Gene Prediction with a Hidden Markov Model

Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultäten
der Georg-August-Universität zu Göttingen

vorgelegt von

Mario Stanke

aus Witzenhausen

Göttingen, 2003

D7

Referent:	Prof. Dr. Stephan Waack
Korreferent:	Prof. Dr. Burkhard Morgenstern
Tag der mündlichen Prüfung:	21.01.2004

Contents

Für meine Eltern

Abstract

Annotation of the large and rapidly increasing amount of genomic sequence data requires computational tools for finding genes in DNA sequences. This thesis presents a computational method for finding protein-coding genes encoded in DNA sequences of eukaryotes (plants and animals). It consists mainly of two parts.

One major part of the thesis is the introduction of a so-called generalized Hidden Markov Model (GHMM) for eukaryotic genomic sequences. This model, called AUGUSTUS, is a probabilistic model of a DNA sequence with the gene structure underlying the sequence. It defines a probability distribution on the set of all possible pairs of a DNA sequence and its annotation of protein-coding regions. Genes in an input DNA sequence can be uncovered by finding the gene structure which is most likely in the probabilistic model given the input DNA sequence. The most likely gene structure of the input DNA sequence is searched by a computer program, which can be done both exactly and efficiently because of the relatively simple dependency structure of the distribution defined by a GHMM. However, the most likely gene structure with respect to the model is not necessarily correct. In order for the model to fit well the actual distribution of true sequences and their annotations several new methods have been applied. A GHMM for gene prediction contains probabilistic state models for different functional parts of the genomic sequence, such as translational and splicing signals and coding regions. For the splice sites new probabilistic submodels are introduced. A method is used to better estimate the parameters of the model depending on the average base frequency. Further, the following issue is addressed. A GHMM may model the length distribution of certain structural parts of the sequence, such as introns. The disadvantage of the existing procedures was that they either caused prohibitively long running times or they modeled the true length distribution inadequately. An approach presented here allows to approximate a given length distribution by an arbitrary initial part and a geometric tail at relatively low computational cost. Furthermore, this thesis presents an example that shows that incorporating a state model of a GHMM, that is a better probabilistic model for the true distribution, can worsen the prediction accuracy of the overall GHMM.

A computer program based on this model has been tested on DNA sequences with known annotation from human and the fruit fly *Drosophila melanogaster*. The accuracy of the predictions compare favorably to that of other well known, established gene prediction programs.

The second major part of the thesis addresses insecure external information about the gene structure and presents a method for integrating external information into a GHMM for gene prediction. With the increasing number of sequenced species and an increasing size of protein and EST databases the relevance of so-called *extrinsic* gene prediction methods

becomes all the more important. Extrinsic gene prediction methods use external evidence about the DNA sequence whose gene structure needs to be found. This external evidence can, as employed in this thesis, be a local similarity of the input DNA sequence to a known EST sequence or a local similarity of the translated DNA sequence to a known protein sequence. But also a second DNA sequence with unknown gene structure can be used to infer knowledge about the gene structure of the input sequence if this second sequence codes for a similar protein of a different species in a suitable evolutionary distance. This thesis presents a flexible model for integrating various types of external information into a GHMM for gene prediction.

The GHMM AUGUSTUS is extended to a new GHMM AUGUSTUS+ which is a probabilistic model of all possible *triplets* formed by a DNA sequence, its annotation, and external information about the sequence. The gene prediction program then finds the most likely annotation given *both* the input DNA sequence *and* the external evidence. It accounts for the fact that such external evidence can be misleading. The parameters corresponding to the distribution of the external information can be easily estimated. This leads to a naturally justified increase in likelihood of gene structures respecting the external information compared to the likelihood in the previous model. The method allows to make use of evidence about a *range* of the DNA sequence, e.g. evidence that a certain range of the sequence is protein-coding, without preferring gene structures that only 'partially respect' that evidence over those which do not respect at all the evidence. Another advantage of the method presented here, compared to ad-hoc methods for integrating external information into existing programs, is that the underlying theory of GHMMs still applies to the model. Experiments with AUGUSTUS+ show that the use of extrinsic information coming from EST database searches can significantly improve the prediction accuracy of gene prediction programs when combined with protein database searches.

Acknowledgments

I am much obliged to my supervisor Professor Stephan Waack for his support throughout the last three years. He has guided me to this thesis topic which later turned out to be very motivating and promising. In the numerous discussions with him I got many helpful hints and ideas. He has coordinated the diploma thesis of Oliver Schöffmann and my thesis so our collaboration became fruitful. I return thanks to Oliver Schöffmann for providing me with the essential input to AUGUSTUS+ by writing AGRIPPA and following my many request for trying out different settings on many data sets. Thanks goes to the Institut für Mikrobiologie und Genetik for the computer resources to run AGRIPPA. I am grateful to Emmanouil Stafilarakis for writing the input modules of AUGUSTUS and a useful floating point type and laying of the foundation stone to the program AUGUSTUS. Many thanks to Rasmus Steinkamp, who has set up a web server and written a web interface for AUGUSTUS. Special thanks belongs to Oliver Keller, for discussing problems with me and proofreading the draft. I also thank Geneviève Macias for proofreading the abstract. I am grateful to Professor Burkhard Morgenstern for making it possible that I could finish this thesis while working in his work group and for proofreading a paper deriving from this thesis.

Abbreviations

ASS	acceptor splice site
bp	base pairs
cDNA	complementary DNA, a DNA copy of an mRNA molecule
DNA	deoxyribonucleic acid, the genetic material for all cellular life forms
DSS	donor splice site
EST	expressed sequence tag
GFF	General Feature Format
GHMM	generalized Hidden Markov Model
GHz	giga hertz
GOBICS	Göttingen Bioinformatics Compute Server
HMM	Hidden Markov Model
IMM	interpolated Markov Model
Kb	kilo bases = 1000 bases
MB	mega bytes
Mb	mega bases = 1000000 bases
RNA	ribonucleic acid
WWAM	windowed weight array model
$\#(\ldots)$	the number of ...
$\lvert A \rvert$	the size of set A

Chapter 1

Introduction

A DNA sequence can be represented by a word over the alphabet with the four letters A,C,G and T standing for the *nucleotides* or *bases* adenine, cytosine, guanine and thymine. Over the past few years the DNA sequences of many organisms have been determined. Two types of organisms can be distinguished. *Eukaryotes* like animals and plants are organisms whose cells contain membrane-bound compartments. *Prokaryotes* are organisms whose cells lack extensive internal compartments, namely bacteria and archae. So far, 21 eukaryotic genomes have been sequenced and their sequence published (see http://igweb.integratedgenomics.com/GOLD/). Examples are the bakers yeast (1997), the worm *Caenorhabditis elegans* (1998), the fruit fly *Drosophila melanogaster* (2000), the plant *Arabidopsis thaliana*, the human (2001), the malaria parasite *Anopheles gambiae* (2002) and the mouse (2002). 360 additional eukaryotic genome sequencing projects are currently under way. For prokaryotes these figures are even higher.

These sequencing efforts generate a large amount of raw data as the DNA sequence of a eukaryote is often longer than a hundred million base pairs. The genome of humans has approximately a size of $3.2 \cdot 10^9$ base pairs. The annotation of these sequences by biological means can by far not keep pace with the speed with which the data is accumulated and therefore computational tools to find genes are needed. Locating the genes is helpful and often even a prerequisite for further analysis such as the characterization of the function of the gene product, determining the phylogeny of different species or understanding gene regulation.

But the problem of finding genes in a genomic DNA sequence is difficult and – despite extensive efforts – has not been solved yet satisfactorily. The accuracy of current gene prediction programs is not high enough to yield reliable information. Nevertheless, the results of such programs are used for the automated annotation of genomes and there is a demand for fast gene finders which are as accurate as possible. The research area of finding the protein coding genes in a eukaryotic genomic sequence by computational

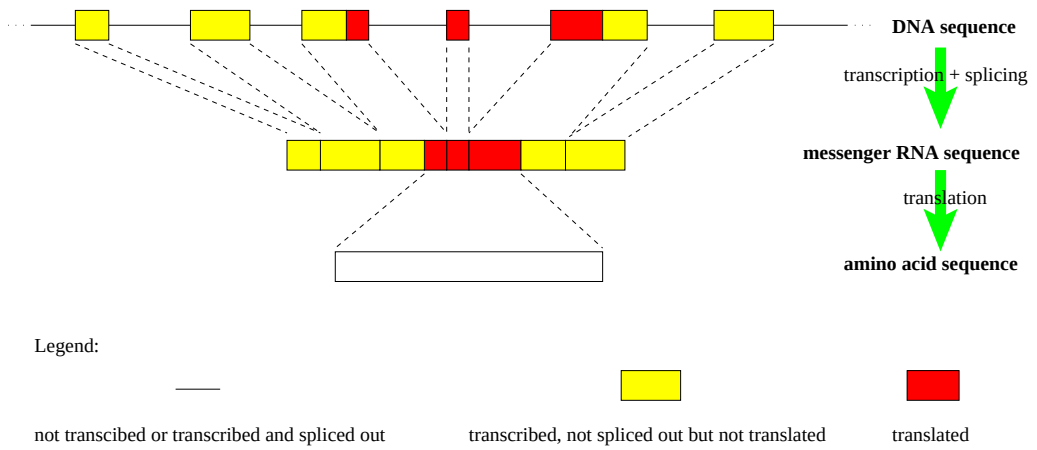

Figure 1.1: A simplified scheme of gene expression.

methods has become very competitive. In [MSSR02] an overview over such programs is given, referencing 22 homology-based programs plus 25 *ab initio* programs (possibly with homology integration).

Most of the estimated 30000-40000 human genes code for proteins. In Figure 1.1 the simplified process is shown how the eukaryotic cells make a protein molecule from a DNA sequence containing the gene. First a copy of a contiguous region of DNA larger than the actually coding part is made. This process is called *transcription*. Out of this sequence the *introns* are spliced out and the result is the concatenation of the *exons*, the so called messenger RNA (mRNA). The number of exons per gene varies. Some genes consist of only one exon and therefore do not contain introns. The human muscle protein *titin* constitutes the other extreme. It consists of 178 exons. In humans there are on the average about nine exons per gene [Bro02]. An internal contiguous part of the mRNA is then translated sequentially to an amino acid sequence in chunks of three consecutive nucleotides called *codons* according to the genetic code. The translation stops directly after the first appearance of one of three stop codons: TAA, TAG, TGA. Finally the amino acid sequence is folded 3-dimensionally in a way determined (almost always) by the amino acid sequence itself. The *reading frame* of a protein-coding sequence is the information which bases of the sequence are at the first, second or third position of a codon.

In the chromosomes the DNA sequence is found in a double stranded sequence, where each nucleotide in one strand is opposed by a complementary nucleotide in the other strand. A is the complement of T and vice versa. In the same way C and G are complementary bases. As the sequence of nucleotides of one strand is fully determined by the sequence of nucleotides on the opposite strand, we only need to know the sequence of one of the strands. We call this strand the *forward strand* and its complementary strand the *reverse strand*. Both strands do have a (chemically) distinguished direction. The sequence at

the top in Figure 1.1 is the forward strand. Here, and in all figures below, the forward sequence has its *downstream* end at the right and its *upstream* end at the left. The reverse strand goes in the opposite direction. Genes can be on both of the two strands.

Usually two neighboring genes (regardless whether they are on the same strand or on opposite strands) are separated by an *intergenic region*, but exceptions to this organization are known. In the human mitochondrial genome the coding regions (sic) of the genes ATP8 and ATP6, which are on the same strand, overlap by 46 nucleotides (the reading frames in the overlapping region are different). An example of a gene lying within another gene is the human neurofibromatosis type I gene on chromosome 17, which has three short genes on the opposite strand within one of its introns. Each of these internal genes has introns itself. These exceptions seem to be rare, though.

Chapter 2

Generalized Hidden Markov Models

2.1 Notation

The following notation about strings will be used in this chapter and the following ones. An alphabet is a countable set. The elements of an alphabet are called letters or characters. For an alphabet Σ the set Σ^+ denotes the set of all possible strings (words) of arbitrary finite length $n \geq 1$ which can be build from characters of Σ. For a string $\sigma = \sigma_1\sigma_2\cdots\sigma_n$ we denote with $|\sigma| = n$ the length of σ. The set Σ^n denotes the set of words of length n. The symbol ε denotes the empty string, which consists of 0 characters and has length 0. We define $\Sigma^* := \Sigma^+ \cup \{\varepsilon\}$. For a string $\sigma = \sigma_1\sigma_2\cdots\sigma_n$ and indices $1 \leq i < j \leq n$ we denote with $\sigma[i,j], \sigma(i,j], \sigma[i,j)$ the substrings $\sigma_i\cdots\sigma_j$, $\sigma_{i+1}\cdots\sigma_j$ and $\sigma_i\cdots\sigma_{j-1}$, respectively. For two strings σ and τ and indices $a < b$ the statement $[\sigma = \tau]_{a..b}$ is an abbreviation of the statement $\sigma[a,b] = \tau[a,b]$.

2.2 Definition of a Generalized Hidden Markov Model

2.1 Definition (Markov chain)
A sequence of random variables $X_1, X_2, \ldots$ which take values in a countable set Q, is called a *Markov chain* of order $k \geq 1$ if for all $i > k$ and all $x_1, x_2, \ldots, x_i \in Q$

$$\mathrm{P}(X_i = x_i \,|\, X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) = \mathrm{P}(X_i = x_i \,|\, X_{i-k} = x_{i-k}, \ldots, X_{i-1} = x_{i-1}).$$

The sequence is called a *homogenous* Markov chain if $\mathrm{P}(X_i = x_{k+1} \,|\, X_{i-k} = x_1, \ldots, X_{i-1} = x_k)$ does not depend on i $(x_1, \ldots, x_{k+1} \in Q)$, otherwise it is called *inhomogeneous*. For a homogenous Markov chain of order 1 the matrix $A = (a_{r,s})_{r,s\in Q}$ with $a_{r,s} = \mathrm{P}(X_i = s \,|\, X_{i-1} = r)$ is called the *transition matrix*. The set Q is called the *state space*. If $X_i = q$, then the process is said to be in *state* q at *time* i.

If the order of a Markov chain is not mentioned, it is assumed to be 1. To fully specify the distribution of a Markov chain the distribution of X_1, the *initial distribution*, must also be given. In order to simplify the notation we introduce a special *initial state* q_{init} and another constant random variable $X_0 \equiv q_{\text{init}}$. Then the distribution of the Markov chain $X_0, X_1, \ldots$ is fully determined by the transition matrix. We also introduce a special *terminal state* q_{term} that – once entered – cannot be left. We write

$$Q^+ = Q \cup \{q_{\text{init}}, q_{\text{term}}\}. \tag{2.1}$$

The extended transition matrix $A = (a_{i,j})_{i,j \in Q^+}$ must satisfy

$$\begin{aligned} a_{q,q_{\text{init}}} &= 0 \quad (q \in Q^+) \\ a_{q_{\text{init}},q_{\text{term}}} &= 0 \\ a_{q,q_{\text{term}}} &> 0 \quad (\text{ for at least one } q \in Q) \\ a_{q_{\text{term}},q_{\text{term}}} &= 1. \end{aligned} \tag{2.2}$$

In words this means that the process starts in the initial state, immediately leaves that state, stays some time within the states of Q, possibly jumps to the terminal state at some time, and then stays in the terminal state forever. Let

$$T := \begin{cases} \inf\{t \,|\, X_t = q_{\text{term}}\} - 1. & \text{if this set is not empty;} \\ \infty & \text{otherwise.} \end{cases} \tag{2.3}$$

We demand that the transition matrix is such that T is almost surely finite:
$P(T < \infty) = 1$. Then the process almost always eventually enters the terminal state and T is the number of states passed before the process enters the terminal state, i.e. $X_T \neq q_{\text{term}}$ and $X_{T+1} = q_{\text{term}}$. There are two reasons for introducing such a terminal state. One reason is the requirement that the sequence $X_1, X_2, \cdots, X_T$ be finite as any real world sequence. The other reason is that it will later easily enable us to consider only sequences which end in certain states.

2.2 Definition (GHMM)
Let the state space Q^+ and the transition matrix A be as specified above and let Σ be a countable set, the *emission alphabet*. Further, let the probabilities $e_{i,j,\tau}(\sigma)$ be defined for $i, j \in Q^+, \tau, \sigma \in \Sigma^*$. A *generalized Hidden Markov Model (GHMM)* with state space Q^+, transition matrix A and *emission probabilities* $e_{i,j,\tau}(\sigma) \quad (i, j \in Q^+, \tau, \sigma \in \Sigma^*)$ is a sequence

$$(X_0, Y_0), (X_1, Y_1), (X_2, Y_2), \ldots$$

where $X_0 \equiv q_{\text{init}}$, the sequence $X_0, X_1, X_2, \ldots$ is a homogenous Markov chain on the state space Q^+ with transition matrix A and where $Y_0, Y_1, \ldots$ is a sequence of random variables with values in Σ^* such that $Y_0 \equiv \varepsilon$ and

$$\begin{aligned} e_{x_{i-1},x_i,\tau}(y_i) &= \mathrm{P}(Y_i = y_i \,|\, X_{i-1} = x_{i-1},\, X_i = x_i,\, Y_0 Y_1 \cdots Y_{i-1} = \tau) \\ &= \mathrm{P}(Y_i = y_i \,|\, X_0 = x_0,\, \ldots, X_i = x_i,\, Y_0 = y_0,\, \ldots,\, Y_{i-1} = y_{i-1}) \end{aligned}$$

for all $i > 0$, for all $x_0, \ldots, x_i \in Q^+$, $y_0, \ldots y_i \in \Sigma^*$ and $\tau = y_0 y_1 \cdots y_{i-1}$. The initial and terminal state always emit the empty word which is never emitted by any other state:

$$e_{r,s,\tau}(\varepsilon) = 0 \quad (r \in Q^+, s \in Q, \tau \in \Sigma^*) \text{ and}$$
$$e_{r,q_{\text{init}},\tau}(\varepsilon) = 1, e_{r,q_{\text{term}},\tau}(\varepsilon) = 1 \quad (r \in Q^+, \tau \in \Sigma^*).$$

We denote with $\mathbf{X}$ the sequence of states $X_0, X_1, \ldots$ and with $\mathbf{Y}$ the sequence of *observations* $Y_0, Y_1, \ldots$.

Let $x_1, \ldots, x_n \in Q$ and $d_1, \ldots, d_n \geq 1$. The vector

$$\varphi = ((x_1, d_1), \ldots, (x_n, d_n)) \tag{2.4}$$

is called a *parse* of length ℓ if $d_1 + \cdots + d_n = \ell$. It *ends* in x_n and consists of n *steps*. The parse $\varphi(\mathbf{X}, \mathbf{Y})$ induced by $(\mathbf{X}, \mathbf{Y})$ is defined by

$$\varphi(\mathbf{X}, \mathbf{Y}) := ((X_1, |Y_1|), \ldots (X_T, |Y_T|)) \tag{2.5}$$

For $(\mathbf{x}, \mathbf{y}) = ((x_0, x_1, \ldots), (y_0, y_1, \ldots))$ and $\ell \geq 1$, the *ℓ-truncated* parse *induced* by $(\mathbf{x}, \mathbf{y})$ is defined by

$$\varphi_l(\mathbf{x}, \mathbf{y}) := ((x_1, |y_1|), \ldots (x_r, |y_r|)) \text{ with } r := \max\{n \,|\, |y_1| + \cdots + |y_n| \leq \ell \text{ and } x_n \neq q_{\text{term}}\} \tag{2.6}$$

and can be interpreted as the longest initial subparse whose emission length does not exceed ℓ. Note that $\varphi_l(\mathbf{x}, \mathbf{y})$ is a parse of length ℓ if $|y_1| + \cdots + |y_n| = \ell$ for some n. Let $\sigma(\mathbf{Y})$ denote the string obtained by concatenating the strings $Y_0, Y_1, \ldots$.

We are now ready to explain the practical intention behind this formal model. The Y_i's are called *emissions* and $\sigma(\mathbf{Y})$ is observable. But the answer to the question in which state each of the letters of $\sigma(\mathbf{Y})$ was emitted is unknown and needs to be guessed. In other words, the parse $\varphi(\mathbf{X}, \mathbf{Y})$ is 'hidden' and, in the applications of GHMMs, needs to be uncovered using the observation $\sigma(\mathbf{Y})$. The word 'generalized' in GHMM refers to the fact that – as opposed to an ordinary HMM – the states in a GHMM may emit more than one symbol. In an ordinary HMM the strings Y_i all have length 1.

Immediately from the definition of GHMMs follows the following lemma, which states how the probability of a sequence of states and emissions can be computed.

2.3 Lemma

Let $t > 0, x_1, \ldots x_t \in Q^+$, $y_1, \ldots, y_t \in \Sigma^*$, $x_0 = q_{\text{init}}$, $y_0 = \varepsilon$. Then

$$P(((X_1, Y_1), \ldots, (X_t, Y_t)) = ((x_1, y_1), \ldots, (x_t, y_t))) = \prod_{i=1}^{t} a_{x_{i-1}, x_i} e_{x_{i-1}, x_i, y_0 \cdots y_{i-1}}(y_i) \tag{2.7}$$

Proof:

$$\begin{aligned} & P(((X_1,Y_1),\ldots,(X_t,Y_t)) = ((x_1,y_1),\ldots,(x_t,y_t))) \\ = \; & P((X_1,Y_1)=(x_1,y_1)) \\ & \cdot P((X_2,Y_2)=(x_2,y_2)\,|\,(X_1,Y_1)=(x_1,y_1)) \\ & \ldots. \\ & \cdot P((X_t,Y_t)=(x_t,y_t)\,|\,(X_1,Y_1)=(x_1,y_1),\cdots,(X_{t-1},Y_{t-1})=(x_{t-1},y_{t-1})) \\ = \; & \prod_{i=1}^{t} a_{x_{i-1},x_i}\cdot e_{x_{i-1},x_i,y_0\cdots y_{i-1}}(y_i) \end{aligned}$$

□

Remark: Some authors ([Bur97], [Ree00]) describing GHMMs seem to forget that in their model the emissions do not only depend on the current state but also on the sequence τ emitted so far. Their prediction programs use Markov chains for the emission in the states modeling non-coding regions. In that case the emission distribution depends on bases that have been emitted in previous steps.

2.3 Viterbi Algorithm

Given an observation $\sigma \in \Sigma^*$ of length t, an intuitive choice as a guess for the unknown parse is to take a most likely parse, given the observation. Such a parse ψ_{vit} is called *Viterbi parse*:

$$\psi_{\text{vit}} \in \underset{\psi \text{ parse of length } t}{\operatorname{argmax}} \mathrm{P}(\varphi(\mathbf{X},\mathbf{Y})=\psi \,|\, \sigma(\mathbf{Y})=\sigma). \tag{2.8}$$

We call the conditional distribution of $(\mathbf{X},\mathbf{Y})$ given that $\sigma(\mathbf{Y})=\sigma$ the *a-posteriori* distribution. Thus the Viterbi parse is a parse with maximum a-posteriori probability. A Viterbi parse can efficiently be calculated using a variant of the so-called *Viterbi algorithm* [Vit67] which is described in the following. Let an input sequence σ of length t be given. We define the *Viterbi variables*

$$\gamma_{q,\ell} := \max_{\substack{\psi \text{ parse of length } \ell \\ \text{ending in } q}} \mathrm{P}(\varphi_l(\mathbf{X},\mathbf{Y})=\psi,\, [\sigma(\mathbf{Y})=\sigma]_{1..l}) \tag{2.9}$$

for all $q \in Q$ and $1 \le \ell \le t$. For convenience we also set $\gamma_{q_{\text{init}},0}=1$ and $\gamma_{q,0}=0$ for all $q \neq q_{\text{init}}$. The Viterbi variables can be computed using a simple recursion. This recursion can be deduced from first conditioning on whether ψ has more than one step and, if so,

conditioning on the last step of ψ.

$$\begin{aligned}
\gamma_{q,\ell} &= \max\Big\{ \mathrm{P}\big(\varphi_\ell(\mathbf{X},\mathbf{Y})=\psi,\, [\sigma(\mathbf{Y})=\sigma]_{1..\ell}\big) \,\Big|\, \psi=((q,\ell)) \text{ or } \\
&\qquad\qquad \psi=(\psi',(q,d)) \text{ parse of length } \ell \text{ ending in } q \Big\} \qquad (2.10)\\
&= \max\Big\{ \quad a_{q_{\text{init}},q}\cdot e_{q_{\text{init}},q,\varepsilon}(\sigma), \\
&\qquad \max_{\substack{q'\in Q,\,\ell'=\ell-d\\ \psi' \text{ parse of length}\\ \ell' \text{ ending in } q'}} \mathrm{P}\big(\varphi_{\ell'}(\mathbf{X},\mathbf{Y})=\psi',[\sigma(\mathbf{Y})=\sigma]_{1..\ell'}\big)\cdot a_{q',q}\cdot e_{q',q,\sigma[1,\ell']}(\sigma(\ell',\ell]) \Big\}\\
&= \max\Big\{ a_{q_{\text{init}},q}\cdot e_{q_{\text{init}},q,\varepsilon}(\sigma),\, \max_{1\le\ell'<\ell,q'\in Q} \gamma_{q',\ell'}\cdot a_{q',q}\cdot e_{q',q,\sigma[1,\ell']}(\sigma(\ell',\ell]) \Big\}
\end{aligned}$$

Making use of the definition $\gamma_{q_{\text{init}},0}=1$ we receive the **Viterbi recursion**

$$\boxed{\gamma_{q,\ell} = \max_{\substack{1\le\ell'<\ell,q'\in Q\\ \text{or } q'=q_{\text{init}},\ell'=0}} \gamma_{q',\ell'}\cdot a_{q',q}\cdot e_{q',q,\sigma[1,\ell']}(\sigma(\ell',\ell])} \qquad (2.11)$$

2.4 Theorem

Let σ be an emission of length t. Let $\psi=((x_1,d_1),\ldots,(x_n,d_n))$ with $x_1,\ldots,x_n\in Q$ be a parse of length t and let $x_0:=q_{\text{init}}$. Define $s_i:=d_1+\cdots+d_i \quad (i=1,\ldots,n)$. If ψ satisfies

$$\gamma_{x_n,t}\cdot a_{x_n,q_{\text{term}}} = \max_{q\in Q}\gamma_{q,t}\cdot a_{q,q_{\text{term}}} \qquad (2.12)$$

and, for all $i=1,\ldots,n$

$$\gamma_{x_i,d_1+\cdots+d_i} = \gamma_{x_{i-1},d_1+\cdots+d_{i-1}}\cdot a_{x_{i-1},x_i}\cdot e_{x_{i-1},x_i,\sigma[1,s_{i-1}]}(\sigma(s_{i-1},s_i]) \qquad (2.13)$$

then ψ is a Viterbi parse.

Proof: As $\mathrm{P}\big(\varphi(\mathbf{X},\mathbf{Y})=\psi,\sigma(\mathbf{Y})=\sigma\big)=\mathrm{P}\big(\sigma(\mathbf{Y})=\sigma\big)\cdot P\big(\varphi(\mathbf{X},\mathbf{Y})=\psi\,|\,\sigma(\mathbf{Y})=\sigma\big)$ every parse of length t that maximizes $P\big(\varphi(\mathbf{X},\mathbf{Y})=\psi,\sigma(\mathbf{Y})=\sigma\big)$ is a Viterbi parse. Let p_{vit} be this maximal probability and let ψ be a parse of length t that satisfies (2.12) and (2.13). Then

$$\begin{aligned}
P(\varphi(\mathbf{X},\mathbf{Y})=\psi,\sigma(\mathbf{Y})=\sigma) &= \left(\prod_{i=1}^{n} a_{x_{i-1},x_i}\cdot e_{x_{i-1},x_i,\sigma[1,s_{i-1}]}(\sigma(s_{i-1},s_i])\right)\cdot a_{x_n,q_{\text{term}}}\\
&= \gamma_{x_n,t}\cdot a_{x_n,q_{\text{term}}}\\
&= \max_{q\in Q}\gamma_{q,t}\cdot a_{q,q_{\text{term}}}\\
&= p_{\text{vit}}
\end{aligned}$$

Here the first line follows from (2.7), the second line by induction using (2.13) and the last by the definition of the $\gamma_{q,t}$'s. □

Theorem 2.4 suggests the

Viterbi algorithm:

1) Compute and store the table of the Viterbi variables $\gamma_{q,\ell}$ for $q \in Q, 1 \leq \ell \leq t$ by increasing ℓ.

2) Set $q_1 \leftarrow \underset{q \in Q}{\operatorname{argmax}} \gamma_{q,t} \cdot a_{q,q_{\text{term}}}$
$\ell_1 \leftarrow t$

3) $i \leftarrow 2$
While $\ell_{i-1} > 0$ do
Begin
Set $(q_i, \ell_i) \leftarrow \underset{(q,\ell) \in Q \times [1,\ell_{i-1}) \cup \{(q_{\text{init}},0)\}}{\operatorname{argmax}} \gamma_{q,\ell} \cdot a_{q,q_{i-1}} \cdot e_{q,q_{i-1},\sigma[1,\ell]}(\sigma(\ell, \ell_{i-1}])$
$i \leftarrow i + 1$
End

4) $n \leftarrow i - 2$
Output the parse $\psi = ((q_n, \ell_n - \ell_{n+1}), \ldots, (q_1, \ell_1 - \ell_2))$ as Viterbi parse.

The space complexity of this direct implementation is $O(|Q| \cdot t)$. The time complexity depends much on the time needed to compute the arguments of the maximum in the Viterbi recursion (2.11). In a usual application the transition probabilities can be stored beforehand and be accessed in constant time. In simple cases (e.g. if the GHMM is a HMM) the emission probabilities can also be stored in memory. But in a usual application of GHMMs the emission probabilities will be computed according to some formula only when they are needed.

2.4 Forward Algorithm

An algorithm that is closely related to the Viterbi algorithm is the so-called *forward algorithm*. The Viterbi algorithm allows us to compute a parse ψ_{vit} with maximal a-posteriori probability, but it does *not* allow to actually compute the a-posteriori probability of ψ_{vit}. This can be achieved by the forward algorithm, which can be used to compute the probability of an emission σ, and thus the a-posteriori probability of parses. It is also needed as a preprocessing step for the sampling algorithms discussed in the next section.

Analogously to the Viterbi variables we define the *forward variables*

$$\begin{aligned} \alpha_{q,\ell} &:= \mathrm{P}(\varphi_\ell(\mathbf{X}, \mathbf{Y}) \text{ is a parse of length } \ell \text{ ending in } q,\ [\sigma(\mathbf{Y}) = \sigma]_{1..\ell}) \\ &= \sum_{\substack{\psi \text{ parse of length } \ell \\ \text{ending in } q}} \mathrm{P}(\varphi_\ell(\mathbf{X}, \mathbf{Y}) = \psi,\ [\sigma(\mathbf{Y}) = \sigma]_{1..\ell}) \end{aligned} \tag{2.14}$$

for all $q \in Q$ and $1 \leq \ell \leq t$. Again, we also set $\alpha_{q_{\text{init}},0} = 1$ and $\alpha_{q,0} = 0$ for all $q \neq q_{\text{init}}$. Based on Formula (2.14) we can derive a recursion for the forward variables in the same

way as for the Viterbi variables just with maxima replaced by sums. This yields the following **forward recursion** for the forward variables.

$$\boxed{\alpha_{q,\ell} = \sum_{\substack{1\le\ell'<\ell, q'\in Q \\ \text{or } q'=q_{\text{init}}, \ell'=0}} \alpha_{q',\ell'} \cdot a_{q',q} \cdot e_{q',q,\sigma[1,\ell']}(\sigma(\ell',\ell])} \tag{2.15}$$

The *forward algorithm* simply consists of the computation of the forward variables.

