Accurate Prediction of Protein-Coding Genes with Discriminative Learning Techniques

vorgelegt von
Diplom-Physikerin
Gabriele Beate Schweikert
aus Heidenheim a.d. Brenz

Von der Fakultät IV — Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades eines
Doktor der Naturwissenschaften
Dr. rer. nat.

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. Manfred Opper
Berichter: Dr. Gunnar Rätsch
Berichter: Prof. Dr. Klaus-Robert Müller
Berichter: Prof. Dr. Bernhard Schölkopf

Tag der mündlichen Aussprache: 24.11.2010

Berlin 2010
D 83

Erklärung

Hiermit erkläre ich, dass ich diese Schrift selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die im Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben der Quellen kenntlich gemacht sind.

Göttingen, 2010 *Gabriele Schweikert*

Für Antonia

Abstract

Nowadays, the sequences of complete genomes for more and more organisms arrive at great pace. The aim of this thesis was therefore, to develop a novel, efficient and accurate method for the detection of protein-coding genes by computational means. The focus was put on eukaryotic genomes, where the open reading frames of these genes are typically interrupted by non-coding introns. In contrast to most existing gene finders, I created a purely *discriminative* system, thus achieving significant improvements in the accuracy of the predictions. In particular, the segmentation problem was solved with hidden semi-Markov support vector machines (HSM-SVMs), which have been shown to perform well on label sequence prediction tasks. However, applying this technique to data sets of the size and complexity of genomic-scale sequences posed a substantial challenge. It was solved by a two-step architecture: initially, the problem is partitioned into several independent sub-problems, namely the detection of signals on the genomic DNA induced by functional elements such as promoters or splice sites. For these tasks, support vector machines (SVMs) were used that are each capable of exploiting high-order information from millions of training examples. Subsequently, the integration and weighting of the individual components is efficiently learnt in the HSM-SVM framework. It is thus possible to process thousands of sequences in a reasonable time span, while taking full advantage of the wealth of encoded information. To aid the highly involved process of annotation, a web server was developed that allows to perform the complex process of gene prediction on the push of a button. This service includes optional re-training for species-specific models.

With the gene finding system, mGene, we participated in an international gene prediction competition on the genome of the model organism *C. elegans*. An independent evaluation revealed mGene's high prediction quality when compared to 47 submitted sets from 17 different groups: Our contributions were most accurate in seven out of twelve evaluation criteria, ranking second in three more measures. With further improvements to mGene, implemented after the competition, I created predictions that were most accurate according to 10 out of 12 evaluation criteria.

I subsequently generated genome-wide predictions for *C. elegans* and predicted $\approx 2,000$ genes that were not contained in the current annotation. The expression of some of the potentially novel genes were tested by experiment, which showed a remarkably high success rate. These findings suggest that even the gene catalog of such a well-studied organism can be improved by mGene predictions.

Finally, I predicted complete gene sets for four other nematodes, which previously lacked high quality annotations. In this context, I also investigated new techniques to adapt a learnt model for the prediction on related organisms. Initial results show that prediction accuracy can thus be improved significantly on various genomic signal prediction tasks.

Zusammenfassung

Zur Zeit werden die Genome einer Vielzahl von Organismen vollständig sequenziert. Die vorliegende Arbeit hatte daher zum Ziel, eine neue, gleichermaßen effiziente wie genaue Methode zu entwickeln, die es erlaubt, Protein-kodierende Gene mit Hilfe eines Computer-Programms zu finden. Betrachtet wurden eukaryotische Genome, bei denen die *Offenen Leserahmen* der Gene durch nicht-kodierende Introns unterbrochen werden. Im Gegensatz zu den meisten bereits bestehenden Ansätzen wurden ausschliesslich *diskriminative* Lerntechniken verwendet, wodurch eine signifikante Steigerung der Vorhersagegenauigkeit erzielt werden konnte. Um die *Segmentierung* der DNA in kodierende und nicht-kodierende Abschnitte korrekt vorherzusagen, wurden *hidden semi-Markov support vector machines (HSM-SVMs)* trainiert. Die Anwendung dieser Technik auf Datensätze, die den Umfang und die Komplexität von genomweiten DNA-Sequenzen aufweisen, stellte allerdings eine erhebliche Herausforderung dar. Daher wurde ein zweistufiges Verfahren gewählt: Zunächst wurden mehrere unabhängige Teilprobleme gelöst — wie etwa die Erkennung von verschiedenen funktionalen Elementen und *Signalen* auf der DNA-Sequenz, z.B. von Promotoren oder Spleiß-Stellen. Zu diesem Zweck wurden *support vector machines* (SVMs) eingesetzt, die in der Lage sind, Informationen hoher Ordnung aus Millionen von Trainingsbeispielen zu nutzen. Anschließend wird die geeignete Gewichtung der einzelnen Komponenten mit Hilfe des HSM-SVM-Systems erlernt, so dass korrekte *Genstrukturen* bestimmt werden können. Dadurch wird es möglich, tausende Beispiele in einer angemessenen Zeitspanne auszuwerten und dabei die Vielzahl verschlüsselter Informationen weitestgehend auszuschöpfen. Um den komplizierten Prozess der Genvorhersage quasi auf Knopfdruck durchführen zu können, wurde außerdem ein Webserver entwickelt. Dieser Dienst enthält die Option, Spezies-spezifische Modelle neu zu erzeugen.

Mit dem resultierenden System mGene nahmen wir an einem internationalen Wettbewerb zur Genvorhersage teil. Eine unabhängige Evaluierung bewies die hohe Vorhersagequalität von mGene im Vergleich zu 47 eingereichten Datensätzen von 17 verschiedenen Gruppen: Unsere Beiträge erwiesen sich als die genauesten in sieben von insgesamt zwölf Evaluationskriterien, in weiteren drei Kriterien kamen sie auf den zweiten Platz. Nach dem Wettbewerb wurde mGene weiter verbessert, so dass Vorhersagen ermöglicht wurden, die sich als die genauesten in zehn von zwölf Evaluationskriterien etablierten.

Infolgedessen wurden genomweite Vorhersagen für *C. elegans* erzeugt, wobei ca. 2000 Gene vorhergesagt wurden, die nicht in der aktuellen Annotation vorkamen. Bei der experimentellen Überprüfung dieser potentiell neuen Gene wurde eine sehr hohe Erfolgsrate erzielt. Diese Ergebnisse zeigen, dass sogar der Gen-Katalog von so intensiv untersuchten Organismen wie von *C. elegans* durch mGene-Vorhersagen verbessert werden kann.

Schließlich, wurden Genvorhersagen für vier weitere Nematoden generiert, für die zuvor keine ausreichenden Annotationen existierten. In diesem Zusammenhang, wurden auch Techniken untersucht, um ein auf einem bestimmten Organismus erlerntes Modell auf andere Genome zu übertragen. Erste Ergebnisse zeigen, dass die Genauigkeit für die Vorhersagen genomischer Signale auf diese Weise signifikant verbessert werden kann.

Preface

Initially trained as a physicist, I became interested in the biological sciences early in my undergraduate studies. I therefore started a *Diplom* thesis in the group of Prof. Baumeister, where cryo-electron tomography is used to visualize cells in 3D.

In the context of image processing, I became aware of the power of modern machine learning techniques to extract patterns from diverse, complex data. To learn more about these methods and to apply them to biologically relevant questions, I joined the groups of Bernhard Schölkopf, Detlef Weigel and Gunnar Rätsch. I started to work on the detection of sequence variation, analyzing a huge set of hybridization data from re-sequencing tiling arrays. We revealed hundreds of thousands of single nucleotide polymorphisms (SNPs) in the genome of 20 varieties of the plant *Arabidopsis thaliana*, including many "major effect" SNPs that introduced premature stop codons, or destroyed the consensus dimer of splice sites [32]. Due to space constraints, these results are not included into this thesis. Subsequently, I turned my attention to a problem that is substantially more difficult to solve, namely to identify those SNPs *in the vicinity* of splice sites that had a similarly large effect on the gene product by changing the splice form. While the initial results of this effort were not satisfying — mainly due to the lack of high quality labelled data for training and assessment — I became interested in splice site prediction, and together with Sören Sonnenburg and Petra Philips I studied the problem in various organisms [148]. I also made myself familiar with the system mSplicer that predicts splice forms when the location of a gene is known.

At the end of 2006, the international wormbase consortium launched a genome annotation competition (nGASP) to assess the quality of existing gene finders. We took this opportunity to advance our methods, to apply them to the provided data and accordingly generated gene predictions for the nematode *Caenorhabditis elegans*. As this process was highly time critical, several people were involved in the effort. The main contributors included, next to myself, Gunnar Rätsch, Alexander Zien and Georg Zeller. While we submitted several data sets that performed well in the evaluation [34], we did not have a fully automatic gene finding system in place at this stage. Many of the essential steps were performed manually and the tedious generation of a single complete set of predictions put a strain on several people over several weeks. Later, I assembled the complete gene finder mGene, including a web server [139], together with Jonas Behr, whose student research project and *Diplom* thesis I was co-supervising. I also produced genome-wide annotations for *C. elegans* and four related nematodes, which were provided to the wormbase consortium for inclusion into the official annotation. Furthermore, I planned and analyzed experiments to validate our predictions. These tests were carried out by the technical assistants Lisa Hartmann, Anja Bohlen and Nina Krüger. For a comparative analysis of the five nematode gene catalogues, I collaborated with Christoph Dieterich [141]. At this point, I became interested in methods that can be used to adapt a model learnt on one organism to predict on another one. I therefore co-supervised the *Diplom* thesis of Christian Widmer and studied domain adaptation methods for computational gene finding [140].

Acknowledgments

Foremost, I would like to thank my thesis committee, including Gunnar Rätsch, Bernhard Schölkopf, Detlef Weigel and Klaus-Robert Müller, for making this thesis possible. I very much appreciated their interest and support of my work, the insightful discussions as well as the freedom to explore.

I express my sincere gratitude to my advisers Gunnar Rätsch, Bernhard Schölkopf and Detlef Weigel. Their combined expertise and guidance in the very different fields of machine learning and biology enabled and propeled this work and it was a true privilege to work in the excellent scientific environment at the Max Planck Campus, Tübingen. They also allowed me to participate at various conferences, and thus to learn at the fore front of research, broadening the scientific and personal horizon in many different ways.

Moreover, I would like to thank Gunnar Rätsch, for his day-to-day supervision, for providing direction as well as inexhaustible support in all questions and situations. The inspiring atmosphere created in his group added considerably to my graduate experience and led to a number of precious and lasting friendships.

My special thanks go to Cheng Soon Ong, whose insightful and patient teaching substantially eased the start in the team. His encouragement gave me strength and confidence. Likewise, I thank Alexander Zien for his mentoring, for sharing his vast knowledge and insight, allowing me to gain a deeper understanding of the field. I am very grateful to Petra Philips for countless discussions, and for putting things in the right perspective during our little walks. Thanks go to Sören Sonnenburg for his unrestricted assistance, in particular with respect to technical problems concerning Shogun. In addition, his frequent visits to Tübingen were always welcome and pleasant changes. I also gladly acknowledge my fellow PhD students Georg Zeller, Fabio De Bona and Elisabeth Georgii. It was fun working with them. I thank Chris Widmer and Jonas Behr. I enjoyed co-supervising their Diplom thesis. It taught me a lot. Additionally, I would like to thank all people in Gunnar's, Bernhard's and Detlef's lab for inspiring discussions at many occasions ranging from the retreats, countless bio-breakfasts and cake talks.

Many of the projects in Gunnar's group are large, highly interwoven, and encourage to team up for the best result. Therefore, computational tools are reused frequently and I extend my thanks to colleagues who provided data or source code, including Gunnar Rätsch, Jonas Behr, Sören Sonnenburg, Alexander Zien, Cheng Soon Ong, Georg Zeller, Petra Philips, and Fabio De Bona. Furthermore, I thank the system administrators, Andre Noll and Sebastian Stark, for maintaining an excellent computing facility. Thanks go to the technical assistants Lisa Hartmann, Anja Bohlen and Nina Krüger for their quick and uncomplicated aid. I also thank Richard Clark and Christoph Dieterich for collaborating and working with me in a very dedicated way. I thank Alexander Zien, Jonas Behr, Regina Bohnert, Sören Sonnenburg and Ulrich Zachariae for critically reading the manuscript.

Finally, I express my deepest gratitude to my parents, who have created a firm foundation for me to stand on; to my soul mate Uli and to Antonia, who are my joy and happiness.

Contents

1. Introduction

> "The double helix is indeed a remarkable molecule. Modern man is perhaps 50,000 years old, civilization has existed for scarcely 10,000 years [...]; but DNA and RNA have been around for at least several billion years. All that time the double helix has been there, and active, and yet we are the first creatures on Earth to become aware of its existence."
> Francis Crick [42]

Subject of the thesis at hand is the reading and understanding of the material that is thus addressed by Francis Crick. The remarkable substance was first discovered in the castle of Tübingen, where, in 1869, Friedrich Miescher studied the chemical components of cells. He could show that the isolated molecule was distinctly different from all known proteins or lipids [104], and, as it was derived from the cells' nuclei, he called it nuclein, which is still reflected in its present name: deoxyribonucleic acid, DNA. Careful analysis allowed him to correctly characterize some important properties of DNA and with astonishing foresight he speculated about the function of the substance, that it might have a role in the transmission of hereditary traits. This discovery, together with the laws of inheritance, formulated three years earlier by Gregor Mendel, therefore lay the foundation of molecular and classical genetics [43]. In the following eighty years the functional role of the DNA was confirmed [8, 68] and its structure resolved [57, 181]. Eventually, the mechanism of heredity was elucidated and the genetic code deciphered [40].

Research in the field was further catalyzed when the first sequencing technologies were established in the 1970s, mainly by Frederick Sanger [130, 132]. With the shotgun-sequencing method [7, 150], the first complete genome, that of the bacteriophage ΦX174, was sequenced [131]. It took another twenty years before the genome sequence of a free-living multi-cellular organism, that of the roundworm *Caenorhabditis elegans* could be fully determined [166]. In the year 2000, vibrant research on DNA reached another prominent milestone when the completion of the initial survey of "the entire human genome" was announced [167]. This achievement was assumed to have a tremendous impact on our lives, as a revolution in the diagnosis, prevention and treatment of many human diseases was anticipated [45]. To date, many promised benefits have remained elusive and in fact it is now realized that we are only at the beginning to understand the complex dynamics of genomes, such

that the notion starts to prevail, that there is not *one* genome of an organism. However, the importance of the public human genome project (HGP) is unquestioned and attested every day by millions of search queries on the main genome browsers (UCSC [123], EBI Ensembl [55]) as well as the NCBI genome database [183]. It has also fuelled many follow up studies: To mention but a few, the *ENCyclopaedia Of DNA Elements (ENCODE)* was launched in 2003 to comprehensively identify the functional elements in the genome [52]. With the *HapMap* project, a detailed analysis of genome variation was reported in 2005, allowing the identification of many genetic factors for common diseases [79]. The *Cancer Genome Atlas (TCGA)* started to catalog the genetic and epigenetic abnormalities in thousands of different tumor specimens in the same year [28]. Going beyond medical research, the HGP has also greatly aided in comparative genomics to elucidate evolutionary forces that have shaped genome sequences during millions of years of evolution.

In the light of all of these large efforts and achievements, it is a puzzling fact that within the last ten years it was not possible to precisely answer a simple and obvious question [112]: How many genes are contained in the human genome? The inability to resolve this issue and ultimately to detect all human genes reflects how far away we still are from "speaking the language of the genome fluently," as the HGP director Francis Collin put it [45]. The approximately 22,000 protein coding genes — with an uncertainty of around 2,000 genes [112] — are well hidden among the total of three billion DNA letters: they only account for 1.5% of the sequence of the human genome. Moreover, the difficulty in annotating DNA sequences is not only a fact for the human genome but also for all other genomes of higher eukaryotes.

Since the 1990s, experimental techniques to detect the location and structure of genes have been accompanied by extensive computational strategies. While these methods greatly aided in the generation of initial annotations, high quality was typically only achieved through labor-intensive manual inspection by experts performed at large sequencing centers. Recently, however, computational gene finding has experienced a major breakthrough by adopting discriminative learning techniques [23], which have the potential to be more accurate than classical generative techniques. With this work, a gene prediction tool, called mGene [141] was developed that goes beyond other existing gene finders as it combines state-of-the-art machine learning techniques, hidden semi-Markov support vector machines (HSM-SVMs) [119, 172] with a very elaborated underlying biological model that also includes untranslated regions. To predict genes in a given DNA sequence, mGene conceptually performs two subsequent steps: First, it detects specific *signals* on the DNA that mark boundaries in the sequence and, simultaneously, it examines the sequence *content* of whole segments. The corresponding sub-models are designed carefully, using support vector machines (SVMs) with very efficient and specialized string kernels. It is thus possible to take full advantage of high-order information and long-range interac-

tions from millions of training examples, both of which could only be exploited to a very limited degree in earlier systems. In a second step, the individual components are integrated to deduce complete and biological meaningful *gene structures*. Here, HSM-SVMs are employed for the first time in a computational gene finder. This discriminative structure prediction method has been shown to be highly accurate on various label sequence learning tasks, however, applying it to data sets of the size and complexity of genome sequences posed a severe challenge. By solving this problem, it was indeed possible to outperform state-of-the-art competitors in an objective comparison.

An additional line of research during my doctoral studies, was the question how data derived from one (*source*) organism can be used to improve predictions on other, more or less closely related (*target*) organisms for which little training data exist. To address this problem, domain adaptation methods, which are being developed in natural language processing, have been investigated systematically and applied for the first to the genomic signal detection problem. It could be shown that the right combination of target and source models can lead to significant improvements relative to predictions with either one alone, in particular if the *target* data are sparse and if the organisms are not too similar. This finding can help to direct further research to employ known models for new annotations thus avoiding experimental efforts to generate large training sets for each newly sequenced genome.

Organization of this Thesis and Own Contributions

I will start in the following Chapter 2 with an introduction of the biology of genes. It also contains a mathematical view of the problem of gene finding as well as a thorough analysis of earlier, *generative*, approaches. The observed shortcomings will serve as a motivation for some design choices in mGene. Therefore, Chapter 3 contains a description of the employed *discriminative* machine learning methods, including SVMs for binary classification and, most importantly, label sequence learning techniques such as HM-SVMs and their extension to HSM-SVMs. I will discuss the relation of HSM-SVMs to other max-margin methods on the one hand, and generative hidden Markov models on the other hand. Also, the two-stage learning strategy that was employed in mGene is justified from a theoretical point of view. In Chapter 4, the binary classification techniques are adopted to design signal detectors for the recognition of genomic features. Some of these, like the splice site detector, are based on earlier work by Sören Sonnenburg, Alexander Zien and Gunnar Rätsch. For other signals, like the trans-splicing signal, I designed new sensors. In addition, I automated the complete process from data generation to prediction and unified the divers tasks from splice site recognition to promoter detection in a common framework, which is, however, flexible enough to accommodate signal specific differences.

Additionally, by applying a five-fold cross validation scheme in combination with a post-processing step to properly rescale the predictions from the different classifiers, I was able to construct a system that generates unbiased predictions for whole genomes. For shorter run-times, parallel computing was implemented. Next, the problem of gene structure prediction is solved with the HSM-SVM approach as described in Chapter 5. The resulting algorithm is conceptually similar to the mSplicer system, introduced earlier by Sören Sonnenburg and Gunnar Rätsch. Therefore, the Viterbi-like decoding algorithm, implemented by Gunnar Rätsch, could be re-used. However, the underlying model in mGene is far more elaborated, containing many more states and features, also leading to a more complex scoring function. The training procedure that follows a so-called column generation scheme was therefore newly realized. Also, in this theses, a more complex loss term is introduced, which better corresponds to the similarity measure employed for the evaluation of prediction accuracy. Applications of the described methods are presented in Chapter 6: First, I generated genome-wide splice site predictions for several model organisms. Their accuracy is compared to predictions generated by Petra Philips and Jonas Behr using a generative base-line method. Second, I present results of the international nGASP challenge. Together with Alexander Zien, Georg Zeller and Gunnar Rätsch I produced initial predictions with mGene. I subsequently generated an improved prediction set and thoroughly evaluated all data sets including those of competitors. Third, I created a genome-wide gene catalogue for *C. elegans*, compared it to earlier annotations, and planned and analyzed wet-lab experiments to confirm novel genes. Lastly, I predicted comprehensive gene sets for four other nematodes. These served as basis for a comparative study of the different organisms, which was accomplished in collaboration with Christoph Dieterich. In Chapter 7, I introduce a web service that allows a wider use of mGene. In particular, users without programming experiences are now enabled to create gene annotations for sequenced genomes. It contains many different tools and was realized together with Jonas Behr and Gunnar Rätsch. Finally, I investigated domain adaptation techniques to transfer a model that was learnt on a source distribution to data derived from a different target distribution. These methods are discussed in Chapter 8, which also contains the results for a splice site detection task across different organisms. As the number of these experiments was very large, some of them were run by Christian Widmer. The thesis is summarized in Chapter 9, where I also discuss directions of future research.

2. Preliminaries on Computational Gene Finding

This chapter contains the ingredients needed to build a computational gene finder. It will start with some principal facts on the biology of genes and experimental techniques to detect the location of genes on the genome. I will also briefly discuss some evolutionary aspects that concern gene finding in different organisms. Additionally, the model organism *Caenorhabditis elegans* is introduced, as its genome is studied later on in this thesis. I will then look at the problem of gene finding from a mathematical point of view and introduce several evaluation criteria that are typically employed to compare computational approaches to gene prediction. Eventually, concepts of classical computational gene finders are elucidated. Their limitations are discussed and will serve as a motivation for the presented approach in mGene.

2.1. The Biology of Genes

In the following considerations, the subject is covered in so far as is relevant for computational gene prediction and is therefore by no means close to completeness with regard to today's knowledge. It starts with a simple, classical view on protein-coding genes. As non-coding genes are not treated in this thesis, they will simply be referred to as *genes* in the following. Although many of the discussed principles are universal to all known live forms, I will mainly focus on eukaryotes (i.e. higher organisms that possess cell nuclei), and some mechanisms may differ in prokaryotes. For a more conclusive presentation, see for instance [94].

Genes as Sub-sequences on the DNA

Historically, the term "gene" was introduced at the beginning of the 20th century to define a basic unit of heredity in living organisms. The molecular basis of genes was only established thirty years later: Genes are parts of the DNA, which is organized in chromosomes within the cell nucleus. This large molecule consists of two long polymer chains that are themselves composed of many subunits, called nucleotides (see Figure 2.1). Each subunit contains a phosphate group, a deoxyribose sugar ring, and

a nucleobase. The later is the reason why a nucleotide is simply referred to as 'base' in the context of genomics and it also defines the type of the subunit: There are four different nucleobases, namely adenine, thymine, cytosine, and guanine, abbreviated with `'A','C','G'` and `'T'`, respectively. Chemically, adenine and guanine are similar bicyclic structures called purines; cytosine and thymine are smaller structures with a single ring called pyrimidines. Within one strand, successive nucleotides are linked by phosphodiester bonds. Therefore, DNA has the potential to form arbitrary sequences using the four 'letters' `'A','C','G',` and `'T'`, thus storing complex genetic information over millions of years of evolution.

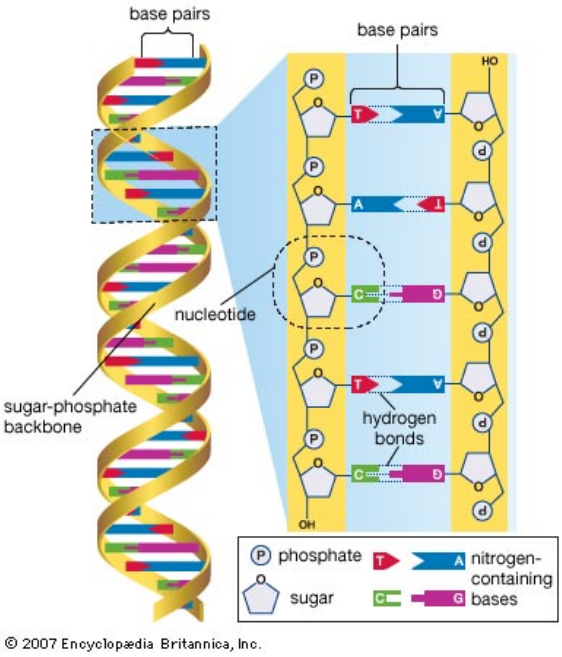


Figure 2.1.: Schematic drawing of a small piece of DNA. The illustration is taken from the Encyclopedia Britannica, Inc. [53]

The alternating phosphate and sugar residues form the backbone of the DNA strand. As a result the strand has an orientation: The end with a terminal phosphate group is called the 5' (five prime) end, and the one with a terminal hydroxyl group the 3' (three prime) end. Important cellular processes, such as replication or transcription, work in a unique direction, that is from the 5' to the 3' end. Nucleotides closer to the 5' end relative to a given position will therefore also be called 'upstream' and nucleotides closer to the 3' end, 'downstream' of this location. Also, whenever features on the DNA are described, it is assumed that the sequence under consideration is given in the 5' to 3' direction.

In cells, DNA usually exists in a double helix structure, where two strands twist around each other in a right-handed spiral. Hydrogen bonds are formed across the helix between ever one base attached to the two opposing strands. Notably, the

two strands are arranged in an anti-parallel fashion: The base at the 5' end of one strand binds to the base at the 3' end of the other. Due to geometrical constraints there are only two stable possibilities for base pairing: adenine forms a base pair with thymine (`A-T`) with two hydrogen bonds, and guanine forms one with cytosine (`G-C`) with three hydrogen bonds. Consequently, the double helix is more stable if its sequence has a high `GC` content. Due to the complementarity of the base pairing, the sequence of nucleotides on one strand (the forward strand) is sufficient to determine the sequence on the other strand (the reverse strand). The reverse complement is derived by replacing the bases with their complement and subsequently reversing the order. The length of a sequence, i.e. the number of nucleotides, will in the following be measured in units of base pairs (bp), kilo base pairs (kbp) or mega base pairs (Mbp).

The entire DNA of an organism is organized in one or more chromosomes. Its entire sequence is called its genome. Genes can reside on the forward, as well as on the reverse strand. The specific location of a gene on the chromosome is called locus. By convention, coordinates of a feature on the sequence are zero-based and given with respect to the forward strand. A complete localization of a feature requires the indication of the chromosome, the strand and the position.

The Dogma of the Triad

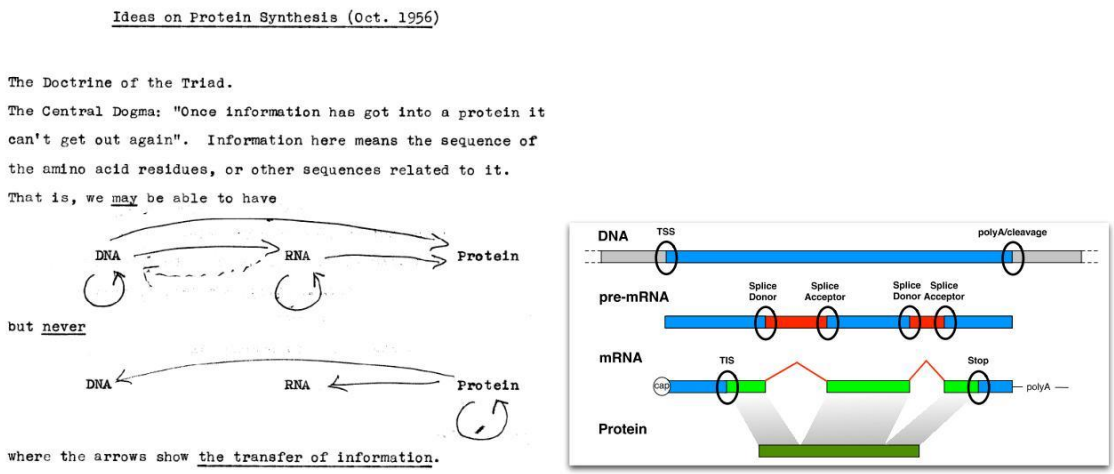


Figure 2.2.: The central dogma of molecular biology. Left: Original Drawing by Francis Crick [40]. Right: Schematic drawing of the three steps: DNA transcription, RNA processing, and translation.

According to the central dogma of molecular biology (Figure 2.2), defined portions of the DNA sequence, the genes, are transcribed nucleotide by nucleotide into pre-messenger RNAs (pre-mRNAs), which are subsequently processed to mature mRNA

and translated corresponding to the genetic code into polypeptides, i.e. chains of amino acids. These fold into functional proteins, which perform most of the vital processes of cells. Each of the steps involved in the expression of protein-coding genes requires the binding of specialized proteins, also called *trans-acting* factors, to specific motifs on the DNA or the corresponding RNA sequence, which are termed *cis*-motifs (see for example Figure 2.3). In some cases, these motifs are highly variable between individual genes of the same genome - they are degenerate - and also their relative localization in the gene can differ drastically. In the following paragraphs, transcription, modification and translation and their respective *imprints* on the DNA will be described in more detail as they provide important clues for computational gene prediction.

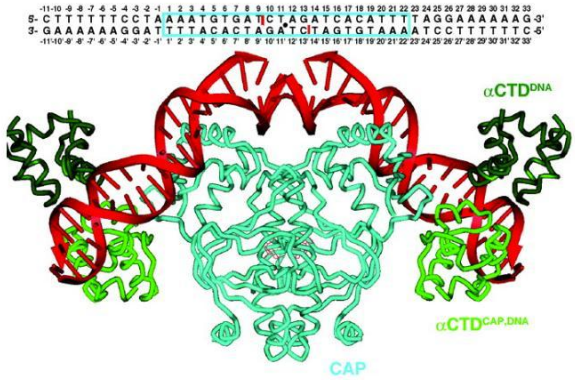


Figure 2.3.: Example of DNA-protein complex. The catabolite activator protein (CAP), depicted in cyan binds to a short DNA fragment, drawn in red to activate transcription. The corresponding 44 bp DNA sequence is given on top. This image is taken from [15]

Transcription

During transcription, a particular DNA sub-sequence on one strand is read in the 5' to 3' direction by the enzyme RNA polymerase II (RNA pol II) and a complementary, anti-parallel pre-mRNA strand is synthesized. In the resulting RNA the base thymine, `T`, is replaced by uracil, `U`. For notational simplicity, however, all sequences will be given as the corresponding DNA sequences in this thesis, substituting all `U`s with `T`s.

In eukaryotes, transcription is a remarkable complex process (for recent reviews see [72, 80, 170]). It involves the interplay of the core RNA pol II transcriptional machinery with a network of thousands of sequence-specific DNA-binding factors, as well as co-regulators that connect the DNA-binding factors to the transcriptional machinery. Additionally, a number of chromatin remodeling factors are necessary

to mobilize nucleosomes, and a variety of enzymes are required to catalyze covalent modification of histones and other proteins [2].

The initiation alone requires the assembly of RNA pol II and a multitude of *general transcription factors* to form a pre-initiation complex. They bind to short double-helical DNA sequences in the *core promoter region*, which is located in an approximately 60 bp DNA sequence that overlaps the transcription start site (tss), see Figure 2.4. Core promoter elements, such as the TATA-box — thus called, due to its consensus sequence `TATAAA` —, or the initiator element (Inr), serve as recognition sites and they become nonfunctional if moved even short distances from the tss. Some, all, or none of these elements may be included in a particular core promoter [2], and the presence of several of them can lead to synergetic effects. In addition, there are typically several clusters of *proximal promoter elements* within 70-200 bp upstream of the tss. They increase the frequency of transcription initiation and are potentially responsible for the recruiting of *long-range regulatory elements* including enhancers and silencers that mediate the complex patterns of gene expression in different cell types, depending on different developmental stages and environmental factors. The respective binding sites can be scattered over an average distance of 10 kb upstream of the TSS. In different genes, the set of represented transcription factor binding sites (TFBS), the number of a distinct element, their order, orientation and position can vary widely, thus expanding the information content stored on the genetic level. It also makes the computational detection of gene beginnings a particularly difficult problem. The termination of the transcription process is also not well understood, however, it requires the cleavage and polyadenylation of the 3' end of the transcript (see below).

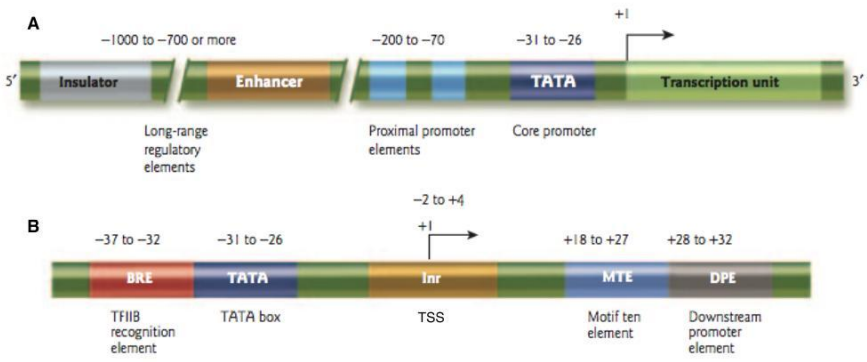


Figure 2.4.: **A** Typical eukaryotic transcription unit. **B** RNA pol II core promoter motifs. Illustrations taken from [2].

Post-transcriptional Modifications

The nascent transcript is called pre-mRNA and is subjected to several post-transcriptional modifications, which are vital for the correct translation into proteins:

- **Processing of 3' ends**
  The 3' end is cleaved at a cleavage site (often containing the sequence `CA`) and a tail of about 200 adenine residues, the so-called polyA tail, is appended. The signal for polyadenylation is located near the 3' end of the mRNA and is often similar to `AATAAA`.

- **Processing of 5' ends**
  The 5' end of the transcript is also processed to protect it against degradation. A so-called 5'-cap is therefore added to the transcript.

- **Splicing**
  Additionally, most pre-mRNAs are spliced by a protein complex called spliceosome. During this process, some regions of the transcript, called introns, are removed and discarded, while the remaining segments, called exons, are joined together. Depending on the organism, the removed introns can be very large. For example, the transcribed sequence of the largest known human gene has a length of 2.5 Mbp, of which more than 99% is located within 79 introns and thus removed [1] [2].

  Splicing requires the correct detection of intron boundaries, where the 5' end, the donor splice site, almost invariantly displays a '`GC`' consensus sequence within a larger, less conserved consensus sequence. Likewise, the 3' boundary, the acceptor splice site is characterized by an '`AG`' consensus. Upstream of the `AG`, a region of high pyrimidine content, the poly-pyrimidine tract can be found. The mechanism underlying splicing additionally requires a so called branch point upstream of the poly-pyrimidine tract, which includes an adenine nucleotide.

  It is also well known that the lengths of the individual segments follow specific distributions (see Figure 2.5) and are important properties for proper biological function. For example, it has been shown that the removal of internal portions of exons to sizes below 50 bp often results in exon skipping, i.e. improper splicing and exclusion from the final major mRNA [50]. On the other hand, there is evidence that the extension of exons to lengths greater than 300 bp may interfere with correct splicing [109, 124]. The lengths preferences for

[1] This is particularly remarkable when considering the fact that it takes over 16 hours to produce a single transcript.

exons also appear to depend on the position of the exon within the gene (see Figure 2.5).

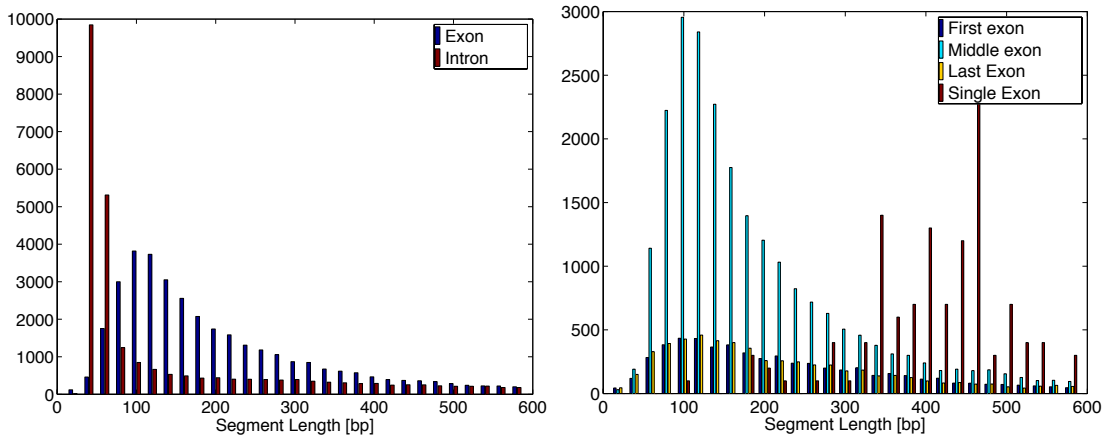


Figure 2.5.: Histogram of segment lengths, derived from 6245 confirmed annotated *C. elegans* genes (WS180), with a bin size of 20 bp. Left: Occurrence of exon and intron lengths. The bar plot is cut at 600 bp, covering 94% and 85% of all observed exons and introns, respectively. Maximal observed exon length is 4876 bp and longest intron measures 72,475 bp. Right: Length distribution for first, middle, last and single exons separately. Number of single exons are scaled by a factor 100.

Translation

After completion of the modifications, the mature mRNA is exported from the cell nucleus into the cytoplasm, where it is used as a template to synthesize proteins. Ribosomes read the nucleotide sequence in triplets, called codons, and add one specific amino acid per codon to an emerging polypeptide chain. To do so, they initiate the binding of a particular transfer RNA (tRNAs) with corresponding anti-codon, to each codon, where a specific amino acid is attached to the different types of tRNAs. A start codon, which is `ATG` in most cases, initiates this process and the ribosome moves on in steps of three nucleotides, again in the 5' to 3' direction. The sequence upstream of the start codon is called 5' untranslated region (5' UTR) and it contains many regulatory motifs that direct the synthesis of the protein. The translation stops before the actual end of the mRNA, when the ribosome encounters one of three possible stop codons, namely `TAA,TAG or TGA`. The remaining sequence downstream of the stop codon is called 3' untranslated region (3' UTR). The three positions of a codon are in phase 0, 1 or 2, and a feature starting in phase 0 is simply called to be 'in-phase'. The regions between the start codon and the stop codon,

which are thus translated, are collectively called coding segments (CDS). As codons on the CDS are non-overlapping, the length of the CDS is required to be a multiple of three. To avoid an early stop of the translation, no other stop codons may be contained in-phase before the final stop codon.

The translation from codon to amino acid follows the genetic code, which is universal to all living organisms. While there are $4^3 = 64$ different possible codons, only 20 standard amino acids are used. Hence the code is redundant, and multiple, synonymous codons code for the same amino acids. In particular, the first base of the anti-codon in the tRNA is spatially less constrained by the ribosome than the other two bases, such that different nucleotides at the third position of the codon may bind to the same type of tRNA, thus specify the same amino acids [33, 41]. This degenerate site is also called the wobble position. Notably, the frequency of the different codons is highly non-uniform in coding sequences. This observation is known as codon bias and it is widely exploited in most gene finding systems.

The structure of a typical gene is summarized in Figure 2.6. It includes all important signals as well as the corresponding segments of a gene.

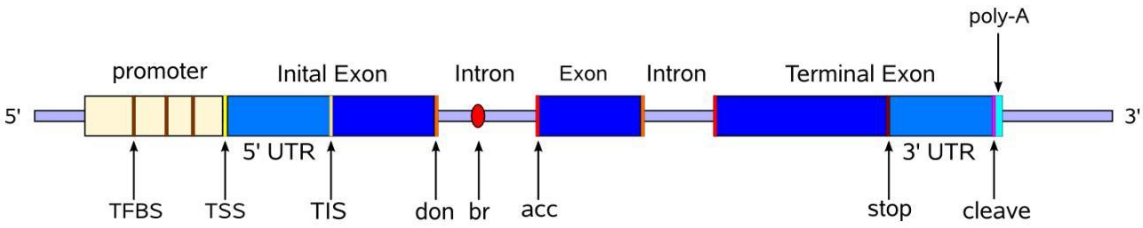


Figure 2.6.: Structure of a typical gene including a promoter region, 5'UTR, several exons and introns and the 3' UTR. Functional elements that induce segment boundaries include, transcription factor binding sites (TFBS), the transcription start site (TSS), the translation initiation site (TIS), several donor and acceptor splice sites (don, acc) and the corresponding branch points (br), the stop codon, a poly-A signal and the cleavage site (cleave). The image is taken from [184]

Alternative Transcripts

The one-gene-one-protein hypothesis as described above was challenged in the late 1970s, when it became apparent that several structurally different products, or isoforms, can stem from the same gene, thus widely increasing the diversity of proteins encoded by the genome. The variation can occur on all steps of gene expression: Alternative transcription start or cleavage sites can be used, or translation may start or stop at different sites. More over, by alternative splicing, the exons of a gene can be reconnected in multiple ways. In particular, five different main modes are distinguished, namely exon skipping, intron retention, mutually exclusive exons, al-

ternative donor or acceptor sites. The different forms are displayed in Figure 2.7. In eukaryotic cells, alternative splicing is in fact a frequent phenomenon. For example, it is estimated that in humans more than 90% of all multi-exon genes are subject to alternative splicing [179]. The regulation occurs by specific splice factors that bind to certain motifs on the RNA thus promoting silent splice sites or repressing otherwise active ones. While it is known that alternative splicing plays an important role in tissue and condition specific expression of genes, this splicing code is still not entirely understood, despite remarkable progress recently reported [10]. It is also one of the complications that impede accurate gene finding.

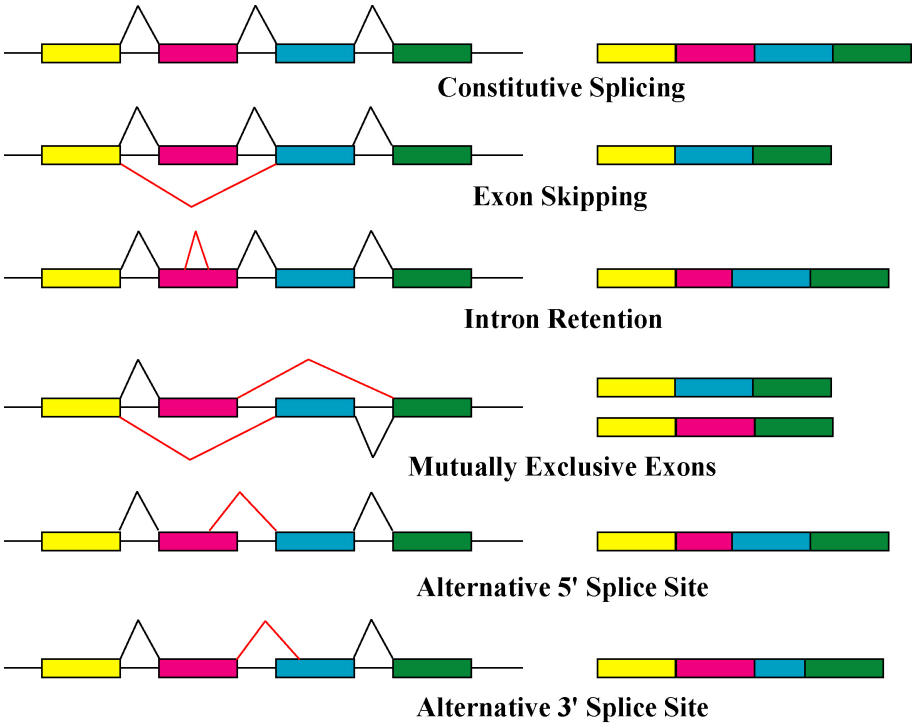


Figure 2.7.: Types of alternative splicing. Depicted is an exemplary gene with four exons, where alternatively used splice sites are connected in red. Resulting gene products are shown on the right. This illustration is taken from [102].

Pseudo-genes

Due to the effect of evolutionary forces, the DNA contains constructs that, albeit defunct, have the appearance of genes. These so-called pseudo-genes comprise another biological phenomenon that is likely to mislead the annotation of genomes. There are three mechanisms of origin distinguished ([110], see also Figure 2.8): *Processed pseudo-genes* are produced by retro transposition, where portions of the mature mRNA are reverse transcribed and inserted back into the DNA. These pseudo-genes

are usually intron-less, and, as the promoter region is missing, they are 'dead on arrival'. *Non-processed pseudo-genes*, on the other hand, are the result of gene duplication. They have an intact intron-exon structure, as well as an active promoter region. These genes are initially active gene copies, which are called paralogous genes. Due to little conservation pressure, they may accumulate mutations and subsequently be inactivated. There are also examples of *disabled genes*, where mutations have rendered a gene nonfunctional while the genome lacks an additional active gene copy [189].

The annotation of pseudo-genes vs. functional gene copies is generally difficult and often requires manual curation, which has led to false classifications of actually true genes. Also, interestingly, some pseudo-genes seem to have regained function [163]. Therefore, the very definition of pseudo-genes is presently under discussion.

Non-coding genes

There are also a number of different types of transcriped genes that are functionally important but not translated into proteins, so-called RNA genes. The resulting non-coding RNAs (ncRNA) include highly abundant classes such as transfer RNAs (tRNAs) and ribosomal RNAs (rRNAs), which play crucial roles in translation. Other ncRNAs (like, for example, snoRNAs, microRNAs, siRNAs) are involved in RNA modifications and gene regulation. RNA genes may overlap protein-coding genes. However, their architecture is quite different from protein-coding genes: Their sequence is typically shorter, many are not polyadenylated and not spliced. Most importantly, they lack an open reading frame and their sequence doesn't show the typical codon bias. On the other hand, their secondary structure is essential for proper function. To date, there are several computational methods that detect RNA genes, however, they are typically not considered in gene finders that are designed for protein-coding genes.

2.2. Experimental Methods for Gene Detection

There are several methods to experimentally detect transcripts. One commonly used technology generates so-called Expressed Sequence Tags (ESTs), which are single DNA sequencing reads. To generate ESTs, (usually poly-adenylated) mRNA is isolated from cells and a reverse transcriptase enzyme is used to convert the unstable mRNA into the more stable complementary DNA (cDNA), which represents the expressed DNA sequence. The cDNAs are then sequenced from either end to generate EST sequences of usually around 500 nucleotides. To identify the loci of expressed genes, these fragments are aligned to the genome by computational means. A major draw-back of this method is that ESTs usually do not cover the complete gene

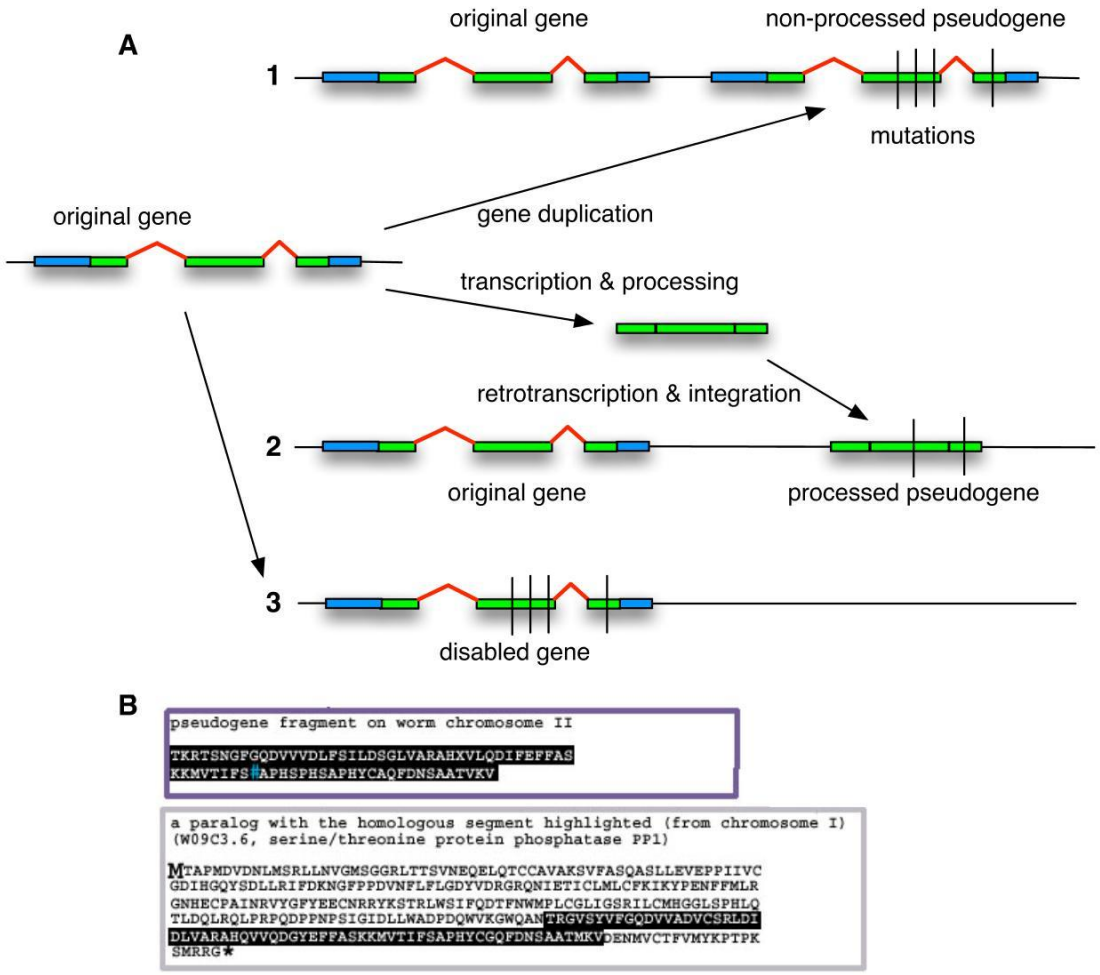


Figure 2.8.: **A** Mechanisms of pseudogene creation: 1) gene duplication can lead to non-processed pseudo-genes, 2) processed pseudo-genes are created by retro-transcription and insertion of major mRNA transcripts. 3) In rare cases single copy genes are disabled by gathering mutations. **B** Example of truncated amino acid sequence encoded by a pseudogene with respect to its paralog. Example taken from [74].

and the recovered sequences are often not very accurate. Gene starts and ends are particularly difficult to assign. Additionally, low expressed genes or isoforms are often missed completely. However, ESTs have been generated by large sequencing centers as well as individual scientists and are stored in the public dbEST database, which now contains approximately 63 million ESTs from many different species. An alternative to ESTs are full-length cDNAs that cover complete genes and sequences that are recovered via mapping of cDNAs are usually more precise i.e. include less sequencing errors. EST and cDNA data can be used to generate confirmed gene

sets. These sets are necessary to train computational gene finders. Often, they are also employed in combination with gene finders, which use the experimental data as additional hints to predict gene structures.

Recently, high-throughput sequencing technologies are used for RNA-sequencing, (RNA-Seq). While a plethora of new data is produced with these techniques — it allows to analyse complete transcriptomes — the precise information about the location of genes and their structure is not easily extracted and requires the development of new computational methods.

