|\zeta|} \left(\frac{1}{35}\right)^{|\zeta|} \prod_{i \in \zeta} P_{\zeta_t|X_t}(z_i|x_i)\, dz$$

where $\zeta = \{i : z_i \neq \omega\}$. The conditional distributions $P_{\zeta_t|E}$ and $P_{\zeta_t|N}$ are the discrete estimates of the expressed and non-expressed distributions (similar to those of Figures 4.4 and 4.6). For the SMC statistic ζ_t assumes integer values between -100 and 100; for the MISMC statistic ζ_t assumes integer values between 0 and 100. In practice, treating the tiling emissions as discrete random variables, instead of continuous, is likely to have little effect. The continuous density at a given point and the probability of its nearest integer being emitted are, upto a constant scale factor, likely to be very similar. This constant scale factor will have no effect on Viterbi decoding. The current tiling array does not probe the repeat portion of the genome. These regions are assumed to be not transcribed. In these cases a base-line tiling array emission is artificially inserted into the observation data at regular 35bp intervals. An HMM using this emission policy and the emission and transition distributions previously described was also implemented in `Python`.

4.6 Results

Table 4.1 shows the results of the probe level transcript mapping HMM. Interpretation of the results is somewhat complicated since it is unclear whether a low specificity is caused by false positive or unknown true positives. Regardless, it is clear the MISMC statistic performs consistently better than the SMC statistic. The union of all transfrags predicted by Manak identifies more nucleotides in the expressed annotation but identifies fewer of the (limited) non-coding RNAs in the region. This suggests that correlations may be a more sensitive method of finding small or less highly expressed transcribed elements. It may seem surprising that any of the ncRNAs are identified since only RNA fragments longer than 200 nucleotides were hybridised to the array. However, an inspection of the correlation scores show a well defined peak aligning exactly with the ncRNA so it is unlikely these are false positives. These smaller RNAs must have been sampled along with the expected larger RNAs.

Of the HMMs, the best performing dataset (having the highest sensitivity) is the PM-MM sliding window dataset. This is exactly the dataset the transfrag predictions are based on so it is surprising that both correlation methods identify fewer expressed nucleotides than the standard thresholding approach used by Manak. However, note that the sensitivity, measured against the full annotation, is higher for the HMM. Genes in the expressed annotation are predominantly highly expressed. Genes in the full annotation, however, will exhibit a range of expression levels. It may be that the correlation statistics are better able to detect the moderately or lowly expressed genes, hence the improved performance, relative to the full annotation, of the HMM.

The GC-RMA dataset performs worse than either PM-MM datasets. As mentioned previously, this may in part be due to the method's minimum intensity threshold. Neighbouring probes which show base-line intensities for all but one or two timepoints can be highly correlated. The implementation of GC-RMA provided by `aroma.affymetrix` unfortunately does not allow this threshold to be changed. It would likely be worth redoing the analysis using a different implementation.

Looking at the results of the HMM using the MISMC statistic, based on the PM-MM sliding window data: the mean transfrag length is 366bp. Of these transfrags, 44% correspond to coding sequence, 24% to intronic sequence, 17% to UTR and 14% to intergenic regions. In total 17% of the 2.9 million bp are identified as expressed at some point in drosophila embyrogenesis. In contrast, Manak finds the mean transfrag length to be 328bp and classifies transfrags (in this region) as 46% exonic, 24% intronic, 17% UTR and 13% intergenic sequence with 14% of the region predicted as transcribed at some time point. These figures differ slightly from the genome wide statistics reported by Manak though remain indicative of wide-spread, unannotated expression in drosophila. The number of predicted transcribed elements varies, in particular there is a large discrepancy between the number of elements identified by Manak and the HMMs. The Manak dataset is a union of separate transcribed elements across all 12 time points and thus may count the same transcript more than once and over-estimate the number of discrete transcribed elements.

Probe level transcript mapping							
	GC-RMA		PM-MM		PM-MM sliding		
	SMC	MISMC	SMC	MISMC	SMC	MISMC	transfrags
Sn (expressed)	67	74	72	77	73	78	82
Sn (all genes)	38	44	43	45	45	47	44
Sn (ncRNA)	0	0	0	12	45	48	16
Sp	50	49	47	49	45	44	44
# elements	874	769	598	727	617	1152	6683

Table 4.1: Performance of probe-level HMM transcript mapping methods compared to the union of transfrags predicted in Manak et al. [2006] across all time points. Sensitivity and specificity are measured at the nucleotide level.

Table 4.2 shows the results of the nucleotide-level transcript mapping HMM. The MISMC statistic performs better than the SMC statistic and the PM-MM sliding window dataset, again, is the most sensitive. Compared to the probe-level HMMs, the number of predicted transfrags for each model has decreased significantly. A number of different model parameters were tried in an attempt to match these results to the probe-level HMM results but with no success. It remains unclear why this change in behaviour is observed. One possible solution to the problem is to use Baum-Welch estimation to determine optimal parameters but this was not explored. The mean length of predicted transfrags, 1259bp, has increased significantly as a result of this change. Adjacent transcribed probes are more likely to be predicted as one transfrag rather than multiple which results in the observed increase in sensitivity and decrease in specificity. The breakdown of the predictions is similar: 42% in exons, 28% in introns, 18% in UTR and 13% in intergenic sequence, with 17% of the region being predicted as transcribed at some time point.

Nucleotide level transcript mapping							
	GC-RMA		PM-MM		PM-MM sliding		
	SMC	MISMC	SMC	MISMC	SMC	MISMC	transfrags
Sn (expressed)	73	81	80	86	82	86	82
Sn (all genes)	41	49	47	49	48	51	44
Sn (ncRNA)	3	0	12	12	52	52	16
Sp	48	34	44	36	40	40	44
# elements	283	222	282	254	299	387	6683

Table 4.2: Performance of nucleotide-level HMM transcript mapping methods compared to the union of transfrags predicted in Manak et al. [2006] across all time points. Sensitivity and specificity are measured at the nucleotide level.

Chapter 5

Gene prediction with tiling array data

This chapter develops the program TileGene, which predicts both protein coding genes and non-protein coding gene fragments by making use of the transcript mapping correlation methods described in Chapter 4 and the gene prediction methods described in Chapter 3.

5.1 Considering multiple observations

In order to incorporate additional forms of evidence into a (G)HMM gene predictor so called informant techniques have been developed. An overview of some of these methods can be found in Majoros [2007]. Recall from Chapter 3 that our goal is to find the optimal parse, $\mathbf{x}^*$, given by:

$$\begin{aligned}
\mathbf{x}^* &= \operatorname*{argmax}_{\mathbf{x}} \Pr(\mathbf{x}, \mathbf{y}|\lambda) \\
&= \operatorname*{argmax}_{\mathbf{x}} \pi_{x_1} d_{x_1}(l_1) b_{x_1}(\mathbf{y}_1) \prod_{t=2}^{T} a_{x_{t-1}x_t} d_{x_t}(l_t) b_{x_t}(\mathbf{y}_t)
\end{aligned}$$

The theory of HMMs can be extended to emit multiple observations. Here we extend our HMM to emit two observation sequences – DNA sequence and (processed) tiling array data. As in Chapter 4, call the tiling array observation sequence $\mathbf{z}$, having an emission distribution given by $c_x(z) = \Pr(Z = z|X = x)$. In this case the sequence $\mathbf{z}$ corresponds to the nucleotide-level tiling array emissions described in Section 4.5. The goal then becomes to find:

$$\begin{aligned}
\mathbf{x}^* &= \underset{\mathbf{x}}{\operatorname{argmax}} \Pr(\mathbf{x}|\mathbf{y},\mathbf{z},\lambda) \\
&= \underset{\mathbf{x}}{\operatorname{argmax}} \frac{\Pr(\mathbf{x},\mathbf{y},\mathbf{z}|\lambda)}{\Pr(\mathbf{y},\mathbf{z}|\lambda)} \\
&= \underset{\mathbf{x}}{\operatorname{argmax}} \Pr(\mathbf{x},\mathbf{y},\mathbf{z}|\lambda) \\
&= \underset{\mathbf{x}}{\operatorname{argmax}} \pi_{x_1} b_{x_1}(\mathbf{y}_1) c_{x_1}(\mathbf{z}_1) d_{x_1}(l_1) \prod_{t=2}^{T} a_{x_{t-1}x_t} b_{x_t}(\mathbf{y}_t) c_{x_t}(\mathbf{z}_t) d_{x_t}(l_t)
\end{aligned}$$

In terms of Viterbi decoding the addition of another emission probability is trivial to implement requiring only the following modifications to the algorithm presented in Chapter 3. Here we omit the structure ψ for clarity.

---

1. Intialisation:

   $$\delta_1(i) = \pi_i b_i(y_1) c_i(z_1) \gamma_i$$

   for $i : S_i \in \mathcal{N}$.

2. Recursion:

   $$\begin{aligned}
   \delta_{l+1}(k) = \max\{&\delta_l(k)\gamma_i b_i(y_{l+1}) c_i(z_{l+1}), \\
   &\max_{(\mu,i,j)\in\Theta_{k,l}} \delta_{\mu-1}(i)[1-\gamma_i] a_{ij} d_j(l-\mu+1) b_j(y_\mu^l) c_j(z_\mu^j) a_{jk}\gamma_k b_k(y_{l+1}) c_k(z_{l+1})\}
   \end{aligned}$$

   for $k : S_k \in \mathcal{N}, 1 < l < L$.

3. Termination:

   $$\delta_{L+1}(k) = \max\{\delta_L(k), \max_{(\mu,i,j)\in\Theta_{k,L}} \delta_{\mu-1}(i)[1-\gamma_i] a_{ij} d_j(l-\mu+1) b_j(y_\mu^l) c_j(z_\mu^l) a_{jk}\}$$

   for $k : S_k \in \mathcal{N}$.

This is the approach adopted by gene finders Twinscan (Korf et al. [2001]) and HMMGene (Krogh [2000]). Other decompositions of the objective function (see for example GenomeScan (Yeh et al. [2001])) are possible but this is perhaps the simplest. Following our terminology we denote the model for evaluating $c_j(z)$ an expression sensor.

It is worth drawing a contrast here between the HMM and the conditional random field (CRF) which is found in similar applications (see Lafferty et al. [2001]). Conditional random fields are used in image processing and parsing of natural language and more recently have appeared in gene prediction. As for the HMM, a CRF can be defined as a graphical model in which vertices represent random variables and (lack of) edges represent conditional (in)dependence assumptions (as in Section 2.8). CRFs are an undirected graphical model.