Forward algorithm:

1. Iteratively compute the forward variables $\alpha_{q,\ell}$ by increasing ℓ using a dynamic programming table and recursion (2.15)

The forward variables can be used to compute the probability $P(\sigma(\mathbf{Y}) = \sigma)$ of an emission σ.

2.5 Theorem

Let $\sigma \in \Sigma^*$ be an emission of length t. Then

$$P(\sigma(\mathbf{Y}) = \sigma) = \sum_{q\in Q} \alpha_{q,t} \cdot a_{q,q_{\text{term}}} \tag{2.16}$$

Proof: For all $q \in Q$ we have

$$\alpha_{q,t} \cdot a_{q,q_{\text{term}}} = \mathrm{P}(\varphi(\mathbf{X},\mathbf{Y}) \text{ is a parse of length } t \text{ ending in } q,\, [\sigma(\mathbf{Y}) = \sigma]_{1..t})$$

Thus (2.16) follows by the application of the law of the total probability. □

Knowing the probability of an emission σ enables us to compute the a-posteriori probabilities of parses via the following

2.6 Theorem (a-posteriori probability of a parse)

Let $\sigma \in \Sigma^+$ be an emission of length t and let $\psi = ((x_1, d_1), \ldots, (x_n, d_n))$ be a parse of length t. Define $y_1, y_2, \ldots y_n \in \Sigma^*$ such that these strings concatenate to σ, $y_1 y_2 \cdots y_n = \sigma$, and such that $|y_i| = d_i$ for $i = 1, \ldots n$ and let $x_0 := q_{\text{init}}, y_0 := \varepsilon$. Then

$$P(\varphi(\mathbf{X},\mathbf{Y}) = \psi \,|\, \sigma(\mathbf{Y}) = \sigma) = \frac{\left(\prod_{i=1}^n a_{x_{i-1},x_i} \cdot e_{x_{i-1},x_i,y_0\cdots y_{i-1}}(y_i)\right) \cdot a_{x_n,q_{\text{term}}}}{\sum_{q\in Q} \alpha_{q,t} \cdot a_{q,q_{\text{term}}}}$$

Proof:

$$\begin{aligned}
& P(\varphi(\mathbf{X},\mathbf{Y}) = \psi \,|\, \sigma(\mathbf{Y}) = \sigma) \\
= \;& \frac{P(\varphi(\mathbf{X},\mathbf{Y}) = \psi, \sigma(\mathbf{Y}) = \sigma)}{P(\sigma(\mathbf{Y}) = \sigma)} \\
= \;& \frac{P(((X_1,Y_1),\ldots,(X_n,Y_n)) = ((x_1,y_1),\ldots,(x_n,y_n))) \cdot a_{x_n,q_{\text{term}}}}{P(\sigma(\mathbf{Y}) = \sigma)} \\
= \;& \frac{\left(\prod_{i=1}^n a_{x_{i-1},x_i} \cdot e_{x_{i-1},x_i,y_0\cdots y_{i-1}}(y_i)\right) \cdot a_{x_n,q_{\text{term}}}}{\sum_{q\in Q} \alpha_{q,t} \cdot a_{q,q_{\text{term}}}}
\end{aligned}$$

Here, the first line is the definition of the conditional probability, the second follows as the parse and the overall emission together determine the realization $(\mathbf{X}, \mathbf{Y})$ of the GHMM and the third by Lemma 2.3 and Theorem 2.5.

□

2.5 Sampling Algorithm

With the Viterbi algorithm we have a means of finding one particular parse. This parse might be wrong in the sense that it differs from the correct parse $\varphi(\mathbf{X}, \mathbf{Y})$. Besides, in the application of gene prediction, alternatively spliced genes correspond to different parses which can all be considered as correct. The sampling algorithm described in this section is a method of 'drawing random samples' of the set of all parses according to their a-posteriori probability. I.e. this method yields a parse ψ with its a-posteriori probability $P(\varphi(\mathbf{X}, \mathbf{Y}) = \psi \,|\, \sigma(\mathbf{Y}) = \sigma)$. Therefore, parses with high a-posteriori probability have a relatively high likelihood of getting sampled. In particular, a Viterbi parse is a parse which is most likely to get sampled. Section 4.1.3 shows some results of applying the sampling algorithm to gene prediction.

The sampling algorithm proceeds by constructing a parse $\psi = ((x_1, d_1), \ldots, (x_n, d_n))$ stepwise in the reverse order by random choices of $x_n, d_n, x_{n-1}, d_{n-1}, x_{n-2}, \ldots, x_1, d_1$.

Let an emission σ of length t be given. We assume that $P(\sigma(\mathbf{Y}) = \sigma) > 0$.

sampling algorithm

1) Compute and store the table of the forward variables
   $\alpha_{q,\ell}$ for $q \in Q, 1 \leq \ell \leq t$.
2) Choose $q_1 \in Q$ at random according to the probability distribution on Q:
   $p_1(q) = \frac{\alpha_{q,t} \cdot a_{q,q_{\text{term}}}}{\sum_{r \in Q} \alpha_{r,t} \cdot a_{r,q_{\text{term}}}}$.
   $\ell_1 \leftarrow t$
3) $i \leftarrow 2$
   While $\ell_{i-1} > 0$ do
   Begin
   Choose an element (q_i, ℓ_i) at random according to the probability distribution on $Q \times [1, \ell_{i-1}) \cup \{(q_{\text{init}}, 0)\}$:

$$p_i(q, \ell) = \frac{\alpha_{q,\ell} \cdot a_{q,q_{i-1}} \cdot e_{q,q_{i-1},\sigma[1,\ell]}(\sigma(\ell, \ell_{i-1}])}{\alpha_{q_{i-1},\ell_{i-1}}}.$$

   $i \leftarrow i + 1$
   End
4) $n \leftarrow i - 2$
   Output the parse $\psi = ((q_n, \ell_n - \ell_{n+1}), \ldots, (q_1, \ell_1 - \ell_2))$.

2.7 Theorem

The sampling algorithm outputs parse ψ with probability $P(\varphi(\mathbf{X}, \mathbf{Y}) = \psi \,|\, \sigma(\mathbf{Y}) = \sigma)$.

Proof: First observe that all probabilities of each random choice sum up to 1. In step 2) this is immediately clear and in step 3) this is because of the forward recursion (2.15).

The probability that parse $\psi = ((q_n, \ell_n - \ell_{n+1}), \ldots, (q_1, \ell_1 - \ell_2))$ gets sampled is the product of all probabilities of the random choices made in constructing ψ.

$$\begin{aligned} P(\psi \text{ gets sampled}) &= p_1(q_1) \cdot \prod_{i=2}^{n+1} p_i(q_i, \ell_i) \\ &= \frac{a_{q_1, q_{\text{term}}}}{\sum_{q \in Q} \alpha_{q,t} \cdot a_{q, q_{\text{term}}}} \prod_{i=2}^{n+1} a_{q_i, q_{i-1}} \cdot e_{q_i, q_{i-1}, \sigma[1, \ell_i](\sigma(\ell_i, \ell_{i-1}])} \qquad (2.17) \\ &= P(\varphi(\mathbf{X}, \mathbf{Y}) = \psi \,|\, \sigma(\mathbf{Y}) = \sigma) \end{aligned}$$

Line (2.17) follows as the denominator α_{q_1, ℓ_1} of $p_2(q_2, \ell_2)$ cancels out with the forward variable in the nominator of $p_1(q_1)$ and the denominator of $p_i(q_i, \ell_i)$ cancels out with the forward variable in the nominator of $p_{i-1}(q_{i-1}, \ell_{i-1})$ $(i = 3, \ldots, n+1)$, and as $q_{n+1} = q_{\text{init}}, \ell_{n+1} = 0$ and therefore $\alpha_{q_{n+1}, \ell_{n+1}} = 1$. The last line follows by Theorem 2.6. □

Chapter 3

A Generalized Hidden Markov Model for Genomic Sequences

3.1 History of GHMMs for Gene Prediction

Surveys over computational gene prediction are given in [Cla97] and [MSSR02]. I will here only restate the roots of Generalized Hidden Markov Models for gene prediction and mention some stepping stones in the development.

One of the earliest published methods for identifying protein coding regions with the computer is reported in [SM82]. They used codon usage statistics to predict which one of the three reading frames is correct in a sequence assumed to be completely coding. The reading frame – also cased phase – of a coding sequence is the information at which of the 3 positions modulo 3 the codons start. They applied their method to a sliding window of a given input DNA sequence and analyzed the results 'manually'. Only a couple of months later Fickett [Fic82] published a method to distinguish between coding and non-coding regions by computing a simple statistic measuring how much the distribution of a nucleotide at the three codon positions deviates from a uniform distribution. These numbers, together with the A-,C-,G- and T-content in the sequence, were used to classify a given input sequence as either coding or non-coding. Again, Fickett assumed that the input sequences were either completely coding or completely non-coding. GENMARK [BM93] is the first gene finder which uses a Markov model. It uses a 3-periodic Markov chain of order 5 for coding regions and another Markov chain of order 5 for non-coding regions. Within a sliding window the probability that this window is coding in any of the 6 reading frames (3 each on the forward and reverse strand) is computed and plotted. This tackled also a common source of error: The prediction of shadow genes on the reverse strand. ECOPARSE [KMH94] is the first gene prediction program which is based on a *Hidden* Markov Model. It predicts genes in prokaryotic DNA (Escherichia coli). In prokaryotes

the genes are contained in contiguous *open reading frames* (ORF, in this thesis: substring and associated reading frame which contains no in-frame stop codon) which are on average about 1000 bases long. This is usually much longer than one would statistically expect in non-coding regions. This fact helps a lot in identifying bacterial genes. In eukaryotes the presence of introns which interrupt the coding sequence and the fact that many exons are not long enough for their open reading frame being statistically conspicuous makes gene prediction much more difficult. VEIL [HSF97] uses a Hidden Markov Model for segmenting eukaryotic DNA into exons, introns and intergenic regions. In this model each state emits exactly one symbol. GENIE [KHRE96] is the first program for the task of gene prediction which bases on a GHMM. The states may emit a whole string of bases at once according to an arbitrary probability model, e.g. the whole splice site region is emitted at once according to a splice site submodel. This allows to take into account longer range dependencies between base positions emitted in one step. When introduced, the GHMM-based program GENSCAN [Bur97], was probably the most accurate tool which made it to a prevalent tool up to the present. GENIE and GENSCAN also use 5th order Markov models as content sensors for the coding and non-coding regions.

3.2 Definition of the GHMM AUGUSTUS

We call the Generalized Hidden Markov Model introduced in this chapter AUGUSTUS. We also refer to the implementation of this model as AUGUSTUS. AUGUSTUS is a model for genomic DNA sequences of arbitrary finite length and the protein coding genes contained in the sequence. The genes modeled here constitute the protein coding part of the DNA sequence, which is drawn red (dark) in Figure 1.1. Below we will refer to the collection of the coding parts of the exons of a protein coding gene as 'the gene', thus not taking into consideration the non-coding exons. Also, we will use the term 'exon' in the following for protein coding exons only. The number of genes is also arbitrary, including the case of a gene-less sequence. Any series of genes on both strands is allowed where the genes do not overlap, neither on the same strand nor on opposite strands. We call the collection of genes together with the exact location of all the exon boundaries a *gene structure*. The model is such that there is a one-to-one correspondence between parses and gene structures, i.e. each parse defines unambiguously a gene structure for the sequence and for each gene structure there is exactly one parse. (In some other GHMMs a gene structure may have several parses: In the model of HMMGene a *set* of parses maps to each gene structure and the probability of this set is estimated by an approximation algorithm [Kro97]. GENSCAN contains a promoter model and different promoter positions in different parses may yield the same gene structure [Bur97]. Nevertheless, Burge uses the Viterbi algorithm.) Genes at the upstream and downstream boundaries of the sequence may be partial (incomplete)

in the sense that only some of the exons of the gene are completely contained in the sequence and the other exons lie beyond the boundary of the sequence.

The model parameters have been estimated using training sets of annotated sequences (see section 7.2). We use different models for human genomic sequences and for those genomic sequences of the fruit fly *Drosophila melanogaster*. These models are very similar: the same states are used, the same transitions are possible and mostly the same methods are used to estimate emission probabilities (from the data of two different training sets). Whenever the models differ we will explicitly say so.

3.2.1 States and Transitions

The states in the state set Q of AUGUSTUS correspond to a biological meaning (e.g. intron, exon, splice site). Transitions between these states are only allowed in a biologically meaningful way (e.g. an acceptor splice site must follow an intron). AUGUSTUS uses the following 47 states:

$$Q = \{\mathrm{IR}, \mathrm{E}_{\mathrm{single}}, \mathrm{E}_{\mathrm{term}}, \mathrm{rE}_{\mathrm{single}}, \mathrm{rE}_{\mathrm{init}}\} \bigcup \bigcup_{i=0}^{2} \{\mathrm{E}^i_{\mathrm{init}}, \mathrm{DSS}^i, \mathrm{I}^i_{\mathrm{short}}, \mathrm{I}^i_{\mathrm{fixed}}, \mathrm{I}^i_{\mathrm{geo}}, \mathrm{ASS}^i, \mathrm{E}^i, \mathrm{rE}^i_{\mathrm{term}}, \mathrm{rDSS}^i, \mathrm{rI}^i_{\mathrm{short}}, \mathrm{rI}^i_{\mathrm{fixed}}, \mathrm{rI}^i_{\mathrm{geo}}, \mathrm{rASS}^i, \mathrm{rE}^i\}$$

$$Q^+ = Q \cup \{q_{\mathrm{init}}, q_{\mathrm{term}}\}$$

We will use in this section the notation of Chapter 2. The states in Q are shown in Figure 3.1. The arrows in this figure denote the non-zero transition probabilities between states in Q. The transition matrix $A = (a_{i,j})_{i,j \in Q^+}$ is defined as follows. The transition probabilities $a_{i,j}$ for $i, j \in Q$ are given in Figure 3.1 rounded to 6 decimal places. The transitions to and from the intron states $\mathrm{I}^j_{\mathrm{geo}}, \mathrm{I}^j_{\mathrm{short}}, \mathrm{I}^j_{\mathrm{fixed}}, \mathrm{rI}^j_{\mathrm{geo}}, \mathrm{rI}^j_{\mathrm{short}}, \mathrm{rI}^j_{\mathrm{fixed}}$ are left out in this figure. These intron transitions implicitly model the intron lengths and are different in the human and fruit fly version. They are defined in Section 3.2.2. The transitions out of state q_{init} (initial probabilities) are

$$a_{q_{\mathrm{init}},q} := \begin{cases} 0 & \text{if } q \in \{q_{\mathrm{init}}, q_{\mathrm{term}}\}; \\ 0.9 & \text{if } q = \mathrm{IR}; \\ 0.01 & \text{if } q \in \{\mathrm{I}^0_{\mathrm{geo}}, \mathrm{I}^1_{\mathrm{geo}}, \mathrm{I}^2_{\mathrm{geo}}, \mathrm{rI}^0_{\mathrm{geo}}, \mathrm{rI}^1_{\mathrm{geo}}, \mathrm{rI}^2_{\mathrm{geo}}\}; \\ 0.001 & \text{otherwise.} \end{cases} \tag{3.1}$$

The remaining transitions into state q_{term} (terminal probabilities) are: $a_{q_{\mathrm{term}},q_{\mathrm{term}}} = 1$ and for all $q \in Q$

$$a_{q,q_{\mathrm{term}}} = \delta \cdot a_{q_{\mathrm{init}},q} \tag{3.2}$$

with $\delta := 10^{-7}$, i.e. the terminal probabilities are proportional to the initial probabilities.

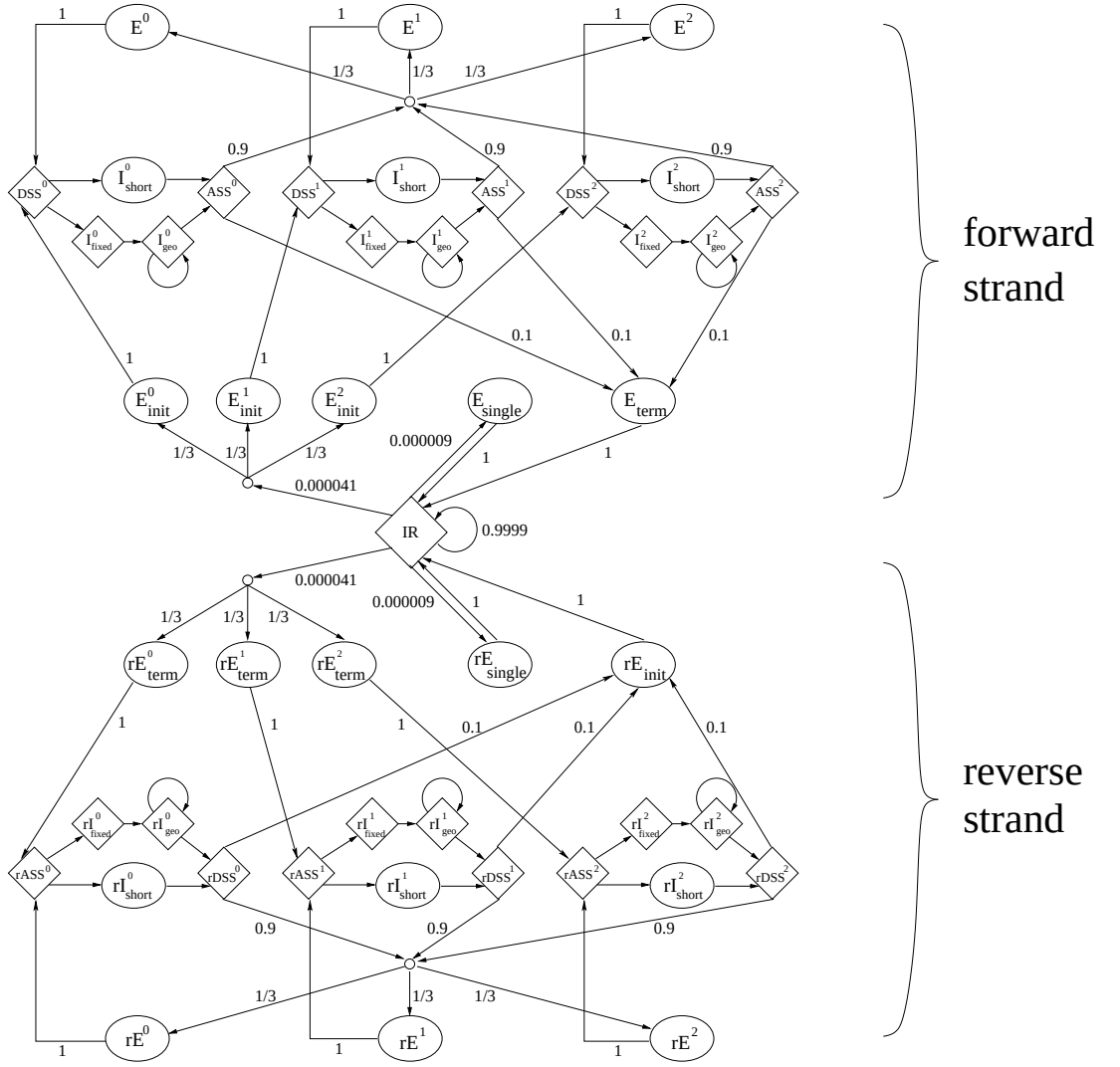

Figure 3.1: The states of AUGUSTUS except $q_{\text{init}}, q_{\text{term}}$ and the possible transitions between them. The states with names beginning with 'r' model the same as those without 'r' but on the reverse strand. E_{single}: a single exon gene. E_{init}: The first coding exon of a multi exon gene. DSS: the donor (5') splice site. I_{short}: an intron at most d nucleotides long. I_{fixed}: the first d nucleotides of a longer intron. I_{geo}: the individual nucleotides after the first d nucleotides of a longer intron. ASS: the acceptor (3') splice site including branch point. E: an internal (coding) exon. E_{term}: the last exon of a multi exon gene. IR: the intergenic region. Diamonds stand for states which emit strings of fixed length, ovals for states with explicit length distribution. The numbers at the arrows are the transition probabilities. The remaining transition probabilities for the intron states are shown in Figure 3.4, they depend on the species. The exponents 0,1,2 stand for the phase of the reading frame. The two small circles are silent states and were only introduced here for clarity.

Remark: Multiplying all terminating probabilities with δ and leaving everything unchanged has no effect on the results of the formula for the a-posteriori probabilities of the parses, which we are mainly interested in. Of course the other probabilities should then be scaled in order to sum up to 1 so that A is again a probability matrix. δ is simply chosen small enough, so that this scaling has no noticeable effect.

The states in the upper half of Figure 3.1 model genes on the forward strand. The state $\mathrm{E}_{\mathrm{single}}$ corresponds to a gene consisting of just one exon. All other states in the upper half correspond to parts of genes with more than one exon, i.e. genes containing introns. In order to 'memorize' the reading frame position of the previous exon there are three states for each type, except for the terminal exon $\mathrm{E}_{\mathrm{term}}$. For the initial exon states $\mathrm{E}^0_{\mathrm{init}}, \mathrm{E}^1_{\mathrm{init}}, \mathrm{E}^2_{\mathrm{init}}$ – corresponding to the first exon of a gene – and the internal exon states $\mathrm{E}^0, \mathrm{E}^1, \mathrm{E}^2$– corresponding to interior exons of a gene – the superscript indicates the reading frame in which the exons *ends*. The superscript is 0 if the exon ends with a complete codon, i.e. the codon boundary and the exon end are the same. The superscript is 1 if the last codon boundary is 1 position before the end of the exon and the superscript is 2 if the last codon boundary is 2 positions before the end of the exon. For the donor splice site states DSS^j, the acceptor splice site states ASS^j and the intron states $\mathrm{I}^j_{\mathrm{short}}, \mathrm{I}^j_{\mathrm{fixed}}, \mathrm{I}^j_{\mathrm{geo}}$ the superscript j stands for the reading frame of the preceeding exon. This construction with a state for each reading frame allows to account for the reading frame in the emission distribution of the exon states, and in particular to exclude in-frame stop codons in an exon sequence. The states in the lower half of Figure 3.1 model genes on the reverse strand. When the forward sequence is stepped through from left to right the parts of a gene on the reverse strand are stepped through in the reverse order as compared to if the gene was on the forward strand. The reverse-strand states on the lower half each have a leading 'r' in the state specifier. They have the same biological meaning as the corresponding states on the forward strand. On the reverse strand the superscripts also indicate the reading frame at the rightmost (with respect to the forward strand input sequence) position of the exon. This is the reason why on the reverse strand there are three states for a terminal exon and only one initial exon state.

3.2.2 Emission Distributions

The emission alphabet Σ of AUGUSTUS is the set of nucleotides: $\Sigma = \{A, C, G, T\}$. We will define the distribution of the random emissions $Y \in \Sigma^*$ (given the current and the previous state) in two steps. First we define a length distribution, $\mathrm{P}(|Y| = \ell)$ for all lengths ℓ, and then $\mathrm{P}(Y = \sigma \,|\, |Y| = \ell)$ for all strings $\sigma \in \Sigma^*$ of length ℓ. Then $\mathrm{P}(Y = \sigma) = \mathrm{P}(|Y| = \ell) \cdot \mathrm{P}(Y = \sigma \,|\, |Y| = \ell)$ is implicitly defined.

Remark: Actually the input sequence may contain a fifth letter (N) for an unknown

nucleotide, which is rare in the test and training sequences when compared to the frequency of the other nucleotides. This case is treated with the following heuristic. Whenever the emission probability of a string containing this unknown nucleotide needs to be computed, a discrete uniform distribution is assumed for some short part of the sequence; for example a nucleotide emission probability of $\frac{1}{4}$ in the case of a Markov model.

Length distributions

The length distributions and the transition probabilities $a_{i,j}$ determine together the length distributions of the biological exons, introns and the intergenic region in our model. We define the length distributions and transition probabilities in such a way that the resulting length distributions of initial exons, internal exons, terminal exons, single exons, introns and intergenic regions are good approximations to the observed length distributions, estimated from the training data.

The diamond-shaped states in Figure 3.1 have a trivial length distribution: Only one fixed length is possible. $\mathrm{I}^j_{\mathrm{geo}}$, the corresponding reverse states and IR always emit strings of length 1, i.e.

$$\sum_{\sigma\in\Sigma} e_{i,j}(\sigma) = 1 \qquad \text{for all } i \in Q^+, j \in \{\mathrm{I}^1_{\mathrm{geo}}, \mathrm{I}^2_{\mathrm{geo}}, \mathrm{I}^3_{\mathrm{geo}}, \mathrm{rI}^1_{\mathrm{geo}}, \mathrm{rI}^2_{\mathrm{geo}}, \mathrm{rI}^3_{\mathrm{geo}}, \mathrm{IR}\}.$$

The states DSS^j, ASS^j and the corresponding reverse states also always emit strings of a certain specific length given on page 34. The states $\mathrm{I}^j_{\mathrm{fixed}}$ and the corresponding reverse states emit only strings of length d which is given on page 31.

Exon length distribution

The length distributions of the states $\mathrm{E}_{\mathrm{single}}, \mathrm{E}_{\mathrm{term}}, \mathrm{E}^0_{\mathrm{init}}, \mathrm{E}^1_{\mathrm{init}}, \mathrm{E}^2_{\mathrm{init}}, \mathrm{E}^0, \mathrm{E}^1, \mathrm{E}^2$ and the corresponding reverse states determine the length distribution of the biological single, terminal, initial and internal exons because the length of the biological exon is always the length of the emission in the exon state plus some constant length of parts of the sequence modeled by the splice site models. We will here describe the length distribution of the biological exons.

In order to estimate the length distribution of the exons we made use of training sets described in Section 3.4. For each of the two species – human and *Drosophila* – and for each of the four types of exons – single, initial, internal, terminal exon – we separately estimate the length distribution from a sample of lengths $\ell_1, \ell_2, \ldots, \ell_n$ given by the training set. From these given exon lengths we compute an estimate for the probabilities π_i that a random exon has length i ($i = 1, 2, \ldots$). The simple solution of using the empirical length distribution, i.e. to estimate π_i by the the relative frequency of length i in the sample cannot be chosen because this would overfit the data. The empirical distribution needs to be smoothed such that random accumulations play a smaller role. It cannot be determined

alone from the sample, whether an accumulation of exons of an approximate length is pure random or a mode of the distribution. Although there are theoretical measures for the quality of such a smoothing technique, it is theoretically not clear how this tradeoff between fitting well and smoothing enough should be settled in favor of the prediction accuracy of the resulting GHMM. We use a kernel estimator with discrete Gaussian kernel function and variable bandwidth. This technique is, for example, described in [BA97].

For a *bandwidth* $\sigma > 0$ let $f_\sigma(k) := c \cdot \frac{1}{\sigma} \cdot \varphi(\frac{k}{\sigma})$ $(k \in \mathbb{Z})$, where φ is the density of the standard Gaussian distribution and c is chosen such that $\sum_{k\in\mathbb{Z}} f_\sigma(k) = 1$. Then on the integers f_σ is close to the density of a Gaussian distribution with mean 0 and standard deviation σ. These *kernel functions* f_σ are used to retrieve a smoothed empirical length distribution in the following way.

$$\pi'_i := \frac{1}{n} \sum_{j=1}^{n} f_{\sigma(\ell_j)}(i - \ell_j) \qquad (i \in \mathbb{Z}) \tag{3.3}$$

It can easily be checked that $(\pi'_i)_{i\in\mathbb{Z}}$ is a probability distribution on the integers. Finally, we exclude the non-positive integers and renormalize to get

$$\pi_i := \pi'_i / \sum_{j=1}^{\infty} \pi'_j \qquad (i = 1, 2, \ldots). \tag{3.4}$$

We let the bandwidth σ depend on the position ℓ. Informally spoken, with this method we distribute the empirical weight $\frac{1}{n}$ of every observed length ℓ to an area centered around ℓ. The 'width' of this area is given by the bandwidth. Lengths close to ℓ get a higher weight than lengths further away from ℓ because of the shape of the density of the Gaussian distribution. We choose the bandwidth such that the width of this area grows as the expected probability of length ℓ decreases. One reason why the latter is necessary is that with increasing lengths the sample data is increasingly sparse such that a using a fixed bandwidth would result in an overfitted distribution.

Heuristically, we expect the probability of a particular length ℓ to be small, when ℓ is large or when few sample lengths lie close to the left of ℓ and few sample lengths lie close to the right of ℓ. Let $L := \{\ell_1, \ldots, \ell_n\}$, a be a positive real constant and m be a positive integer constant. We chose the following bandwidth

$$\sigma(\ell) := \max \begin{cases} \frac{a}{\sqrt[5]{n}}\ell \\ \min\{r \geq 1 \,|\, |\{\ell, \ldots, \ell + r - 1\} \cap L| \geq m \text{ or } |\{\ell - r + 1, \ldots, \ell\} \cap L| \geq m\} \end{cases} \tag{3.5}$$

The proportionality of the bandwidth in the data-independent first case of the maximum in (3.5) to the inverse of the fifth root of the sample size n has been chosen because this proportionality is optimal in the case of density estimation with respect to the mean

integrated square error [BA97]. In the second case of the maximum the bandwidth for length ℓ is chosen to be 1 plus the distance to the m-th nearest neighbor of ℓ in L to the left or to the right, whichever is smaller. We take the minimum of the distances on the two sides because the distribution may drop abruptly at the minimum possible length. The values of the constants are $a = 0.5$ and $m = 8$; the resulting exon length distributions are shown in Figure 3.2.

In order to save space when storing these distributions in a parameter file, we only store the individual probabilities up to a length of 3000 and assume a geometric tail of the length distribution. Only 0.3% of the human and 1% of the *Drosophila* exons were longer than 3000 nucleotides.

We now define the length distributions of the eight forward exon states $\mathrm{E}_{\mathrm{single}}$, $\mathrm{E}^0_{\mathrm{init}}$, $\mathrm{E}^1_{\mathrm{init}}$, $\mathrm{E}^2_{\mathrm{init}}$, E^0, E^1, E^2, $\mathrm{E}_{\mathrm{term}}$. Between the length ℓ of an exon (single, initial, internal, terminal) and the length r of the emission of the corresponding exon state is a simple correspondence. The two lengths differ by a relatively small constant number of nucleotides: some coding bases are emitted from a splice site state and some non-coding bases are emitted in an exon state (at the translation start point). Let $\delta \in \mathbb{Z}$ be that length difference such that $\ell = r + \delta$. We also need to consider the reading frame of the current exon state and the previous state to ensure that the exon lengths match the reading frames. Let q be one of the eight states mentioned above. Let f_2 be the reading frame (superscript) of the current state q; if $q = \mathrm{E}_{\mathrm{single}}$ or $q = \mathrm{E}_{\mathrm{term}}$ let $f_2 = 0$. Let f_1 be the reading frame of the previous state in the process. If the previous state is IR, let f_1 be 0. Let $(\pi_i)_{i\in\mathbb{Z}}$ be the estimated length distribution of the exon type corresponding to q ($\pi_i = 0$ for $i \leq 0$). Then we define the length distribution of state q as follows.

$$\mathrm{P}(|Y| = \ell - \delta) := k \cdot \begin{cases} \pi_\ell & \text{if } f_1 + \ell \equiv f_2 \mod 3 \text{ and } \ell - \delta \geq 1; \\ 0 & \text{otherwise.} \end{cases} \qquad (l \in \mathbb{Z}) \qquad (3.6)$$

Here, k is a norming constant, in practice close to 3, ensuring that the distribution sums up to 1: $\mathrm{P}(|Y| \geq 1) = 1$. Equation (3.6) ensures that the length of an exon is such that the reading frame position at the end of the exon is correct.