2.3. Evolutionary Considerations

The common descent of all living species has fostered the strategy to study a particular organism as a prototype with regard to certain biological questions. Subsequently, it is often possible to deduce more general principles from the acquired knowledge, which are then valid for a wider range of organisms. This strategy has also facilitated many valuable insights into potential causes and treatments of human diseases, where feasibility and ethical considerations would otherwise prohibit the necessary experimentation.

In this thesis, I will mainly examine the behaviour of the gene finding system mGene when applied to the genome of the model organism *Caenorhabditis elegans*. However, the long-term aim remains to construct a universal system that is able to predict protein-coding genes in all eukaryotes. Viewed superficially, the transfer to other organisms should not pose big challenges because the biological principles of transcription, post-processing and translation as described in Section 2.1 are common to all eukaryotes. However, although the basic principals are conserved, the signals on the DNA, as well as corresponding trans-factors have considerably evolved. For instance, the unique splicing mechanisms for several organisms have been investigated in [138] and the respective motifs are also shown in Figure 2.9. Note, that considering multiple organisms is an additional dimension of complexity, as the motifs observed for different genes *within* one organism already vary widely among each other. To extract common rules, it is usually necessary to analyze as many known examples as possible. This large amount of data is only available for a few model organisms. In Chapter 8 we will therefore investigate methods that allow the adaptation of a learnt model to other organisms, for which only little data is available.

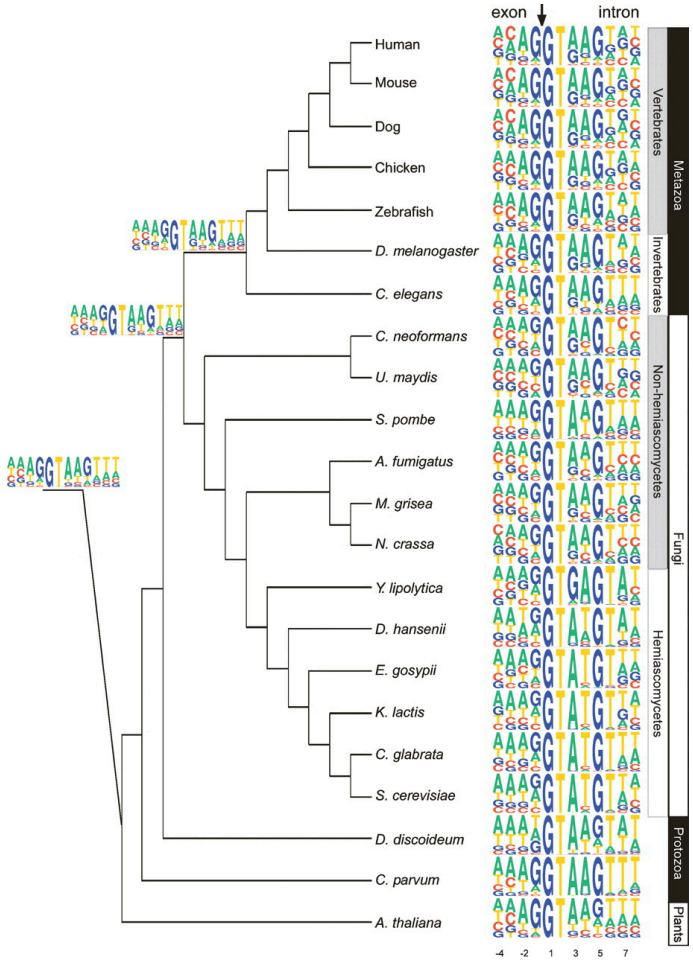


Figure 2.9.: Sequence logos of the donor splice sites of 22 organisms spanning the kingdoms of eukaryotes from plants to animals. The organisms are displayed according to their phylogenetic grouping. For each organism, position-specific scoring matrices (PSSMs) containing the frequency of each nucleotide at each position were generated (i.e. the height of each letter is proportional to the frequency). Additionally, shown are pictograms of reconstructed sequence motifs at three ancestral nodes. Apparent is the strong evolution of the 5' splice site: It is strongly conserved among some fungi, but highly degenerate among most other organisms. This figure is taken from [138].

2.4. C. elegans as a Model Organism for Gene Finding

As a playground for the gene detection task I will mainly focus on the nematode (roundworm) *Caenorhabditis elegans*, which was chosen in the 1970's by Sydney Brenner as a model animal for developmental biology and genetics research [22]. Today, there is a large dedicated on-line repository, the WormBase database, that collects and actively curates all published research results on this species. *C. elegans*

was also the first multi-cellular organism with a completely sequenced genome. The official version of this sequence continues to change when new evidence reveals errors in the original sequencing. The reference genome is 100.2Mb long and contains approximately 20,000 genes distributed over six chromosomes. Notably, the organism is simple enough to be studied in detail, however, its genes share most of the complex properties of mammalian genes. In particular, most genes are spliced, and also alternative splicing occurs frequently. Additionally, a splice leader sequence (known as SL1 and SL2) that originates from a different genomic location is added to many pre-mRNAs. This process is known as trans-splicing and it prevails in nematodes (e.g. 70% of pre-mRNAs in *C. elegans*) [65] while absent or fairly insignificant in most other organisms. Another peculiarity are operons, where several genes are simultaneously transcribed, regulated by a common promoter.

Recently, the genome sequence of other nematodes from the same genus such as *C. briggsae*, *C. remanei*, *C. japonica* and *C. brenneri*, have been determined, allowing a comparative genomics study of these organisms.

2.5. Mathematical Framework of the Gene Prediction Problem

To locate transcripts in the genome and to determine the correct structure of genes, it would be desirable to find a physical description of all the cellular process described above. This would not only allow the localization of genes and protein-coding regions on the DNA, but also result in an explanatory model of the processes that give rise to these structures. However, such an approach would require modeling all protein-DNA interactions, which is unrealistic while the three dimensional structure of many involved proteins is still unknown and in fact key players might even be undiscovered. Therefore, to date, successful gene prediction algorithms are rather based on supervised learning techniques, where statistical properties on the DNA of known genes are observed to make predictions of undiscovered genes on unseen DNA sequences.

In general, given a genomic DNA sequence, the correct annotation that separates intergenic from genic regions, and coding from non-coding segments has to be found. From now on, any finite sequence of symbols over the four-letter alphabet $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ will be denoted with $\mathbf{x}$, e.g. $\mathbf{x} = x_1, x_2, \ldots, x_L$ with $x_i \in \Sigma$, $\forall i = 1, \ldots, L$. The corresponding annotation $\mathbf{y}$ can now either be described as a sequence of labels, where a tag $y_i \in \mathcal{Q}$ from the set of all labels is assigned to each single entity x_i: $\mathbf{y} = y_1, y_2, \ldots, y_L$. Alternatively, the partitioning can also be described as a sequence of segments: $\mathbf{y} = \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M$, with $M \leq L$. Here, the segments are described as four tuples $\mathbf{y}_j = (b_j, b'_j, y_j, y'_j)$, which are characterized by their start and end

positions, b_j and b'_j, defined relative to the beginning of the sequence $\mathbf{x}$, and the labels of the segment boundaries at these positions, $y_j, y'_j \in \mathcal{Q}$. The segments are considered as right-opened intervals $[b_j, b'_j[$ and for all that follows I will require a complete segmentation, such that each segment must begin at the same position where the previous one ends, i.e. $b_j = b'_{j-1}$; together, all segments need to cover the input sequence $\mathbf{x}$, i.e. $b_0 = 0$ and $b'_M = L + 1$. Additionally, the label at the beginning of any segment needs to be identical to that at the end of the preceding segment, i.e. we have $y_j = y'_{j-1}$. Also, in the presented setting, the type of the segment is unambiguously defined by the types of the two boundaries, y_j and y'_j, e.g. an exon segment is determined by an acceptor boundary preceding a donor boundary (compare Figure 2.10).

Note that, while the two formulations for the segmentation $\mathbf{y}$ are equivalent, the set of employed labels $\mathcal{Q}$ are different in the two cases: In general I will distinguish between labels at segment boundaries, like acceptor or donor splice sites and labels for inner-segmental positions or complete segments, like introns or exons. An illustrative example of the two different formulations is given in Figure 2.10. Depending on the described method, I will switch between the two representations.

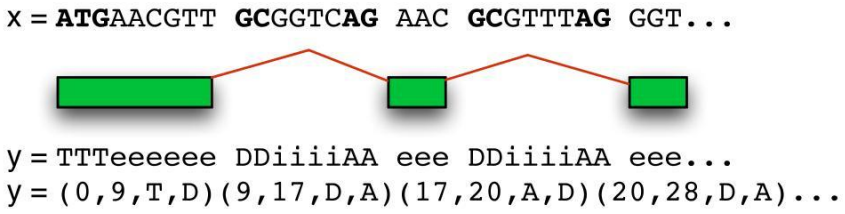


Figure 2.10.: Example segmentation $\mathbf{y}$ of a DNA sequence $\mathbf{x}$. The boundary labels include the Translation Initiation Site ('T') and donor and acceptor splice sites ('D' and 'A'). Segment labels include exonic and intronic positions ('e' and 'i').

The task of computational gene finding can now be approached in different ways: A naive strategy would be to query a label $y_i \in \mathcal{Q}$ for every single base x_i and to formulate the respective multi-class classification problem. The input to this problem could be the sequence $\mathbf{x}_{[i-d,i+d]}$ within a defined window $[i-d, i+d]$ around each base and it could be solved with independent binary classification tasks for each label $q \in \mathcal{Q}$:

$$F^q : \mathcal{X}^{[2d+1]} \to \{\pm 1\},$$

where $\mathcal{X}^k$ denotes the set of all sequences of length k. The learnt function F^q assigns a value $+1$ to $\mathbf{x}_{[i-d,i+d]}$ if the associated label is q and -1 otherwise. I will show that this strategy works well for some labels, particular segment boundaries. However, it is incapable of capturing and reproducing the correct syntax of valid genes.

An alternative approach is to consider the complete sequence $\mathbf{x}$ and to treat each possible label sequence $\mathbf{y} \in \mathcal{Y}$ as a distinct "class" in a multi-class classification problem with independent classes:

$$G : \mathcal{X} \to \mathcal{Y},$$

with $\mathcal{X} := \Sigma^*$ and $\mathcal{Y} := \mathcal{Q}^*$, where * denotes the Kleene closure of the set, i.e. the set of all strings over the symbols in the set. This problem, however, is not able to capture common properties shared by different classes and therefore prohibits the generalization across classes. Both approaches have the drawback that valuable information about the context and internal structure of the label sequence $\mathbf{y}$ is discarded and not employed to improve prediction accuracy.

This information, the *grammar* of genes, is often represented in a directed graph, $\mathcal{G}(\mathcal{Q}, \mathcal{T})$, with a set of vertices or states, $\mathcal{Q}$, that are connected by edges or transitions, $\mathcal{T}$ (see Figure 2.11). Note that in this representation, the set of states is equivalent to the set of labels and only grammatically allowed transitions between states are shown as directed edges. Therefore, not all segmentations are possible, but a valid segmentation has to obey certain biological rules. Examples of such rules are: 'A gene starts with a transcription start site' or 'Any donor has to be followed by an acceptor'. The gene finding task is then treated as a parsing problem akin to problems arising in natural language processing or speech recognition, which account for a complex grammar observed in the output sequence. Parsing techniques developed in these fields are particularly well suited for the problem as they take the high ambiguity into consideration that is a characteristic of both, natural language, and the language of genes and distinguishes them from computer language. One of the most successful of such techniques are hidden Markov models, which are also widely applied in conventional gene finding algorithms and will therefore be described in Section 2.7. Initially, however, evaluation criteria for the comparison of methods will briefly be defined.

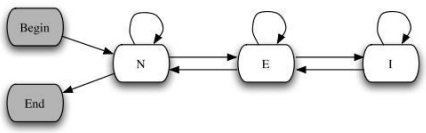


Figure 2.11.: Example graph with the states: 'N' for intergenic, 'E' for exonic and 'I' for intronic bases and additionally a start and end state.

2.6. Evaluation Criteria

If the annotation of sequences is treated as a number of binary classification tasks, the area under the receiver operator characteristic curve (auROC) or the area under the precision recall curve (auPRC) can be used as measures to evaluate the performance of the algorithms. Therefore, the following definitions are necessary: The sensitivity (also known as recall or true positive rate), SN, is defined as the fraction of correctly classified positive examples among the total number of positive examples. Analogously, the fraction of negative examples wrongly classified positive is called the false positive rate (FPR). The specificity (also called precision or positive predictive value) on the other hand is given as the fraction of correct positive predictions among all positively predicted examples:

$$\begin{aligned} SN &= \frac{TP}{TP+FN} \\ FPR &= \frac{FP}{FP+TN} \\ SP &= \frac{TP}{TP+FP}. \end{aligned}$$

Plotting FPR against SN results in the receiver operator characteristic curve (ROC curve) [46, 54]. Plotting the specificity against sensitivity, leads to the precision recall curve (PRC) [100, 103]. For very unbalanced datasets the auROC is less meaningful, since this measure is independent of class ratios.

The output of gene finding algorithms is more complex than a simple class label, which naturally leads to intricate routines for the comparison of predictions and the evaluation of prediction methods. The task is further complicated by the fact that due to alternative splicing, several label sequences have to be taken into account as potential true annotations. And likewise, as some gene finders are capable of generating more than one prediction for a given locus, multiple predictions have to be considered simultaneously. Yet another pitfall arises from the fact that the labels are often afflicted by uncertainties: Even in the sequence regions that are used for evaluation, not all genes might yet be identified. On the other hand, annotated genes might originate from earlier predictions and lack experimental confirmation.

The highest uncertainties are observed for gene starts and stops. Therefore, most evaluation criteria only analyze coding regions. There are typically several measures of importance in gene finding (see Figure 2.12) : First, prediction and annotation are compared on a per nucleotide basis. Second, the co-occurrence of whole segments in prediction and annotation is counted. This measure particularly considers the correct detection of segment boundaries. Eventually, the number of completely agreeing transcripts is counted, where each labeled and each predicted isoform is

treated independent of the respective locus from which it stems. For a transcript to be counted as correct, each labeled coding exon has to be correctly predicted, and in the predicted transcript no additional coding exon may occur. As a somewhat relaxed measure there is also the gene count, that assumes a gene to be correctly predicted if at least one transcript is completely correct. For each of the four levels, i.e. nucleotide, exon, transcript and gene, sensitivity and specificity are reported and for a better comparison between algorithms I will also often give the mean between sensitivity and specificity. These evaluation criteria have similarly been used in the genome annotation competitions EGASP [71] and nGASP [34].

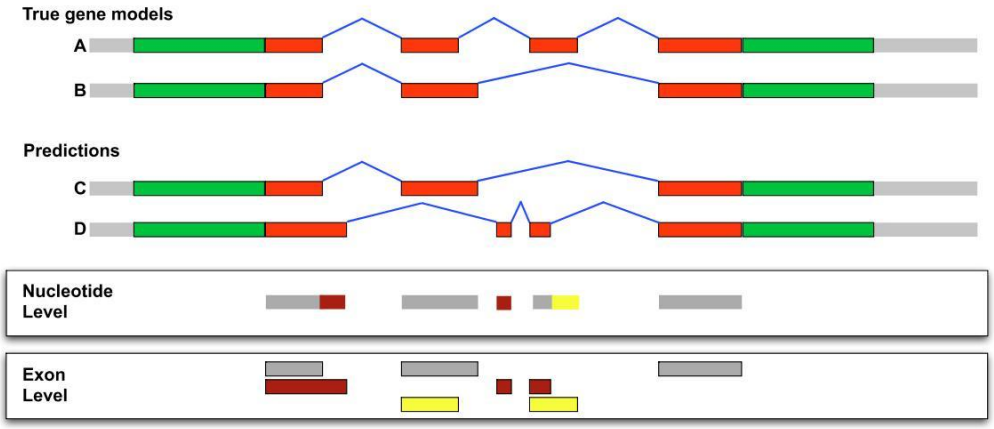


Figure 2.12.: Evaluation example. Annotated is an mRNA with two isoforms (A and B) and a total of five different coding exons. Predicted is a mRNA with also two isoforms (C and D) and a total of six coding exons. The first predicted transcript C is identical to the second annotated one, B. However, in the second predicted isoform D, the first exon is predicted too long and the second exon has no counterpart in the annotation at all. The added coding nucleotides are counted as false positives (marked in dark red) in the nucleotide measure. Additionally, the third exon is predicted too short. The corresponding nucleotides are counted as false negatives (marked in yellow). For the respective length of the segments, this results in a sensitivity of 0.9 and a specificity of 0.85. On the exon level, there are three correctly predicted exons, three false positives and two false negatives, amounting to a sensitivity of 0.6 and a specificity of 0.5. Transcript sensitivity and specificity are both 0.5, and gene sensitivity and specificity are 1.0.

2.7. Typical Design of Computational Gene Finders

Most early gene finders were based on generative approaches like HMMs [51, 98], these include for example GeneMark.hmm [18, 97], HMMgene [88], and Unveil [99]. An extension of HMMs called generalized HMMs (gHMM) or hidden semi-Markov models [89, 116] was incorporated in some systems, most notably Fgenesh [127] and Augustus [153]. All of these algorithms are conceptually fairly similar, as they share the generative approach to gene finding. The underlying HMM framework will be explained in more detail in this section. It is based on the presentation of the subject in [98].

Hidden Markov Models

An HMM is a stochastic machine that captures the statistical and syntactical properties of sequences, thereby generating strings in a non-deterministic manner. It is defined over an alphabet Σ, a state set $\mathcal{Q}$, and the probability distributions for transitions $p_t(q'|q)$ between pairs of states q and q' as well as the distributions for emissions of symbols s when in state q, $p_e(s|q)$. Included in the set of states are a start state q_{Begin}, with zero in-coming edges and a stop state q_{End} with zero outgoing edges. The operation of the HMM is depicted in Figure 2.13: It begins in the start state $y_0 = q_{Begin}$, and subsequently transitions stochastically from one state to another, according to the probability distribution $p_t(y_i|y_{i-1})$. As the current state is only depending on the immediately preceding state, this model is called a first-order HMM. At each traversed state y_i, it randomly emits a symbol x_i from the alphabet Σ, according to the probability distribution $p_e(x_i|y_i)$. In this model, there are no direct dependencies between observations. Eventually the machine ends in the final state $y_{L+1} = q_{end}$. The start and end states, q_{Begin} and q_{End}, are both silent states, i.e. they don't emit any symbols. During run time of the HMM the sequence of states is concealed, only the emitted symbols are visible.

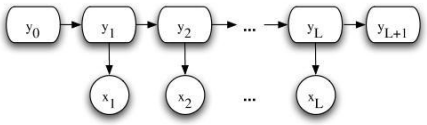


Figure 2.13.: An HMM generates a sequence $\mathbf{x} = x_1, x_2, .., x_n$ by starting in a silent state $y_0 = q_{begin}$ and stochastically advancing from state y_{i-1} to state y_i according to the transition probability $p_t(y_i|y_{i-1})$, while emitting at each state a symbol x_i according to $p_e(x_i|y_i)$.

A graphical representation of a simple HMM, $M = (\Sigma, \mathcal{Q}, p_t, p_e)$, for gene structures is shown in Figure 2.14. Note, that in general the topology of an HMM is

assumed to be fully connected, i.e. there are transitions between all pairs of states. In practice, however, many transitions are assigned zero probability, prohibiting their use. These transitions are omitted in the graph.

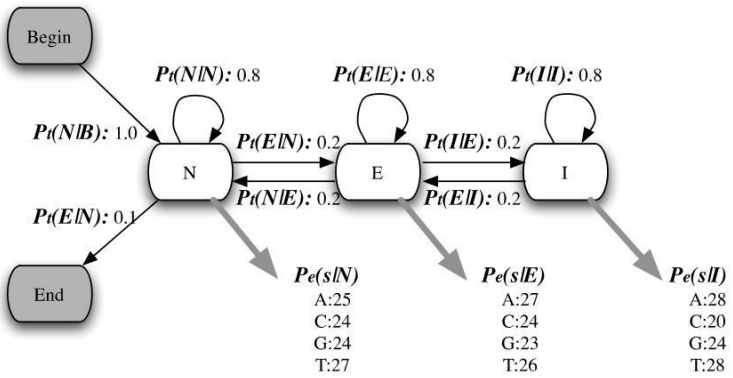


Figure 2.14.: A simple model for gene structure prediction, $M = (\Sigma, \mathcal{Q}, p_t, p_e)$, consisting of three states for intergenic N, exonic E and intronic I positions, and additionally the start and stop states: $\mathcal{Q} = \{N, E, I, Begin, End\}$. For all allowed transitions, the respective transition probabilities $p(q'|q)$ are shown. At each state, one of the four letters $s \in \Sigma$ will be emitted according to a probability $p_e(s|q)$.

Decoding

For gene finding, one is interested in the sequence of states $\mathbf{y}$ that generates the observed sequence $\mathbf{x}$. However, in HMMs the sequence of states are hidden, and the applied models tend to be highly ambiguous in the sense, that many different state sequences $\mathbf{y}$ are capable of producing the same sequence $\mathbf{x}$. An appropriate decoding algorithm has to be applied to determine the most probable path $\mathbf{y}^*$ for a particular sequence $\mathbf{x}$, given model M:

$$\begin{aligned} \mathbf{y}^* &= \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}} \log p(\mathbf{y}, \mathbf{x}|M) \\ &= \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}} \log p(\mathbf{y}|\mathbf{x}, M). \end{aligned} \tag{2.1}$$

as the later equality follows from the invariance of $p(x)$ under the argmax, $\mathbf{y}^*$ is equivalent to the maximum *a posteriori* path.

The log-probability $\log p(y|x)$ is positionally decomposable:

$$\log p(\mathbf{y}|\mathbf{x}, M) = \sum_{i=1}^{L+1} (\log p_t(y_i|y_{i-1}) + \log p_e(x_i|y_i)) \quad (2.2)$$

$$= \sum_{i=1}^{L+1} g(y_{i-1}, y_i, x_i). \quad (2.3)$$

Therefore, the decoding procedure is typically done by a dynamic program, the Viterbi algorithm. It uses the Viterbi matrix V, which is recursively defined as:

$$V(i,k) = \begin{cases} \max_{l \in \{1..|\mathcal{Q}|\}} (V(i-1,l) + g(q_l, q_k, x_i)) & if \quad i > 1 \\ 0 & if \quad i = 1 \end{cases}, \quad (2.4)$$

where $V(i,k)$ represents the probability of the most probable path under the model M, which ends in state q_k at position i and emits the sequence $x_1...x_i$.
To keep track of the respective preceding path, the traceback matrix T stores the link that gave rise to $V(i,k)$, i.e. the state at position $i-1$ of the optimal path ending in q_k at position i:

$$T(i,k) = \begin{cases} \text{argmax}_{l \in \{1..|\mathcal{Q}|\}} (V(i-1,l) + g(q_l, q_k, x_i)) & if \quad i > 1 \\ 0 & if \quad i = 1 \end{cases} \quad (2.5)$$

The optimal path $\mathbf{y}^*$ is found by following the traceback pointers T in reverse, starting at $T(L+1, q_{End})$, until $T(0, q_{Begin})$ is reached. For a sequence of length L, the run time for the Viterbi algorithm is $\mathcal{O}(|\mathcal{Q}|^2 \times L)$ as for each of the $|\mathcal{Q}| \times L$ cells in the Viterbi matrix the function g has to be evaluated $|\mathcal{Q}|$ times.

Training

In the supervised setting, the model parameters are estimated from N labeled examples $(\mathbf{x}^n, \mathbf{y}^n)$. This is typically done by computing the maximum likelihood estimates for the transmission probabilities p_t for all pairs of labels, and likewise for the emission probabilities p_e for all symbol-label pairs:

$$\begin{aligned} p_t(q_k|q_l) &= \frac{A_{k,l}+c}{\sum_{m=1}^{|\mathcal{Q}|}(A_{k,m}+c)} \quad \text{with} \quad A_{k,l} = \sum_{n=1}^{N}\sum_{i=1}^{L_n} \mathbb{I}(y_i^n, q_k)\mathbb{I}(y_{i-1}^n, q_l), \\ p_e(s_k|q_l) &= \frac{E_{k,l}+c}{\sum_{m=1}^{|\mathcal{Q}|}(E_{k,m}+c)} \quad \text{with} \quad E_{k,l} = \sum_{n=1}^{N}\sum_{i=1}^{L_n} \mathbb{I}(y_i^n, q_k)\mathbb{I}(x_i^n, s_l). \end{aligned} \quad (2.6)$$

where I used the indicator function $\mathbb{I}(x, x')$, which is 1 if $x = x'$ and 0 otherwise.

$A_{k,l}$ is the number of times label q_k is observed to immediately follow label q_l in N training examples, with L_n being the length of sequence $\mathbf{x}_n$, and $E_{k,l}$ is the number of times symbol s_k is observed to be emitted by state q_l. To ensure that the probabilities remain non-zero even if a certain event does not occur in the training examples, pseudo-counts c are introduced (i.e. small numbers of artificial observations of each type of event) [51].

Note, that this form of maximum likelihood estimation amounts to maximizing the joint probability of the input sequences $\mathbf{x}^n$ and their paths $\mathbf{y}^n$ for the training set $n = 1...N$:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left(\prod_{n=1}^{N} p(\mathbf{x}^n, \mathbf{y}^n | \boldsymbol{\theta}) \right), \tag{2.7}$$

where $\boldsymbol{\theta}$ denotes the complete set of estimated model parameters, i.e. all values of p_e and p_t. While this is easily computable, it might not be optimal for predicting the most accurate parse with the Viterbi algorithm as it does not guarantee a large discrimination between different possible labels. Currently, discriminative training techniques for probabilistic models are being developed [90], which approximate the conditional maximum likelihood parametrization that seems to be a more appropriate training goal.

Generalized Hidden Markov Models

One major shortcoming of the HMM approach as described above is the fact that each state is constrained to emit a single symbol at a time. This effectively leads to modeling the lengths of segments with a geometric distribution, $p^d \cdot (1-p)$, regardless of the true length distribution of the segment [98]. As a simple example consider a two-stage HMM for exons, with only one non-silent state e, and the start and end state q_0, then the probability of a putative exon $\mathbf{x}_{[1,d]}$ is given by:

$$p(\mathbf{x}_{[1,d]}|\boldsymbol{\theta}) = \left(\prod_{i=1}^{d} p_e(x_i|e) \cdot p_t(e|e) \right) \cdot p_t(q_0|e) \tag{2.8}$$

$$= \left(\prod_{i=1}^{d} p_e(x_i) \right) \cdot p_t^d (1 - p_t), \tag{2.9}$$

where I have omitted the conditional for the emission probabilities in the second equation, because there is only a single emitting state possible. Also, for the transmission probabilities, there is only the probability for self transitioning $p_t = p_t(e|e)$, and for ending of the exon $p_t(q_0|e) = 1 - p_t$.

The important extension in generalized hidden Markov models (gHMM) is, that the machine is allowed to emit a string of arbitrary length at each state [89, 116]. With this generalization it becomes possible to model more adequate length distributions, e.g. arbitrary histograms. For that, the emission probability is separated from the duration probability by introducing the additional term $p_d(d|q)$ denoting the probability that a sequence of length d will be emitted in state q, and additionally conditioning the emission probability for a string $\mathbf{s}$ on the duration d, $p_e(\mathbf{s}|q,d)$. Equation thus 2.2 becomes:

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}) &= \sum_{j=1}^{M+1}(\log p_t(y_j|y_{j-1}) + \log p_e(\mathbf{x}_{[b_j,b'_j]}|y_j,d_j) + \log p_d(d_j|y_j)) \\ &= \sum_{j=1}^{M+1} g'(y_{j-1},y_j,\mathbf{x},b_j,b'_j), \end{aligned} \tag{2.10}$$

where b_j and b'_j are the start and stop position of segment j, $d_j = b'_j - b_j$ denotes its length and $\mathbf{x}_{[b_j,b'_j]}$ the emitted sub-sequence. The decoding problem of gHMMs can be solved with a generalized version of the Viterbi algorithm. In this case, however, dealing efficiently with variable length states can be challenging: For each cell in the Viterbi matrix not only the cells in the immediately preceding column have to be considered, but also cells arbitrarily far back, if segments of arbitrary length are allowed. The Viterbi matrix is then given by:

$$V(i,k) = \max_{l=1\ldots|\mathcal{Q}|} \max_{d=1\ldots i-1} (V(i-d,l) + g'(q_l,q_k,\mathbf{x},i-d,i)) \tag{2.11}$$

if $i > 1$ and $V(i,k) = 0$ otherwise. The optimal label sequence can be obtained as before by backtracking. However, to do so, saving the states of the optimal predecessor only is not sufficient, but additionally the respective positions $i-d$ have to be remembered:

$$T_1(i,k) = \operatorname*{argmax}_{l=1\ldots|\mathcal{Q}|} \max_{d=1\ldots i-1} (V(i-d,l) + g'(q_l,q_k,\mathbf{x},i-d,i)) \tag{2.12}$$

$$T_2(i,k) = i - \operatorname*{argmax}_{d=1\ldots i-1} \max_{l=1\ldots|\mathcal{Q}|} (V(i-d,l) + g'(q_l,q_k,\mathbf{x},i-d,i)) \tag{2.13}$$

for $i > 1$ and $T1,2(i,k) = 0$ otherwise. Without any further assumptions the run-time becomes quadratic in the length of the sequence $\mathcal{O}(|\mathcal{Q}|^2 \times L^2)$ as for each cell in the Viterbi matrix, g' has to be evaluated $|\mathcal{Q}| \times L$ times. In practice, the maximal look-back is determined by the maximal segment length D, leading to a run-time

$\mathcal{O}(|\mathcal{Q}|^2 \times L \times D)$. This can still be problematic in gene finding, as some segments, like intergenic or intronic regions can be of great length (e.g. > 400 *kbp* for human introns). For an efficient implementation, most gene finders therefore exploit some additional assumptions, which reduce the number of potential predecessors [98]. For example, a geometric length distribution can be used as a good approximation for intergenic segments and — for most organisms— also for long intronic segments. Additionally, the emission probabilities are required to be factorable, such that the log probability can be expressed as a sum of terms evaluated at each position in the sequence:

$$\log p_e(\mathbf{x}|y_i) \quad = \quad \sum_{i=1}^{L} \log p_e(x_i|y_i). \tag{2.14}$$

With these two assumptions, the number of potential predecessor *positions* for starts of non-coding segments $N_{pred}^{NC\_Start}$ can be reduced according to Burge's non-coding predecessor theorem [26, 98]. On the other hand, the maximal look-up for starts of coding regions can be constrained because stop codons accumulate in random sequences, such that the start of a coding segment cannot be arbitrarily distant from the end of the segment, leading to $N_{pred}^{C\_Start}$ potential predecessor positions. Therefore, one needs to keep an active, type-specific *queue* of potential predecessors for each state, each with at most $N_{pred}^{state} < D$ elements.

Additionally, the topology of typical gene models can be assumed to be sparse, such that each traversed state has only a small number of valid predecessor states. The run-time is thus reduced to $\mathcal{O}(|\mathcal{Q}| \times L \times N_{pred} \times M_{in})$, where $M_{in} < |\mathcal{Q}|$ is the maximal number of incoming edges for any state.

Common Models and Sensors

In principale, a rather simple model like the one in Figure 2.14 should be sufficient for a simple gene finding and gene structure prediction tool. However, a major incentive for the design of the model is to incorporate as much biological knowledge as possible, while at the same time giving the method enough freedom for generalization. Typical gene finders therefore use far more sophisticated models, that capture many known rules in the transcription, RNA processing, and translation processes. However, as the model is static and predefined in advance, caution must be taken as defects cannot be compensated for during training. The model usually consists of several sub-models, which typically include sensors for discrete signals at segment boundaries, like splice sites, and detectors for variable length features, e.g. exons, which are called content sensors (see below). The sub-models are often trained in-

dependently [98]. The final HMM is then obtained by subsequently merging the individual sub-models into a meta-model, for which the transition probabilities between the sub-models have to be learned. While the composed model can be defined such that it is equivalent to a globally trained model, it is far more flexible as individual sub-models can easily be retrained or improved. Also, it allows the so-called parameter tying by pooling features into a single model: For example, it is necessary to include several exon states into the model to track the phase constraints across introns. Besides the phase, these exons might have very similar statistical properties. To use the available training data efficiently, it might therefore be advantageous to train a single exon sub-model, which is included several times into the meta-model. To allow the integration into an HMM, the algorithms employed for the sub-models are restricted to methods that compute properly normalized probability values.

Content Sensors

The content sensors typically comprise detectors for exonic and intronic segments and sometimes also intergenic regions. In contrast to signal sensors, the regarded sub-sequences have variable length, and also the relative position of strings within the sub-sequences is less important. For these models, variants of Markov chains are usually applied. They can be defined as 2-state HMMs, $\mathcal{Q} = \{q_o, q_1\}$, where q_o is a silent start and end state, and only the state q_1 is non-silent. The transition probabilities are then defined by a single parameter p_t, the probability of self-transitioning in state q_1, and the emission probabilities p_e are only defined for state q_1. The class conditional likelihood for an k-th order Markov chain is then given by:

$$p(\mathbf{x}_{[u,u']}|\boldsymbol{\theta}_c, d) = \prod_{i=u}^{u'} p_e(x_i|\mathbf{x}_{[i-k,i-1]}, d, \boldsymbol{\theta}_c), \tag{2.15}$$

where I have additionally conditioned on the length d of the considered sequence. Maximum likelihood estimation is used to determine the model parameters $\boldsymbol{\theta}_c$, i.e. in this case the emission probabilities p_e.

In inhomogeneous, or non-stationary Markov chains the model parameters change over the sequence positions. A specific type are three-periodic Markov chains, which are used to account for the codon usage biases in coding sequences. Here, one model $\boldsymbol{\theta}_\omega$ is trained for each of the three phases, and the class conditional probability of a sequence $\mathbf{x}_{[u,u']}$, now also depends on the phase in which the sequence starts:

$$p(\mathbf{x}_{[u,u']}|\boldsymbol{\theta}_\omega, d, \omega) = \prod_{i=u}^{u'} p_e(x_i|\mathbf{x}_{[i-k,i-1]}, d, \boldsymbol{\theta}_{(\omega+i) \mod 3}). \tag{2.16}$$

Another common variant of Markov chains are interpolated Markov models, where a weighted average of different order estimates is used [128, 129].

Signal Sensors

The signal sensors modeled by most gene finders comprise detectors for acceptor and donor splice sites and start and stop codons. Some gene finders additionally detect the transcription start site or the polyadenylation signal at the end of genes and other signals. All positions that exhibit the required consensus sequence for a given signal are queried by analyzing the DNA sequence surrounding the potential signal candidate. The position-dependent occurrence of letters or strings relative to the consensus is thereby crucial.

Techniques that are most often used include the position-specific scoring matrix (PSSM, PWM or PSM) [151], also called weight matrix method (WMM), weight array matrices (WAM) [193] and the windowed weight array matrices (WWAM) [27]. In all three methods, the respective model parameters $\boldsymbol{\theta}_s$ are determined by maximum likelihood estimation on the training set. For the simplest method, the PSSM, this comprises the nucleotide distribution for each position within a fixed-length window. The emission probabilities are thus of 0th order and the PSSM is recovered as zero-order inhomogeneous Markov chain. The WAM model is a generalization thereof to order k, i.e. emission probabilities are computed position-wise, conditioned on k previous bases. The WWAM approach is a further extension that allows some positional variation of occurring motifs: When the distribution of occurring k-mers is computed for each position, nearby occurring sub-strings are pooled.

Under these models the class conditional probabilities of a putative signal s at position i with a surrounding sequence $\mathbf{x}_{[i-\Delta_l,i+\Delta_r]}$ are computed as follows:

$$p_s^{PSSM}(\mathbf{x}_{[i-\Delta_l,i+\Delta_r]}|\boldsymbol{\theta}_s) = \prod_{j=-\Delta_l}^{\Delta_r} p(j, x_{i+j}|\boldsymbol{\theta}_s) \tag{2.17}$$

$$p_s^{WAM/WWAM}(\mathbf{x}_{[i-\Delta_l,i+\Delta_r]}|\boldsymbol{\theta}_s) = \prod_{j=-\Delta_l}^{\Delta_r} p(j, x_{i+j}, \mathbf{x}_{[i+j-k,i+j-1]}|\boldsymbol{\theta}_s) \tag{2.18}$$

$$\tag{2.19}$$

The class conditional probabilities of the signal models are used directly in the meta-model as emission probabilities $p_e(\mathbf{x}_{[i-\Delta_l,i+\Delta_r]}|q)$, where q is an appropriate state, e.g. an acceptor state when the applied signal sensor s is an acceptor sensor.

If the sensors are not integrated into a meta-model but used to classify individual sites, typically two models, $\boldsymbol{\theta}^+$ and $\boldsymbol{\theta}^-$, trained independently on positive and

negative examples are used. The posterior-log odds is than defined as:

$$g(\mathbf{x}) = log(p(\mathbf{x}|\boldsymbol{\theta}^+)) - log(p(\mathbf{x}|\boldsymbol{\theta}^-)) + b, \tag{2.20}$$

where b is a bias term and

$$f(\mathbf{x}) = sign(g(\mathbf{x}))$$

is used for classification. This will be used in Section 4 to compare initial results of signal sub-models.

Incorporation of Additional Data

So far, gene finding was defined as the problem of deciphering the DNA sequence in the same manner the cellular machinery sees and reads it to construct proteins. As most of these processes act on specific motifs, either on the DNA or on the RNA — where the latter can be back projected to the DNA — *ab initio* gene finding assumes that knowledge of the DNA sequence is sufficient to detect genes and to predict their precise structure. Clearly, this is an over-simplification, and other, maybe unknown aspects may also play a role in the various processes described above. Additionally, while DNA motifs *do* constitute very strong signatures of genes they are in most cases hard to decipher.

To improve the accuracy achieved by computational methods, a multitude of additional external information has been used to guide gene finding algorithms. Most notably is the alignment of EST sequences to the genome. These may be derived from the very target genome, or from other (related) genomes. Also, the alignment may be stringent to allow only the detection of the original source gene (referred to as cis-alignment) or more relaxed to allow the detection of homologous regions (trans-alignment). Additionally, amino acid sequences from the known protein universe are reverse translated into DNA sequences and likewise aligned to the genome.

A different line of research compares genomes of related organisms to exploit the evolutionary forces by which genomes are shaped: Mutations accumulate mainly in non-functional sequences, while functional meaningful sequences are typically highly conserved. Therefore, exons exhibit a higher conservation than introns and intergenic regions. Furthermore, there is a typical pattern in exons observable, namely the third base within a codon (the wobble position) is less conserved than the other two. To exploit information from conservation patternsxk for gene finding, the genome of one or more related organisms must be sequenced and multiple genome alignments have to be obtained. It is also crucial, that the organisms have a suitable evolutionary distance: If they are too closely related, their intronic and intergenic regions have little diverged and cannot be separated from the exonic regions. On the

other hand, if the organisms are two distant, then their coding segments are little conserved. Systems that have successfully implemented the use of external data include for example Twinscan [87], N-scan [69], GenomeScan [190], SGP2 [111], Fgenesh+ [127], Augustus [153] or Contrast [70].

For a more detailed review of gene finding systems that incorporate external information see [23, 98].

2.8. Motivation and Introduction to the mGene Approach

My main goal is to build a very strong *ab initio* gene finder as a core to the system that can subsequently be improved with additional features when available. In this sense I built strongly on the expertise gained by earlier systems, however, the design of conventional gene finders described in previous sections has several limitations: First of all, the training of the meta-model, i.e. the HMM, is performed in a generative way, modeling the joint probability of input and output sequences $p(\mathbf{x}, \mathbf{y})$. The obtained parameter set $\boldsymbol{\theta}$ might, however, be sub-optimal for the prediction task, where the conditional probability $p(\mathbf{y}|\mathbf{x})$ is computed. Additionally, the set of possible algorithms employed for the sub-models of signal detection, is restricted to such techniques that compute probability values as outputs. However, the prediction of genomic signals is itself an active field of research and some of the most successful systems employ SVMs, which do not output probability values [31, 59, 146, 148]. Strictly speaking, SVMs for signal detection can therefore not be employed in combination with a HMM meta-model and less accurate sensors have to be used instead. Another consequence of the generative approach is that adjacent features are not allowed to have overlapping sensor windows, otherwise the resulting probabilities would not be properly normalized anymore and — as certain nucleotide sequences would be counted multiple times — this would lead to an unintended bias towards genes with smaller numbers of exons. For gene finding, this is a strong limitation, as binding motifs for certain factors may be hundreds of nucleotides away from the actual signal. Disregarding these motifs for signal detection will likely lead to a degradation of the prediction performance for these signals. On the other hand, the sequence between the motif and the signal should also be analyzed with regard of its content. Again, this conflict can be avoided by discriminative methods, where the distribution of the input sequences is not explicitly modeled. A last problem of the HMM framework is posed by the possibly different confidence in different sub-models: For example, there might be two candidate signals close to each other, where one has a significant higher probability. The relative short sequence in between, might, however, lead to a high content contribution enforcing the use of the

signal position with the lower probability. To solve this conflict, several heuristics have been proposed, like the local optimality criterion [21, 113]. However, in the probabilistic framework the proper weighting between the different sub-models is not learnt during training.

To overcome these problems we propose to use a discriminative method for structure prediction as meta-model: So-called hidden Markov support vector machines (HM-SVMs) optimize the parameter set $\boldsymbol{\theta}$ through an objective function that is more suited for the prediction task. They also have been shown to outperform generative HMMs on sequence labeling tasks in natural language processing as well as genome annotation [119, 171]. To correctly model segment lengths, I will employ a semi-Markovian extension similar to gHMMs, as realized in hidden semi-Markov support vector machines (HSM-SVMs). This framework also allows us to use discriminatively trained SVMs with overlapping sequence windows as sub-models. Reaping this additional flexibility, considerable energy was put in the design of accurate detectors that extract as much information from the sequence as possible. In particular, recent advances in the implementation of SVMs with string kernels now allow us to train on millions of examples with large sequence windows. Eventually, the reconciliation of individual sensors is realized in the meta-model by adjusting the individual contributions to a cumulative score during training.

3. Discriminative Machine Learning for Genome Analysis

This chapter gives a basic overview over the machine learning techniques applied in this thesis. It starts with some general concepts of machine learning and continues with a basic introduction of support vector machines with string kernels. Eventually, HM-SVMs are described as tools for label sequence prediction, and their extension to HSM-SVMs are discussed.

3.1. General Concepts

Machine learning algorithms have been developed to recognize complex patterns in data thus being able to make adequate decisions or predictions on unseen instances. To this end, a function is learnt that assigns an output to each input object. For our purposes we will mainly focus on supervised learning techniques, where N labeled training examples, i.e. pairs of input objects, $\mathbf{x}^n \in \mathcal{X}$, with their respective outputs, $\mathbf{y}^n \in \mathcal{Y}$, are available, i.e. $(\mathbf{x}^n, \mathbf{y}^n)_{n=1...N}$, and exploited to learn the decision rule. The training examples are drawn *independent and identical*, i.i.d., from an unknown distribution $\mathcal{D}(\mathcal{X}, \mathcal{Y})$. Under the assumption that novel, previously unseen examples are generated from the same distribution, a well-trained classifier will assign a value $f(\mathbf{x})$ to a new instance, which allows the inference of the true, (unknown) label $\mathbf{y}$. The intention is therefore to find a function that minimizes the expected error or risk:

$$R[f] = \int \mathcal{L}(\mathbf{x}, \mathbf{y}, f(\mathbf{x}))d\mathcal{D}, \tag{3.1}$$

where $\mathcal{L}(\mathbf{x}, \mathbf{y}, f(\mathbf{x}))$ denotes a suitable loss, which is a non-negative function of the input pattern and the predicted and the true label. The problem is, however, that the risk functional depends on the unknown distribution $\mathcal{D}$. Therefore, R has to be approximated from the finite number of known training instances, for example by the empirical risk:

$$R_{emp}[f] = \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}(\mathbf{x}^n, \mathbf{y}^n, f(\mathbf{x}^n)). \tag{3.2}$$

However, counting exclusively on the training examples can lead to poor generalization properties, in particular when the given instances are few and noisy. To avoid this problem of over-fitting, the possible solutions are therefore usually restricted to an admissible set of functions, $f \in \mathcal{F}$, and a regularization term $\Omega(f)$ is added to the original objective. In general, the type of algorithms that are considered here, will therefore be of the form:

$$\min_{f} \quad \Omega(f) + CR_{emp}[f], \tag{3.3}$$

where $C > 0$ is a hyper-parameter that specifies the trade-off between minimization of the training error and enforcing smooth or simple solutions with small $\Omega(f)$.

The class of the preferred algorithm depends on the types of input and output data. The inputs are most often (high dimensional) real valued vectors. In the context of gene finding, however, we will mainly encounter sequences as inputs. The outputs, on the other hand, are in the simplest case integers denoting one of several different classes. Here, we will in particular treat the binary classification task, which will be solved using support vector machines (SVMs). This technique employs kernel functions as similarity measures. To account for the special input data, namely DNA sequences, we will use specialized string kernels. SVMs will are employed in Chapter 4 to design genomic signal and content detectors.

The second type of algorithms addresses problems involving complex outputs, such as multiple dependent output variables or structured output spaces. Again we will concentrate on a special case, namely label sequences. To solve this task, we will describe a combination of SVMs and gHMMs, termed hidden semi-Markov support vector machines (HSM-SVM). This algorithm will be used in Chapter 5 to predict a segmentation of a DNA sequence, i.e. for the localisation of genes and prediction of their structure.

For both cases, binary classification and label sequence learning, we use a discriminative training framework that is based on a soft margin criterion.

3.2. Binary Classification with Support Vector Machines

Large-Margin Classification

In the case of binary classification all outputs adopt one of two possible values, $y \in \{-1, +1\}$, which denote the class of the example. Linear classifiers, such as SVMs, then learn a linear discriminant function $f : \mathbf{x} \to \mathbb{R}$, which is parametrized

with a weight vector $\mathbf{w}$ and a bias term $b \in \mathbb{R}$:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \ . \tag{3.4}$$

In this case, the points satisfying $\mathbf{w} \cdot \mathbf{x} + b = 0$ correspond to a *separating hyperplane*, which partitions the input space into two half spaces as shown in Figure 3.1. The sign of $f(\mathbf{x})$ indicates on which side of the hyperplane a point $\mathbf{x}$ is located and it can therefore be used for classification:

$$y^* = \text{sign}\left(f(\mathbf{x})\right). \tag{3.5}$$

The absolute value of $f(\mathbf{x})$ corresponds to the distance of the example to the hyperplane and it can be interpreted as the confidence of a prediction. For datasets that are linearly separable, there exist many separating hyperplanes. Statistical learning theory suggests that classifiers work best on unseen test examples, if the hyperplane separates the data by a large margin, i.e. if the minimal distance of all training examples to the hyperplane is large. An intuitive motivation of this choice is that the classifier is more robust against noise in the training data as well as perturbation of the parameter vector $\mathbf{w}$ [136].

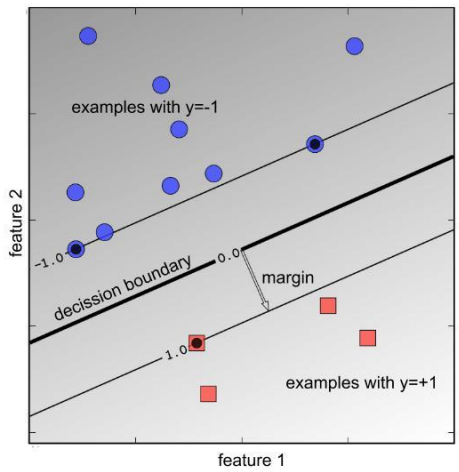


Figure 3.1.: Binary classification with a large margin classifier. The x and y axes correspond to two features of the input examples $\mathbf{x} \in \mathbb{R}^2$. Blue circles correspond to examples with label $y = +1$ and red squares to examples with $y = -1$, respectively. The gray-scale level represents the value of the discriminant function $f(\mathbf{x})$: dark for low values and a light shade for high values. The space is divided into two half spaces by the decision boundary with $f(\mathbf{x}) = 0$, such that the two classes are separated from each other depending on the sign of $f(\mathbf{x})$. The examples highlighted with black centers are closest to the decision boundary, they are called support vectors and define the margin. Figure modified from [13]

Regularizer

It is easy to show, that maximizing the margin can be achieved by minimizing the norm of the weight vector $\|\mathbf{w}\|$ or $\|\mathbf{w}\|^2$, while rescaling the margin to $1/\|\mathbf{w}\|$ [19, 136, 177]. In SVMs, this corresponds to a regularizing term,

$$\Omega[f] = \frac{1}{2}\|\mathbf{w}\|^2 \ , \tag{3.6}$$

which is used to limit the complexity of the function space $\mathcal{F}$ assuring good generalization properties.

Loss function

To allow for the misclassification of some outliers, a so-called soft margin, or Hinge loss [14, 37] is implemented:

$$\mathcal{L}(y, f(\mathbf{x})) = \max(1 - yf(\mathbf{x}), 0) \ . \tag{3.7}$$

With this loss, so called margin errors, i.e. examples that are either classified incorrectly or that lie to close to the decision boundary, are penalized with a linear penalty, while no penalty occurs if the patterns are classified correctly with a large confidence, in particular if $yf(\mathbf{x}) \geq 1$. The resulting optimization problem (Equation 3.3) is non-differentiable. Hence an equivalent problem with inequality constraints is formulated that uses a differentiable objective.

Optimization Problem

The resulting problem can be constructed as a quadratic programming problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi^n} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi^n \\ \text{s.t.} \quad & y^n\left(\mathbf{w}\cdot\mathbf{x}^n + b\right) \geq 1 - \xi^n \\ & \xi^n \geq 0 \quad \forall\ n = 1, \ldots, N \end{aligned} \tag{3.8}$$

where the constraints in principle ensure that training examples are classified correctly. However, these inequality constraints are relaxed by the introduction of so-called slack variables ξ^n that allow an example to be in the margin or misclassified. The margin errors are penalized by the second term in the objective to avoid the excess use of slack variables. C is a hyper-parameter that determines the trade-off between the regularizer (i.e. margin maximization) and the amount of slacks (i.e.

training error minimization). Finding a large margin hyperplane is therefore equivalent to minimizing an upper bound on the regularized empirical risk. Standard tools for convex optimization, can now be used to solve the optimization problem [20]. In practice, the problem is mostly solved by maximizing the Lagrangian dual problem, which requires the computation of dot products between input data points.

Representer Theorem

For many complex problems, a linear classifier is incapable of separating the data well. To allow for more general decision rules, the input data can be mapped into a (high dimensional) feature space by a non-linear transformation $\Phi : \mathcal{X} \to \mathcal{H}$. If the mapping is chosen wisely, the data becomes linearly separable in the new feature space and can be classified accordingly. It is only required that the feature space is equipped with a dot product. The expansive calculations of the dot products $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}^n) \rangle$, can be reduced significantly by using a positive definite kernel $\mathcal{K}(\mathbf{x}, \mathbf{x}^n) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}^n) \rangle$. The generated classification function can be written as:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{n=1}^{N} y^n \alpha^n \mathcal{K}(\mathbf{x}^n, \mathbf{x}) + b\right), \tag{3.9}$$

where the α_i's are Lagrange multipliers and b is the usual bias, which are the results of SVM training [84, 176].