Formally, let $G = (V, E)$ be a graph such that $X = \{X_t\}_{t\in V}$ so that X is indexed by the vertices of G. The random variable X is a sequence of labels and let Y be a set of observable random variables. Then (X, Y) is a conditional random field if, when conditioned on Y the random variables, X, obey the Markov property with respect to the graph. In other words $\Pr(X_v|Y, X_w, w \neq v) = \Pr(X_v|Y, X_w, w \sim v)$. Where $w \sim v$ means that w and v are neighbours in G. Here we see that since the model is defined by conditioning on the observation variable, Y, that it is *not* a generative model. That is, Y is considered as given and so is not modeled explicitly. Hammersley and Clifford [1971] show that the joint distribution of a parse, $\mathbf{x}$, given an observation sequence, $\mathbf{y}$ is of the form:

$$\Pr(\mathbf{x}|\mathbf{y}, \lambda) = \frac{1}{Z(\mathbf{y})} \exp\left[\sum_{c \in \mathcal{C}} \Phi_c(\mathbf{x}_c, \mathbf{y})\right]$$

in which $\mathcal{C}$ is the set of all *u-cliques* of the graph, Φ_c is known as the potential function and Z is the partition function:

$$Z = \sum_{\mathbf{x}} \exp\left[\sum_{c \in \mathcal{C}} \Phi_c(\mathbf{x}_{\mathbf{c}}, \mathbf{y})\right]$$

In the present context X is a linear chain as for the traditional HMM gene finder, meaning the u-cliques (unobserved cliques)[1] are the singleton labels or pairs of adjacent labels. The potential functions are typically decomposed into the following form:

$$\Phi_c(\mathbf{x}_c, \mathbf{y}) = \sum_{i \in \mathcal{F}} \eta_i f_i(\mathbf{x}_c, \mathbf{y})$$

The interpretation of the potential functions and u-cliques now becomes more clear. f_i are known as feature sensors and can in theory be arbitrary real valued functions. The potential functions defined over singleton cliques can be interpreted as emission probabilities and the potential functions defined over the adjacent label pairs can be interpreted as transition probabilities. An important difference is that the transition and emission probabilities are conditional on the entire observation sequence allowing for long-range dependencies to be modeled if desired. Feature sensors can be sequence content sensors and signal sensors as for a GHMM gene predictor. The benefit of using a CRF is that the feature sensors do not need to be content or signal sensors but can be chosen to model an arbitrary number of other features – they need not even be probabilities.[2] The set $\{\eta_i\}$ weights the contribution from each feature sensor and is chosen in a discriminative fashion. That is, the weights are chosen to maximise the classification performance against some training set of observations. Though the formulation may be different, CRF gene predictors are similar in implementation to HMM gene predictors. The key difference both in training and in decoding being the use of weights $\{\eta_i\}$. See gene predictor CONTRAST (Gross et al. [2007]) for a more detailed description.

CRFs are described briefly because the above detailed dual emission GHMM was found to perform better when an *ad hoc* weight was added to the emission distribution $c_j(\mathbf{z})$, trained

[1]From graph theory. A clique is any set of vertices such that every vertex shares an edge with every other vertex.

[2]In natural language, feature sensors may be Boolean functions which, for example, are true if the word Y_i is upper case and the label X_i is 'proper noun'.

discriminatively. That is, we introduce η:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}} \, \pi_{x_1} b_{x_1}(\mathbf{y}_1) c^{\eta}_{x_1}(\mathbf{z}_1) d_{x_1}(l_1) \prod_{t=2}^{T} a_{x_{t-1}x_t} b_{x_t}(\mathbf{y}_t) c^{\eta}_{x_t}(\mathbf{z}_t) d_{x_t}(l_t)$$

Such *ad hoc* weights have been used by a number of (G)HMM *ab initio* gene predictors incorporating other evidence sources in later versions. Including weights for all terms in the above expression converts the problem to decoding a CRF with the equivalent feature sensors and a similar (though undirected) graph structure. Though there are other issues, it is possible that conditional random fields are better suited to the task of gene prediction than HMMs.

5.2 Model architecture

In addition to incorporating tiling array data, the state architecture of the gene predictor (see Figure 3.5) needs to be modified. By expanding the state space of the model, non-coding gene fragments can be predicted along side protein-coding genes, allowing TileGene to classify observed transcription as either coding or non-coding. This is accomplished by the addition of an intergenic 'transfrag' or non-coding gene fragment state, $5'$ and $3'$ UTR states and intra-intronic transfrag states. The transfrag states are a novel addition to the conventional gene prediction framework but are well justified. Without these states the gene predictor would be coerced (incorrectly) into predicting protein-coding genes wherever there was evidence of expression. The model is shown in Figure 5.1.

Some assumptions are required to keep the model relatively simple. Firstly, we assume that there exists no significant change in sequence content within the intergenic or the intra-intronic transfrag state or either the $5'$ or $3'$ UTR states. From Chapter 4, we assume that the length of the transfrag states is geometrically distributed with a mean of approximately 160bp. While neither of these are true they avoid over specifying the character of the non-coding genes we are attempting to model. It is hoped that the predictions of the model are insensitive to these assumptions. We assume there exists no alternative splicing that would contribute different tiling array intensities to different exons of a gene. We assume that every gene contains both $5'$ and $3'$ UTR, again geometrically distributed. We make no attempt to model signals delimiting the boundaries of either the transfrag states or the UTR states.

Another class of states, $\mathcal{E}$, is added to contain the states expressed but not protein-coding. Both $\mathcal{E}$ and $\mathcal{C}$ states use the same expression sensor, trained on the set of expressed genes described in Section 4.5. This assumes that features in $\mathcal{E}$ and $\mathcal{C}$ are equally expressed which is unlikely to be true. $\mathcal{N}$ state expression sensors are trained on the set of not transcribed features from Section 4.5. UTR geometric state length parameters were estimated from a set of genes downloaded from UCSC table browser, the mean length of the $3'$ UTR state was approximatey 550 bp and the mean length of the $5'$ state was approximately 200bp. Transfrag length parameters were chosen such that the mean length of each state is approxmately 150bp. The intergenic transfrag and UTR states use an intergenic content sensor. The intra-intron transfrag state uses an intron content sensor. Transition parameters were chosen such that the model is equally likely to transition into a gene or into a transfrag state. Parameters were

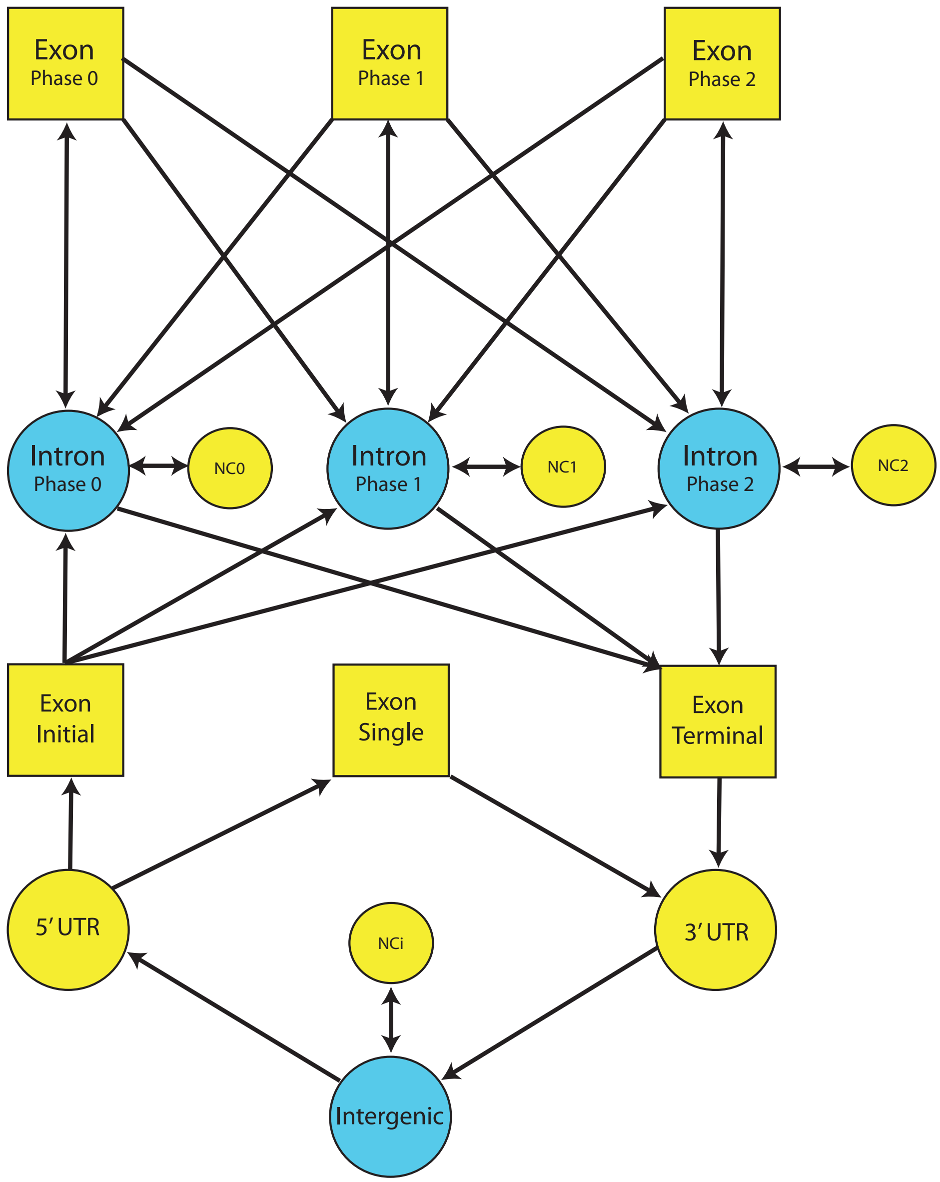


Figure 5.1: The state architecture of TileGene. Additional 'transfrag' states are added to model transcription from intergenic and intra-intronic transfrags. Both 5′ and 3′ UTR states are added to model transcription flanking gene start and stop sites. Forward strand states shown only. States in yellow are expressed, states in blue are not expressed. Squares indicate generalised, empirical length distribution states, cirlces indicate non-generalised geometric length distribution states. Additional states to Figure 3.5 are NCi = intergenic non-coding transfrag, 5′ UTR, 3′ UTR and NC0, NC1, NC2 = intra-intronic transfrag states within introns of phase 0, 1, 2 respectively.

chosen such that the probability of transitioning into a intra-intronic transfrag was one third that of transitioning to an exon. These estimates are a consequence of using a generative model and having insufficient training data. The effect these choices had on predictions or indeed how realistic they were was not investigated. Ideally Baum-Welch or some equivalent could be used to provide some assistance but this is somewhat more complicated for the current generalised hidden Markov model. TileGene uses 2nd order inhomogenous Markov chain signal sensors, 2nd order 3-period Markov chain content sensors in coding states and 2nd order homogenous Markov chain content sensors in non-coding states. The gene model uses a long AG signal sensor, as described in Section 3.7.3. As mentioned in Section 3.7.3, using a realistically sized training set is important if the model performance is to be extrapolated to any other genome or compared to any other gene predictor. Consequently, the TileGene transition parameters, state length distributions, signal and content sensors were trained on the Augustus training set of 400 genes – see Appendix B.

The alternating coding/non-coding state structure shown in Figure 3.5 no longer applies to the model of Figure 5.1 so the Genscan method of decoding has to be generalised slightly. Ideally a different method of decoding would be implemented which supported fully generalised models for both coding and non-coding states and hence did not require the alternating $\mathcal{C}/\mathcal{N}$ structure. In this case however, since a Genscan implementation is readily available it is easiest to simply modify it. At each position, the added non-coding to non-coding transitions need to be included when decoding. This requires the following modifications:

---

1. Intialisation:
$$\delta_1(i) = \pi_i b_i(y_1) c_i(z_1) \gamma_i$$
for $i : S_i \in \mathcal{N} \cup \mathcal{E}$.

2. Recursion:
$$\begin{aligned}\delta_{l+1}(k) = \max\{&\delta_l(k)\gamma_i b_i(y_{l+1}) c_i(z_{l+1}),\\ &\max_{(\mu,i,j)\in\Theta_{k,l}} \delta_{\mu-1}(i)[1-\gamma_i] a_{ij} d_j(l-\mu+1) b_j(y_\mu^l) c_j(z_\mu^j) a_{jk}\gamma_k b_k(y_{l+1}) c_k(z_{l+1}),\\ &\max_{i:S_i\in\mathcal{N}\cup\mathcal{E}, a_{ik}\neq 0} \delta_l(i)[1-\gamma_i] a_{ik}\gamma_k b_k(y_{l+1}) c_k(z_{l+1})\}\end{aligned}$$
for $k : S_k \in \mathcal{N} \cup \mathcal{E}, 1 < l < L$.