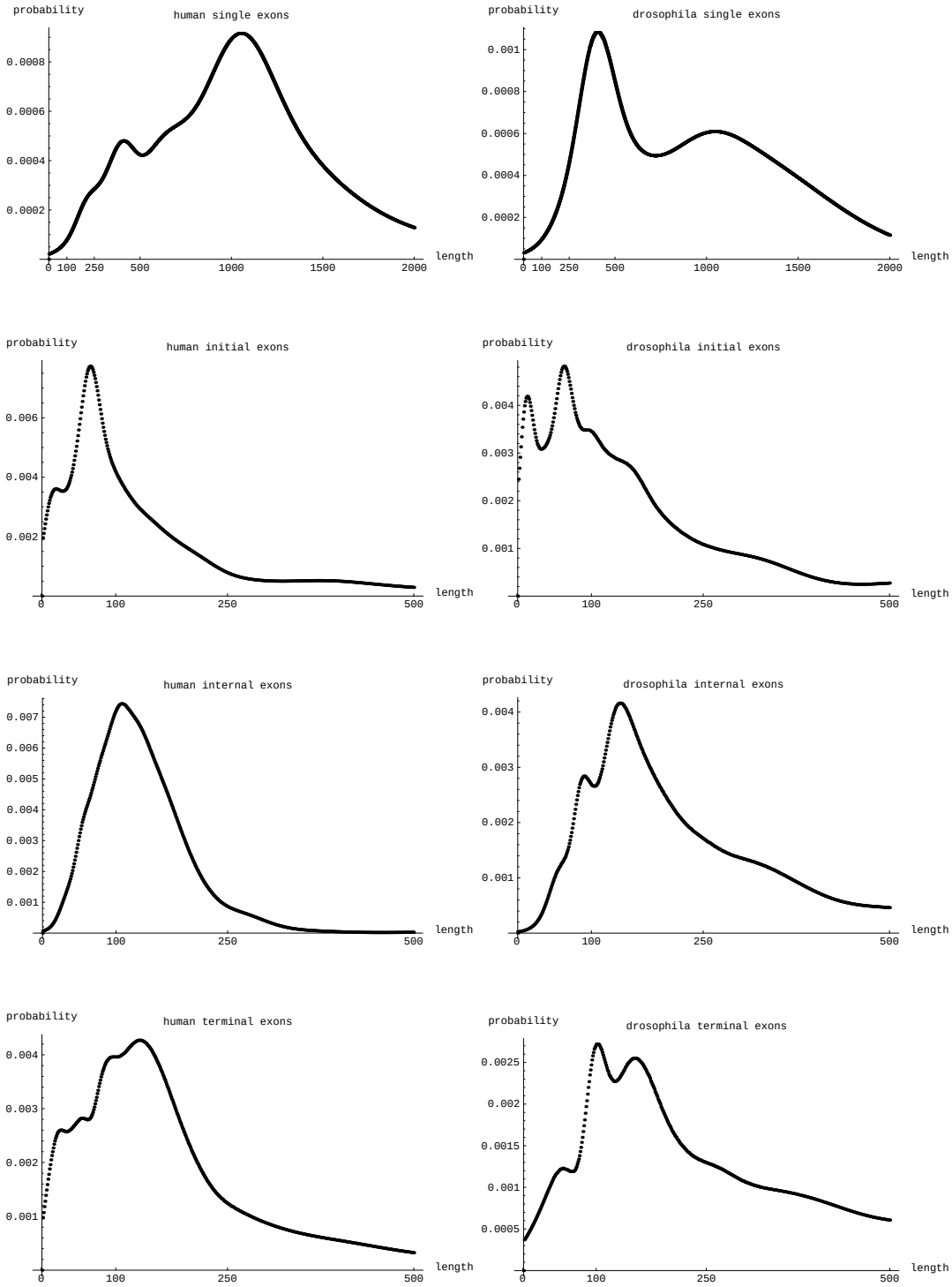

Figure 3.2: Exon length distributions. The distributions for humans (left) and *Drosophila* (right) often have the same modes with different intensities. The number n of exons used for estimation of the length distributions are. human: single, initial, internal, terminal: $n = 462, n = 822, n = 4334, n = 822$, respectively; *Drosophila*: single, initial, internal, terminal: $n = 76, n = 324, n = 917, n = 324$, respectively.

Intron length distribution

A short fixed-length initial part of a biological intron on the forward strand is emitted from state DSS^j for some j and a terminal part of fixed length is emitted from state ASS^j. The middle part of the intron sequence is emitted either in one step from state $\mathrm{I}^j_{\mathrm{short}}$ or in several steps from states $\mathrm{I}^j_{\mathrm{fixed}}$ and $\mathrm{I}^j_{\mathrm{geo}}$. The length distribution of the biological introns is determined by the length distributions of the intron states $\mathrm{I}^j_{\mathrm{short}}, \mathrm{I}^j_{\mathrm{fixed}}$ and $\mathrm{I}^j_{\mathrm{geo}}$ and the transition probability $a_{\mathrm{I}^j_{\mathrm{geo}},\mathrm{I}^j_{\mathrm{geo}}}$. The corresponding applies to introns on the reverse strand. The reason for introducing these three intron states instead of just one single state is explained next.

The geometric approximation. Generalized Hidden Markov Models for gene prediction have one or more states modeling a biological intron. The states of such a model can have an explicit length distribution of the sequence emitted in this state, or the length can be implicitly modeled by emitting just one nucleotide at a time but allowing to transition back to the same state. States with an explicit length distribution allow an accurate modeling of the length at the cost of computation time. If no further heuristic is used the computation time of the typical algorithms (Viterbi, forward algorithm) is at least proportional to the maximal possible length of this state. Introns can be very long: the human neurexin-3 gene on chromosome 14 has an intron of length 479 Kb [WPY01]. It is therefore practically infeasible to explicitly model the whole length distribution in a GHMM. The method of using a state which emits just one nucleotide and allowing transitions back to the state is computationally efficient. The algorithms only require constant time for each position of the sequence for this state. But this option limits the length distribution of introns to a 'shifted' geometric distribution which assigns length $\ell > \delta$ the probability $q(1-q)^{\ell-1-\delta}$ with parameters $0 < q < 1$ and an integer δ. δ would be the length of those parts of an intron which are modeled in other states such as possibly the splice sites. For example, the GHMM-based gene prediction programs GENSCAN, GENIE, TWINSCAN and DOUBLESCAN all use a model in which the introns have a shifted geometric length distribution.

The solid line in the 4 graphs of Figure 3.3 shows the smoothed length distribution of human and *Drosophila* introns of our training sets. This distribution was retrieved with the same method as the length distributions of the exons. We used the constants $a = 0.4$ and $m = 3$ in Formula (3.5). The two lower graphs have both axes on logarithmic scale so that the length distribution for large lengths can be visualized. The mean intron lengths are 1187 (human) and 896 (*Drosophila*). The figures also show the geometric length distribution (short dashes) with the parameter estimated by the maximum likelihood method: $P(L = l) = q(1-q)^{l-1}$, with $q = 1/1187$ (human) and $q = 1/896$ (*Drosophila*).
The graphs show two shortcomings of the geometric distribution as a model for intron lengths. One problem is that a (shifted) geometric distribution always assigns the highest

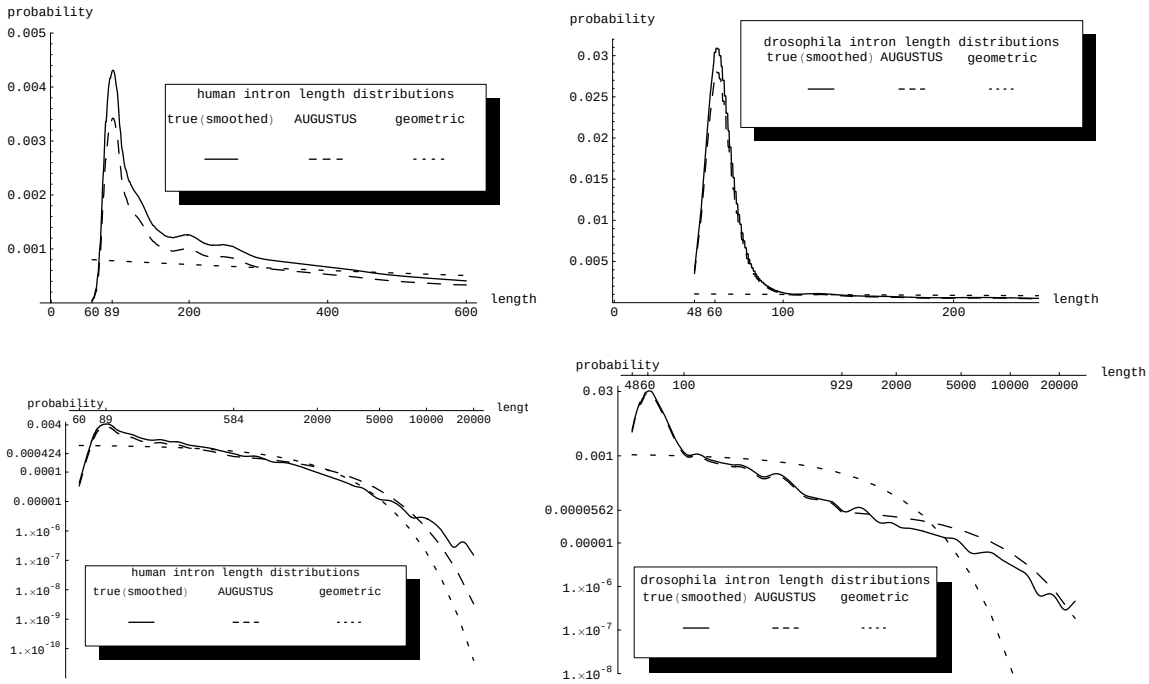

Figure 3.3: Intron length distributions. The two graphs on the left are for human introns, the ones on the right are for *Drosophila*. Top: The smoothed length distribution of the introns in the respective training set, the length distribution of introns of the AUGUSTUS model and the geometric distribution. The geometric distribution is a bad approximation for short introns. The geometric approximation is better in the human case than in the *Drosophila* case. 26% of the human introns were shorter than 200 nucleotides. 63% of the *Drosophila* introns are shorter than 100 nucleotides. Bottom: The same curves as above but with both axes on logarithmic scale. Up to $d = 584$ (human) and $d = 929$ (*Drosophila*) AUGUSTUS uses approximately the 'true' length distribution, the tail of AUGUSTUS' distribution is geometric, too. About 46% of the human introns are longer than 584 nucleotides. About 13% of the *Drosophila* introns are longer than 929 nucleotides.

probability to the shortest possible length. But in our *Drosophila* test set the shortest intron had length 48 and there were 12 introns with a length between 48 and 52 but there were 223 introns with a length between 58 and 62. A program that uses the geometric intron distribution must either allow no such short introns or must assign a higher probability to their length than it assigns to any larger length. Human intron lengths peak around a length of 89, but this peak is much less pronounced than that of *Drosophila*. The other problem of a geometric distribution is that, when q is realistically chosen, long introns become much less likely than they really are. Again this is more obvious for *Drosophila*. Reese et al. explain the fact that many long *Drosophila* introns are not recognized by their program GENIE as follows "...the length distribution of introns, a geometric distribution that favors short introns, is the reason for so many split genes" [RKTH00]. Brejova and

Vinar have published a new dynamic programming algorithm to incorporate into a Hidden Markov Model length distributions with geometric tail [BV02]. We present a simple method for achieving the same goal that allows us to keep the algorithms of Chapter 2.

A new way of modeling the length distribution. We combine states with and without explicit length distribution in order to model an *initial part* of length d of the length distribution more accurately and the remaining part with a geometric distribution. This makes the resulting implicit length distribution much more accurate while at the same time not losing too much computational efficiency. We use the model shown in Figure 3.4 for introns.

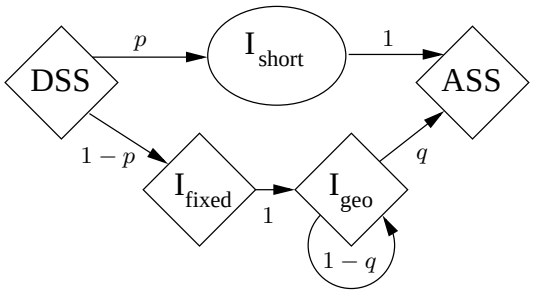

Figure 3.4: AUGUSTUS' intron model. The arrows denote possible transitions and are marked with the transition probabilities.

In this paragraph let the term *inner intron* refer to the part of the intron not modeled in the splice sites, i.e. the intron with the two relatively short fixed-length pieces cut off at the ends. Assume L is the length of a random inner intron, estimated with a kernel estimator as described above. Let M be the random length of an inner intron generated by our model in Figure 3.4. The state I_{short} has an explicit length distribution with maximal length d, namely length ℓ with $0 \leq \ell \leq d$ has probability $P(L = \ell)/P(L \leq \ell)$. The state I_{fixed} emits a string of fixed length d and the state I_{geo} emits just one nucleotide but implicitly has a geometric length distribution with parameter q. There is a one-to-one correspondence between intron lengths and paths from DSS to ASS. If the intron has length at most d the corresponding path goes through I_{short} and if it has length $l > d$ the path goes first through I_{fixed}, then $l - d$ times to state I_{geo} and then leaves I_{fixed} to ASS. The distribution of M is as follows. For $\ell \leq d$ we have $P(M = \ell) = pP(L = \ell)/P(L \leq \ell)$ (= transition probability to I_{short} times length probability). For $\ell > d$ we have $P(M = l) = (1-p)(1-q)^{\ell-d-1}q$ (= product of all transition probabilities). Now q, p and d are still free parameters. We set q such that the expectation of M, given $M > d$, is the expectation of L, given $L > d$, i.e. $d + 1/q = E[L \mid L > d]$. Then we set p such that $P(M = d + 1) = P(M = d)$ and there is no jump in the distribution of M between positions d and $d+1$. Then it remains to choose the parameter d which is a trade-off between accuracy (large d) and speed (small d). We

choose d to be smallest such that $p \approx P(L \leq d)$. We get $q \approx 1/4894, p \approx 0.78, d = 929$ for *Drosophila* and $q \approx 1/1688, p \approx 0.43, d = 584$ for humans. The running time of AUGUSTUS is about 6 minutes for the 1.6 mega bases of the *Drosophila* test set on a PC with 2.4 GHz.

This model architecture also allows to use different content models for long and for short introns. Also additional splice site models could be integrated into the short intron model so that different splice site models could apply to short and long introns. This is suggested by the assumption that the splicing process is typically different for long and short introns [LB01]. The resulting model would also allow to take into account dependencies between the donor and acceptor splice sites of short introns.

Sequence distributions

In this section we specify for all states the conditional distribution of an emission given its length, which we call the sequence distribution. Let $Y^{i,j,\tau}$ be a random emitted string in current state j with previous state i and previous emission τ, i.e. the random string $Y^{i,j,\tau}$ has the distribution given by $e_{i,j,\tau}$. In the previous section we defined the distribution of $|Y^{i,j,\tau}|$, the length of the emission, which depends for some current states j on the previous state i in order to account for the reading frame. We now define the conditional distribution of $Y^{i,j,\tau}$ given the events $|Y^{i,j,\tau}| = \ell \quad (\ell = 1, 2 \ldots)$. Then the distribution of $Y^{i,j,\tau}$ is fully defined.

In AUGUSTUS the conditional distribution of $Y^{i,j,\tau}$ given the event $|Y^{i,j,\tau}| = \ell$ does not depend on the previous state i anymore ($\ell = 1, 2 \ldots$): For all $j \in Q^+, \tau \in \Sigma^*, \ell \geq 1$, and $\sigma \in \Sigma^+$ with $|\sigma| = \ell$

$$\mathrm{P}(Y^{i,j,\tau} = \sigma \,|\, |Y^{i,j,\tau}| = \ell) \text{ is equal for all } i \in Q^+ \text{ such that } \mathrm{P}(|Y^{i,j,\tau}| = \ell) > 0$$

Informally spoken, the distribution of the emission depends – except for its length – only on the current state and the emission so far. Thus we only have to specify the sequence distribution once for each state j and not for combinations of j and i. The distribution will also depend, if at all, only on the last few bases of τ (when a Markov chain is used for the distribution of state j, e.g. in the state for intergenic region).

In the following we describe the sequence distribution for each state. Each state uses one or more simple stochastic submodels for different parts of the sequence. Figure 3.5 shows these parts and the underlying submodels for humans. Table 3.1 shows the submodels each state uses.

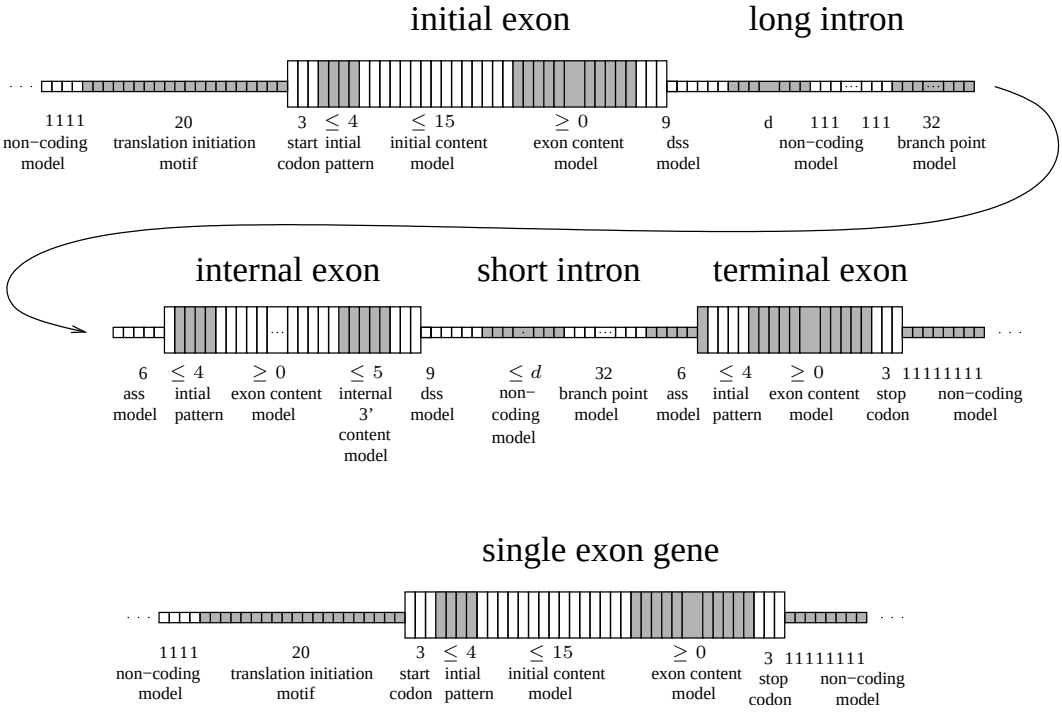

Figure 3.5: Example of a gene with 3 coding exons (above) and an intron-less gene (below). Certain parts of the DNA sequence are modeled using certain submodels. Below each part is written its length in the human version of the program and the name of its submodel.

state	submodels
E_{single}	translation initiation motif, start codon, initial pattern, initial content model, exon content model, stop codon
E^j_{init}	translation initiation motif, start codon, initial pattern, initial content model, exon content model
E^j	initial pattern, exon content model, internal 3' content model
E^j_{term}	initial pattern, exon content model, stop codon
IR	non-coding model
DSS^j	dss model
ASS^j	branch point model, ass model
I^j_{short}	non-coding model
I^j_{fixed}	non-coding model
I^j_{geo}	non-coding model

Table 3.1: For each forward state the submodels used to define the sequence distribution are shown in order. For the reverse strand states the order of the submodels is reversed.

The submodels of the exon states may have a variable length indicated in Figure 3.5.

If the length ℓ of a sequence emitted in such a state is large enough, then the sequence lengths covered by the submodels with bounded lengths are chosen maximal and the exon content model has the remaining length. Example: If the sequence length of a (human) initial exon state is $\ell = 100$ the first 20 bases are emitted using the translation initiation motif, the next 3 bases are emitted using the start codon model, the next 4 using the initial pattern model, the next 15 using the initial content model and the remaining 58 bases are modeled using the exon content model. If ℓ is not long enough for all submodels to have maximal length then some submodels model only shorter sequences. Then the exon content model is not used (length 0) and among the other submodels with variable length the priority goes from left to right, meaning that from left to right the submodels are used for as many bases as possible until all bases of the given length are emitted. Example: If the sequence length of a (human) initial exon state is $\ell = 25$ the first 20 bases are emitted using the translation initiation motif, the next 3 bases are emitted using the start codon model, and the remaining 2 bases by the initial pattern model.

In order to define the sequence distribution for each submodel we made use of established models such as Markov chains, a higher order windowed weight array model (WWAM) ([BK97]), interpolated Markov Models (IMM) and introduced a simple method we call similarity-based weighting of sequence patterns.

A *WWAM of order k and of window size* $2r + 1$ is an inhomogeneous Markov chain $X_1, X_2, \ldots$ of order k on the state space $\{A, C, G, T\}$ in which the probability of observing nucleotide x_i at position i given that the preceding k nucleotides are $x_{i-k}, \ldots, x_{i-1}$ is estimated by the relative frequency of observing x_i after nucleotides $x_{i-k}, \ldots, x_{i-1}$ in the training data at one of the positions in the window $i - r, \ldots, i + r$. For that purpose the training data is aligned with respect to the biological signal that is modeled. Formally, let $s_{p,q}$ be the nucleotide in the p-th row and the q-th column of the alignment of the training sequences. Then

$$\begin{aligned} & P(X_i = x_i \,|\, X_{i-k} = x_{i-k}, \ldots, X_{i-1} = x_{i-1}) \\ := \quad & \frac{|\{(p,q) \,|\, i - r \leq q \leq i + r, s_{p,q} = x_i, s_{p,q-k} = x_{i-k}, \ldots, s_{p,q-1} = x_{i-1}\}|}{|\{(p,q) \,|\, i - r \leq q \leq i + r, s_{p,q-k} = x_{i-k}, \ldots, s_{p,q-1} = x_{i-1}\}|} \end{aligned}$$

By an *interpolated Markov Model (IMM) of order $k \geq 2$* we denote a Markov chain $X_1, X_2, \ldots$ of order k on the state space $\{A, C, G, T\}$, in which for some sequence patterns $x_{i-k}, \ldots, x_{i-1}$, the probability of observing nucleotide x_i after that pattern was estimated as if the chain was of only of order $k-1$. An IMM can be both homogenous and inhomogeneous. We here use a special case of the IMM described in [SDSO98], in which only the transition probabilities of orders k and $k - 1$ are considered and the respective interpolation weights are either 0 or 1. The idea behind this model is that it is usually better to use a higher order Markov chain as long as the amount of training data allows a reliable estimation of the parameters. However, in practice, the amount of available

data for estimating an individual transition probability out of a pattern depends strongly on that pattern. The conditional probability of observing nucleotide x_i after nucleotides $x_{i-k}, \ldots, x_{i-1}$ is

$$\begin{aligned} & P(X_i = x_i \mid X_{i-k} = x_{i-k}, \ldots, X_{i-1} = x_{i-1}) \\ = & \begin{cases} \frac{\#(x_{i-k},\ldots,x_i)}{\#(x_{i-k},\ldots,x_{i-1})} & \text{if } \#(x_{i-k}, \ldots, x_{i-1}) \geq 400; \\ \frac{\#(x_{i-k+1},\ldots,x_i)}{\#(x_{i-k+1},\ldots,x_{i-1})} & \text{otherwise.} \end{cases} \end{aligned}$$

Here the character # in front of a pattern denotes the frequency of the pattern in the training sequences (in the appropriate reading frame if applicable). The sequences were each weighted with an integer factor between 1 and 10 according to their GC content. The threshold 400 was chosen empirically. We added a pseudo count of 5 to all pattern frequencies of $k+1$-mers. (Or, equivalently, we added 5 sequences which each contain all $k+1$-mers exactly once to the training set.)

The submodels are:

translation initiation motif: WWAM of order 3 and window size 5 for the 20 bases before the translation start.

start codon: Emit ATG with probability 1.

initial pattern: Emit pattern p of length at most 4 with the probability given by the relative frequency of this pattern in the corresponding reading frame among all coding sequences of the training set. The pattern has length 4 unless the exon length allows only shorter patterns. The reason for introducing this submodel is a technical one. If it was left out then the probability of the first bases after the start codon or after the acceptor splice site would be directly determined with a Markov model and therefore the nucleotides of the start codon or splice site would determine the emission probabilities of the following bases. But the start codon and splice site bases are always or often the same and an exception as far as typical coding sequences are concerned.

initial content model: interpolated 3-periodic Markov model of order 4. The length of the emitted sequence is 15 if the exon length allows it. The model is trained on the corresponding 15 nucleotides of single and initial exons of the training set. We also tried a corresponding terminal content model in the region around the stop codon as this was suggested by a bias in the distribution of these bases compared to the models we actually use. However, this model did not yield any improvement.

exon content model: interpolated Markov model of order 4 trained on all corresponding coding sequences of the training set. Only in earlier stages of AUGUSTUS, when we used fewer submodels, the order 5, which is commonly used in other programs, yielded better results.

DSS model: We only consider canonical splice sites obeying the GT-AG rule as this rule accounts for about 99% of mammalian splice sites [BSS00]. The donor splice site

model emits the 3 last nucleotides of the exon, then the consensus dinucleotide GT, and 4 more nucleotides of the intron (*Drosophila*: 2 before, 4 after GT). For the distribution of the 7 free nucleotides we use a model we call *similarity-based sequence weighting*. The method of similarity-based weighting of sequence patterns is as follows. Given a fixed sequence pattern size, training patterns $q_1, \ldots, q_m$ and a similarity scoring function s, weighting pairs of patterns, we estimate the probability that a random pattern equals a given pattern q as

$$p(q) = c \sum_{i=1}^{m} s(q, q_i),$$

where c is chosen so that the sum of all $p(q)$ is 1. The choice of s depends on the particular purpose. For the donor splice site we use

$$s(r,q) = \begin{cases} 1 & \text{if } r = q; \\ 0.001 & \text{if } r \text{ and } q \text{ differ at exactly one position;} \\ 0 & \text{otherwise.} \end{cases}$$

This way, sequences obtained by a single point mutation from a typical splice site get a bonus in comparison with the empirical distribution. The resulting distribution is the discretely smoothed empirical distribution. When sufficiently much data is available for estimation, the empirical distribution respects the complicated statistical dependencies that exist between the nucleotide positions.

non-coding model : Markov model of order 4 trained on all non-coding sequences of the training set.

branch point model: WWAM of order 3 and window size 7 emitting 32 nucleotides.

ass model: The acceptor splice site model emits 3 nucleotides of the intron before the AG dinucleotide consensus, then AG and the first nucleotide of the exon. A pattern of the 4 free nucleotides gets as probability the relative frequency of these 4 nucleotides at the corresponding positions in the training set.

internal 3' content model: interpolated 3-periodic Markov model of order 4 trained on the 5 nucleotides at positions -8 to -4 with respect to the donor splice site using all internal exons in the training set. Observe that this model, which helps locating the donor splice site, makes use of the reading frame of the coding nucleotides as opposed to the DSS model for nucleotides -3,-2 and -1. This model is not used for *Drosophila*.

stop codon: Emit TAG, TGA or TAA with probabilities 24%, 48% and 28%, respectively.

3.3 Statistics of Selected Submodels

In this section three of the submodels of AUGUSTUS are described.

Initial Content Motif

The first coding nucleotides of a gene may have a different distribution than the overall average of all coding sequences. This has been exploited before by the author of GENSCAN. We have compared the amino acid frequencies of the coding sequences with the amino acid frequencies of the first 5 codons of a genes using the human and a *Drosophila* training set. Let N_i be the frequency of amino acid i ($i = 1, \ldots, 20$) in the protein sequences of the training set. And let n_i be the relative frequency of amino acid i among the 4th through 8th amino acid of the protein sequences of the training set, i.e. the first 5 amino acids after the start codon and the initial pattern model. The relative amino acid frequencies and the relative synonymous codon frequencies in the human training set are shown in Table 7 on page 94. To determine whether the observed difference in the two distributions can have occurred by chance we carried out a χ^2-test. Let $N := N_1 + \cdots N_{20}$ be the overall number of amino acids in the training sequence and let $n := n_1 + \cdots n_{20}$ be the number of amino acids among the 3rd through 7th of a gene. As N is very large (human: $N = 510980$, fly:$N = 239744$) compared to n (human:$n = 6420$, fly:$n = 2000$) we assume that $p_i := N_i/N$ is a sufficient approximation to the overall probability of amino acid i. The χ^2-test checks whether the sample $(n_1, \ldots, n_{20})$ can have a multinomial distribution with parameters $p_1, \ldots, p_{20}$ by checking whether the statics

$$\chi^2 = \sum_{i=1}^{20} \frac{(np_i - n_i)^2}{np_i}$$

is significantly large. In both cases – human and fly – given the level of significance $\alpha = 0.01$ we can reject the null hypothesis that the frequencies of the first 5 amino acids have the same distribution as that of all amino acids. (p-values: $p < 10^{-10}$ (human), $p \approx 0.0003$ (fly)). We introduced the initial content motif because of this result and the fact that the introduction of this submodel improves somewhat the accuracy of the predictions (see Table 7.6). Remark: The codon usage at the terminal end of the amino acid sequence was also found to differ statistically significantly from the overall average (data not shown) but this fact could not be exploited to improve the prediction accuracy.

Branch Point Model

The branch point (branch site) is the position of a nucleotide adenine 10 to 50 base pairs ([Zha98], [SGH$^+$98]) upstream of the 3' end of an intron which interacts during the splicing process with the guanine at the 5' end of the intron. The consensus around the branch point is weak, and no reliable computational method to identify its location is known. Figure 3.6 shows a pictogram of the branch point region of human and *Drosophila*, respectively. The 32 bases covered by the branch point model of AUGUSTUS are shown. These are positions -37 to -6 relative to the acceptor splice site position (the rightmost position of the

intron has position -1). The graphs show that the base composition continuously changes within this region. For example, in both species there is a tendency of observing thymine from left to right in increasing frequency. What the pictograms not show is a special sequence composition which helps identifying the branch point.