String Kernels

In our case, the kernels compare pairs of sequences in terms of their matching substrings. Therefore, $\mathbf{x} \in \mathcal{X}$ will again denote a sequence of symbols $x_i \in \Sigma$. We use three different types of string kernels, namely the *spectrum kernel* [93], the *weighted degree kernel* [121] and the *weighted degree kernel with shift* [120].

To train signal and content sensors and to predict with them, we use an efficient implementation of all three kernels [147]. It is publicly available in the Shogun machine learning toolbox (cf. `http://www.shogun-toolbox.org`) and allows for large scale applications.

Spectrum kernel

The spectrum kernel [93] of order k counts all matching k-mers, irrespective of their position within the two compared sequences $\mathbf{x}$ and $\mathbf{x}'$. The corresponding feature map $\Phi_k^{Spec} : \mathcal{X} \to \mathbb{R}^{|\Sigma|^k}$ is defined by:

$$\Phi_k^{Spec}(\mathbf{x}) := (\phi_{\mathbf{s}}(\mathbf{x}))_{\mathbf{s} \in \mathcal{X}^k}, \tag{3.10}$$

where $\phi_{\mathbf{s}}(\mathbf{x})$ denotes the number of times a sub-string $\mathbf{s} \in \mathcal{X}^k$ occurs in the sequence $\mathbf{x}$. For another variant of the kernel, we can also assign binary values to the map, i.e. 1, if $\mathbf{s}$ does occur in $\mathbf{x}$ and 0 otherwise. The spectrum kernel is then given by:

$$\begin{aligned} K_k^{Spec}(\mathbf{x}, \mathbf{x}') &= \langle \Phi_k^{Spec}(\mathbf{x}), \Phi_k^{Spec}(\mathbf{x}') \rangle \\ &= \sum_{\mathbf{s} \in \mathcal{X}^k} \phi_{\mathbf{s}}(\mathbf{x}) \phi_{\mathbf{s}}(\mathbf{x}') \end{aligned} \tag{3.11}$$

The spectrum kernel captures typical *compositions* of sequences. To detect motifs that only occur at specif positions, the kernel is not suited. SVMs with a Spectrum kernel can therefore be understood as the discriminative counterpart to the Markov chains, as described in Section 2.7. We will also apply a coding-frame specific spectrum kernel, where only sub-sequences occurring in a given phase are considered. Other variants of the spectrum kernel include the weighted spectrum kernel [145]:

$$K_d^{WSpec}(\mathbf{x}, \mathbf{x}') = \sum_{k=1}^{d} \beta_k K_k^{Spec}(\mathbf{x}, \mathbf{x}') \tag{3.12}$$

and the spectrum kernel with mismatches [92]. However, the later is not further considered in this work.

Weighted degree kernel

The weighted degree (WD) kernel considers matching k-mers of all lengths k from 1 up to d, but only if they occur at the same position in both sequences [118]. SVMs with WD kernels thus model precisely localized motifs. The WD kernel is defined as

$$K_d^{WD}(\mathbf{x}, \mathbf{x}') = \sum_{k=1}^{d} \beta_k \sum_{l=1}^{L-k+1} \mathbb{I}(\mathbf{x}_{[l,l+k-1]}, \mathbf{x}'_{[l,l+k-1]}) \,, \tag{3.13}$$

where $\mathbb{I}$ denotes the indicator function. The contributions of matching strings depend on their length, through the weights

$$\beta_k = 2\frac{d-k+1}{d(d+1)}. \tag{3.14}$$

With this implementation longer matches appear to be down weighted, as $\beta_{k+1} < \beta_k$. However, larger sub-strings also contain smaller sub-strings, which all contribute additionally. The WD can be computed very efficiently without even extracting and

enumerating all sub-sequences of the sequences [147]. In contrast to the spectrum kernel, the WD kernel requires a fixed length for all examined sequences.

Weighted degree kernel with shift

The WD kernel with shift (WDS) allows for slightly displaced matches, yet they are down-weighted relative to matches at the exact corresponding positions. The WDS kernel is defined as

$$\mathcal{K}_d^{WDS}(\mathbf{x},\mathbf{x}') = \sum_{k=1}^{d} \beta_k \sum_{l=1}^{L-k+1} \sum_{\substack{s=0 \\ s+l\leq L}}^{S(l)} \delta_s \; \mu_{k,l,s}(\mathbf{x},\mathbf{x}')\,, \tag{3.15}$$

$$\begin{aligned}\mu_{k,l,s}(\mathbf{x},\mathbf{x}') = \mathbb{I}(\mathbf{x}_{[l+s,l+s+k-1]},\mathbf{x}'_{[l,l+k-1]}) \\ +\mathbb{I}(\mathbf{x}_{[l,l+k-1]},\mathbf{x}'_{[l+s,l+s+k-1]})\,,\end{aligned} \tag{3.16}$$

where β_k is as before, $\delta_s = 1/(2(s+1))$ is the weight assigned to shifts (in either direction) of extent s, and $S(l)$ determines the shift range at position l. Here, we choose $S(l) = \sigma|l - l_c|$, where l_c is a fixed position within the sequence, e.g. the splice site. An efficient implementation for this kernel allowing large scale computations is described in [147].

For both the WD and WDS kernel, the following normalization is used

$$\tilde{\mathcal{K}}(\mathbf{x},\mathbf{x}') = \frac{\mathcal{K}(\mathbf{x},\mathbf{x}')}{\sqrt{\mathcal{K}(\mathbf{x},\mathbf{x})\mathcal{K}(\mathbf{x}',\mathbf{x}')}}\,. \tag{3.17}$$

3.3. Max-Margin based Label Sequence Learning

The methods described above are powerfull tools to discriminate instances of different independent classes, but they are not capable of exploiting structure in the output variables like dependencies between neighboring labels. I will therefore now turn to the HM-SVM approach, which is an generalization of multi-class Support Vector Machines [38, 39, 182] to the broader class of learning structured response. It is conceptually similar to HMMs, as they are both based on a state model with appropriate state transitions. Also similar to HMMs, it assumes a Markov chain dependency structure that leads to an efficient dynamic programming architecture. However, HM-SVMs overcome some of the main short-comings of the HMM. In particular, they are trained in a discriminative manner. Therefore, discriminant functions that exploit the structure in the output space are introduced. In this respect, HM-SVMs follow the approach introduced by Collins [35, 36]. These methods

permit the construction of overlapping features, such that labels can depend directly on features of past or future observations. However, Collins applies a perceptron based algorithm for parameter estimation, while in the HM-SVM framework, the maximum margin principle adopted from SVMs is used. This allows to apply more appropriate objective functions that are more relevant for performance measures employed in practice. The method is described in more detail in [6, 165, 172]. It has been successfully used in a wide range of applications, like natural language processing (e.g. [105, 142, 165]) or computer vision (e.g. [91, 164]). In bioinformatics, it was for instance used for spliced sequence alignment [137], protein side chain [192] and secondary structure prediction [60]. We will also discuss the semi-Markovian extension to HM-SVMs leading to hidden semi-Markov support vector machines (HSM-SVM), which were first described by [122] and employed for genomic sequence annotation [119].

3.3.1. Hidden Markov Support Vector Machines: HM-SVMs

In the HM-SVM framework we learn a function $G : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, parametrized by $\boldsymbol{\theta}$, which assigns a real-valued score to any pair of input sequence $\mathbf{x}$ and label sequence $\mathbf{y}$. $G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ can thus be thought of as a compatibility measure between $\mathbf{x}$ and $\mathbf{y}$. In the HM-SVM framework, the class of allowed discriminant functions will be restricted to the ones that are linear in some joint representation of $(\mathbf{x}, \mathbf{y})$ [6], the scoring function is thus of the form

$$G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \;=\; \sum_k \theta_k \Phi_k(\mathbf{x}, \mathbf{y}) = \boldsymbol{\theta}^\top \Phi(\mathbf{x}, \mathbf{y}) \;, \tag{3.18}$$

where k is the index into the dimensions of $\boldsymbol{\theta}$. Similar to the SVM case, the construction of an appropriate feature map Φ is therefore crucial for this approach. However, while in the binary classification scenario the features are extracted from the input patterns only, they are now derived from input-output pairs. With the joint feature map it is thus possible to accommodate the fact that the compatibility between $\mathbf{x}$ and $\mathbf{y}$ may depend on properties of $\mathbf{x}$ in association with certain characteristics of $\mathbf{y}$, and, vice versa, certain features that describe the internal structure $\mathbf{y}$ can interact with particular characteristics of the input patterns $\mathbf{x}$ [6].

Prediction with Viterbi Decoding

For a given set of parameters $\boldsymbol{\theta}$, the prediction for a sequence $\mathbf{x}$ is given by the segmentation $\mathbf{y}^*$ with the highest score of all segmentations,

$$\mathbf{y}^* = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\, G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \ . \tag{3.19}$$

The space of potential output sequences $\mathcal{Y}$ grows exponentially with the length of the sequences. It is thus crucial to effectively solve Equation 3.19. In particular, efficient solutions depend on the decomposability of the scoring function over components of $\mathbf{y}$:

$$G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{L+1} g_{\boldsymbol{\theta}}(y_{i-1}, y_i, \mathbf{x}, i) \ . \tag{3.20}$$

In this case dynamic programming based decoding algorithms [64], like the Viterbi algorithm [51] described in Section 2.7 can be used to solve Equation 3.19. It requires the computation of the Viterbi Matrix according to Equation 2.4, as well as the traceback matrix (Equation 2.5).

Feature Maps

With the restriction of equation 3.20, the design of the feature map is also constrained such that each feature $\Phi_k(\mathbf{x}, \mathbf{y})$ has to be composed of mapping functions that are again positional decomposable:

$$\Phi_k(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{L+1} \phi_k(y_{i-1}, y_i, \mathbf{x}, i) \ . \tag{3.21}$$

If this restriction holds, the types of features employed only depend on the application. For example, inspired by HMMs (compare equation 2.6), one could think of features $\phi_t(y_{i-1}, y_i)$ that capture interactions between neighboring labels (analogue to transition probabilities $p_t(y_i|y_{i-1})$) or features $\phi_s(\mathbf{x}, y_i, i)$ that model interactions between the observation and a specific label (analogue to emission probabilities $p_t(x_i|y_i)$, however allowing more general dependencies for on overlapping input sequence windows)[6].

The Training Procedure: Parameter Estimation

The parameters $\boldsymbol{\theta}$ of the scoring function are learnt on a set of labeled training examples $\left\{(\mathbf{x}^n, \mathbf{y}^n)\right\}, n = 1, \ldots, N$, such that for each instance the true labeling $\mathbf{y}^n$ scores higher than all other possible labelings $\mathbf{y} \in \mathcal{Y}^n$, i.e. $G_{\boldsymbol{\theta}}(\mathbf{x}^n, \mathbf{y}^n) \gg G_{\boldsymbol{\theta}}(\mathbf{x}^n, \mathbf{y})$. Analogously to the SVM architecture we do not only want the correct labeling to

score maximal, but additionally require that it is separated from all other labelings by a *large margin* [175]. Therefore, the optimal parametrization $\boldsymbol{\theta}$ is learnt by maximizing the minimal margin. Like in the standard form of SVMs, the problem can be formulated as a quadratic program, when fixing the functional margin ($G_{\boldsymbol{\theta}}(\mathbf{x}^n, \mathbf{y}^n) - G_{\boldsymbol{\theta}}(\mathbf{x}^n, \mathbf{y}) \geq 1$) and minimizing the norm of the parameter vector $\|\boldsymbol{\theta}\|^2$ [4, 136]. Additionally, margin violation can be accommodated by the introduction of slack variables ξ^n for every training sequence, thus penalizing the largest margin violation for every example [171].[1] This leads to the following soft-margin optimization problem [4]:

$$\min_{\boldsymbol{\xi}\in\mathbb{R}^N, \boldsymbol{\theta}} \quad \frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\sum_{n=1}^{N}\xi^n \tag{3.22}$$

$$\text{s.t.} \quad G_{\boldsymbol{\theta}}(\mathbf{x}^n, \mathbf{y}^n) - G_{\boldsymbol{\theta}}(\mathbf{x}^n, \mathbf{y}) \geq 1 - \xi^n \tag{3.23}$$

$$\xi^n \geq 0 \quad \forall\, n = 1, \ldots, N,\ \mathbf{y} \in \mathcal{Y}^n,$$

where C is a hyper-parameter that controls the trade-off between training error minimization and margin maximization and $\mathcal{Y}^n$ is the set of all possible labelings for example n.

Solving the QP via Column Generation

The number of constraints in Equation 3.22 can be extremely large — they grow exponentially with the length of the sequence — which poses a major computational challenge. Therefore, the optimization problem can not be solved directly. In practice, however, most constraints are inactive. Hence, working set methods can be applied to solve the problem [39, 172]. The idea is to start with an empty *working set* of constraints and an arbitrary parametrization $\tilde{\boldsymbol{\theta}}$. Subsequently, new constraints corresponding to those paths that maximally violate the margin constraints Equation 3.22 are identified:

$$\tilde{\mathbf{y}}^{n*} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^n}(1 - \Delta G_{\tilde{\boldsymbol{\theta}}}(\mathbf{y}^n, \mathbf{y})) \tag{3.24}$$

$$\tilde{\mathbf{y}}^{n*} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^n} G_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}^n, \mathbf{y}), \tag{3.25}$$

where $\Delta G_{\boldsymbol{\theta}}(\mathbf{y}^n, \mathbf{y}) = G_{\boldsymbol{\theta}}(\mathbf{x}^n, \mathbf{y}^n) - G_{\boldsymbol{\theta}}(\mathbf{x}^n, \mathbf{y})$. For this task, the same inference algorithm that is used for prediction can be employed (compare Equation 3.19). We

[1] In contrast to this strategy, one could also introduce one slack variable for every violater, i.e. $\xi^{n,y}$, thus providing an upper bound for the rank loss [4]. However, this results in a substantial larger amount of active constraints and is therefore not considered here.

therefore use the Viterbi algorithm to compute the path that scores highest under the given parametrization. If it returns the true path or a wrong labelling that does not violate the margin, no constraint is added; otherwise the generated constraint is added to the working set and the resulting intermediate optimization problem is solved. Alternation between constraint generation and solving the optimization problem is repeated until no more margin violators can be found. The algorithm for the method described above is given in Table 3.1. It is also known as column generation method or cutting-plane algorithm and can be shown to converge to the optimal solution [75, 117, 172].

1	**Input:** N labelled training examples, $(\mathbf{x}^n, \mathbf{y}^n, \forall n)$
2	
3	**Initialization:**
4	start with an arbitrary parametrization $\tilde{\boldsymbol{\theta}}, \tilde{\xi}^n$
5	and an empty working set, i.e. $\tilde{\mathcal{Y}}^n = \emptyset, \forall n$
6	
7	**repeat**
8	**for** each training example $(\mathbf{x}^n, \mathbf{y}^n)$
9	
10	use the Viterbi algorithm for decoding:
11	$\mathbf{y}^{n*} = \text{argmax}_{\mathbf{y} \in \mathcal{Y}^n} G_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}^n, \mathbf{y})$
12	
13	add maximal margin violator to the active set:
14	**if** $\mathbf{y}^{n*} \neq \mathbf{y}^n \wedge \Delta G(\mathbf{y}^n, \mathbf{y}^{n*}) \leq 1 - \tilde{\xi}^n$
15	$\tilde{\mathcal{Y}}^n \leftarrow \tilde{\mathcal{Y}}^n \cup \mathbf{y}^{n*}$
16	**end if**
17	
18	**end for**
19	
20	solve the intermediate training problem Equation 3.35
21	with the updated set of constraints $(\mathcal{Y}^n = \tilde{\mathcal{Y}}^n, \forall n)$
22	$\tilde{\boldsymbol{\theta}}, \tilde{\xi}^n \leftarrow solve\_QP(\mathbf{x}^n, \mathbf{y}^n, \tilde{\mathcal{Y}}^n, \forall n)$
23	
24	**until** no more margin violators found

Table 3.1.: The HM-SVM training algorithm: In each iteration constraint generation (line: 7-18) is alternated with solving of an intermediate optimization problem (line: 20-21).

Loss Function

The optimization problem Equation 3.22 can be re-written without constraints as:

$$\min_{\boldsymbol{\theta}} \quad \Omega(\boldsymbol{\theta}) + C \sum_{n=1}^{N} \mathcal{L}(\mathbf{x}^n, \mathbf{y}^n, G_{\boldsymbol{\theta}}), \tag{3.26}$$

with a (non-linear) soft-margin loss function:

$$\mathcal{L}(\mathbf{x}^n, \mathbf{y}^n, G_{\boldsymbol{\theta}}) \quad = \quad \max(0, \max_{\mathbf{y} \in \mathcal{Y}^n} (1 - \Delta G_{\boldsymbol{\theta}}(\mathbf{y}^n, \mathbf{y})). \tag{3.27}$$

Note, that the incurred loss is in fact identical for all possible labelings associated with a given input $\mathbf{x}^n$ [2]. It only depends on the maximal score difference and is *independent* of the similarity of a path $\mathbf{y}$ to the true labeling $\mathbf{y}^n$. This is, however, not well suited for most structured output scenarios [5]. For example, a segmentation that differs from the true segmentation only in a few segments, should be treated differently than one that is completely incorrect. In general, the choice of the loss function should reflect the quantity that measures how well the system performs. To accomplish this, two approaches have been proposed [172]: In the first one, the slack variables are re-scaled according to a loss term $\ell(\mathbf{y}^n, \mathbf{y})$ that encodes a dissimilarity measure for a pair of label sequences. Slack-rescaling leads to the following constraints:

$$\text{Slack-rescaling:} \quad G_{\boldsymbol{\theta}}(\mathbf{x}^n, \mathbf{y}^n) - G_{\boldsymbol{\theta}}(\mathbf{x}^n, \mathbf{y}) \geq 1 - \frac{\xi^n}{\ell(\mathbf{y}^n, \mathbf{y})} \quad \forall\, \mathbf{y} \in \mathcal{Y}^n, \tag{3.28}$$

which corresponds to the following loss function in Equation 3.26:

$$\mathcal{L}(\mathbf{x}^n, \mathbf{y}^n, G_{\boldsymbol{\theta}}) = \max(0, \max_{\mathbf{y} \in \mathcal{Y}^n} (\ell(\mathbf{y}^n, \mathbf{y})(1 - \Delta G_{\boldsymbol{\theta}}(\mathbf{y}^n, \mathbf{y})))), \tag{3.29}$$

Therefore, margin violating paths with high loss terms are penalized stronger than those with a small loss. An additional advantage of this approach is, that the scaling of the loss term relative to the scoring function is — to some extend — controllable by tuning the trade-off parameter C, e.g. if the loss term is rescaled by some constant, i.e. $\ell' = c \cdot \ell$, the resulting optimization problem leads to the same optimal parameter set $\boldsymbol{\theta}^*$, if $C' = C/c$ is set in the objective Equation 3.26 [172].

[2] This is a consequence of the employed strategy to use a single slack variable per training example.

A drawback of the slack-rescaling approach is, that solving the resulting optimization problem with the column generation technique poses a severe problem. To generate new constraints, the labeling $\tilde{\mathbf{y}}^{n*}$ has to be inferred (compare Equation 3.24 and Table 3.1), which maximizes the product of loss term and score:

$$\begin{aligned} \tilde{\mathbf{y}}^{n*} &= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^n} \ell(\mathbf{y}^n,\mathbf{y})(1-\Delta G_{\tilde{\boldsymbol{\theta}}}(\mathbf{y}^n,\mathbf{y})) && (3.30)\\ &= \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^n} \ell(\mathbf{y}^n,\mathbf{y}) G_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}^n,\mathbf{y}). && (3.31)\end{aligned}$$

This product is not positionally decomposable — even if the loss term and the score function are — and therefore the Viterbi decoding method cannot be used

An alternative is the margin-rescaling approach [39, 165, 172], which tries to ensure that the separation of the correct segmentation from incorrect ones is equal to the error of the prediction. The constraints take the following form:

$$\text{Margin-rescaling:} \qquad G_{\boldsymbol{\theta}}(\mathbf{x}^n,\mathbf{y}^n) - G_{\boldsymbol{\theta}}(\mathbf{x}^n,\mathbf{y}) \geq \ell(\mathbf{y}^n,\mathbf{y}) - \xi^n \quad \forall\ \mathbf{y}\in\mathcal{Y}^n \tag{3.32}$$

Here, the corresponding loss function is given by:

$$\mathcal{L}(\mathbf{x}^n,\mathbf{y}^n,G_{\boldsymbol{\theta}}) = \max(0, \max_{\mathbf{y}\in\mathcal{Y}}(\ell(\mathbf{y}^n,\mathbf{y}) - \Delta G_{\boldsymbol{\theta}}(\mathbf{y}^n,\mathbf{y})))) \tag{3.33}$$

This method therefore requires a loss term $\ell(\mathbf{y}^n,\mathbf{y})$ that is linearly comparable to the scoring function — and thus demands a more precise calibrating between the feature mapping and the loss function than the slack re-scaling approach [134].

The inference problem, which is solved for constraint generation in the column generation algorithm is now of the form:

$$\tilde{\mathbf{y}}^{n*} = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}^n}(\ell(\mathbf{y}^n,\mathbf{y}) + G_{\tilde{\boldsymbol{\theta}}}(\mathbf{x},\mathbf{y}))\ , \tag{3.34}$$

which is positionally decomposable if the loss term and the score functions are positionally decomposable. Therefore, the Viterbi algorithm can now be applied. However, a problem arises from the fact that outputs with large errors and low score (i.e. correctly separated from the true labelling) are revalued by the sum and potentially added to the active set at the expense of instances that are not even separated. The margin constraints can therefore be dominated by incorrect segmentations that are very unlikely. For these reasons, it is believed to be less accurate than the slack-rescaling approach. Due to the more convenient inference problem,

the margin re-scaling approach will nevertheless be used in this work. The resulting optimization problem is given by:

$$\begin{aligned} \min_{\boldsymbol{\xi}\in\mathbb{R}^N,\boldsymbol{\theta}} \quad & \Omega(\boldsymbol{\theta}) + C\sum_{n=1}^{N}\xi^n \\ \text{s.t.} \quad & G_{\boldsymbol{\theta}}(\mathbf{x}^n,\mathbf{y}^n) - G_{\boldsymbol{\theta}}(\mathbf{x}^n,\mathbf{y}) \geq \ell(\mathbf{y}^n,\mathbf{y}) - \xi^n \\ & \xi^n \geq 0 \quad \forall\, n=1,\ldots,N,\ \mathbf{y}\in\mathcal{Y}^n, \end{aligned} \tag{3.35}$$

where we have additionally introduced a more general regularizer $\Omega(\boldsymbol{\theta})$.

3.3.2. Two-Stage Learning

In [5] Altun et al. suggest to leverage the linearity of G in order to use Kernel functions $\mathcal{K}((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')) = \langle\Phi(\mathbf{x},\mathbf{y}),\Phi(\mathbf{x}',\mathbf{y}')\rangle$. By applying the representer theorem the explicit computation of the mapping Φ can be avoided while learning non-linear discriminant functions [9, 84]. For $\Omega(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|^2$ the scoring function can be re-written as the expansion [5]:

$$G_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}) \;=\; \sum_{n=1}^{N}\sum_{\mathbf{y}'\in\mathcal{Y}} \alpha^n(\mathbf{y}')z(\mathbf{y}')\mathcal{K}((\mathbf{x}^n,\mathbf{y}'),(\mathbf{x},\mathbf{y}))\ , \tag{3.36}$$

where $\alpha^n(\mathbf{y}) \geq 0$ are the Lagrange multipliers for the constraint on example n and labelling $\mathbf{y}$. With the introduction of the functions $z(\mathbf{y}^n) = 1$ and $z(\mathbf{y}\neq\mathbf{y}^n) = -1$, the problem is in principle transformed to a binary classification problem, where $(\mathbf{x}^n,\mathbf{y}^n)$ are treated as positive examples and $(\mathbf{x}^n,\mathbf{y}), \mathbf{y}\neq\mathbf{y}^n$ as negative pseudo-examples. In [5], Equation 3.36 is subsequently solved in the dual.

However, one of the largest problems with HM-SVMs is their high computational complexity, thus solving the resulting optimization problems may become computationally infeasible already for a few hundred examples. To tackle significantly larger label sequence problems (with several thousands of sequences) we use a very effective strategy of partitioning the problem into independent sub-problems as suggested in [119]. To do so, we express the kernel $\mathcal{K}((\mathbf{x}^n,\mathbf{y}^n),(\mathbf{x},\mathbf{y}))$ as a sum of S kernel functions each acting *locally* on individual positions y_i^n, y_i of the segmentations $\mathbf{y}^n,\mathbf{y}$ and only on certain dimensions $k_s = K_s\ldots K_s'$ of the feature vector. We will therefore first use the definition of the Kernel function and subsequently partition the spanned feature space in S subspaces:

$$\begin{aligned}\mathcal{K}((\mathbf{x}^n,\mathbf{y}^n),(\mathbf{x},\mathbf{y})) &= \langle\Phi(\mathbf{x}^n,\mathbf{y}^n),\Phi(\mathbf{x},\mathbf{y})\rangle \\ &= \sum_{k=1}^{K}\Phi_k(\mathbf{x}^n,\mathbf{y}^n)\Phi_k(\mathbf{x},\mathbf{y}) \\ &= \sum_{s=1}^{S}\sum_{k_s=K_s}^{K'_s}\Phi_{k_s}(\mathbf{x}^n,\mathbf{y}^n)\Phi_{k_s}(\mathbf{x},\mathbf{y}) \ .\end{aligned} \tag{3.37}$$

Next, the positional decomposability of the features is used Equation 3.21, to reorder the summations:

$$\begin{aligned}\mathcal{K}((\mathbf{x}^n,\mathbf{y}^n),(\mathbf{x},\mathbf{y})) = &\sum_{s=1}^{S}\sum_{k_s=K_s}^{K'_s}\sum_{i=1}^{L+1}\phi_{k_s}(y^n_{i-1},y^n_i,\mathbf{x}^n,i)\sum_{i'=1}^{L'+1}\phi_{k_s}(y_{i'-1},y_{i'},\mathbf{x},i') \\ = &\sum_{s=1}^{S}\sum_{i=1}^{L+1}\sum_{i'=1}^{L'+1}\sum_{k_s=K_s}^{K'_s}\phi_{k_s}(y^n_{i-1},y^n_i,\mathbf{x}^n,i)\phi_{k_s}(y_{i'-1},y_{i'},\mathbf{x},i') \ ,\end{aligned} \tag{3.38}$$

where L is the length of the segmentation $\mathbf{y}^n$, and L' the length of $\mathbf{y}$, respectively. This partitioning may be used to define separate kernels for each label $q \in \mathcal{Q}$ or each label pair $q, q' \in \mathcal{Q}$. In this case, a feature ϕ_s can be defined such that it has only non-zero values at positions that have a corresponding label q_s, and equivalently a feature ϕ_c has only non-zero values at positions that have a corresponding label q_c preceding a label q'_c:

$$\begin{aligned}\phi_s(y_i,\mathbf{x},i) &= \mathbb{I}(y_i,q_s)\varphi_s(\mathbf{x},i) \\ \phi_c(y_{i-1},y_i,\mathbf{x}^n,i) &= \mathbb{I}(y_i,q_c)\mathbb{I}(y_{i-1},q'_c)\varphi_c(\mathbf{x},i) \ .\end{aligned} \tag{3.39}$$

In analogy to the HMM framework, ϕ_s can be interpreted as a signal contribution, stemming from a single unit of the label sequence and ϕ_c can be interpreted as a content contribution, or, setting $\varphi_c(\mathbf{x},i) = 1$, a contribution derived from the transition from q'_c to q_c. The resulting kernel functions then only depend on the input sequence, or certain regions thereof:

$$\mathcal{K}^s((\mathbf{x}^n,i),(\mathbf{x},i')) = \sum_{k_s=K_s}^{K'_s}\varphi_{k_s}(\mathbf{x}^n,i)\varphi_{k_s}(\mathbf{x},i'). \tag{3.40}$$

For a more compact presentation we only consider signal contributions, the derivation for features depending on label pairs is analogously and can be found in the Appendix A. We therefore get:

$$\mathcal{K}((\mathbf{x}^n, \mathbf{y}^n), (\mathbf{x}, \mathbf{y})) \;=\; \sum_{s=1}^{S} \sum_{i=1}^{L+1} \sum_{i'=1}^{L'+1} \mathbb{I}(y_i^n, q_s)\mathbb{I}(y_{i'}, q_s)\mathcal{K}^s((\mathbf{x}^n, i), (\mathbf{x}, i')) \;. \quad (3.41)$$

This kernel allows us to re-write the scoring function of Equation 3.36 in the following, thus making an efficient partitioning of the problem possible:

$$G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \;=\; \sum_{s=1}^{S} \sum_{i'=1}^{L'+1} \mathbb{I}(y_{i'}, q_s)\hat{f}^s(\mathbf{x}, i') \;, \quad \text{with} \quad (3.42)$$

$$\hat{f}^s(\mathbf{x}, i') \;=\; \sum_{n=1}^{N} \sum_{\mathbf{y}' \in \mathcal{Y}} \alpha^n(\mathbf{y}')z(\mathbf{y}') \sum_{i=1}^{L+1} \mathbb{I}(y_i^n, q_s)\mathcal{K}^s((\mathbf{x}^n, i), (\mathbf{x}, i')) \;. \quad (3.43)$$

In these functions we effectively enumerate over all potential sites stemming from any possible segmentation of one of the training examples. We therefore rewrite these functions as single-sum linear combinations: $\hat{f}^s(\chi) = \sum_{n=1}^{\hat{N}_s} z^n \alpha^n \mathcal{K}^s(\chi^n, \chi)$, where a sub-sequence of $\mathbf{x}$ at a given position is denoted with χ. Recall, that pseudo-labels z are used, which take the value $+1$, if the example is taken from a true labeling and -1 otherwise. The functions $\hat{f}^s$ thus take the form of SVM classification functions (compare Equation 3.9). In the HM-SVM framework, all of these functions — which correspond to binary classification tasks for the individual atomic labels $q \in \mathcal{Q}$ — are optimized simultaneously. Additionally, the relative relevance of the different components is adjusted. While this global learning approach is desirable, the combination of these tasks for large problems soon reaches the limits of computational resources. Following the strategy proposed in [119] we therefore formulate independent binary classification tasks and only learn the interaction between these functions in a relatively small optimization problem that can be solved rather efficiently (see Table 3.2). Thus, we define a scoring function where a transformation $t^s_{\boldsymbol{\theta}}$ for each of the pre-computed functions f^s is learnt:

$$G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \;=\; \sum_{s=1}^{S} \sum_{i'=1}^{L'+1} \mathbb{I}(y_{i'}, q_s)t^s_{\boldsymbol{\theta}}(f^s(\mathbf{x}, i')) \;. \quad (3.44)$$

The contributions of the functions to the total score are therefore properly scaled. We will use piece-wise linear functions for these transformations.

```
1   Input: N labelled training examples, (x^n, y^n, ∀n)
2
3   for each signal s
4
5       generate labeled examples for signal s:
6       initialize:
7       sequences of signal training examples: XT^s
8       labels of signal training examples: LT^s
9       for each training example (x^n,y^n)
10          for i = 1 ... L^n
11              χ_i^n = win^s(x^n, i)
12              XT^s ← [XT^s, χ^T]
13              if y_i^n = q_s
14                  LT^s ← [LT^s, +1]
15              else
16                  LT^s ← [LT^s, −1]
17              end if
18          end for
19      end for
20
21      train signal detector:
22      w_s, b_s ← solve_SVM(XT^s, LT^s)
23
24      predict signal for all positions:
25      for each training example (x^n,y^n)
26          for i = 1 ... L^n
27              f^s(x^n, i) = w_s · χ_i^n + b_s
28              FSX_{i,s}^n = f^s(x^n, i)
29          end for
30      end for
31
32  end for
33
34  train HM-SVM (see Table 3.1):
35  θ ← solve_HMSVM(x^n, y^n, FSX^n, ∀n)
```

Table 3.2.: Two-stage learning algorithm for HM-SVM training with signal detectors: In the first stage (line: 3-32) signals are treated independently: For each signal s, labelled training examples (XT^s, LT^s) are generated from the labeled input sequences $(\mathbf{x}^n, y^n)$, where $win^s(\mathbf{x}, i)$ takes a signal specific sub-sequence from $\mathbf{x}$ relative to the position i (line: 5-19). Subsequently SVMs are trained (line: 21-22) and signal predictions are computed for each position i in each training sequence $\mathbf{x}^n$ (line: 24-30). The predictions of all signals for a given example are stored in a matrix $\mathbf{FSX}^n$. These matrices are used in the second stage (line: 34-35) in conjunction with the label sequences $\mathbf{y}^n$ to train a HM-SVM according to Table 3.1.

3.3.3. Feature Mapping with Piece-wise Linear Functions (PLiFs)

For efficient training, it is essential that $G_{\boldsymbol{\theta}}$ is a linear function of a relatively small number of parameters. In the simplest case, the mappings t can be defined as linear functions: $t = v \cdot f$. The HM-SVM would then only learn one single weight per feature and state. However, the function $G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ also needs to be expressive enough to allow separation of correct segmentations from incorrect ones, therefore non-linear dependencies in the input space should be taken into account. To resolve this apparent conflict, we define a proper mapping into a feature space, and use piece-wise linear functions, PLiFs for this mapping. The PLiFs, $t_{\mathbf{u},\mathbf{v}} : \mathbb{R} \to \mathbb{R}$, are defined in the following way:

$$t_{\mathbf{u},\mathbf{v}}(x) \quad = \quad \begin{cases} v_1 & x < u_1 \\ \frac{v_k(u_{k+1}-x)+v_{k+1}(x-u_k)}{u_{k+1}-u_k} & u_k \leq x < u_{k+1} \\ v_K & u_K \leq x \end{cases} , \tag{3.45}$$

with K supporting points $\mathbf{u} = (u_k)_{k=1,\ldots,K}$ (satisfying $u_k < u_{k+1}$) and corresponding function values $v_k = t_{\mathbf{u},\mathbf{v}}(u_k)$. Note, that for given supporting points $\mathbf{u}$, the application of a PLiF to a real-valued input x is equivalent to expanding each single input value into a sparse vector of length K, which is then linearly weighted with a weight vector $\mathbf{v} \in \mathbb{R}^K$:

$$t_{\mathbf{u},\mathbf{v}}(x) \quad = \quad \mathbf{v}^\top \psi_{\mathbf{u}}(x) \ , \tag{3.46}$$

where $\psi_{\mathbf{u}}(x) \in \mathbb{R}^K$ is a histogram representation of x: If the value of x lies between u_k and u_{k+1} its entries are 0 everywhere except at the k-th and $(k+1)$-th components:

$$\psi_k(x) \quad = \quad \frac{u_{k+1} - x}{u_{k+1} - u_k} \quad \text{and} \quad \psi_{k+1}(x) = \frac{x - u_k}{u_{k+1} - u_k} \ . \tag{3.47}$$

If x is smaller than the first supporting point or larger than the last one, than only the first/the last component has value $\psi_1(x) = x$, or $\psi_K(x) = x$, respectively. While t is thus linear – in the values $\mathbf{v}$ as parameters – for given support points $\mathbf{u}$, it effectively models a non-linear transformation of the input x. In Equation 3.44, the different PliFs, $t^s_{\boldsymbol{\theta}}$, are applied to the respective input features f^s to compute the scoring function $G_{\boldsymbol{\theta}}$. In this case, the values of the support points $\mathbf{u}^s$ are determined on the training set and all the values of the weight vectors $\mathbf{v}^s$ will be learned simultaneously during global training (see Section 5.3 for further details).

3.3.4. Semi-Markov Extension to HM-SVMs: HSM-SVMs

Similar to the conventional HMM the HM-SVM suffers from the limitation that each state has to emit exactly one symbol at a time. We will therefore now generalize the HM-SVM setting to allow the machine to persist in one state for more than one time unit and to have a non-Markovian behaviour during this segment of time [122]. This extension is conceptually similar to the gHMM approach in Section 2.7 and allows to model arbitrary segment length distributions. We will now again treat $\mathbf{y} = \mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_M$ as a sequence of M segments, $\mathbf{y}_j = (b_j, b'_j, y_j, y'_j)$, see Section 2.5, rather than a simple sequence of atomic labels. We also assume that the subsequence $\mathbf{x}_{[b_j, b'_j]}$ starting at b_j and ending at b'_j is emitted from state $\mathbf{y}_j$. The scoring function (Equation 3.20) then needs to be generalized such that it can be decomposed into segmental contributions:

$$G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \quad = \quad \sum_{j=1}^{M+1} g'_{\boldsymbol{\theta}}(y_{j-1}, y_j, \mathbf{x}, b_j, b'_j). \tag{3.48}$$

Again, $g'_{\boldsymbol{\theta}}$ only depends on the current and one preceding label, y_j and y_{j-1} respectively, as well as the sequence of the segment. The feature mappings (Equation 3.21) are generalized accordingly to:

$$\Phi_k(\mathbf{x}, \mathbf{y}) \quad = \quad \sum_{j=1}^{M+1} \phi_k(y_{j-1}, y_j, \mathbf{x}, b_j, b'_j). \tag{3.49}$$

With this formulation, features may depend on start and stop positions of segments. This additional flexibility allows to model features that depend not only on individual positions but on complete segments.

It is now also possible to consider loss terms that correspond to complete segments, instead of individual positions. This is desirable, especially if the performance of the system is measured with respect to correctly predicted segments. The loss term thus has to be of the form:

$$\ell(\mathbf{y}, \mathbf{y}') \quad = \quad \sum_{j_1=1}^{M_1+1} \sum_{j_2=1}^{M_2+1} l(\mathbf{y}_{j_1}, \mathbf{y}'_{j_2}) \ . \tag{3.50}$$

A common choice is the Hamming loss, where we have $l(\mathbf{y}^n_{j_1}, \mathbf{y}_{j_2}) = 1$ if the considered segments are identical, $\mathbf{y}^n_{j_1} = \mathbf{y}_{j_2}$ and 0 otherwise.

The advantages of the semi-Markovian extensions also come with a drawback:

Solving the inference problem (3.19) for prediction, now requires an extension of the Viterbi decoding algorithm, which demands the computation of the Viterbi matrix according to Equation 2.11 and traceback matrices Equation 2.12. As discussed in Section 2.7, this has a substantial higher run time. The inference problem — with the additional loss term (see equation Equation 3.34) also has to be solved for constraint generation, and therefore presents a major bottle-neck for efficient training. Our HSM-SVM approach for gene finding is presented in more detail in Section 5.

4. Genomic Feature Recognition

In the following sections, I will describe the SVM-based sensors to detect genomic features on the DNA sequence. I will distinguish signal sensors at segment boundaries, with strongly localized motifs, and content sensors, that capture the sequence composition of a given segment (see Figure 4.1). In this work, a common framework was designed for all signal detectors. This is described in the beginning, followed by signal specific differences and details. Aq similar framework was also deduced for all content detectors. Furthermore, the incorporation of additional features, like conservation scores or EST and protein alignments will be discussed as extensions to the ab-inito scenario.

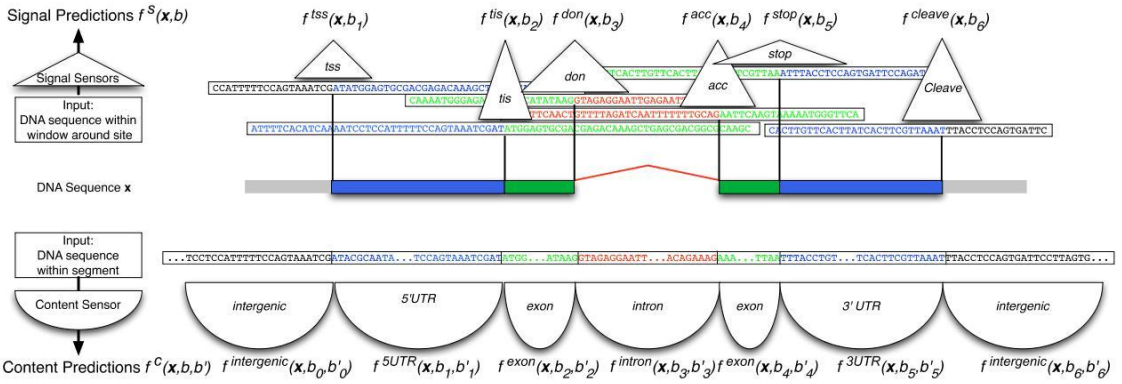


Figure 4.1.: In a first step the genomic sequence is scanned using SVM-based sensors trained to recognize DNA signals, such as transcription start sites (*tss*), translation initiation sites (*tis*), acceptor (*acc*) and donor (*don*) splice sites, stop codons (*stop*) and cleavage sites (*cleave*). Input to the signal detectors are DNA sequences within a pre-defined signal-specific window around the candidate site. These windows are allowed to overlap. Each signal sensor assigns a score $f^s(\mathbf{x}, b) \in \mathbb{R}$ to each candidate site b (depicted are only true sites). Additionally, several content sensors are employed, including *intergenic*, *exon*, *intron*, and *UTR* content sensors, that analyze the composition of complete segments $[b, b']$ and assign scores $f^c(\mathbf{x}, b, b') \in \mathbb{R}$ to them (again, only true segments are drawn). The output of these sensors will serve as input to the gene structure prediction tool (see Chapter 5).

4.1. A common framework for Genomic Signal Sensors

In general, it is assume that each genomic signal that we are interested in, resides at a segment boundary and that it is induced by certain motifs in its — more or less close — vicinity on the DNA sequence (see Section 2.1). Furthermore, most signals are characterized by a compulsory variant of a consensus sequence, which is necessary but not sufficient to define the signal and therefore defines a candidate set of positions. For example, true translation initiation sites occur at the boundaries between UTRs and coding sequences and carry the compulsory consensus sequence `ATG` at the first three positions of the coding sequence. This consensus three-mer, however, is not sufficient to define a *tis*. In fact, assuming a random sequence with uniform nucleotide distribution, the string `ATG` occurs with a probability of $1/4^3$, leading to over 1.5 million candidate sites in a 100 Mbp genome (like the one of *C. elegans*). In contrast, there are only around 20.000 true *tis* sites in the nematode genome.

In principle, each signal sensor is therefore designed as binary classification task, distinguishing between true signal sites and decoy sites, where training and prediction can be restricted to the candidate sites. Only in the case of transcription start and cleavage sites, there is no compulsory consensus sequence and therefore any position in the sequence has to be included in the set of candidates.

The example above also gives an estimate of the dimension of the posed problems, where one encounters millions of examples. The task is additionally complicated as the problems are extremely unbalanced with only less than 1% positive candidates. Besides high classification accuracy a major criterion will be that the detectors have to work efficiently on millions of instances. Therefore, SVMs with specific string kernels are used.

In summary, the generic framework for signal detection includes the generation of labelled training examples, the SVM-based binary classifiers, a cross-validation scheme to generate unbiased predictions for all candidate positions, and a final post-processing step. These parts will be described in more detail in the following sections.

4.1.1. Example extraction

For good generalization properties of supervised learning methods, it is important to train on a good, i.e. representative, training set. In gene finding, this is a particular involved process. Special care must be taken to exclude positive examples that are themselves derived from predictions and, when collecting negative examples, potential unknown positives must be avoided. (The consequences of training on a

predicted set will be further elucidated in Section 6.3.) On the other hand, using data purely derived from experimental evidence is also no guarantee for an unbiased annotation: For example, in EST sequences highly expressed genes are typically over-represented. There are different sources of varying quality and the two stage-learning method allows us to use and integrate data of various origins to generate labels for the different signals.

The first data source are official genome annotations, which are usually available in generic feature format (GFF) files[1], or files of similar formats (see Section 5.5 for an excerpt of an example GFF file). Among other features, they usually contain experimentally confirmed as well as predicted gene segment annotations. These official annotations are usually the result of intense community efforts that include the integration, reconciliation, and manual curation of divers sources. Therefore, depending on the responsible institution and proper filtering, these files can be used to generate high quality labeled examples for the different types of signal features. However, in some cases the origin and processing of contained information remains obscure — in particular whether it is predicted or confirmed.

Therefore, annotations can also be generated solely by EST or cDNA alignments (see Section 2.2 and [122, 148]). To align ESTs and cDNA sequences against the genomic DNA, `blat` is used, which is able to handle introns [82], i.e. large gaps in the alignments. If the sequence cannot be unambiguously matched, only the best hit is considered. The alignments are refined by correcting typical sequencing errors, for instance by removing minor insertions and deletions. If an intron does not exhibit the consensus `GT/AG` or `GC/AG` at the 5' and 3' ends, the boundaries are shifted up to two base pairs. If this still does not lead to the consensus, the sequence are split into two parts and consider each sub-sequence separately. Then, alignments are merged if they do not disagree and if they share at least one complete exon or intron. The next step involves clustering of alignments: In the beginning, each of the above EST and cDNA alignments is in a separate cluster. They are iteratively joined, if any two sequences from distinct clusters match to the same genomic location (this includes many forms of alternative splicing). From the clustered alignments a compact splicing graph representation can be obtained, which is used to generate a list of positions of true acceptor and donor splice sites. The inference of transcription start and stop sites, as well as translation initiation sites and stop codons is more complicated, in particular in the case of EST alignments, which in general do not cover the complete genes. In these cases heuristics are used, to generate true examples with relative high confidence (see below).

For some signals, for example for transcription start sites or trans-splice sites, there also exist a number of dedicated organism specific databases. The modular structure of our system allows to generate labels from these different sources as well.

[1]The file format is for instance described at http://www.sequenceontology.org/resources/gff3.html

In general, not all true signals are represented by any of these means. Therefore, negative examples are only drawn from genomic positions that are covered by a known gene. To further avoid the generation of false negative examples, genic regions that are known to allow alternative transcripts are also excluded. Within the boundaries of the alignments (10 bp at both ends of the alignments are cut out to exclude potentially undetected signal sites), all potential candidate positions that are not in the list of confirmed sites are identified.

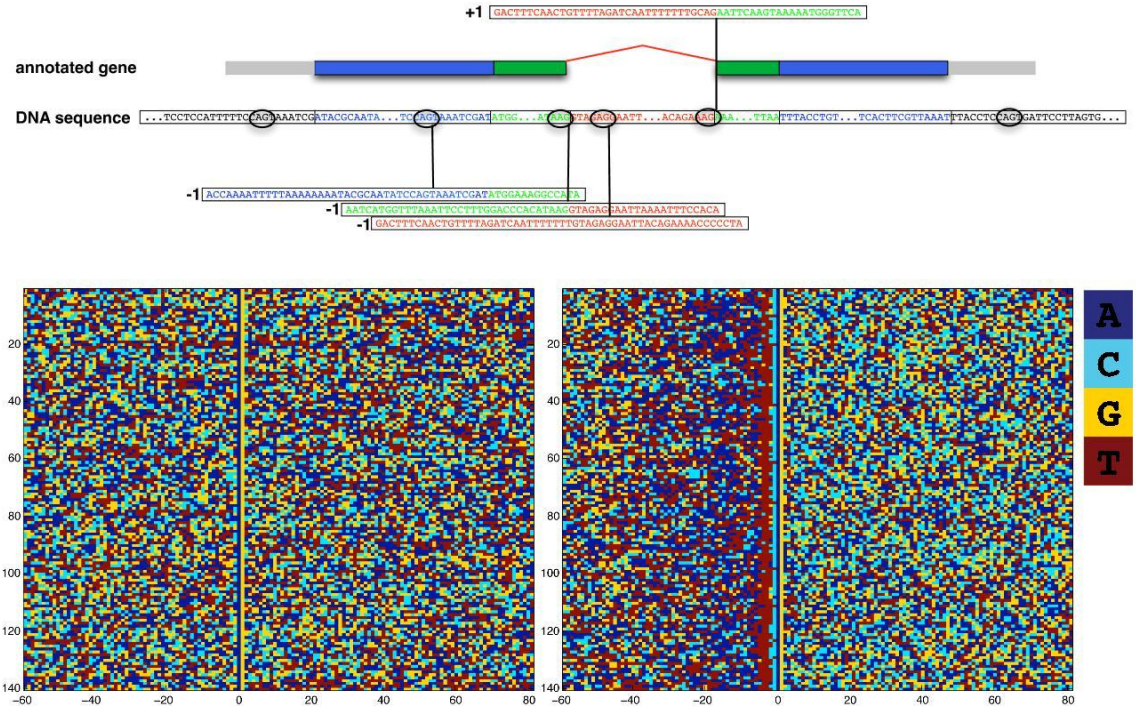


Figure 4.2.: Upper panel: Labeled examples (in this case for the *acc* detector) are extracted from the DNA sequence and annotated genes. Boundaries between segments define true sites. Any other genic site that is not within an alternative region and shows the consensus sequence is used as negative example. The sequence within a window around a candidate site is obtained and used along with the label information as input to the training algorithm. Lower panel: Aligned sequences around candidate acceptor splice sites derived from the genome of the model organisms *C. elegans*. The x axis defines the distance to the consensus site. The decoy sites are depicted on the left panel, true splice sites on the right. The color code is as follows: blue: Adenine, cyan: Cytosine, yellow: Guanine, red: Thymine. In both sets, the consensus sites can be recognized as the blue and yellow lines at $x = 0$.

4.1.2. Design of SVM-based Signal Detectors

We expect certain motifs in the vicinity of segment boundaries to guide the respective trans-factors towards the true sites. We therefore use the sequence within a window around the candidate positions as input to our classifiers. Figure 4.2 shows some examples of aligned sequences surrounding true and decoy acceptor splice sites (derived from *C. elegans*). It is immediately obvious that these sequences have quite different characteristics: the decoy sites appear more or less random; the true sites, on the contrary, exhibit some strong patterns. For example, immediately (1 bp) upstream of the `AG` site, a `C` is strongly preferred. This is preceded by a sequence of `T`s, the poly-pyrimidine tract. Like other, more degenerate motifs that cannot be detected here, (e.g. the branch point, which is about 20 nt upstream of the acceptor), these sequence patterns occur at specific positions relative to the segment boundary. To exploit these motifs for classification SVMs with the WDS kernel are used, as they take positional information of occurring k-mers into consideration (see Section 3.2). To account for the high variability of these motifs, kernels up to order 22 and with shifts ≤ 30 bp are employed to capture high order dependencies [148]. For some signals, it was observed that constructing the feature space from one continuous region around the consensus site is not enough to generate accurate predictions [146]. To account for long-range interactions (see for example Figure 2.4), sequences in windows further away from the actual boundary also have to be considered. We therefore take advantage of the fact that the sum of several kernels (possibly multiplied by non-negative weights) is again a valid kernel. Thus, several kernels aimed at modeling diverse types of evidence can be combined to achieve the best accuracy. We generally follow a common scheme: The central, position-dependent kernel is potentially flanked by two spectrum kernels, which act on sequences upstream and downstream of the potential signal site and are intended to model local sequence compositional features.