3. Termination:
$$\delta_{L+1}(k) = \max\{\delta_L(k), \max_{(\mu,i,j)\in\Theta_{k,L}} \delta_{\mu-1}(i)[1-\gamma_i] a_{ij} d_j(l-\mu+1) b_j(y_\mu^l) c_j(z_\mu^l) a_{jk}\}$$
for $k : S_k \in \mathcal{N} \cup \mathcal{E}$.

The key change being the added line in the recursion step to consider transitions between two non-coding states. This is not included in the termination since it is pointless to consider a state transition at an unobserved nucleotide.

5.3 Implementation

The above modifications were made to the *ab initio* gene predictor presented in Chapter 3. The resulting program we call TileGene. In addition to the sequence and configuration file the program has an additional input for tiling array data. Presently a number of preprocessing steps are required by the user although this can be easily automated. The file required to be input is of the format:

```
13506243 45.6
13506274 48.5
13506293 50.6
...
```

In which the first column is position along chromosome and the second is the tiling array intensity score, pre-normalised and summarised. This is similar to the wiggle format, with no header.[3] Following this, a per nucleotide intensity file is computed containing a symbol (either a score or ω) for every nucleotide. The file requires a header matching the fasta format for the corresponding sequence.

The addition of eleven extra states unfortunately adds significantly to the program run time (recall that Viterbi decoding is $O(N^2)$ in number of states N). On the 3MB sequence analysed the program now takes approximately three hours to run on a 2.33 GHz Intel Core 2 Duo machine with 2GB of memory, running OS X.

The output now includes non-coding states denoted by type `noncoding` in the output GFF file. The character of each state is inferred from its position in relation to the exons (if present) of the same gene. For example:

```
Adh3 GHMM noncoding 410985 411401 . + . Gene 1
Adh3 GHMM noncoding 412747 412767 . + . Gene 2
Adh3 GHMM noncoding 415576 416211 . -. Gene 3
Adh3 GHMM exon 416212 416749 . -. Gene 3
Adh3 GHMM noncoding 417100 417111 . -. Gene 3
Adh3 GHMM exon 417658 417739 . -. Gene 3
Adh3 GHMM noncoding 417740 418121 . -. Gene 3
```

Contains an two intergenic transfrags, a gene with both $5'$ and $3'$ UTR and an intra-intronic transfrag.

5.4 Results

We test performance on the *Adh* sequence and annotation described in Section 4.1. TileGene's performance is compared to other annotation methods in Tables 5.1. The sequence and annotation are different to that described in Section 3.7.1 so the *ab initio* gene prediction results for both GHMM models and Augustus differ slightly to those previously reported. The annotations include multiple transcripts for the one gene which complicates computing

[3] `http://genome.ucsc.edu/goldenPath/help/wiggle.html`

sensitivity. This is addressed as follows: at the nucleotide level any overlapping transcripts are compressed to a single transcript allowing sensitivity to be computed as normal; at the exon level multiple transcripts are not compressed into a single transcript since there is no guarantee the result will be an allowed gene, the additional transcripts will artificially decrease sensitivity and increase specificity; at the gene level sensitivity is computed as:

$$Sn = \frac{TP}{\#genes}$$

making the distinction between the number of genes and number of transcripts in the region. The full annotation contains 347 transcripts and 259 genes. The expressed annotation contains 84 transcripts and 57 genes.

TileGene (without weight η) identifies more expressed nucleotides within exons than either the union of transfrags across time points, identified by Manak, or the nucleotide-level, MISMC-based, transcript mapping HMM. This is a result of the content sensors employed by the gene prediction model, as evidenced by the equally high performance of both *ab initio* gene predictors. At the nucleotide level, TileGene correctly predicts approximately 50% of the annotated ncRNA genes in the region, which matches that of the transcript mapping HMM. This suggests that protein-coding genes and non-coding RNA gene fragments are being successfully classified as such by TileGene. This is encouraging though there is room for improvement. The predicted transcribed regions now breakdown as follows: 58% in exons, 18% in introns, 17% in UTR and 8% in intergenic sequence. The model makes less use of the transfrag states than might be expected. Indeed, Section 4.6 described approximately 13% of intergenic sequence as transcribed. The number of intergenic transfrags is only 43, with mean and median length of 376bp and 321bp respectively. A number of predicted intra-intron transfrags span a large proportion of the intron which is indicative of an unspliced RNA being present in some of the samples. While it is possible TileGene is combining a number of separate adjacent transcribed elements a visual inspection of the scores shows consistently high correlation in these introns. This was not reported in Manak. Measured against the full annotation, TileGene correctly predicts 30% of the UTR nucleotides as such. Inspection of a number of genes shows TileGene can successfully predict the beginning and end of the transcribed genes although since no signal sensors were implemented these predictions are approximate only. On a number of occasions the observed correlation scores, and hence the predicted UTR, differ noticably from the annotated UTR which may be indicative of alternative transcription start sites. Such observations need further study.

As a gene predictor, when measured against the expressed annotation, TileGene shows increased performance in relation to the *ab initio* GHMM predictor it is based on. The decreased sensitivity, in relation to the full annotation, is expected since TileGene (in general) is less likely to predict exons that are not expressed. However, a visual inspection of the predictions shows that on a number of occasions TileGene does predicts exons where there is little apparent indication of transcription. This occurs when there is particularly strong evidence (codon bias, signals, etc) of a gene. Compared to the *ab initio* predictor, TileGene shows most improvement at the gene level. The increase in specificity is partly because fewer genes are being predicted – 161 compared to 222. The increase in sensitivity is also significant but it is less clear where, exactly, the improvement comes from. At both the exon and the nucleotide level, TileGene shows only marginal improvement over the *ab initio* sensitivities and specificities. TileGene has a significantly lower percentage of missed exons, which indicates more expressed exons are identified. The transcription data provides approximate information about the location of genes (hence the increased specificity at the gene level), but may not be of high enough resolution to improve the gene prediction results at the exon level.

The weight described in Section 5.1 shows (limited) performance increase. The 'optimal' weight was determined using a simple grid search method. Ideally, a more efficient optimisation routine would have been implemented but there was insufficient time. The weight used was $\eta = 0.8$. The experiments (data not shown) indicate that as the weight is further decreased to zero the performance converges to the *ab initio* GHMM performance results. The weighted TileGene model shows largest performance increase on statistics measured against the full annotation (Sn (all genes) and WE%). As discussed in Section 1.3, the genome is a complicated network of transcribed elements and what constitutes such an element is not clearly defined, since they can overlap on the same or reverse strand. This complication is difficult to force into a GHMM gene model. Given this difficulty is a trade-off between identifying and classifying transcription as coding or non-coding and providing an accurate gene prediction when this transcription is classified as protein-coding. The discriminatively trained weight, in some sense, aids in this compromise. Though a comparison of TileGene's performance to Augustus's show there is still much room for improvement, these results are encouraging. TileGene is build on a relatively simple gene model, whereas Augustus is a state-of-the-art *ab initio* predictor. Figure 5.2 shows an example of TileGene successfully predicting the gene structure, including UTR of a gene in the *Adh* region of drosophila.

In Section 3.7.3 we saw that a larger training set improved *ab initio* performance significantly. It is worth seeing if similar behaviour is observed for TileGene, or whether the additional expression data the model uses is sufficiently guiding TileGene to more correct predictions, without the need for a larger training set. To this end, we investigate TileGene's performance using our two content and signal sensor training sets: the 400 gene Augustus training set and the 977 gene UCSC dataset – see Appendix B. The rest of TileGene's parameters and models remain the same. The results are detailed in Table 5.2. As found previously, the larger training set improves the performance of *ab initio* gene predictors, particularly at the gene level. However, when TileGene is trained on a larger training set the results, particularly at the gene level, decrease significantly. This suggests that, indeed, the tiling array data is, in some sense, acting as a surrogate for a larger training set, making the need for one redundant. However, it is a strange result that the inclusion of *more* training data can actually decrease performance – especially when it *increased* performance for the *ab initio* model. Any machine learning method needs to select training sets of appropriate size and must be cautious of over-training a model, or training some components of the model in an unbalanced manner, as may be the case here. Compared to the *ab initio* predictor TileGene's gene model is more complicated and contains a number of poorly trained parameters. As such, TileGene may be more sensitive to such imbalances in training data – the gene model being trained very well and the transfrag model being trained very poorly.

		Prediction method					
		Manak	Transfrag	TileGene1	TileGene1 + w	GHMM1	Augustus
Nucleotide	Sn (expressed)	82	86	92	92	94	92
	Sn (all genes)	44	51	61	64	74	73
	Sn (ncRNA)	16	52	51	51	-	-
	Sp	44	40	94	94	93	97
Exon	Sn (expressed)	-	-	61	62	61	62
	Sn (all genes)	-	-	42	43	50	54
	Sp	-	-	62	63	59	78
	ME%	-	-	31	31	42	41
	WE%	-	-	19	17	21	6
Gene	Sn (expressed)	-	-	43	43	37	60
	Sn (all genes)	-	-	27	28	29	47
	Sp	-	-	44	44	34	61

Table 5.1: Performance of various method in characterising the *Adh* region of drosophila. Sensivitiy is measured against a range of annotations, as indicated. Specificity for TileGene includes protein-coding predictions only. Specificity is measured against the full annotation. Percentage of missed exons is computed against the expressed annotation, percentage of wrong exons is computed against the full annotation. Augustus annotation downloaded from UCSC table browser, drosophila, assembly April 2004. The Manak column represents the union of transfrags across all time points as identified by Manak et al. [2006]. The Transfrag column represents all predicted transfrags using the nucleotide-level HMM based on the MISMC statistic on the sliding window, PM-MM corrected dataset as in Section 4.6. '-' indicates the statistic is not applicable.

		Prediction method					
		GHMM1	TileGene1	TileGene1+w	GHMM2	TileGene2	TileGene2+w
Nucleotide	Sn (expressed)	94	92	92	94	92	93
	Sn (all genes)	74	61	64	77	62	65
	Sn (ncRNA)	-	51	51	-	51	45
	Sp	93	94	94	93	93	94
Exon	Sn (expressed)	61	61	62	60	59	60
	Sn (all genes)	50	42	43	51	40	41
	Sp	59	62	63	65	61	63
	ME%	42	31	31	36	34	32
	WE%	21	19	17	15	18	17
Gene	Sn (expressed)	37	43	43	40	37	35
	Sn (all genes)	29	27	28	35	25	26
	Sp	34	44	44	40	39	40

Table 5.2: Performance of various implementations of TileGene compared to *ab initio* gene prediction results. GHMM1, TileGene1 and TileGene1+w content and signal sensors are trained on the Augustus training set of 400 genes. GHMM2, TileGene2 and TileGene2+w content and signal sensors are trained on the secondary training set obtained from the UCSC genome browser. TileGene1+w and TileGene2+w use a weighted expression sensor with $\eta = 0.8$. '-' indicates the statistic is not applicable.