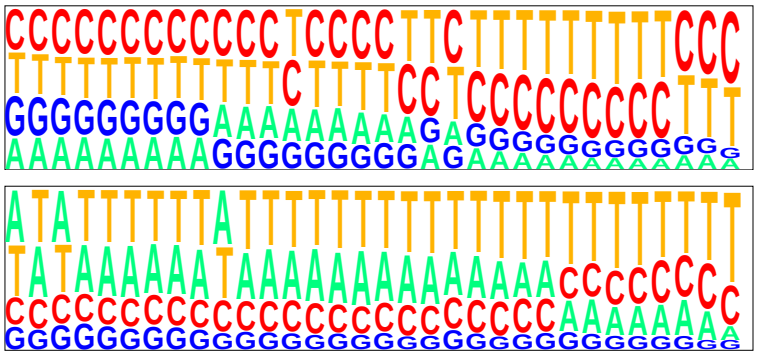

Figure 3.6: The relative frequencies of the bases at positions -37 to -6 relative to the acceptor splice site position in human (above) and *Drosophila* (below). The size of the letters is proportional to the frequency of the base and bases are ordered according to this frequency from top to bottom. The high frequency of pyrimidines (T and C) close to the 3' splice site corresponds to the *polypyrimidine tract* known to play an important role in human pre-mRNA splicing [CSP97]. This graph was created using Pictogram (http://genes.mit.edu/pictogram.html) by Chris Burge.

As the distance of the branch point to the acceptor splice site is variable, a position specific weight matrix, as is visualized by the pictogram, is a bad means to find a consensus sequence of a signal. We counted for each species, for each of the 175 4-mers containing at least one A and for each of the positions in the range of the branch point model the number of occurrences of the 4-mer starting at the position in the respective set of training sequences. The largest frequency had pattern CTGA at position -23 (human) and pattern TAAT at position -19 (*Drosophila*). These patterns are consistent with the consensus given in [LB01]. Figure 3.7 shows the distribution of these two patterns in this range. The patterns are much more often observed around the mode of the distribution than elsewhere and, indeed, the position of the pattern – if present at all – varies from intron to intron. The same is true for other frequently observed patterns (data not shown). The WWAM of order 3 and window size 7 that we chose to model this region accounts for both the approximate distribution of positions of patterns of size 4 and the continuous change in base composition. Again, the order and window size determine the balance between modeling true details of the distribution (large order and small window size) on the one hand and overfitting on the other hand and were chosen to maximize prediction accuracy.

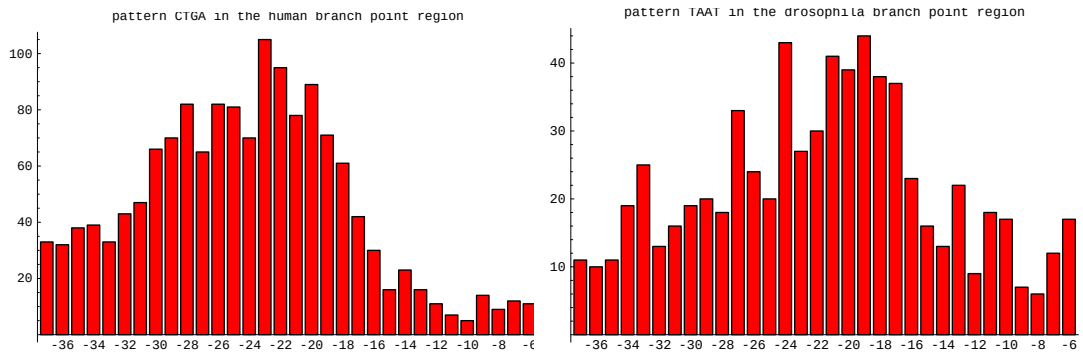

Figure 3.7: Distribution of the positions where sequence patterns CTGA (left, human) and TAAT (right, *Drosophila*) begin. The horizontal axis shows the position relative to the 3' end of the intron. The vertical axis shows the absolute frequency in the training set of AUGUSTUS.

Internal 3' Content Model

The internal 3' content model is a model only used in the human version of AUGUSTUS, for the 5 bases at positions -8 to -4 relative to the donor splice site (if -1 is the last position of the exon). This model directly precedes the donor splice site model but can be thought of as a part of the donor splice site signal model as it helps locating the splicing position. The left side of Table 3.2 shows the relative frequency of the four nucleotides in this range of 5 bases in the human training set. The nucleotides are broken down by their position in the codon. A clear dependency on the reading frame is visible as is the case with the overall nucleotide distribution in the coding regions shown in the right matrix. For example in each of the two models the C is more frequent in the third codon position than in the first two codon positions. Thus, as anticipated from the fact that these bases are coding, a probabilistic model for these 5 bases in the $[-8,-4]$-window should take the reading frame into account. On the other hand the distribution of the nucleotides in this $[-8,-4]$-window is different from the distribution in the coding regions corresponding to the exon content model: Three chi-square tests checking for each reading frame whether the number of A, C, G and T's in the $[-8,-4]$-window could be distributed as in the exon content model on the right of Table 3.2 showed a significant deviation (p-values were between 10^{-13} and 10^{-16}). This is not much of a surprise, either. Some stretch of sequence directly upstream of the donor splice site has two functions at the same time. It is both coding and involved in splicing. The internal 3' content model is a 3-periodic Markov chain of order 4. As such it considers the distribution of sequence patterns of length 5 ending in the $[-8,-4]$-window and also the phase of the exon.

	internal 3' content model		
	$f=0$	$f=1$	$f=2$
A	0.291	0.317	0.166
C	0.252	0.213	0.357
G	0.291	0.184	0.275
T	0.165	0.286	0.202

	exon content model		
	$f=0$	$f=1$	$f=2$
A	0.248	0.291	0.146
C	0.264	0.243	0.351
G	0.321	0.201	0.312
T	0.166	0.265	0.19

Table 3.2: Nucleotide frequencies at the three frame positions. $f=0, f=1, f=2$: first, second or third base of a codon. The left side table shows the distribution for the window $[-8,-4]$ relative to the donor splice site covered by the internal 3' content model. The right side shows the distribution for the bases modeled with the exon content model.

3.4 GC Content Dependent Training

The GC content of a sequence is the sum of the relative frequencies of the bases G and C in the sequence. The composition of the bases varies along the human genome but there are compositionally fairly homogenous regions of size at least 200 Kb, called 'isochores'. As Burge has pointed out in his thesis, many parameters of a gene prediction model strongly correlate with GC content. Burge took this into account by using 4 classes of GC content ($<43\%, 43\%-51\%, 51\%-57\%, >57\%$) and building 4 different parameter sets by estimating most of the parameters only from sequences of the class. For a given input sequence the model parameters of its GC class are more appropriate but were estimated using only about one fourth of the training sequences. But for a reliable estimation of the parameters a large enough training set is important.

We use a method that allows us to use even more GC content specific parameter sets without the disadvantage of decreasing the training set size. We generate 10 different parameter sets for different GC contents of the input sequence. However, for constructing each of these parameter sets we use *all* sequences of the training set. In constructing a parameter set for mean GC content α we weighed each sequence of the training set with an integer weight $1 \leq w \leq 10$ depending on its GC content β. Similar GC contents get a higher weight. As a weight function we used

$$w(\alpha,\beta) = \lceil 10\exp(-200(\alpha-\beta)^2)\rceil \quad (0<\alpha,\beta<1)$$

($\lceil \ldots \rceil$ means rounding up). We then trained the parameters as if each sequence was w times in the training set (Burge's method can be regarded as a special case in which $w(\alpha,\beta)=1$ if α and β are in the same class and $w=0$, otherwise). For the prediction AUGUSTUS chooses for each input sequence the parameter set with mean GC content closest to the one of the input sequence. The length distributions, transition probabilities

and splice site models were trained independently of GC content. For all other submodels there are 10 different versions.

3.5 Variants of the Model

The herein before mentioned model allows the prediction of any number of possibly partial genes on both strands. No genes may overlap not even on opposite strands. But sometimes a user may have additional information on the gene structure in the input sequence. For instance, he may know from experiments that the sequence contains at least one gene. Or he may want to find a gene within an intron of another gene and thus cannot assume that genes do not overlap. For those purposes the model AUGUSTUS has variants which are options to the program AUGUSTUS.

- **"only predict complete genes"**
  By changing the transition probabilities such that

  $$a_{q_{\text{init}},\text{IR}} = 1 \text{ and } a_{q,q_{\text{term}}} > 0 \text{ only for } q = \text{IR}$$

  we achieve that only parses which start and end in the intergenic region are possible. Therefore no partial genes are predicted. Remark: The program GENIE, which also bases on a GHMM, predicts no partial genes.

- **"only predict complete genes – at least one"**
  In this variant we introduce a second state IR$'$ for the intergenic region which has the same emission distribution as IR. The original IR then stands for the intergenic region *before the first gene* and IR$'$ stands for all subsequent intergenic regions. A parse is forced to start in IR and end in IR$'$. Once left, state IR is never entered again and IR$'$ takes the role of the intergenic region. The transition probabilities are changed as follows. Let $A' = (a'_{i,j})_{i,j \in Q^+}$ be a new transition matrix. A' is equal to A except for the following entries: $a'_{q_{\text{init}},\text{IR}} = 1$, $a'_{q_{\text{init}},q} = 0 \quad (q \neq \text{IR})$, $a'_{q,\text{IR}'} = a_{q,\text{IR}} \quad (q \notin \{q_{\text{init}}, \text{IR}\})$, $a'_{\text{IR}',\text{IR}'} = a_{\text{IR},\text{IR}}$, $a'_{q,\text{IR}} = 0 \quad (q \notin \{q_{\text{init}}, \text{IR}\})$, $a'_{\text{IR}',q} = a_{\text{IR},q} \quad (q \notin \{q_{\text{term}}, \text{IR}'\})$, $a'_{\text{IR}',q_{\text{term}}} = \epsilon$ The transition matrix used in this variant is A' renormalized so that the sum of the transition probabilities out of each state is 1. (This condition is only minimally violated by A'.)

- **"predict exactly one gene"**
  In this variant AUGUSTUS predicts exactly one complete gene (if possible at all). The transition matrix is the same as in the previous option with one change. The only transitions allowed from state IR$'$ are into IR$'$ itself and to the terminal state. This way all parses enter IR first, return to IR for some time, enter some 'gene'

states corresponding to one gene and enter the IR′, from which the terminal state is reached.

- **"ignore conflicts with other strand"**
  The standard model has been constructed to avoid a common error when predicting genes on the two strands independently. A gene prediction program that searched genes only on one strand, disregarding the opposite strand, tends to find 'shadow genes' in the vicinity of real genes on the opposite strand. This is because the Markov chain model for the coding regions often yields also a relatively high probability for the reverse complement of a real coding region. In order to avoid this error AUGUSTUS uses a technique proposed previously [BM93]: Using a competing model for the reverse strand. This has also been implemented in GENSCAN where they introduced 'shadow' states for the genes on the reverse strand. This architecture ensures that predicted genes do not overlap, even on different strands. Usually the true gene fits the model better than its 'shadow' and therefore the parse corresponding to the true gene has higher probability than the competing parse corresponding to the shadow gene. Although this usually increases accuracy there may be cases when one wants to ignore those conflicts of a gene structure with a gene structure on the opposite strand; for example, in the case of a gene within an intron on the opposite strand. With this option set, AUGUSTUS uses none of the reverse states. Only the 24 states of the upper half of Figure 3.1 including state IR are used. Then the genes on the forward strand are predicted. The genes on the reverse strand are predicted using the same model on the reverse complement of the input sequence and reversing the prediction afterwards. The prediction output by AUGUSTUS simply is the union of the predicted genes on both strands, which then may contain overlapping genes.

  While the previous three options and the standard option to predict any number of (partial) genes exclude each other mutually, this option can be combined with the others.

By default the program AUGUSTUS reports genes on both strands but it may also be asked to report only the genes on one of the strands. Then the predicted genes on the other strand are simply filtered out after the algorithms have been run.
Some of the test sets used for determining the accuracy of gene prediction programs contain exactly one gene per sequence, as for example the test set h178 described in section 7.1. When one of the three options where AUGUSTUS predicts only complete genes is set the accuracy of AUGUSTUS' predictions increases significantly on the test set h178. The sensitivity on the gene level (see 7.3 for the definition) increases from 48% to 55%-58%. The explanation for this is as follows. AUGUSTUS may predict a complete gene partially

by missing for example the first exon only. With one of these options set AUGUSTUS is 'forced' to find an initial exon if the other exons fit the model well. Although these options elevate prediction accuracy we did not use them in the evaluation (section 7.3) because we think that the setting, where nothing about the gene structure of the input sequence is known, is more typical for applications.

Chapter 4

Further Analysis

4.1 A-Posteriori Probabilities and Sampling of Gene Structures

In AUGUSTUS there is a one-to-one correspondence between gene structures and parses. We therefore identify a parse with the corresponding gene structure. For an input sequence σ and a parse ψ let

$$p(\psi|\sigma) := P(\varphi(\mathbf{X}, \mathbf{Y}) = \psi \,|\, \sigma(\mathbf{Y}) = \sigma)$$

denote the a-posteriori probability of the parse ψ given σ. And let

$$\Theta(\sigma) := \{\text{parses } \psi \,|\, p(\psi|\sigma) > 0\}$$

be the set of all possible parses of sequence σ, in the sense that they have non-zero a-posteriori probability. In AUGUSTUS $\Theta(\sigma)$ contains all parses corresponding to a possible gene structure of σ in the following, weak sense. The splice sites obey the minimal dinucleotide consensus, genes start with a start codon, end with a stop codon, have no in-frame stop codons within the exons, are reading-frame consistent and the exon and intron lengths exceed a minimal length. Every gene structure that obeys these rules has non-zero a-posteriori probability in AUGUSTUS. In this section we examine some properties of this a-posteriori distribution on the set $\Theta(\sigma)$.

4.1.1 A-Posteriori Probability of the Predicted Gene Structure

The size of $\Theta(\sigma)$, which is the number of possible parses of σ, can be determined with a dynamic programming algorithm similar to the ones described in the second chapter. We determined the number of possible parses for all sequences in the human test set h178. For example the average-length sequence σ with the name HSLCATG of length 6901, has $|\Theta(\sigma)| \approx 2 \cdot 10^{85}$ possible parses. The longest sequence of length 86640 had about $2 \cdot 10^{1063}$

possible parses. As

$$\sum_{\psi \in \Theta(\sigma)} p(\psi|\sigma) = 1,$$

a question that arises is, how equally or unequally the a-posteriori probabilities distribute on this huge set of all possible parses. The predicted gene structure of sequence σ is the one corresponding to the computed Viterbi parse ψ_{vit} and has the largest a-posteriori probability

$$p(\psi_{\text{vit}}|\sigma) = \max\{p(\psi|\sigma)|\psi \in \Theta(\sigma)\}.$$

We computed these probabilities for the 178 sequences σ in h178, which are on average about 7Kb long and contain one gene each.

As Figure 4.1 shows, the largest a-posteriori probabilities are not as small as might have been expected. Half of the $p(\psi_{\text{vit}}|\sigma)$ values are larger than 0.43. Even the smallest value was almost 2 percent. Given the enormous size of the set $\Theta(\sigma)$, one can say that often a relatively large part of the probability mass lies in just one point of the set, the Viterbi parse. The bottom picture of Figure 4.1 also shows that longer sequences have a tendency to have a Viterbi parse with lower a-posteriori probability. This is intuitive as sometimes several likely local alternatives in the gene structure at different positions can be independently combined.

4.1.2 A-Posteriori Probability of the True Gene Structure

Another naturally occurring focus of interest is the a-posteriori probability of the true gene structure. We call the unique parse corresponding to the true gene structure of the input sequence the *induced parse* ψ_{ind}. Of course, the hope is that the Viterbi parse and the induced parse are identical. But if not, one may hope that at least $p(\psi_{\text{ind}}|\sigma)$ is close to $p(\psi_{\text{vit}}|\sigma)$. One advantage of this is that then the true gene structure has a high probability of getting sampled. And the other advantage of $p(\psi_{\text{ind}}|\sigma)$ being close to $p(\psi_{\text{vit}}|\sigma)$ is that then a small change of the model and therefore a change of the a-posteriori distribution of parses might have the effect of turning ψ_{ind} into the Viterbi parse. The same hope applies to the integration of extrinsic information. If the true gene structure is not much less likely than the (false) predicted gene structure, then the change in the a-posteriori distribution, that the integration of the extrinsic information means (see chapter 5), has a better 'chance' of 'making' the prediction for this sequence correct.

If $\psi_{\text{ind}} \in \Theta(\sigma)$ we define

$$q(\sigma) := \frac{p(\psi_{\text{vit}}|\sigma)}{p(\psi_{\text{ind}}|\sigma)}$$

to be the *error quotient* for sequence σ. The error quotient is always at least 1. If $q(\sigma) = 1$ the induced parse is a Viterbi parse which means that the predicted gene structure is correct, provided the Viterbi parse is unique (this seems to be usually the case). The

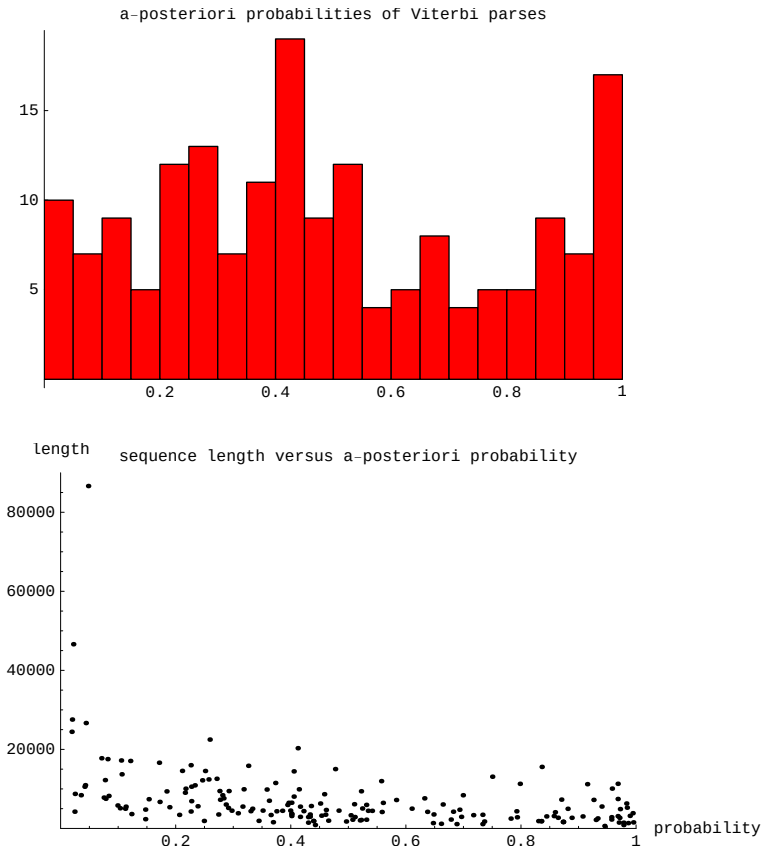

Figure 4.1: Above: A histogram of the 178 a-posteriori probabilities of the predicted gene structures in test set h178. Below: A scatterplot of the pairs (p_i, ℓ_i), where p_i is the a-posteriori probability of the predicted gene structure of the i-th sequence and ℓ_i is the length of the i-th sequence. The correlation coefficient is $\rho \approx -0.40$.

following table shows a summary of the error quotients for the sequences in the test set h178.

error quot.	$q = 1$	$1 < q \leq 2$	$2 < q \leq 10$	$10 < q \leq 1000$	$1000 < q \leq 10^{10}$	$q \geq 10^{10}$
num. seqs	78	13	24	26	28	9

In 78 cases the predicted gene structure was identical to the true gene structure, i.e. the annotated gene of the test sequence was correctly predicted and no other genes or exons were predicted in that sequence. The largest error quotient was $4 \cdot 10^{187}$. This belongs to a 47Kb sequence (HSU34879) with an annotated six exons gene. This gene is actually correctly predicted by AUGUSTUS but also 4 additional non-annotated genes on both strands. We believe the annotation is incomplete in this case because there are also very high scoring blastx hits against the nr database in the range of the additional predicted genes. The second largest error quotient of $5 \cdot 10^{93}$ belongs to a sequence (HSADH6) with a gene with 8 annotated exons, of which all 7 introns have the unrealistic short length 25. The third and fourth largest error quotients of $1.27 \cdot 10^{40}$ and $2.05 \cdot 10^{-31}$,

respectively, belong to annotations (of sequences HSADH6, HSUSF2) that have previously been reported as wrong [YLB01]. Above examples show that the error quotient is possibly useful for detecting wrong annotations.

In order to value the magnitude of the error quotient that can be overcome by a change in the model parameters we made the following experiment. We changed the order of the Markov chain in the exon model from 4 to 5, reestimated the Markov chain parameters and left everything else unchanged. Let $p'(\phi_{\text{ind}}|\sigma)$ be the a-posteriori probability of the true gene structure in this slightly changed model. Let $A^+ = \{\sigma \in \text{ h178 } |p'(\phi_{\text{ind}}|\sigma) > p(\phi_{\text{ind}}|\sigma)\}$ be the set of sequences in h178 where the change increased the a-posteriori probability of the true parse. This was the case for $|A^+| = 66$ sequences. Then the geometric mean of the relative increase in the a-posteriori probability was

$$\left(\prod_{\sigma\in A^+} \frac{p'(\phi_{\text{ind}}|\sigma)}{p(\phi_{\text{ind}}|\sigma)}\right)^{1/|A^+|} \approx 3.54$$

The largest relative increase was $2 \cdot 10^5$, most were below 1000. We have the hypothetical aim of making changes to the AUGUSTUS model or its parameters such that all predictions become correct. From this result we draw the following conclusion. A part of the wrong predictions with smaller error rates (maybe below 1000) can theoretically be corrected by small adept changes to the model parameters. But another part of the wrong predictions with very high error rates (maybe above 1000) can only be corrected by a drastic change of the model.

4.1.3 Sampling Gene Structures

The sampling algorithm is one method to find several parses with high a-posteriori probability without additional memory usage in addition to the forward table. In AUGUSTUS, after the forward table has been computed, 20 runs of the sampling algorithm take about the same time as the Viterbi algorithm. A sampling algorithm has been used by [CP03] in their program SLAM to predict genes in alternative splice forms. We have not examined whether the different sampled parses of AUGUSTUS could correspond to alternative splice forms. Instead we addressed the question whether sampling could help in predicting one given gene structure. Suppose we had a method that could decide for a given small set of at most k possible gene structures which contains the true gene structure, which of the gene structures in the set is correct. Then we could run the sampling algorithm to produce this set of likely gene structures first and then apply the method afterwards.

On test set h178 we tried the following. For each sequence we ran the sampling algorithm 1000 times and took the (at most) k parses of largest a-posteriori probability within the sampled parses. The a-posteriori probability was computed using Theorem 2.6. We

discarded parses that had been sampled before and sorted the parses according to their a-posteriori probability. Let $g(k)$ be the fraction of the 178 test sequences where the set of the k most likely parses contained the induced parse. We got

k	1	2	3	4	5
$g(k)$	43.8%	54.5%	58.4%	62.9%	64.6%

For example one could raise the fraction of correctly predicted parses from 43.8% to 54.5% if one had a method of picking the correct gene structure out of a choice of two.

4.2 Improving State Models Can Worsen the Overall Model

In a GHMM the sequence emitted by a state usually corresponds to a part of the sequence which has some structural meaning. For example, a donor splice site state in AUGUSTUS corresponds to a biological donor splice site sequence. The emission distribution of a state defines a stochastic model itself, which we refer to as the *state model* of that state. For example, the donor splice site state defines a (stochastic) model of donor splice site sequences. In this section we do not consider the case when the emission distribution also actually depends on the previous state and the previous emission. Then the prediction performance of a state model can be separately evaluated. We leave it open at this point how exactly the 'prediction performance' of a state model should be measured. In fact there are many papers separately examining and comparing the prediction accuracy of splice site models (e.g. [PLS01]). There are also papers examining the prediction accuracy of promoter sequences, translation initiation sites ([Hat02]) and coding regions ([FT92]). There also is a paper ([TDZ01]) about a program that predicts only 3'-terminal exons. The intention behind most of these efforts is: One hopes that improving such a model of a part of the gene – in the sense of increasing its isolated prediction accuracy – will also improve the gene model when this model is integrated. The following toy example shows that this intuition can be wrong in the case where a stochastic state model is integrated in a GHMM.

Suppose the actual sequence data was distributed according to the following 'true' GHMM shown in Figure 4.2. The alphabet is $\Sigma = \{0, 1\}$. States E and F, which are equally likely, both emit a binary sequence of length 10. In state E the emission distribution is as follows. With probability 0.75 the 10 letters are independent Bernoulli distributed with probability $p_1 = 0.2$ for letter 1 and $p_0 = 0.8$ for letter 0. And with probability 0.25 the 10 letters are independent Bernoulli-distributed with probability $q_1 = 1$ for letter 1 and $q_0 = 0$ for letter 0. So, either the letter 1 is relatively unlikely at each position in the word or all letters are 1. Formally, for an emission string $\sigma = \sigma_1, \ldots, \sigma_{10}$ of length 10 the emission probability

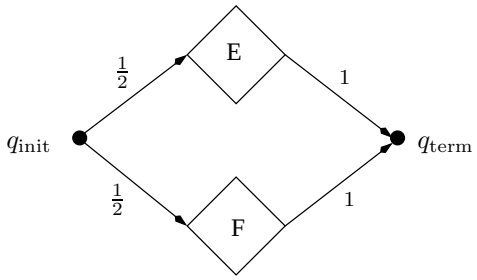

Figure 4.2: A toy GHMM

in state E is

$$e_E(\sigma) := e_{q_{\text{init}},E,\varepsilon}(\sigma) = 0.75 \prod_{i=1}^{10} p_{\sigma_i} + 0.25 \prod_{i=1}^{10} q_{\sigma_i}.$$

In state F emission distribution is independent Bernoulli-distributed with probability $r_1 = 0.1$ for letter 1 and $r_0 = 0.9$ for letter 0.

$$e_F(\sigma) := e_{q_{\text{init}},F,\varepsilon}(\sigma) = \prod_{i=1}^{10} r_{\sigma_i}$$

If an emission σ has been emitted from E we say it is of type E, otherwise it is of type F. Let $X \in \{E, F\}$ be the random type and $Y \in \Sigma^{10}$ be the random emission of this true GHMM. Observe that in this model every observed emission σ can have been emitted in either of the two states. So there is no chance of faultlessly finding out whether σ is of type E or F.

Now, consider the following two variants, named 1 and 2, of this true GHMM. These variants can be thought of attempts to model the true but unknown distribution. The two variants are identical to above true GHMM except for the emission distribution in state E. For state E the emission distribution in the two variants is according to a Markov chain. In variant 1 it is a Markov chain of order 0 (i.e. an independent and identically distributed sequence) and in variant 2 it is a Markov chain of order 1. In both cases the parameters of the Markov chain are as if they were estimated with the maximum likelihood method from an infinite number of training examples of emissions of type E. The probability of letter 1 at any position in an emission of type E is $t_1 := 0.75 \cdot 0.2 + .25 \cdot 1 = 0.4$ and accordingly the probability of letter 0 is $t_0 = 0.6$. The emission probabilities of an emission $\sigma = \sigma_1, \ldots, \sigma_{10}$ in state E in the two variants are

$$e^1_E(\sigma) = \prod_{i=1}^{10} t_{\sigma_i}$$

$$e^2_E(\sigma) = t_{\sigma_1} \prod_{i=2}^{10} m_{\sigma_{i-1},\sigma_i} \quad \text{with transition matrix } \begin{pmatrix} m_{0,0} & m_{0,1} \\ m_{1,0} & m_{1,1} \end{pmatrix} = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}$$

The exponent of e stays for the variant, either 1 or 2. In variant 1 this can be seen as the best approximation of distribution e_E with a 0th order Markov chain, and in variant 2 this can be seen as the best approximation of distribution e_E with a first order Markov chain. But this unproven observation is not necessary for the point made with this example. Observe that the first order Markov chain regards the dependencies between pairs of positions of emissions of type E. For example, given that a certain letter is a 1, the probability that any other letter is also 1 is high (70%) compared with the overall probability of letter 1 (40%).

We claim that – separately seen – the 1st order Markov chain of variant 2 is a better model for sequences of type E than the simpler model of variant 1. The concepts of 'being better' we apply are 1) the Kullback-Leibler distance and 2) the ROC curve.

1) The *Kullback-Leibler distance* (also called relative entropy) measures the predictive accuracy of an estimated distribution by computing a discrepancy value between the true distribution (here e_E) and an estimated distribution (here e_E^1 or e_E^2). Let

$$\mathrm{KL}(e_E, e_E^i) := \sum_{\sigma \in \Sigma^{10}} e_E(\sigma) \log \frac{e_E(\sigma)}{e_E^i(\sigma)} \qquad (i = 1, 2)$$

be the Kullback-Leibler distance between the true emission distribution of state E and one of the two Markov chain approximations. Here, $\mathrm{KL}(e_E, e_E^1) \approx 2.41 > 1.26 \approx \mathrm{KL}(e_E, e_E^2)$. So the 1st order Markov chain of variant 2 is a better approximation to the true distribution than the 0th order Markov chain of variant 1 with respect to this measure.

2) We examined how well the two variant models of type E strings perform in classifying strings as either type E or not type E. Fix a variant model $i \in \{1, 2\}$ of state E. The classification process is as follows. Given a threshold α and a string $\sigma \in \Sigma^{10}$, the string is classified as type E if and only if $e_E^i(\sigma) \geq \alpha$. If σ is classified as type E and σ really is of type E then this is counted as true positive (TP), if if is classified as type E but it really is of type F then it is counted as false positive (FP). Let $\mathrm{PP}_i(\alpha) := \{\sigma \in \Sigma^{10} \,|\, e_E^i(\sigma) \geq \alpha\}$ be the set strings classified as type E (predicted positive). We assume that the strings σ are generated according to the true distribution and define for each of the two variants the true positives and false positives rates as a function of the threshold:

$$\mathrm{TP}_i(\alpha) := P(Y \in \mathrm{PP}_i(\alpha) | X = E) = \sum_{\sigma \in PP_i(\alpha)} e_E(\sigma)$$

$$\mathrm{FP}_i(\alpha) := P(Y \in \mathrm{PP}_i(\alpha) | X = F) = \sum_{\sigma \in PP_i(\alpha)} e_F(\sigma)$$

As α decreases from 1 to 0 the true positive and false positive rates each increase in discrete steps from 0 to 1. The ROC 'curve' is a plot of the true positive rate against the false positive rate for different thresholds. It is shown in Figure 4.3.