For each signal, this leaves several parameters to be specified. For WD kernel and spectrum kernel, the order needs to be determined, i.e. the length of the considered k-mers; the WDS kernel additionally requires to set the maximal allowed shift. For each kernel, the exact location and extend of the sequence window also needs to be defined. To set all these parameters, we first fix a broad set of values, supposed to cover the entire reasonable range. Then, a cross-validation scheme is used to determine suitable combinations of parameter settings. This is done separately for each signal sensor, as the mutual dependence of the signals is not yet taken into account at this stage. Note, that the optimal parameter setting is also organism specific, the model selection process therefore in principle has to be repeated for each new genome.

4.1.3. Cross-validation for Training, Model Selection and Prediction

For the application of the signal sensors to gene finding in a complete genome, it is essential to generate unbiased predictions for each single candidate position. The predictions are therefore computed using five-fold cross validation (see Figure 4.3): the sensor is trained on three fifths of the labeled data only and hyper-parameters are tuned on one fifth of the data, such that unbiased predictions can be generated and evaluated on the remaining fifth. The whole genome was therefore divided into regions, which are disjoint contiguous sequences containing at least two complete genes (as marked by the annotation or an EST/cDNA alignment); if an adjacent gene is less than 250 bp away, the adjacent genes are merged into the region. The splits are defined by randomly assigning clusters of regions to the splits, i.e. genes in the same region are assigned to the same cross-validation split. The procedure of training, parameter tuning and evaluation is repeated five times, cycling through the chunks of the partitioned data, to obtain predictions for the whole training set.

The reason for this rather complicated procedure is the following: In general the amount of label information is limited. The annotated genomic regions, from which the labeled set for training of signal sensors is derived, is therefore re-used for training the gene structure prediction system (i.e. the *second stage* of the gene prediction system, see Section 3.3.2). This can lead to problems, as SVM prediction values on examples that were included in the training set show a different distribution from prediction values on unseen examples. In particular, support vectors have an output of 1 or -1. The SVM outputs on the training regions, however, are input to the training of the second layer. And training on those might lead to a gene structure model that is sub-optimal on unseen regions, with a different SVM output distributions. Therefore, it was ensured that for each predicted position an SVM predictor is used that was trained only on other positions of the genome (a similar strategy has been used in [32]). For any position that was never used for training, I randomly chose one of the available SVM predictors to perform the signal prediction. Eventually, every candidate position b is decorated with a respective sensor output $f^s(\mathbf{x}, b)$.

4.1.4. Post-Processing of SVM Outputs

One potential problem of SVMs is the fact that their outputs are in general not normalized, i.e. it is unclear in advance, which range they will cover, in particular for different hyper-parameter settings (like SVMs' regularization and the kernel parameters). This might be the case for the five different SVMs, such that their outputs might not be comparable. To achieve a robust scoring system, we post-process the

outputs of each SVM: the posterior probability of observing a true site is estimated, thereby obtaining values in the range $[0, 1]$.

This normalization is performed by applying piecewise linear functions (see Section 3.3.3) that are determined on the corresponding validation sets. I used the $1/K$ quantiles taken on the SVM output values as supporting points $\mathbf{s}$. For all $K = 20$ points s_k I calculate v_k-values, which estimate the probability of being a true positive given an output between s_k and s_{k+1}, and additionally v_k^c-values, as cumulative scores for the probability of being a true positive for all positions with output values $f^s(\mathbf{x}, b) \leq s_l$:

$$v_k = \frac{n_{TP}(k)}{n(k)} \qquad \text{and} \qquad v_k^c = \frac{n_{c,TP}(k)}{n_c(k)}, \tag{4.1}$$

where $n(k)$ and $n_c(k)$ are the numbers of examples in the evaluation set with output values $s_k \leq f^s(\mathbf{x}, b) \leq s_{k+1}$ and $s_k \leq f^s(\mathbf{x}, b)$, respectively, and $n_{TP}(k)$ and $n_{c,TP}(k)$ denote the number of positively labeled examples in the same output ranges. The v values were further optimized by minimizing the relative entropy, using a second order Taylor development under the constraints: (i) monotonicity: $v_k > v_{k-1}$ and $v_k^c > v_{k-1}^c$ and (ii) $v_k^c > v_k \ \forall k$.

From now on, I will only consider the cumulative scores, as they are more robust. For any output value $f^s(\mathbf{x}, b)$ the corresponding confidence value $\hat{f}^s(\mathbf{x}, b)$ is then given by linear interpolation according to Equation 3.45:

$$\hat{f}^s(\mathbf{x}, b) \quad = \quad t_{\mathbf{s},\mathbf{v}}(f^s(\mathbf{x}, b)) \ . \tag{4.2}$$

This function is determined on the validation set for each SVM separately. In contrast, the transformations described in Section 5.3 are learnt simultaneously for *all* signals. For notational simplicity, in the following $f^s(\mathbf{x}, b) = \hat{f}^s(\mathbf{x}, b)$ will be set.

4.1.5. Implementation in mGene

The complete workflow for signal prediction is depicted in Figure 4.3. It is designed in a generic way, such that model selection, training and prediction of all signals can be performed within the same framework. In mGene, it is implemented using Matlab and can also be run under the free software Octave. For SVM training and prediction the freely available machine learning toolbox shogun is used, which offers interfaces to MatLab, Octave, and other programming languages [149]. Training and prediction of all models on all cross-validation splits can be performed in a parallel manner on a computer cluster.

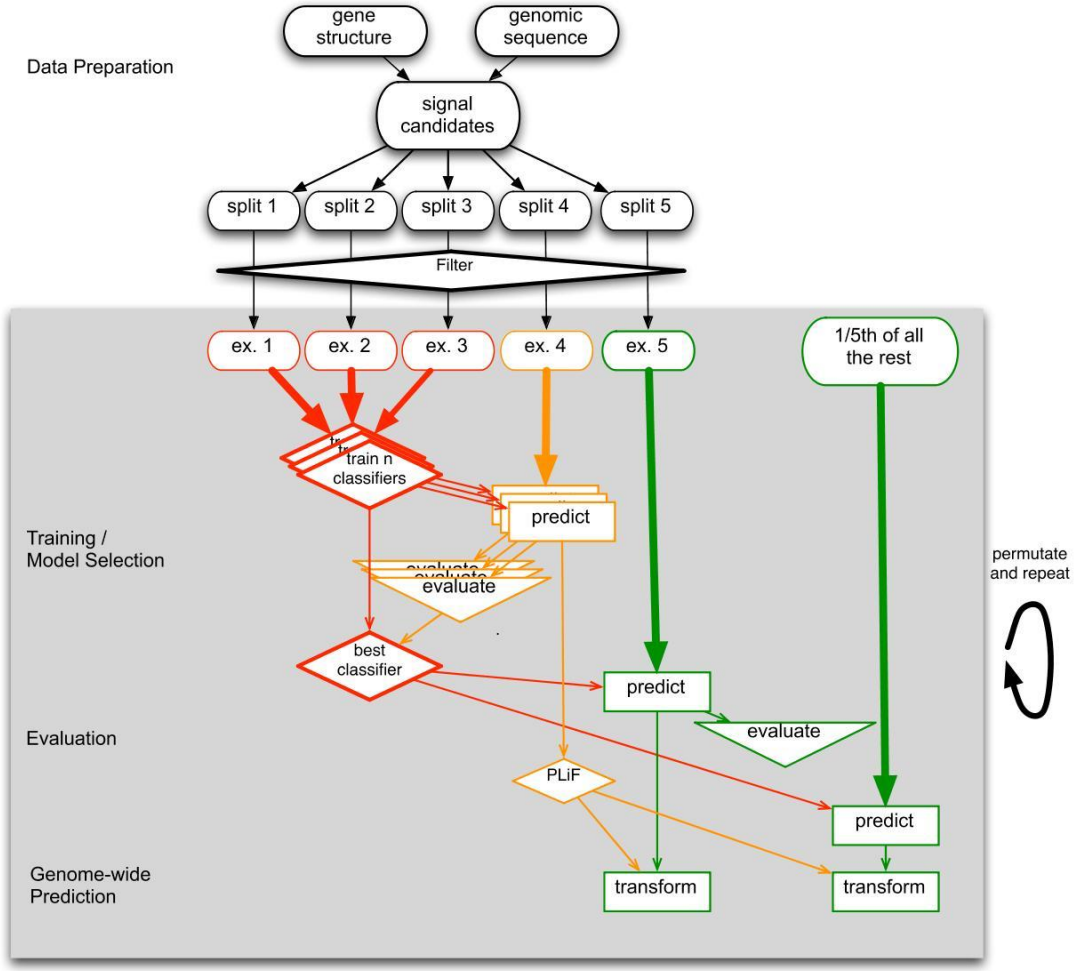


Figure 4.3.: Workflow for signal sensors: All genomic regions that are annotated for training are distributed into 5 different splits, from which candidate sites are derived. To avoid training (and evaluation) on erroneously labeled examples, filters are employed (for instance these filters remove examples from regions with incidence of alternative splicing). Three of the resulting sets (red) are used for training n different models, one (orange) is used to determine the best model and to estimate the piece-wise linear function (PLiF) for transformation of outputs. On the remaining set (green) unbiased predictions can be generated. The sets are permuted such that eventually there are unbiased predictions for any candidate position. For prediction on candidate positions that are never used in this procedure, one of the five SVMs is chosen at random.

4.2. Types of Signal Sensors

The strongest indicators for eukaryotic genes are splice sites, i.e. acceptor and donor sites that define exons and introns. Other strong signals include: translation start and stop sites, at the boundaries between untranslated regions (UTRs) and coding regions, and transcription start and cleavage sites between intergenic and genic segments. Moreover, I model poly-adenylation consensus signals, which are characterized by a 6-mer similar to `AATAAA` around 20nt upstream of the cleavage site. For *C. elegans* and other nematodes it is also important to model trans-splicing events, thereby enhancing the identification of gene starts.

In principle, achieving optimal gene finding results requires an extensive set of experiments, analyzing the employed methods with respect to state-of-the-art systems. Doing this for each single employed signal goes beyond the scope of this work. I have therefore only performed a detailed analysis for the splice sites, which are the strongest signals for gene detection.

4.2.1. Acceptor and Donor Splice Sites

All `AG` and `GC` or `GT` sites are considered as potential acceptor and donor splice sites, respectively. True splice sites are derived from confirmed annotated genes, or cDNA/EST alignments. The sensors are designed as described in the preceding section. As the local motifs around splice sites are very strong, they are the only signals that require only a central kernel for high accuracy performance. To compare our SVM-based approach to state-of-the-art methods, the problem was analyzed in three pilot studies.

Pilot Study I: Comparison of Classifiers on Different Data Sets

For these experiments we first used the data sets introduced by [11] and we compared our SVM-based method to all approaches proposed in [11, 30]. As a benchmark method, we also train higher order Markov Chains (MCs) (e.g. [51]) of "linear" structure and predict with the posterior log-odds ratio (see Section 2.7). The results of the comparison are presented in detail in [148]. Here I only note, that our approach is superior to the compared methods for all acceptor recognition tasks, however it is outperformed by the MC based predictions on one donor recognition task. In [148] we could show that this is partly a result of how the data set was selected and designed: Especially for the donor tasks the sequences used were very short, namely between 15 and 18 bp. The influence of the sequence length will be discussed in the following in more detail.

We also compared our approach to SpliceMachine [48], which is the current state-of-the art splice site detector. It is based on a linear SVM, capturing position-

dependent motifs around the splice site. However, it additionally models coding-frame specific compositional context. This is omitted in our approach as we will model it with independent content detectors in the framework of a gene finder. Because SpliceMachine uses longer sequences we generated our own data sets for worm, fly, cress, fish and human by aligning available ESTs to the respective genomes (for details see [148]). We tested SpliceMachine and the WDS-SVM approach on subsets of the human and cress data and could show that our method is superior in three out of four tasks. The details are again presented in [148].

Pilot Study II: Choice of Input Features

To observe the influence of the considered sequence window, we then performed the following experiments: We started with a subset of our new data sets for worm and human. Training and testing sequences were shortened from 218 bp in steps of 10 bp down to 18 bp. We then trained and tested MCs and WD-SVMs for the sets of sequences of different length. Figure 4.4 shows the resulting values for the auPRC on the test data for different sequence lengths. For the short sequences, the prediction accuracies of MCs and SVMs are close for both organisms. For human donor sequences of length 18 bp, MCs outperform SVMs. With increasing sequence length, however, the auPRC of SVMs rapidly improves while it degrades for MCs.

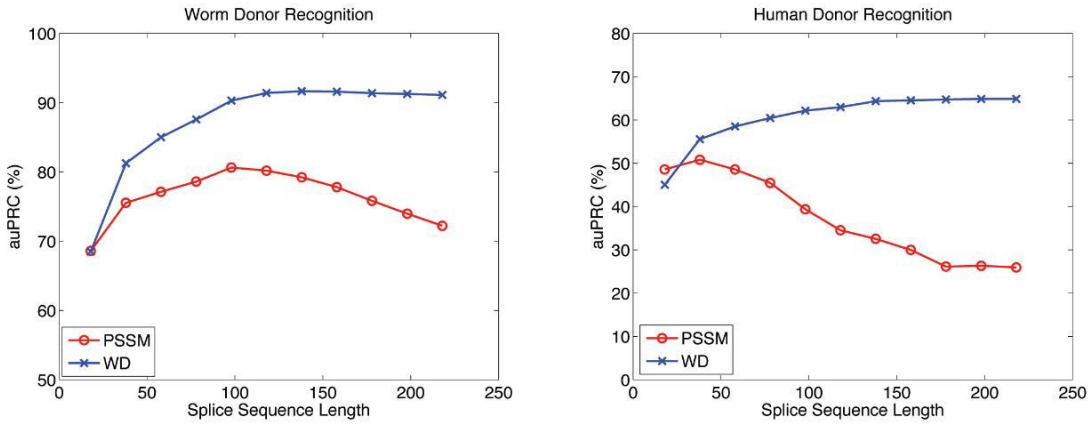


Figure 4.4.: Comparison of classification performance of the weighted degree kernel based SVM classifier (WD) with the Markov chain based classifier (MC) on a subset of our *C. elegans Donor* and *Human Donor* data sets for sequences of varying length. For each length, we performed a full model selection on the training data to choose the best model. The performance on the test sets, measured through area under the Precision Recall Curve (auPRC), is displayed in percent.

We therefore conclude that discriminative information between true and decoy splice sequences lies not only in the close vicinity of the splice site but also further

away. Importantly, our SVM-based approach is capable of exploiting this information. We established a window of length 141 bp for splice sites, which is relatively long compared to what is used in most other gene finders (typically 10-15 bp). Therefore, these signal sensors do not only consider motifs in the immediate neighborhood of the targeted signal site, but also more distantly located patterns, like intronic or exonic splice enhancers or silencers. This strategy leads to an improved prediction performance.

Pilot Study III: Influence of Training Set Size

With our subsequent set of experiments we question whether it is indeed beneficial to use as many labelled examples as possible for training. Figure 4.5 shows the prediction performance in terms of the auROC and auPRC of SVMs using the MC and the WD kernel on the human acceptor and donor splice data for varying training set sizes. For training, we use up to 80% of all examples and the remaining examples for testing. MCs and SVMs were trained on sets of size varying between 1000 and 8.5 million examples. Here we sub-sampled the negative examples by a factor of five. We observe that the performance steadily increases when using more data for training. For SVMs, over a wide range, the auPRC increases by about 5% (absolute) when the amount of data is multiplied by a factor of 2.7. In the last step, when increasing from 3.3 million to 8.5 million examples, the gain is slightly smaller ($3.2 - 3.5\%$), indicating the start of a plateau. Similarly, MCs improve with growing training set sizes. As MCs are computationally a lot less demanding, we performed a full model selection over the model order and pseudo counts for each training set size. For the WD-SVM, the parameters were fixed to the ones found optimal in the results section. Nevertheless, MCs did constantly perform inferior to WD-SVMs. We may conclude that one should train using all available data to obtain the best results. If this is infeasible, then we suggest to only sub-sample the negatives examples in the training set, until training becomes computationally tractable. The class distribution in the test set, however, should never be changed unless explicitly taken into account in evaluation.

4.2.2. Translation Initiation Sites and Stop Codons

All positions with a consensus `ATG` are potential canonical *tis* sites, i.e. translation initiation sites[2], positions with `TAG, TGA` or `TAA` are candidate *stop* codons. These signals mark the boundaries between untranslated regions (UTRs) and coding regions. True sites are generally acquired from annotations. If this is not sufficient, they can also be derived from the alignment of full-length cDNAs. In this case, it is

[2]There exist rare cases of non-canonical translation initiation sites, for example in plants with a consensus `ATT` [135]. These are not considered in this work.

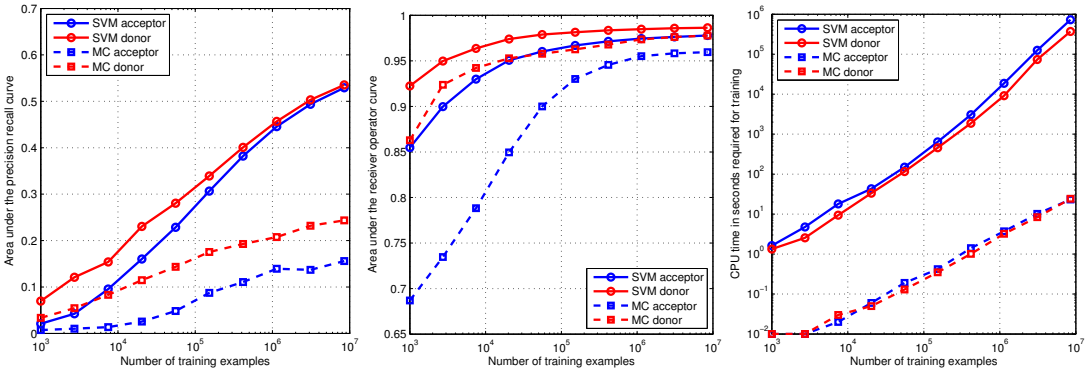


Figure 4.5.: Comparison of the classification performance of the weighted degree kernel based SVM classifier (SVM) with the Markov chain based classifier (MC) for different training set sizes. The area under the Precision Recall Curve (auPRC; left) and the area under the Receiver Operator Curve (auROC; middle) are displayed in percent. On the right the CPU time in seconds needed to train the models is shown.

assumed that the *longest* open reading frame is also the correct one, thus defining the true *tis* and *stop* of a gene.

As described above, we use SVMs with a central WDS kernel, and two flanking spectrum kernels for prediction. Previously, SVMs have been used with polynomial, locality improved and Salzberg kernels for the task of *tis* prediction [194]. Also, in [95], SVM-based *tis* detectors have been suggested with a class of edit kernels, and in [59] an SVM-based approach was successfully applied for *tis* detection in prokaryotes. Less work is available on the prediction of stop codons, probably because they can be inferred if the proper *tis* and splice form are known. While I have not analyzed the performance of our method compared to other approaches, in the above mentioned studies the SVM-based methods consistently outperformed generative approaches. However, most state-of-the-art gene finding systems model the *tis* (and *stop*) by much simpler means, mostly using low-order WMMs or WAMs on relatively short sequences (see Section 2.7). One exception is the gene finder contrast that employs SVMs with quadratic kernels on sequence windows of length 6 bp. In mGene, sequence windows up to length 200 bp are considered and kernels up to order 24 are applied.

4.2.3. Transcription Start and Cleavage Sites

In contrast to all other segment boundaries, transcription start sites, *tss*, and cleavage sites, *cleave*, do not show a compulsory consensus sequence. In this respect, any position in the genome is a potential candidate site. True sites are collected from official annotations. However, they can also be derived from the alignment of

full-length cDNAs or from specialized databases such as dbtss [178].

For the detector, we again use SVMs with a central WDS kernel and two flanking spectrum kernels. These kernels act on particular large sequence windows thus capturing long-range interactions. The design of the *tss* detector is strongly based on previous work by [146], which also showed the excellent performance when compared to other gene start predictors. We compute *tss* and *cleave* prediction for each single position. However, for efficiency reasons only the sites with the highest prediction within a window of length 20 bp are considered later on.

4.2.4. PolyA Consensus Sites

Albeit their strong signature, polyA sites are typically not included in a genome annotation. They are also not at the boundary between two different segments, but within the 3' UTR. For training, a signal sensor we first need not only to generate labeled examples but also to determine the most likely consensus sequences. We therefore look for variants of `AATAAA` with at most two mismatches that are significantly overrepresented at the expected region around 30 bp upstream of the cleavage site (in the case of nematodes). As a background model, sequence regions further away from the cleavage site are considered. For the prediction we again use SVMs with a combination of several kernels. During preliminary analysis, optimal performance was found with five spectrum kernels and one central WDS kernel.

4.2.5. Trans-splice Acceptor Sites

To improve gene start recognition in nematodes, we have additionally designed a trans-splicing sensor, *trans*, that distinguishes cis-splicing from trans-splicing at acceptor sites. In this case, all true *cis*-acceptor sites serve as negative examples. If trans-splicing is not annotated and no information on it can be extracted from an additional database, we use a heuristic to guess true trans-splice sites at the beginning of genes: We therefore look for strong acceptor signals in the very close vicinity of annotated gene starts that occur before the annotated *tis*. Eventually, for every consensus site `AG` we compute trans-acceptor score.

4.2.6. Summary

Table 4.1 provides details about the signal detectors, including the considered sequence regions and the employed combination of kernels. Eventually each candidate locus b for a signal s is furnished with the score from the appropriate sensor, $f^{\mathtt{s}}(\mathbf{x}, b)$. In the second stage (the gene structure prediction), only positions b are considered

that are in the union of all candidate positions. The signal score for locations without the required consensus for a given signal is set to $-\infty$, thus preventing its use as this signal in the second layer.

	left kernel		central kernel		right kernel	
	win	type	win	type	win	type
tss	[-400 200]	Sp (d=4)	[-50 70]	WDS (d=14,s=28)	[-100 800]	SP (d=4)
cleave	[-400 100]	Sp (d=5)	[-60 60]	WDS (d=14,s=28)	[0 800]	SP (d=3)
tis	[-200 0]	Sp (d=3)	[-30 110]	WDS (d=24,s=0)	[0 180]	SP (d=5)
stop	[-200 0]	Sp (d=3)	[-110 10]	WDS (d=20,s=0)	[0 180]	SP (d=4)
acc	—	—	[-60 80]	WDS (d=22)	—	—
don	—	—	[-80 60]	WDS (d=22)	—	—
trans	—	—	[-190 80]	WDS (d=30)	—	—
polya	{[-300 -150] [-149 0]}	SP (d=4)	[-300 306]	WD (d=18)	{[-300 306] [1 150] [151 306]}	SP(d=4)

Table 4.1.: Sequence windows and SVM kernels used for signal sensors in mGene as determined for *C. elegans*. The coordinates are given with respect to the signal (consensus) site.

For comparison, employed signal sensors for other state-of-the-art gene finders are described in Table 4.2. Despite the fact that SVMs have been shown to perform superior to PWMs, or WWAM in individual studies on signal detection, most gene finders still use these simple methods in their sub-models. One exception is Contrast, which uses SVMs with quadratic kernels on relatively short sequence windows.

	Augustus [152]			Conrad [47]		Contrast [70]			Craig [16]		
	method	par	len	method	len	method	par	len	method	par	len
tss	—			—		—			—		
cleave	—			—		—			—		
tis	WWAM	d=2,s=5	20	PWM	?	SVM, pol	d=2	14	PWM,WWAM	d=3,s=5	11,14
stop	—		3	PWM	?	SVM, pol	d=2	6	PWM,WWAM	d=3,s=5	10,14
acc	PWM		6	PWM	?	SVM, pol	d=2	30	PWM,WWAM	d=2,s=3	10,20
bp	WWAM	d=3,s=7	32	—		—			WWAM	d=3,s=5	16
don	SBSW		9	PWM	?	SVM, pol	d=2	11	PWM		10

Table 4.2.: Signal sub-models for some state-of-the-art gene finders, including Augustus, Conrad, Contrast and Craig. Given are the employed methods with the respective parameters and the length of the considered sequences. For the WWAM approach the parameters include the order d and the size of the window, s, within which strings are pooled. For the donor recognition tasks, Augustus uses a similarity-based sequence weighting (SBSW, see [152] for further details). In the case of Conrad, the sequence length for the PWMs could not be determined. Contrast uses SVMs with polynomial kernels of order 2. Craig employs a combination of PWMs and WWAMs for the individual signals. Note, that due to the complexity of the systems the sub-models may not be completely comparable, and furthermore as the development of the methods has been continued since the original publications the models might have changed in the mean time.

4.3. Content Sensors

In the sequence examples around acceptor splice sites in Figure 4.2 we can not only observe position-dependent patterns around slice sites, but also more global trends that can be associated to complete segments: for example, exonic regions (to the right of the acceptor site) show a far higher GC content than intronic regions (to the left of the consensus). Compositional differences can also be observed in two-mers as demonstrated in Figure 4.6.

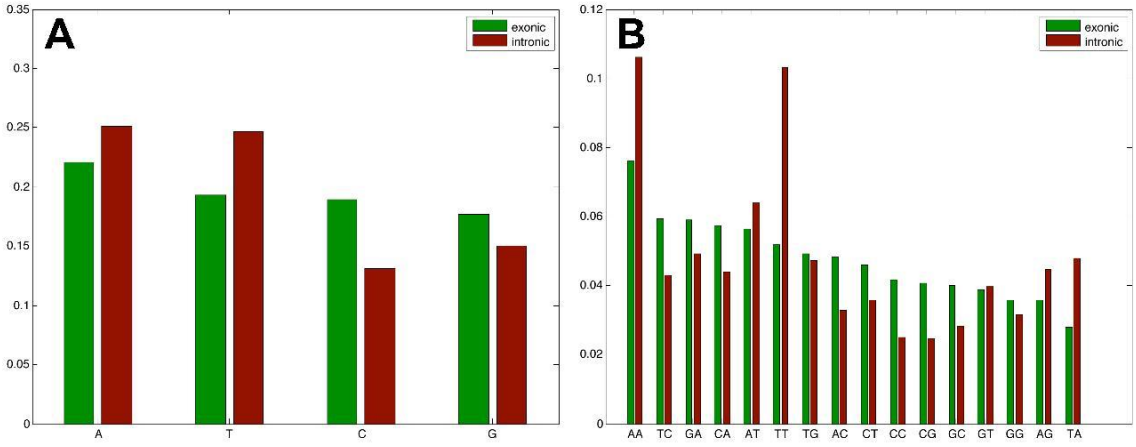


Figure 4.6.: Sequence composition of exons vs. introns: relative frequency of occurring 1-mers and 2-mers in a 50 bp sequence window downstream and upstream of acceptor splice sites, respectively. (Values averaged over the examples depicted in 4.2)**A** occurrence of all 1-mers **B** occurrence of all 2-mers.

We have therefore designed content sensors to recognize the typical sequence composition of the individual segments. With some extensions the same framework as for signal prediction can be applied. Therefore, we set up binary, one-against-the-rest classification problems for each of five different content segment classes — namely, *intergenic*, *inter-cistronic*, *UTR*, *coding exon* and *intron*. Examples are generated from annotations, where negative examples correspond to segments of a different kind than the segment of interest. To avoid the influence of different length distributions of the segments on the discrimination, negative examples are chosen such that their length distribution equals that of the positive sequences. The content sensors are also realized by SVMs with sequence kernels. To model the k-mer compositions of segment types, we use the spectrum kernel of order k. As oligomers of several lengths may be indicative of the segment type, we in fact use a sum of several spectrum kernels. For each task four SVMs are trained with spectrum kernels of order $k = 3$ to $k = 6$ (Table 4.3).

For coding sequences, a modified version of the spectrum kernel was additionally implemented. It is especially suitable for picking up periodic characteristics like

	seq kernel 1		seq kernel 2	
	step	orders	*step*	orders
intergenic	1	3-6		
inter-cistronic	1	3-6		
UTR	1	3-6		
coding exon	1	3-6	3	3,6
intron	1	3-6		

Table 4.3.: Kernels used for content sensors in mGene.

coding potential. It only takes those k-mers into account that are located at a distance from the segment start that is a multiple of a number to be specified, called *step*. Obviously, the standard spectrum kernel is contained as special case for $step = 1$. For the coding sequence sensors, we use $step = 3$.

Similar to signal sensors, it is also important for the content sensors to avoid biased predictions, i.e. predictions with models that were learned on the same examples. We therefore employ a five-fold cross-validation scheme as described in the section for signal sensors. In regions that are not used for training, any one of five trained SVM's is randomly chosen for prediction. Eventually we require for every possible candidate segment $[b, b']$ of length $b' - b + 1$, the computation of a SVM score $f^c(\mathbf{x}, b, b')$ for each possible content sensor c. The allowed content types are defined by the boundary signals, i.e. if positions b and b' constitute an acceptor and a donor consensus, respectively, than the only allowed segment types that need to be interrogated are *coding exon* or *UTR*. The problem in this case is, that the segmentation is not known *a priori*: In principle, any random pair of `GC` or `GT` and `AG` needs to be interrogated for *coding exon* and *UTR* content. If the exon length was restricted to 8000 bp this would lead to approximately three billion candidate exon segments. It is therefore difficult to efficiently precompute the content sensors, as they depend on the start and the end of the segment, which is only known during computation of the segmentation. In contrast to the signal sensors — the outputs of which are pre-computed in order to save substantial amounts of computation time — the content predictions were calculated "on the fly" while determining the segmentation (in the dynamic program). In a later version of mGene, we exploited the fact that the content predictions with spectrum kernels are positional decomposable. We therefore pre-computed $f^c(\mathbf{x}, 0, b)$ for all positions b, such that a prediction for a segment $[b, b']$ is given by $f^c(\mathbf{x}, b, b') = f^c(\mathbf{x}, 0, b') - f^c(\mathbf{x}, 0, b)$.

4.4. Exploiting Additional Information

As briefly discussed in Section 2.7, additional information can be used as external hints to improve gene finding. We consider two types of additional information: transcription evidence based on proteins and EST alignments and conservation information which is derived from whole genome alignments. The incorporation of these data types was implemented in an initial, rather simple, preliminary version. In the case of EST and protein alignments, we applied a post-processing step to the signal predictions. The genome alignments were used both, for signal *and* content sensors, by employing an additional kernel to the conservation scores.

EST and Protein Sequence Alignments

To utilize known EST,cDNA or protein sequences as hints to improve the prediction accuracy, we first align the sequences to the genome according to the method described above (Section 4.1.1). The alignments are than used to define regions $R^{s,l}$ that have a certain likelihood l to contain a particular signal s. As described in Figure 4.7, we have $l \in \{$ *very likely*, *likely*, *very unlikely* $\}$. For example, all introns inferred from EST/cDNA alignments were used as strong evidence for the occurrence of splice sites at their boundaries. On the other hand, it is *very unlikely* to find splice sites within introns or exons. If a candidate position b is within one of the regions defined for the corresponding signal, than its SVM output $f^s(\mathbf{x}, b)$ is transformed according to $(f^s(\mathbf{x}, b))^{p_l}$ to enhance or suppress signal use during the gene structure prediction. The parameters p_l were chosen heuristically, as well as the extend of the regions. Note, that by exploiting EST and protein alignments in this way, we can use the *ab initio* model, and don't need to retrain.

Multiple Genome Alignments

To utilize conservation information, we first preprocess the multiple genome alignment by deriving a conservation score for each position, which is essentially a count of pairwise nucleotide matches. This score is then discretized into four bins, which enables the application of the nucleotide sequence kernels to these data. In the case of signal sensors, these additional kernels were applied to the same sequence windows as the central kernels for each signal, and the degree was determined separately by cross-validation. For all content sensors, we used two additional spectrum kernels with $step = 3$, one of order 3 and one of order 6.

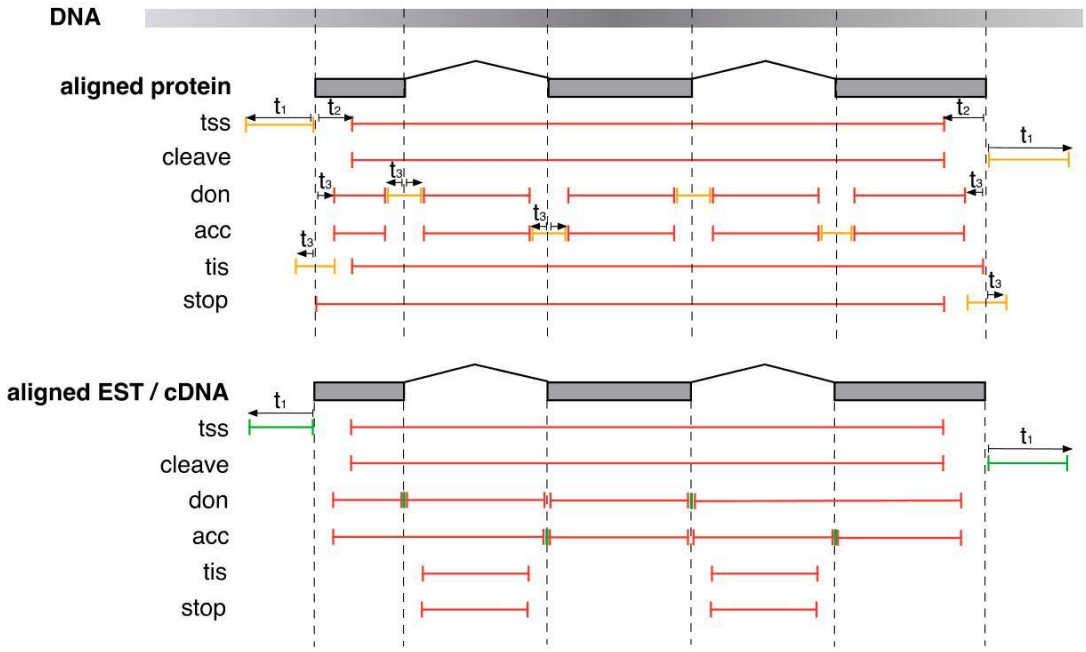


Figure 4.7.: External information used to improve gene finding accuracy: Based on EST/cDNA and protein alignments, we define regions that are very likely (green), likely (orange), or unlikely (red) to contain a certain signal. The extends of the regions are set heuristically.

5. Gene Structure Prediction

At this point let me recapitulate the ultimate goal of the presented project: We are given a genomic DNA sequence and the task is to compute a complete, biologically likely segmentation of this sequence that determines the exact positions and structures of encoded genes. As opposed to this goal, we have so far only generated several unrelated and independently computed lists of predictions for different genomic features (see upper panel of Figure 5.1). Now, we use these predictions to solve the segmentation task. Selecting the actual signal sites based on a simple thresholding approach on the prediction scores would ignore the *grammar* for gene structures and lead to biologically implausible predictions. Therefore, we employ a strategy to solve this label sequence learning problem that is based on the HSM-SVM approach described in Section 3.3.4. The individual signal outputs are thereby reconciled according to a biological model for gene structures (see left inlet of Figure 5.1). In the following paragraphs, I will first introduce the state model that we used to describe valid gene structures. This is followed by a short discussion of the actual input features and a description of the discriminant function $G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$, which is employed to score possible gene structures. I will subsequently present the regularizer and the loss function used for training. The chapter will be closed by a description of the implementation in the gene finding system mGene.

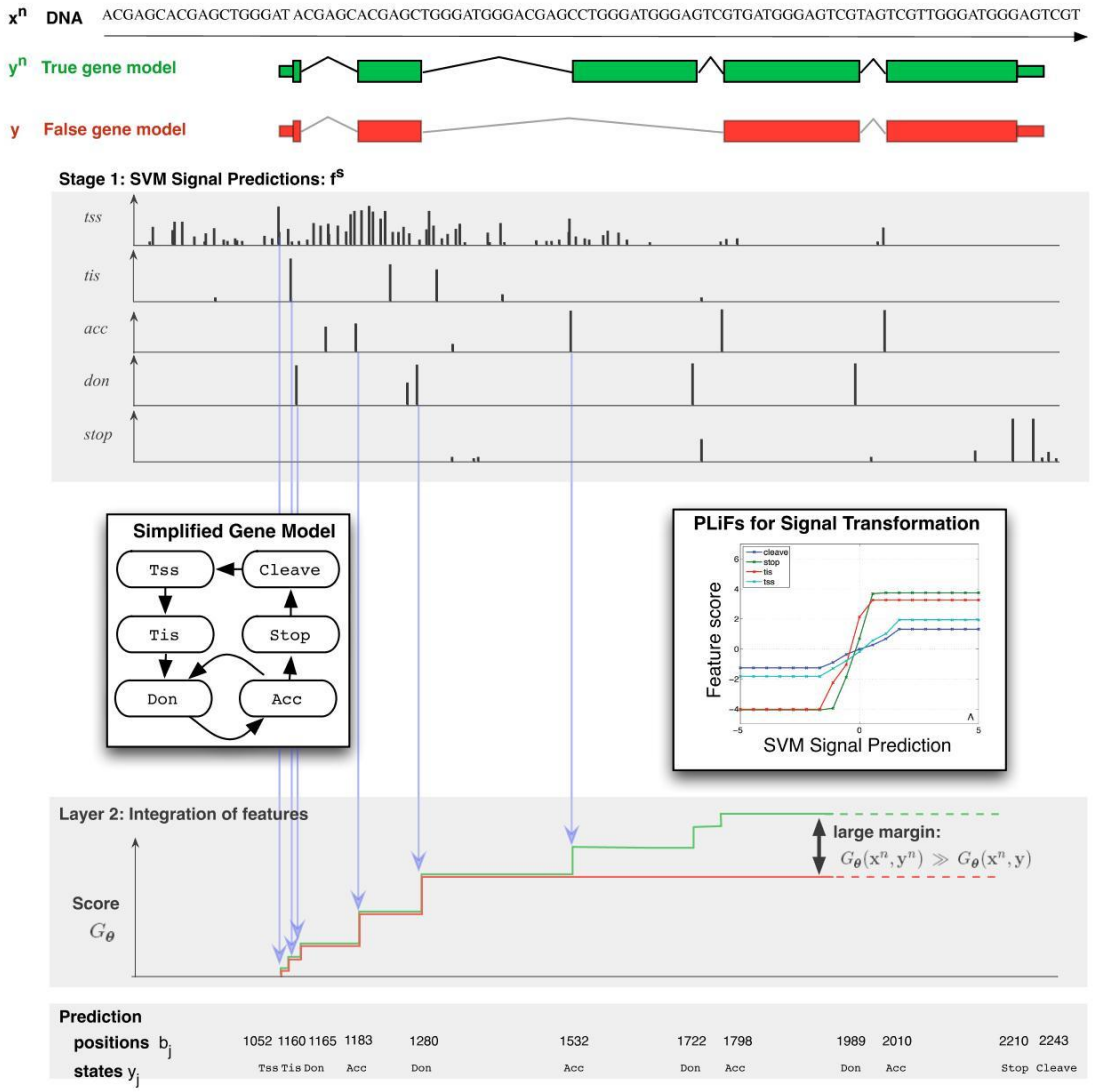


Figure 5.1.: Independently predicted scores for genomic features are input to the gene structure predictor. In combination with additional information including scores for segment lengths (not shown), these scores contribute to the cumulative score $G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ for a putative segmentation $\mathbf{y}$. The bottom graph illustrates the computation of the cumulative scores for two gene structures shown at the top, where the score at the end of the sequence is the final score of the gene structure. The contributions of the individual detector outputs, lengths of segments as well as properties of the segments to the score are adjusted during training using piece-wise linear functions (PLiFs). They are optimized such that the *margin* between the true gene structure (shown in green) and all other (wrong) segmentations (one of them is shown in red) is maximized.

5.1. A State Model Defining the Gene Structure

According to our pre-knowledge, not all possible segmentations are allowed as a solution to the gene finding problem, but a valid segmentation has to obey certain biological rules. These rules are captured in a state model, which we define such that states $\mathcal{Q}$ correspond to segment boundaries, while transitions $\mathcal{T}$ between states correspond to whole segments.

Eventually, we have arrived at the model, depicted in Figure 5.2. As this model includes many known rules in the transcription, RNA processing, and translation processes, it is predefined in advance as a part of the gene prediction system. Each segmentation has to start in the `Begin` state and stop in the `End` state. The model allows for zero, one or more detected genes. In contrast to many other gene finders, it includes a model for the untranslated regions. Besides the fact that the knowledge of UTRs constitutes valuable biological information, it is also expected to improve the overall accuracy for prediction [24]. Compared to the simple model (see left inlet of Figure 5.1) we have introduced more states for two reasons: First, we can thereby restrict allowed transitions to a given context, that depends not only on the immediately preceding label. For example, splicing (transitions from donors to acceptors) can occur in the untranslated regions, as well as in the coding regions. However, from the 3'UTR, the segmentation must eventually pass through the translation initiation start (`Tis`), which cannot be reached again after splicing of coding regions. As a second advantage, we can carry additional information through the system that is not included in a given signal itself. For example, the splice sites in the coding regions are further split into states `Don0` - `Don2` and `Acc0` - `Acc2`, which is necessary to propagate the coding frame information across an intron to the next exon. Additionally, the out-of-frame states `Don1`, `Don2` and `Acc1`, `Acc2` are further divided to avoid the creation of an in-frame stop codon through splicing: For example, a transition from a frame 1 donor that is preceded by a `'T'`, i.e. `Don1b`, to an acceptor that is followed by `'AA'`, `'AG'` or `'GA'`, i.e. `Acc1a`, is prohibited, as this would generate an in-frame stop codon.

To account for the specific nematode gene biology, we model trans-splicing with `SL1` and `SL2` states (see Section 2.4). The model also considers polyA sites in the 3' UTR. We have also added some mixed states to accommodate special cases like combinations of acceptor splice sites with translation initiation sites (i.e. `AccTis`, with the consensus `AG|ATG`, where `|` denotes the intron-exon boundary) or of acceptor splice sites with the stop codon (i.e. `AccStop`, with the consensus `AG|TAA`, `AG|TAG` or `AG|TGA`).[1]

[1] In the implementation, the position *after* the `AG` (the first base of the exon) is associated with the acceptor site, while the position *before* the `ATG` (the first base of the ORF) is associated with the TIS. In `AG|ATG`, the position of the second `A` corresponds to both, acceptor and TIS.

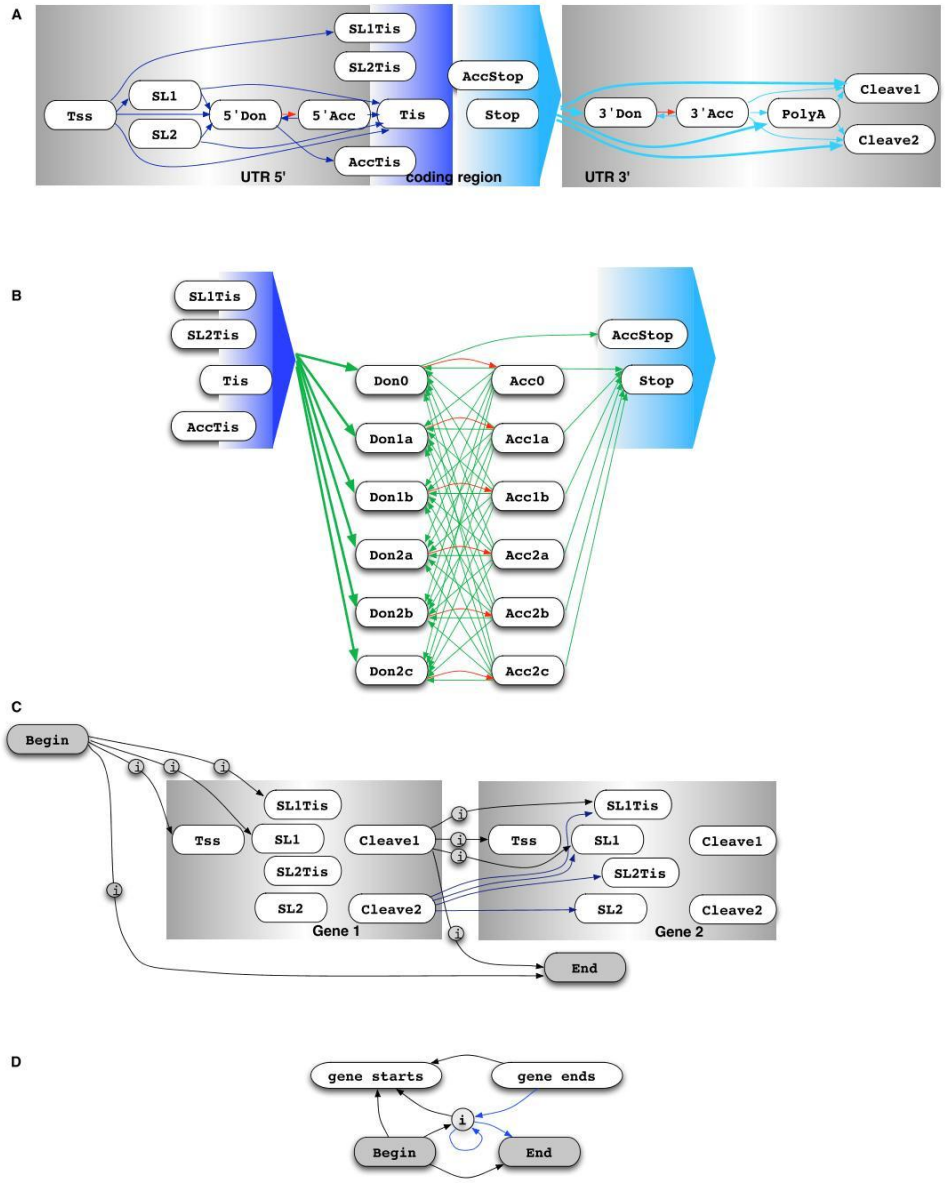


Figure 5.2.: State model for gene predictions in nematodes, represented in four levels of detail. States that share all their outgoing transitions are drawn in boxes with right-pointing triangles (e.g. `AccStop` and `Stop`). The outgoing transitions are represented only once for all the grouped states. **A** Gene model, with all details for the untranslated regions. Blue transitions are associated with *UTR* content detectors, red transitions with *intron* content detectors. **B** Additional states and transitions in the coding region. Green arrows are associated with *exon* and *frame* content detectors, red ones with *intron* detectors. **C** Transitions between genes. All transitions within a gene are removed and the states that mark the beginning and end of a gene are represented twice, once for gene 1 and once for gene 2. Intergenic transitions are marked with a circle and treated specifically, see below. The state `cleave 2`, is used to model operons. All transitions that are out-going from this state (blue arrows) are associated with the *intercistronic* content detector, all others (black arrows) with the *intergenic* content detector. **D** Intergenic regions can be particularly long. For efficient decoding, a dummy state `i` is used.

5.2. Input Features to the Gene Structure Prediction System

In principle, the genomic DNA sequence should be used as input $\mathbf{x}$ to the gene structure prediction system. The HSM-SVM would then optimize predictive functions for all genomic signatures simultaneously and also determine their relative importance. However, when high-order dependencies are expected, training on a large number of examples soon becomes intractable. We have therefore applied the two-stage learning scenario as described in Section 3.3.2. In particular, we do not only take the DNA sequence $\mathbf{x}^n$ itself, but also the sequences of precomputed signal predictions, $f^s(\mathbf{x}^n, b)$, (see Section 4.1) as input to the label sequence learning problem. They are stored in matrices $\mathbf{FSX}^n$, with $\mathbf{FSX}^n(b, s) = f^s(\mathbf{x}^n, b)$. With this strategy, the number of parameters that have to be optimized in the integration step are dramatically reduced, as only the scaling and weighting of the individual features has to be learnt. Further, we have two approaches that handle the content predictions, $f^c(\mathbf{x}^n, b, b')$ in different ways (Section 4.3): The first approach, computes the content predictions on the fly during the dynamic program. Therefore, the system is provided with the trained content models. In a more recent design the content predictions are also positionally precomputed and given to the gene structure prediction algorithm as input $\mathbf{FCX}^n$, with $\mathbf{FCX}^n(b, c) = f^c(\mathbf{x}^n, 0, b)$. Due to the additivity of the sensors, the content score for a segment is then given by $f^c(x^n, b, b') = \mathbf{FCX}^n(b', c) - \mathbf{FCX}^n(b, c)$.

In addition to signal and content predictions, segment length information are used as a third type of feature. In terms of length characteristics $\mathcal{L}$, we distinguish between first, middle and last coding exons as well as single exons of unspliced genes (compare Figure 2.5). Additionally, we consider 5' and 3' UTR exons, trans-exons (the segment between a trans acceptor site and the translation initiation site), polyA-tails, introns, intergenic and intercistronic segments. As the length of a candidate segment $[b, b']$ is trivially computed if the positions of the segment boundaries are known, these features do not require any pre-computation: $f^l(b, b') = b' - b$, with $l \in \mathcal{L}$. We use annotated training examples to determine the allowed length range for a given length feature.

Finally, we consider features corresponding to occuring transitions from a state q to q' (similar to transition probabilities used in HMMs see Section 2.7). They are simply given by a constant $f^{(q,q')} = 1$ (see below).

In our approach, the considered signal features at a given position depend on the assumed state at this position. For example, if at a given position an acceptor state occurs, then only the acceptor signal feature is taken into account. Similarly, the content and length features applied depend on the considered transition. Therefore,

in addition to the state model, we define tables that specify which features will be associated with a given state or transition. This is summarized in Table 5.1 for states, which are linked to signal features and in Table 5.2 for transitions, which are linked to content and length features. This strategy also allows a flexible incorporation of additional features that are active at particular segments or segment boundaries. To distinguish between states and transitions, ($\mathcal{Q}$,$\mathcal{T}$), on the one hand, and features ($\mathcal{S}$,$\mathcal{C}$,$\mathcal{L}$) on the other, we have set state and transition names in typewriter typeface and employ italics for feature names in the following. It is noteworthy that we have defined 32 states and 123 transitions, while we have only employed 8 signal and 8 content sensors. There is therefore no one-to-one mapping between states and signals, or transitions and contents. Some states are linked to more than one signal, like the trans-acceptor sites (`SL1` and `SL2`), which use both the *acc* and *trans-acc* sensors. Also, the same features are employed for different signals. For example, the signal feature *acc* is employed for all acceptor states and additionally the trans-acceptor states. In the following the look-up functions that return sets of features for a given state or transition, will be denoted by $sig(q)$, $cont(q, q')$ or $len(q, q')$.