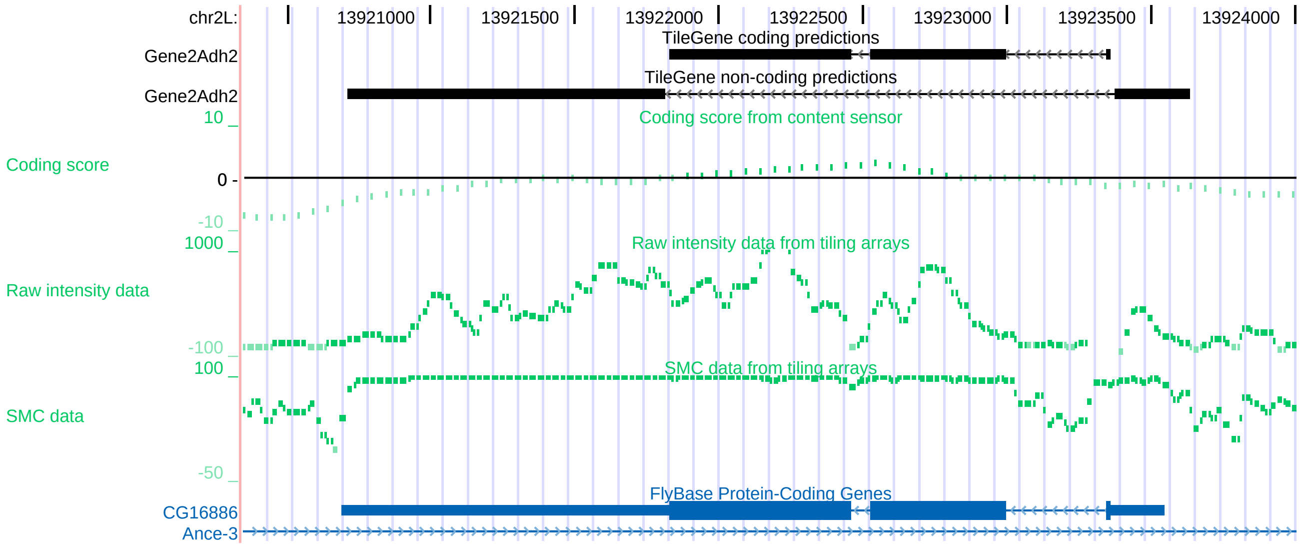


Figure 5.2: Example of a TileGene gene prediction. TileGene prediction (top) includes coding (exon) and non-coding predictions. In this case the non-coding predictions correspond to UTR. Shown alongside a sliding window exon content score (as in Section 3.2), the raw tiling array intensities and the SMC statistic. The 'raw' intensity track is the sum of PM-MM intensities across all 12 time points. The SMC statistic is based on the sliding window, PM-MM corrected dataset. FlyBase annotations taken from UCSC genome browser, drosophila, April 2004 assembly.

Chapter 6

Discussion

6.1 Performance

This thesis explored two common computational methods for analysing and identifying functional elements of a eukaryotic genome. Firstly, the field of computational gene prediction was described. An *ab initio* gene predictor was developed both as an illustration of the concepts and to serve as a basis for the program TileGene later designed. The model implemented was based primarily on Genscan, but some inspiration was also taken from Augustus. A number of modeling decisions were experimented with including the length distribution models, the content sensor orders and the signal sensor orders and length. In contrast to other gene predictors, fifth order Markov chain content sensors were found to be the worst performing. Second order Markov chains (based only on single codon biases opposed to hexamer biases) were found to be optimal. Maximum likelihood parameter estimates based on an annotated training set were used. The gene predictor was tested on the fruit fly, *Drosophila melanogaster*, in particular a well studied region of chromosome 2L, the *Adh* region. Performance was found to be comparable to older gene predictors, such as the original GHMM gene predictor, Genie and HMMGene, but was not able to match the more sophisticated Augustus.

Secondly, the application of tiling arrays in transcript mapping was described. Two statistics for identifying transcribed elements using evidence from multiple tiling arrays were implemented, the SMC statistic and the MISMC statistic. Both statistics attempt to identify correlations between neighbouring probes as evidence of differential expression. The statistics were computed on the same dataset, normalised in three different ways. The *Adh* region of drosophila was again used as a test set. A simple two-state HMM was implemented to determine the statistics' ability to perform accurate transcript mapping. Parameters were again estimated from an annotated training dataset. The MISMC statistic consistently performed better than the SMC statistic, presumably because it incorporates information about the absolute value of a probe's intensity and not just its relation to other neighbouring probes. It was observed that GC-RMA background correction resulted in fewest expressed probes being correctly identified as such whilst a sliding window average and PM-MM correction resulted in the most expressed probes being correctly identified. Neither of the correlation methods identified as many expressed probes as one other method provided for comparison. However, the MISMC methods identifies more protein-coding nucleotides in total and more ncRNA genes in the region than the other method, indicating it may be a more sensitive method for finding small or lowly expressed transcripts. The HMM was modified to emit symbols per nucleotide instead of per probe and was shown to increase sensitivity and decrease specificity. Though it remains unclear exactly why this occurs.

Finally, this thesis explored in what way these two methods can be combined in order to provide a more accurate annotation of a genome. The program TileGene was presented as a proposed model. The state architecture was expanded to include models for UTR and unidentified intergenic or intra-intronic transcription. Though further study is needed, by including tiling array data into a gene prediction framework, TileGene is able to distinguish coding and non-coding transcription and make a (relatively) accurate gene prediction when appropriate. The model is able to identify UTR but it less willing to predict intergenic transcribed elements, in comparison to the two-state HMMs implemented. The model, on occasion, predicted transcribed elements spanning introns and alternative transcription start sites or transcription termination sites. These observations demonstrate the potential application of TileGene, however need investigation. The inclusion of tiling data into a gene predictor showed significant predictive improvements compared to the *ab initio* predictor when measured at the gene level. When trained on a larger training set, however, TileGene's performance decreased. This is likely a result of the model being over-trained, or in someway unbalanced, and also warrants further study.

6.2 Future work

There are a number of possible extensions to the work presented in this thesis. Firstly, the *ab initio* gene predictor could be improved. This would involve experimenting with the signal sensor models and content sensor models as in Section 3.7.3. Variable order Markov chains could be implemented instead of fixed order models. The state architecture could be expanded to model more features of eukaryotic genes, such as the polyadenylation signal and TATA box. Though these are worthwhile improvements, such attempts are a separate project in themselves and are only worth taking so far in this thesis.

A number of issues with the statistics and analysis of Chapter 4 were identified which could be better addressed. To investigate whether inferior performance of the GC-RMA corrected dataset was due to the suggested mechanism, the background correction could be retried, using another implementation that allowed parameters to be modified. The prediction of fewer, longer transfrags of the nucleotide-level HMM compared to the probe-level HMM could be investigated and perhaps a more suitable incorporation method could be used. Other training methods such as Baum-Welch estimation could be used, instead of guessing 'reasonable' parameters.

A number of modifications to TileGene could be attempted to improve performance. Forcing each gene to have both $5'$ and $3'$ UTR is perhaps a bit restrictive so this could be relaxed. Signal sensors for the UTR states could be implemented in order to more precisely predict the beginning and end of transcription. Assumptions regarding the length and frequencies of non-coding genes could be modified to be more realistic. Indeed, one of the problems encountered in designing TileGene was the choice of some model parameters. Lacking proper training data in such cases makes parameter estimation difficult and is one of the drawbacks of implementing a generative model such as an HMM. Performance was shown to increase when a weight was included and so the next logical step is to reimplement the model as a conditional random field. The main task would be to design and implement a discriminative training method.

Finally, it is worth mentioning the so-called next-gen sequencing technology now available which is used for RNA-seq. Such ultra-high-throughput sequencers allow for a comprehensive sequencing of the transcriptome. Mapping the reads back to the genome gives, just as for tiling arrays, a picture of which elements on the genome are transcribed. RNA-seq and other methods related to these technologies may supersede microarray experiments for some applications. It is worth seeing whether such sequencing data could be incorporated into TileGene. This could be achieved in a similar fashion to the tiling array data, although the MISMC statistic would need to be replaced with an alternative, or could be modeled as a pair HMM which emits DNA sequence and aligned transcript sequence (as in NSCAN, Gross and Brent [2006]).

6.3 Concluding remarks

The advent of tiling microarray technology and recent evidence suggesting non-coding RNAs play a significant role in development and regulation require a novel set of bioinformatics methods. This project has (to the best of the author's knowledge) been the first attempt at expanding the conventional gene prediction framework to predict more than just protein-coding genes. It is also one of the first to incorporate tiling array data to improve gene predictions.[1]

Successful annotation of a genome is of key importance to the understanding a sequenced organism and computational approaches for the task are continually improving. TileGene, with some additional work, may prove to be such an improvement. With some success, we have, at present, presented a model that is able to characterise transcription as coding or non-coding. Further, the model can predict the structure of protein-coding genes, including UTR, when appropriate.

[1] See mGene, `http://www.fml.tuebingen.mpg.de/raetsch/projects/mgene` and the presentation `www.fml.tuebingen.mpg.de/raetsch/lectures/RaetschGenomeInformatics08.pdf`

Appendix A

Removing in-phase stop codons

Here we present an algorithm to generate a structure Θ, where $\Theta_{k,l}$ is a list of tuples (μ, i, j). μ is the starting position of the coding emission in state S_j from state S_i, ending at position l and transitioning into state S_k. Each emission contains no in-phase stop codons. Briefly in words: for each position in the sequence (`front_pos`) the algorithm finds all subsequent signals (`back_pos`, `back_signal`) that need to be considered (all those before `maximumstop`) and finds any coding states that emit the pair of signals (`front_signal`, `back_signal`). Then the `statetriples` structure is created to contain all potential transitions that can occur at positions (`front_pos`, `back_pos`). For each of these triples (a preceding non-coding state, coding state, and succeeding non-coding state) if the change in phase of the coding state matches the change in phase specified by the flanking non-coding states and if the coding emission contains no in-phase stop codons (`NOT eclipsed`) then it is added to the structure `theta[poststate, back_pos]`.

In slightly more detailed pseudo-code:

```
:frontsignals <- generateSignalList(sequence)
:backsignals <- frontsignals
:stopsignals <- onlyStopCodonSignals(frontsignals)
:FOR (front_pos, front_signal) IN frontsignals DO
:  maximumstop <- computeFurthestStopCodonInAnyFrame(front_pos, stopsignals)
:  backsignals <- removeBackSignalsBehindFrontPos(backsignals, front_pos)
:  FOR (back_pos, back_signal) IN backsignals DO
:    frontandbackpairs <- EMPTYLIST
:    IF back_pos <= maximumstop + stopcodonlength THEN
:      IF front_pos + frontsignallength <= backpos THEN
:        frontandbackpairs APPEND (front_pos, front_signal, back_pos, back_signal)
:    ELSE
:      break
:    statetriples <- validStateTransitionsForPairedSignals(frontandbackpairs)
:    FOR (prevstate, state, poststate, front_pos, back_pos) IN statetriples DO
```

```
:       codingstart, codingend <- computeSignalOffsets(state, front_pos, back_pos)
:       changeinphase <- poststatephase - prevstatephase mod 3
:       frame <- codingstart - prevstatephase mod 3
:       eclipsed <- isBackSignalBehindStopCodonInFrame(stopsignals, frame, coding_end)
:       IF changeinphase == codingstart - codingend mod 3 AND NOT eclipsed THEN
:         theta[poststate, back_pos] APPEND (front_pos, prevstate, state)
```

For clarity this code only finds valid emissions on one strand. Adding emissions on the reverse strand requires some choices in how to deal with coordinates on different strands, as well as when and if to swap `prevstate` and `poststate` on the opposite strand. These additions are not described. Note the distinction between frame and phase, not made in the main text, here frame is defined as phase of a nucleotide, relative to the start of the sequence, that is $frame = position \mod 3$. Since it is not clear what phase a stop codon will be in beforehand it makes more sense to store its position and its frame in the list `stopsignals`.

Appendix B

Training datasets

The following 400 genes (listed here as FlyBase gene IDs) constitute the Augustus gene prediction training set, designed to be disjoint with the *Adh* test region. Available from `http://augustus.gobics.de/datasets/`. See Section 3.7.3 for more detail.