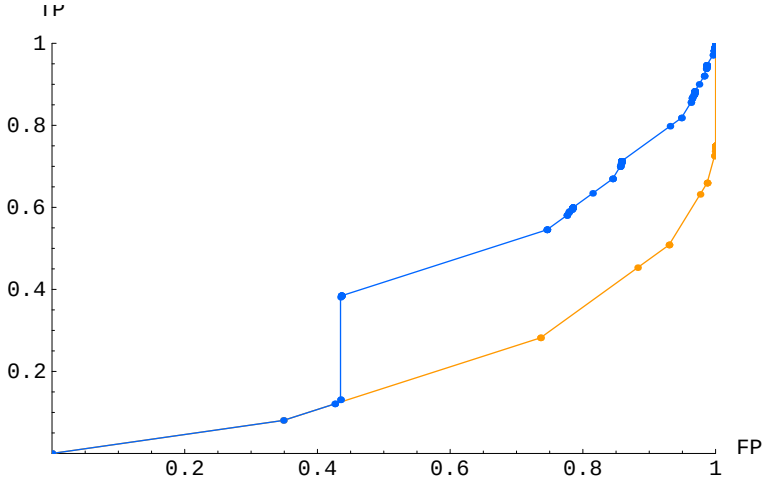

Figure 4.3: The ROC curves for the models of state E. The blue (upper, dark grey) curve is for variant 2, the orange (lower, light grey) curve is for variant 1. Up to a false positive rate of about 0.42 the true positive rates of the two variants are equal. Then $\sigma = 1111111111$ is introduced in $\mathrm{PP}_2(\alpha)$ which causes a jump in the true positive rate of size 0.25. The state model of E of variant 2 has a better ROC curve than that of variant 1.

While the 1st order Markov chain is a better model for state E than the 0th order Markov chain with respect to above two reasonable concepts of 'being better', it actually is the worse of the two models when integrated in the GHMM. Suppose we are given a random emission Y of type X coming from the true model. Let $\Phi^i_{\mathrm{vit}}(Y)$ be the Viterbi parse of emission Y in model variant $i \in \{1, 2\}$. (In this example the Viterbi parse is always unique.) Then the Viterbi parse is correct if it uses state X, i.e. if $\Phi^i_{\mathrm{vit}}(Y) = ((X, 10))$. In that case the prediction is correct. This probability is

$$\begin{aligned} P(\Phi^i_{\mathrm{vit}}(Y) \text{ is correct}) &= P(e^i_E(Y) > e_F(Y)\,,\, X = E) + P(e^i_E(Y) \leq e_F(Y)\,,\, X = F) \\ &= \sum_{\sigma \text{ s.t. } e^i_E(\sigma) > e_F(\sigma)} e_E(\sigma)/2 + \sum_{\sigma \text{ s.t. } e^i_E(\sigma) \leq e_F(\sigma)} e_F(\sigma)/2. \end{aligned}$$

For the two variants the probabilities of a correct prediction are

$$P(\Phi^1_{\mathrm{vit}}(Y) \text{ is correct}) \approx 71.1\% \qquad \text{and} \qquad P(\Phi^2_{\mathrm{vit}}(Y) \text{ is correct}) \approx 68.8\%$$

Therefore the GHMM of variant 1 is better than the GHMM of model variant 2. (Compare: The Viterbi parse of the *true* GHMM is right about 72.7% of the time. This is here also the theoretical optimum achievable with any decision procedure.)

How could it happen that the better stand-alone state model is the worse model to be integrated in the GHMM? There are two possible reasons why an input sequence can be parsed correctly in variant 1 but incorrectly in variant 2. Either the sequence is of type E

and the state model of E of variant 2 assigns a lower probability to the sequence as the state model of E of variant 1. Or the sequence is *not* of type E and the state model of E of variant 2 assigns a higher probability to the sequence as the state model of variant 1. An example of the first kind is the sequence 1010101010, which is more likely to be of type E than of type F. Assume it was of type E. We have

$$e_E^1(1010101010) \approx 0.00080, \quad e_E^2(1010101010) \approx 0.0000016, \quad e_F(1010101010) \approx 0.0000059$$

So in this case, variant 1 makes the correct prediction but variant 2 does not.
A question arising is: Given two variants e_q^1 and e_q^2 of a state model of state q are there possibilities to decide which one yields better predictions when integrated in a given GHMM *without* making the decision depend on the GHMM? As we have seen, the Kullback-Leibler distance and the ROC curve are no such possibilities. A trivial condition on the two variants that ensures that the GHMM in the second variant is at least as good as in the first variant is to demand that for every possible true positive example for a sequence emitted in state q the probability of the second state model is at least as large as the probability in the first state model, and that for every possible true negative example the probability of the second state model is at most as large as the probability in the first state model. But when all sequences can be both true positive and true negative as in the above example this condition holds only if the two variants actually are the same. We have found no decision method which is helpful in practice and believe that it usually depends on the GHMM which state model is better.
A conclusion we draw is that one GHMM may perform better with a certain state model and another GHMM may perform better when a different state model for that state is used.
In the light of this conclusion we can understand why AUGUSTUS performs better with a 4th order Markov Model for the coding regions while GENSCAN and GENIE apparently perform better with a 5th order Markov Model (see section 7.3.2).

Chapter 5

Using Extrinsic Information – AUGUSTUS+ and AGRIPPA

The model described in Chapter 3 is designed for the prediction of genes solely based on the DNA sequence. But often additional indication for a gene in certain regions of the sequence can be acquired from elsewhere. We call that *extrinsic information* when the source of the information lies beyond the sequence itself. The most important examples are

- Match in a Protein database. A part of the DNA sequence which is similar to an amino acid sequence of some protein when translated in one of the six possible reading frames is by experience more likely to be coding itself. This holds even when the protein is from a different organism. The 'nr' database we used holds about 1.5 million amino-acid sequences.

- Match in an EST database. An EST (expressed sequence tag) is a short (mostly 300-500 bp long) piece of cDNA. cDNA is a DNA copy of the mRNA sequence of a gene (the sequence in the middle level in Figure 1.1). An EST contains no intron sequences. It consists of coding sequence and/or non-coding exons. Large publicly available databases of EST sequences exist, too. A part of the input DNA sequence that is both similar to an EST and in translation to an amino-acid sequence in the database is also likely to be coding itself. The reliability here is larger than in the case with only a protein similarity.

- Regions of varying levels of conservation in an inter-species sequence comparison. When two sequences of two different species have collinear regions of high similarity and other regions of low similarity this can be regarded as evidence that the regions of high similarity correspond to functional parts of the sequence and therefore are more likely to be coding.

- Anchor constraints of the user. The user may have certain information about a gene in the sequence. For example she may know the position of the translation start site beforehand or she simply wants to assume that a certain part of the sequence is coding. In this case she can 'tide an anchor' and force AUGUSTUS to predict only gene structures which are consistent with her assumptions.

In the first three cases extrinsic information is insecure in the sense that the gene structure might be different than indicated by the extrinsic information. Often a piece of extrinsic information is also imprecise, for example a match in a protein database suggests an exon in a certain region but the exact boundaries cannot be determined by the match. This chapter deals with the incorporation of the insecure extrinsic information derived by protein and EST database searches. The method used here allows to specify together with a piece of information measures of its level of insecurity. This will automatically give a possibility to let the user tide anchors, because those anchors can be regarded as secure extrinsic information.

5.1 The Methods of Other Programs

Programs which adopt an extrinsic approach using sequence similarity information as in the first three cases above are called *homology-based* as opposed to *ab initio* programs which do not use this information. Many such programs exist and in [MSSR02] an overview over 22 such programs is given listing their methods. The methods include *spliced alignments* between the input DNA on the one hand and a protein sequence (PROCRUSTES, GENEWISE), a cDNA sequence (GeneSequer) or EST sequences (TAP [KRGS01]) from a database on the other hand, *cross-species alignments* of homologous genomic DNA sequences (AGenDA [MRA$^+$02], [RM02], [TRG$^+$03], CEM [BH00]) and ad-hoc methods to change existing ab-initio methods (SGP2 which bases on GeneID [PAA$^+$03]).
In this section I will only describe methods to integrate extrinsic information into a program based on a Hidden Markov Model.
The program GENIE ([KHRE97], [Ree00], [Kul03]) which is based on a GHMM integrates protein and EST homology information. Some of the EST matches which indicate an intron are used to anchor the gene structure prediction: 'The content sensor models for splice sites and introns are modified such that the probability was artificially raised for these so-called EST introns, effectively constraining the system to ensure that the introns were correctly annotated according to the EST/cDNA evidence.'[Ree00]. Extrinsic information from protein database matches were used to possibly raise the emission probability in an exon containing the protein match. In doing so the match was either fully considered or not at all. The emission probability of an exon which only *partly* covers the protein match

is not increased. Unfortunately, the method used to assign an emission probability to a protein database match is not described in detail. The emission probability in regions without database matches remains unchanged in comparison to the ab initio version.
The program HMMGene [Kro00] integrates – among others – protein and EST database hits. As opposed to the ab-initio-HMMGene, the joint probability of a sequence of states (parse) and the emission in the extended HMMGene, includes a factor for each base which depends on the type of extrinsic information given at this base. For example for each base of the input DNA sequence covered by a protein database match with 100% identity the factor is twice as large in an exon state than in an intron state. Therefore a parse with an exon including the region of the match gets a relative bonus of 2 to the power of the length of the match when compared to a parse with an intron including the region of the match. At positions which are not covered by a database match the factor is equal for all states of the model. Therefore the absence of database hits does not influence the prediction. Integrating EST matches with this method did not improve the prediction accuracy of HMMGene noticeably. Krogh explains this as follows '... the probability of the region is this probability [above factor] raised to the power of the length of the match. ... for ESTs experimentation with other types of length dependences is necessary.'
GENOMESCAN [YLB01] is an extension of GENSCAN [Bur97] which integrates BLAST hits of the DNA input sequence in protein sequences. They use the following heuristic in their program. The information given by a gap-less protein hit is reduced to the BLAST p-value and a certain position, called centroid, in the middle of the range of the hit. The authors of GENOMESCAN assume that the probability of a BLAST hit in a protein database to be artifactual and therefore misleading is approximately the 10th root of its p-value (for small e-values, below 0.01, the p-value and e-value are almost the same). A parse complies with the hit when this centroid is a coding base in the parse. The emission probabilities of parses of which comply with the information given by a BLAST hit with p-value p receive a relative bonus factor which is in the order of $p^{-\frac{1}{10}}$. Which means for example that a BLAST hit with e-value 10^{-120} leads to a relative bonus factor of approximately 10^{12} whereas a BLAST hit of e-value 10^{-10} leads to a bonus factor of only approximately 10. In this case the small e-value has a bonus by a factor of 10^{11} larger than the small e-value.
TWINSCAN [KFDB01] is a reimplementation of GENSCAN which additionally integrates information retrieved from non-annotated DNA sequences from other species homologous to the input DNA sequence (the third source of extrinsic information). For example, human genomic sequences may serve TWINSCAN as informant sequences when predicting genes in mouse input DNA sequences. BLAST is used to find high-scoring local alignments of the input DNA sequence to informant sequences. Then each base of the input DNA sequence is classified into one of three conservation categories (unaligned, matched or

mismatched) according to the results of the BLAST searches. The model of TWINSCAN then assigns a probability to each parsed DNA sequence together with the the parallel sequence of conservation categories, called the conservation sequence. These two sequences are considered to be independent and the distribution of the conservation sequence is that of a fifth-order Markov chain depending on the state emitting the sequence. These Markov Models for the conservation sequence were also trained on annotated sequences.
Another Hidden Markov model which uses homologous sequences from two species is DOUBLESCAN [MD02]. As opposed to TWINSCAN, this program simultaneously predicts the gene structures of two DNA input sequences which are required to be orthologous or paralogous (The genes diverged in evolution after a speciation or a duplication event, respectively). The model uses a so-called *pair HMM* which emits two DNA sequences instead of one. It has states corresponding to matching exons on both sequences, i.e. pairs of homologue exons, which emit the DNA sequences of the exons in the two sequences simultaneously codon by codon. And it has states corresponding to exons in one of the sequences without counterpart in the other. Thus emitting codons of only one of the two sequences.

5.2 Extrinsic Information about Genes

The extrinsic information we currently use as input to AUGUSTUS, is automatically generated by a program of Oliver Schöffmann, called AGRIPPA [Sch03]. The eponymous Roman general Agrippa was an adviser and close associate of the Roman emperor Augustus. This program is briefly described in the following section 5.2.1. In section 5.2.2 we list the different types of extrinsic information we distinguish.

5.2.1 The Program Agrippa

AGRIPPA uses two databases to infer information about the coding regions in the input DNA sequence. The whole protein database *nr* which contains (possibly partial) amino acid sequences of proteins. The whole EST database *dbEST* which had about 5 400 000 entries for human and 260000 for *Drosophila melanogaster* in August, 2003. The program bases on the local alignment search tool BLAST [AGM$^+$90], which efficiently finds local alignments of the input DNA sequence to a similar sequence in a large database. Before such a database search is initiated, putative repetitive elements in the input DNA sequence are masked using the program RepeatMasker (http://ftp.genome.washington.edu/RM/RepeatMasker.html, unpublished results). Below we will use the word *segment* for a (contiguous) subinterval of a sequence.

Using protein database matches

When run on the protein database, AGRIPPA uses the results of a BLAST search (more precisely: blastx with standard parameters) of the input DNA sequence against the database. The hits reported by BLAST are local alignments between the input DNA sequence and a target amino acid sequence, possibly with gaps.
AGRIPPA assumes that the segments of the input sequence that are aligned to segments of the target amino acid sequence, are themselves predominantly coding. The reading frame and the strand of the presumable coding parts can be determined by the alignment. If the alignment contains a large enough gap in the target sequence this is considered evidence for an intron in the input DNA sequence which is aligned to this gap. At the boundaries of this gap the alignment is often of bad quality and not reliable. The same holds if BLAST reported two local alignments of the same amino acid sequence to the input DNA sequence, which overlap only in the amino acid sequence. Then possible splice site pairs which obey the GT/AG consensus are searched for in the bordering region of the presumable intron. That pair of possible splice sites is chosen and output as presumable splice sites, that maximizes the number of identically matched amino acids in the alignment in the neighboring coding regions defined by the choice of splice sites.
If the segment between two presumable splice sites found this way – acceptor site upstream, donor site downstream – has been aligned to a segment of the target sequence a possible exon is output by AGRIPPA. If an alignment matches a segment of the input DNA sequence to the target sequence and no evidence for splice sites was found at *both* boundaries, this segment is considered to be part of a possibly larger coding exon, and hence is output.
If an alignment matches a codon of the input DNA sequence to the first amino acid of some protein in the database this is interpreted as evidence for a translation start site (start codon) at this position in the input DNA sequence. Analogously, a possible translation termination site is output if a stop codon in the input sequence follows directly downstream of an alignment which aligns the last amino acid of a protein to the input DNA sequence.

Using EST database matches

When run on the EST database, AGRIPPA also uses the results of a BLAST search (more precisely: blastn with standard parameters) of the input DNA sequence against the database. In this case the local alignments are between two DNA sequences and usually contain few mismatches and few short gaps. The strand (orientation) of an EST is unknown. Also – as ESTs are parts of the whole cDNA – it is not known whether they come from coding or non-coding parts (the so-called *untranslated region*) of the mRNA.
Again, if an alignment contains a long gap in the target EST sequence an intron is inferred

and thus also two presumable splice sites. Then the strand is determined by the splice sites. In AGRIPPA different local alignments which overlap in the input DNA sequence and do not lead to contradicting intron information, are clustered to larger alignments which are contiguous in the input sequence. Those alignments are filtered by the alignment score and clipped off somewhat at the ends. Then segments of the input sequence which are aligned without gaps to segments of an EST sequence and are bordered by two presumable splice sites are output as possible exons. Those segments which are aligned to a segment of an EST sequence but not bordered by presumable splice sites on both sides, are output as possible parts of an exon.
There is a systematic error being made here. It is to due to the fact that ESTs can theoretically only be used to infer the mRNA sequence of a gene. So this method also finds presumable *non-coding* exons. Which part of that mRNA sequence is coding cannot be derived by ESTs alone. Therefore, AGRIPPA also tries to verify which parts of the partially reconstructed mRNA is coding by performing a protein database search with this sequence.

Combining EST with protein database matches

After the EST database has been used to partially reconstruct the mRNA, each presumable part τ of an mRNA sequence is searched against the protein database. In this search the BLAST algorithm does not need to detect long gaps because τ does not contain introns. The parts of τ which are aligned to an amino acid sequence are relatively likely to be coding. Then the information from the original alignment of the ESTs to the input DNA sequence can be used to infer a partial presumable intron/exon structure. Again, a protein hit can be used to infer a translation start or stop site if the first or last amino acid of a protein has been aligned, respectively. Figure 5.1 illustrates with an errorless example the way Schöffmann concludes.

5.2.2 Types of Extrinsic Information

The types of extrinsic information retrieved this way are

1. *start*. A presumable translation start site of a gene; the start codon.
2. *stop*. A presumable translation termination site of a gene; the stop codon.
3. *ASS*. A presumable acceptor (3') splice site of a gene.
4. *DSS*. A presumable donor (5') splice site of a gene.
5. *exonpart*. A segment of the sequence presumably coding: part of an exon. The actual exon may properly contain this segment or may be equal to the segment.

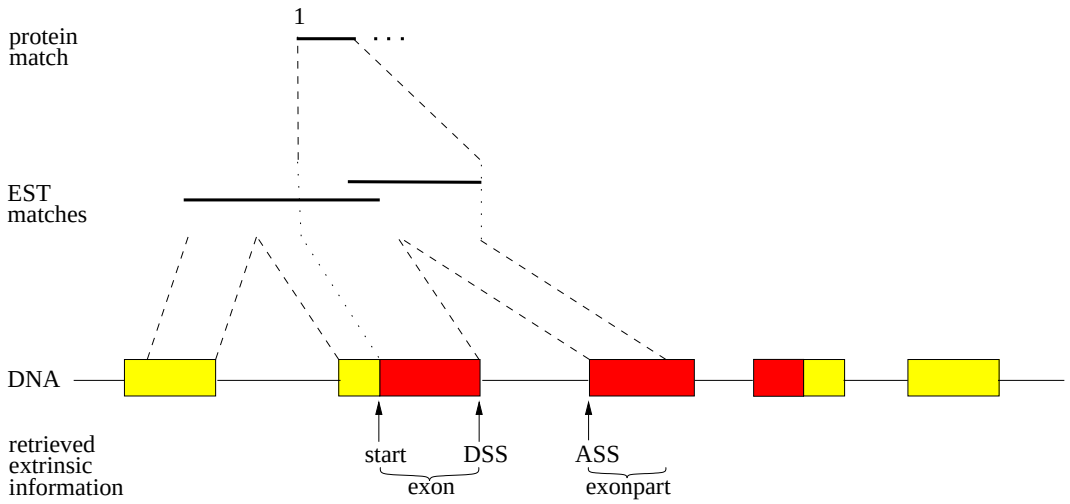

Figure 5.1: The information retrieved from a combination of EST and protein database searches. The input DNA sequence contains one gene of which the dark (red) boxes are the coding parts. First ESTs matching the DNA sequence are found and clustered. The concatenation of the segments of the input DNA sequence which are aligned to the clustered ESTs is searched against a protein database. The protein match can be used to infer which part of the EST consensus sequence was coding. In this example the alignment of the protein started at the first position in its amino acid sequence. Thus a likely translation start site (start) can be inferred.

6. *exon.* A complete presumable coding exon.

We will call an individual piece of extrinsic information a *hint*. Each hint has a *grade* which is from a discrete set G_j corresponding to the type $j \in \{1, \ldots, 6\}$ from above enumeration. The grade is assigned to the hint depending on the type of process leading to the hint (e.g. protein, user anchor). It may for example also depend on the BLAST e-value, but we chose to ignore that (see section 5.3.3). The grade will later help assessing the reliability of the hint.

5.3 Extended Model - AUGUSTUS Takes Hints

5.3.1 Goals of the Modeling

Our aims in constructing a new model incorporating extrinsic information were the following.

Firstly, in the search for an optimal gene structure a gene structure which regards a hint should get a 'bonus' over one that ignores this hint. Suppose S_1 and S_2 were two gene structures that have equal a-posteriori probabilities in the ab initio model AUGUSTUS. And suppose we had one single hint which supports S_1 but not S_2. Then in the new model

S_1 should get a higher a-posteriori probability than S_2.
Secondly, the bonus of a gene structure respecting a hint which refers to a *range* of the input sequence (exonpart and exon) should – if at all – only moderately depend on the length of that range. Experience showed that *long* matches of the DNA input sequence in a protein or EST sequence are not much less likely to be misleading or artifactual than *shorter* ones (section 5.3.3).
Thirdly, gene structures which only 'partially respect' a hint which refers to a range should not be rewarded at all. For example, exons covering only half of an EST match do not get a bonus. If this exon was correct the EST would be wrong or belong to a different form of alternative splicing.
Fourthly, the program should not be forced to regard an insecure hint, as this can be wrong. (Advisors sometimes give ill counsel.) As Figure 5.2 shows, hints can be misleading. If the a-posteriori probability of the most likely gene structure is very high in the ab initio model, an uncertain hint which is incompatible with this gene structure should not necessarily lead to a different prediction in the new model.
And fifthly, if actual genes usually are supported by extrinsic information, a gene structure with genes for which there is no supporting extrinsic information should get a 'malus'. Suppose again we had two gene structures S_1 and S_2 which have the same a-posteriori probability in the ab initio model and are equal except that S_1 contains an additional gene or exon that S_2 does not contain. Further suppose that no extrinsic information supports this additional gene or exon. Then S_1 should have a lower a-posteriori probability than S_2 in the new model. This aim needs some explanation. Hints found through database searches all support coding regions in some way. We follow the guideline: 'no information' is also information. As an example consider the extrinsic information of type start. Suppose the reliability of the process generating the start hints was so high that for almost all genes a start hint supporting it was given, and only a very small fraction of the start hints was wrong. Then, intuitively, a predicted gene without supporting start hint would be suspicious, because it would violate the practical experience that very rarely true genes have no supporting hint.
Remark: While accomplishing the first aim tends to increase exon-level sensitivity by giving some exons a bonus, accomplishing the third aim tends to increase exon-level specificity because some exons 'get punished' through the malus and are therefore not predicted.
All of the programs mentioned in section 5.1 reach the first of these goals. Only GENOMESCAN reaches the second goal, since here, the extrinsic information referring to a range is reduced to a single base. For the other programs, the relative bonus of a parse which respects a homology on a certain range is a product over all bases/codons of that range and approximal exponential in the length of that range. Only GENIE reaches the third goal. All programs reach the fourth goal with one exception: GENIE is forced to respect

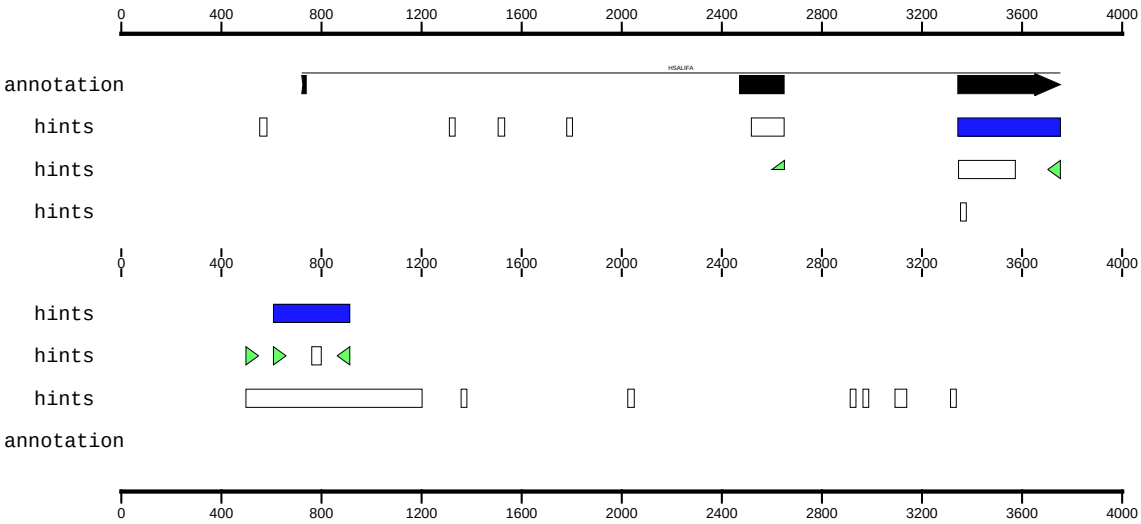

Figure 5.2: Section of an input sequence containing one gene and extrinsic information. The upper half of the graph refers to the forward strand, the lower half to the reverse strand. The first line shows an annotated gene with three exons (black). No exons on the reverse strand were annotated. The lines labeled 'hints' show the extrinsic information retrieved by AGRIPPA from the results of a BLAST search in a protein database. The white boxes are exonpart hints, the two black boxes are exon hints. The grey (green if viewed in color) triangles on the forward strand at 3753, and on the reverse strand at 498 and 608 are stop hints. The one on the forward strand coincides with the annotated position. The right grey triangle on the reverse strand is a start hint and the grey 'half-triangle' on the forward strand at 2650 is a DSS hint at the correct position.

extrinsic information about introns. Only TWINSCAN reaches the fifth of our goals. In the TWINSCAN model missing extrinsic information corresponds to a conservation sequence with the classification 'unaligned', which is presumably more likely to be emitted from non-coding states than from coding states. The other programs which have an ab initio version behave as in the ab initio version, when no extrinsic information was found in a region. The approach explained below has been chosen to attain above goals.

5.3.2 The extended Emission Alphabet

The input to a GHMM is a sequence which contains all information used by the Viterbi and forward algorithm to assess possible underlying structures. It is therefore natural to try to interpret the available information as sequence information. In the case of the first four types of hints this is obviously possible. They contain a precise local information. For example, 'at position 2650 on the forward strand is a DSS hint with grade *Protein*' or 'at position 2651 in the input sequence is no DSS hint'. In the case of exonpart and exon hints, which refer to a range in the input sequence, this is also possible but with a

notional trick. We assign the hint to the base at the right end of the range and let the *length of the range be part of the information* given by the hint. This defines a one-to-one correspondence between the exonpart intervals on the one hand and the end point and the length on the other hand. For example, 'at position 1202 on the reverse strand is an exonpart hint with length 705' contains the same information as if we would have said that the exonpart hint starts at position 498 and ends at position 1202. The exon hints are also assigned to their rightmost position (including the splice site dinucleotide consensus, if applicable).

In this way all the extrinsic information to a DNA sequence together with the input sequence can be regarded as one sequence over a new extended countable alphabet Σ', which is formally defined below. The emission of the extended GHMM then is a DNA sequence together with an annotation of all extrinsic information belonging to this sequence.

For each type $j \in \{1, \ldots, 6\}$ let G_j be the set of possible grades. In the application given here, $G_1 = \cdots = G_6$ is a subset of the set $\{Manual, Protein, EST, Combined\}$, depending on which sources of information actually have been exploited. The grade is the source of the extrinsic information: manual anchor, protein homology, EST homology or combined EST and protein homology. These sets G_j can be extended as needed. Let the special letter $\pitchfork$ (pitchfork) denote 'no special hint', let '$\rightarrowtail$' and '$\leftarrowtail$' stand for the forward strand and the reverse strand, respectively. Define

$$\mathcal{F}_j := (G_j \times \{\rightarrowtail, \leftarrowtail\}) \cup \{\pitchfork\} \qquad (j = 1, 2, 3, 4)$$

and

$$\mathcal{F}_j := (G_j \times \{\rightarrowtail, \leftarrowtail\} \times \mathbb{N} \times \{0, 1, 2\}) \cup \{\pitchfork\}. \qquad (j = 5, 6)$$

Then

$$\Sigma' := \Sigma \times \mathcal{F}_1 \times \mathcal{F}_2 \times \mathcal{F}_3 \times \mathcal{F}_4 \times \mathcal{F}_5 \times \mathcal{F}_6$$

is the extended emission alphabet of the new model. Let $(b, f_1, f_2, f_3, f_4, f_5, f_6) \in \Sigma'$ be the 'extended letter' observed at position i of the emitted sequence. This is interpreted in the following way.

$b \in \{A, C, G, T\}$ is the nucleotide at position i in the DNA sequence.

f_1 specifies the start hint belonging to position i. If $f_1 = \pitchfork$, then no special start hint is observed at position i. And if $f_1 = (g, d)$ with $g \in G_1, d \in \{\rightarrowtail, \leftarrowtail\}$ then a start hint about a start codon on strand d (**d**irection) with grade g ending at position i is given.

f_2, f_3 and f_4 specify the stop, ASS and DSS hint belonging to position i in the same way as f_1 specifies the start hint.

f_5 specifies the exonpart hint belonging to position i. If $f_5 = \pitchfork$, then no special exonpart hint is observed at position i. Otherwise, let $f_5 = (g, d, \ell, r)$. g is the grade of the hint, d is the strand and $r \in \{0, 1, 2\}$ defines the **r**eading frame. ℓ is the length of the exonpart interval and the exonpart interval goes from bases $i - \ell + 1$ to i.

f_6 specifies the exon hint belonging to position i in the same way as f_5 the exonpart hint. Note that $i - \ell + 1$ and i are the left and right end of the exon hint interval, which includes the splice site consensus dinucleotides if the boundary of the exon is a splice site. The exonpart interval boundaries may therefore differ a little from the boundaries of the biological exon.
With this extended alphabet Σ' the whole information given by the input DNA sequence and the extrinsic information of the six types obtained through the database searches can be regarded as a string over Σ'. To each given hint of type j corresponds exactly one position in the sequence where the component f_j of the observed extended letter $(b, f_1, f_2, f_3, f_4, f_5, f_6)$ is different from $\Cap$.
Remark: We allow at most one hint per position per type. If the process generating the hints produces multiple hints we delete all but one. However, multiple hints at the same position of different types are allowed.

5.3.3 The extended Emission Distribution

The emission distribution e' of the GHMM AUGUSTUS extended this way needs to be redefined to account for the extended emission alphabet Σ'. We will call this extended model AUGUSTUS+ ('plus advisor'). It is defined as the GHMM with the same state space Q^*, the same transition matrix A (see chapter 3) but new emission probabilities $e'_{p,q,\tau'}(\sigma')$ $(p, q \in Q^*, \tau', \sigma' \in \Sigma'^*)$. Let q be the current state, p be the previous state, and $\tau' \in \Sigma'^*$ be the previous emission. Let τ be the previously emitted DNA sequence, given by τ'. The definition of the new emission distribution e' was chosen to leave as much as possible unchanged of the previous definition of e. The distribution of the DNA string σ emitted in each state is the same as defined in chapter 3. The probability of an emission

$$\sigma' = (\sigma_1, f_{1,1}, \ldots, f_{1,6})(\sigma_2, f_{2,1}, \ldots, f_{2,6}) \cdots (\sigma_n, f_{n,1}, \ldots, f_{n,6})$$

is the emission probability of the DNA sequence $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$ times the product of the probabilities $P(F_{k,j} = f_{k,j})$ of the hint character for each position k and each type j:

$$e'_{p,q,\tau'}(\sigma') = e_{p,q,\tau}(\sigma) \cdot \prod_{\substack{1 \le k \le n \\ 1 \le j \le 6}} P(F_{k,j} = f_{k,j}).$$

$F_{k,j}$ is the random hint – possibly $\Cap$ – of type j at the kth position of the emitted string. It does not depend on the previous state but it may depend on q, τ and σ.
Before we specify the probability $P(F_{k,j} = f_{k,j})$ of a hint, we introduce some new terms. A full piece of extrinsic information is given by a hint $f \neq \Cap$, its position i, and the type j. For brevity, in cases where the position and type are clear, we will sometimes refer to the whole information as 'hint f', instead of 'hint f of type j at position i'. Whether the

hint is consistent with a parse depends only on that step in the parse, where position i is emitted. Let q be that state, emitting the sequence from positions a to b with $a \leq i \leq b$ in one step. The hint f is said to be *respected* by the the parse if and only if the gene structure corresponding to the parse is compatible with the hint of type j given by f at position i. For each type j this compatibility has a specific meaning.