State, $q \in \mathcal{Q}$	Consensus	State Description	Signal Features, $sig(q) \in \mathcal{S}$
Begin	—	beginning of sequence	—
Tss	—	gene start	*tss*
SL1	AG	trans acceptor site of type SL1 (potential gene start)	*trans-acc*, *acc*[1]
SL1Tis	AGATG	SL1 and Tis share same position, (i.e. 5' UTR has length 0)	*trans-acc*, *acc*[1], *tis*
SL2	AG	trans acceptor site of type SL2 (potential gene start within an operon)	*trans-acc*, *acc*[1]
SL2Tis	AGATG	SL2 and Tis share same position, (i.e. 5' UTR has length 0)	*trans-acc*, *acc*[1], *tis*
5'Don	GC,GT	donor in the 5' UTR	*don*
5'Acc	AG	acceptor in the 5' UTR	*acc*
AccTis	AGAGT	Acc and Tis share same position, (UTR exon has length 0)	*acc*, *tis*
Tis	AGT	Translation initiation site	*tis*
Don0	GC,GT	frame 0 donor	*don*
Acc0	AG	frame 0 acceptor	*acc*
Don1a	vGC,vGT	frame 1 donor	*don*
Acc1a	AG	frame 1 acceptor	*acc*
Don1b	tGC,tGT	frame 1 donor	*don*
Acc1b	yyAG	frame 1 acceptor	*acc*
Don2a	vyGC,vyGT	frame 2 donor	*don*
Acc2a	AG	frame 2 acceptor	*acc*
Don2b	taGC,taGT	frame 2 donor	*don*
Acc2b	aAG	frame 2 acceptor	*acc*
Don2c	tgGC,tgGT	frame 2 donor	*don*
Acc2c	bAG	frame 2 acceptor	*acc*
AccStop	AGTAA,AGTAA,AGTGA,	Acc0 and Stop share same position, (i.e. last exon has length 3 bp)	*acc, stop*
Stop	TAA,TAA,TGA,	stop codon	*stop*
3'Don	GC,GT	donor in the 3' UTR	*don*
3'Acc	AG	acceptor in the 3' UTR	*acc*
PolyA	AATAAA, AATGAA, TATAAA, CATAAA, ATTAAA, AGTAAA, TATATA, AATAAG, GAAAAA	polya consensus site	*polya*
Cleave1	—	gene end	*cleave*
Cleave2	—	internal gene end (within an operon)	*cleave*
i	—	state introduced for technical reasons within long intergenic regions	—
End	—	end of sequence	—

Table 5.1.: To account for the complicated grammar of gene structures we designed a model with 31 states in total. Most of these states require a specific consensus sequence. To accomodate coding frame information, several acceptor and donor states are defined in the coding regions. Frame 1 and 2 splice sites are additionally split into several states, to avoid the construction of in-frame stop codons across splice sites. In these cases, the consensus sequences are expanded as indicated with lower case letters in the second column. Here, v stands for "not T"; y for "T or C" and b for "not A"; Each state is associated with one or more signal features.

Allowed Transitions, $\mathcal{T} \subseteq (\mathcal{Q} \times \mathcal{Q})$	Count	Segment	Content Features, $cont(q, q') \in \mathcal{C}$	Length Features, $len(q, q') \in \mathcal{L}$
`Begin` → `i`	1	intergenic	*intergenic*	*intergenic*
`Begin` → `End`	1	intergenic	*intergenic*	*intergenic*
`Begin` → `Tss, SL1, SL1Tis`	3	ntergenic	*intergenic*	*intergenic*
`Tss` → `SL1, SL1Tis`	2	5' UTR	*UTR*	*trans-exon*
`Tss, SL1,SL2` → `5'Don, Tis`	6	5' UTR	*UTR*	*5'UTR-exon*
`5'Don` → `5'Acc, AccTis`	2	intron	*intron*	*intron*
`5'Acc` → `5'Don, Tis`	2	5' UTR	*UTR*	*5'UTR-exon*
`SL1Tis, SL2Tis, AccTis,Tis` → `DonXx`	24	coding exon	*exon, frame0-2*	*first-exon*
`SL1Tis, SL2Tis, AccTis,Tis` → `Stop`	4	coding exon	*exon, frame0-2*	*single-exon*
`DonXx` → `AccXx`	6	intron	*intron*	*intron*
`AccXx` → `DonXx`	36	coding exon	*exon, frame0-2*	*middle-exon*
`Don0` → `AccStop`	1	intron	*intron*	*intron*
`AccXx` → `Stop`	6	coding exon	*exon, frame0-2*	*last-exon*
`Stop, AccStop` → `3'Don, PolyA, CleaveX`	8	3' UTR	*UTR*	*3'UTR-exon*
`3'Don` → `3'Acc`	1	intron	*intron*	*intron*
`3'Acc` → `3'Don`	1	3' UTR	*UTR*	*3'UTR-exon*
`3'Acc` → `PolyA, CleaveX`	3	3' UTR	*UTR*	*3'UTR-exon*
`PolyA` → `CleaveX`	2	3' UTR	*UTR*	*polyA-tail*
`Cleave2` → `SL1, SL1Tis, SL2, SL2Tis`	4	intercistronic	*intercistronic*	*intercistronic*
`Cleave1` → `Tss, SL1, SL1Tis`	3	intergenic	*intergenic*	*intergenic*
`Cleave1` → `i`	1	intergenic	*intergenic*	*intergenic*
`Cleave1` → `End`	1	intergenic	*intergenic*	*intergenic*
`i` → `i`	1	intergenic-long	—	—
`i` → `Tss, SL1, SL1Tis,End`	4	intergenic-long	—	—

Table 5.2.: In total there are 123 allowed transitions in the model applied in mGene. Each transition between two states corresponds to a genic segment type and is associated with a content feature and a length feature. If there is more than one out-going or one in-coming state, then all combinations are allowed and the count indicates the total number of transitions. Furthermore, `DonXx` and `AccXx` stand for any splice site in the coding regions and `CleaveX` denots both states, `Cleave1` and `Cleave2`.

5.3. The Gene Scoring Function

Recall that we use the scoring function $G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ to determine which segmentation $\mathbf{y}$ best fits a given genomic sequence $\mathbf{x}$. Here, $\mathbf{y}$ is again a sequence of segments, i.e. $\mathbf{y} = \mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_M$, where each segment is described by the start and stop position of the segment, as well as the state labels at these positions, i.e. $\mathbf{y}_j = (b_j, b'_j, y_j, y'_j)$. The gene scoring function $G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ is a sum over the individual feature contributions at all the considered positions of the segmentation:

$$G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{M+1} g_{\boldsymbol{\theta}}(\mathbf{x}, b_j, b'_j, y_j, y'_j) \ , \tag{5.1}$$

with

$$\begin{aligned} g_{\boldsymbol{\theta}}(\mathbf{x}, b, b', y, y') &= \sum_{s \in sig(y)} t_{\boldsymbol{\theta}}^{s,y} \left(f^s(\mathbf{x}, b) \right) \\ &+ \sum_{c \in cont(y,y')} t_{\boldsymbol{\theta}}^{c,(y,y')} \left(f^c(\mathbf{x}, b, b') \right) \\ &+ t_{\boldsymbol{\theta}}^{len(y,y')} (b' - b) \\ &+ T_{\boldsymbol{\theta}}(y, y') \ , \end{aligned} \tag{5.2}$$

where the four components arise from the signal, content, and length features as well as a contribution according to the transitions between neighboring states (y, y'). How much the individual features contribute to the total score is determined by the transformation functions $t_{\boldsymbol{\theta}}$ and the transition matrix $T_{\boldsymbol{\theta}}$. The parameters of these functions are learnt simultaneously during the training procedure, and are jointly summarized by the parameter vector $\boldsymbol{\theta}$ (see below). Piecewise linear functions are utilized that transform the individual features into score contributions: $t_{\mathbf{v}}(f) = \mathbf{v}^\top \psi(f)$ (see Section 3.3.3). Therefore, the feature values are expanded into histogram representations with predefined abscissa values (i.e. support points $\mathbf{u}$ of the PLiFs). The corresponding function values $\mathbf{v} = (v_k)_{k=1,...,K}$ are learnt during global training. For all signal and content transformations, we chose $K = 20$ support points, equally spaced on the range [0,1] of post-processed SVM outputs (see Section 4.1.4). For each length feature, the support points are set to be the K quantiles of the empirical distribution of corresponding lengths. Note, that in general, some features are applied to more than one state or segment, and their contribution to the total gene score might depend on which state or segment they are associated with. Therefore, the transformation of the score depends on both, the feature (e.g. $s \in sig(q)$, or $c \in cont(q, q')$) and the state q or state pairs (q, q'): $t_{\boldsymbol{\theta}}^{s,q}$ or $t_{\boldsymbol{\theta}}^{c,(q,q')}$,

respectively. In principle the number of transformation is given by:

$$N_{PLiF} = \sum_{q \in \mathcal{Q}} |sig(q)| + \sum_{(q,q') \in \mathcal{T}} (|cont(q,q')| + |len(q,q')|) \,. \tag{5.3}$$

However, in our applied model most features that are applied to more than one state are in fact all captured in a single histogram, and we learn only a single transformation. This reflects our belief, that, for example, the contribution of the *don* feature to the total score should be independent of whether it is seen at a frame 0 donor, `Don0` or a frame 1 donor, `Don1`. One exception is the acceptor feature, which is associated with the trans-acceptor states and treated differently than the acceptor feature associated with cis acceptor states. In our implementation the number of histograms amounts to:

$$N_{PLiF} = (|\mathcal{S}| + 1) + |\mathcal{C}| + |\mathcal{L}| = (8+1) + 8 + 13 = 30; \tag{5.4}$$

The parameter vector $\boldsymbol{\theta}$ will now be defined by concatenating the values $\mathbf{v}$ of all the PLiFs, and the entries of the transition score table T:

$$\boldsymbol{\theta} \;=\; \begin{pmatrix} (\mathbf{v}^{s,q})_{s \in sig(q), q \in \mathcal{Q}} \\ \\ (\mathbf{v}^{c,(q,q')})_{c \in cont(q,q'),(q,q') \in \mathcal{T}} \\ \\ (\mathbf{v}^{l})_{l \in \mathcal{L}} \\ \\ (T(q,q'))_{(q,q') \in \mathcal{T}} \end{pmatrix} \tag{5.5}$$

In total, this leads to a dimensionality of $\boldsymbol{\theta}$ of

$$N = N_{PLiF} \times K + |T| = 30 \times 20 + 123 = 723, \tag{5.6}$$

which is also equal to the number of parameters learnt during training. By increasing the number K of supporting points, PLiFs can approximate any kind of transformation as accurately as desired. However, as the number of parameters are linearly increasing with K, this leads to an increase in training time and space requirements and possibly to over-fitting problems. Because the parameters are learnt during global training, new features can readily be integrated to our system, without defining their weighting relative to other features in advance.

Similar to the parameter vector $\boldsymbol{\theta}$ that was constructed from the individual vectors $\mathbf{v}$, we can also design a joint feature map $\Phi(\mathbf{x}, \mathbf{y})$ from the mappings ψ, which are the histograms over the occurrences of feature values in the given segmentation, and additionally the occurances of transitions:

$$\Phi(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} (\phi^{s,q}(\mathbf{x}, \mathbf{y}))_{s \in sig(q), q \in \mathcal{Q}} \\ (\phi^{c,(q,q')}(\mathbf{x}, \mathbf{y}))_{c \in cont(q,q'),(q,q') \in \mathcal{T}} \\ (\phi^{l}(\mathbf{x}, \mathbf{y}))_{l \in \mathcal{L}} \\ (\phi^{(q,q')}(\mathbf{x}, \mathbf{y}))_{(q,q') \in \mathcal{T}} \end{pmatrix} \tag{5.7}$$

with

$$\phi^{s,q}(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{M+1} \mathbb{I}(y_j, q) \psi^{s,q}\left(f^s(\mathbf{x}, b_j)\right) \tag{5.8}$$

$$\phi^{c,(q,q')}(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{M+1} \mathbb{I}(y_j, q) \mathbb{I}(y'_j, q') \psi^{c,(q,q')}\left(f^c(\mathbf{x}, b_j, b'_j)\right) \tag{5.9}$$

$$\phi^{l}(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{M+1} \mathbb{I}(y_j, q) \mathbb{I}(y'_j, q') \psi^{l,(q,q')}\left(f^l(b_j, b'_j)\right) \tag{5.10}$$

$$\phi^{(q,q')}(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{M+1} \mathbb{I}(y_j, q) \mathbb{I}(y'_j, q'). \tag{5.11}$$

With this feature representation $\Phi(\mathbf{x}, \mathbf{y})$ and the concatenated parameter vector $\boldsymbol{\theta}$, $G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ can now be expressed as a linear function of the feature vector as required in Section 3.3.1: $G_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = \boldsymbol{\theta}^\top \Phi(\mathbf{x}, \mathbf{y})$. The mappings are also segmentally decomposable with each entity only depending on $\mathbf{x}$, and the considered segment of $\mathbf{y}$ (this accounts for the semi-Markovian property), thus allowing the use of efficient decoding algorithms, like the Viterbi method.

5.4. The Training Procedure

As a reminder, during training of the HSM-SVM, we solve an optimization problem, thereby tuning the parameters $\boldsymbol{\theta}$ such that for a set of N labeled training examples $\left\{(\mathbf{x}^n, \mathbf{y}^n)\right\}, n = 1, \ldots, N$, each true labeling $\mathbf{y}^n$ scores higher than all other possible labelings $\mathbf{y} \in \mathcal{Y}$ by a large margin, i.e. $G_{\boldsymbol{\theta}}(\mathbf{x}^n, \mathbf{y}^n) \gg G_{\boldsymbol{\theta}}(\mathbf{x}^n, \mathbf{y})$ (see Equa-

tion 3.35). We will now first describe the regularizer and subsequently the employed loss function in more detail.

Regularizer

For the regularizer $\Omega(\boldsymbol{\theta})$ we use a convex quadratic function of $\boldsymbol{\theta}$, which allows us to efficiently solve the training optimization problem (Equation 3.35). This implies that $\Omega(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{Q}\boldsymbol{\theta} + \boldsymbol{c}^\top\boldsymbol{\theta}$ for some positive-semidefinite symmetric matrix $\boldsymbol{Q}$ and some vector $\boldsymbol{c}$. However, instead of numerically providing $\boldsymbol{Q}$ and $\boldsymbol{c}$ it is much more enlightening to describe their structure, which we will do in the following.

The purpose of the regularizer is to avoid overfitting, to make the solutions insensitive to minor variations in the training data and thus to achieve good generalization performance by drawing the solution towards a parametrization that respects our prior beliefs. For our gene structure prediction algorithm we implement the belief that absolute parameter values should be small, which is equivalent to maximizing the score gap between the correct gene prediction and the best wrong prediction (analogously to margin maximization in SVMs see Section 3.2 and Equation 3.6 therein). Additionnally, we want to favor transformation functions $t_{\boldsymbol{\theta}}$ that are "smooth", or simple, therefore large variation in the piecewise linear functions are penalized. These prior beliefs about good solutions are implemented by adding terms to the objective to be minimized:

$$\begin{aligned}\Omega(\boldsymbol{\theta}) &= C_1^{small} \sum_{h=1}^{N_{PLiF}} \sum_{k=1}^{K} \left|v_k^h\right| + C_{trans,1}^{small} \sum_{(q,q')\in\mathcal{T}} |T(q,q')| \\ &+ \frac{1}{2} C_2^{small} \sum_{h=1}^{N_{PLiF}} \sum_{k=1}^{K} (v_k^h)^2 + \frac{1}{2} C_{trans,2}^{small} \sum_{t,t'\in\mathcal{T}} T(q,q')^2 \\ &+ C_1^{smooth} \sum_{h=1}^{N_{PLiF}} \sum_{k=1}^{K-1} \left|v_k^h - v_{k+1}^h\right| + \frac{1}{2} C_2^{smooth} \sum_{h=1}^{N_{PLiF}} \sum_{k=1}^{K-1} \left(v_k^h - v_{k+1}^h\right)^2 \quad .\end{aligned} \tag{5.12}$$

In this term we have included both, linear and quadratic regularization, where more gradually changing scoring functions are obtained by a quadratic regularization.[2] We have also introduced several hyper-parameters C, which allow to tune the strength of the individual components of the regularization term. This is done during model selection.

In addition, we know that PLiFs responsible for signal and content detectors, should be monotonically increasing when applied to the output or their associated

[2] As a technical note, we mention that while the absolute value function is are not linear, it can be implemented by linear operations when real values are represented by differences of non-negative values.

detectors. Therefore, we augment the set of constraints with the following inequality constraints:

$$v_k^h \leq v_{k+1}^h \quad \forall k = 1, \ldots, K-1, \forall h \in PLiF_{s,c} \ , \tag{5.13}$$

where $PLiF_{s,c}$ denots the set of PLiFs acting on signal and content detectors.

Loss Function

As discussed in Section 3.3.1, the loss function employed in the training algorithm should ideally correspond to the similarity measure used to determine the prediction performance. In gene finding, algorithms are usually judged on several different criteria, including an atomic metric on the nucleotide level, a segmental metric on the exon level and a more complex metric on complete genes or transcripts (see Section 2.6). Furthermore, due to uncertainties in the labels, some segments have higher importance than others (e.g. exons vs. UTRs) and should be predicted with higher confidence. In the design of the loss function we have tried to reflect these considerations, while still enabling efficient decoding via the Viterbi algorithm for constraint generation. In particular, the HSM-SVM approach allows to introduce a segmental loss term (compare Section 3.3.4), where we have used a variant of the Hamming distance, which is the weighted sum of the number of mismatching segments (see Figure 5.3). Here, the joint segmentation is considered, which respects all segment boundaries from both compared segmentations, as indicated in the figure. We do not penalize mismatches between intergenic and intercistronic regions as these differences are also not considered during evaluation; Likewise a confusion of 5' UTR regions with "trans-exon regions" and 3' UTR regions with "polyA regions" are ignored.

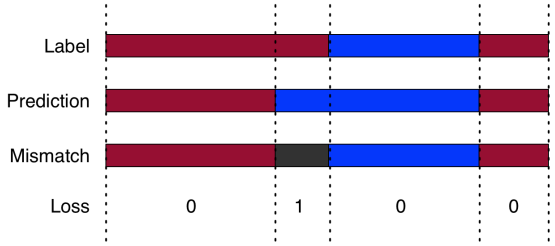


Figure 5.3.: Schematic representation of segment loss. An example with only two segment types (red and blue) is considered. The mismatching segment (gray) incurs a loss that is independent of the length of the segment.

In a first version, all mismatched segments (with the above exceptions) had the same weight, and therefore the Hamming loss was essentially the count of mis-

matched segments. However, in our latest version, we have increased the penalty for mismatches between intergenic / intercistronic regions and other regions and could thereby increase the prediction performance.

We have further introduced a tolerance for the transcript boundaries. More specifically, if segments disagree by no more than 10 nucleotides at a TSS, polyA, or cleavage site, they are not considered to mismatch. This is done to account for the evident uncertainty about the exact location of these signals, be it due to insufficient knowledge or even due to inherent biological variability.

So far, we refrained from the inclusion of a *gene loss*, which would probably lead to an increased gene or transcript accuracy. However, this would either require to introduce additional states or prohibit the use of the Viterbi algorithm in its present form for decoding and would thus lead to a significant higher run-time.

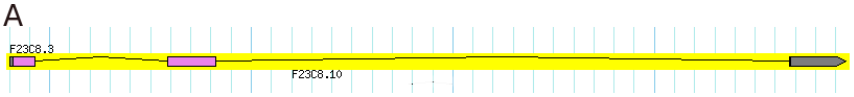
5.5. Implementation in mGene

The gene structure prediction system was implemented in mGene using Matlab and can also be run under the free software Octave. Performance-critical components, like the dynamic program, are written in C++. This part is integrated code of the freely available machine learning toolbox shogun [149], which offers interfaces to MatLab, Octave, and other programming languages.

The gene structure training of mGene starts by processing the labelled training examples. In particular, it converts gene annotation files such as GFF files into mGene specific segmentations $\mathbf{y}$, which also considers coding frame and other information, that have to be derived from the GFF file, see Figure 5.4.

The subsequent training procedure follows, in principle, the algorithm presented in Table 3.1, however, for a more efficient computation we introduced several modifications, see Table 5.3: (1) As input we use not only sequence label pairs $(\mathbf{x}^n, \mathbf{y}^n)$, but also the precomputed sequences of signal and content detectors $(\mathbf{FSX}^n, \mathbf{FCX}^n)$ at the candidate positions pos^n. (2) For a better initiation of the parameter vector $\boldsymbol{\theta}$, we generate a small set of initial negative examples by slightly changing the structure of annotated genes without violating the biological rules, for example by removing some exons whose lengths are multitudes of 3. With these sets $\tilde{\mathcal{Y}}^n$, we solve the QP for the first time, generating an initial parametrization $\tilde{\boldsymbol{\theta}}$. Subsequently, we start to alternate between constraint generation via Viterbi decoding and QP-solving. (3) For faster convergence, we update the parametrization not only after a complete cycle through all examples N, but whenever a predefined number N' of examples have been decoded. (4) The training is terminated if no more margin violators are found, the change in the objective is small, or a maximal number of iterations is reached.

A



B

I	Coding_transcript	gene	2438802	2442949	.	+	.	Alias=F23C8.3;ID=Gene:WBGene00017732;Name=WBGene00017732
I	Coding_transcript	mRNA	2438802	2442949	.	+	.	Alias=WBGene00017732;prediction_status=Confirmed; ID=Transcript:F23C8.3;Parent=Gene:WBGene00017732;Name=F23C8.3
I	Coding_transcript	five_prime_UTR	2438802	2438815	.	+	.	ID=16180;Parent=Transcript:F23C8.3;Name=F23C8.3
I	Coding_transcript	CDS	2438816	2438923	.	+	.	ID=16181;Parent=Transcript:F23C8.3;Name=F23C8.3
I	Coding_transcript	CDS	2439584	2439821	.	+	.	ID=16182;Parent=Transcript:F23C8.3;Name=F23C8.3
I	Coding_transcript	CDS	2442668	2442672	.	+	2	ID=16183;Parent=Transcript:F23C8.3;Name=F23C8.3
I	Coding_transcript	three_prime_UTR	2442673	2442949	.	+	.	ID=16184;Parent=Transcript:F23C8.3;Name=F23C8.3
I	Coding_transcript	exon	2438802	2438923	.	+	.	ID=66517;Parent=Gene:WBGene00017732
I	Coding_transcript	exon	2439584	2439821	.	+	.	ID=66518;Parent=Gene:WBGene00017732
I	Coding_transcript	exon	2442668	2442949	.	+	.	ID=66519;Parent=Gene:WBGene00017732
I	Coding_transcript	intron	2438924	2439583	.	+	.	confirmed_est=yk1035a05.3;ID=66520;Parent=Gene:WBGene00017732
I	Coding_transcript	intron	2439822	2442667	.	+	.	confirmed_est=yk1035a05.3;ID=66521;Parent=Gene:WBGene00017732

C

pos	...	2438802	2438815	2438923	2439584	2439821	2442668	2442672	2442949	...
state		2	3	6	14	7	15	21	24	

Figure 5.4.: Example gene F23C8.3 on the forward strand of chromosome I A) as displayed on wormbase. B) as part of the GFF file and C) as parsed segmentation including a sequence of positions and corresponding states, with the following ids: 2:Tss, 3:Tis, 6:Don0, 14:Acc0, 7:Don1a, 15:Acc1a, 21:Stop and 24:Cleave.

To solve the QP, the commercial optimization software packages CPLEX[3] or Mosek[4] are used. It is done on a single computer using as many CPUs as supported by the QP-solver.

The decoding procedure is the most time critical step during gene structure training. Here, mGene supports parallel computing. We have implemented an extended Viterbi algorithm as shown in Table 5.4. It differs from a simple Viterbi algorithms in the following points: (1) As input we take a sequence label pair $\mathbf{x}, \mathbf{y}$ and the precomputed sequences of signals and contents **FSX**, **FCX** at candidate positions *pos*. Note, that $\mathbf{x}$ is of length L, $\mathbf{y}$ of length $M < L$ and *pos* of length $M < P < L$. In contrast to a general version of the Viterbi algorithm, which considers every position in $\mathbf{x}$, our implementation only considers candidate positions included in *pos*. Therefore, the run-time can be reduced by starting with a small set of very likely candidate positions (with large values in **FSX**). Additionally, only positions p are considered for a given state q, that have a corresponding signal score $\mathbf{FSX}_{p,q} \neq -\infty$. (2) To exploit the sparsity of the model, we only consider a sub-set of states $\mathcal{T}_q \subset \mathcal{Q}$ as potential predecessor states for a given state q. (3) Potential positions for a predecessor can be up to D_{max} away from the current position, thus implementing the

[3]http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/
[4]http://mosek.com/

Input: N labelled training examples with precomputed signals and contents:
$(\mathbf{x}^n, \mathbf{y}^n, pos^n, \mathbf{FSX}^n, \mathbf{FCX}^n, \forall n)$
Initialization:
generate small sets of false labelings $\tilde{\mathcal{Y}}^n$ for each training example
compute initial parametrization (Equation 3.35) with $\mathcal{Y}^n = \tilde{\mathcal{Y}}^n, \forall n$:
$\tilde{\boldsymbol{\theta}}, \tilde{\xi}^n \leftarrow solve\_QP(\mathbf{x}^n, \mathbf{y}^n, \tilde{\mathcal{Y}}^n, \forall n)$

repeat
  repeat
    for N' training examples

      use the Viterbi algorithm (Table 5.4) for decoding:
      $\mathbf{y}^{n*} \leftarrow solve\_DynProg(\mathbf{x}^n, \mathbf{y}^n, pos^n, \mathbf{FSX}^n, \mathbf{FCX}^n, \tilde{\boldsymbol{\theta}})$

      add maximal margin violator to the active set:
      if $\mathbf{y}^{n*} \neq \mathbf{y}^n \wedge \Delta G(\mathbf{y}^n, \mathbf{y}^{n*}) \leq \ell(\mathbf{y}^n, \mathbf{y}^{n*}) - \tilde{\xi}^n$
        $\tilde{\mathcal{Y}}^n \leftarrow \tilde{\mathcal{Y}}^n \cup \mathbf{y}^{n*}$
      end if

    end for

    solve the intermediate training problem (Equation 3.35)
    with the updated set of constraints: $\mathcal{Y}^n = \tilde{\mathcal{Y}}^n, \forall n$
    $\tilde{\boldsymbol{\theta}}, \tilde{\xi}^n \leftarrow solve\_QP(\mathbf{x}^n, \mathbf{y}^n, \tilde{\mathcal{Y}}^n, \forall n)$

  until all N training examples are considered
until no margin violators OR change in objective small OR max number of iterations

Table 5.3.: The HM-SVM training algorithm as implemented in mGene.

semi-Markovian extension. (4) We enforce that the potential segment is compatibel with ORF constraints, if the considered segment is a coding segment, i.e. its length has to be a multitude of 3, and it is not allowed to contain any in-frame stop codons. (5) Signal, content, and length feature contributions are transformed into score contributions with there respective PLiFs. (6) For constraint generation, we want to compute the path that maximizes the sum of the score and a loss term: $\tilde{\mathbf{y}}^{n*} = \text{argmax}_{\mathbf{y} \in \mathcal{Y}^n}(G_{\tilde{\boldsymbol{\theta}}}(\mathbf{x}, \mathbf{y}) + \ell(\mathbf{y}^n, \mathbf{y}))$. Therefore, we need to compute the loss for any potential segment.

With these modifications the run time is $\mathcal{O}(|\mathcal{Q}| \times T_{max} \times P \times D_{max})$, where T_{max} is the maximal number of incoming edges for any state $q \in \mathcal{Q}$. The maximal lookback D_{max} is particularly problematic, as it depends on the maximal segment length, which can be very large for intergenic regions (> 100 kbp). We therefore introduced an auxiliary state, `i`, into our model (Figure 5.2. D), that interrupts long intergenic segments (> 20 kbp).

```
1   Input: sequence label pair with precomputed signals and contents, parameter vector
2       (x, y, pos, FSX, FCX, θ)
3
4   precompute list of stop codons:
5   pos_stop ← stop_codons(x)
6
7   apply PLliFs to signals:
8   for p = 1 ... P
9       for q = 1 ... |Q|
10          FSX_{p,q} ← t^s_θ(FSX_{p,q})
11      end for
12  end for
13
14  Initialization: V, T_1, T_2
15  for p = 1 ... P
16      for q = 1 ... |Q|
17          if FSX_{p,q} = −∞
18              break
19          end if
20          get allowed predecessor states for state q: T_q
21          compute segment loss, for all states q' ∈ T_q and all positions, p − 1 ... p − D_max:
22          Δ ← comp_loss(y, p, D_max, T_q)
23          for q' ∈ T_q
24              while pos(p) − pos(p − d) ≤ D_max(q, q')
25                  if FSX_{q',p−d} = −∞
26                      break
27                  end if
28                  ok ← check_ORF(x, q, q', pos(p), pos(p − d), pos_stop)
29                  if !ok
30                      break
31                  end if
32                  trans ← T(q, q')
33                  len ← t^{l(q,q')}_θ(d)
34                  cont ← t^{c(q,q')}_θ(FCX_{p,q} − FCX_{p−d,q'})
35                  loss ← Δ(d, q')
36                  score ← V(p − d, q') + trans + len + cont + loss
37                  if score > V(p, q)
38                      V(p, q) ← score
39                      T_1(p, q) ← q'
40                      T_2(p, q) ← p − d
41                  end if
42              end
43          end for
44          V(p, q) ← V(p, q) + FSX_{p,q}
45      end for
46  end for
47
48  y* ← trace_back(T_1, T_2)
```

Table 5.4.: Extended Viterbi-like decoding in mGene. We assume one signal per state. The vector $\boldsymbol{\theta}$ contains the parameters of the PLiFs ($t_{\boldsymbol{\theta}}$) and transition table T. See main text for further explanations.

6. Applications and Results

This chapter contains several applications using the methods described above. It will start with the genome-wide prediction of splice sites. For this first study we chose five organisms that are evolutionarily far apart, ranging from plants to humans. Subsequently, we employed the complete gene finding system mGene. Here, we concentrated on a single genus of roundworms. We initially participated in a competition to annotate sub-regions of the *C. elegans* genome and the results of this competition are presented. We then continued to generate a genome-wide annotation for *C. elegans* and four other nematodes. The insights gained from these efforts will also be summarized in this chapter.

6.1. Genome-wide Splice Site Prediction in Different Organisms

In a first set of experiments, we applied our SVM-based splice site prediction suite to the genomes of mouse-ear cress (*Arabidopsis thaliana*), worm (*Caenorhabditis elegans*), fruit fly (*Drosophila melanogaster*), zebrafish (*Danio rerio*), and human (*Homo sapiens*). The design of the genome-wide predictor is based on our preliminary studies as described in Section 4. The results have been published in [148] and this section contains modified excerpts from this paper.

6.1.1. Experimental Protocol

Generating Labelled Sets for Splice Sites

We first generated genome-wide labelled data sets for the five organisms. As our large-scale learning methods allow us to use millions of training examples, we included all available EST information from the dbEST [17]. We additionally used EST and cDNA sequences available from other organism specific databases such as wormbase [73] for worm, (see the Appendix B for more details).

We used the method described in Section 4.1.1 to align ESTs and cDNA sequences against the genomic DNA (releases WS170, dm5, ath1, zv6, and hg18, for worm, fly, cress, fish, and human, respectively), and to generate labelled acceptor and donor splice site examples. We arrived at training data sets of considerable size containing

sequences of sufficient length (see Table B.2). For fish and human, the training datasets were sub-sampled to include only 1/3 and 1/5 of the negative examples, leading to a maximal training set size of 9 million sequences for human donor sites.

Training and Prediction

For a subsequent use in a gene finder system we aimed at producing unbiased predictions for all candidate splice sites. Therefore, we employed nested five-fold cross-validation (see Section 4.1.3). We applied two support vector machine approaches, one with the weighted degree kernel (WD) and the other one with the weighted degree kernel with shifts (WDS). For comparison, we also applied the Monte Carlo based classifier (MC) in the same manner to the data sets. The determined parameter sets are can be found on the project web site[1] and in [145]. For the SVM based classifiers, we additionally estimated posterior probabilities to obtain interpretable and comparable scores for the outputs of the different SVM classifiers as described in Section 4.1.4.

6.1.2. Performance of Genome-wide Splice Site Predictor

We compared the predicted splice sites with the true labelled data sets and calculated the auPRC scores on each of the five test sets for a given organism. Table 6.1 shows the averaged scores for all five organisms comparatively and Figure B.1 (in the Appendix) displays the corresponding precision recall curves. Confirming our evaluations in the pilot studies (see Chapter 4), kernel methods outperform the MC methods in all eight classification tasks. For worm, fly, and cress, the improvement in the performance accuracy for the SVM in comparison to MC lies in a similar range of 4-10% (absolute), both for donor and for acceptor tasks. However, for fish, and especially for human the performance gain is considerably higher. For human, MCs only achieve 16% and 25% auPRC scores, whereas WDS reaches 54% and 57% for acceptor and donor recognition, respectively. The severe decrease in performance from worm to human for all classification methods in the auPRC score can partially be explained by the different fractions of positive examples observed in the test set. However, a weaker decline can also be observed in the auROC scores (also Table 6.1) that are independent of the class skew (e.g. for acceptor sites from 99.6% on worm to 96.0% on human for MC, and from 99.8% to 97.9% for WDS).

[1] `http://www.fml.mpg.de/raetsch/projects/splice`

	Worm		Fly		Cress		Fish		Human	
	Acc	**Don**	**Acc**	**Don**	**Acc**	**Don**	**Acc**	**Don**	**Acc**	**Don**
MC										
auROC(%)	99.62±0.03	99.55±0.02	98.78±0.10	99.12±0.05	99.12±0.03	99.44±0.02	98.98±0.03	99.19±0.05	96.03±0.09	97.78±0.05
auPRC(%)	92.09±0.28	89.98±0.20	80.27±0.76	78.47±0.63	87.43±0.28	88.23±0.34	63.59±0.72	62.91±0.57	16.20±0.22	24.98±0.30
WD										
auROC(%)	99.77±0.02	99.82±0.01	99.02±0.09	99.49±0.05	99.37±0.02	99.66±0.02	99.36±0.04	99.60±0.04	97.76±0.06	98.59±0.05
auPRC(%)	95.20±0.29	95.34±0.10	84.80±0.35	86.42±0.60	91.06±0.15	92.21±0.17	85.33±0.38	85.80±0.46	52.07±0.33	54.62±0.54
WDS										
auROC(%)	99.80±0.02	99.82±0.01	99.12±0.09	99.51±0.05	99.43±0.02	99.68±0.02	99.38±0.04	99.61±0.04	97.86±0.05	98.63±0.05
auPRC(%)	95.89±0.26	95.34±0.10	86.67±0.35	87.47±0.54	92.16±0.17	92.88±0.15	86.58±0.33	86.94±0.44	54.42±0.38	56.54±0.57

Table 6.1.: Performance evaluation on the genome-wide data sets for worm, fly, cress, fish, and human. Displayed are auROC and auPRC scores for acceptor and donor recognition tasks as archived by the MC method and two support vector machine approaches, one with the weighted degree kernel (WD) and one with the weighted degree kernel with shifts (WDS). Shown are averaged values with standard deviation over the five different test partitions.

6.1.3. Summary and Discussion

We generated new splice site data sets for worm, fruit fly, mouse-ear cress, zebrafish, and human. These data sets contain sufficiently long sequences and — for human — as many as 9 million training examples. Based on previous work on large scale kernel learning by Sonnenburg *et al.* [145, 147], we were able to train SVM classifiers on these rather big data sets.

The classification task on the human genome seems to be a considerably more difficult problem than the same one on the worm genome. We may speculate that this can be partially explained by a higher incidence of alternative splicing in the human genome. These sites usually exhibit weaker consensus sequences and are therefore more difficult to detect. Additionally, they often lead to mislabeled examples in training and testing sets. Finally, it might also be due to the protocol used for aligning the sequences, which may generate more false splice sites in human than in other organisms. This hypothesis is supported by the fact that the performance significantly increases if one only considers cDNA confirmed genes (data not shown).

However, it is interesting to observe that — while there is still room for improvement — our SVM based approach can particularly unfold its strength on these more difficult problems and the biggest difference between the methods is measured for acceptor and donor recognition on human DNA.

To facilitate the use of our classifiers for other studies, we provide whole genome predictions for the five organisms. The predictions, data sets and the stand-alone prediction tool are available for download on the website `http://www.fml.mpg.de/raetsch/projects/splice`.

6.2. mGene at the nGASP Challenge

At the end of 2006, the international wormbase consortium launched a genome annotation competition to assess the quality of existing gene finders. One major incentive of this nematode genome annotation assessment project (nGASP) was, to determine the gene finding systems that are best suited for the annotation of other *Caenorhabditis* genomes. The organizers created strictly controlled conditions, precisely specifying any data that was allowed to be used. We took this opportunity to apply our methods to the provided data and accordingly generated gene predictions for the nematode *Caenorhabditis elegans*. The following objective analysis of the submissions by the consortium showed that our initial predictions achieved very high accuracy, when compared with the other participants [34]. After the competition, we further enhanced our methods and I applied the new system, mGene, to the nGASP data to generate new, improved predictions. The results of the initial version as well as the improved system were presented in our paper [141], which is also partly included into the following sections.

6.2.1. The nGASP Setting

Training and Test Regions

For training, participants were provided with 10 representative 1Mb regions from the genome of *C. elegans* (sequence version WS160, see the Appendix C for further information). The organizers also provided repeats found by applying RepeatMasker [144]. For these training regions, an annotation was supplied consisting of 432 confirmed and 1461 unconfirmed (i.e. predicted or partially confirmed) genes.

The test set also consisted of 10 1Mb windows, chosen with the same criteria as the training set. Test and training set windows showed no overlap. No annotation was provided for the test regions. Sequences as well as annotation files can be downloaded from [187] as FASTA, or GFF files, respectively, and more information on these data sets can be found in [34]

Additional Data and Categories

In addition to the DNA sequences and the partial annotation, which are the basic inputs for any gene finding system, some additional data was provided. Depending on the utilized data, participants could submit to the following nGASP categories:

Category 1 — *ab initio* systems, allowing only the genomic sequence, the repeats found in the training regions and the partial annotation as input.

Category 2 — systems that utilize conservation information; In this category the DNA sequence for *C. briggsae* and for *C. remanei* could additionally be used, as well as a multi-genome alignment between *C. elegans*, *C. briggsae* and *C. remanei*,

Category 3 — systems that make use of alignments of proteins, ESTs, and cDNA sequences to the genome; For this category the organizers provided the amino acid sequences of 42,496 proteins that have BLAST [3] matches to the test or training regions. Additionally, the nucleotide sequences of 20,141 *C. elegans* ESTs/cDNAs that have BLAT matches to the test or training regions were provided, as well as the coordinates of the BLAST and BLAT matches in the test and training regions.

Category 4 — combiners, systems that benefit from all available information including predictions submitted for Categories 1-3. Additionally, the training set was enlarged for this category: The annotation of 386 confirmed isoforms of 242 genes in the 5' halves of each of the test regions was added to the training set. Because of this, combiners were evaluated on the 3' halves of the test regions.

More details can be found in the Appendix C and [34].

6.2.2. Experimental Protocol

Curation of Annotation and Generation of Labelled Training Set

We first processed the provided annotation of the 10 training regions by inferring missing UTRs and poly-adenylation sites as well as *trans*-splicing sites in a semi-automatic manner (for details see Section 4.2). We further corrected minor inconsistencies and excluded genes with non-consensus splice sites or missing reading frames[2]. Finally, we grouped genes with a distance less than 250nt from each other into operons. Based on the augmented annotation we sub-divided the training regions into 747 consecutive sub-regions with at least two genes each, to retain complete intergenic regions.

From the sub-regions, we generated labeled data for training and model selection of signal and content detectors according to the procedure described in Section 4.2. For instance, for every annotated splice site we generated a positive example and for every other consensus position a negative example. For signals without consensus we considered all other positions, but sub-sampled to speed up training.

For training of the HSM-SVM based gene structure prediction tool, we generated complete, non-overlapping segmentations $\mathbf{y}^n$ for each sub-region from the annotated

[2]The resulting annotation can be obtained from the project website: http://mgene.org/datasets

genes as displayed in Figure 5.4 in Section 5.5. If a gene had more than one annotated isoform, individual segmentations were generated for each transcript.

Submitted (Developmental) Versions of mGene

We participated in Categories 1-3 with developmental versions of mGene, which we call mGene.init (dev), mGene.multi (dev), and mGene.seq (dev), respectively.

Category 1 Submissions For our *ab initio* predictions we only used the DNA sequence and the partial annotation. The additionally provided repeats were not considered.

Signals were trained in a five-fold cross validation scheme. On the test regions we randomly chose one of the five obtained models to make a prediction for each candidate position. The predictions were then post-processed according to the transformations that were learnt on the corresponding set for tuning (see Sections 4.1 for further details).

For the training of the content detectors we also used a five-fold cross validation scheme, however, the predictions were not pre-computed and the learnt content models were given to the gene structure prediction system that computed content predictions on the fly. More details on the content detectors are given in Section 4.3

The produced signal predictions on the sub-regions, as well as the content models were subsequently used along with the processed annotation for the HSM-SVM training We ran the algorithm on 500 sub-regions to obtain optimized parameters $\boldsymbol{\theta}$. The remaining 247 sub-regions were used for tuning of the hyper-parameters, see Section 5.4. The optimal hyper-parameter set is given in the Appendix C. The resulting final model was used to predict genes in the 10 test regions. These predictions were submitted to the organizers of the nGASP competition (referred to as mGene Category 1 version 2 on the nGASP web page and as mGene.init (dev) in [141]).[3]

Category 2 Submissions: For nGASP Category 2 we implemented a first version, mGene.multi (dev), that exploits multiple genome alignments as additional features. Therefore, for each genomic position, the corresponding column of the alignment was summarized into a discrete conservation score. This information was subsequently used in two different ways: First, we have improved the signal sensors by combining the DNA sequence kernels with additional kernels that act on the corresponding conservation scores in the same windows. Appropriate kernels (linear and/or WD

[3]We had three submissions in the *ab initio* category: (a) an initial submission mGene Category 1 version 1 with suboptimal parameters, (b) mGene Category 1 version 2 with improved parameters, and (c) mGene Category 1 version 3 as in (b) but including predictions of alternative splicing (description goes beyond the scope of this work).

kernels) where chosen during model selection for the individual signals. Second, we have designed additional conservation sensors with spectrum kernels that act on the conservation scores within a given segment. For further details, see Section 4.4.

Category 3 Submissions: For mGene.seq (dev) we used EST and cDNA sequences to support our gene predictions. However, before the closure of the competition we had not enough time to consider protein sequences. For the alignment of the EST and cDNA sequences, we employed the procedure described in Section 4.1.1. We then added a heuristic prior to the gene structure prediction to boost (suppress) signal predictions that agreed to (conflicted with) the alignments (more details are given in Section 4.4). Note that this extension did not require retraining.

Improved Versions of mGene

After the nGASP competition, we have continued to refine mGene and have used it to obtain predictions according to the rules in Categories 1 & 3 (denoted by mGene.init and mGene.seq). We have thereby not consciously used any information that was not available during the actual competition. Also, we have not taken advantage of experiences gained by successful competitors. Before the deadline, we only managed to train the system a single time and did not have the time for sufficient tuning using the available data. After the deadline, we automated many steps that were previously performed manually. Many of the changes also included bug fixes or a cleaner implementation. We made several improvements to the loss function, further optimized the tuning of the recognition of gene boundaries and used *all* 747 regions for training the HSM-SVM. Finally, for the Category 3 prediction, we also consider protein alignments.

Evaluation

The submitted predictions on the test set were compared against an annotation consisting of two sets of genes: A set of all 605 fully confirmed isoforms from 493 different genes in the test set, denoted with **ref1** and a second set that additionally contained predicted or partially-confirmed isoforms, resulting in 2250 isoforms from 1956 genes, denoted with **ref2**. These sets were only considered after the competition was closed for the purpose of evaluation. The quality assessment was performed by an independent team from the wormbase consortium. They used the set of highly confirmed genes (**ref1**) for sensitivity assessment, while specificity was determined on the broader set (**ref2**). The performance metrics employed by the official evaluation team were similar to those used in the EGASP competition [71]: Sensitivity and specificity were determined on the level of nucleotides, exons, transcripts, and genes while considering coding regions only (see also 2.6). The results of the evaluation

were published in [34]. Additionally, we have followed their assessment strategy to write our own evaluation procedure.

6.2.3. Quality Assessment of Gene Predictions

Including mGene, there were 47 submitted prediction sets from seventeen groups worldwide. We have downloaded the respective gene catalogs from the nGASP website [158] and evaluated them as well as the improved mGene predictions with our own evaluation routine. Some of the numbers slightly deviate from the official nGASP evaluation, however the ranking of the best performing methods is mostly unchanged and these differences do not affect the overall conclusions. (The official results from the competition organizers can be found in the Appendix C). In the following paragraphs, I will only report the results for the top performing methods, which makes differences in prediction accuracy appear small. However, when all participating methods are compared, performance margins can reach 20 percentage points on gene level [e.g. when comparing mGene.init (dev) to SNAP, 86].

The results, summarized in Table 6.2 and in [34] show mGene.init (dev) to be more specific than all competing methods (including Fgenesh [127], Augustus [152], and Craig [16]) in Category 1 (*ab initio*) on all four levels. The second most specific gene finder was Craig, which had significantly lower sensitivity than mGene. Fgenesh and Augustus achieved higher sensitivity, albeit at the cost of lower specificity. With respect to the average of sensitivity and specificity, mGene.init (dev) was the best among all Category 1 submissions on all levels except for the gene level, where it was second to Augustus.

The fully developed version mGene.init makes predictions that are more accurate than any competing submission to nGASP in Category 1. Figure 6.1 A-D illustrates the improvements in sensitivity and specificity of mGene.init compared to the best submissions in Category 1. Note that these improvements correspond to substantial reductions of the error rate (defined as $1-\frac{Sn+Sp}{2}$) relative to the second best methods (in parentheses) on each level, namely 8.9% for nucleotides (Craig), 13.4% for exons (Fgenesh), 8.7% for transcripts (Augustus), and 4.1% for genes (Augustus).

In Category 2, mGene.multi (dev) outperformed all competing methods including N-Scan [69] and Eugene [56] for all evaluation criteria. It is noteworthy that despite the use of multi-genome alignments as additional information, Eugene and N-Scan were not able to achieve the performance of the best *ab initio* gene finders. Only mGene.multi (dev) managed to improve slightly over the best methods in Category 1.

For predictions submitted in Category 3, the development version mGene.seq (dev) only utilized EST and cDNA alignments, but not the provided protein sequences. Nevertheless, among the seven competitors in this category, only Augustus and Fgenesh were able to achieve higher accuracy (defined as average of sensitivity and

Cat.	Method	Nucleotide			Exon			Transcript			Gene		
		Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$
1	mGene.init	96.78	90.87	93.83	85.11	80.17	**82.64**	49.59	42.25	**45.92**	60.73	42.25	**51.49**
1	mGene.init (dev)	96.85	91.59	**94.22**	84.17	78.63	81.40	44.30	38.69	41.50	54.25	40.13	47.19
1	Craig	95.54	90.92	93.23	80.17	78.15	79.16	35.70	35.44	35.57	43.72	35.44	39.58
1	Eugene	93.96	89.47	91.72	80.28	73.00	76.64	49.09	28.19	38.64	60.12	28.19	44.16
1	Fgenesh	98.20	87.11	92.65	86.37	73.55	79.96	47.11	34.11	40.61*	57.69	34.11	45.90
1	Augustus	97.01	89.01	93.01	86.12	72.55	79.34	52.89	28.64	40.77*	64.37	34.47	49.42
2	mGene.multi (dev)	97.70	90.91	**94.31**	85.81	78.30	**82.06**	51.24	40.87	**46.05**	62.68	43.83	**53.25**
2	N-Scan	97.39	88.07	92.73	83.51	70.83	77.17	39.17	27.69	33.43	48.07	28.39	38.23
2	Eugene	96.23	86.48	91.36	82.75	72.82	77.79	50.25	30.19	40.22	61.66	31.36	46.51
3	mGene.seq	98.83	90.09	94.46	92.21	83.45	**87.83**	65.45	52.49	58.97	80.16	52.44	**66.30**
3	mGene.seq (dev)	98.71	91.88	95.30	90.99	80.61	85.80	58.68	45.93	52.30	71.66	47.80	59.73
3	Gramene	98.20	95.42	**96.81**	88.45	71.76	80.11	44.96	19.13	32.04	52.63	28.60	40.62
3	Fgenesh++	97.57	89.70	93.64	90.43	80.93	85.68	65.62	52.91	**59.27**	78.54	52.07	65.30*
3	Augustus	98.96	90.52	94.74	92.45	80.20	86.33	69.09	46.45	57.77	80.97	49.95	65.46*

Table 6.2.: Comparison of top performing gene finding systems that participated in the nGASP challenge [34]: Shown are sensitivity (Sn), specificity (Sp), and their average (each in percent) on nucleotide, exon, transcript, and gene level in Categories 1-3 (if several submissions were made for one method, we chose the version with the best gene level average of sensitivity and specificity). The predictions of mGene.init and mGene.seq were generated after the deadline but according to the rules of the nGASP challenge. The result of the best performing method within a category and according to each of the evaluation levels is set in bold face. For reference, we also provide the results of the evaluation by the nGASP team [34] in the Appendix C. The numbers slightly deviate on the transcript and gene level due to minor differences in the evaluation criteria. These differences, however, do not change the ranking except in two cases, where the performances are very close together (relevant results are marked with an asterisk).

specificity) than mGene.seq (dev). The current prediction system, mGene.seq, exploits EST, cDNA, and protein sequence alignments to improve gene predictions, which are now most accurate according to the evaluation on exon and gene level—and only second to Fgenesh on transcript level.

6.2.4. Analysis of Predicted Signals

Performance of the Individual mGene Signal Detectors

To assess the strengths of the different gene finders and eventually to construct an improved system, it would be interesting to compare the accuracy of their respective sub-models. However, this intermediate data was only available for our own system mGene.