FBgn0000022, FBgn0000024, FBgn0000052, FBgn0000071, FBgn0000084, FBgn0000120, FBgn0000137, FBgn0000146, FBgn0000180, FBgn0000216, FBgn0000256, FBgn0000308, FBgn0000352, FBgn0000376, FBgn0000377, FBgn0000413, FBgn0000442, FBgn0000447, FBgn0000448, FBgn0000463, FBgn0000486, FBgn0000529, FBgn0000556, FBgn0000579, FBgn0000588, FBgn0000658, FBgn0000826, FBgn0000928, FBgn0001075, FBgn0001077, FBgn0001104, FBgn0001114, FBgn0001142, FBgn0001281, FBgn0001297, FBgn0001987, FBgn0001994, FBgn0002023, FBgn0002183, FBgn0002306, FBgn0002525, FBgn0002535, FBgn0002563, FBgn0002571, FBgn0002607, FBgn0002643, FBgn0002715, FBgn0002716, FBgn0002789, FBgn0002849, FBgn0002922, FBgn0002932, FBgn0002970, FBgn0003022, FBgn0003044, FBgn0003046, FBgn0003053, FBgn0003060, FBgn0003075, FBgn0003087, FBgn0003114, FBgn0003117, FBgn0003130, FBgn0003140, FBgn0003141, FBgn0003150, FBgn0003205, FBgn0003231, FBgn0003268, FBgn0003274, FBgn0003278, FBgn0003319, FBgn0003353, FBgn0003386, FBgn0003391, FBgn0003435, FBgn0003448, FBgn0003462, FBgn0003470, FBgn0003483, FBgn0003513, FBgn0003517, FBgn0003527, FBgn0003557, FBgn0003575, FBgn0003607, FBgn0003701, FBgn0003733, FBgn0003751, FBgn0003861, FBgn0003941, FBgn0003961, FBgn0003963, FBgn0003964, FBgn0003979, FBgn0003996, FBgn0004053, FBgn0004054, FBgn0004108, FBgn0004168, FBgn0004170, FBgn0004177, FBgn0004240, FBgn0004360, FBgn0004363, FBgn0004364, FBgn0004378, FBgn0004397, FBgn0004513, FBgn0004556, FBgn0004567, FBgn0004577, FBgn0004584, FBgn0004606, FBgn0004618, FBgn0004619, FBgn0004636, FBgn0004638, FBgn0004647, FBgn0004654, FBgn0004784, FBgn0004832, FBgn0004838, FBgn0004861, FBgn0004892, FBgn0004915, FBgn0004957, FBgn0005411, FBgn0005614, FBgn0005631, FBgn0005642, FBgn0005648, FBgn0005649, FBgn0005654, FBgn0005664, FBgn0005671, FBgn0005672, FBgn0010194, FBgn0010220, FBgn0010246, FBgn0010265, FBgn0010296, FBgn0010316, FBgn0010333, FBgn0010342, FBgn0010385, FBgn0010410, FBgn0010426, FBgn0010516, FBgn0010851, FBgn0011230, FBgn0011260, FBgn0011273, FBgn0011277, FBgn0011294, FBgn0011570, FBgn0011584, FBgn0011598, FBgn0011643, FBgn0011648, FBgn0011659, FBgn0011701, FBgn0011722, FBgn0011746, FBgn0011763, FBgn0011823, FBgn0011824, FBgn0013272, FBgn0013300, FBgn0013325, FBgn0013469, FBgn0013746, FBgn0013753, FBgn0013756, FBgn0013763, FBgn0013764, FBgn0013765, FBgn0013972, FBgn0014009, FBgn0014019, FBgn0014020, FBgn0014028, FBgn0014033, FBgn0014076, FBgn0014127, FBgn0014151, FBgn0014179, FBgn0014861, FBgn0014870, FBgn0015247, FBgn0015268, FBgn0015277, FBgn0015288, FBgn0015298, FBgn0015320, FBgn0015323, FBgn0015553, FBgn0015561, FBgn0015582, FBgn0015584, FBgn0015589, FBgn0015621, FBgn0015623, FBgn0015737, FBgn0015754, FBgn0015763, FBgn0015797, FBgn0015803, FBgn0015816, FBgn0015838, FBgn0015844, FBgn0016075, FBgn0016078, FBgn0016122, FBgn0016675, FBgn0016697, FBgn0016726, FBgn0016762, FBgn0016978, FBgn0017414, FBgn0019624, FBgn0019828, FBgn0019925, FBgn0019932, FBgn0019948, FBgn0019949, FBgn0019968, FBgn0020245, FBgn0020258, FBgn0020370, FBgn0020443, FBgn0020493, FBgn0020506, FBgn0020510, FBgn0020545, FBgn0020611, FBgn0020616, FBgn0020641, FBgn0020911, FBgn0020912, FBgn0021874, FBgn0022073, FBgn0022699, FBgn0022740, FBgn0022772, FBgn0022960, FBgn0023083, FBgn0023130, FBgn0023143, FBgn0023170, FBgn0023171, FBgn0023179, FBgn0023180, FBgn0023181, FBgn0023216, FBgn0023415, FBgn0023495, FBgn0024230, FBgn0024234, FBgn0024285, FBgn0024291, FBgn0024432, FBgn0024841, FBgn0024923, FBgn0024947, FBgn0024958, FBgn0025109, FBgn0025117, FBgn0025373, FBgn0025458, FBgn0025469, FBgn0025573, FBgn0025581, FBgn0025582, FBgn0025595, FBgn0025608, FBgn0025623, FBgn0025637, FBgn0025781, FBgn0025802, FBgn0025820, FBgn0025878, FBgn0025885, FBgn0026136, FBgn0026141, FBgn0026150, FBgn0026324, FBgn0026417, FBgn0026438, FBgn0026439, FBgn0026619, FBgn0026721, FBgn0026753, FBgn0026761, FBgn0027104, FBgn0027841, FBgn0027885, FBgn0028381, FBgn0028412, FBgn0028675, FBgn0028687, FBgn0028689, FBgn0028707, FBgn0028708, FBgn0033189, FBgn0043020, FBgn0000014, FBgn0000046, FBgn0000075, FBgn0000108, FBgn0000147, FBgn0000228, FBgn0000251, FBgn0000273, FBgn0000316, FBgn0000409, FBgn0000439, FBgn0000482, FBgn0000986, FBgn0001202, FBgn0001254, FBgn0001258, FBgn0001291, FBgn0001308, FBgn0002031, FBgn0002440, FBgn0002527, FBgn0002531, FBgn0002561, FBgn0002736, FBgn0002856, FBgn0002936, FBgn0002941, FBgn0003091, FBgn0003116, FBgn0003255, FBgn0003410, FBgn0003460, FBgn0003463, FBgn0003480, FBgn0003863, FBgn0003884, FBgn0004034, FBgn0004395, FBgn0004428, FBgn0004551, FBgn0004580, FBgn0004583, FBgn0004597, FBgn0004867, FBgn0004873, FBgn0004908, FBgn0005670, FBgn0010213, FBgn0010258, FBgn0010288, FBgn0010323, FBgn0010355, FBgn0010356, FBgn0010407, FBgn0010415, FBgn0010423, FBgn0010441, FBgn0010453, FBgn0010501, FBgn0010770, FBgn0011282, FBgn0011674, FBgn0011692, FBgn0011703, FBgn0013726, FBgn0013762, FBgn0014006, FBgn0014462, FBgn0014863, FBgn0015011, FBgn0015380, FBgn0015381, FBgn0015513, FBgn0015765, FBgn0015776, FBgn0015929, FBgn0016054, FBgn0016119, FBgn0017551, FBgn0020385, FBgn0020764, FBgn0020765, FBgn0022710, FBgn0023549, FBgn0024352, FBgn0025186, FBgn0025352, FBgn0026378, FBgn0026760, FBgn0027054, FBgn0027095, FBgn0027594, FBgn0041186, FBgn0027053.

The following 977 genes were selected randomly from chromosome 2R of drosophila and constitute the second, larger gene prediction training set. They were downloaded from the drosophila April 2006 assembly using the UCSC table browser. See Section 3.7.3 for more detail.

CG40130-RA, CG18001-RA, CG40293-RA, CG17486-RA, CG17706-RA, CG17528-RA, CG41441-RA, CG2944-RA, CG2682-RA, CG10465-RA, CG10395-RA, CG10396-RA, CG32838-RA, CG17510-RA, CG11665-RA, CG1344-RA, CG8376-RA, CG34200-RA, CG7897-RA, CG14468-RA, CG14470-RA, CG7882-RB, CG7881-RA, CG12051-RA, CG7865-RB, CG7861-RA, CG34211-RA, CG7856-RA, CG7845-RA, CG7849-RA, CG1765-RA, CG8345-RA, CG11211-RA, CG15234-RA, CG3161-RA, CG3152-RA, CG3274-RA, CG3174-RA, CG33134-RA, CG30443-RA, CG15908-RA, CG9410-RA, CG3271-RA, CG17266-RA, CG30445-RA, CG15909-RA, CG3403-RA, CG3269-RA, CG3409-RA, CG9432-RB, CG3265-RA, CG9436-RA, CG3283-RA, CG15237-RA, CG34215-