For the start type ($j = 1$) this means that q is an initial or single exon state and the rightmost position of the start codon emitted in state q is exactly the position i of the hint.

Similarly, a stop, ASS or DSS hint at position i is respected by the parse if the position of the stop codon, the acceptor splice site or the donor splice site defined by q, a and b correspond to that hint.

For an exonpart hint $f = (g, d, \ell, r)$ of length ℓ, strand d and reading frame r, f is respected by the parse if q is an exon state on the strand given by d, if b, i and the reading frame of q match r correctly and if the interval of positions $(i - \ell, i]$ is contained in the interval of coding bases corresponding to q.

When $f = (g, d, \ell, r)$ is an exon hint, f is respected by the parse if, again, the strand and reading frame given by q, a and b match the strand and reading frame of f and the boundaries of the biological exon corresponding to q, a and b match exactly the boundaries of the biological exon corresponding to i, ℓ and q.

Observe that the statement 'parse ϕ respects hint f at position i' is even well defined if only an initial part of ϕ is specified, as long as sequence position i is included in this initial part.

When the (presumably) true annotation is given, we call the parse corresponding to the true gene structure the *true parse*. For each hint retrieved from the database searches using a training set we know, whether it was respected by the true parse or not. We call those hints *good* or *bad*, respectively.

A parse ψ *supports* type $j \in \{1, 2, 3, 4, 5, 6\}$ at position i in DNA sequence σ if and only if some hint f of type j is respected at i. Examples: A parse supports the start type at position i if in the gene structure given by the parse a start codon ends at i. A parse supports the exonpart type at all coding positions.

Let $1 \leq j \leq 6$ and f be a hint of type j, let $\rho \in \Sigma^*$ be a DNA string. Then f is said to be *observable* at position i in sequence ρ if and only if a minimal consistency condition for the hint f of type j at position i in sequence ρ is satisfied:

$j = 1$. A start hint $f = (g, d)$ is only observable if either $d =\rightarrowtail$ and the string 'ATG' ends at position i in ρ or $d =\leftarrowtail$ and the reverse complement of the string 'ATG' ends at position i in ρ.

$j = 2$. A stop hint $f = (g, d)$ is only observable if $d =\rightarrowtail$ and a stop codon ends at position

i in ρ or $d = \leftarrowtail$ and the reverse complement of a stop codon ends at position i in ρ.

$j = 3$. An ASS hint $f = (g, d)$ is only observable if $d = \rightarrowtail$ and the string 'AG' ends at position i in ρ or $d = \leftarrowtail$ and the reverse complement of 'AG' ends at position i in ρ.

$j = 4$. A DSS hint $f = (g, d)$ is only observable if $d = \rightarrowtail$ and the string 'GT' ends at position i in ρ or $d = \leftarrowtail$ and the reverse complement of 'GT' ends at position i in ρ.

$j = 5$. An exonpart hint $f = (g, d, \ell, r) \in \mathcal{F}_5$ is only observable if the substring $\rho(i - \ell, i]$ contains no stop codon in the reading frame given by r on the strand given by d.

$j = 6$. An exon hint $f = (g, d, \ell, r) \in \mathcal{F}_5$ is only observable if the substring $\rho(i-\ell, i]$ contains no stop codon on the strand given by d in the reading frame given by r and one of the following minimal consensus conditions holds. If $d = \rightarrowtail$, at position $i - \ell + 1$ the substring 'ATG' or 'GT' must start and at position i a stop codon or the string 'AG' must end. If $d = \leftarrowtail$, at position $i - \ell + 1$ the reverse complement of a stop codon or the string 'AG' must start and the reverse complement of 'ATG' or 'GT' must end and at position i in ρ.

With this definition a hint which is not observable contradicts biological knowledge or is a rare exception we do not account for. We consider only observable hints, because parses respecting non-observable hints have zero probability in AUGUSTUS anyway. Then every respected hint is observable.

For the first four types, the strand of an observable hint is determined by the sequence and the position. For example, for the start type it is not possible that both an ATG and the reverse complement of an ATG are at the same position in the sequence. Therefore, apart from the grades, there is at most one observable hint of type $j \in \{1, 2, 3, 4\}$ possible at each position.

Let p_j^+ and p_j^- be discrete probability density functions on G_j and let $r_j^+ \geq 0$ and $r_j^- \geq 0$ be constants to be specified below ($j \in \{1, \ldots, 6\}$). We are now ready to define the probability with which an individual random hint $F_{k,j}$, emitted in the current state, takes a value $f \neq \Uparrow$. Let ρ be the DNA sequence emitted so far, including the sequence emitted in the current state. Let g be the grade of f, let j be the type of f and let i be the position of the hint in ρ.

$$\mathrm{P}(F_{k,j} = f) = \begin{cases} p_j^+(g) r_j^+ & \text{if } f \text{ is respected by the parse;} \\ p_j^-(g) r_j^- & \text{if } f \text{ is observable at } i \text{ in } \rho \text{ but } f \text{ is not resp. by the parse;} \\ 0 & \text{if } f \text{ is not observable at } i \text{ in } \rho. \end{cases} \tag{5.1}$$

$p_j^+(g)$ is the probability that a respected hint of type j has grade g and $p_j^-(g)$ is the probability that an non-respected hint of type j has grade g. The constants r_j^+ and r_j^-

and the distributions p_j^+ and p_j^- of the grades were estimated from a training set (see section 7.2).

According to this definition, the probability of a hint depends only on its type, on its grade, on whether the hint is respected by the parse and on whether it is observable. In practice, $r_j^+ > r_j^-$ and hints which are not respected by the parse, but observable, get a relatively small probability. This accounts for those hints which are bad. They are less likely but still possible, so that the parse has the 'chance' of ignoring them. The hints which are respected by the parse are usually relatively likely compared to the non-respected hints.

Remarks: 1) The attentive reader has noticed that for those exons followed by a splice site not the whole coding sequence of an exon is emitted in the exon state but a few coding bases and the GT are emitted from the DSS state. In those cases we let the exon state emit the exon and exonpart hints for these bases instead of the DSS state. This is possible as the position of the splice site is already determined by the emission in the exon state. 2) The constants r_j^+ and r_j^- turn out small enough that under reasonable assumptions on the maximal length of open reading frames $P(F_{k,j} = \Uparrow) = 1 - \sum_{f \in \mathcal{F}_j \setminus \{\Uparrow\}} P(F_{k,j} = f) > 0$. Often, $P(F_{k,j} = \Uparrow)$ will be close to 1.

Relevance of the length of hints

This section applies to hints of type exonpart and exon only. They refer to a range of the sequence which is possibly homologous to some other (protein or EST) sequence and can have a variable length. As mentioned above, several other programs indirectly attribute great importance to this length because the emission probability from their ab initio version is changed in the extrinsic version by a certain factor *for each of the bases* of that range. Therefore the relative bonus in probability of a gene structure respecting that hint is exponential in the length of that hint. This leads to a an extreme bias towards respecting longer hints.

We examined using our human training set how the length is actually correlated with the classification good hints versus bad hints. We consider only observable hints, i.e. in particular, they are contained in an open reading frame. Figure 5.3 shows histograms of the lengths of good and bad exonpart hints from protein and EST database searches. The vertical axes are scaled to show probabilities. The length distribution of good and bad exonpart hints from proteins are very similar and there is no reason to prefer respecting long hints over shorter ones. For ESTs the two distributions are somewhat different. Good exonpart hints tend to be shorter than bad ones. So if the only information about an EST exonpart hint was its length, we should rather prefer short hints over long ones because they are more likely to be correct. Despite this result that the length of the EST exonpart hints actually provides some information in this case, we chose to not take this

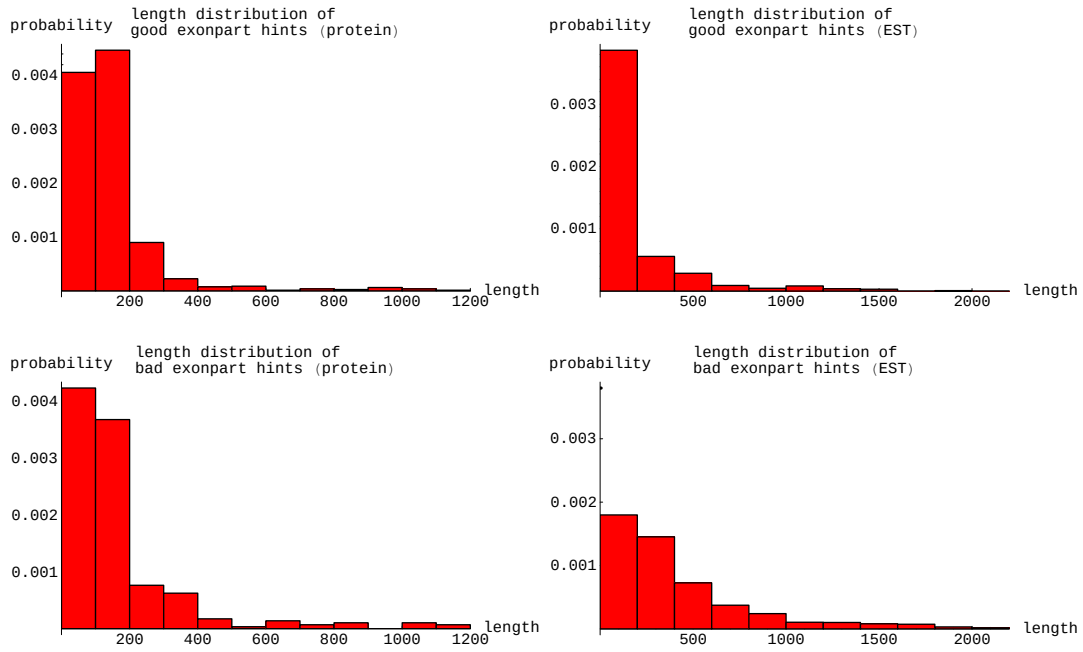

Figure 5.3: The graphs on the left side show the length distribution of correct (top) and incorrect (bottom) exonpart hints coming from a BLAST search in a protein database. The graphs on the right side show these distributions for the search in an EST database. An exonpart hint is considered correct only if it really is completely contained in the coding sequence. For proteins, the length does not seem to contain any information about the correctness. For ESTs there is the tendency that an EST hit is more likely to be correct if it is shorter.

into account, because the effect is relatively weak. But if we had taken the lengths into account by adding extra grades for different classes of lengths, the effect would be the opposite of of what is achieved by other programs as HMMGene: Shorter hints would have been preferred over longer ones. The lengths of good and bad hints of the type exon also have similar distributions (data not shown) and were not considered for the prediction.

Relevance of the BLAST e-value

The hints retrieved from database searches were constructed from BLAST hits. Each BLAST hit has a so-called *e-value* assigned to it, which has the following meaning. Initially, BLAST-hits are assigned a score defined through a scoring matrix and gap penalties. The score is the higher the more significant the hit presumably is. The BLAST search leading to this hit was performed in some search space given by the input sequence and a database. The e-value of a hit with score s is the expected number of hits with a score at least s in this search space under the assumption that the sequences of the search space are independent

and randomly distributed according to some simple distribution model [KA90]. E-values are nonnegative and the e-value is the smaller the larger the corresponding score is. We used an e-value cutoff of 10, which means that for each BLAST hit used in the construction of hints the expected number of hits by pure chance with this score or better was at most 10. Most e-values of the hits we used were below 1, many below 10^{-50}. For a distribution of the e-values leading to exonpart hints see Figure 5.4.

One might expect the following relationship between the e-values and the reliability of hints: The smaller the e-value the less likely is it that the hint is bad. This has been incorporated in the program GENOMESCAN by making the bonus of a gene structure that classifies some middle point of the reported hit as coding, inversely proportional to the 10th root of the p-value. This yields an extreme preference towards respecting BLAST hits with a small p-value (and therefore e-value). In our examination of the hints reported by AGRIPPA the distribution of the e-values of good hints of a type was very similar to the distribution of e-values of bad hints of this type, for all 6 types. Figure 5.4 shows a histogram of the e-values of $n_1 = 810$ good and $n_2 = 295$ bad exonpart hints derived from protein hits with the training set e500. Astonishingly, the fraction of *bad* exonpart hints with a small e-value (say less than 10^{-50}) is almost the same as the fraction of *good* exonpart hints with a small e-value. Based on these results it would not be reasonable to prefer exonpart hints with a small e-value over ones with a larger e-value. Of course, AGRIPPA has filtered and slightly modified the original BLAST hits, so there might be a correlation between the e-values of the original BLAST hits and their classification as good or bad. Distinguishing different grades for any of the types according to the e-value did not improve prediction accuracy. Thus we chose not to do so.

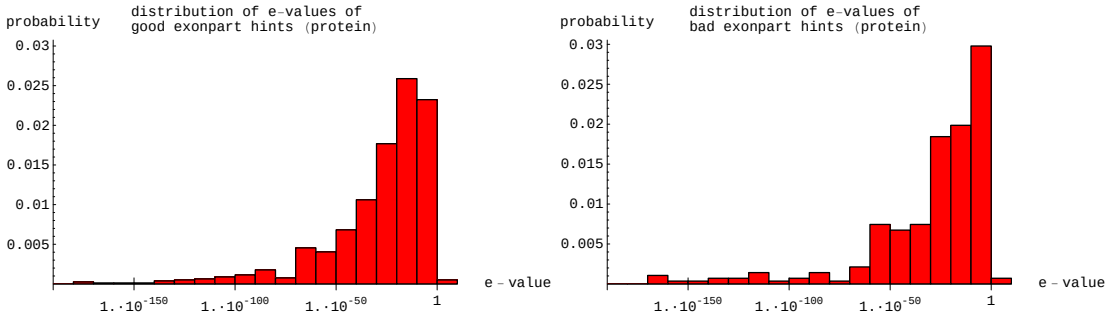

Figure 5.4: Distribution of the e-values of the hits in a BLAST search against a protein database. The left graph shows only the good hints and the right graph shows only the bad exonpart hints. Somewhat counterintuitive the distributions are fairly similar.

Choice of the set of grades

In this section we will define for each $j \in \{1, \ldots, 6\}$ the set of possible grades G_j and the two distributions on the grades p_j^+ and p_j^-. The purpose of the grades is to allow in the model for hints of each type different levels of reliability and frequency. For example the exonpart hints inferred from a combination of an EST and protein database search are more reliable but less frequent than start hints inferred from just a protein search. Suppose we combine at the same time extrinsic information coming from several sources. For example, some hints are manual anchors, some are from protein database searches and some are from EST database searches. Then we would like to be able to make use of our prior knowledge about these different sources as in above example. As the set of grades G_j we chose for each type j the subset of

$$\{Manual, Protein, EST, Combined\}.$$

of those sources actually used in the prediction. For example, if we use the hints from a protein and an EST database search then $G_j = \{Protein, EST\}$, no matter if we found hints. Here, each grade stands for a different source of the information. But the grades could also be used to incorporate prior knowledge about for example protein hit e-values. Then we would use instead of one grade *Protein*, say, 2 grades *Protein1* and *Protein2* for protein hints with low or high e-value, respectively. The configuration files of AUGUSTUS actually allow for any number of classes for each source, defined by ranges of scores in the score column of the GFF format. This feature has not yet been made use of in lack of an informative score for the hints.
The probabilities $p_j^+(g)$ $(g \in G_j)$ are estimated according to the relative frequency of grade g among all good hints of type j in the training set. Correspondingly, the probabilities $p_j^-(g)$ $(g \in G_j)$ are estimated according to the relative frequency of grade g among all bad hints of type j in the training set.
The grade *Manual* is an exception. We introduced the grade *Manual* in order to be able to let the user set absolutely trustworthy 'anchor hints'. I.e. hints that *must* be respected under all circumstances. Methodically, this can be achieved by setting the probability

$$p_j^-(Manual) = 0 \qquad (j = 1, \ldots, 6).$$

In words, in the model we never observe un-respected manual hints. We set

$$p_j^+(Manual) = \frac{1}{2} \qquad (j = 1, \ldots, 6).$$

Remark: As long as $p_j^+(Manual) > 0$ this probability is arbitrary and has no influence on the a-posteriori probabilities of parses: For each hint of type j of grade 0, all parses which do not respect the hint have emission probability 0 as $p_j^-(Manual) = 0$. All other parses

have for this hint the same multiplier of $p_j^+(\mathit{Manual})$, which does not change the ratios of emission probabilities among respecting parses and therefore not the ratios of a-posteriori probabilities.
If G_j contains *Manual*, the probabilities $p_j^+(g)$ of all other possible grades are proportional to their relative frequency in the training sets but scaled such that $\sum_{g \in G_j} p_j^+(g) = 1$.

Estimating r_j^+ and r_j^-

The constants r_j^+ and r_j^- were estimated in the human version using the training set e500. The numbers below refer to the human version. For simplicity in the description below assume that the sequences and their annotation in e500 were concatenated and we had only one DNA sequence σ of length $L = 4718341$. As the sum of the $p_j^+(g)$ over all possible g is 1, the number r_j^+ *is the probability of observing a respected feature f of type j* (with the given length and reading frame if $j = 5$ or $j = 6$). I.e., if it was not for the grade, in this model all respected features had equal probability. We use the following empirical method to estimate r_j^+. We assume the number of actually *observed* good features is equal to the *expected* number of good features given the true parse, which depends on r_j^+.
For the start type ($j = 1$) this means the following. Given the true parse ψ of the sequences in e500 we compute the expected number of respected start hints in this model. The number of bases where a start hint can be respected is the number of translation start sites which equals the number of genes: 500. We denote this number with #(translation initiation sites). At each of these bases the probability of observing any start hint is according to (5.1)

$$\sum_{g \in G_1} p_1^+(g) r_1^+ = r_1^+$$

as the strand of a respected hint is given by ψ. Therefore the expected number of start hints respected by the true parse is

$$\#(\text{translation initiation sites}) \cdot r_1^+ \tag{5.2}$$

Assuming 5.2 equals the number of observed good start hints yields

$$r_1^+ := \frac{\#(\text{good start hints})}{\#(\text{translation initiation sites})} \tag{5.3}$$

Analogously, we compute r_2^+, r_3^+ and r_4^+:

$$r_2^+ := \frac{\#(\text{good stop hints})}{\#(\text{translation termination sites})} \tag{5.4}$$

$$r_3^+ := \frac{\#(\text{good ASS hints})}{\#(\text{acceptor splice sites})} \tag{5.5}$$

$$r_4^+ := \frac{\#(\text{good DSS hints})}{\#(\text{donor splice sites})} \tag{5.6}$$

$$\tag{5.7}$$

Because of the fact that a good exonpart hint ($j = 5$) can have different lengths, the computation here is a little different. Let E be the set of all exons in e500; for an exon $e \in E$ let $\ell(e)$ be its length. The expected number of good exonpart hints given the true parse ψ is

$$\begin{aligned} & \sum_{\substack{i \text{ s.t. base } i \\ \text{is coding in } \psi}} \sum_{\substack{f=(g,d,l,r)\in\mathcal{F}_5 \\ \text{good at } i}} p_5^+(g) r_5^+ \\ = \quad & r_5^+ \sum_{e\in E} \sum_{k=1}^{\ell(e)} k \qquad (5.8) \\ = \quad & r_5^+ \sum_{e\in E} \ell(e)(\ell(e)+1)/2 \end{aligned}$$

Step (5.8) is true because the strand and reading frame of a good exonpart feature at a certain position are fixed by the true parse and a good exonpart hint at position k relative to the exon start can have k possible lengths. The number

$$\text{sql} := \sum_{e\in E} \ell(e)(\ell(e)+1)/2 = 362718220.$$

is the number of possible good exonpart hints. We estimate r_5^+ as

$$r_5^+ := \frac{\#\text{good exonpart hints}}{\text{sql}}. \tag{5.9}$$

The computation for the exon hints ($j = 6$) again is similar to that of the first four types because for a given position at most one good exon hint is possible. The length, reading frame and strand are fixed by the true parse. And there is exactly one position per exon in the training set at which a good exon hint is possible. Therefore

$$r_6^+ := \frac{\#\text{good exon hints}}{\#\text{exons}}. \tag{5.10}$$

Our method for estimating the r_j^- is similar. For each type j we assume that the actually observed number of bad hints equals the expected number of bad hints given the true parse ψ. This yields a formula for r_j^- in each case.

$j = 1$. The expected number of bad start hints can be estimated as follows. For each ATG in the training sequence σ and each reverse complement of ATG which are not translation

starts we have the chance r_1^- of observing a bad start hint. On a given strand in the training set, the string ATG occurred on the average every 65.5 bases as a substring: rate(ATG)=#(ATG)/L=1/65.5. Ignoring the relatively small number of cases where an ATG indeed was a translation start and assuming that the reverse complement has the same distribution, this yields an expected number of bad start hints of $2 \cdot rate(ATG) \cdot L \cdot r_1^-$. Our estimate of r_1^- therefore is

$$r_1^- := \frac{\#(\text{bad start hints})}{2 \cdot \text{rate(ATG)} \cdot L}. \tag{5.11}$$

Similarly, we estimate the numbers for bad stop, ASS and DSS hints.

$$r_2^- := \frac{\#(\text{bad stop hints})}{2 \cdot \text{rate(stop codon)} \cdot L}, \tag{5.12}$$

$$r_3^- := \frac{\#(\text{bad ASS hints})}{2 \cdot \text{rate(AG)} \cdot L}, \tag{5.13}$$

$$r_4^- := \frac{\#(\text{bad DSS hints})}{2 \cdot \text{rate(GT)} \cdot L}. \tag{5.14}$$

$$\tag{5.15}$$

Here, rate(stop codon) = 1/23.4, rate(AG) = 1/13.7 and rate(GT) = 1/19.3 is L divided by the number of occurrences of a stop codon (TGA, TAG, TAA) an AG or a GT in the training set, respectively.
The estimation of r_5^- again is somewhat different because of the length of exonpart hints. For $1 \leq i \leq L, d \in \{\rightarrowtail, \leftarrowtail\}, r \in \{0,1,2\}$ let

$$\gamma(i,d,r) := i - \max\{j \leq i-2 \,|\, \text{stop codon in frame } r \text{ starts at } j \text{ on strand } d \text{ in } \sigma\} \cup \{0\}$$

be the length of the longest exonpart with frame r and strand d observable at position i. And let $\bar{\gamma}$ be the mean of $\gamma(i,d,r)$ over all possible arguments. We computed $\bar{\gamma} \approx 96.7$, i.e. starting from a random position one can go on average about 97 bases until one reaches a stop codon in a given reading frame. The expected number of bad exonpart hints given parse ψ is

$$\begin{aligned}
& \sum_{i=1}^{L} \sum_{g \in G_5} \sum_{d \in \{\rightarrowtail,\leftarrowtail\}} \sum_{r \in \{0,1,2\}} \sum_{l \leq \gamma(i,d,r)} \begin{cases} p_5^-(g) r_5^- & \text{, if } f=(g,d,l,r) \text{ bad at } i; \\ 0 & \text{, if } f=(g,d,l,r) \text{ good at } i. \end{cases} \\
= \; & r_5^- \sum_{i=1}^{L} \sum_{d \in \{\rightarrowtail,\leftarrowtail\}} \sum_{r \in \{0,1,2\}} \sum_{l \leq \gamma(i,d,r)} \left(1 - \begin{cases} 0 & \text{, if } f=(g,d,l,r) \text{ bad at } i; \\ 1 & \text{, if } f=(g,d,l,r) \text{ good at } i. \end{cases} \right) \qquad (5.16) \\
= \; & r_5^- \left(L \cdot 2 \cdot 3 \cdot \bar{\gamma} - \text{sql} \right) \qquad (5.17)
\end{aligned}$$

In (5.16) we have used that the $p_5^-(g)$ sum up to 1 and in (5.17) we plugged in the definition of $\bar{\gamma}$ and have made use of the previously calculated number sql of possible good hints.

This yields the following estimation for r_5^-

$$r_5^- := \frac{\#(\text{bad exonpart hints})}{L \cdot 2 \cdot 3 \cdot \bar{\gamma} - \text{sql}} \tag{5.18}$$

Let the quadruple (i, l, d, r) be an *exon candidate* if an exon hint of length l, strand d and reading frame r is observable at position i in σ. Then we denote with #(exon candidates) the total number of exon candidates in the training set. We counted the number of exon candidates on the forward strand with the computer:

$$\mathrm{E}[\#(\text{exon candidates})] = L \cdot 2.811$$

$$\mathrm{E}[\#(\text{bad exon candidates})] = \mathrm{E}[\#(\text{exon candidates})] - \#(\text{exons})$$

As #(exons) is negligible in comparison with E[#(exon candidates)] we estimate

$$r_6^- := \frac{\#(\text{bad exon hints})}{2.811 \cdot L} \tag{5.19}$$

Below we will also need the probability of not observing a special hint at a certain position, i.e. of observing the hint $\Cap$. This depends much on whether the parse supports a hint at that position. Let a type j be given and let F_u be a random hint of type j at a position where the parse does *not support* a hint of type j. The index u stands for **u**nsupported. In our applications

$$P(F_u = \Cap) \approx 1.$$

for all types. For example, the probability of not observing a start hint at a position different from a translation start is very close to 1, even if a start hint is observable and an 'ATG' ends at that position. The same holds for all other types.
Let F_s be a random hint of type j at a position where type j is **s***upported* by the parse. Although $P(F_s = \Cap)$ is implicitly defined with (5.1), we rather estimate it directly from the data:

$$P(F_s = \Cap) = \begin{cases} \frac{\#(\text{trans. init. sites without hints})}{\#(\text{trans. init. sites})} & \text{, if } F_s \text{ is of type start} \\ \frac{\#(\text{trans. term. sites without hints})}{\#(\text{trans. term. sites})} & \text{, if } F_s \text{ is of type stop} \\ \frac{\#(\text{acceptor sites without hints})}{\#(\text{acceptor sites})} & \text{, if } F_s \text{ is of type ASS} \\ \frac{\#(\text{donor sites without hints})}{\#(\text{donor sites})} & \text{, if } F_s \text{ is of type DSS} \\ \frac{\#(\text{coding bases without hints})}{\#(\text{coding bases})} & \text{, if } F_s \text{ is of type exonpart} \\ \frac{\#(\text{exons without well-positioned hint})}{\#(\text{exons})} & \text{, if } F_s \text{ is of type exon} \end{cases} \tag{5.20}$$

In the last line the expression #(exons without well-positioned hint) denotes the number of exons in the training set where no hint of type exon is at the position supported by the exon. Table 5.1 shows the probabilities $P(F_s = \text{⫛})$.

5.3.4 Impact of the Hints on the Prediction

The Viterbi algorithm chooses the parse with highest a-posteriori probability. The question naturally coming up is: How does a given hint influence the a-posteriori probabilities of parses?

5.1 Definition (bonus)
For $j \in \{1, \ldots, 6\}$ and $g \in G_j, g \neq \textit{Manual}$ let

$$\text{bonus}(g, j) := \frac{p_j^+(g) r_j^+}{p_j^-(g) r_j^-}$$

The case $g = \textit{Manual}$ is excluded because then $p_j^-(g) = 0$ and the denominator vanishes. For the choice $G_j = \{\textit{Protein}, \textit{EST}, \textit{Combined}\}$ the bonuses are listed in Table 5.1. All bonuses are greater than 1. The bonus is the factor by which the emission probability of a special observable hint of type j and grade g is increased when the parse respects the hint as opposed to when the parse does not respect the hint.
Remark: We assumed that only one hint of each type can be observed at each position. However, when the extrinsic information comes from different sources, it can happen that there is more than one hint of a type at a certain position, for example a DSS hint from a protein search and an EST search. In that case we keep only the hint of that grade (i.e. of that source) of which the relationship from good hints to bad hints in the training set was best.

5.2 Definition (malus)
Let $\sigma \in \Sigma^*$ be a DNA sequence, i a position in σ and $j \in \{1, \ldots, 6\}$ be a type. Let F_s be the random hint of type j at position i in a parse that supports type j at position i in σ and let F_u be the hint of type j at position i in a parse that does not support type j at position i in σ. Then

$$\text{malus}(j) := \frac{\text{P}(F_s = \text{⫛})}{\text{P}(F_u = \text{⫛})}$$

5.3 Theorem
Let $\sigma \in \Sigma'^*$ be an input sequence to AUGUSTUS+ of length n that contains no special hint of type $j \in \{1, \ldots, 6\}$ at position h $(1 \leq h \leq n)$ and let $\sigma_f \in \Sigma'^*$ be the same sequence, except that an observable hint $f \neq \text{⫛}$ of type j and grade g is given at position h. Let ψ and ψ_r be two parses of length n with positive a-posteriori probability given emission σ

type	j	r_j^+	r_j^-	bonus			P(F_s =⫛)
				Protein	*EST*	*Combined*	
start	1	0.82	$4.08 \cdot 10^{-4}$	1828	-	2210	0.18
stop	2	0.94	$1.84 \cdot 10^{-4}$	3073	-	7170	0.06
ASS	3	0.32	$2.16 \cdot 10^{-4}$	1252	53	3388	0.68
DSS	4	0.32	$4.46 \cdot 10^{-4}$	786	33	2500	0.68
exonpart	5	$5.6 \cdot 10^{-6}$	$8.22 \cdot 10^{-7}$	17.7	2.5	21.1	0.996
exon	6	0.822	$2.76 \cdot 10^{-5}$	38724	3575	74470	0.178

Table 5.1: This table was constructed using the human training set e500 and all hints from protein, EST and combined EST/protein database search. For all types the *Combined* hints had a better relationship between good and bad hints than the *Protein* hints, which in turn had a better relationship than the *EST* hints.

such that ψ does not respect f and ψ_r respects f. Let $\Phi = \varphi(\mathbf{X}, \mathbf{Y})$ be the random parse of AUGUSTUS+ and let $\Upsilon = \sigma(\mathbf{Y})$ be the random emission.
Then

$$\frac{\mathrm{P}(\Phi = \psi_r \,|\, \Upsilon = \sigma_f)}{\mathrm{P}(\Phi = \psi \,|\, \Upsilon = \sigma_f)} = \frac{\mathrm{bonus}(g, j)}{\mathrm{malus}(j)} \frac{\mathrm{P}(\Phi = \psi_r \,|\, \Upsilon = \sigma)}{\mathrm{P}(\Phi = \psi \,|\, \Upsilon = \sigma)} \tag{5.21}$$

Proof: By the definition of conditional probabilities the claim is equivalent to

$$\frac{\mathrm{P}(\Phi = \psi_r, \Upsilon = \sigma_f)}{\mathrm{P}(\Phi = \psi, \Upsilon = \sigma_f)} = \frac{\mathrm{bonus}(g, j)}{\mathrm{malus}(j)} \frac{\mathrm{P}(\Phi = \psi_r, \Upsilon = \sigma)}{\mathrm{P}(\Phi = \psi, \Upsilon = \sigma)} \tag{5.22}$$

The proof bases on the fact that in the products of transition and emission probabilities of the joint probability of parse and emission only one factor which corresponds to the hint is different. Let $\psi_r = ((x_1, d_1), \ldots, (x_t, d_t))$ and let $y_1, \ldots y_t, y_1^f, \ldots y_t^f \in \Sigma'^*$ be such that $|y_i| = d_i, |y_i^f| = d_i$ $(i = 1, \ldots, t)$ and $y_1 \ldots y_t = \sigma$, $y_1^f \ldots y_t^f = \sigma_f$. Let $y_0 = y_0^f = \varepsilon$. Let $k := \min\{i \,|\, d_1 + \cdots + d_i \geq h\}$ be the step in the parse ψ_r when position h is emitted. Then the products

$$\mathrm{P}(\Phi = \psi_r, \Upsilon = \sigma_f) = \prod_{i=1}^{t} a_{x_{i-1}, x_i} e'_{x_{i-1}, x_i, y_1^f \ldots y_{i-1}^f}(y_i^f) \tag{5.23}$$

and

$$\mathrm{P}(\Phi = \psi_r, \Upsilon = \sigma) = \prod_{i=1}^{t} a_{x_{i-1}, x_i} e'_{x_{i-1}, x_i, y_1 \ldots y_{i-1}}(y_i) \tag{5.24}$$

have identical factors except for $e'_{x_{k-1}, x_k, y_0^f \ldots y_{k-1}^f}(y_k^f)$ and the corresponding factor in the lower product. These emission probabilities themselves are products of a DNA emission probability and hint emission probabilities. The products again differ only in one factor: The probability of emitting the hint, f or ⫛, of type j at position h. In (5.23) this factor

is $p_j^+(g)r_j^+$ as ψ_r respects f and in (5.24) it is $P(F_s = \pitchfork)$, because a hint of type j is supported at position h. Therefore

$$\mathrm{P}(\Phi = \psi_r, \Upsilon = \sigma_f) = \frac{p_j^+(g)r_j^+}{P(F_s = \pitchfork)} \quad \mathrm{P}(\Phi = \psi_r, \Upsilon = \sigma) \tag{5.25}$$

Analogously,

$$\mathrm{P}(\Phi = \psi, \Upsilon = \sigma_f) = \frac{p_j^-(g)r_j^-}{P(F_u = \pitchfork)} \quad \mathrm{P}(\Phi = \psi, \Upsilon = \sigma) \tag{5.26}$$

Combining equations (5.25) and (5.26) and plugging in the definition of bonus and malus yields equation (5.22). □

The factor bonus$(g, j) > 1$ can be thought of a reward on the emission probability a parse gets for respecting a given hint and the factor malus$(j) < 1$ can be thought of a penalty on the emission probability a parse gets at a certain position for introducing an unsupported start codon, stop codon, donor splice site, acceptor splice site or exon. Observe that this bonus is not a bonus on the a-posteriori probability as introducing a hint usually reduces the probability of the emission $\mathrm{P}(\Upsilon = \sigma)$. The next theorem demonstrates how introducing a hint f changes the a-posteriori probabilities. The a-posteriori probabilities of all parses respecting f increase by a constant factor and the a-posteriori probabilities of all parses not respecting f decrease by a different constant factor.