We assessed the performance of the individual signal detectors, to verify the validity of these detectors devised for mGene. Note, that, as we have employed a five-fold cross validation scheme, the determined error on the nGASP training re-

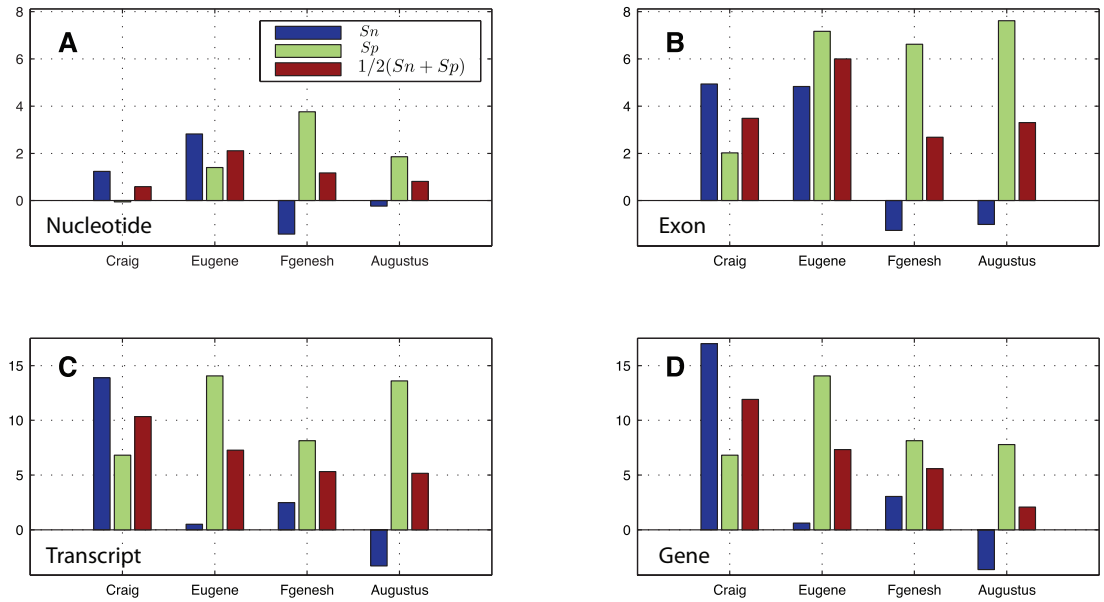


Figure 6.1.: Improvement of mGene.init *ab initio* predictions on several evaluation levels: A) nucleotide, B) exon, C) transcript, and D) gene (each restricted to coding regions). mGene.init's predictions are compared to the predictions of the best submissions in Category 1: Craig, Eugene, Fgenesh, and Augustus. Shown are differences of percent values for sensitivity (Sn; blue), specificity (Sp; green), and their average (red).

gions is indeed an out-of-sample test error. The accuracy of each signal detector is quantified by comparing it to the respective annotation. In Table 6.3, I report the auPRC and the auROC (see 2.6 for further details). Due to the sub-sampling of the negative examples, positive examples may be over-represented in both training and evaluation as compared to the real-life situation of predicting on genomic sequences (in particular for transcription start and cleavage sites). We therefore also report the fraction of positives in the generated sets of examples.

Performance of Gene Prediction Systems on Individual Signals

While a performance estimation of the individual sensors is very informative for the design and improvement of gene finders, it is of course, most interesting to observe how the individual components act in concert when integrated into a gene finding system. In addition to the global evaluation of predicted gene structures, we compared the prediction systems with respect to individual sequence signals. We performed such comparisons on the *ab initio* systems as indirect assessment of the

	tss	*trans*	*tis*	*stop*	*acc*	*don*	*polyA*	*cleave*
Ratio of positives	0.01	0.03	0.02	0.01	0.02	0.01	0.01	0.01
auPRC	0.57	0.33	0.31	0.30	0.88	0.82	0.14	0.24
auROC	0.95	0.89	0.88	0.93	0.99	0.99	0.87	0.92

Table 6.3.: Evaluation of individual signal detectors: transcription start sites (*tss)*, trans-splice sites (*trans*), translation start sites (*tis*), translation termination sites (*stop*), acceptor splice sites (*acc*), donor splice sites (*don*), poly-adenylation sites (*polyA*) and cleavage sites (*cleave*). Given is the ratio of positive examples among the labeled examples and the areas under the precision-recall-curve (auPRC) and under the receiver operating characteristic curve (auROC) for predictions on the nGASP training regions.

underlying sequence features. While Table 6.3 indicates that our signal detectors work reasonably well when applied individually, the accuracies further increase when the surrounding context (e.g. information from other signals or contents), is also taken into account.

	tss			*tis*			*acc*			*don*			*stop*			*cleave*		
Method	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$
mGene.init	32.8	11.5	**22.2**	71.1	65.4	**68.2**	88.9	85.54	**87.2**	90.0	85.7	**87.9**	80.3	76.4	**78.4**	29.5	15.7	22.6
mGene.init (dev)	28.4	10.7	19.5	66.6	67.5	67.1	88.7	85.02	86.8	89.8	84.9	87.3	76.8	79.2	78.0	15.5	10.1	12.8
Craig	16.7	7.0	11.9*	57.1	65.2	61.2	85.8	84.78	85.3	86.9	85.2	86.0	70.7	77.5	74.1	3.4	2.1	2.8*
Eugene	17.0	4.3	10.7	70.7	46.3	58.5	84.0	78.65	81.3	86.7	81.0	83.8	86.0	59.5	72.8	16.9	7.3	12.1
Fgenesh	21.4	6.5	14.0*	73.1	57.6	65.4	90.7	79.20	85.0	91.3	79.5	85.4	81.5	66.7	74.1	3.9	1.6	2.8*
Augustus	15.8	4.3	10.1	72.4	50.6	61.5	89.7	78.92	84.3	90.8	79.8	85.3	86.2	65.6	75.9	38.7	17.11	**27.9**

Table 6.4.: Sensitivity (Sn), specificity (Sp) and their average for various signal predictions by *ab initio* gene finding systems (Category 1): transcription start sites (*tss*), translation initiation sites (*tis*), acceptor (*acc*) and donor (*don*) splice sites in coding regions, translation termination sites (*stop*), cleavage sites (*cleave*). The evaluation was performed on the two gold standard gene sets of the nGASP test regions (referred to as **ref1** and **ref2** in [34]). For *tss* and *cleave* the evaluation is similar to the promoter evaluation at the EGASP competition [71]: Within 20 nucleotides of an annotation a prediction is counted as correct. However, for each annotation only one positive is counted. Alternative annotations are counted more than once if a prediction is within 20 nucleotides of both annotations. Note that Craig and Fgenesh are not able to predict untranslated regions. We therefore used the translation start and stop as transcription start and cleavage site predictions (affected results are marked with $*$).

In Table 6.4 we evaluate the most accurate *ab initio* gene finders of nGASP when used to detect individual signals. Figure 6.2 shows the improvement of mGene's signal predictions relative to the top nGASP submissions: We achieved significantly higher accuracy on all signals except for the cleavage site, where Augustus' predictions were found to be more accurate.

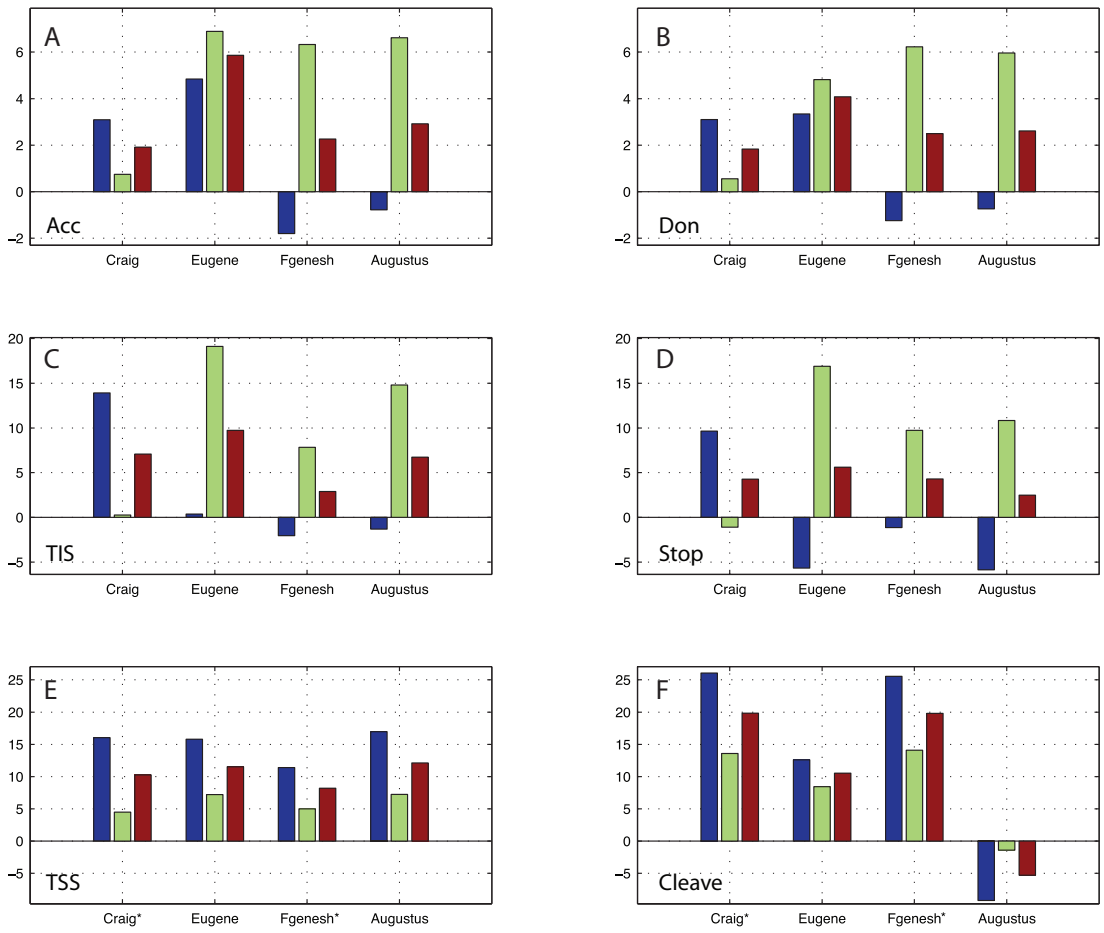


Figure 6.2.: Improvement of mGene.init *ab initio* predictions on selected signals: A) acceptor splice sites, B) donor splice sites, C) translation initiation sites, D) translation termination sites, E) transcription start sites (±20nt), and F) cleavage sites (±20nt). mGene.init's predictions are compared to the predictions of the best submissions in Category 1: Craig, Eugene, Fgenesh, and Augustus. Shown are differences of percent values for sensitivity (Sn; blue), specificity (Sp; green), and their average (red). Note that Craig and Fgenesh are not able to predict UTRs. We therefore used the predicted translation start and stop as an estimate of gene start and stop (relevant results are marked with an asterisk).

Analysis of Gene Starts and Ends

Because start and end sites of transcription are particularly difficult to predict correctly, even as parts of entire predicted genes, we investigate the deviation of the predictions from the annotation in more detail. Figure 6.3 depicts the distributions

of predicted gene starts and ends relative to the closest corresponding site of a high-quality gene annotation (**ref2**). For this analysis we have excluded annotated genes with a UTR length smaller than 10 nucleotides, because we assume that these annotated UTRs are very likely incomplete. The resulting set consists of 430 and 587 annotated transcription start and cleavage sites, respectively.

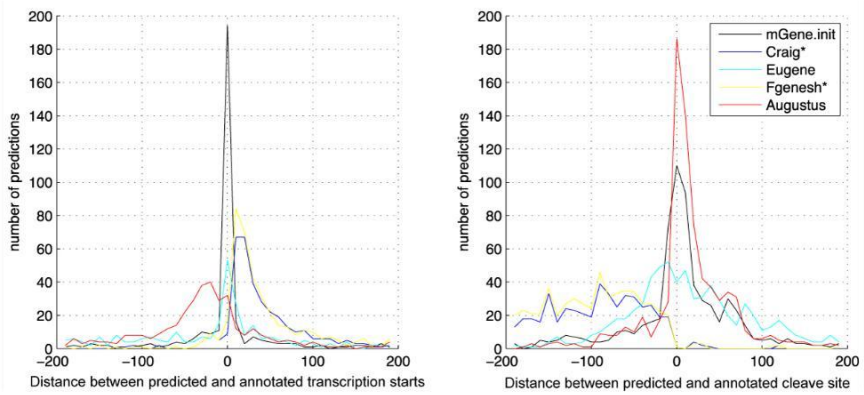


Figure 6.3.: Distributions of predicted gene starts (left: tss) and ends (right: cleave) relative to the closest corresponding site (according to the annotation) on the nGASP test regions. As Craig and Fgenesh do not predict UTRs, the translation boundaries are used as transcription start and cleavage site, which results in consistent under-prediction of the UTRs. Note that the total numbers of predictions can be different for each gene finder.

For the gene start predictions, mGene is very accurate, with low variance and essentially no bias. This can be attributed in part to the *trans*-splicing model of mGene rather than to the *tss* signal detector alone. Eugene also produces a favorable distribution of *tss* predictions, while Augustus is biased towards too long 5' UTRs.

For the cleavage sites, Eugene again does not seem to consistently prefer too short or too long UTRs, but often misses the true boundaries (high variance). While mGene does reasonably well, with medium variance and a tendency to predict too long 3' UTRs, Augustus can be seen to place a higher fraction of predictions close to the annotated site, suggesting that mGene could be further improved by adopting a cleavage site model similar to that of Augustus.

6.2.5. Feature Contributions to the Discriminative Score

Training the HSM-SVM of mGene amounts to determining a suitable parametrization $\boldsymbol{\theta}$, which is basically a concatenation of function values $\mathbf{v}$ of transformations (PLiFs) on the input features (see Section 5.3). We analyzed this parameters, to understand which components are mostly responsible for the performance of the complete system. Some of the learnt PLiFs are depicted in Figure 6.4. Here, transformed score contributions are plotted verses post-processed SVM outputs. For the

signal sensors, we observe that the resulting PLiFs have sigmoid shape. Hence, small differences matter for positions where the prediction is on the boundary between positive and negative. Once the signal is strong enough, the contributions stays the same.

The different PLiFs can be directly related to each other as they additively contribute to the discriminative score $G(\mathbf{x}, \mathbf{y})$. Therefore, the range on the y-axis indicates the importance of the features for accurate prediction. In Figure 6.5, one can for instance observe that acceptor and donor splice sites as well as coding exon and intron content are particularly important.

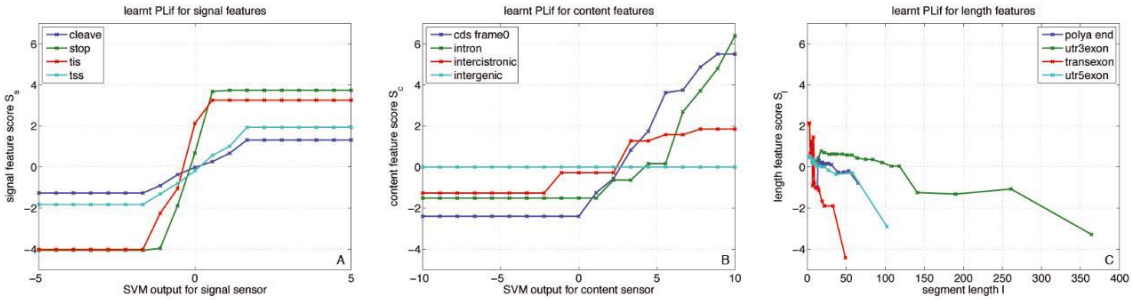


Figure 6.4.: Piece-wise linear functions transforming the SVM outputs of a few signal and content sensors as well as the segment length. The y-axes correspond to the scores contributing to $G(\mathbf{x}, \mathbf{y})$ for a given SVM output or length on the x-axis. The range on the y-axis indicates the importance of the feature for accurate prediction.Note that this figure was generated based on the development version of mGene, which used a different range (-5 to 5) for the x-axis of the PLiFs.

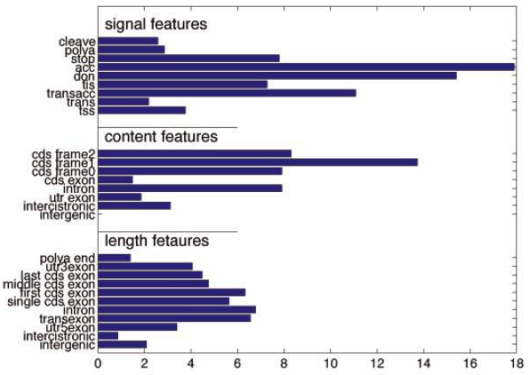


Figure 6.5.: Range on the y-axis of the piece-wise linear functions for signal, content and length contributions. Splice sites signals and coding exon content contribute most to the score $G(\mathbf{x}, \mathbf{y})$.

6.2.6. Summary and Discussion

The good performance of our gene finder mGene was proved in an objective competition based on the genome of the nematode *Caenorhabditis elegans* [34]. Considering the average of sensitivity and specificity, the developmental version of mGene exhibited the best prediction performance on nucleotide, exon, and transcript level for *ab initio* and multiple-genome gene prediction tasks. The fully developed version shows superior performance in ten out of twelve evaluation criteria compared to the other participating gene finders.

The runners-up in the *ab initio* category, Augustus and Fgenesh, both use generative training algorithms and have earlier been shown to perform well on human and fly genomes. In particular, Augustus was one of the best-performing methods in the *ab initio* category of the human annotation competition EGASP [71]. In nGASP, the only other participating discriminative gene finder, Craig, did not perform as well, although it was reported to outperform Augustus on a human data set [16]. The relatively weak performance of Craig may indicate the importance of good feature models for accurate gene prediction: With Craig, Bernal et al. [16] attempted to tackle the gene finding problem in a single step, simultaneously learning local feature properties and global characteristics of gene structures through an integrated training procedure. While this approach is conceptually appealing, it is very demanding in terms of computational resources for state-of-the-art discriminative learning algorithms. It thus prohibits to simultaneously use high-order signal detectors and large amounts of training data, as was done in training mGene (compare Section 4.2.6). The evaluation corroborates the notion that the disadvantage of simpler signal detectors is not fully compensated for by Craig's global parameter optimization strategy.

Recently, two other discriminatively trained methods, Conrad [47] and Contrast [70], were published. Both have a two-layered architecture similar to mGene and its predecessor mSplicer [122]. They yielded very promising results on human and fungal genomes. Unfortunately, they did not participate in the nGASP competition, and a direct empirical comparison is therefore pending. However, there are substantial differences in terms of 1) the underlying model, 2) the employed feature scores, and 3) the training algorithm: 1) Among the discriminative gene finders, mGene is the only one that is capable of predicting UTRs, *trans*-splicing, and polyadenylation sites. 2) For the feature detectors, we employ SVMs with one or more high-order DNA sequence kernels (up to order 22) on large sequence windows (up to 1000 bp) to obtain the most accurate models for segment boundary detection. In contrast, Conrad only uses position weight matrices (PWMs; corresponding to order 1) for signal detection, which have been shown to be suboptimal on such tasks (see Section 4.2.1 and [146, 148]). Contrast exploits DNA sequence features within very

small windows (6-30 bp) with SVMs and simple quadratic kernels considering second order relations. On the other hand, well-designed features describing multiple genome alignments, a strength of Conrad and Contrast in many applications, appear to be less crucial for nematode gene predictions. 3) The integrative step in Contrast is based on a standard conditional random field (CRF,[90]) framework that does not allow semi-Markov dependencies between labels. Consequently, Contrast is unable to model segment lengths appropriately. The explicit incorporation of length features is only realized in mGene and Conrad. However, while mGene employs HSM-SVMs, Conrad is based on semi-Markov CRFs [133]. Both approaches are state-of-the-art structured output prediction techniques. Although the optimization problems solved in the training step differ, their prediction accuracy has often been found to be similar when compared on the same data with the same underlying model [81]. It remains to be investigated how the two learning techniques compare for gene finding when identical features are employed.

Due to its discriminative framework and the two-layered architecture, mGene is a very flexible system. One advantage of this is that separate data sets can be used for training its individual parts (e.g. the signal detectors). In contrast to less modular systems, it can thus exploit the available data to a fuller extent. As another benefit, mGene's architecture readily allows the integration of additional information from diverse data sources. In the context of nGASP, we implemented first versions of mGene that utilize sequence conservation or known transcript sequences. Although mGene outperforms its competitors in both corresponding categories, these versions may not yet fully exhaust the potential of the input information sources. for instance, the more sophisticated features describing multiple alignments used in Contrast and Conrad may help to further improve mGene's performance.

However, it was an unexpected finding that none of the participating gene finders in category 2 could take advantage of the multi-genome alignments. There were no significant improvements observed relative to the *ab initio* predictions. One reason may be sub-optimal evolutionary distances between *C. elegans* and the aligned genomes (*C. briggsae* and *C. remanei*). Importantly, this reflects the need for accurate *ab initio* gene finders for genomes lacking deep alignments to other genomes.

6.3. Genome-wide Predictions for C. elegans and Discovery of Novel Genes

The nGASP challenge was launched with the main objective to assess the accuracy of current state of the art gene prediction algorithms in *C. elegans*, thereby ultimately improving the genome annotation for *C. elegans* and other *Caenorhabditis* genomes. Since mGene was among the best gene finders in the competition, we were asked — along with the FGenesh, Augustus and Jigsaw teams — to provide genome-wide predictions for *C. elegans* to be included into the official wormbase annotation. In the following sections, I will briefly describe how these predictions were generated and also give a comparative analysis of the different prediction sets. Eventually, we set out to confirm genes, predicted by mGene, that did not overlap with any genes previously annotated.

6.3.1. Data Set Generation

Generating Genome-wide mGene Predictions

For the genome-wide predictions on *C. elegans* we used the model mGene.init, which was trained on the nGASP training regions. However, to improve accuracy, we applied the model in a category 3 setting (mGene.seq), i.e. we utilized alignments of protein and EST sequences available at prediction time. Here, we used the confirmed protein sequences (8,134 fully EST/cDNA confirmed translated coding regions) from the *C. elegans* annotation WS180. Additionally, we used annotated peptide sequences from *C. briggsae* (19,334) and *Saccharomyces cerevisiae* (11,081). Furthermore, we obtained mRNA sequences from dbEST [17] and the NCBI Nucleotide database for *C. elegans* (346,064 sequences).

Preparing the Annotated Gene Set WS180

We used the Wormbase genome annotation version WS180 and parsed all coding transcript entries. We called the reconstructed gene set Anno180. Additionally, we parsed all pseudo-genes and all non-coding RNAs from the WS180 annotation. These sets will be denoted as pseudo180 and ncRNA180. More details can be found in the Appendix D.

Deducing Exons from EST Alignments

In addition to the annotation WS180, we have generated a set of genes from the alignment of EST sequences derived from the NCBI Nucleotide database for *C. elegans* (346,064 sequences).

6.3.2. Quality Assessment of the Prediction Set

mGene on Wormbase

The genome-wide mGene predictions were provided to the wormbase consortium and can be displayed in the Wormbase genome browser. By visual inspection, we observe a very good agreement of our predictions with experimental evidence (see Figure 6.6). Additionally, we provide the predictions of individual signal sites for two popular genome browsers (Wormbase and UCSC), which may help human curators to annotate genes in the future.

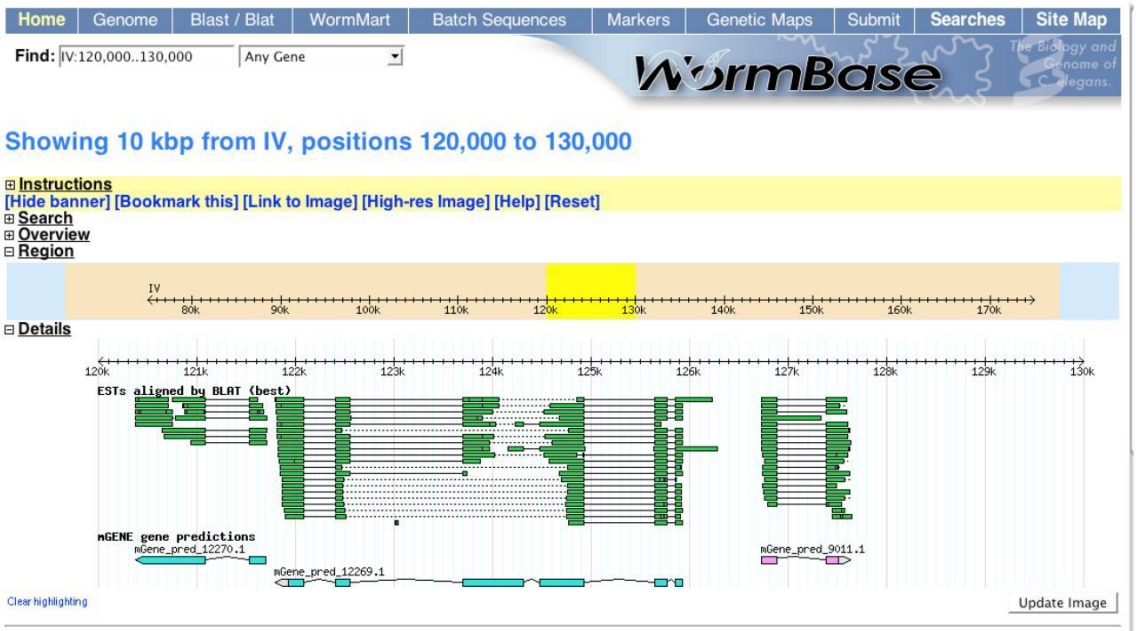


Figure 6.6.: Screen shot from the Wormbase genome browser, showing a 10 kbp region on chromosome IV of the *C. elegans* genome. Displayed are three gene models that were predicted by mGene. The predictions show very good agreement with experimental evidence, i.e. EST alignments.

Comparison of Prediction Sets with the Annotation

To verify that the performance estimations measured on the nGASP test regions (see Table 6.2) are indeed representative for the whole genome, we analysed the mGene predictions with respect to the *C. elegans* annotation WS180. From now on, the full set of 20,133 genes from this annotation will be called *all genes*. It corresponds to the **ref2** set that was used for the specificity analysis in the nGASP competition. We additionally generated a set of high quality genes, referred to as *confirmed genes*, consisting of 6496 genes. For this set, we removed all predicted or only partially confirmed transcripts. It is therefore comparable to the **ref1** set on the nGASP

test regions, which was used for the sensitivity measurement. We define a third set, *unconfirmed genes*, that contains 4634 genes with purely predicted isoforms. We measured sensitivity and specificity values, which are reported in Table 6.5 with respect to these sets.

	Exon				Gene			
	Sn			Sp	Sn			Sp
	All	Unc.	Conf.	All	All	Unc.	Conf.	All
mGene.seq	85.0	68.2	92.4 (92.2)	80.8 (83.5)	56.6	26.1	82.2 (80.2)	53.0 (52.4)
Fgenesh++	**92.4**	**90.3**	92.0 (90.4)	**88.7**(80.9)	**82.5**	**77.7**	83.3 (78.5)	**71.0** (52.1)
Augustus	84.5	63.1	**93.7** (92.5)	83.5 (80.2)	60.3	29.4	**86.8** (81.0)	56.4 (50.0)
Jigsaw	86.9	71.9	92.3 (90.5)	89.8 (87.4)	68.8	42.5	86.5 (80.0)	67.8 (61.0)

Table 6.5.: Sensitivity (Sn) and Specificity (Sp) on exon and gene level for genome-wide predictions on *C. elegans* as compared to the annotation WS180. The result of the best performing method according to each of the evaluation levels is set in bold face. The gray columns correspond to the nGASP evaluation settings and the respective numbers achieved on the nGASP test regions are given in brackets (compare Table 6.2): Sensitivity is measured on all confirmed genes (Conf.) of the annotation, specificity is measured on all annotated genes (All). We additionally report sensitivity measures on all and all unconfirmed (Unc.) genes, to better understand the differences compared to Table 6.2

For mGene.seq, the sensitivity values on the confirmed genes and the specificity values on all genes agree very well with the reported numbers in Table 6.2. We also downloaded the gene prediction sets provided by Augustus, Fgenesh++, and the combining system Jigsaw from the wormbase website. The performance evaluation is included in Table 6.5. Remarkably, the FGenesh++ prediction set shows significantly higher sensitivity and specificity values than reported on the nGASP test set: For example on the gene level, specificity is almost 19% better than estimated on the nGASP test set. At the same time, FGenesh++ predicts as much as 83% of the confirmed genes completely correct. However, the high agreement between FGenesh++ and the annotation holds not only for the confirmed set of genes, but also for the unconfirmed genes, where Augustus and mGene.seq show strong disagreement with respect to the annotation. An ultimate conclusion from this finding is difficult, because the training conditions were not well defined. However, a possible explanation for the very good reproduction of even uncertain parts of the annotation, is, that FGenesh++ was possibly retrained on the complete genome including unconfirmed genes. It is also possible that it used additional hints, not only from EST, cDNA and protein alignments, but also from predicted genes. A third explanation could be that predictions from FGenesh++ were already included in the annotation WS180, or that the gene finder used to generate predictions for the annotation is very similar to FGenesh++. Sensitivity and Specificity measured for Augustus are also higher than for mGene.seq and the values measured for Augustus on the nGASP

test set. This might reflect the advantage of Augustus to include alternative isoforms derived from the alignments. However, it could also be explained by the possibility that the external hints used by Augustus are covering the confirmed genes in the annotation better than the hints used by mGene.

Comparison of Prediction Sets with Experimental Evidence

To further examine this question, we compared the predictions with a set of transcripts, which were generated by aligning ESTs to the genome. For the analysis, we concentrated on internal exons, because the alignments might cover genes only incompletely. While this set of exons should be contained in the confirmed part of the annotation, we indeed observe that only around 50% are covered by confirmed annotated exons (however, of these, 94% are correct). On the contrary, 32% of confirmed annotated exons are not overlapping with our EST alignments. (This is however expected as we ignore evidence from protein alignments or other sources.) In any case, the set of confirmed genes from the annotation is quite different from our set of exons, which was deduced by EST alignments. In Table 6.6, we report sensitivity and specificity on the set of internal exons. In this case, a predicted exon was counted as correct if both boundaries were correct, and as a false prediction if it overlapped a region covered by an EST alignment, but did not exactly match an EST-confirmed exon. (We therefore observe higher specificity values than in Table 6.5, as we here only evaluate regions covered by an EST alignment.) On this set, mGene.seq achieves higher sensitivity and specificity than Augustus and Fgenesh. Notably in this case, the considered alignments were used as external hints during predicting with mGene.seq. On the other side, it is unclear, which evidences were used by the Augustus and Fgenesh teams, potentially a set that is more similar to the confirmed annotated genes.

Table 6.6.: Sensitivity (Sn) and Specificity (Sp) on exon level for genome-wide predictions on *C. elegans* as compared to 56,970 internal exons inferred from EST alignments. We counted a predicted exon as correct if both boundaries were correct, and as a false prediction if it overlapped a region covered by an EST alignment, but did not exactly match an EST-confirmed exon. Note that we observe higher values than in Table 6.5, as we here only evaluate regions covered by EST alignments.

	Sn	Sp
mGene.seq	90.1	99.0
Fgenesh++	85.4	94.6
Augustus	90.1	96.1
Jigsaw	88.1	99.1

Characteristics of Gene Sets

In Table 6.7 and Table 6.8, we present a short summary of the main differences between the *C. elegans* annotation WS180 and the gene predictions. The gene finders predict between 82% and 86% of the confirmed annotated genes completely correct. We observe that Augustus is the gene finder that cuts the highest amount of annotated genes into two predictions. mGene on the other hand, merges many annotated genes, where mostly two genes are merged into one predicted gene.

	Genes (1)	Correct (2)		Cut (3)		Merged (4)		New (5)		Missed (6)	
		All	Conf	Anno	Pred	Anno	Pred	All	Unique	All	Conf.
mGene.seq	21,491	11,392	5343	958	1997	3004	1424	2197	809	297	31
Fgenesh++	23,368	16,603	5413	722	1495	582	273	2988	974	217	54
Augustus	21,525	12,136	5639	1090	2291	1627	793	1729	345	704	72
Jigsaw	20,423	13,851	5621	853	1759	1501	737	970	8	822	78
WS180	20,133		6496								

Table 6.7.: Summary statistics of the predictions on the *C. elegans* genome of mGene.seq, Fgenesh++, Augustus, and Jigsaw. Reported are (1) the predicted number of genes, (2) the number of all and all confirmed genes for which at least one transcript is predicted completely correct (in the coding region) by a given gene finder, (3) the number of annotated genes that overlap with more than one predicted gene and the number of predicted genes that are involved in this process, (4) the number of annotated genes that are merged into one predicted gene, and the number of predicted genes that are involved in this process, (5) the number of predicted genes that show no overlap with an annotated gene, and the number of predicted genes that show no overlap with any other predicted or annotated gene, and (6) the number of all and all confirmed annotated genes that have no overlap with a predicted gene. In all cases, an *overlap* may also include untranslated regions.

Additionally, we observe that the gene finding systems detect between 970 (Jigsaw) and 2,988 (Fgenesh++) *new* genes, i.e. genes that do not overlap with an annotated gene, while completely missing between 217 (Fgenesh++) and 822 (Jigsaw) annotated genes. A majority of the new genes predicted by the combiner Jigsaw were also found by the other three gene finding systems (82%). mGene missed the fewest confirmed genes, namely only 0.7% of all confirmed genes. Of its predicted genes, 809 are neither present in the annotation nor found by any other gene finder. As we show below, there is good reason to believe that many of them are genuine.

The basic characteristics (mean number of exons per gene, median lengths of exons, introns, and ORFs) between the catalogs are fairly similar (see Table 6.8). The corresponding distributions are depicted in Figure 6.7. The biggest differences between the gene sets can be observed in the number of exons per gene and the length of the ORF. The distributions of exon and intron lengths on the other hand are very similar. We also examined the characteristics of newly predicted and missed

genes. While new genes are fairly similar to the complete sets, missed genes tend to have significantly fewer exons per gene and short exons and ORFs. It is likely that these genes are most difficult to predict. However, it is also possible that a large fraction of these annotated genes are actually wrong. We will further examine this question below.

	No. of	No. exons/gene		Exon length [bp]		Intron length [bp]		ORF length [bp]	
	genes	mean	median	mean	median	mean	median	mean	median
mGene.seq	21,491	6.2	5	200.3	146	327.3	65	1249.4	990
Fgenesh++	23,368	5.7	5	202.5	146	305.8	71	1150.4	939
Augustus	21,525	5.8	5	202.1	146	294.7	64	1174.2	945
Jigsaw	20,423	6.0	5	207.6	148	297.7	64	1252.4	996
WS180	20,133	6.0	5	203.5	147	292.7	63	1229.0	996
New Genes									
mGene.seq	2196	6.1	5	208.8	146	403.4	85	1266.9	981
Fgenesh++	2988	6.5	5	212.4	147	353.1	82	1374.0	1013
Augustus	1729	6.2	5	209.1	150	344.9	92	1290.2	978
Jigsaw	970	6.7	6	212.4	151	340.7	79	1433.7	1085
Missed Genes									
mGene.seq	297	2.1	2	174.1	135	318.5	120	362.3	309
Fgenesh++	217	2.6	2	156.8	116	161.9	53	410.4	279
Augustus	704	3.1	2	167.3	120	234.2	70	514.3	339
Jigsaw	822	3.0	2	161.7	123	230.8	68	483.1	363

Table 6.8.: Summary statistics of the predictions on the *C. elegans* genome of mGene.seq, Fgenesh++, Augustus, and Jigsaw. Reported are the predicted number of genes, average and median number of exons per gene, exon length, intron length, and ORF length.

As Figure 6.7 shows, the distribution of ORF lengths in the unconfirmed annotated genes displays a strong peak at a length of 1000 bp. It turns out that this peak is likewise observable in all prediction sets and preserves quite well the same feature in the WS180 annotation (and also in earlier annotations and in the nGASP training genes). However, when analyzing the ORF lengths of the annotation WS180 in terms of confirmed, partially-confirmed and predicted genes, this peak is only present in the predicted set of genes and the partially confirmed set (data not shown). We speculate, that this peak could be an artifact of the original GeneFinder predictions that were used for generating the annotation (personal communication with L. Stein, 2008). However, it is interesting to see how the subsequently applied gene finders all mimic the same distribution.

We also examined the predicted lengths of 5' and 3' UTRs. As depicted in Figure 6.7, we find that mGene is capable of accurately reproducing the 5'UTR length distribution of confirmed genes. In contrast, Augustus and Jigsaw tend to predict

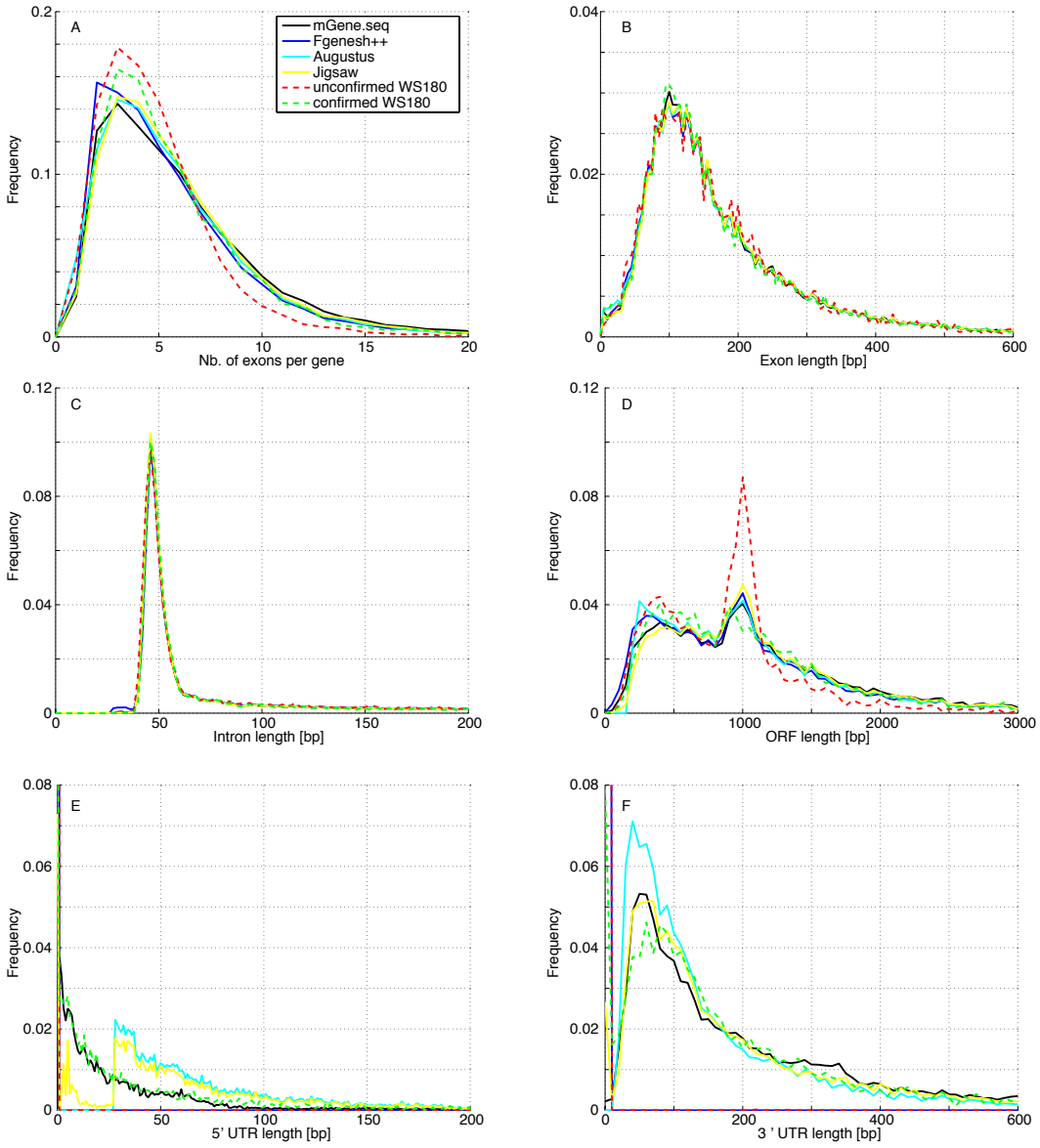


Figure 6.7.: Distribution of the predicted number of exons per gene, the length of exons, introns, coding regions, and 5' and 3' UTRs, for the predicted gene catalogues of mGene.seq, Fgenesh++, Augustus, and Jigsaw. Additionally, the distributions for unconfirmed and confirmed genes from the annotation are displayed.

too long 5' UTRs. This finding confirms our results on the nGASP test regions (compare Figure 6.2), where we have seen that mGene predicts the gene starts most accurately. This is most probably due to the trans-splicing model that is incorporated in mGene. The length distributions of the 3' UTR are fairly similar between the different gene sets.

6.3.3. Domain Content Annotation of New Genes

mGene predicted 2197 novel genes that are not part of the WS180 annotation. We are particularly interested in this set, and therefore we investigated if these sequences are likely to be true genes or rather false positives of our gene prediction system by examining if known protein features can be identified in the predicted sequences. To this end, we computed protein domain predictions with the iprscan application (version 4.3.1) using default parameters [191][4]. All domain predictions are based on the Interpro collection of protein domain databases (release 16.2) [78]. We could associate 621 (28.3%) of the new genes with Interpro domains. We performed the same analysis on the 297 genes in WS180 completely missed by mGene. In this case, we could only associate 27 (9.1%) with Interpro domains, suggesting that these genes are less likely to be functional.

Figure 6.8 depicts the ten most abundant domain categories expressed in terms of matching novel genes. The seven-transmembrane chemo-receptors (103 genes) constitute the largest addition to the current *C. elegans* annotation. For most chemoreceptor families, *C. elegans* has the largest number of genes when compared to *C. briggsae*, suggesting changes in the importance of chemo-reception between the species [169]. Another interesting group are the F-box proteins. It has been hypothesized, based on patterns of molecular evolution, that most members of the F-box superfamily are adapters that target foreign proteins for proteolysis. Thomas [168] speculates that this system functions to combat viral pathogens or bacterial protein toxins.

Another hint for functionality of a given sequence is the conservation between organisms. Intriguingly, 1855 (85%) of the novel genes show significant protein sequence similarity to mGene gene predictions in other *Caenorhabditis* species (see below, Section 6.4).

6.3.4. Confirmation of Novel Genes

We then tested for the fraction of newly predicted and missed unconfirmed genes for which experimental evidence of gene expression can be found. We conducted a set of validation experiments based on RT-PCR and sequencing of mRNA fragments of selected genes. Genes targeted for verification were selected based on a set of predictions produced earlier. In this case, we had predicted on the genomic sequence of *C. elegans* version WS170, and the resulting gene set was transferred in a subsequent step to genome version WS180. It will be called mGene170. In this set, we found 1077 gene predictions that did not overlap with an annotated gene.

[4]This analysis was carried out by Christoph Dieterich

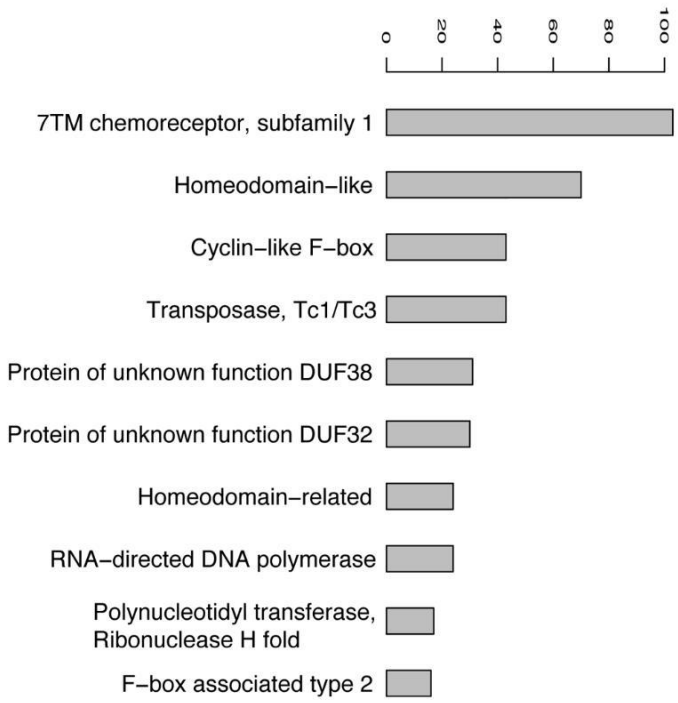


Figure 6.8.: The ten most abundant protein domain annotations for the group of novel mGene genes.

Among those, 425 gene predictions overlapped with annotated pseudo-genes and 17 predictions overlapped with non-coding RNAs. On the other hand, 347 annotated genes were missed by the *mGene170* predictions.

To assess the quality of our predictions, we examined three different sets of predicted genes:

mGene170-pseudo — mGene170 genes that had no overlap with any segment from Anno180, yet overlapped with segments from pseudo180.

mGene170-nc — mGene170 genes that had no overlap with any segment from Anno180, yet overlapped with segments from ncRNA180.

mGene170-novel — mGene170 genes that had no overlap with any segment from Anno180, pseudo180, or ncRNA180.

Additionally, we examined a fourth group of unconfirmed annotated genes that had no overlap with the predictions:

missed-unconf180 — Anno180 genes that were unconfirmed and did not overlap with any gene from mGene170.

From each of these four groups, we selected genes with at least two coding exons and designed primer pairs in the first and last coding exon using primer3 (cf. [125]

and `http://primer3.sourceforge.net`) with default parameters. We checked that the primer sequences had only a single match in the *C. elegans* genome and chose a subset at random if the number of resulting genes exceeded 50. The subsequent experimental procedure is described in more detail in Appendix D.

The obtained sequences were aligned against the genome using blat [82]. Subsequently, we checked whether the alignment had any overlap (> 1nt) with the corresponding targeted gene. If so, and if, in addition, the alignment hit was unique or the score of the hit to the targeted gene was highest and the difference to the second best hit was sufficiently large (> 5%), we counted the prediction as correct. Furthermore, we counted the number of genes that had at least one splice site confirmed by the aligned sequence. In total, we tested 157 RNA sequences. All experiments are summarized in Table 6.9. More detailed results for each experiment can be found in the Appendix D.

	total number	number tested	sequence results	hit to target	unambiguously	correct structure
mGene170-pseudo	425	46	35	29	25	25
mGene170-nc	17	13	6	6	5	5
mGene170-novel	635	52	35	31	31	26
missed-unconf180	347	46	14	13	11	7
DNA control	-	20	20	18	17	-

Table 6.9.: Summary of validation results by RT-PCR and sequencing of mGene170 predictions in the three different categories mGene170-pseudo (overlap with pseudogenes), mGene170-nc (overlap with non-coding RNAs), and mGene170-novel (new predictions), as well as unconfirmed genes in the WS180 annotation that mGene did not predict.

For the three mGene prediction sets the RNA sequences of 46% to 76% of the candidates could be generated, and 38% to 59% could be unambiguously aligned to the target DNA locus. On the contrary, only 30% of the tested missed genes could be sequenced and only 24% aligned unambiguously to the target.

We also compared the sequencing results with the new set of mGene predictions on the WS180 genome, denoted as mGene.seq. However, not all previously analyzed cases could be assigned to a corresponding prediction: For the results shown in Table 6.10 we only considered the cases where both primer pairs could be found in correct orientation in a predicted transcript in mGene.seq's predictions. The number of cases usable for the evaluation of the mGene.seq predictions and missing WS180 annotations is, therefore, considerably reduced: 57 and 24 instead of 111 and 46 cases, respectively (see Tables 6.10 and 6.9). While this may look like a big change in the prediction sets, please note that the targeted genes are among the

ones hardest to detect: they are also missing in the annotation, possibly due to low expression or weak signals. The genome-wide predictions mGene170 and mGene.seq, however, are very similar for a large proportion of the genes. The success rates of the validation experiments for the reduced sets of genes (mGene.seq) agree well with the success rates determined with the earlier predictions (mGene170). We can observe that a significantly larger fraction of novel genes can be validated as compared to unconfirmed genes missed by mGene.seq and also to previous validation studies in human [71].

	No. of genes	No. tested	No. confirmed	Frac. confirmed
New genes	2,197	57	24	$\approx 42\%$
Missed unconf. genes	205	24	2	$\approx 8\%$

Table 6.10.: Experimental analysis of the difference between the *C. elegans* annotation WS180 and mGene.seq predictions. Shown are the number of new genes and unconfirmed genes missed by mGene.seq relative to WS180, the number of genes considered in validation experiments, and the number and fraction of such genes that were verified to have mRNA expression. If a sequenced fragment could be mapped unambiguously to the genomic region of the corresponding gene, we considered the expression of the gene as experimentally validated.

6.3.5. Summary and Discussion

With mGene, we have generated genome-wide predictions for the nematode *C. elegans*. An in-depth analysis of these predictions showed that the accuracy measurements for mGene estimated on the nGASP test set are indeed representative for the whole genome. Despite this assuring finding, the analysis was particularly difficult as there is no defined gold standard, to which the predictions can be compared. The problem is that the purpose of each new prediction is to *improve* the annotation, which consists of gene sets of variable confidence. At the same time, a good result is normally assumed if the prediction set shows high *similarity* to the annotation. This was also put into practice for the nGASP evaluation, where parts of the annotation were used as reference set. To overcome this problem, we have generated a new set of exons from EST alignments, thereby assuring that there is experimental evidence available for each entity. The high quality of mGene predictions could also be confirmed on this set. Here, however, we encounter another problem: To provide the best possible prediction set to the community, we have generated category 3 predictions, i.e. we have incorporated external hints to improve the accuracy. These hints include EST alignments. It is therefore not very remarkable that the performance of mGene is high on exons derived from these alignments. Consequently, the

evaluation on the EST gene set is not representative for predictions of genes that have not previously been covered by EST evidence. This problem will be discussed further in the next section. The entanglement between new predictions and the annotation, which consists partially of earlier prediction, becomes clear in a third issue: In the annotated as well as all predicted sets, we have detected a strong bias towards genes with an ORF length of 1000 bp. We have strong reasons to believe that this is an artifact of the initial system GeneFinder that was used to generate the annotation, as the bias is not observed in the set of confirmed genes. This stresses the importance of training set selection: To avoid the propagation of systematic errors, gene finders should only be trained on a set of highly trusted genes. Recently, whole-genome, experimental data of a deep sequencing effort has become available [76]. While the resulting gene sets are also not without bias, a thorough comparison with the predictions will shed further light on the quality of computational gene finding.

A further examination has revealed that between 1000 and 2000 predicted genes are not contained in the current genome annotation. We have therefore investigated if these predictions are indeed true genes or represent false positives. In this respect, pseudo genes probably pose the biggest problem as they have the appearance of true genes and are also difficult to annotate [107]. In fact, many pseudo-genes can be adjusted into a form that has coding potential. However, when they are derived from other genes but with miss-functional domains, or contain frameshift mutations, they are much more likely to be pseudogenes than real genes. In fact, almost 40% (425 out of 1077) of the novel genes overlap with annotated pseudo-genes. Also, around 10% of the novel gene predictions could be annotated with 7TM receptor domains and it is known that many pseudogenes in *C. elegans* are derived from 7TM family genes. Another group of predictions was annotated as transposases, which are regarded as parasitic genes, and their proliferation is deleterious for the host. Therefore, transposition is commonly followed by inactivation [74].

In a set of validation experiments, we could show that a large number of predicted sequences are transcribed into mRNA. This includes over 50% of the tested predicted genes that overlapped an annotated pseudo-gene. While this is no proof for functionality — pseudogenes can also well be inactivated on the level of translation —, it suggests that many of the *de novo* predictions represent functional genes (perhaps with vestigial products) rather than outright pseudogenes.

In summary, both experimental and *in silico* analyses indicate that a large fraction of newly predicted genes are expressed, demonstrating that even the gene catalog of a well-studied organism like *C. elegans* can be substantially improved with mGene predictions.