RA, CG9453-RI, CG30158-RA, CG33351-RA, CG33348-RA, CG3572-RB, CG17002-RB, CG30159-RA, CG18816-RA, CG10106-RA, CG12142-RA, CG12844-RB, CG2374-RA, CG12839-RA, CG4471-RA, CG12832-RA, CG12831-RA, CG12833-RA, CG4486-RA, CG12172-RA, CG11060-RA, CG1851-RA, CG11101-RA, CG12164-RA, CG17800-RBA, CG1708-RA, CG1712-RA, CG1707-RA, CG11113-RB, CG11123-RA, CG30502-RA, CG11125-RA, CG11141-RA, CG11139-RA, CG1621-RA, CG1620-RA, CG1602-RA, CG1600-RA, CG30499-RA, CG1363-RA, CG2092-RA, CG30494-RA, CG30491-RA, CG2064-RA, CG12042-RA, CG12107-RA, CG30493-RB, CG34216-RA, CG1942-RA, CG30497-RA, CG1555-RA, CG11217-RA, CG1550-RA, CG1882-RA, CG1548-RA, CG18853-RA, CG30377-RA, CG1877-RA, CG8791-RA, CG30380-RA, CG14764-RA, CG34431-RA, CG11165-RA, CG2915-RA, CG2910-RA, CG8726-RA, CG18455-RA, CG12769-RA, CG8715-RA, CG12770-RA, CG8714-RA, CG17975-RA, CG11210-RA, CG8711-RA, CG2158-RA, CG8710-RC, CG2160-RA, CG11508-RA, CG2163-RA, CG30374-RA, CG8707-RA, CG30373-RA, CG11482-RA, CG14756-RA, CG2291-RA, CG12126-RA, CG14755-RA, CG30364-RA, CG34218-RA, CG30366-RA, CG2127-RA, CG8701-RA, CG2121-RA, CG11650-RA, CG8697-RA, CG8695-RA, CG8693-RA, CG30360-RA, CG11669-RA, CG8690-RA, CG8734-RA, CG12376-RA, CG14746-RA, CG14743-RA, CG8577-RA, CG14745-RA, CG30357-RA, CG8746-RA, CG33141-RA, CG8266-RA, CG8258-RA, CG8252-RA, CG34219-RA, CG13749-RA, CG13748-RA, CG8230-RA, CG8226-RA, CG8172-RA, CG11778-RA, CG13742-RA, CG12759-RA, CG8777-RA, CG8069-RA, CG8058-RA, CG30344-RA, CG8008-RA, CG8046-RA, CG2412-RA, CG8029-RA, CG2040-RA, CG30343-RA, CG2049-RB, CG34141-RA, CG1968-RA, CG2078-RA, CG13739-RA, CG12158-RA, CG8804-RA, CG8805-RA, CG11804-RA, CG33757-RA, CG1809-RA, CG1868-RA, CG1827-RA, CG1782-RA, CG10459-RA, CG1776-RA, CG33800-RA, CG1688-RA, CG1648-RA, CG1656-RA, CG34033-RA, CG12924-RA, CG1665-RA, CG18446-RA, CG1472-RA, CG30007-RA, CG1441-RA, CG1429-RA, CG15863-RA, CG12133-RA, CG12131-RA, CG12134-RA, CG2328-RA, CG15862-R2-RA, CG12129-RA, CG2249-RA, CG12918-RA, CG12921-RA, CG1362-RA, CG1371-RA, CG1381-RA, CG17870-RA, CG11866-RA, CG12917-RA, CG33135-RA, CG12214-RA, CG34221-RA, CG12912-RA, CG12914-RA, CG12210-RA, CG12913-RA, CG18408-RI, CG12902-RA, CG12897-RA, CG12895-RA, CG18377-RA, CG12892-RA, CG11777-RA, CG12891-RA, CG17765-RA, CG12052-RG, CG11883-RC, CG16728-RA, CG12934-RA, CG11895-RA, CG30018-RC, CG7637-RA, CG12936-RA, CG12341-RA, CG12342-RA, CG12323-RA, CG12938-RA, CG12325-RA, CG7704-RA, CG18003-RA, CG30020-RA, CG11979-RA, CG12943-RA, CG30490-RA, CG33503-RA, CG13232-RA, CG7737-RA, CG30021-RA, CG13217-RA, CG34224-RA, CG9080-RA, CG13226-RA, CG13223-RA, CG13222-RA, CG13214-RA, CG9073-RA, CG9070-RA, CG9067-RA, CG13221-RA, CG30033-RB, CG13211-RA, CG13208-RA, CG12389-RA, CG13206-RA, CG13209-RA, CG12350-RA, CG18681-RA, CG30025-RA, CG12351-RA, CG12384-RA, CG12352-RA, CG30030-RA, CG13204-RA, CG13203-RA, CG9027-RD, CG30022-RA, CG30026-RA, CG34054-RA, CG30034-RA, CG34228-RA, CG13198-RA, CG8234-RA, CG8996-RA, CG13195-RA, CG13193-RA, CG12444-RA, CG13192-RA, CG34230-RA, CG8986-RA, CG13190-RA, CG30038-RA, CG13183-RA, CG8972-RA, CG8967-RA, CG8964-RA, CG17509-RA, CG13186-RA, CG30040-RA, CG18343-RA, CG8881-RA, CG8378-RA, CG13178-RA, CG8878-RA, CG34232-RA, CG8439-RA, CG8862-RA, CG8860-RA, CG13175-RA, CG16792-RA, CG8457-RA, CG8858-RA, CG8857-RA, CG13170-RA, CG13168-RC, CG8850-RA, CG30046-RB, CG13163-RA, CG34021-RA, CG8493-RA, CG30042-RA, CG13155-RA, CG8501-RA, CG8505-RA, CG8511-RA, CG33627-RA, CG8515-RA, CG13157-RA, CG8520-RA, CG30334-RA, CG8545-RA, CG8824-RB, CG30044-RB, CG8581-RA, CG33752-RA, CG30056-RA, CG33775-RA, CG8592-RA, CG17759-RA, CG33671-RA, CG8767-RA, CG12373-RA,

CG12374-RA, CG17579-RA, CG17580-RA, CG17577-RA, CG30488-RA, CG3644-RA, CG3830-RA, CG13319-RA, CG13322-RA, CG33798-RA, CG3915-RA, CG34438-RB, CG3969-RA, CG4604-RA, CG4016-RA, CG4616-RA, CG30058-RA, CG4627-RA, CG4630-RA, CG4643-RA, CG4663-RA, CG12765-RA, CG4679-RA, CG4688-RA, CG4714-RA, CG4716-RA, CG17041-RD, CG4734-RA, CG17047-RA, CG17048-RA, CG17050-RA, CG17049-RA, CG4744-RA, CG18278-RA, CG33470-RA, CG4812-RA, CG30061-RA, CG4832-RA, CG30062-RA, CG5912-RA, CG13329-RA, CG6016-RA, CG13331-RB, CG13332-RA, CG13335-RA, CG12464-RA, CG17716-RA, CG6220-RA, CG6280-RA, CG6305-RA, CG6315-RA, CG13339-RA, CG6337-RA, CG10808-RA, CG30485-RA, CG13344-RA, CG33155-RA, CG13348-RA, CG6704-RA, CG6701-RA, CG13350-RA, CG8085-RC, CG30482-RA, CG18371-RA, CG30483-RA, CG8241-RA, CG8257-RA, CG8323-RA, CG30069-RA, CG8404-RA, CG30071-RA, CG8415-RA, CG8468-RA, CG8485-RA, CG8553-RA, CG8561-RA, CG8589-RA, CG30067-RA, CG17385-RA, CG17388-RA, CG17390-RB, CG10110-RA, CG30197-RA, CG30076-RA, CG10119-RA, CG12868-RA, CG10130-RA, CG10131-RA, CG12862-RA, CG10143-RA, CG12866-RA, CG10816-RA, CG10151-RA, CG10200-RA, CG10212-RA, CG33505-RA, CG10240-RA, CG10243-RA, CG10249-RA, CG30475-RA, CG7639-RA, CG7449-RA, CG11798-RA, CG18285-RA, CG11807-RA, CG11808-RA, CG8079-RA, CG8095-RB, CG16827-RA, CG34123-RD, CG8102-RA, CG8153-RA, CG8155-RA, CG8156-RA, CG8157-RA, CG8171-RA, CG30472-RA, CG34318-RA, CG30470-RA, CG8179-RA, CG30469-RA, CG30468-RA, CG30467-RA, CG8192-RA, CG8200-RA, CG30466-RA, CG33139-RA, CG30090-RA, CG30087-RA, CG33461-RA, CG30083-RA, CG30089-RA, CG30084-RA, CG30321-RA, CG33465-RB, CG8246-RA, CG30085-RA, CG8355-RA, CG33463-RA, CG8291-RA, CG8295-RA, CG8320-RA, CG8322-RA, CG8370-RA, CG8392-RA, CG8401-RA, CG8403-RA, CG18250-RA, CG8424-RA, CG30095-RA, CG8428-RA, CG8430-RB, CG10734-RA, CG8441-RA, CG8445-RA, CG8448-RA, CG15706-RA, CG3666-RA, CG15701-RA, CG7786-RA, CG15707-RA, CG3730-RA, CG15704-RA, CG30100-RA, CG3615-RA, CG34399-RC, CG7997-RA, CG15710-RA, CG6262-RA, CG15711-RA, CG33960-RA, CG7848-RA, CG33550-RA, CG33459-RA, CG4439-RA, CG4700-RC, CG4750-RA, CG4905-RE, CG8380-RA, CG4927-RC, CG4918-RA, CG8306-RA, CG5089-RA, CG34458-RA, CG15616-RA, CG5267-RA, CG5550-RA, CG5935-RA, CG6435-RA, CG34190-RA, CG34191-RA, CG9010-RA, CG9000-RA, CG8980-RA, CG8963-RA, CG9635-RD, CG30459-RA, CG9640-RA, CG9646-RA, CG6953-RA, CG8950-RA, CG30460-RA, CG8938-RA, CG15611-RA, CG18730-RA, CG17876-RA, CG15918-RA, CG11400-RA, CG15917-RA, CG17290-RA, CG12699-RA, CG33197-RA, CG30101-RA, CG4798-RA, CG6550-RA, CG4802-RA, CG6546-RA, CG18468-RA, CG6536-RB, CG34192-RA, CG6520-RA, CG30103-RA, CG18431-RA, CG4844-RA, CG4866-RA, CG4886-RA, CG6510-RA, CG4903-RA, CG6501-RA, CG30105-RA, CG10683-RA, CG4954-RA, CG30108-RA, CG14485-RA, CG10933-RA, CG34195-RA, CG6406-RA, CG6401-RA, CG4996-RA, CG5002-RA, CG5005-RA, CG6370-RA, CG18635-RA, CG30325-RA, CG14491-RA, CG33996-RA, CG10910-RB, CG5770-RA, CG14495-RA, CG5765-RA, CG10911-RA, CG10912-RA, CG5098-RB, CG12767-RA, CG5738-RA, CG5733-RA, CG5134-RB, CG10916-RA, CG5140-RA, CG5721-RA, CG14500-RA, CG30114-RA, CG5581-RA, CG14502-RA, CG18537-RA, CG18538-RA, CG18539-RA, CG18540-RA, CG15066-RA, CG18108-RA, CG18106-RA, CG16844-RA, CG16836-RA, CG15068-RA, CG17523-RA, CG17524-RA, CG17533-RA, CG5190-RA, CG17669-RA, CG10924-RA, CG5327-RA, CG5335-RA, CG5482-RA, CG30122-RB, CG33724-RA, CG5469-RB, CG33136-RA, CG15085-RA, CG15087-RA, CG17821-RA, CG33147-RA, CG15078-RA, CG15098-RA, CG15083-RA, CG15101-RA, CG12758-RA, CG18608-RA, CG30126-RA, CG15116-RA, CG30125-RA, CG15110-RA, CG15118-RA, CG15112-RA, CG10737-RA, CG7097-RA, CG11949-RA,