5.4 Theorem

Let $f, g, j, \sigma, \sigma_f, \Phi$ and Υ be as in Theorem 5.3. Let

$$c := \mathrm{bonus}(g, j)/\mathrm{malus}(j) \geq 1, \qquad R := \mathrm{P}(\Phi \text{ respects } f \mid \Upsilon = \sigma)$$

and let

$$m := \frac{1}{1 - R + cR}, \qquad b := cm.$$

Then $b \geq 1$ and $m \leq 1$. For any parse ψ_r which respects f we have

$$\mathrm{P}(\Phi = \psi_r \mid \Upsilon = \sigma_f) = b \cdot \mathrm{P}(\Phi = \psi_r \mid \Upsilon = \sigma), \tag{5.27}$$

and for any parse ψ which does not respect f we have

$$\mathrm{P}(\Phi = \psi \mid \Upsilon = \sigma_f) = m \cdot \mathrm{P}(\Phi = \psi \mid \Upsilon = \sigma). \tag{5.28}$$

Proof: The statement that $c \geq 1$ implies $b \geq 1$ and $m \leq 1$ is trivial given $0 \leq R \leq 1$. Rearrange (5.21) to get

$$\mathrm{P}(\Phi = \psi_r \mid \Upsilon = \sigma_f) \cdot \mathrm{P}(\Phi = \psi \mid \Upsilon = \sigma) = c \cdot \mathrm{P}(\Phi = \psi_r \mid \Upsilon = \sigma) \cdot \mathrm{P}(\Phi = \psi \mid \Upsilon = \sigma_f) \tag{5.29}$$

Summing up this equation over all parses ψ_r which respect f and all parses ψ which do not respect f yields

$$\begin{aligned} & \mathrm{P}(\Phi \text{ respects } f \mid \Upsilon = \sigma_f) \cdot \mathrm{P}(\Phi \text{ respects not } f \mid \Upsilon = \sigma) \\ = \; & c \cdot \mathrm{P}(\Phi \text{ respects } f \mid \Upsilon = \sigma) \cdot \mathrm{P}(\Phi \text{ respects not } f \mid \Upsilon = \sigma_f) \end{aligned} \tag{5.30}$$

Defining $S := \mathrm{P}(\Phi \text{ respects } f \mid \Upsilon = \sigma_f)$ and using $\mathrm{P}(\Phi \text{ respects not } f \mid \Upsilon = \sigma_f) = 1 - S$ and $\mathrm{P}(\Phi \text{ respects not } f \mid \Upsilon = \sigma) = 1 - R$ yields

$$S(1 - R) = cR(1 - S) \tag{5.31}$$

Solving this for S yields

$$S = \frac{cR}{1 - R + cR} \tag{5.32}$$

which we need below. Fixing a parse ψ_r and summing up (5.29) over all parses ψ not respecting f yields

$$\mathrm{P}(\Phi = \psi_r \mid \Upsilon = \sigma_f) \cdot (1 - R) = c \cdot \mathrm{P}(\Phi \text{ respects } f \mid \Upsilon = \sigma) \cdot (1 - S) \tag{5.33}$$

which proves (5.27) using (5.31) and (5.32). Similarly, (5.28) can be shown by summing up (5.29) over all parses ψ_r respecting f.

Chapter 6

Implementation

6.1 The Programs

The program AUGUSTUS has been implemented using the programming language C++ [Str91]. It has been compiled using the compiler gcc (version 3.2) on a PC under Linux and FreeBSD. The architecture of the model and the parameters are read in at run time from configuration files and data files. The configuration files can be manually edited to change the number of states and the possible transitions or meta-parameters like window sizes or the order of a Markov chain. The data files with the parameters of the state models, e.g. the Markov chain transition probabilities, are generated by a separate training program that takes an annotated sequence set in Genbank format and outputs these data files. The size of the source code of the two programs amounts to approximately 16000 lines of code. One obstacle in implementing recursions (2.11) and (2.15) is that the probabilities stored in the Viterbi and forward matrix get too small for using standard C++ floating point types. Instead in AUGUSTUS a real number q is stored in a purpose-build data structure in the form $q = a \cdot 4^n$, where a is of type *double* (8 bytes on my platform), and n is of type *long int* (4 bytes on my platform). This data structure and the standard arithmetic operations on it have been implemented by Emmanouil Stafilarakis.

6.2 Space and Running Time

The Viterbi algorithm and the forward algorithm require almost the same computations. Therefore we always compute the two dynamic programming matrices, the Viterbi matrix and the forward matrix, at the same time. Let m (=47) be the number of states and t be the input sequence length. Then the space required for storing the Viterbi and forward matrix, which have dimensions m and t, is linear in the sequence length:

$$\text{space} = \Theta(mt)$$

Each entry of the dynamic programming matrices takes 8+4 = 12 bytes computer memory on my PC. This totals to theoretical $2 \cdot 47 \cdot 12 \cdot 1000$bytes ≈ 1.08 mega bytes per kilo base sequence length for the storage of both matrices. Indeed, in practice the amount of computer memory needed (determined by regression on experimental data) is approximately 20 mega bytes + 1.08 mega bytes per kilo base sequence length. Where the constant 20 mega bytes are needed to store the model parameters and the program itself. This means for example that an input sequence of length 300000 needs about 344 mega bytes memory. AUGUSTUS has an option to cut the input sequence in pieces when it is too long for the amount of computer memory available. The parting points are then assumed to be in the intergenic region and the predictions for the pieces are composed afterwards. For example the 2.9 Mb of the Adh region were cut input pieces of length 400 Kb because AUGUSTUS was run on a PC with 512 MB RAM.

The running time of the Viterbi algorithm (and the forward algorithm) is dominated by the time needed to construct the dynamic programming matrix. For each position $1 \leq i \leq t$ in the input sequence and each state $q \in Q$ a list of all combinations of possible predecessor states q' and their ending positions l' needs to be searched (see the recursion (2.11)) and the probability of the emission needs to be calculated. Let us call this list the predecessor list. The probability of each emission can be calculated in *constant time*, when some results are stored after their calculation: The only submodels which have no maximal length are Markov chains. The probability of a piece a of sequence in a Markov chain model can be calculated in constant time when the probability of a piece b with endpoints differing by at most 1 from those of a is known. We process the list of predecessor positions l' from right to left and save the Markov chain probabilities intermediately. Therefore the running time required for position i and state q is proportional to the size of the predecessor list. For all states, except the $\mathrm{I}^j_{\mathrm{short}}$ and the exon states, the size of this list is at most m. The are at most m such states. For the intron states $\mathrm{I}^j_{\mathrm{short}}$ the size of this list is at most m. For an exon state the size of this list is at most proportional to ORFlength, where ORFlength is the distance of i to the next stop codon left of position i with respect to the reading frame given by the exon state. This is so because AUGUSTUS assumes that the exons contain no in-frame stop codons and therefore it may truncate the predecessor list beyond the position of the first stop codon. In the worst case there may be no in-frame stop codons and every of the $i-1$ positions before i may be the end position of the previous state. This means a worst-case running time which is quadratic in t. Fortunately, on the average ORFlength is short. Under the assumption that the bases of the input sequence are independent and uniformly distributed on $\{A, C, G, T\}$ the expectation of ORFlength would be E[ORFlength] $\approx$ 64 bases. Indeed, in the human test set h178 the mean of ORFlength was about 89 (the longest ORFlength was 2263), which is significantly smaller

than the constant d for the introns. The expected running time is

$$\text{Expected Running Time} = O\left(mt(m + d + \text{E[ORFlength]})\right)$$

which is also linear in t under reasonable assumptions on the mean length of open reading frames. The actual running time of AUGUSTUS on the sequences of h178 on a single-processor PC with 2.4 GHz is shown in Figure 6.1.

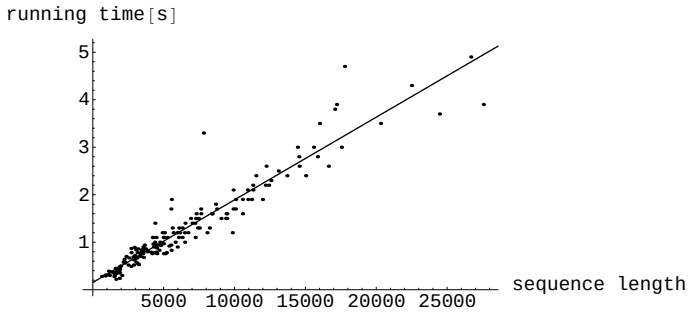

Figure 6.1: Running time of AUGUSTUS on a set of 178 human sequences. The regression line has slope 1.74 s / 10000 bp.

6.3 Output of AUGUSTUS

AUGUSTUS takes the input DNA sequence in (multiple) *FastA* format, which is a simple standard format for biological sequences. It contains for each sequence a '>' followed by the sequence name and in the subsequent lines the sequence itself. AUGUSTUS outputs its result in the 'General Feature Format' (GFF) proposed by Richard Durbin and David Haussler. An example output is shown below.

```
# This output was generated with AUGUSTUS (version 1.0).
# No extrinsic information on sequences given.
#
# ----- prediction on sequence number 1 (length = 5200, name = examplesequence) -----
#
# Predicted genes for sequence number 1 on both strands
examplesequence AUGUSTUS   stop_codon      219     221     .       -       0       "g1"
examplesequence AUGUSTUS   single          219     1850    .       -       0       "g1"
examplesequence AUGUSTUS   start_codon     1848    1850    .       -       0       "g1"
# protein sequence = [MRPSCCEGYEGSVENCKPVCRQQCPQHGFCSSPNTCSCNAGYGGIDCHPVCPTVCGK
# NEFCDRPGVCSCQNGYKRTSPSDNCLPVCEKECGHHSFCSEPGKCECEPGYEKVGNGTVFPDGYKNNSNGNCSPICP
# KDCGQNSRCVRPGVCECENGYAGDDGGTNCRPVCSTCPENGLCLSPGVCVCKPGYVMRNDLCQPHCEKCSDNAHCVA
# PNQCECFPGYESSGADKKCVPKCSKGCTNGFCFAPETCVCSIGYQMGPNQVCEPKCSLNCVHGKCTSPETCTCDPGY
# RFKDNSHHECDPICDSGCSNGHCVAPNFCICHDGYQLNSTNPVTSMCQPICKGCQFGDCVAPNVCECNVGYENINGL
# CELQTTTDSYEYSTTTVELQSSTVDPQLQTSTSEVPHSNCTAGCMCWIEYDGMGTFNTAKCAKICVDPQDKPCLNLD
# NCQCDLSSGQLVCQEDSDMDYSGENSRYVCHILPEQGARSEAEVRIPDRTGSSNKWMIIVGSCAGMIIGVAATIIGI
# KYYRRSTSRRNFEAEEAIVECDFE]
examplesequence AUGUSTUS   start_codon     2869    2871    .       +       0       "g2"
examplesequence AUGUSTUS   initial         2869    3056    .       +       0       "g2"
```

```
examplesequence AUGUSTUS   intron          3057    3200    .       +       .       "g2"
examplesequence AUGUSTUS   internal        3201    3447    .       +       1       "g2"
examplesequence AUGUSTUS   intron          3448    3511    .       +       .       "g2"
examplesequence AUGUSTUS   internal        3512    3616    .       +       0       "g2"
examplesequence AUGUSTUS   intron          3617    3703    .       +       .       "g2"
examplesequence AUGUSTUS   internal        3704    4250    .       +       0       "g2"
examplesequence AUGUSTUS   intron          4251    4316    .       +       .       "g2"
examplesequence AUGUSTUS   internal        4317    4583    .       +       2       "g2"
examplesequence AUGUSTUS   intron          4584    4724    .       +       .       "g2"
examplesequence AUGUSTUS   terminal        4725    4888    .       +       2       "g2"
examplesequence AUGUSTUS   stop_codon      4886    4888    .       +       0       "g2"
# protein sequence = [MVLITLTLVSLVVGLLYAVLVWNYDYWRKRGVPGPKPKLLCGNYPNMFTMKRHAIYD
# LDDIYRQYKNKYDAVGIFGSRSPQLLVINPALARRVFVSNFKNFHDNEIAKNIDEKTDFIFANNPFSLTGEKWKTRR
# ADVTPGLTMGRIKTVYPVTNKVCQKLTEWVEKQLRLGSKDGIDAKHMSLCFTTEMVTDCVLGLGAESFSDKPTPIMS
# KINDLFNQPWTFVLFFILTSSFPSLSHLIKLRFVPVDVERFFVDLMGSAVETRRAQLAAGKQFERSDFLDYILQLGE
# KRNLDNRQLLAYSMTFLLDGFETTATVLAHILLNLGRNKEAQNLLREEIRSHLQDGTIAFEKLSDLPYLDACVQETI
# RLFPPGFMSNKLCTESIEIPNKEGPNFVVEKGTTVVVPHYCFMLDEEFFPNPQSFQPERFLEPDAAKTFRERGVFMG
# FGDGPRVCIGMRFATVQIKAAIVELISKFNVKINDKTRKDNDYEPGQIITGLRGGIWLDLEKL]
```

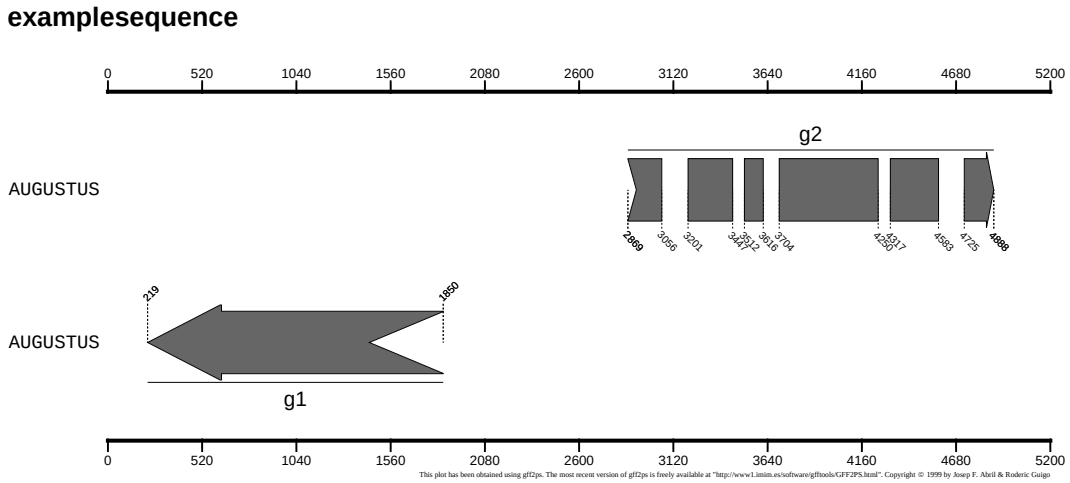

Figure 6.2: Visualization of the predicted genes in the example output.

This output was generated using AUGUSTUS on a part of the *Drosophila* Adh region with the command line parameters '−−strand=both −−introns=on'. The first one tells AUGUSTUS to predict genes on both strands, the latter tells it to additionally output the intron positions. The lines starting with a '#' character are comments in the GFF. The other lines contain the information about the gene prediction and can be used as input to a visualization tool like *gff2ps* [AG00]. Each non-comment line corresponds to a predicted exon, intron, start codon or stop codon. In the first column the sequence name (examplesequence) is given, the second column specifies the source of this annotation (AUGUSTUS) the third column contains the 'feature' name: 'single', 'initial', 'internal' and 'terminal' stand for a single, initial, internal or terminal exon. 'start_codon' and 'stop_codon' stand for the first and the last codon of the gene. 'intron' stands for an intron. The 4th and the 5th column specify the beginning and end position of the feature.

The 7th column has a '+' for the forward stand and a '-' for the reverse strand. The 8th column specifies for exons the reading frame position. It is the number of bases of the exon (0, 1 or 2) upstream of the most upstream codon boundary of the exon. The last column contains a name for the gene, which is a 'g' plus a consecutive number.
Figure 6.2 shows a graphical representation of the predicted genes made with gff2ps. The single gene on the reverse strand is shown in the lower row and the multiple exon gene on the forward strand is shown in the top row.

6.4 Web Server

AUGUSTUS has been made available publicly in the form of a web interface. This web interface has been set up by Rasmus Steinkamp on the *Göttingen Bioinformatics Compute Server* (GOBICS). A screenshot of the web page `http://augustus.gobics.de/submission` showing the web interface is shown in Figure 6.3.

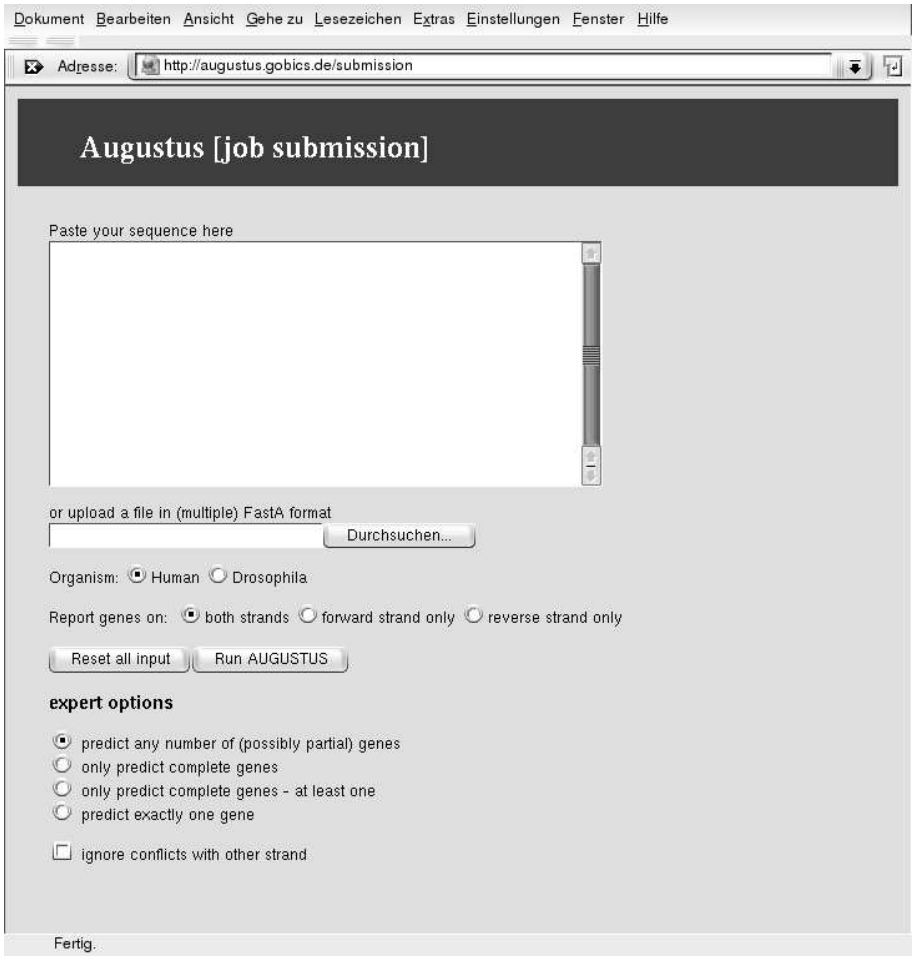

Figure 6.3: Screenshot of the AUGUSTUS web interface.

The user may either paste her input DNA sequence into a text field or upload the file

directly from a directory on her computer. She may choose between the options mentioned in section 3.5 and receives her results both as a text file in GFF and as a postscript picture similar to that in 6.2.
Furthermore, the executable program AUGUSTUS as well as all training sequences are freely available from `http://augustus.gobics.de`.

Chapter 7

Results of AUGUSTUS

In this chapter we report estimates about the reliability of AUGUSTUS' predictions. These estimates were, as it is customary, retrieved by comparing the predictions to the annotation for a set of sequences where the annotation is relatively certain to be true. Such a set is called a *test set*. Out test sets are described in section 7.1. The parameters of AUGUSTUS were trained using *training sets*, which contain other genes than the test sets. They are described in section 7.2. In section 7.3 we describe the accuracy results of AUGUSTUS, AUGUSTUS+ and variants of the model for different test sets and when using extrinsic information from different sources.

7.1 Test Sets

We tested AUGUSTUS on four data sets which we call fly100, adh222, h178 and sag178.
fly100 is a set of 100 sequences of the fruit fly *Drosophila melanogaster* which has served as a model organism for genetic studies for almost a century [SGH+98]. The annotation of each sequence contains one gene which is on the forward strand. 18 of the 100 genes were single exon genes. The mean sequence length is 16.1 kilo bases (shortest 2, longest 104 kilo bases). The sequences were retrieved from FlyBase and have been filtered for annotation errors and redundancies as described in section 7.2.
adh222 is a single sequence of *Drosophila melanogaster* and 2.9Mb long. It is a well-characterized sequence contig from the Adh region and has been used in the Genome Annotation Assessment Project (GASP) [RGH+00]. They constructed two sets of annotations. The first, smaller set, called std1, was chosen so that the genes in it are likely to be annotated correctly and the second, larger set, called std3, was chosen to be as complete as possible 'while maintaining some confidence' about the correctness. In the corrected version std1 contains 38 genes with a total of 111 exons and std3 contains 222 genes with a total of 909 exons. The genes lie on both strands. Both the authors of GENIE and of GENEID [PEG00] have used these two annotation sets for testing their programs. It

should be noted that std1 was chosen to contain only splice sites with a high score in a neural network model used in GENIE and provided by M. Reese.

h178 is a set of 178 human genomic sequences with one complete gene each. Each contains one gene and a little flanking DNA. The sequences are from EMBL, were compiled by [GAA$^+$00] and have also been used by the author of GENSCAN for evaluation [YLB01]. The mean sequence length is 7169 bases (shortest 622, longest 86640 bases). This set is known to contain three annotated genes which are very likely, if not known, to be wrong. But we kept them for reasons of comparability.

sag178 is a set of 43 sequences with 178 human genes on both strands. These sequences were also taken from [GAA$^+$00] and are semi-artificial in the following sense. Guigó et al took the 178 sequences from h178 and generated long intergenic regions randomly using a Markov model of order 5. They write 'Some of the resulting parameters, such as average G+C content of 40%, a gene every 43Kb, and a coding density of 2.3% are in agreement with that for the overall human genome'. 40 of the 178 genes were single exon genes. The mean sequence length is 177 kilo bases (shortest 70, longest 282 kilo bases) and each sequence contained on the average 4.1 genes.

7.2 Training Sets

The training set for the human version of the program was retrieved in October 2002 from Genbank. Sequences with inconsistent notation were deleted as well as sequences that were overlapping with a sequence in one of the human test sets. This was done using blastn [AGM$^+$90] with an e-value cutoff of 10^{-100} (blastall -p blastn ... -S 1 -G 9 -q -9 -e 1e-100). The rest was cleaned for redundancies and 1284 sequences with one gene each remained. Additionally, we use for the human parameter set the splice sites from 11739 human introns that were each not contained in the test sets (data originally retrieved from http://genomic.sanger.ac.uk/spldb/HumanCanonicalSites.ESTsupp). For the *Drosophila* training and test set we took single gene sequences from FlyBase in December 2001. These were cleaned for genes with known alternative splicing, incomplete annotation, in-frame stop codons, non-canonical splice sites and for redundancies within the data set. The resulting 420 sequences were randomly divided into a training set of 320 and a test set of 100 sequences, again with no BLAST match with e-value smaller than 10^{-100} between the data sets. For the runs on the adh222 test set, we took these 420 *Drosophila* sequences and removed those 20 sequences that had a blastn hit with e-value less than 10^{-10} when run on the Adh sequence. Except for the different training sets, the same parameters were used for training and testing the two test sets for each species. For comparison: The training set of GENSCAN consists of 380 single gene sequences plus additional unpublished 1619

cDNA sequences. Comparing the genomic sequences of h178 and the 380 training genes of GENSCAN shows that 91 of the 178 test genes have a blastn hit in GENSCANs training set with e-value below 1e-100, most of them because the same gene is annotated. The data sets used here can be downloaded from `http://augustus.gobics.de/datasets/`. For training the additional parameters of AUGUSTUS+ $(r_j^+, r_j^-, p_j^+(g), p_j^-(g))$ we also used a training set. For *Drosophila* this training set was identical to the training set of AUGUSTUS. For the human version this was a randomly chosen subset, **e500** , of 500 sequences of the training set of the human version of AUGUSTUS. We chose to restrict ourselves to this subset because the BLAST searches in the protein and EST databases were time consuming.

7.3 Accuracy

We measured the gene prediction accuracy with the customary measures, sensitivity and specificity. For a predicted feature (coding base, exon, gene) the *sensitivity* is defined as the number of correctly predicted features (= true positives) divided by the number of annotated features (= actual positives)

$$\text{sensitivity} = \frac{\text{true positives}}{\text{actual positives}}.$$

The *specificity* is the number of correctly predicted features divided by the number of predicted features (= predicted positives).

$$\text{specificity} = \frac{\text{true positives}}{\text{predicted positives}}.$$

A predicted exon is considered correct if both splice sites are at the annotated position of an exon. A predicted gene is considered correct if all exons are correctly predicted and no additional exons are predicted. Predicted partial genes were counted as predicted genes. For each data set these measures were computed globally (once for all sequences together) and in sag178 and adh222 the forward and reverse strands were treated as different sequences.

7.3.1 Comparison to Other Programs

For comparison of the ab-initio model we used GENSCAN (version 1.0), GENEID and GENIE. We took GENSCAN as it is the most commonly used gene prediction program and as it is considered one of the best programs for humans. Also our HMM is similar to that of GENSCAN. GENSCAN was run using its human parameter set for both human and *Drosophila* as recommended. We used GENEID (version 1.1) as there is a special *Drosophila* parameter set available for it and as it uses a different approach not modeling the lengths. GENEID first finds splice site candidates, then exon candidates

fly100		AUGUSTUS	GENSCAN	GENEID	AUGUSTUS+ *Protein*	AUGUSTUS+ *EST*	AUGUSTUS+ *Combined*	AUGUSTUS+ *Protein*, *EST* and *Combined*	GENOMESCAN
base	sn	97%	97%	95%	99.6%	98%	98%	99.7%	97%
	sp	59%	33%	53%	59%	59%	59%	59%	55%
exon	sn	80%	68%	65%	94%	82%	88%	94%	81%
	sp	49%	22%	39%	54%	49%	52%	54%	46%
gene	sn	53%	37%	31%	77%	56%	72%	78%	61%
	sp	27%	10%	14%	37%	29%	35%	37%	29%

Table 7.1: Accuracy results on *Drosophila* data set fly100. Only genes on the forward strand were considered. A part of the 'false' positives accounting for the low specificity of all methods probably can be attributed to non-annotated genes in the sequences.

using the splice site candidates and then genes using the exon candidates. GENEID was run using the parameter sets human3iso.param and dros.param, respectively. In one case we also compare to GENIE, because this program compared favorably to the other ab initio programs in the GASP experiment. GENSCAN and GENEID were downloaded from the Internet. For comparison of AUGUSTUS+ to a program which also uses extrinsic information we used GENOMESCAN, which was reported to be superior to the spliced-alignment-programs GENEWISE and PROCRUSTES on dataset h178 if only protein matches with a BLAST p-value above 10^{-120} were used [YLB01]. The Genbank gene annotation (build 25 to build 33) included predictions of GENOMESCAN. GENOMESCAN is available through a web interface which we used for our own test runs of it in December 2003.

AUGUSTUS

Tables 7.1 to 7.4 show a summary of the results of the programs on the test sets. On the *Drosophila* test sets (Tables 7.1 and 7.2) AUGUSTUS outperforms the three other ab-initio programs on each of the three levels.