6.4. Genome-wide Predictions for Other Nematode Genomes

In recent years, not only the genome sequences of several distant model organisms have been completed, but several projects have also sampled the genomes of related species [101, 156]. With this strategy, it will become clear to what extent the model organism is indeed a good representative of the genomes of related species. Also, comparative sequence analysis allows the identification of evolutionarily constrained regions, thereby improving the annotation of functional elements. Additionally, it serves as a valuable resource to examine the dynamics of genome evolution. Currently, the Washington University Genome Sequencing Center is sequencing the genomes of the nematodes *C. remanei*, *C. japonica* and *C. brenneri* (coverage $\geq$ 6-fold), bringing the number of Caenorhabditis genomes to five when added to the existing *C. elegans* and *C. briggsae* genome sequences [157, 159]. In contrast to the extensively annotated genome of the *C. elegans*, few analyses have been performed on the other nematode genomes. We therefore used mGene to generate gene sets for each of these species. The results are presented in this chapter.

6.4.1. Data Set Generation

Generating Genome-wide mGene Predictions

For the generation of genome-wide gene predictions for *C. briggsae*, *C. brenneri*, *C. remanei*, and *C. japonica* we followed a similar approach as that described in Section 6.3.1: We applied the system mGene.seq, trained on the *C. elegans* nGASP test regions. To improve accuracy, we used available EST, cDNA, and protein data in the way described above. Here, we employed the confirmed protein sequences (8,134 fully EST/cDNA confirmed translated coding regions) from the *C. elegans* annotation WS180 for alignment against the five nematodes' genomes. Furthermore, we obtained mRNA sequences from dbEST [17] and the NCBI Nucleotide database for *C. remanei* (20,292 EST sequences), *C. briggsae* (2,424 EST and 108 other sequences) and *C. brenneri* (2,474 EST and 6 other sequences).

Genome-wide predictions for all five nematodes were provided to the Wormbase consortium in the form of GFF3 files and can also be downloaded from the project web site. Starting with the Wormbase WS190 release, mGene predictions will be standard features in future Wormbase genome browsers. Additionally, we provided genome-wide signal predictions, which can be accessed from the project web site and displayed in genome browsers as custom tracks.

Deducing Exons from EST Alignments

In the absence of high quality annotations for the four nematodes, we aligned EST sequences to assess the quality of the predictions. We obtained the most recent sets of EST and mRNA sequences from the NCBI Nucleotide database leading to 33,050, 29,929, and 16,537 new EST sequences for *C. japonica*, *C. brenneri*, and *C. briggsae*, as well as 143 new mRNA sequences for *C. remanei*, respectively. The sequences were aligned against their respective genomes using the procedure described in Section 4.1.1.

6.4.2. Characteristics of Predicted Gene Sets

The characteristics of the predicted gene catalogs for the five different nematodes are detailed in Table 6.11.

Genome	No. of	Genome	No. of	No. exons/gene		Exon length [bp]		Intron length [bp]		ORF length [bp]	
	contigs	size [Mbp]	genes	mean	median	mean	median	mean	median	mean	median
C. elegans	6	100.27	21489	6.3	5.0	200.3	146.0	327.1	65.0	1249.5	990.0
C. remanei	3670	235.94	31503	5.7	4.0	212.6	148.0	284.3	52.0	1186.1	951.0
C. japonica	4657	266.90	20121	5.3	4.0	228.2	147.0	598.6	71.0	1180.0	879.0
C. brenneri	10292	453.09	41129	5.4	4.0	226.1	152.0	288.8	53.0	1194.6	942.0
C. briggsae	12	108.48	22542	6.0	5.0	206.9	148.0	348.7	54.0	1231.9	957.0

Table 6.11.: Summary of gene characteristics for genome-wide predictions by mGene.seq on five nematode genomes. Shown are basic properties of the genomes: number of contigs, size of genomes, the number of predicted genes as well as the average and median exon, intron and ORF lengths.

As Table 6.11 shows, the genome sizes vary considerably for the respective assemblies, with *C. elegans* and *C. briggsae* having the smallest genome with around 100 Mbp, while the provided sequence for *C. brenneri* is about four times as large. In contrast to *C. elegans* and *C. briggsae*, the genome assemblies of the other three nematodes was still on-going, and their sequence was therefore only available in a large number of contigs. This will likely lead to difficulties in gene prediction as genes might be split and are potentially allocated to different contigs. Therefore, some of the differences in the gene catalogs might be attributed to this issue. For example, the number of predicted genes lies between 20,000 for *C. japonica* and more than 40,000 for *C. brenneri*. Also, the median number of exons per genes and the resulting length of open reading frames is slightly smaller for *C. brenneri*, *C. remanei*, and *C. japonica*, possibly indicating that some genes are predicted in fragments. Also, the lengths of predicted introns and exons show some variability, however there is no consistent trend emerging that hints at incomplete predictions in the unfinished assemblies.

6.4.3. Estimation of Prediction Accuracy

With the different characteristics of the gene predictions for the four nematodes, the question arises of what quality the mGene annotations are. Is mGene able to adapt to evolutionary distant genomes and how stable are the predictions on unfinished assemblies? Furthermore, we are also interested in investigating how the performance of mGene in this respect compares to that of the best competing gene finders. To answer this question, we downloaded genome-wide predictions for all considered nematodes with the Category 3 versions of Fgenesh, Augustus, and Jigsaw (Category 4) from the nGASP ftp site (`ftp://ftp.wormbase.org/pub/wormbase/nGASP/final_gene_predictions/predictions/*/processed/GFF3`).

We examined the agreement of the predictions of the four gene finding methods mGene.seq, Fgenesh, Augustus, and Jigsaw with internal exons of the aligned EST sequences. While this approach may not be unbiased (in Category 3 the same type of information is actually used to generate predictions), there does not appear to be an alternative in the absence of an independent high quality annotation. The results of the comparison are given in Table 6.12.

	No. of exons		Fgenesh			Augustus			Jigsaw			mGene.seq		
	total	new	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$
C. elegans	56970	0%	85.43	94.61	90.02	90.11	96.06	93.09	88.12	99.13	93.62	90.06	99.00	**94.53**
C. remanei	11500	1%	81.21	93.52	87.36	91.70	95.93	93.81	83.74	95.26	89.50	94.33	98.79	**96.56**
C. japonica	8385	100%	72.30	94.82	83.56	81.35	96.07	88.71	82.60	95.76	89.18	88.85	97.65	**93.25**
C. brenneri	12832	84%	73.56	93.12	83.34	82.50	93.15	87.83	75.16	97.05	86.11	89.16	96.90	**93.05**
C. briggsae	64968	95%	65.70	84.69	75.20	76.59	87.48	82.04	74.57	85.41	79.99	85.13	88.81	**86.97**

Table 6.12.: Consensus of internal exons inferred from aligned EST sequences with exons predicted by mGene, Fgenesh++, Augustus, and Jigsaw. We counted a predicted exon as correct if both boundaries were correct, and as a false prediction if it overlapped a region covered by an EST alignment, but did not exactly match an EST-confirmed exon. Shown are sensitivity (Sn), specificity (Sp), and their average. Also provided are numbers of considered exons found by EST alignments and fractions thereof that were utilized for prediction by the gene finding systems.

We observed that mGene.seq maintains a relatively high exon level accuracy of 93-97% for all organisms, except for *C. briggsae* (87%), where a non-negligible drop is observed. We find that mGene outperforms all other gene finding systems for all organisms, assuring that the model trained using data from *C. elegans* is general enough to yield accurate predictions on related genomes.

The values in Table 6.12 might not be fully conclusive, since some of the ESTs that have been used for evaluating prediction accuracy had been utilized in computing the predictions before. The second and third columns indicate the total number of internal EST-confirmed exons and the fraction thereof that was new to the evaluation, i.e. not previously utilized for prediction. Such ESTs exist due to

the time gap between prediction and evaluation and the intermittent growth of the dbEST database. This effect varies strongly over the considered species, which may render the interpretation of the results difficult. However, we observed that for organisms for which initially no or few EST sequences have been available (for instance *C. japonica*), mGene can still yield strong performance. In a next step, we restricted the evaluation to exons that had not been covered by any EST utilized at prediction time. The corresponding results, presented in Table 6.13, essentially confirm the conclusions drawn from Table 6.12.

	N	Fgenesh			Augustus			Jigsaw			mGene		
		Sp	Sn	$\frac{Sn+Sp}{2}$	Sp	Sn	$\frac{Sn+Sp}{2}$	Sp	Sn	$\frac{Sn+Sp}{2}$	Sp	Sn	$\frac{Sn+Sp}{2}$
C. elegans	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C. remanei	82	101.33*	92.68	**97.01**	97.53	96.34	96.94	95.77	82.93	89.35	100.00	93.90	96.95
C. japonica	8385	94.82	72.30	83.56	96.07	81.35	88.71	95.76	82.60	89.18	97.65	88.85	**93.25**
C. brenneri	10754	93.19	74.02	83.60	93.41	83.17	88.29	96.94	75.53	86.23	96.58	89.19	**92.89**
C. briggsae	62210	84.72	65.60	75.16	87.44	76.40	81.92	85.39	74.43	79.91	88.64	85.02	**86.83**

Table 6.13.: Summary of genome-wide predictions on five nematode genomes (see description in Table 6.12). Here, the evaluation was done only on exons that were not confirmed by an EST alignment used for gene predictions. *The submission by Fgenesh includes a small number of overlapping genes. Correctly predicted exons might therefore be counted more than once.

6.4.4. Comparative Analysis

We carried out a comparative analysis with Multiparanoid [1] on the newly predicted proteomes of *C. briggsae* (22,542 predictions), *C. brenneri* (41,129), *C. remanei* (31,503), and *C. japonica* (20,121). This analysis served two purposes: first, to determine gene homology relations, and second, to estimate the number of species-specific gene creations. We could identify orthology relations for a proportion of $63-84\%$ of a species' gene set, with *C. brenneri* being at the bottom and *C. elegans* at the top of the list (see Table 6.14). Particularly, we found 5,339 genes that are present in all five nematode species as single-copy orthologs. Considering also genes with potentially more than one copy per genome, we found 9885 orthologous groups common to all five nematode species that possibly existed in a common ancestor (see Figure 6.9). Substantial sequence similarity (≥ 50 bits) was found for a proportion of $82-94\%$ of the predicted protein sets, with *C. japonica* ranking lowest. These numbers demonstrate that the vast majority of predicted proteins have a phylogenetic counterpart in at least one other species. The genome of *C. japonica*, which is most distant from any of the other genomes [85], shows the greatest proportion of potentially species-specific genes ($\geq 18\%$).

Not surprisingly, there are 341 (31.2%) novel genes (see Section 6.3) among the species-specific gene predictions for *C. elegans* (see Figure 6.10). We followed up

	C. briggsae	*C. remanei*	*C. brenneri*	*C. elegans*	*C. japonica*
One:One orthologs	5,339 (23.7%)	5,339 (17.0%)	5,339 (13.0%)	5,339 (27.3%)	5,339 (26.5%)
Other orthologs	11,946 (53.0%)	16,588 (52.6%)	20,459 (49.7%)	11,002 (56.2%)	8,387 (41.7%)
Protein similarity	2,709 (12.0%)	4,819 (15.3%)	8,324 (20.2%)	2,150 (11.0%)	2,736 (13.6%)
No similarity	2,548 (11.3%)	4,757 (15.1%)	7,007 (17.0%)	1,092 (5.6%)	3,659 (18.2%)

Table 6.14.: Comparison of the gene sets predicted by mGene.seq for different nematodes: Shown are the number of protein coding genes predicted for each organism with one-to-one orthologs, other orthologs, with weak, and with no significant protein sequence similarity.

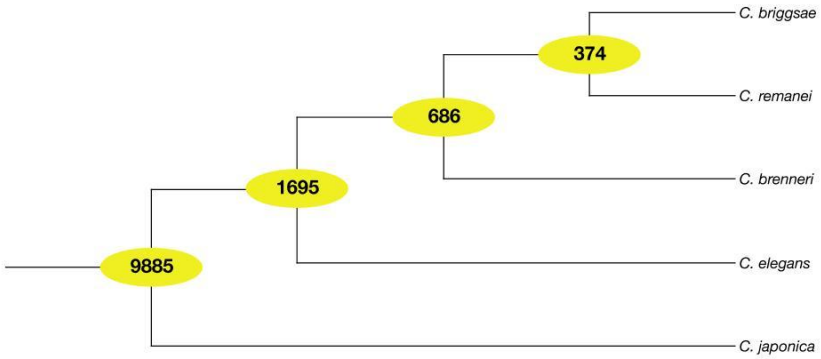


Figure 6.9.: Number of orthologous groups (9,885) shared among all five nematodes, as well as the number of additional orthologous groups shared across sub-trees of more closely related species, which are defined by the corresponding ancestral node.

on the potential function of species-specific genes by comparing them to the known protein universe as defined by the Uniref90 database [188] and the recently sequenced satellite nematode, *Pristionchus pacificus* [49]. Out of all the species-specific gene predictions, we found 541 (50%) matching entries for *C. elegans* in the Uniref90 database. Furthermore, among all novel gene predictions, only 51 match the Uniref90 database. Hence, the novel genes are highly enriched in the set of unknown genes, which suggests that they are genuinely novel (although we cannot exclude that some of these are false positive predictions).

For *C. briggsae*, we found 676 (27%) matching entries in the Uniref90 database, and only a very small fraction for the other nematodes. This is not surprising as the protein sequences of these little annotated genomes have not yet been entered into the Uniref database. Nevertheless, there are some remarkable exceptions. For instance, we found 90 *C. remanei* gene predictions, which show substantial similarity to bacterial genes from the genus *Acinetobacter* [soil bacteria; 62]. These predictions contain introns, a fact that argues against bacterial sequence contamination

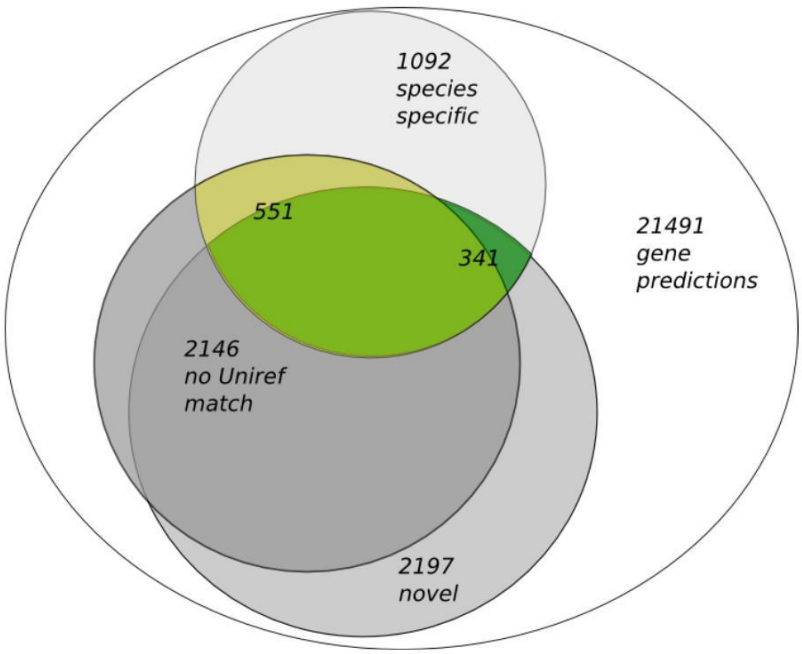


Figure 6.10.: Predicted gene set for *C. elegans*, including 1092 genes with no orthologs in any one of the other four nematodes, and 2197 novel gene predictions, among which we only found Uniref matches for 51 genes.

as spliceosomal introns are absent in bacterial genes.

Generally, we observe no similarity to *P. pacificus* gene predictions, which is in agreement with the phylogenetic position of *P. pacificus* as an outgroup. This further supports our interpretation of these *Caenorhabditis* predictions being species-specific gene inventions.

6.4.5. Summary and Discussion

We generated genome-wide gene predictions for five *Caenorhabditis* species. During the course of this work, one problem arose from the fact that some of the genome sequences were only available in unfinished assemblies. This poses additional challenges to gene finders as genes can be split and thus potentially occur in fragments. Also, the quality assessment of the predictions was particularly difficult in the absence of a *gold standard* annotation. Therefore, we generated a set of well trusted exons from EST alignments and compared the predictions against this set. Additionally, we have excluded those exons from this set that were used as hints in the prediction process and we could confirm that the accuracy measured on the full set is representative also for the *ab initio* parts of the predictions.

In the analysis, the question was investigated if and to what extent the system mGene is able to generalize from the species it has been trained for — here *C. elegans* — to related, but different species. We could show that the mGene model is able to capture the signatures of genic DNA signals to accurately annotate the remaining genomes. This is remarkable if contrasted with the average protein sequence conservation, which is just 78% for the most distant species pair (*C. briggsae* and *C. japonica*). The study therefore demonstrates the importance of a strong *ab initio* gene finder that is able to make accurate predictions without explicitly exploiting sequence conservation. (As shown in Section 6.2, category 2 gene finders using whole-genome alignment did not achieve a significant improvement in this particular setting). Additionally, our comparative analysis of the gene sets in the five species showed that between 6% and 18% of a nematode's gene set are species specific (at least as far as the phylum has been sampled so far).

In summary, mGene facilitates whole genome annotation for related species, even though they diverged from their last common ancestor more than 100 million years ago [157]. Therefore, other comparative sequencing efforts (e.g. in flies and vertebrates) may benefit from employing mGene as one of the primary gene prediction tools.

7. mGene.web: A Web Server for Automatic Gene Finding

It was already pointed out in previous chapters how new sequencing technologies have led to a dramatic increase of fully sequenced genomes in recent years. The demand for efficient, highly automated DNA sequence analysis tools is therefore greater than ever. In particular, we expect that the task of genome annotation will increasingly be performed by individual labs rather than large sequencing centers with dedicated resources and specialized expertise. Therefore, a feasible gene finding system does not only have to be highly accurate, but is additionally required 1) to be easy to use even for researchers with no programming experience, 2) has to be applicable to a large variety of newly sequenced organisms, and 3) should produce genome-wide predictions within a reasonable time. While *mGene* was proven to produce state-of-the-art predictions on Nematode genomes, it lacked - in its original form - the additional qualities demanded. We have therefore developed a web service, mGene.web, that offers convenient access to *mGene*. The system is integrated into the Galaxy framework [63], which also offers handy access to existing genome annotation databases as well as other computational tools. We have described mGene.web in [139] and the present chapter is extensively based on this publication. The service is available at `http://www.mgene.org/webservice`, it is free of charge, and can be used for eukaryotic genomes of small to moderate size (several hundred Mbp).

7.1. The Galaxy framework

The Galaxy framework was developed by the Center for Comparative Genomics and Bioinformatics at the Pennsylvania State University as both, a development framework and a ready-to-use meta-server to simplify the process of genomic analysis. It supports most common types of genomic, phenotypic and functional data and offers convenient tools to extract this data from genomic databases like the UCSC Genome Browser or EncodeDB. Files can also be uploaded from the user's computer and the resulting datasets are kept within a personal storage area. Additional, data sources can also be provided within data libraries, which can be shared with other users. A

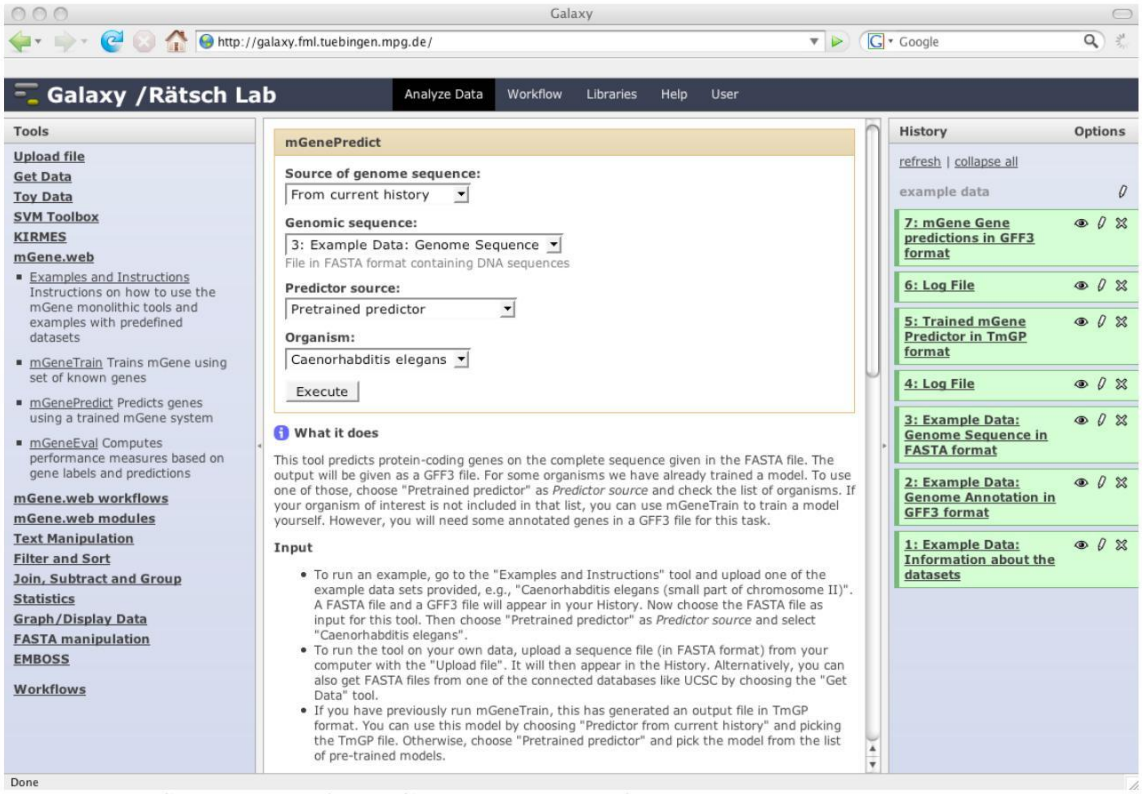


Figure 7.1.: Screenshot of the Galaxy user interface. It contains three main areas: The available tools are listed in the left column, details on a current action are given in the middle field, and the history panel on the right shows stored data, which are either uploaded by the user or results of analysis performed by the user.

large variety of analytical tools are integrated such that tasks like alignments, evaluation of DNA properties or taxonomical manipulations can easily be performed. To this end Galaxy provides a common graphical interface to manipulate data sources and launch analytical programs, Figure 7.1. For efficient computing, it handles multiple parallel processes. Galaxy also supports workflows that allow pre-defining the order in which tools are applied to data files to achieve a certain goal. These workflows can also be edited by, and shared among, users. Workflows can help to simplify the relatively complex process of creating a gene finding system, which involves many different steps. To run a workflow, one only needs to specify a few input arguments. As a result, one obtains all intermediate results allowing detailed inspection. Making it easy to keep track of a complete experiment, Galaxy's history system provides an analyses record, where all used and created data sets are visible. Thus, not only individual files, but complete analysis pipelines can be saved and shared, enabling an exceptional degree of reproducibility for entire experiments.

To add mGene.web to the Galaxy framework, we have downloaded Galaxy and are running a local instance at http://galaxy.fml.tuebingen.mpg.de/, where the mGene tools can be launched among other programs developed at the Rätsch lab.

7.2. Using mGene.web

Essentially there are two possibilities to use mGene.web for gene predictions: The simplest way is to use the monolithic tools mGenePredict and mGeneTrain. However, for a more flexible use and a more detailed output one can also use the individual modules together with workflows. We provide instructions and example runs for all tools at the FML Galaxy web-site. Also, information of the employed data types can be found in the Appendix E.

mGene.web monoliths

In case one wishes to annotate a newly sequenced genome mGeneTrain can be used to train a new model. This tool takes a FASTA file with the DNA and a GFF3 file with a set of known genes as input and returns a trained mGene predictor object that can be used together with mGenePredict to predict genes on given DNA sequences. We also provide a set of pre-trained models for a growing number of organisms. For organisms that are already in the list, users can skip the training and use mGenePredict directly. In this case, the function only requires a FASTA file with the DNA sequence as input and it outputs the GFF3 file containing gene predictions. The predictor and the predictions can be easily shared with other users via Galaxy's "share history" functionality. We also offer the tool mGeneEval to compare two GFF3 files, presumable one containing the ground truth (an annotation file) and another one containing the predictions. The resulting performance report includes sensitivity and specificity values on nucleotide, exon, transcript and gene level as well as for several signals and enables a convenient quality control.

mGene.web modules

The web service currently provides 15 core modules. They can be grouped into four groups: data preparation, signal training and prediction, content training and prediction, and gene structure training and prediction. A detailed description of the individual tools can be found in the Appendix E.

For a more convenient handling we also provide several workflows for sub-tasks, e.g. to train a signal predictor (Figure 7.2). Analogous to the simple black-box versions for training and predictions, we offer the two workflows:[1] mGenePredict and mGeneTrain. They are also displayed in the Appendix Figures E.2, E.1.

[1]The monolithic workflows described above are equivalent to the modular versions. They show less internal details and are slightly more efficient.

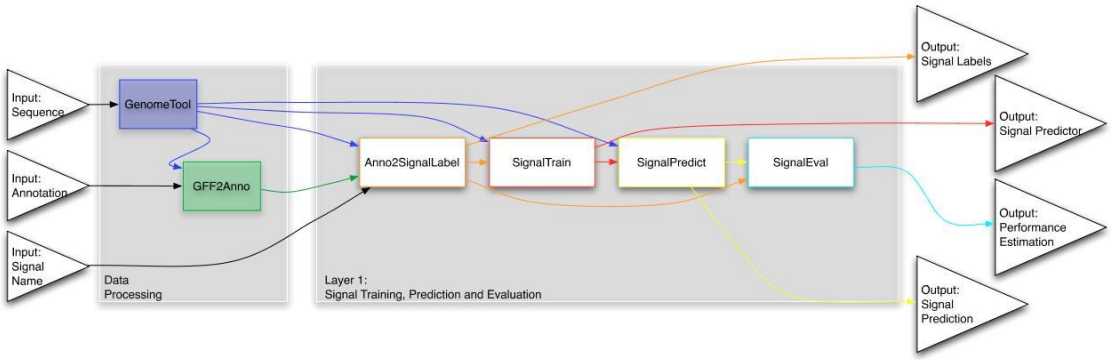


Figure 7.2.: To train a signal predictor six mGene.web Tools need to be applied in a defined order with specific inputs. To facilitate this complex process, we have defined a Galaxy workflow. The input to it are the DNA sequence in FASTA format and a partial annotation in GFF3 format. Additionally, the user has to choose the Signal, for which the training should be performed. The data transfer between tools is than internally defined (colored arrows). Eventually four output files are delivered to the user, including a file containing the signal labels, one containing the learnt model (SignalPredictor), one file with all signal predictions on the training regions, and one evaluation file.

7.3. Computing Time and Resources

The web service is running on a cluster with 84 AMD Opteron CPUs (2.2 Ghz) with 8 GB of RAM per 4 CPUs, which is shared with other web services. Training and prediction is split into several parallel sub-tasks to reduce the waiting time for users. Depending on the load of the cluster, the prediction of all six signals for the *C. elegans* genome (100 Mbp) takes about two hours (≈1 Mbp/min), the prediction of the five content types also takes about hours hours (≈1 Mbp/min), and gene prediction using these signals takes about six hours (≈300 kbp/min). The time for training the predictors strongly depends on the amount of available training data. While signal or content sensor training can typically be completed for a genome like that of *C. elegans* within a few hours, training the second layer of mGene.web may take 48 hours (depending on the size of the training set).

7.4. Summary and Discussion

We have developed the web service, mGene.web for computational gene finding. With mGene.web users can choose among an increasing list of models, pre-trained on different organisms and apply the selected model to predict on their input sequence. Additionally, they have the possibility to train the system with their own data for other organisms on the push of a button, a functionality that will greatly accelerate the annotation of newly sequenced genomes. mGene.web is built in a highly modular

way, such that individual components can be used autonomously. In combination with Galaxy workflows these individual units can easily be replaced by the user when other, possibly more advanced tools for a given sub-task become available. We expect that this particular feature may guide the systematic exploration of further improvements in the complex process of computational gene finding, thereby ultimately leading to more accurate gene predictions.

8. Extension: Learning from Related Organisms

While automatic procedures like mGene.web, facilitate the convenient generation of a new annotation for a given DNA sequence, the existence of — preferentially large — cDNA or EST libraries remains a prerequisite for the accurate training of a species-specific gene finder. To overcome this bottleneck, we examined the possibility of transferring a learnt model from a well annotated model organism to other genomes, for which little labeled data exists. This problem is put into the machine learning frame-work of domain adaptation. We investigated several methods and observe how the performance of these methods depend on the distance between the source and the target organisms, as well as the size of the target sample. In this study, the model organism *C. elegans* is used as an informant or source domain. The examined target organisms include two more nematode genomes, namely *C. remanei* and *P. pacificus*, as well as the very distantly related organisms *D. melanogaster* and *A. thaliana*. As a preliminary study, we focused on the problem of splice side detection. The results have been presented in a conference contribution [140] and are also partially included in [184].

8.1. Information Transfer and Domain Adaptation

So far we have considered supervised learning methods that use a set of labeled training examples to infer a model that generalizes well to unseen examples, and therefore allows to make predictions for these instances. One of the underlying assumptions was that training and test data are drawn from the same distribution. However, one often faces the situation where little or no training data is available for the problem of interest (the target domain), whereas a large corpus of labeled data exists for a related but different problem (the source domain). While it is highly desirable to exploit the labeled source data points to improve the performance on the target domain, it clearly violates the basic assumptions of supervised learning.

From now on, I will consider the case where we have ample labeled examples $(\mathbf{x}^n, y^n)$, $n = 1, \ldots, N_S$ from the source domain, but only few such examples $(\mathbf{x}^n, y^n)$, $n = N_S + 1, \ldots, N_T$ for the target domain ($N_T \ll N_S$). The examples are assumed to be drawn independently from the joint probability distributions $\mathcal{D}_S(\mathcal{X}, \mathcal{Y})$ and

$\mathcal{D}_T(\mathcal{X}, \mathcal{Y})$, respectively. These distributions $\mathcal{D}_S(\mathcal{X}, \mathcal{Y}) = \mathcal{D}_S(\mathcal{Y}|\mathcal{X}) \cdot \mathcal{D}_S(\mathcal{X})$ and $\mathcal{D}_T(\mathcal{X}, \mathcal{Y}) = \mathcal{D}_T(\mathcal{Y}|\mathcal{X}) \cdot \mathcal{D}_T(\mathcal{X})$ can differ in two different ways:

(1) In the classical *covariate shift* case [143, 160, 161], it is assumed that only the distributions of the input features $\mathcal{D}(\mathcal{X})$ vary between the two domains: $\mathcal{D}_S(\mathcal{X}) \neq \mathcal{D}_T(\mathcal{X})$. The conditional, however, remains invariant, $\mathcal{D}_S(\mathcal{Y}|\mathcal{X}) = \mathcal{D}_T(\mathcal{Y}|\mathcal{X})$. For a given feature vector $\mathbf{x}$, the label y is thus independent of the domain from which the example stems. In genetics, an example thereof could be the result of a duplication event in one organisms, where parts of the DNA occur in multiple copies, i.e. the composition has changed, while the biological function of the DNA elements remains conserved.

(2) In a more difficult scenario the conditionals differ between domains, $\mathcal{D}_S(\mathcal{Y}|\mathcal{X}) \neq \mathcal{D}_T(\mathcal{Y}|\mathcal{X})$, while $\mathcal{D}(\mathcal{X})$ may or may not vary. This is the more common case and will therefore be considered in the following. Therefore, the label y could be different for a given feature vector $\mathbf{x}$, depending on the distribution from which it is derived. In this case, a certain biological function may have changed due to evolutionary pressures. The evolutionary distance may be a good indicator for how well the function is conserved. If this distance is small, we have reason to believe that the conditionals may not be completely different, and knowledge of one of them should then provide us with some information also about the other one.

In a nutshell, the strategies to cope with this problem can be grouped into the following classes: (a) *instance weighting methods* that perform adaptation by reweighting examples, (b) *representation-based methods* attempt to find a representation of examples such that source and target distributions become similar, (c) *ensemble methods* build a combined classifier from a set of single classifiers, (d) *prior methods* exploit prior information derived from a source model and (e) *multitask methods* learn solutions for several domains simultaneously. I will first explore some of these methods for the case where the source data is derived from a single domain and subsequently generalize some of the methods to the case where training data from multiple source domains is available. As in Section 4.1, I will use SVMs with the WD kernel Section 3.2.[115]

Single Source Domain Scenario

The principles of the methods are schematically depicted in Figure 8.1. We will first start with three baseline methods.

SouceOnly (SVM$_S$): In this case only examples from the source data set are used for training and data from the target domain is ignored completely. One may

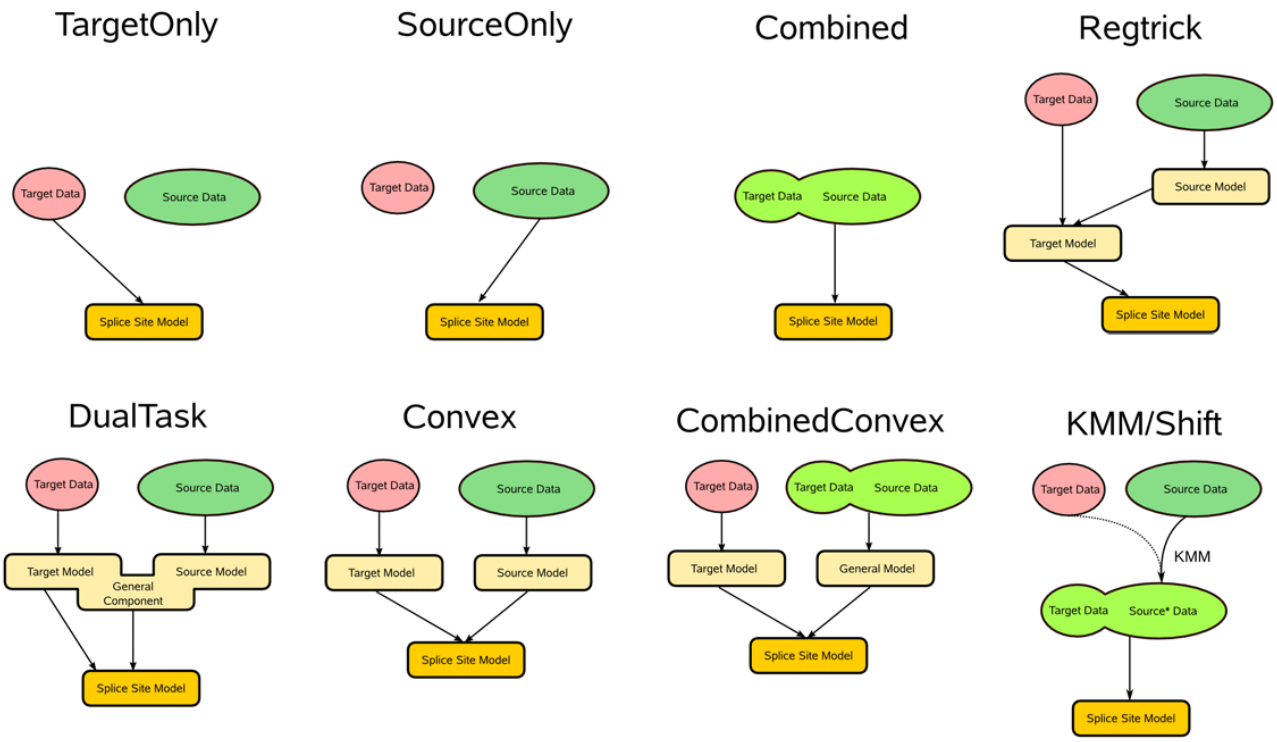


Figure 8.1.: Schematic Representation of different strategies to handle the domain adaptation problem ([185]).

thus obtain a lower bound for the expected performance, if the problem of domain adaptation is ignored while training and prediction data are derived from different distributions.

TargetOnly (SVM_T:) Now, the classifier is only trained on target examples. This method gives an estimate of the achievable performance, when no source data is available.

Union (SVM_{S+T}): The classifier is trained on target and source examples together, however, it is unaware of the origin of the examples.

The above three methods require the training of a single SVM according to Equation 3.8, and have therefore the single hyper-parameter C, that needs to be determined.

Combined (SVM_{S+T}): This is a generalization of the previous methods, where examples derived from the source domain are weighted differently than examples from the target domain. We have therefore integrated the relative importance of each domain into the loss term of the SVM. It can be adjusted by setting domain-

dependent cost parameters C_S and C_T:

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C_S \sum_{n=1}^{N_S} \xi^n + C_T \sum_{n=N_S+1}^{N_S+N_T} \xi^n \\ \text{s.t.} \quad & y^n(\langle \mathbf{w}, \Phi(\mathbf{x}^n)\rangle + b) \geq 1 - \xi^n \\ & \xi^n \geq 0 \quad \forall n = 1, \ldots, N_S + N_T \end{aligned} \tag{8.1}$$

Note, that SVM_S and SVM_T can be recovered from this formulation as the extreme settings where $C_T = 0$, or $C_S = 0$, respectively. Likewise SVM_{S+T} is resumed if $C_T = C_S$. In general, the predictor will represent a model that is sub-optimal for both domains individually, but lies in between the true models. How strongly the solution is dominated by one domain does not only depend on the values of the cost parameters, but also on the relative numbers of examples used. This method has two model parameters and requires training on the union of the training sets. Since the computation time of most classification methods increases super-linearly and full model selection may require to train many parameter combinations, this approach is computationally quite demanding.

The strategy pursued in the following two methods is to change the source distribution such that it better matches the target domain. Therefore, a weight for each example will be learnt. This can be done by a pre-processing step prior to training the actual classifier.

Kernel Mean Matching, KMM, ($SVM_{S\to T}$) The idea of the KMM is to re-weight examples from the source domain, such that the two distributions become more similar.

As discussed previously, kernel methods project the data into a reproducing kernel Hilbert space (RKHS) by applying a mapping $\Phi : \mathcal{X} \to \mathcal{H}$ related to a positive definite kernel via $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}')\rangle$. Depending on the choice of the kernel, the space of $\mathcal{H}$ may be spanned by a large number of higher order features of the data. In such cases, higher order statistics for a set of input points can be computed in $\mathcal{H}$ by simply taking the mean (i.e. , the first order statistics). In fact, it turns out that for a certain class of kernels, the mapping

$$\mu : (\mathbf{x}^1, \ldots, \mathbf{x}^N) \mapsto \frac{1}{N} \sum_{n=1}^{N} \Phi(\mathbf{x}^n) \tag{8.2}$$

is injective [66] — in other words, given knowledge of (only) the mean (the right hand side), we can completely reconstruct the set of points. For a characterization of this class of kernels, see for instance [58].

In [67] it was proposed that one could use this for covariate shift adaptation, moving the mean of a source distribution (over the inputs only) towards the mean of a target distribution by re-weighting the source training points. To obtain weights $\beta_n \in \mathbb{R}$ for each source example $\mathbf{x}^n$, the following optimization problem has to be solved (see [67] for details):

$$\begin{aligned} \min_{\beta} \quad & \frac{1}{2}\beta^\top \mathcal{K}\beta - \kappa^\top \beta \\ \text{s.t.} \quad & \beta_n \in [0, B] \\ & N_S(1-\epsilon) \leq \sum_{n=1}^{N_S} \beta_n \leq N_S(1+\epsilon) \ , \end{aligned} \tag{8.3}$$

where $\kappa_m := \frac{N_S}{N_T} \sum_{n=1}^{N_s} k(x_m, x'_n)$. The first constraint limits the scope of discrepancy between source $\mathcal{D}_S(\mathcal{X})$ and target distributions $\mathcal{D}_T(\mathcal{X})$, while the second constraint ensures that the re-weighted source distribution (i.e. $\beta(x)\mathcal{D}_T(x)$) is a probability distribution. Note that the constant B is used to determine how strongly the source data set is re-weighted.

Other than [67], we do have a certain amount of labels also for the target distribution. We make use of them by performing the re-weighting separately for each class $y \in \{\pm 1\}$. This means that instead of attempting to re-weight examples from the source sample, such that $\mathcal{D}_S(\mathcal{X}) = \mathcal{D}_T(\mathcal{X})$, we separately re-weight examples from the positive class to obtain $\mathcal{D}_S(\mathcal{X}|y=+1) = \mathcal{D}_T(\mathcal{X}|y=+1)$ and examples from the negative class to obtain $\mathcal{D}_S(\mathcal{X}|y=-1) = \mathcal{D}_T(\mathcal{X}|y=-1)$. Thus, we do not only correct a covariate shift but also to some extent for a different conditional. The resulting weight vector β can be readily plugged into the loss-term of the SVM. We will then train on the union of the re-weighted source data points and the original target data points.

Kernel Mean Matching by Translation, Shift, ($SVM_{S\to T}$) In this variant, we do not re-weight the source points, but rather we translate each point towards the mean of the target inputs:

$$\hat{\Phi}(\mathbf{x}^n) = \Phi(\mathbf{x}^n) - \alpha \left(\frac{1}{N_S} \sum_{n=1}^{N_S} \Phi(\mathbf{x}^n) - \frac{1}{N_T} \sum_{n=N_S+1}^{N_S+N_T} \Phi(\mathbf{x}^n) \right) \quad \forall n = 1, \ldots, N_S. \tag{8.4}$$

This also leads to a modified source distribution that is statistically more similar to the target distribution. It is thus used to improve performance when training the target task. Again, we calculate the shift separately for the two classes, and subsequently train on the shifted source data points and the original target examples.

Feature Augmentation ($SVM_{S\times T}$) Instead of re-weighting the examples, one can also change the weighting of individual features. In [44] a method was proposed that augments the features of source and target examples in a domain-specific way:

$$\begin{aligned} \hat{\Phi}(\mathbf{x}) &= (\Phi(\mathbf{x}), \Phi(\mathbf{x}), \mathbf{0})^\top \quad \text{for } i = 1, \ldots, N_S \\ \hat{\Phi}(\mathbf{x}) &= (\Phi(\mathbf{x}), \mathbf{0}, \Phi(\mathbf{x}))^\top \quad \text{for } i = N_S + 1, \ldots, N_S + N_T. \end{aligned} \tag{8.5}$$

The intuition behind this idea is that there exist one set of parameters that models the properties common to both sets and two additional sets of parameters that model the specifics of the two domains. It can easily be seen that the kernel for the augmented feature space can be computed as:

$$k_{AUG}(\mathbf{x}^n, \mathbf{x}^m) = \begin{cases} 2\langle \Phi(\mathbf{x}^n), \Phi(\mathbf{x}^m) \rangle & \text{if } [\![n \leq N_S]\!] = [\![m \leq N_T]\!] \\ \langle \Phi(\mathbf{x}^n), \Phi(\mathbf{x}^m) \rangle & \text{otherwise} \end{cases} \tag{8.6}$$

This means that the "similarity" between two examples is two times as high, if the examples were drawn from the same domain, as if they were drawn from different domains. Instead of the factor 2, we use a hyper-parameter B in the following.

ConvexCombination (SVM_S+SVM_T) Another straightforward idea for domain adaptation is to reuse the two optimal functions f_T and f_S as generated by the base line methods SVM_S and SVM_T and combine them in a convex manner:

$$F(\mathbf{x}) = \alpha f_T(\mathbf{x}) + (1 - \alpha) f_S(\mathbf{x}). \tag{8.7}$$

Here, $\alpha \in [0, 1]$ is the convex combination parameter that is tuned on the evaluation set. This strategy can be viewed as an application of ensemble methods, where a mixture of experts is considered and the final classification arises from a weighted vote of underlying classification models. The difference is that in our case the same expert is trained on different data sets, resulting in different models, which are subsequently combined. In the classical ensemble scenario, different methods are usually trained on the same data set. A great benefit of this approach is its efficiency.

Source-regularized Target ($\text{SVM}_{T|S}$) The idea is to use $\mathbf{w}_S$ of the source domain classifier SVM_S as a "prior" for a model that is trained on the target data alone. The source $\mathbf{w}_S$ enters the optimization problem via an additional regularization term, that enforces high similarity between the two models:

$$\mathbf{\Omega}_{\text{reg}}(\mathbf{w}_T) = \frac{1}{2}\|\mathbf{w}\|^2 - B\ s(\mathbf{w}, \mathbf{w}_S) \tag{8.8}$$

where s is a similarity measure and B denotes a hyper-parameter that determines how strongly the target solution is attracted towards the source model. In practice the optimal parameter B depends on the distance between the two domains, as well as the number of available target data.

Here, we consider the case $s(\mathbf{w}_T, \mathbf{w}_S) = \langle \mathbf{w}_T, \mathbf{w}_S \rangle$. Please note, that for $B = 1$, the above regularization term is identical to $\frac{1}{2}\|\mathbf{w}_T - \mathbf{w}_S\|^2$ up to a constant ($\frac{1}{2}\|\mathbf{w}_S\|^2$). The primal of the modified SVM looks as follows:

$$\begin{aligned} \min_{\mathbf{w}_T, \xi} \quad & \frac{1}{2}\|\mathbf{w}_T\|^2 - B\langle \mathbf{w}_T, \mathbf{w}_S \rangle + C \sum_{n=N_S+1}^{N_S+N_T} \xi^n \\ \text{s.t.} \quad & y^n(\langle \mathbf{w}_T, \Phi(\mathbf{x}_i) \rangle + b) + \xi^n - 1 \geq 0 \\ & \xi^n \geq 0 \qquad \forall n = N_S + 1, \ldots, N_S + N_T, \end{aligned} \tag{8.9}$$

where $\mathbf{w}_S$ is given in advance. It turns out that this optimization problem can be solved by slightly modifying existing SVM solvers like LibSVM. The resulting algorithm does require just as much computing time as computing the output of the target training examples using the source mode and solving the original SVM problem (see appendix for details). We therefore were able to optimize the two hyper-parameters B and C in model-selection.

Dual-task Learning ($SVM_{S,T}$) One way of extending the weighted combination approach is a variant of multitask learning [29]. The idea is to solve the source and target classification problems simultaneously and couple the two solutions via a regularization term. This idea can be realized by the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}_S, \mathbf{w}_T, \xi} \quad & \frac{1}{2}\|\mathbf{w}_S - \mathbf{w}_T\|^2 + C \sum_{n=1}^{N_S+N_T} \xi^n \\ \text{s.t.} \quad & y^n(\langle \mathbf{w}_S, \Phi(\mathbf{x}^n) \rangle + b) \geq 1 - \xi^n \qquad \forall n = 1, \ldots, N_S \\ & y^n(\langle \mathbf{w}_T, \Phi(\mathbf{x}^n) \rangle + b) \geq 1 - \xi^n \qquad \forall n = N_S + 1, \ldots, N_S + N_T \\ & \xi^n \geq 0 \qquad \forall n = 1, \ldots, N_S + N_T \end{aligned} \tag{8.10}$$

Please note that $\mathbf{w}_S$ *and* $\mathbf{w}_T$ are optimized simultaneously. The above optimization problem can be solved using a standard QP-solver. In a preliminary experiment we used the optimization package CPLEX to solve this problem, which took too long as the number of variables is relatively large. Hence, we decided to approximate the soft-margin loss using the logistic loss $l(f(\mathbf{x}), y) = \log(1 + \exp(-yf(\mathbf{x})))$ and to use a conjugate gradient method[1] to minimize the resulting objective function in terms of $\mathbf{w}_S$ and $\mathbf{w}_T$.

[1] Carl Rasmussen's `minimize` function was used.

Combination of Several Sources

Most of the above algorithms can be extended in one way or another to integrate several source domains. In this work we consider only three possible algorithms: (a) convex combinations of several domains, (b) KMM on several domains and (c) an extension of the dual-task learning approach to multi-task learning. We briefly describe these methods below:

Multiple Convex Combinations ($M\text{-}SVM_S\text{+}SVM_T$) The most general version would be to optimize all convex combination coefficients independently. If done in a grid-search-like manner, it becomes prohibitive for more than say three source domains. In principle, one can optimize these coefficients also by solving a linear program. In preliminary experiments we tried both approaches and they typically did not lead to better results than the following combination:

$$F(\mathbf{x}) = \alpha f_T(\mathbf{x}) + (1-\alpha)\frac{1}{|\mathcal{S}|}\sum_{S\in\mathcal{S}} f_S(\mathbf{x}), \tag{8.11}$$

where $\mathcal{S}$ is the set of all considered source domains. We therefore only considered this way of combining the predictions.

Multiple KMM ($M\text{-}SVM_{S\to T}$) Here, we shift the source examples of each domain independently towards the target examples, but by the same relative distance (α). Then, we train one classifier on the shifted source examples as well as the target examples.

Multi-task Learning ($M\text{-}SVM_{S,T}$) We consider the following version of multi-task learning:

$$\begin{aligned} \min_{\{\mathbf{w}_D\}_{D\in\mathcal{D}},\xi} \quad & \frac{1}{2}\sum_{D_1\in\mathcal{D}}\sum_{D_2\in\mathcal{D}} \gamma_{D_1,D_2}\|\mathbf{w}_{D_1}-\mathbf{w}_{D_2}\|^2 + \sum_i \xi^n \\ \text{s.t.} \quad & y^n(\langle \mathbf{w}_{D_j}, \Phi(\mathbf{x}^n)\rangle + b) \geq 1-\xi^n \\ & \xi^n \geq 0 \end{aligned} \tag{8.12}$$

for all examples $(\mathbf{x}^n, y^n)$ in domain $D_j \in \mathcal{S}\cup\mathcal{T}$, where $\mathcal{S}\cup\mathcal{T}$ is the set of all considered domains. γ is a set of regularization parameters, which we parametrized by two parameters C_S and C_T in the following way: $\gamma_{D_1,D_2} = C_S$ if D_1 and D_2 are source domains and C_T otherwise.

8.2. Learning Splice Site Prediction Across Organisms

Again, we consider the task of identifying acceptor splice sites within a large set of potential splice sites based on a sequence window around a site (compare Section 4.1). The idea is to consider the recognition of splice sites in different organisms: In all cases, we used the very well studied model organism *C. elegans* as the source domain. As target organisms we chose two additional nematodes, namely, the close relative *C. remanei*, which diverged from *C. elegans* 100 million years ago [157], and the more distantly related *P. pacificus*, a lineage that has diverged from *C. elegans* more than 200 million years ago [114]. As a third target organism we used *D. melanogaster*, which is separated from *C. elegans* by 990 million years [173]. Finally, we consider the plant *A. thaliana*, which has diverged from the other organisms more than 1,600 million years ago. It is assumed that a larger evolutionary distance will likely also have led to an accumulation of functional differences in the molecular splicing machinery. We therefore expect that the differences of classification functions for recognizing splice sites in these organisms will increase with increasing evolutionary distance.