CG33453-RA, CG7229-RA, CG11906-RA, CG18606-RA, CG15119-RA, CG7417-RA, CG7461-RA, CG11961-RB, CG10073-RA, CG10081-RA, CG17246-RA, CG9325-RB, CG7626-RA, CG7735-RA, CG9291-RA, CG9277-RA, CG7744-RA, CG8201-RA, CG11228-RA, CG15121-RA, CG16716-RB, CG15905-RA, CG11025-RA, CG15126-RA, CG30129-RA, CG30128-RA, CG11218-RA, CG13873-RA, CG30448-RA, CG13872-RA, CG10822-RA, CG8654-RA, CG16898-RA, CG11048-RA, CG11208-RA, CG9864-RA, CG11055-RA, CG9183-RA, CG8914-RA, CG13868-RA, CG16739-RA, CG16741-RA, CG11192-RA, CG30154-RA, CG18067-RA, CG13427-RA, CG30148-RA, CG30145-RA, CG18066-RA, CG13424-RA, CG13436-RA, CG33786-RA, CG33785-RA, CG11136-RA, CG8994-RA, CG13441-RA, CG34201-RA, CG34202-RA, CG11312-RA, CG30295-RA, CG9235-RA, CG34115-RA, CG15650-RA, CG3221-RA, CG10543-RA, CG30291-RA, CG15653-RA, CG9353-RA, CG9357-RA, CG30293-RA, CG3986-RA, CG10531-RA, CG4030-RA, CG9394-RA, CG9401-RA, CG9406-RA, CG9437-RA, CG34397-RB, CG34203-RA, CG10069-RB, CG30392-RA, CG10071-RA, CG9752-RA, CG30387-RB, CG9754-RA, CG9485-RB, CG15666-RA, CG9822-RA, CG15671-RA, CG10795-RA, CG9841-RA, CG15667-RA, CG10494-RA, CG10079-RA, CG30286-RA, CG10433-RA, CG10080-RA, CG10082-RA, CG10320-RA, CG15678-RA, CG9865-RA, CG10318-RA, CG30263-RA, CG10306-RA, CG10307-RA, CG17921-RA, CG30403-RA, CG30404-RA, CG18735-RA, CG13492-RA, CG34040-RA, CG4363-RA, CG4377-RA, CG9294-RB, CG34369-RA, CG9304-RA, CG13495-RA, CG13501-RA, CG11475-RA, CG6437-RA, CG6562-RA, CG13502-RA, CG11061-RA, CG6698-RA, CG34205-RA, CG6741-RA, CG34206-RA, CG3045-RA, CG11170-RB, CG11275-RA, CG5625-RA, CG3074-RA, CG3290-RA, CG11291-RA, CG5709-RA, CG30278-RA, CG5799-RA, CG5820-RD, CG32885-RA, CG34208-RA, CG10955-RA, CG3624-RA, CG3613-RA, CG5821-RA, CG3633-RA, CG6339-RA, CG10384-RA, CG4554-RA, CG4610-RA, CG6018-RA, CG11362-RA, CG3927-RA, CG4294-RA, CG4046-RA, CG4071-RA, CG5179-RA, CG4207-RA, CG4414-RA, CG30195-RA, CG34446-RA, CG34445-RA, CG4373-RA, CG13511-RA, CG13512-RA, CG12190-RA, CG13521-RA, CG30270-RA, CG30271-RB, CG30266-RA, CG30274-RA, CG30265-RA, CG13532-RA, CG3499-RB, CG3495-RA, CG3700-RA, CG30193-RA, CG3800-RA, CG32834-RA, CG9897-RA, CG13542-RA, CG9895-RA, CG9890-RA, CG9888-RA, CG9882-RA, CG9877-RA, CG9876-RA, CG3215-RA, CG13544-RA, CG12192-RA, CG13545-RA, CG13549-RA, CG3502-RA, CG34210-RA, CG30410-RA, CG3500-RA, CG34423-RA, CG3493-RA, CG13551-RA, CG3520-RA, CG9812-RB, CG13557-RA, CG13559-RA, CG30181-RA, CG33151-RA, CG33150-RA, CG3957-RA, CG17280-RA, CG3941-RA, CG5372-RA, CG5373-RA, CG5393-RB, CG33706-RA, CG17664-RA, CG5403-RA, CG5428-RA, CG5431-RA, CG4533-RA, CG5472-RA, CG12273-RA, CG4084-RA, CG18128-RA, CG4051-RA, CG5532-RA, CG30177-RA, CG13561-RA, CG5539-RA, CG4817-RA, CG4735-RA, CG11173-RA, CG17611-RA, CG30176-RA, CG4581-RA, CG5575-RA, CG5597-RA, CG5339-RA, CG2827-RA, CG2970-RA, CG16787-RA, CG34213-RA, CG3029-RA, CG3065-RA, CG3832-RA, CG3105-RA, CG3860-RA, CG33519-RB, CG3907-RA, CG30173-RA, CG30172-RA, CG3924-RA, CG3173-RA, CG3988-RA, CG13568-RB, CG3195-RA, CG3997-RA, CG10339-RA, CG16786-RA, CG3231-RA, CG4049-RA, CG3253-RA, CG3260-RA, CG12240-RB, CG13570-RA, CG3362-RA, CG30419-RA, CG3394-RA, CG3401-RA, CG4354-RA, CG3411-RA, CG4527-RB, CG4545-RA, CG3419-RA, CG4563-RA, CG3492-RA, CG13579-RA, CG13589-RA, CG13590-RA, CG13581-RA, CG4612-RA, CG13592-RA, CG13586-RA, CG4634-RA, CG33527-RA, CG3579-RA, CG13588-RA, CG4741-RA, CG3663-RA, CG3683-RA, CG34214-RA, CG4806-RA, CG9196-RA, CG12848-RA, CG16936-RA, CG16910-RA, CG2928-RA, CG18105-RA, CG2917-RA, CG3629-RA, CG3650-RA, CG34413-RB, CG34405-RC, CG2811-RA, CG3770-RA, CG9358-RA, CG2736-RA, CG3829-RA, CG15792-RA, CG34038-RA, CG2665-RA, CG3340-RA, CG30428-RA.

The following genes are provided as a training set of expressed genes, independent of the *Adh* test region. They were obtained from the drosophila, April 2006 genome assembly using the UCSC table browser. See Section 4.5 for more detail.

CG6094-RA, CG7384-RA, CG7400-RA, CG7400-RB, CG6098-RA, CG6113-RA, CG31721-RA, CG6138-RA, CG6138-RB, CG17104-RA, CG6443-RA, CG17118-RA, CG6750-RA, CG6444-RA, CG6743-RA, CG6737-RA, CG12299-RA, CG6729-RA, CG6495-RA, CG6720-RA, CG6720-RB, CG6724-RA, CG17127-RA, CG31869-RA, CG6700-RA, CG6647-RA, CG6647-RB, CG6647-R, CG6627-RA, CG6620-RA, CG6521-RA, CG33129-RA, CG33129-RB, CG33129-R, CG33129-RE, CG4621-RA, CG16743-RA, CG6093-RA, CG6105-RA, CG4636-RA, CG6122-RA, CG12253-RA, CG16833-RB, CG16833-R, CG16833-RA, CG6137-RA, CG6141-RA, CG6141-RB, CG4579-RA, CG4579-RB, CG16840-RA, CG4584-RA, CG4584-RB, CG31868-RA, CG4705-RA, CG4705-RB, CG6181-RA, CG6181-RB, CG6192-RA, CG4713-RA, CG6230-RA, CG6249-RA, CG4738-RA, CG6258-RA, CG4751-RA, CG4779-RA, CG6287-RA, CG33305-RA, CG6320-RA, CG6320-R, CG6320-RD, CG6320-RB, CG4788-RA, CG4807-RA, CG4807-RB, CG32830-RA, CG6392-RA, CG33694-RA, CG33695-RA, CG33695-R, CG33695-RE, CG33695-RD, CG33695-RF, CG4851-RA, CG16928-RA, CG4881-RA, CG4881-RB, CG6464-RA, CG6464-RB, CG4922-RA, CG14928-RA, CG6509-RA, CG6509-RB, CG31705-RA, CG31705-RB, CG31705-R, CG31705-R.

Bibliography

Michelle N. Arbeitman, Eileen E. M. Furlong, Farhad Imam, Eric Johnson, Brian H. Null, Bruce S. Baker, Mark A. Krasnow, Matthew P. Scott, Ronald W. Davis, and Kevin P. White. 'Gene Expression During the Life Cycle of Drosophila melanogaster'. *Science*, 297: 2270–2275, 2002.

M. Ashburner, S. Misra, J. Roote, S. E. Lewis, R. Blazej, T. Davis, C. Doyle, R. Galle, R. George, N. Harris, G. Hartzell, D. Harvey, L. Hong, K. Houston, R. Hoskins, G. Johnson, C. Martin, A. Moshrefi, M. Palazzolo, M. G. Reese, A. Spradling, G. Tsang, K. Wan, K. Whitelaw, B. Kimmel, S. Celniker, and G. M. Rubin. 'An Exploration of the Sequence of a 2.9-Mb Region of the Genome of Drosophila melanogaster: The Adh Region'. *Genetics*, 153:179–219, 1999.

L. E. Baum and T. Petrie. 'Statistical inference for probabilistic functions of finite state Markov chains'. *Annals of Mathematical Statistics*, 37:1554–1563, 1966.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss. 'A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains'. *Annals of Mathematical Statistics*, 41(1):164–171, 1970.

H. Bengtsson, K. Simpson, J. Bullard, and K. Hansen. 'aroma.affymetrix: A generic framework in R for analyzing small to very large Affymetrix data sets in bounded memory'. (745), February 2008.

Axel Bernal, Koby Crammer, Artemis Hatzigeorgiou, and Fernando Pereira. 'Global Discriminative Learning for Higher-Accuracy Computational Gene Prediction'. *PLoS Computational Biology*, 3(3):488–497, 2007.

Paul Bertone, Viktor Stolc, Thomas E Royce, Joel S Rozowsky, Alexander E Urban, Xiaowei Zhu, John L Rinn, Waraporn Tongprasit, Manoj Samanta, Sherman Weissman, Mark B Gerstein, and Michael Snyder. 'Global Identification of Human Transcribed Sequences with Genome Tiling Arrays'. *Science*, 306:2242–2246, 2004.

B. E. Blaisdell. 'Markov chain analysis finds a significant influence of neighboring bases on the occurrence of a base in eucaryotic nuclear DNA sequences both protein-coding and noncoding'. *Journal of Molecular Evolution*, 21(3):278–288, 1984.

B. M. Bolstad, R. A. Irizarry, M. Astrand, and T. P. Speed. 'A comparison of normalization methods for high density oligonucleotide array data based on variance and bias'. *Bioinformatics*, 19:185–193, 2003.

M. Borodovsky and J. McIninch. 'GeneMark: parallel gene recognition for both DNA strands'. *Computers and Chemistry*, 17(19):123–133, 1993.

Bronislava Brejová. *'Evidence Combination in Hidden Markov Models for Gene Prediction'*. PhD thesis, University of Waterloo, 2005.

Christopher Burge. *'Identification of genes in human genomic DNA'*. PhD thesis, Standford University, 1997.

Christopher Burge and Samual Karlin. 'Prediction of Complete Gene Structures in Human Genomic DNA'. *Journal of Molecular Biology*, 268(78-94), 1997.

M. Burset and R. Guigó. 'Evaluation of gene structure prediction programs'. *Genomics*, 34: 353–367, 1996.

M. Burset, I. A. Seledtsov, and V. V. Solovyev. 'SpliceDB: database of canonical and non-canonical mammalian splice sites'. *Nucl. Acids Res.*, 29(1):255–259, 2001. doi: 10.1093/nar/29.1.255. URL `http://nar.oxfordjournals.org/cgi/content/abstract/29/1/255`.

Enrique Castillo, José Manuel Gutiérrez, and Ali S. Hadi. *'Expert Systems and Probabilistic Network Models'*. Springer-Verlag, 1997.

Peter Clote and Rolf Backofen. *'Computational Molecular Biology: An Introduction'*. John Wiley and Sons, New York, USA, 2000.

Lior David, Wolfgang Huber, Marina Granovskaia, Joern Toedling, Curtis J Palm, Lee Bofkin, Ted Jones, Ronald W Davis, and Lars M Steinmetz. 'A high-resolution map of transcription in the yeast genome'. *PNAS*, 103(14):5320–5325, 2006.

David DeCaprio, Jade P. Vinson, Matthew D. Pearson, Philip Montgomery, Matthew Doherty, and James E. Galagan. 'Conrad: Gene prediction using conditional random fields'. *Genome Research*, 17:1389–1398, 2007.

A. P. Dempster, N. M. Laird, and D. B Rubin. 'Maximum likelihood from incomplete data via the EM algorithm'. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

Jiang Du, Joel S Rozowsky, Jan O Korbel, Zhengdong D Zhang, Thomas E Royce, Martin H Schultz, Michael Snyder, and Mark B Gerstein. 'A supervised hidden Markov model framework for efficiently segmenting tiling array data in transcriptional and chIP-chip experiments: systematically incorporating validated biological knowledge'. *Bioinformatics*, 22(24):3016–3024, 2006.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchinson. *'Biological sequence analysis: Probabilistic models of proteins and nucleic acids'*. Cambridge University Press, 1998.

Sean Eddy. 'Non-coding RNA genes and the modern RNA world'. *Nature Reviews Genetics*, 2(12):919–929, 2001.

James W. Fickett and Chang-Shung Tung. 'Assessment of protein coding measure'. *Nucleic Acids Research*, 20(24):6441–6450, 1992.