On data set fly100 it predicts 53% of the genes correctly, GENSCAN and GENEID only 37% and 31%, respectively. More than 3 out of 4 exons predicted by GENSCAN in fly100 are not annotated. Even when taking into account that those sequences may contain non-annotated genes, GENSCAN is likely to predict many more false exons than GENEID and AUGUSTUS. GENSCAN was not run on the Adh region as it required too much computer memory. The test set adh222 is a more realistic test set for gene prediction programs as it is a long relatively well-annotated sequence with a large number of genes on both strands. Here, AUGUSTUS has an exceptionally good gene level sensitivity of 68%. However, as

adh222		AUGUSTUS	GENEID	GENIE
base	sn*	98%	96%	96%
	sp*	93%	92%	92%
exon	sn*	85%	71%	70%
	sp*	65%	62%	57%
gene	sn*	68%	47%	40%
	sp*	38%	33%	29%

Table 7.2: Accuracy results on *Drosophila* data set adh222. The asterisk (*) denotes that sensitivity and specificity were measured using two different sets of annotations. The sensitivity refers to std1 and the specificity refers to std3. The values for GENIE are taken from [RGH+00]

h178		AUGUSTUS	GENSCAN	GENEID	AUGUSTUS+ *Protein*	AUGUSTUS+ *EST*	AUGUSTUS+ *Combined*	AUGUSTUS+ *Protein*, *EST* and *Combined*	GENOMESCAN
base	sn	93%	97%	89%	98%	94%	98%	99%	98%
	sp	90%	86%	91%	92%	89%	93%	94%	92%
exon	sn	80%	83%	66%	88%	81%	92%	93%	89%
	sp	81%	75%	75%	86%	76%	89%	89%	86%
gene	sn	48%	40%	14%	62%	40%	71%	73%	63%
	sp	47%	36%	13%	59%	39%	68%	70%	60%

Table 7.3: Accuracy results on human data set h178.

the sample for the sensitivity consisted only of 38 genes, we conducted McNemar's test, to check whether AUGUSTUS and the second best program, GENEID, actually could have the same gene level sensitivity. GENEID predicted 1 gene correctly where AUGUSTUS failed, and AUGUSTUS predicted 9 genes correctly where GENEID failed. This yields a p-value of 0.0215. Thus, at a confidence level of 5% the hypothesis can be rejected that AUGUSTUS and GENEID have the same gene level sensitivity.

On the human data set h178 with short single gene sequences (Table 7.3) AUGUSTUS and GENSCAN are similarly accurate with respect to the mean of sensitivity and specificity on the base and exon level. GENSCAN is more sensitive, AUGUSTUS more specific. GENEID is worse here. AUGUSTUS predicts more genes (85) correctly than GENSCAN (71) and GENEID (25).

On the long sequences in sag178 containing the same genes (Table 7.4) AUGUSTUS predicts still 41% of the annotated genes correctly. This is remarkable as it has been reported

sag178		AUGUSTUS	GENSCAN	GENEID	AUGUSTUS+ *Protein*	AUGUSTUS+ *EST*	AUGUSTUS+ *Combined*	AUGUSTUS+ *Protein*, *EST* and *Combined*
base	sn	93%	94%	89%	97%	93%	95%	98%
	sp	81%	64%	78%	86%	83%	86%	90%
exon	sn	79%	68%	67%	85%	80%	84%	87%
	sp	71%	45%	60%	80%	72%	79%	84%
gene	sn	41%	18%	17%	55%	44%	53%	62%
	sp	36%	14%	17%	51%	39%	49%	57%

Table 7.4: Accuracy results on human data set sag178. The gene level accuracy measures of GENSCAN on these long genomic sequences are similar to those reported by Korf et al. for long mouse sequences with mean length 112 Kb (sensitivity: 15%-17%, specificity: 11%-16%) [KFDB01].

that 'Computational gene finders produce acceptable predictions of the exonic structure of the genes when analyzing single gene sequences with very little flanking intergenic sequence, but are unable to correctly infer the exonic structure of multi gene genomic sequences.' [GAA+00]. GENSCAN and GENEID predict only 18% and 17% of the genes correctly. GENSCAN here often 'adds' false short exons to an annotated gene and is therefore much less specific than GENEID and AUGUSTUS.

AUGUSTUS+

AUGUSTUS+ has been tested on the test sets fly100, h178 and sag178. On each test set we tested four different settings for the extrinsic information.

- *Protein*: AGRIPPA was used to generate the hints using only the nr protein database.
- *EST*: AGRIPPA was used to generate the hints using only the EST database.
- *Combined*: AGRIPPA was used to generate the hints making a combined EST and protein database search.
- *Protein*, *EST* and *Combined*: All above hints were used but redundant hints were deleted (see remark in section 5.3.4).

On test sets fly100 and h178 we also ran GENOMESCAN. GENOMESCAN must be given at least one amino acid sequence which is similar to the DNA input sequence. We gave GENOMESCAN all complete amino acid sequences of proteins that were used by AGRIPPA for the construction of hints of type *Protein* and *Combined*. Therefore the

protein information available to GENOMESCAN was equal to or a superset of the protein information available to AUGUSTUS+ in the three settings which use proteins (all but the *EST* setting). For h178 there were on average 2.2 informative amino acid sequences per DNA input sequence given to GENOMESCAN. For one of the 178 sequences there was no available protein information. In that case we used GENSCAN instead of GENOMESCAN. For fly100 there were on average 4.6 informative amino acid sequences per DNA input sequence given to GENOMESCAN. Here, all sequences had at least one similar amino acid sequence. GENOMESCAN could not be run on the test set sag178 because a part of those sequences apparently exceed the length limit of the GENOMESCAN server. Table 7.1 shows that the use of ESTs alone can increase the accuracy of AUGUSTUS for *Drosophila* sequences. The sensitivity on the base, exon and gene level increase by 1, 2 and 3 percent points, respectively. And the specificity on the base and exon level stays the same and increases on the gene level by 2 percent points. This is in contrast to the results of A. Krogh with HMMGENE on *Drosophila* sequences who writes 'The specificity drops more than the sensitivity increases when ESTs are used' [Kro00]. Restricting the information from the ESTs to that information, which is additionally supported by proteins as done in the *Combined* setting yields more accurate results than using unfiltered EST information. Using only proteins as source of extrinsic information is for *Drosophila* better than combining EST with protein information, though. The best results for *Drosophila* were achieved in the fourth setting when all information was used together. Then 99.7% of the coding bases were found and 94% of the exons and 78% of the genes were predicted correctly. The performance of GENOMESCAN is worse than that of AUGUSTUS+ in any of the settings except the *EST* setting, where it is more sensitive on the gene level but less specific on the base and exon level.

On the human test set h178 the ESTs were not helpful unless when combined with proteins (Table 7.3). The accuracy results for the *Combined* setting were even much better than those for the *Protein* setting. Again, it was best to use all available hints together. In the setting *Protein*, *EST* and *Combined* 93% of the exons and 73% of the genes were correctly predicted. The accuracy results of GENOMESCAN are very similar to those of AUGUSTUS+ in the *Protein* setting.

On the test set sag178 the EST hints are again helpful for increasing the accuracy of AUGUSTUS. This is astonishing because one might suspect that for sag178 the same hints about the genes should be found as in h178. After all they contain the same genomic sequences, sag178 just contains additional random DNA. However, it turns out that AGRIPPA finds more than 3 times as many hints for test set h178 than for test set sag178. The *Protein* setting here is somewhat better than the *Combined* setting and the best results are again achieved when using all available hints in the setting *Protein*, *EST* and *Combined*.

h178		using moderately similar proteins, e-value $\geq 10^{-30}$		using strongly similar proteins, e-value $< 10^{-30}$	
		AUGUSTUS+	GENOMESCAN	AUGUSTUS+	GENOMESCAN
base	sn	95%	95%	98%	97%
	sp	90%	91%	92%	92%
exon	sn	85%	85%	88%	86%
	sp	84%	83%	85%	84%
gene	sn	54%	51%	64%	61%
	sp	50%	49%	60%	58%

Table 7.5: Accuracy results on subsets of the human data set h178 when either moderately similar amino acid sequences are used (148 sequences) or strongly similar amino acid sequences are used (131 sequences).

When using extrinsic information the accuracy strongly depends on the quality and quantity of the available extrinsic information. As an extreme case consider the case when the amino acid sequence of a gene in the input DNA sequence is known. In such a case a program based on spliced alignment is likely to yield better results than an extrinsic method based on the BLAST search tool. As reported in [GAA+00] and [YLB01] the accuracy of gene finders using protein similarity information increases with the similarity measured by the BLAST e-value. For that reason we examined the accuracy of AUGUSTUS+ in the *Protein* sequences for two different subsets of h178. One subset consisted only of sequences which had a protein match with a BLAST e-value between 10^{-30} and 10. There were 148 such sequences. On these sequences AUGUSTUS+ and GENOMESCAN were tested using only the moderately similar proteins with a BLAST value in that range. A second subset consisted only of those 131 sequences which had a protein match with a BLAST e-value below 10^{-30}. For this set the two programs were given only the strongly similar proteins with a BLAST e-value below 10^{-30}. The results are shown in Table 7.5. AUGUSTUS+ and GENOMESCAN perform very similar on both of the sets when compared to each other. The gene sensitivity of AUGUSTUS+ is somewhat higher than that of GENOMESCAN in both cases. Indeed, both programs perform better when using strongly similar proteins but even the results for only moderately similar proteins are significantly better than the ab-initio results.

7.3.2 Comparison to Variants of AUGUSTUS

In order to find out to which extent the new methods or submodels contribute to the accuracy of AUGUSTUS, we compared AUGUSTUS to versions of AUGUSTUS where one or more feature (method or submodel) had been changed. We did this separately for

feature	human	fly
intron length model	0.3	3.4
initial pattern	1.6	1.0
similarity-based weighting	1.0	1.0
IMM	1.8	0.0
internal 3' content model	0.8	n.a.
translation initiation motif	0.1	1.9
all MM of order 4 instead of 5	2.2	0.2

Table 7.6: The relative mean improvement of sensitivity and specificity in percent on exon and gene level caused by different features of the program. The largest increase in accuracy through a single feature is attributed to the new intron length model, but only for *Drosophila*.

human and *Drosophila* but summarized the results for the two datasets for each species. In particular, we use an – admittedly – coarse measure, the mean increase in sensitivity and specificity on the exon and gene level when the feature is used as compared to when the feature is left out or changed. For example, let $\Delta\text{sn}^i_{\text{exon}}$ be the difference between the sensitivity on the exon level on dataset $i \in \{1, 2\}$ of AUGUSTUS and AUGUSTUS with some feature changed. We weighted the two datasets for each species with the number of annotated genes n_1 and n_2 in the two datasets used to determine the accuracy measure, here the sensitivity. Then $\Delta\text{sn}_{\text{exon}} = (n_1 \cdot \Delta\text{sn}^1_{\text{exon}} + n_2 \cdot \Delta\text{sn}^2_{\text{exon}})/(n_1 + n_2)$ denotes the mean increase in exon sensitivity. We use

$$r := (\Delta\text{sn}_{\text{exon}} + \Delta\text{sp}_{\text{exon}} + \Delta\text{sn}_{\text{gene}} + \Delta\text{sp}_{\text{gene}})/4$$

as a measure to give the reader an idea of the relevance of the feature of the model.
Table 7.6 shows for a selected number of features the relative improvement r. The first line refers to a version of AUGUSTUS, where the intron length was modeled using a shifted geometric distribution with minimum length 48 and the parameter estimated with the maximum likelihood method. In particular the relative improvement for *Drosophila* sequences achieved by our intron length model was 3.4%. The second line refers to the version of AUGUSTUS, where only the initial pattern model was left out, i.e. the start codon model or the ASS model is directly followed by a Markov content model. The third line refers to the version of AUGUSTUS where the donor splice site model simply uses the empirical distribution of the patterns (with pseudo counts). The fourth line refers to the version where all IMMs were substituted by Markov models of the same order. This mostly effects the internal 3' content model in the human version. The fifth line refers to the version of AUGUSTUS where the internal 3' content model was left out. The sixth

line refers to a version of AUGUSTUS with a missing translation initiation motif. The last line of Table 7.6 refers to the version of AUGUSTUS where the Markov models for exons, introns and intergenic region are 5th order Markov models. The largest improvement through a single new feature is obtained for *Drosophila* with the introduction of the new intron length model.

We examined whether the improvement in exon sensitivity for *Drosophila* by introducing the new intron length model might by explained by simple chance. For each of the exons of the 138 genes used to calculate the two exon sensitivities for *Drosophila* we observe two dependent Bernoulli-random variables determining whether it was correctly predicted in the two runs, with or without the feature. McNemar's test for dependent samples yielded a p-value of 0.000034, so that an improvement simply by chance can be ruled out in this case.

The reason that the new intron model does not improve much the predictions for humans can be explained by the fact that short human introns have a much less characteristic length than short *Drosophila* introns.

It should be noted that we have tried a large number of other ideas for the improvement of the model, which turned out to be in vain and are not mentioned in this thesis.

7.3.3 Discussion

The fact that content models of order 4 yield better accuracy results than those of order 5 might be astonishing, as there are enough training data for training models of order 5, and models of higher order model the real distribution more accurately than models of lower order. We conjecture the following explanation for it. In theory, a perfect program should – like the transcription and translation apparatus of the cell – consider the biological signals for prediction, instead of statistical features of the coding and non-coding sequences. For imperfect, current state-of-the-art programs, taking these statistical features into account by using content models helps improving accuracy. However, not rarely the wrong content model yields a higher probability for a stretch of sequence than the correct one, e.g. an untypical short exon gets a larger probability in the non-coding model or a stretch of non-coding sequence gets a higher probability in an exon model. Our observation is that the higher the order of the Markov chain of the content models are, the larger are the differences in the probabilities that a stretch of sequence gets in the coding model versus the non-coding model. A higher order Markov chain may more often correctly classify sequences as coding or non-coding but *if* it misclassifies a sequence it tends to be wider off the correct classification. This means that when using a higher order Markov chain the 'decisions' are made to a greater extent by the content models than by the signal models and errors of the content models have a lower chance of being corrected by the signal

models.

AUGUSTUS is more accurate in many test settings than well known gene finding programs. Especially, on *Drosophila* sequences the advantage of AUGUSTUS over the other programs is significant. As a general tendency, AUGUSTUS tends to have a relatively large gene level accuracy. AUGUSTUS performs relatively well on the task of assembling exons to genes because programs with a similar exon level accuracy often have a lower gene level accuracy. This means those programs more often combine the exons to a wrong gene structure for example by splitting or joining genes. The task of assembling exon candidates to a gene structure may become more important in future. With the growing number of sequenced species also the possibilities of finding approximate possible exons by cross-species alignments of homologue genomic sequences grows. This leaves the task of assembling possible exons to genes. We believe that with our probabilistic model for integrating extrinsic information we have a flexible basis for making use of cross-species alignments, e.g. with DIALIGN 2 [Mor99].

Codonusage in the human training set

aa:freq	relative synonymous codon frequencies					
G :0.0722	GGA:0.19	GGC:0.4	GGG:0.255	GGT:0.155		
D :0.0442	GAC:0.609	GAT:0.391				
E :0.065	GAA:0.326	GAG:0.674				
R :0.0598	AGA:0.143	AGG:0.183	CGA:0.0921	CGC:0.267	CGG:0.235	CGT:0.0808
K :0.0518	AAA:0.35	AAG:0.65				
N :0.0333	AAC:0.603	AAT:0.397				
Q :0.0453	CAA:0.207	CAG:0.793				
S :0.0791	AGC:0.282	AGT:0.132	TCA:0.114	TCC:0.243	TCG:0.0741	TCT:0.155
T :0.0513	ACA:0.231	ACC:0.42	ACG:0.136	ACT:0.214		
A :0.0785	GCA:0.177	GCC:0.453	GCG:0.14	GCT:0.231		
V :0.0622	GTA:0.087	GTC:0.254	GTG:0.519	GTT:0.139		
L :0.1024	CTA:0.0583	CTC:0.217	CTG:0.471	CTT:0.102	TTA:0.0475	TTG:0.105
I :0.0414	ATA:0.116	ATC:0.569	ATT:0.314			
F :0.0377	TTC:0.62	TTT:0.38				
Y :0.0278	TAC:0.637	TAT:0.363				
W:0.0131	TGG:1					
H :0.0241	CAC:0.663	CAT:0.337				
M:0.0210	ATG:1					
C :0.0238	TGC:0.622	TGT:0.378				
P :0.0651	CCA:0.228	CCC:0.369	CCG:0.146	CCT:0.257		

Codonusage of nucleotides 10 to 24 of the genes in the human training set

aa:freq	relative synonymous codon frequencies					
G :0.0672	GGA:0.167	GGC:0.387	GGG:0.299	GGT:0.148		
D :0.0283	GAC:0.632	GAT:0.368				
E :0.0453	GAA:0.316	GAG:0.684				
R :0.0646	AGA:0.137	AGG:0.166	CGA:0.101	CGC:0.251	CGG:0.272	CGT:0.0723
K :0.0481	AAA:0.278	AAG:0.722				
N :0.0227	AAC:0.719	AAT:0.281				
Q :0.0426	CAA:0.237	CAG:0.763				
S :0.1009	AGC:0.248	AGT:0.0957	TCA:0.12	TCC:0.279	TCG:0.12	TCT:0.136
T :0.0573	ACA:0.25	ACC:0.418	ACG:0.147	ACT:0.185		
A :0.0964	GCA:0.153	GCC:0.354	GCG:0.239	GCT:0.254		
V :0.0534	GTA:0.0496	GTC:0.297	GTG:0.51	GTT:0.143		
L :0.14	CTA:0.0456	CTC:0.255	CTG:0.479	CTT:0.0934	TTA:0.0334	TTG:0.0934
I :0.0292	ATA:0.138	ATC:0.644	ATT:0.218			
F :0.0306	TTC:0.65	TTT:0.35				
Y :0.0208	TAC:0.634	TAT:0.366				
W:0.0179	TGG 1					
H :0.0179	CAC:0.696	CAT:0.304				
M:0.0185	ATG 1					
C :0.0229	TGC:0.673	TGT:0.327				
P :0.0746	CCA:0.219	CCC:0.372	CCG:0.196	CCT:0.213		

Table 7: Codon and amino acid (aa) frequencies (freq). The first column shows the one-letter code of the 20 amino acids and the relative frequency in the respective set. The sample sizes are $N = 510980$ (above) and $n = 6420$ (below). For each amino acid the codons coding for it are listed together with the relative frequencies of the codons coding for the amino acid. The upper table shows the data of all codons in the human training set and the lower table shows the data for the codons which begin in the range of the initial content model.

Bibliography

[AG00] J.F. Abril and R. Guigó. gff2ps: visualizing genomic annotations. *Bioinformatics*, 16(8):743–744, 2000.

[AGM+90] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[BA97] A.W. Bowman and A. Azzalini. *Applied Smoothing Techniques for Data Analysis*. Oxford Science Publications, 1997.

[BB99] John Besemer and Mark Borodovsky. Heuristic approach to deriving models for gene finding. *Nucleic Acids Research*, 27(19):3911–3920, 1999.

[BD00] Ewan Birney and Richard Durbin. Using GeneWise in the Drosophila Annotation Experiment. *Genome Research*, 10:547–548, 2000.

[BH00] Vineet Bafna and Daniel H. Huson. The Conserved Exon Method for Gene Finding. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 8:3–12, 2000.

[BK97] Chris Burge and Samuel Karlin. Prediction of Complete Gene Structures in Human Genomic DNA. *Journal of Computational Biology*, 268:78–94, 1997.

[BK98] Chris Burge and Samuel Karlin. Finding the genes in genomic DNA. *Current Opinion in Structural Biology*, 8:346–354, 1998.

[BM93] M. Borodovsky and J. McIninch. Genmark: parallel gene recognition for both DNA strands. *Comp.Chem.*, 17:123–133, 1993.

[Bro02] T.A. Brown. *Genomes 2*. John Wiley & Sons Inc., 2002.

[BSS00] M. Burset, I.A. Seledtsov, and V.V. Solovyev. Analysis of canonical and non-canonical splice sites in mammalian genomes. *Nucleic Acids Research*, 28:4364–4375, 2000.

[Bur97] C.B. Burge. *Identification of Genes in Human Genomic DNA*. PhD thesis, Stanford University, 1997.

[BV02] Brona Brejova and Tomas Vinar. A better method for length distribution modeling in HMMs and its application to gene finding. In A. Apostolico and M. Takeda, editors, *Combinatorial Pattern Matching, 13th Annual Symposium (CPM)*, volume 2373 of *Lecture Notes in Computer Science*, pages 190–202, Fukuoka, Japan, July 3-5 2002. Springer.

[BZ02] V. Brendel and Wei Zhu. Computational modeling of gene structure in Arabidopsis thaliana. *Plant Molecular Biology*, 48:49–58, 2002.

[Cla97] Jean-Michel Claverie. Computational methods for the indentification of genes in vertebrate genomic sequences. *Human Molecular Genetics*, 6(10):1735–1744, 1997.

[CP03] Simon L. Cawley and Lior Pachter. HMM sampling and applications to gene finding and alternative splicing. *Bioinformatics*, 19 Suppl. 2:ii36–ii41, 2003.

[CSP97] Candace J. Coolidge, J. Seely, Raymond, and James G. Patton. Functional analysis of the polypyrimidine tract in pre-mRNA splicing. *Nucleic Acids Research*, 25(4):888–896, 1997.

[Fic82] James W. Fickett. Recognition of protein coding regions in DNA sequences. *Nucleid Acids Research*, 10:5303–5318, 1982.

[FT92] J.W. Fickett and C.S. Tung. Assessment of protein coding measures. *Nucleic Acids Research*, 20:6441–6450, 1992.

[FYSB02] William G. Fairbrother, Ru-Fang Yeh, Philip A. Sharp, and Chris Burge. Predictive Identification of Exonic Splicing Enhancers in Human Genes. *Science*, 297:1007–1012, 2002.

[GAA+00] R. Guigó, P. Agarwal, J. Abril, M. Burset, and J.W. Fickett. An Assessment of Gene Prediction Accuracy in Large DNA Sequences. *Genome Res.*, 10:1631–1642, 2000.

[Hat02] Artemis G. Hatzigeorgiou. Translation initiation start prediction in human cDNAs with high accuracy. *Bioinformatics*, 18(2):343–350, 2002.

[Hau] David Haussler. Computational Genefinding. http://www.soe.ucsc.edu/~haussler/pubs.html.

[HFF+03] L.W. Hillier, R.S. Fulton, L.A. Fulton, T.A. Graves, K.H. Pepin, C. Wagner-McPherson, D. Layman, J. Maas, S. Jaeger, R. Walker, and et al. The DNA sequence of human chromosome 7. *Nature*, 424:157–164, 2003.

[HSF97] J. Henderson, S. Salzberg, and K.H. Fasman. Finding genes in DNA with a Hidden Markov Model. *Journal of Computational Biology*, 4(2):127–141, 1997.

[KA90] Samuel Karlin and Stephen F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl. Acad. Sci. USA*, 87:2264–2268, 1990.

[KFDB01] Ian Korf, Paul Flicek, Daniel Duan, and Michael R. Brent. Integrating Genomic Homology into Gene Structure Prediction. *Bioinformatics*, 1 Suppl. 1:S1–S9, 2001.

[KHRE96] D. Kulp, D. Haussler, M.G. Reese, and F.H. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 4:134–142, 1996.

[KHRE97] D. Kulp, D. Haussler, M.G. Reese, and F.H. Eeckman. Integrating database homology in a probabilistic gene structure model. *Pac. Symp. Biocomput.*, 2:232–244, 1997.

[KMH94] A. Krogh, I.S. Mian, and D. Haussler. A hidden Markov model that finds genes in E. coli DNA. *Nucleic Acids Research*, 22:4768–4778, 1994.

[Kni95] Rolf Knippers. *Molekulare Genetik*. Georg Thieme Verlag, 1995.

[Kre91] Ulrich Krengel. *Einführung in die Wahrscheinlichkeitstheorie und Statistik*. Vieweg, 1991.

[KRGS01] Zhengyan Kan, Eric C. Rouchka, Warren R. Gish, and David J. States. Gene Structure Prediction and Alternative Splicing Analysis Using Genomically Aligned Ests. *Genome Research*, 11:889–900, 2001.

[Kro97] Anders Krogh. Two methods for improving performance of an HMM and their application for gene finding. *Proc. Fifth Int. Conf. Intelligent Systems for Molecular Biology*, pages 179–186, 1997.

[Kro00] Anders Krogh. Using Database Matches with HMMGene for Automated Gene Detection in Drosophila. *Genome Research*, (10):523–528, 2000.

[Kul03] David C. Kulp. *Protein-Coding Gene Structure Prediction using Generalized Hidden Markov Models*. PhD thesis, University of California, Santa Cruz, 2003.

[LB01] Lee P. Lim and Chris Burge. A computational analysis of sequence features involved in recognition of short introns. *Biochemistry*, 98(20):11193–11198, 2001.

[MD02] Irmtraud M. Meyer and Richard Durbin. Comparative ab initio prediction of gene structures using pair HMMs. *Bioinformatics*, 18(10):1309–1318, 2002.

[Mor99] B. Morgenstern. DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15:211–218, 1999.

[MRA+02] B. Morgenstern, O. Rinner, S. Abdeddaim, D. Haase, K. Mayer, A. Dress, and H.-W. Mewes. Exon Discovery by Genomic Sequence Alignment. *Bioinformatics*, 18:777–787, 2002.

[MSSR02] Chatherine Mathé, Marie-France Sagot, Thomas Schiex, and Pierre Rouzé. Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Research*, 30(19):4103–4117, 2002.

[MW02] Rainer Merkl and Stephan Waack. *Bioinformatik Interaktiv*. Wiley-VCH, 2002.

[PAA+03] G. Parra, P. Agarwal, J. Abril, T. Wiehe, J.W. Fickett, and R. Guigó. Comparative Gene Prediction in Human and Mouse. *Genome Research*, 13:108–117, 2003.

[PEG00] Genís Parra, Blanco Enrique, and Roderic Guigó. GeneID in Drosophila. *Genome Research*, 10:511–515, 2000.

[PLS01] Mihaela Pertea, Xiaoying Lin, and Steven L. Salzberg. Genesplicer: a new computational method for splice site prediction. *Nucleic Acids Research*, 29(5):1185–1190, 2001.

[Ree00] Martin G. Reese. *Computational prediction of gene structure and regulation in the genome of Drosophila melanogaster*. PhD thesis, Universität Hohenheim, 2000.

[RGH+00] M.G. Reese, Hartzell G., N.L. Harris, U. Ohler, J.F. Abril, and Lewis S.E. Genome Annotation Assessment in Drosophila melanogaster. *Genome Research*, 10(4):391–393, 2000.

[RKTH00] Martin G. Reese, David Kulp, Hari Tammana, and David Haussler. Gene Finding in Drosophila melanogaster. *Genome Research*, 10:529–538, 2000.

[RM02] Oliver Rinner and Burkhard Morgenstern. AGenDA: Gene prediction by comparative sequence analysis. *In Silico Biology*, 2, 2002.

[RMO01] Sanja Rogic, Alan K. Mackworth, and Francis B.F. Ouellette. Evaluation of Gene-finding Programs on Mammalian Sequences. *Genome Research*, 11:817–832, 2001.

[Ros96] Sheldon M. Ross. *Stochastic Processes*. John Wiley & Sons, 1996.

[Sch03] Oliver Schöffmann. Gewinnung extrinsischer Informationen zur Genvorhersage und Einbindung in ein Hidden Markov Modell. Master's thesis, Universität Göttingen, 2003.

[SDSO98] Steven L. Salzberk, Arthur L. Delcher, Kasif Simon, and White Owen. Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, 26(2):544–548, 1998.

[SGH+98] W. Seyffert, H.G. Gassen, O. Hess, H. Jäckle, and K.-F. Fischbach. *Lehrbuch der Genetik*. Gustav Fischer Verlag, 1998.

[SM82] R. Staden and A.D. McLachlan. Codon preference and its use in identifying protein coding regions in long DNA sequences. *Nucleid Acids Research*, 10:141–156, 1982.

[SS95] E. Snyder and G. Stormo. Identification of protein coding regions in genomic DNA. *Journal of Molecular Biology*, 248:1–18, 1995.

[Str91] B. Stroustrup. *The C++ Programming Language*. Addison-Wesley Series in Computer Science, 1991.

[SW03] Mario Stanke and Stephan Waack. Gene prediction with a hidden markov model and new intron submodel. *Bioinformatics*, 19 Suppl. 2:ii215–ii225, 2003.

[TDZ01] Jack E. Tabaska, Ramana V. Davuluri, and Michael Q. Zhang. Identifying the 3'-terminal exon in human DNA. *Bioinformatics*, 17(7):602–607, 2001.

[TRG+03] L. Taher, O. Rinner, S. Gargh, A. Sczyrba, M. Brudno, S. Batzoglou, and M. Morgenstern. Homology-based gene prediction. *Bioinformatics*, 19:1575–1577, 2003.

[UZB00] Jonathan Usuka, Wei Zhu, and Volker Brendel. Optimal spliced alignment of homologous cDNA to a genomic DNA template. *Bioinformatics*, 16(3):203–211, 2000.

[Vit67] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Informat. Theor.*, IT-13:260–269, 1967.

[WPY01] G.K.-S. Wong, D.A. Passey, and J. Yu. Most of the Human Genome is Transcribed. *Genome Research*, 11:1975–1977, 2001.

[YLB01] Ru-Fang Yeh, Lee P. Lim, and Chris Burge. Computational Inference of Homologous Gene Structures in the Human Genome. *Genome Research*, 11:803–816, 2001.

[Zha98] M.Q. Zhang. Statistical features of human exons and their flanking regions. *Human Molecular Genetics*, 7(5):919–932, 1998.

Lebenslauf

Name	Mario Stanke
Geburtsdatum	18. Juni 1974
Geburtsort	Witzenhausen
Staatsangehörigkeit	deutsch

Schulbildung

1980 bis 1984	Grundschule in Bad Sooden-Allendorf
1984 bis 1986	Förderstufe in Bad Sooden-Allendorf
1986 bis 1993	Rhenanus-Schule Bad Sooden-Allendorf
Juni 1993	allgemeine Hochschulreife

Wehrdienst

Juli 1993 bis Juni 1994	Grundwehrdienst in Bad Frankenhausen und Sontra

Studium

Oktober 1994 bis Juli 1996 und	Studium der Mathematik im Diplomstudiengang an der
Oktober 1997 bis Februar 2000	Georg-August-Universität Göttingen (Nebenfach Informatik)
August 1996 bis Mai 1997	Studium der Mathematik und Informatik an der University of California in Berkeley
Februar 2000	Diplomprüfung in Mathematik
seit Oktober 2000	Promotionsstudium im Fach Informatik an der Georg-August-Universität Göttingen

Berufstätigkeit

Juli 1997 bis September 1997	studentischer Mitarbeiter am Lawrence Berkeley Lab, Berkeley, USA
WS 97/98, WS 98/99, SS 99, WS 99/00	studentische Hilfskraft am mathematischen und stochastischen Institut der Universität Göttingen
Juli 1998 bis September 1998	Praktikum bei der Victoria Versicherung in Düsseldorf
März 2000 bis August 2000	Anwendungsentwicker bei Framfab in Köln
Oktober 2000 bis März 2002	wissenschaftlicher Mitarbeiter am stochastischen Institut der Universität Göttingen
Januar 2002 bis Dezember 2002	wissenschaftliche Hilfskraft am Institut für Numerische und Angewandte Mathematik der Universität Göttingen
Januar 2003 bis Februar 2003	wissenschaftliche Hilfskraft am Institut für Mikrobiologie und Genetik der Universität Göttingen
seit März 2003	wissenschaftlicher Mitarbeiter der Abteilung Bioinformatik des Instituts für Mikrobiologie und Genetik der Universität Göttingen