Malcolm J. Gardner, Neil Hall, Eula Fung, Owen White, Matthew Berriman, Richard W. Hyman, Jane M. Carlton, Arnab Pain, Karen E. Nelson, Sharen Bowman, Ian T. Paulsen, Keith James, Jonathan A. Eisen, Kim Rutherford, Steven L. Salzberg, Alister Craig, Sue Kyes, Man-Suen Chan, Vishvanath Nene, Shamira J. Shallom, Bernard Suh, Jeremy Peterson, Sam Angiuoli, Mihaela Pertea, Jonathan Allen, Jeremy Selengut, Daniel Haft, Michael W. Mather, Akhil B. Vaidya, David M. A. Martin, Alan H. Fairlamb, Martin J. Fraunholz, David S. Roos, Stuart A. Ralph, Geoffrey I. McFadden, Leda M. Cummings, G. Mani Subramanian, Chris Mungall, J. Craig Venter, Daniel J. Carucci, Stephen L. Hoffman, Chris Newbold, Ronald W. Davis, Claire M. Fraser, and Bart Barrell. 'Genome sequence of the human malaria parasite Plasmodium falciparum'. *Nature*, 419:498–511, 2002.

J. Willard Gibbs. *'Elementary principles in statistical mechanics'*. New York: C. Scribner, 1902.

Larry Gonick and Mark Wheelis. *'The cartoon guide to genetics'*. HarperPerennial, New York, 1991.

Samuel Gross, Chuong Do, Marina Sirota, and Serafim Batzoglou. 'CONTRAST: a discriminative, phylogeny-free approach to multiple informant de novo gene prediction'. *Genome Biology*, 8(12):R269, 2007. ISSN 1465-6906. doi: 10.1186/gb-2007-8-12-r269. URL `http://genomebiology.com/2007/8/12/R269`.

Samuel S. Gross and Michael R. Brent. 'using multiple alignments to improve gene prediction'. *Journal of Computational Biology*, 13(2):379–393, 2006. doi: 10.1089/cmb.2006.13.379. URL `http://www.liebertonline.com/doi/abs/10.1089/cmb.2006.13.379`. PMID: 16597247.

Gabor Halasz, Marinus van Batenburg, Joelle Perusse, Sujun Hua, Xiang-Jun Lu, Kevin White, and Harmen Bussemaker. 'Detecting transcriptionally active regions using genomic tiling arrays'. *Genome Biology*, 7(7):R59, 2006. ISSN 1465-6906. doi: 10.1186/gb-2006-7-7-r59. URL `http://genomebiology.com/2006/7/7/R59`.

J. M. Hammersley and P. Clifford. 'Markov fields on finite graphs and lattices'. 1971.

Housheng He, Jie Wang, Tao Liu, X. Shirley Liu, Tiantian Li, Yunfei Wang, Zuwei Qian, Haixia Zheng, Xiaopeng Zhu, Tao Wu, Baochen Shi, Wei Deng, Wei Zhou, Geir Skogerbo, and Runsheng Chen. 'Mapping the C. elegans noncoding transcriptome with a whole-genome tiling microarray'. *Genome Res.*, 17(10):1471–1477, 2007. doi: 10.1101/gr.6611807. URL `http://genome.cshlp.org/cgi/content/abstract/17/10/1471`.

Wolfgang Huber, Joern Toedling, and Lars M Steinmetz. 'Transcript mapping with high-density oligonucleotide tiling arrays'. *Bioinformatics*, 22(16):1963–1970, 2006.

Rafael A Irizarry, Bridget Hobbs, Francios Collin, Yasmin D Beazer-Barclay, Kristen J Antonellis, Uwe Scherf, and Terence P Speed. 'Exploration, normalization, and summaries of high density oligonucleotide array probe level data'. *Biostatistics*, 3(2):249–264, 2003.

Hongkai Ji and Wing Hung Wong. 'TileMap: create chromosomal map of tiling array hybridizations'. *Bioinformatics*, 21(18):3629–3636, 2005.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. 'SciPy: Open source scientific tools for Python', 2001–. URL `http://www.scipy.org/`.

Dione Kampa, Jill Cheng, Philipp Kapranov, Mark Yamanaka, Shane Brubaker, Simon Cawley, Jorg Drenkow, Antonio Piccolboni, Stefan Bekiranov, Gregg Helt, Hari Tammana, and Thomas R. Gingeras. 'Novel RNAs Identified From an In-Depth Analysis of the Transcriptome of Human Chromosomes 21 and 22'. *Genome Research*, 14:331–342, 2004.

D. Karolchik, R. M. Kuhn, R. Baertsch, G. P. Barber, H. Clawson, M. Diekhans, B. Giardine, R. A. Harte, A. S. Hinrichs, F. Hsu, K. M. Kober, W. Miller, J. S. Pedersen, A. Pohl, B. J. Raney, B. Rhead, K. R. Rosenbloom, K. E. Smith, M. Stanke, A. Thakkapallayil, H. Trumbower, T. Wang, A. S. Zweig, D. Haussler, and W. J. Kent. 'The UCSC Genome Browser Database: 2008 update'. *Nucleic Acids Research*, 36:D773–9, 2008.

Donna Karolchik, Angela S. Hinrichs, Terrence S. Furey, Krishna M. Roskin, Charles W. Sugnet, David Haussler, and W. James Kent. 'The UCSC Table Browser data retrieval tool'. *Nucleic Acids Research*, 32:D493–6, 2004.

W. James Kent, Charles W. Sugnet, Terrence S. Furey, Krishna M. Roskin, Tom H. Pringle, Alan M. Zahler, and David Haussler. 'The human genome browser at UCSC'. *Genome Res.*, 12(6):996–1006, 2002.

Ian Korf, Paul Flicek, Daniel Duan, and Michael R. Brent. 'Integrating genomic homology into gene structure prediction'. *Bioinformatics*, 17(Suppl. 1):S140–S148, 2001.

M. Kozak. 'An analysis of 5'-noncoding sequences from 699 vertebrate messenger RNAs'. *Nucleic Acids Research*, 15:8125–48, 1987.

M. Kozak. 'Structural Features in Eukaryotic mRNAs That Modulate the Initiation of Translation'. *Journal of Biological Chemistry*, 266(30):19867–19870, 1991.

Anders Krogh. 'Using Database Matches with HMMGene for Automated Gene Detection in Drosophila'. *Genome Res.*, 10(4):523–528, 2000. doi: 10.1101/gr.10.4.523. URL `http://genome.cshlp.org/cgi/content/abstract/10/4/523`.

Anders Krogh, I. SairaMian, and David Haussler. 'A hidden Markov model that finds genes in E. Coli DNA'. *Nucleic Acids Research*, 22(22):4768–4778, 1994.

David Kulp, David Haussler, Martin G. Reese, and Frank H. Eeckman. *'A generalized hidden markov model for the recognition of human genes in DNA'*. AAAI Press, 1996.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 'Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data'. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1.

S. Lauritzen. *'Graphical models'*. Oxford University Press, 1996.

Wei Li, Clifford A Meyer, and X Shirley Liu. 'A hidden Markov model for analyzing ChIP-chip experiments on genome tiling arrays and its application to p53 binding sequences'. *Bioinformatics*, 21(Suppl. 1):274–282, 2005.

William Majoros. *'Methods for Computational Gene Prediction'*. Cambridge University Press, New York, USA, 2007.

J Robert Manak, Sujit Dike, Victor Sementchenko, Philipp Kapranov, Frederic Biemar, Jeff Long, Jill Cheng, Ian Bell, Srinka Ghosh, Antonio Piccolboni, and Thomas R Gingeras. 'Biological function of unannotated transcription during the early development of Drosophila Melanogaster'. *Nature Genetics*, 38:1151–1158, 2006.

John S. Mattick. 'Challenging the dogma: the hidden layer of non-protein-coding RNAs in complex organisms'. *BioEssays*, 25:930–939, 2003.

John S. Mattick and Igor V. Makunin. 'Non-coding RNA'. *Human Molecular Genetics*, 15: R17–29, 2006.

Irmtraud M. Meyer. 'A practical guide to the art of RNA gene prediction'. *Briefings in bioinformatics*, 8(6):396–414, 2007.

F. Naef and M. Magnasco. 'Solving the riddle of the bright mismatches: labeling and effective binding in oligonucleotide arrays'. *Physical Review E*, 68(1 Pt 1), 2003.

Travis Oliphant. 'Python for Scientific Computing'. *Computing in Science and Engineering*, 9(3):10–20, 2007.

Antonio Piccolboni. 'Multivariate Segmentation in the Analysis of Transcription Tiling Array Data'. *Research in Computational Molecular Biology*, 4453:311–324, 2007.

R Development Core Team. *'R: A Language and Environment for Statistical Computing'*. R Foundation for Statistical Computing, Vienna, Austria, 2008. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

Lawrence R Rabiner. 'A Tutorial on Hidden Markov Models and Selection Applications in Speech Recognition'. *Proceedings of the IEEE*, 77(2):257–285, 1989.

Martin G. Reese, George Hartzell, Nomi L. Harris, Uwe Ohler, Josep F. Abril, and Suzanna E. Lewis. 'Genome Annotation Assessment in Drosophila Melanogaster'. *Genome Res.*, 10: 483–501, 2000.

Elena Rivas and Sean Eddy. 'Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs'. *Bioinformatics*, 16(7):583–605, 2000.

Thomas E Royce, Joel S Rozowsky, and Mark B Gerstein. 'Assessing the need for sequence-based normalization in tiling microarray experiments'. *Bioinformatics*, 23(8):988–997, 2007.

Jeremy Silver. *'Detecting copy-number variation among related individuals'*. Honours thesis, 2007.

Marie Skovgaard, Lars Juhl Jensen, Søren Brunak, David Ussery, and Anders Krogh. 'On the total number of genes and their length distribution in complete microbial genomes'. *Trends in Genetics*, 17(8):425–428, 2001.

George W. Snedecor and William G. Cochran. *'Statistical Methods'*. Iowa State University Press, 1989.

M. Stanke and S. Waack. 'Gene prediction with a hidden Markov model and a new intron submodel'. *Bioinformatics*, 19 Suppl 2, October 2003. ISSN 1460-2059. URL `http://view.ncbi.nlm.nih.gov/pubmed/14534192`.

Guido van Rossum. 'Python reference manual'. 1995. URL `http://www.python.org`.

A. J. Viterbi. 'Error bounds for convolutional codes and an asymptotically optimal decoding algorithm'. *IEEE Transactions on information theory*, IT-13:260–269, 1967.

Chaochun Wei and Michael Brent. 'Using ESTs to improve the accuracy of de novo gene prediction'. *BMC Bioinformatics*, 7(1):327, 2006. ISSN 1471-2105. doi: 10.1186/1471-2105-7-327. URL `http://www.biomedcentral.com/1471-2105/7/327`.

R.J. Wilson, J.L. Goodman, V.B. Strelets, and the FlyBase Consortium. 'FlyBase: integration and improvements to query tools'. *Nucleic Acids Research*, 36:D588–93, 2008.

Zhijin Wu, Rafael A Irizarry, Robert Gentleman, Francisco Martinez-Murillo, and Forrest Spencer. 'A Model-Based Background Adjustment for Oligonucleotide Expression Arrays'. *Journal of the American Statistical Association*, 99(468):909–917, 2004.

Na Xu. *'Transcriptome Detection by Multiple RNA Tiling Array Analysis and Identifying Functional Conserved Non-coding Elements by Statistical Testing'*. PhD thesis, University of California, Berkeley, 2008.

Ru-Fang Yeh, Lee P Lim, and Christopher Burge. 'Computational Inference of Homologous Gene Structures in the Human Genome'. *Genome Research*, 11(5):803–816, 2001.

Georg Zeller, Stefan R Henz, Sascha Laubinger, Detlef Weigl, and Gunnar Rätsch. 'Transcript normalization and segmentation of tiling array data'. *Pacific Symposium on Biocomputing*, pages 527–538, 2008.