Experimental Protocol

For the designed experimental comparison we had to run all algorithms many times for different training set sizes, organisms and model parameters. We chose the source and target training set as large as possible–in our case at most 100,000 examples per domain. Moreover, not for all algorithms we had efficient implementations available that can make use of kernels. Hence, to perform this study and to obtain comparable results, we had to restrict ourselves to a case were we can explicitly work in the feature space, if necessary (i.e. ℓ not much larger than two). We chose $\ell = 1$. Note, that this choice does not limit the generality of this study, as there is no strong reason, why efficient implementations that employ kernels could not be developed for all methods. The development of large scale methods, however, was not the main focus of this study.

Note that the above choices required an equivalent of about 1500 days of computing time on state-of-the-art CPU cores. We therefore refrained from including more methods, examples or dimensions.

In the first set of experiments we randomly selected a source dataset of 100,000 examples from *C. elegans*, while data sets of sizes 2,500, 6,500, 16,000, 40,000 and 100,000 were selected for each target organism. Subsequently we performed a second set of experiments where we combined several sources. For the comparison, we used 25,000 labeled examples from each of four remaining organisms to predict on a target organism. We ensured that the positives to negatives ratio is at 1/100 for all

datasets. Two thirds of each target set were used for training, while one third was used for evaluation during hyper-parameter tuning.Additionally, test sets of 60,000 examples were set aside for each target organism. All experiments were repeated three times with different training splits (source and target), except for the ones with 100,000 examples, which always used the full data set. Reported will be the average area under the precision-recall-curve (auPRC) and its standard deviation, see Section 2.6. The data and additional information will be made available for download on a supplementary website.[2]

How Different Are the Problems?

To better understand the distance between the organisms with respect to the specific task of splice site detection, we performed experiments to classify the example sequences according to their origin. For these pairwise discrimination tasks, we used almost the same method as for the splice site detection task, namely SVMs with the weighted degree kernel ($\ell = 22$). As a performance measure, we used the area under the precision-recall curve (auPRC). In this context, a higher auPRC is associated with a greater difference between the functioning of the splice mechanism. We examined the difference between true splice sites, as well as between decoy sites (i.e. other genomic sequences). For each classifier we used $1,000$ examples from each of the two investigated organisms for training (67%) and evaluation (33%). The results are shown in Table 8.1. We observe that the discrimination performance of discriminating true splice sites from different organisms correlates well with their evolutionary distance. The discrimination performance for decoy examples saturates for organisms more than 100-200 million years apart (the rest of the genome is evolving faster than important functional elements such as splice sites).

	time (10^6 years)	**auPRC+**	**auPRC-**
C. elegans - C. remanei	> 100	63%	67%
C. elegans - P. pacificus	200	93%	81%
C. elegans - D. melanogaster	990	95%	78%
C. elegans - A. thaliana	$> 1,100$	98%	77%

Table 8.1.: Summary of different distance measures between source and target organism. Reported are the time since divergence. Additionally, the performance of classifiers that distinguishes between examples of two organisms. The performance measure is give as auPRC+ for a classifier that distinguished between true splice sites and auPRC- for a classifier that distinguished between decoy sites

[2] `http://www.fml.mpg.de/raetsch/projects/genomedomainadaptation`

Results: Single Source Domain

The detailed results are given in Figure 8.4, where we show the median auPRC of the methods SVM_T, SVM_S, $\text{SVM}_{S\to T}$, SVM_{S+T}, SVM_S+SVM_T, $\text{SVM}_{S\times T}$ and $\text{SVM}_{S,T}$ for the considered tasks. The summary is given in Figure 8.2, where we illustrate, which method performed best (green), similarly well (within a confidence interval of $\sigma/\sqrt{n}$) as the best (light green), considerably worse than the best (yellow), not significantly better than the worst (light red) or worst (red). From these results we can make the following observations:

1. Independent of the task, if there is very little target data available, the training on source data performs much better than training on the target data. Conversely, if there is much target data available then training on it easily outperforms training the source data.

2. For a larger evolutionary distance of the target organisms to source organism *C. elegans*, a relatively small number of target training examples for the SVM_T approach is sufficient to achieve similar performance to the SVM_S approach, which is always trained on 100,000 examples. We call the number of target examples with equal source and target performance the break-even point. For instance, for the closely related organism *C. remanei* one needs nearly as many target data as source data to achieve the same performance. For the most distantly related organism *A. thaliana*, less than 10% target data is sufficient to outperform the source model.

3. In almost all cases, the performance of domain adaption algorithms is considerably higher than source (SVM_S) and target only (SVM_T). This is most pronounced near the break-even point, e.g. 3% improvement for *C. remanei* and 14% for *D. melanogaster*.

4. Among the domain adaptation algorithms, the dual-task learning approach ($\text{SVM}_{S,T}$) performed most often best (12/20 cases). The convex combination approach (SVM_S+SVM_T) performed the second most often best (5/20).

From our observations, we can conclude that the simple convex combination approach works surprisingly well. It is only outperformed by the dual-task learning algorithm, which performs consistently well for all organisms and target training set sizes.

Results: Multiple Sources

In a second set of experiments, we considered the three algorithms combining several sources. The results are given in Figure 8.4 and a summary in Figure 8.3. We can make the following observations:

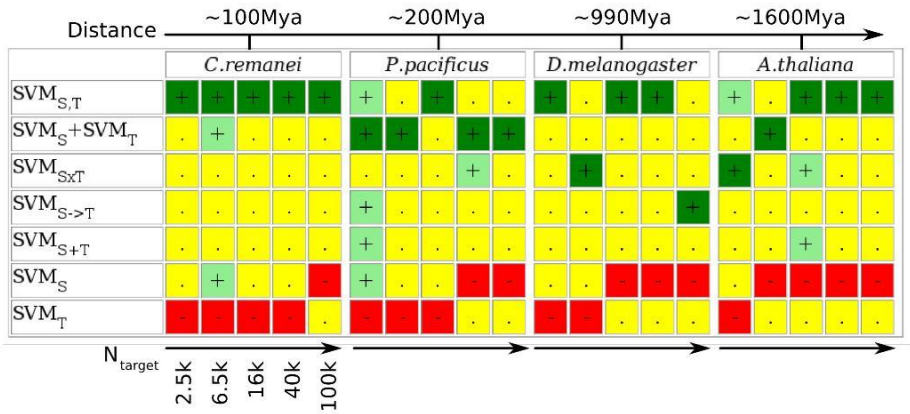


Figure 8.2.: Summary of determined performances of each presented method. Each column contains 5 sub-columns, which correspond to ascending target data set sizes (2,500, 6,500, 16,000, 40,000, 100,000). The method with the highest auPRC score for a given organism and target data set size is assigned a dark green background. Methods that do not perform significantly worse (within a confidence interval of $\sigma/\sqrt{n}$) are shown in light green. Accordingly, the worst performer is shown in red. The remaining methods, that do not fall in any of the above mentioned categories are shown in yellow. This figure above summarizes the methods based on the single-source domain approach.

1. Relative to the single source algorithms, these algorithms perform worse, if the additional source organisms are further away from the target organism than the source used by the single source algorithm. For instance, for *C. remanei* this is expected as fewer training examples of the closely related *C. elegans* organism are available.

2. For distantly related target organisms, such as *D. melanogaster* and *A. thaliana* the usage of multiple sources can lead to drastic improvements relative to the single source algorithms, in particular for small target training set sizes. This is in particular noteworthy in the case of *A. thaliana*, where all four source organisms have similar distance to the target organism. We can therefore conclude that a more general model can be learnt from different source organisms of similar distance as compared to a single source organism.

3. Among the multiple source algorithms, convex combinations and multi-task learning are the most successful ones. The first leads to the best results in 4/20 case and the latter in 11/20 cases.

From these observations we can conclude that it pays off to use diverse multiple sources, if there is no very close relative available. The multi-task learning algorithm outperforms the other methods for distantly related organisms.

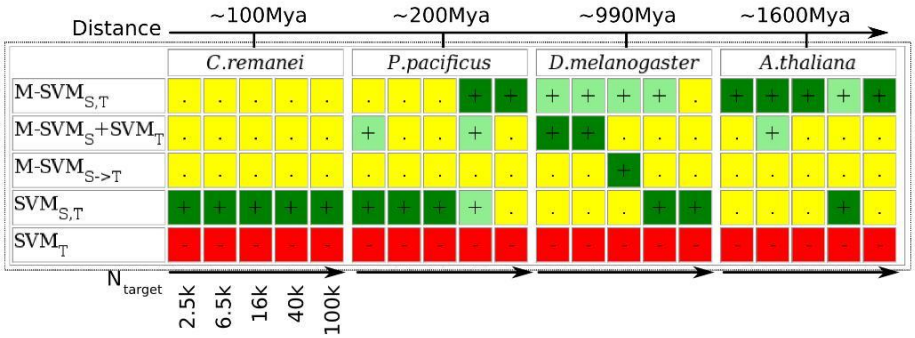
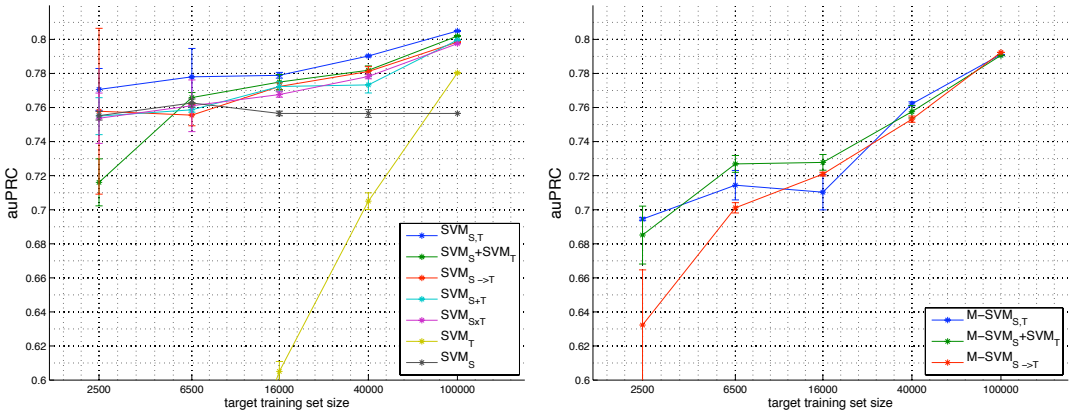


Figure 8.3.: Summary of performance of multi-source domain methods following the scheme defined in Figure 8.2. Briefly, ascending target data set sizes are contained sub-columns, best performing method is shown in green, worst performing method is shown in red, while the remaining colors denote methods in between. For comparison we also included the target only method SVM_T and the dual task method $SVM_{S,T}$ as the best single source method.

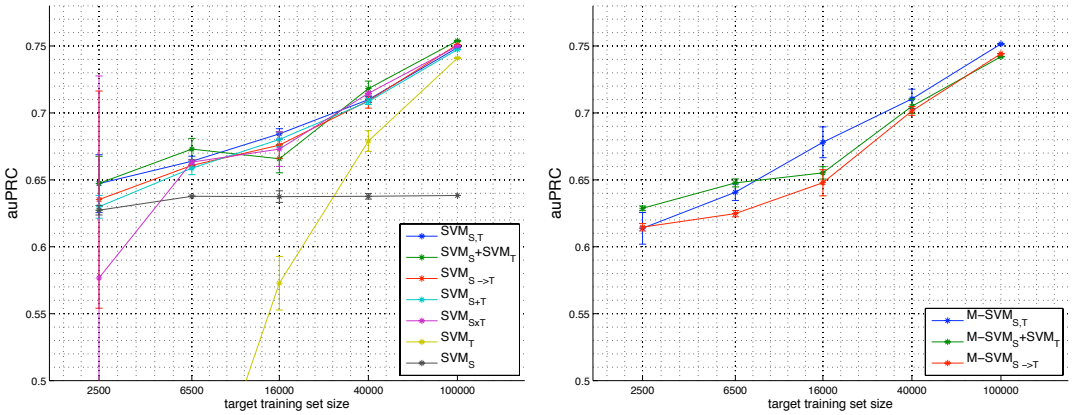
8.3. Summary and Discussion

We studied the problem of domain transfer for a supervised classification task in mRNA splicing. We consider a number of recent domain transfer methods from machine learning, including some that are novel, and evaluate them on genomic sequence data from model organisms of varying evolutionary distance. Based on the observations from our experiments, we can conclude that all domain adaptation algorithms lead to considerable improvements over the source and target models. The improvement over baseline methods is most pronounced if the source organism is only distantly related to the target organism.

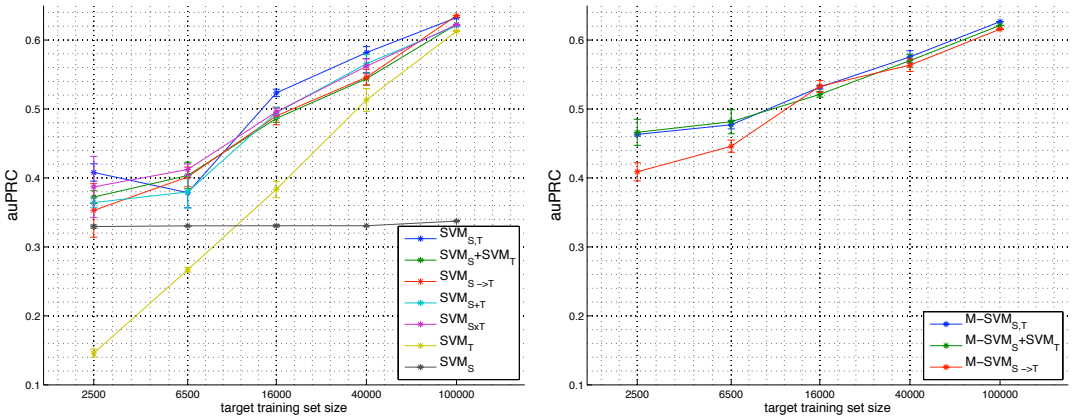
C. remanei (closely related to *C. elegans*)



P. pacificus (relatively closely related to *C. elegans*)



D. melanogaster (distantly related to *C. elegans*)



A. thaliana (most distantly related to *C. elegans*)

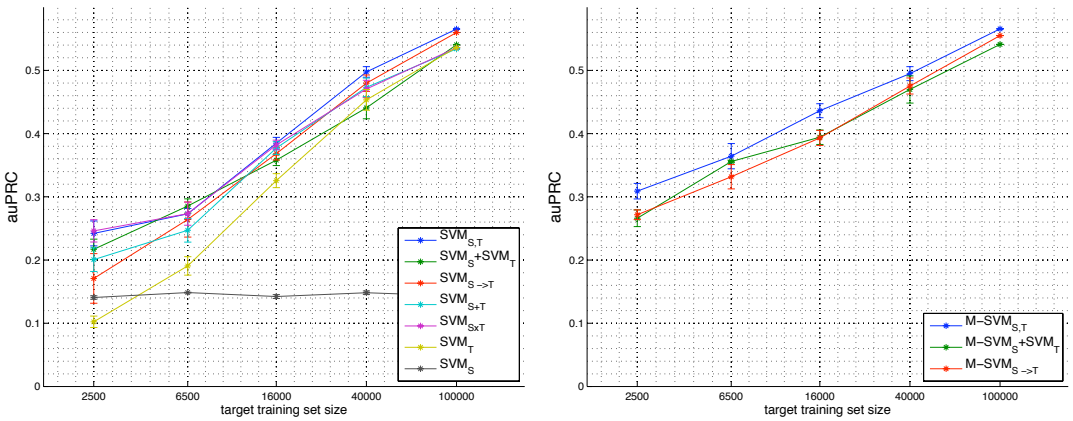


Figure 8.4.: Area under the precision-recall curve for varying target training set sizes for the four considered organisms. Shown are the median values over up to three different training set splits and the error bars indicate the confidence intervals.

9. Conclusion and Outlook

The total body of protein-coding genes encompasses a large part of the most fundamental functional elements of an organism's genome. The lack of complete and precise gene catalogues is therefore a remarkable fact, in particular when considering a model organism like *C. elegans* for which a highly accurate genome sequence has been available for more than ten years, and, during the same time, many systematic efforts have been undertaken to collect experimental evidence of genes. The challenge of the annotation task in eukaryotic genomes arises from the genomic complexity conveyed by large introns, intricate patterns of alternative splicing, as well as a multitude of ancient, degraded gene fragments, i.e. pseudo-genes.

To improve existing annotations as well as to provide high-quality initial annotations for newly sequenced genomes, we developed the computational gene finder mGene. In an objective comparison with state-of-the-art contemporary gene finders performed by an independent consortium, our system indeed proved to be particularly accurate. A combination of several features may account for the good performance of mGene, most importantly:

First, we applied a comprehensive biological model, including transcription start and stop sites to predict UTRs and poly-adenylation sites to support the identification of the transcript 3' end as well as the stop codon. Optionally, we model trans-splicing and operons, which are specific to nematodes, thus improving the recognition of transcript starts.

Second, for gene finding it is pivotal to detect the genomic signals with highest precision. To be able to employ powerful, but computationally demanding SVMs, we decoupled the signal detection tasks from the gene structure prediction problem. The detectors are designed with particular care, employing a combination of higher-order DNA sequence kernels. Treating the binary classification tasks for signal detection independently allowed us to take advantage of millions of available training examples. The potential disadvantage of losing global optimality is apparently compensated for, by the gain resulting form the reduction of one big optimization problem to several more specific optimization problems, which each can be solved very thoroughly.

Third, for the integration of the individual signal predictions we use HSM-SVMs, i.e. a discriminative structured output learning technique with semi-Markovian properties. In contrast to conventional generative methods like HMMs, it allows the use

of overlapping features and the integration of feature inputs that do not correspond to properly scaled probability values.

With our system we generated new gene catalogues for *C. elegans*, as well as five other newly sequenced nematodes. These sets have become long-term features in all the wormbase genome browsers and are used to support future gene curation by the wormbase consortium. Additionally, we put particular care in an objective and unbiased assessment of the resulting genome-wide predictions in comparison to other annotations. During this analysis, we exposed a number of common pitfalls that are often ignored. For example, if *training sets* contain not only experimentally confirmed genes, but also previously predicted ones, artifacts can easily be propagated from one annotation to the next one. In the case of *C. elegans*, it is likely that a preference of the initial employed gene finder to predict ORFs of length 1000 bp, has been conserved for more than 200 iterations of annotation releases, and many passes of different computational gene finders. Part of the problem arises from the fact that new predictions are often *evaluated* using old annotations that also include predictions.

For *C. elegans*, we predicted between 1000 and 2000 *novel* genes that were not contained in the current annotation. We confirmed the transcription of many of these sequences in a set of validation experiments. This finding suggests that a considerable portion of protein-coding genes may be missed in the wormbase annotation. Further evidence supporting this assumption comes from recent experimental results of a deep sequencing effort [76]. A thorough comparison of this type of data with our predictions is pending. It will offer a unique opportunity to further investigate the quality of computational gene predictions.

To increase the usability of our system, we developed a web server that allows to annotate other genomes. With this service we provide initial trained models for several organisms, including *Drosophila melanogaster, Saccharomyces cerevisiae, Arabidopsis thaliana, Aspergillus nidulans* and others. Future work will be necessary to refine some of this preliminary models, and to add more organisms to the list. For the application of our system to the human genome, some changes to the Viterbi-like decoding algorithm are necessary to efficiently handle very long introns. This is already on the way with the *Diplom* thesis of Peter Niermann [108].

In the last few years, the introduction of massively parallel sequencing platforms for Next Generation Sequencing, including RNA-Seq protocols, dramatically changed the studies of genomes. With RNA-Seq methods, it has become possible to obtain a very detailed picture of transcriptomes at single-nucleotide resolution, which has already led to the detection of many previously unknown coding sequences and alternative splice forms [106, 162, 180]. The high sensitivity of the method allows to detect even rare isoforms, which are particularly short and expressed at low levels. It remains under discussion which fraction of these incidences can actu-

ally be attributed to *background noise*, i.e. *accidentally* transcribed sequences without function [174]. In any case, this method additionally offers the possibility to *quantitatively* measure the expression of genes. Going beyond a *static* view on the transcriptome, it is now possible to observe the *dynamics* of gene expression, when studying different cell types, or cell responses to various conditions. In the light of this revolution in sequencing technologies, the advancement of computational gene finders may seem obsolete. However, various limitations and biases of the RNA-Seq measurements render the analysis a difficult task. On the other hand, the adaptive machine learning techniques employed in mGene require only relative simple extensions to accommodate RNA-Seq data as additional hints for transcribed sequences [12]. By combining the strength of a gene finder that captures biological rules in a state model and detects various DNA signals with the wealth of experimental evidence for transcription, it is thus possible to arrive at far more accurate predictions than with any of the two techniques alone [12]. In the era of high-throughput sequencing computational gene finders may therefore change their appearance, however, their contribution to the identification of gene structure will not vanish.

Appendices

A. Two-stage Learning of HM-SVMs

Considering signal and content contributions, i.e. features that depend on label-label pairs, the kernel function of Equation 3.41

$$\begin{aligned}\mathcal{K}((\mathbf{x}^n,\mathbf{y}^n),(\mathbf{x},\mathbf{y})) &= \sum_{s=1}^{S}\sum_{i=1}^{L+1}\sum_{i'=1}^{L'+1}\mathbb{I}(y_i^n,q_s)\mathbb{I}(y_{i'},q_s)\mathcal{K}_s((\mathbf{x}^n,i),(\mathbf{x},i')) + \\ &+ \sum_{c=1}^{C}\sum_{i=1}^{L+1}\sum_{i'=1}^{L'+1}\mathbb{I}(y_i^n,q_c)\mathbb{I}(y_{i'},q_c)\mathbb{I}(y_{i-1}^n,q_c')\mathbb{I}(y_{i'-1},q_c')\mathcal{K}_c((\mathbf{x}^n,i),(\mathbf{x},i'))\ .\end{aligned}$$

With this kernel, the scoring function can be rewritten as:

$$G_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}) = \sum_{s=1}^{S}\sum_{i'=1}^{L'+1}\mathbb{I}(y_{i'},q_s)\hat{f}^s(\mathbf{x},i') + \sum_{c=1}^{C}\sum_{i'=1}^{L'+1}\mathbb{I}(y_{i'},q_c)\mathbb{I}(y_{i'-1},q_c')\hat{f}^c(\mathbf{x},i')\ ,$$

with

$$\begin{aligned}\hat{f}^s(\mathbf{x},i') &= \sum_{n=1}^{N}\sum_{\mathbf{y}'\in\mathcal{Y}}\alpha^n(\mathbf{y}')z(\mathbf{y}')\sum_{i=1}^{L+1}\mathbb{I}(y_i^n,q_s)\mathcal{K}^s((\mathbf{x}^n,i),(\mathbf{x},i')) \\ \hat{f}^c(\mathbf{x},i') &= \sum_{n=1}^{N}\sum_{\mathbf{y}'\in\mathcal{Y}}\alpha^n(\mathbf{y}')z(\mathbf{y}')\sum_{i=1}^{L+1}\mathbb{I}(y_i^n,q_s)\mathbb{I}(y_{i'-1},q_s')\mathcal{K}^s((\mathbf{x}^n,i),(\mathbf{x},i'))\ .\end{aligned}$$

Finally, Equation 3.44 becomes:

$$G_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}) = \sum_{i'=1}^{L'+1}\left(\sum_{s=1}^{S}\mathbb{I}(y_{i'},q_s)t_{\boldsymbol{\theta}}^s(f^s(\mathbf{x},i')) + \sum_{c=1}^{C}\mathbb{I}(y_{i'},q_c)\mathbb{I}(y_{i'-1},q_c')t_{\boldsymbol{\theta}}^c(f^c(\mathbf{x},i'))\right)\ .$$

B. Genome-wide Splice Site Prediction

Data Set Generation

To generate genome-wide data sets for splice site predictions we used all available EST information for the respective organisms from the dbEST [17] as of February 28, 2007, see Table B.1. We additionally used EST and cDNA sequences available from wormbase [73] for worm, from the Berkeley Drosophila Genome Project (BDG) [154, 155] for fly, from the RIKEN Arabidopsis Genome Encyclopedia (RARGE) [126] for cress, from Ensemble [77] for fish, and from the Mammalian Gene Collection (MGC) [61] for fish and human. The characteristics of the resulting splice site data sets are summarized in Table B.2.

	nb of dbEST sequences	additional files
Worm	346,064	wormbase (`confirmed_genes.WS170`)
Fly	514,613	BDG (`na_EST.dros`,`na_dbEST.same.dmel`)
Cress	1,276,130	RARGE (`cDNA_flanking_050524.txt`,`cDNA_full_reading_050524.txt`)
Fish	1,168,572	Ensemble (`Danio_rerio.ZFISH6.43.cdna.known.fa`), MGC (`dr_mgc_mrna.fasta`)
Human	7,915,689	MGC (`hs_mgc_mrna.fasta`)

Table B.1.: Sequences used to generate genome-wide data sets for five different organisms.

	Worm		**Fly**		**Cress**		**Fish**		**Human**	
	Acceptor	Donor	Acceptor	Donor	Acceptor	Donor	Acceptor	Donor	Acceptor	Donor
Training total	1,105,886	1,744,733	1,289,427	2,484,854	1,340,260	2,033,863	3,541,087	6,017,854	6,635,123	9,262,241
Fraction positives	3.6%	2.3%	1.4%	0.7%	3.6%	2.3%	2.4%	1.5%	1.5%	1.1%
Evaluation total	371,897	588,088	425,287	820,172	448,924	680,998	3,892,454	6,638,038	10,820,985	15,201,348
Fraction positives	3.6%	2.3%	1.4%	0.7%	3.6%	2.3%	0.7%	0.4%	0.3%	0.2%
Testing total	364,967	578621	441,686	851,539	445,585	673,732	3,998,521	6,822,472	11,011,875	15,369,748
Fraction positives	3.6%	2.3%	1.4%	0.7%	3.5%	2.3%	0.7%	0.4%	0.3%	0.2%

Table B.2.: Characteristics of the genome-wide data sets containing true and decoy acceptor and donor splice sites. The sequence length in all sets is 141 bp, for acceptor splice sequences the consensus dimer `AG` is at position 61, for donor `GT/GC` at position 81. The negative examples in training sets of fish and human were sub-sampled by a factor of three and five, respectively.

Optimal Model Parameters

	window	C	d	$S(l)$	Θ_+	Θ_-	type	method
human	199 +[−60, +80]	3	22	0	–	–	acceptor	WD-SVM
	199 +[−60, +80]	3	22	0.3	–	–	acceptor	WDS-SVM
	199 +[−25, +25]	–	3	–	10	1000	acceptor	MCs
	199 +[−80, +60]	3	22	0	–	–	donor	WD-SVM
	199 +[−80, +60]	3	22	0.3	–	–	donor	WDS-SVM
	199 +[−17, +18]	–	3	–	0.01	1000	donor	MCs
plant	199 +[−60, +80]	3	22	0	–	–	acceptor	WD-SVM
	199 +[−60, +80]	3	22	0.5	–	–	acceptor	WDS-SVM
	199 +[−80, +80]	–	4	–	10	1	acceptor	MCs
	199 +[−80, +60]	3	26	0	–	–	donor	WD-SVM
	199 +[−80, +60]	3	22	0.5	–	–	donor	WDS-SVM
	199 +[−80, +80]	–	4	–	10	10	donor	MCs
worm	199 +[−60, +80]	3	22	0	–	–	acceptor	WD-SVM
	199 +[−60, +80]	3	22	0.3	–	–	acceptor	WDS-SVM
	199 +[−25, +25]	–	3	–	10	1000	acceptor	MCs
	199 +[−80, +60]	3	22	0	–	–	donor	WD-SVM
	199 +[−80, +60]	3	22	0.3	–	–	donor	WDS-SVM
	199 +[−17, +18]	–	3	–	0.01	1000	donor	MCs
fly	199 +[−60, +80]	3	22	0	–	–	acceptor	WD-SVM
	199 +[−60, +80]	3	22	0.3	–	–	acceptor	WDS-SVM
	199 +[−60, +60]	–	3	–	0	1000	acceptor	MCs
	199 +[−80, +60]	3	22	0	–	–	donor	WD-SVM
	199 +[−80, +60]	3	22	0.3	–	–	donor	WD-SVM
	199 +[−60, +60]	–	3	–	0	1000	donor	MCs

Table B.3.: Optimal parameter settings for genome-wide splice site detectors.

Additional Results

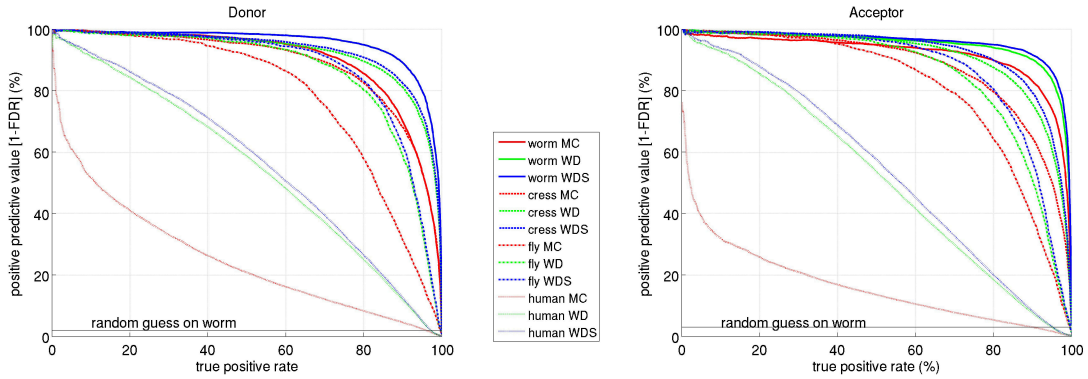


Figure B.1.: Precision Recall Curve for the three methods MC, WD, WDS, estimated on the genome-wide data sets for worm, fly, cress, fish, and human in a nested cross-validation scheme. In contrast to the ROC, the random guess in this plot corresponds to a horizontal line, that depends on the fraction of positive examples in the test set (e.g. 2% and 3% in the case of the worm acceptor and donor data sets, respectively).

C. mGene at the nGASP challenge

Training and Test Regions

The training and test sequences were chosen from a total of 96 1Mb windows that cover the complete genome, except for six smaller left-over segments at the high-coordinate ends of the chromosomes. Each region was classified according to its gene density and level of conservation with respect to strong WABA matches [83] to *C. briggsae*. Eventually, for each of the four combinations of high/low conservation and high/low gene density, two regions were randomly selected from the autosomal chromosomes. Additionally, two high-conservation, low-gene density regions were selected from the X-chromosome.

Additional Data

Additional data included:

Category 2 — conservation information: DNA sequence for *C. briggsae* ('cb1' assembly) and for *C. remanei* ('pcap2' assembly) [157, 186] (www.genome.wustl.edu; R. Wilson, pers. comm), as well as a multi-genome alignment between *C. elegans*, *C. briggsae* and *C. remanei* for the training regions, generated by applying MLAGAN version 1.21 [25].

Category 3 — protein, EST, and cDNA sequence alignments; For this category the organizers provided the amino acid sequences of 42,496 proteins that have BLAST [3] matches to the test or training regions. Matches to proteins encoded by genes in the test regions were excluded. The BLAST matches were made by running BLAST with an E-value cut-off of 0.1 against proteins from the known protein universe (taken from uniprot and several other databases, see [34] for further details). Additionally the nucleotide sequences of 20,141 C. elegans ESTs/cDNAs that have BLAT matches to the test or training regions were provided, as well as the coordinates of the BLAST and BLAT matches in the test and training regions.

Optimal Parameter Set

For the HSM-SVM training, we found the following hyper-parameter settings to be suitable (compare Equation 5.12):
$C_1^{small} = 0$,
$C_{1,trans}^{small} = 0.010$,
$C_2^{small} = 0.001$,
$C_{2,trans}^{small} = 0.010$,
$C_1^{smooth} = 0.010$,
$C_2^{smooth} = 0.010$.

We also noticed that we prediction accuracy could be significantly improved by separately tuning a single parameter, namely, the transition scores from the state recognizing the cleave signal ($t_4^{cleave,*}$), which strongly influences the recognition of gene ends. This parameter was tuned to achieve the best performance on the 247 sub-regions.

segment1	segment2	loss
intergenic	intercistronic	0
intergenic	coding exon	3
intergenic	intron	3
intergenic	5' UTR exon	2
intergenic	transexon	2
intergenic	3' UTR exon	2
intergenic	polya tail	2
utr3exon	polya tail	0
intron	coding exon	3
utr5exon	transexon	0

Table C.1.: Mismatch penalty for mGene Hamming loss. All other mismatches incur penalty 1.

Length Features $l \in \mathcal{L}$	Range [bp]	Length Features $l \in \mathcal{L}$	Range [bp]
5'UTR-exon	1 - 800	*trans-exon*	1 - 1000
single-exon	100 - 800	*polyA-tail*	0 - 300
first-exon	3 - 800	*intron*	30 - 21,000
middle-exon	3 - 800	*intergenic*	0 - 21,000
last-exon	1 - 800	*intergenic-long*	10,450 - 10,550
3'UTR-exon	1 - 800	*intercistronic*	1 - 5000

Table C.2.: Length features, and corresponding length ranges as chosen for gene prediction in *C. elegans*.

Additional Results

As an indication that our own numerical evaluation of gene finder accuracies in the nGASP setting is appropriate, we reproduce an excerpt from the results table of the nGASP evaluation [34] in Table C.3. Shown are values for the same gene finders as in Table 6.2 of the main text, except for the improved versions of mGene, which did not participate in nGASP. While there are several numerical differences to our own evaluation on the transcript and gene levels, they are small, and the ranking of the methods by their accuracy generally remains unchanged.

		Base			Exon			Transcript			Gene		
Cat.	Method	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$	Sn	Sp	$\frac{Sn+Sp}{2}$
1	mGene.init (dev) (v3)	96.9	91.6	**94.2**	84.2	78.6	**81.4**	43.5	40.5	**42.0**	53.4	44.8	49.1
1	Craig	95.6	90.9	93.2	80.2	78.2	79.2	35.7	36.3	36.0	43.8	37.8	40.8
1	Eugene	94.0	89.5	91.7	80.3	73.0	76.6	49.1	28.8	39.0	60.2	30.2	45.2
1	Fgenesh	98.2	87.1	92.7	86.4	73.6	80.0	47.1	34.6	40.9*	57.8	35.4	46.6
1	Augustus (v1)	97.0	89.0	93.0	86.1	72.6	79.3	50.1	28.7	39.4*	61.1	38.4	**49.7**
2	mGene.multi (dev) (v2)	97.7	90.9	**94.3**	85.8	78.3	**82.1**	51.2	40.9	**46.1**	62.7	43.8	**53.3**
2	N-Scan	97.4	88.1	92.7	83.5	70.8	77.2	39.2	27.7	33.4	48.1	28.4	38.2
2	Eugene	96.2	87.5	91.9	82.8	72.8	77.8	50.3	30.2	40.2	61.7	31.4	46.5
3	mGene.seq (dev) (v3)	98.7	91.9	95.3	91.0	80.6	85.8	57.7	48.0	52.9	70.6	51.1	60.9
3	Gramene (v1)	98.2	95.4	**96.8**	88.5	71.8	80.1	41.7	19.6	30.6	48.7	37.2	42.9
3	Fgenesh++	97.6	89.7	93.6	90.4	80.9	85.7	65.5	53.4	**59.5**	78.3	54.2	**66.3***
3	Augustus (v1)	99.0	90.5	94.7	92.5	80.2	**86.3**	68.3	47.1	57.7	80.1	51.8	66.0*
4	Evigan	99.3	89.6	94.4	91.1	82.3	86.7	64.2	52.4	58.3	80.7	52.7	66.7
4	Jigsaw (v1)	98.9	93.2	**96.1**	90.5	87.4	**88.9**	63.6	60.2	**61.9**	79.9	61.0	**70.5**

Table C.3.: nGASP evaluation of selected gene predictors. Sensitivities (Sn) and specificity (Sp) are taken from [34]; to facilitate comparisons, we further report their averages. The numbers slightly deviate from our own evaluation on the transcript and gene level due to minor differences in the evaluation criteria. These differences, however, do not change the ranking except in two cases, where the performances are very close together (relevant results are marked with an asterisk). For the sake of completeness this table also includes performance estimates for Category 4 submissions, including Jigsaw and Evigan [96]. Note, however, that these performance measures are not directly comparable to the other categories, because more training data and different test data were used.

D. Genome-wide Predictions for C. elegans

Data Set Generation

The genome annotation version WS180 was downloaded from `ftp://ftp.wormbase.org/pub/wormbase/genomes/c_elegans/genome_feature_tables/GFF3/elegansWS180.gff3`. For the reconstructed gene set Anno180, we parsed all entries with the source (second column) specified as `Coding_transcript` and the type (third column) specified as `gene`, `mRNA`, `exon`, `CDS`, `intron`, `five_prime_UTR`, or `three_prime_UTR`. Additionally, we parsed all pseudo-genes (source: `Pseudogene` or `history`; type: `pseudogene`), and all non-coding RNAs (source: `miRNA`, `ncRNA`, `rRNA`, `scRNA`, `snRNA`, `snlRNA`, `snoRNA` or `Non_coding_transcript`; type: `ncRNA`) from the WS180 annotation.

Verification of Gene Predictions by RT-PCR

We isolated mRNA from a mixed population of wild type *C. elegans* (TRIzol, Invitrogen; PolyATtract mRNA Isolation System III, Promega), which was then reverse-transcribed using SuperScript II & III reverse transcriptase (Invitrogen) with random primers (Invitrogen). Subsequently, touchdown PCR was performed using TAQ DNA-Polymerase S (Genaxxon). We screened for sufficient PCR amplification at final annealing temperatures of 55 and 52°C. The PCR products were then analyzed by agarose gel electrophoresis. DNA fragments were excised from the gel and purified using Wizard SV Gel and PCR Clean-Up System (Promega). The purified DNA fragments were sequenced from both ends by Big Dye Terminator sequencing (Applied Biosystems, Foster City, CA).

E. mGene.web

Data Types

We use the following data types for data exchange between the tools: FASTA (genomic DNA), GFF3 (genome annotation and gene prediction), SPF (signal prediction format), and CPF (content prediction format). Files in these formats can either be provided by uploading or by running one of the tools. Additionally, we have some data formats representing internal data structures, including the Genome Information Object (GIO) to describe a genome-wide sequence with all its contigs or chromosomes, the Annotation Gene Structure (AGS), as an efficient internal representation of genes, the Trained Signal Predictor (TSP), the Trained Content Predictor (TCP) and the Trained Gene Predictor (TGP) that include all learnt parameters necessary to predict a given signal, content or gene structure, respectively. Eventually the Trained mGene Predictor (TmGP) contains all above mentioned parameters and can be used to predict genes from scratch when given a DNA sequence. (Note, that the TGP object contains the parameters learnt during training layer 2 only, while the TmGP contains *all* parameters necessary to predict genes.)

mGene.web Modules

In the following the individual tools are explained in more detail and more instructions are provided at `http://www.mgene.org/webservice`.

Data preparation

GenomeTool needs a file in FASTA format containing genomic sequences as input, which allows it to create a genome object, stored in a genome information object (GIO), to be used by other mGene tools.

GFF2Anno reads an annotation file in GFF3 format and uses it to generate an annotated gene structure (AGS) .

Signal prediction

Anno2SignalLabel uses an AGS to collect labeled genomic positions for the selected genomic signal. Possible signals include transcription start and stop sites, translation initiation and termination sites, as well as donor and acceptor splice sites. It uses the regions covered by annotated features to generate negative examples at all consensus positions unless they were annotated as true sites. The output is a file in signal prediction format (SPF) providing chromosome/contig name, position, strand, and the label of the example.

SignalTrain trains a signal predictor using SVMs with pre-selected kernels for each signal. Input is a genome information object and an SPF file with labeled genomic positions. The output is a trained signal predictor (TSP) that can be used with SignalPredict to perform predictions on genomic sequences.

SignalPredict uses a GIO and TSP to predict signals on the given DNA sequences. The output is given in signal prediction format (SPF).

SignalEval takes a label file and a prediction file (both SPF files) as input and computes several accuracy measures for the predictions, including the areas under the Receiver-Operator-Curve (ROC) and the Precision-Recall-Curve (PRC). This tool is useful for prediction quality monitoring.

Content prediction

Anno2ContentLabel collects labeled genomic segments for the selected content types, analogous to Anno2SignalLabel. Possible content types include 5' UTR, exonic, intronic, 3' UTR, and intergenic. Any segment included that is not of the specified type is used as a negative example. The output is a file in content prediction format (CPF) providing chromosome/contig name, start position, end position, strand, and the label of the example.

ContentTrain is analogous to SignalTrain, with a GIO and an SPF file as inputs and a trained content predictor object (TCP) as output.

ContentPredict is analogous to SignalPredict, with a GIO and a trained content predictor (TCP) as input and an SPF file as output.

ContentEval, analogous to SignalEval, takes a CPF and SPF file as input and performs an evaluation.

Gene prediction

GeneTrain trains the second layer of mGene.web. Based on the GIO, genome-wide predictions for all relevant signals and content types, and a set of annotated genes, GeneTrain learns to predict gene structures from genomic DNA. The output is an internal data structure containing the trained gene predictor (TGP) that can be used with mGenePredict to predict genes.

GenePredict uses the TGP (either from the current history or from a list of pre-trained predictors) as well as genome-wide signal and content predictions to predict genes from the provided DNA sequences. The output is provided as a GFF3 file.

GeneEval takes two GFF3 files, one containing an annotation, the other the genome-wide gene predictions, and evaluates the prediction performance by comparing the two annotations. Note that the annotated genes should be distinct from the annotated genes used for training, otherwise the training error will be reported. Evaluation criteria include sensitivity and specificity on nucleotide, exon, and gene levels [34, 141, for further details].

ComposeMGenePredictor bundles all necessary trained signal, content, and gene predictor objects into a trained mGene predictor that can be used with mGenePredict to predict genes.

DecomposeMGenePredictor decomposes a trained mGene predictor into its components, i.e. the individual predictors.

A more detailed description of the individual tools can be found in the supplementary webpage.

mGene.web **Workflows**

In addition to the individual modules, we provide several workflows, to facilitate the process of gene finding. These include a workflow that completes the complete training procedure Figure E.1 and one for gene prediction Figure E.2.

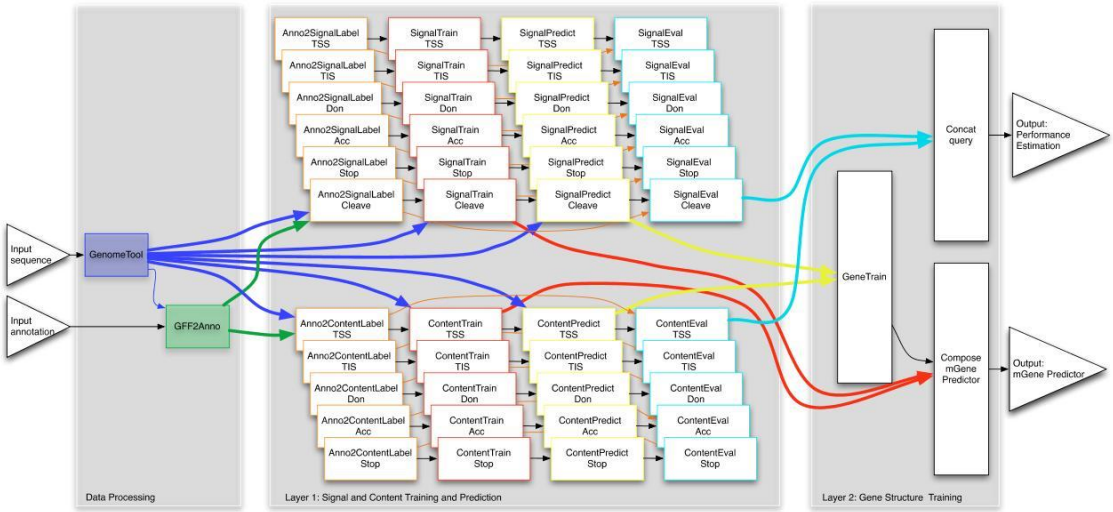


Figure E.1.: Workflow to train a new model for gene prediction. It requires the DNA sequence (FASTA format) and a partial annotation (GFF3) as input. Subsequently, all steps are automatically performed, including the training and prediction of all signal and content types, and the training of the gene structure predictor. Output is a TmGP Object and an evaluation file.

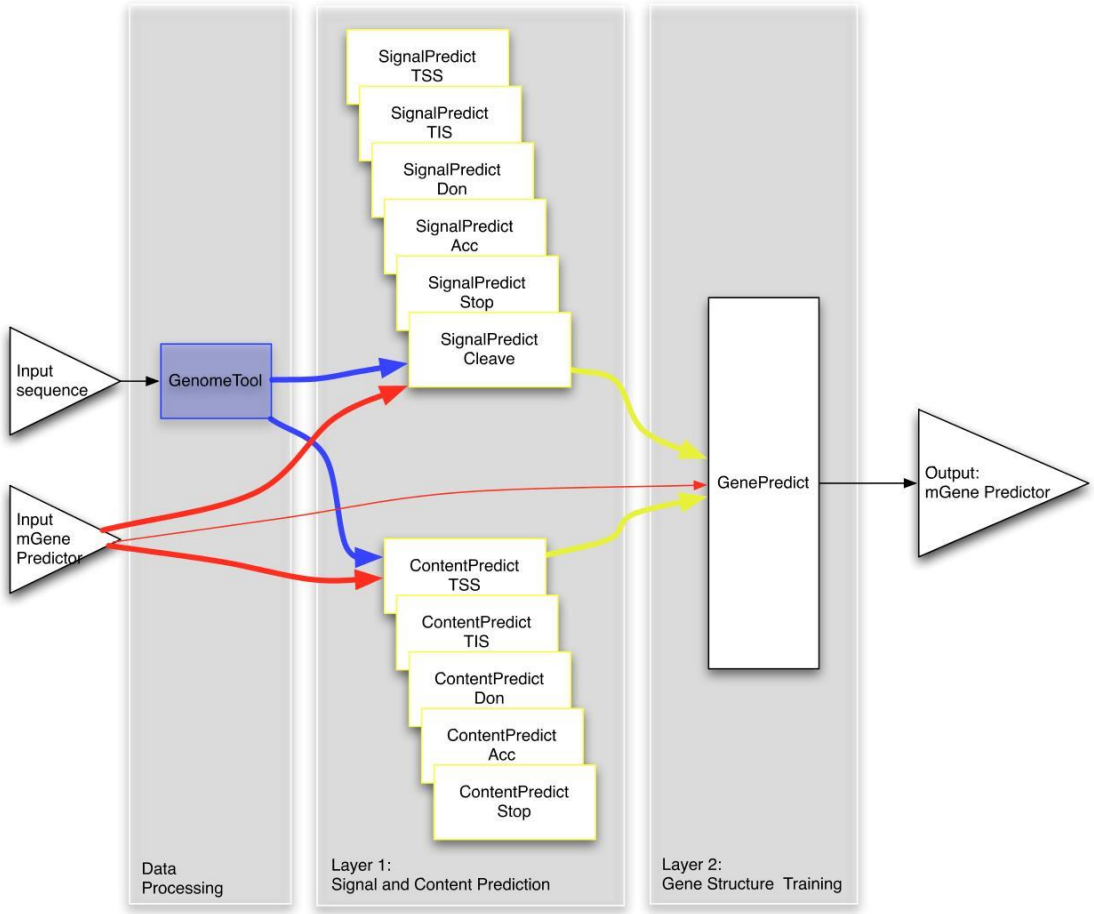


Figure E.2.: mGene.web workflow for the prediction of genes. Input is the considered DNA sequence and a TmGP, which can also be chosen from a list of pre-trained models. It outputs a GFF3 file containing the corresponding gene predictions.

F. Additional Tables for Domain Adaptation Experiments

The results depicted in Figure 8.4 are also given in the following tables:

Tabular Single-Source Results

	2500	6500	16000	40000	100000
C. remanei					
$SVM_{S,T}$	**77.06 (±2.13)**	**77.80 (±2.89)**	**77.89 (±0.29)**	**79.02 (±0.09)**	**80.49 (±0.0)**
SVM_S+SVM_T	71.61 (±2.39)	76.58 (±0.52)	77.50 (±0.77)	78.19 (±0.44)	80.19 (±0.0)
SVM_{SxT}	75.37 (±2.56)	76.10 (±2.62)	76.76 (±0.28)	77.82 (±0.23)	79.75 (±0.0)
$SVM_{S\to T}$	75.78 (±8.43)	75.55 (±1.08)	77.23 (±0.37)	78.11 (±0.49)	79.84 (±0.0)
SVM_{S+T}	75.49 (±1.88)	75.87 (±0.25)	77.23 (±0.47)	77.33 (±0.83)	79.96 (±0.0)
SVM_S	75.52 (±0.44)	76.27 (±0.42)	75.65 (±0.23)	75.65 (±0.4)	75.65 (±0.0)
SVM_T	24.04 (±4.53)	46.45 (±2.29)	60.51 (±1.01)	70.50 (±0.85)	78.04 (±0.0)
P. pacificus					
$SVM_{S,T}$	64.72 (±3.75)	66.39 (±0.66)	**68.44 (±0.67)**	71.00 (±0.38)	74.88 (±0.0)
SVM_S+SVM_T	**64.74 (±3.49)**	**67.30 (±1.38)**	66.58 (±1.82)	**71.82 (±0.97)**	**75.39 (±0.0)**
SVM_{SxT}	57.67 (±26.14)	66.33 (±0.28)	67.29 (±2.24)	71.46 (±0.21)	74.99 (±0.0)
$SVM_{S\to T}$	63.52 (±14.05)	66.07 (±0.07)	67.59 (±1.7)	70.90 (±0.95)	75.11 (±0.0)
SVM_{S+T}	62.99 (±1.48)	65.87 (±0.83)	68.02 (±0.97)	70.85 (±0.37)	74.73 (±0.0)
SVM_S	62.73 (±0.62)	63.77 (±0.04)	63.75 (±0.75)	63.77 (±0.36)	63.83 (±0.0)
SVM_T	20.36 (±3.94)	38.16 (±3.21)	57.28 (±3.46)	67.90 (±1.35)	74.10 (±0.0)
D. melanogaster					
$SVM_{S,T}$	**40.80 (±2.18)**	37.87 (±3.77)	**52.33 (±0.91)**	**58.17 (±1.5)**	63.26 (±0.0)
SVM_S+SVM_T	37.23 (±1.58)	40.36 (±3.32)	48.64 (±0.99)	54.38 (±1.57)	62.26 (±0.0)
SVM_{SxT}	38.71 (±7.67)	**41.23 (±1.4)**	49.58 (±0.91)	56.20 (±1.86)	62.22 (±0.0)
$SVM_{S\to T}$	35.29 (±6.72)	40.15 (±2.47)	48.98 (±2.19)	54.60 (±1.99)	**63.53 (±0.0)**
SVM_{S+T}	36.43 (±1.18)	37.98 (±4.05)	49.46 (±1.38)	56.56 (±2.36)	62.07 (±0.0)
SVM_S	32.95 (±0.38)	33.05 (±0.07)	33.07 (±0.25)	33.07 (±0.01)	33.74 (±0.0)
SVM_T	14.59 (±1.02)	26.69 (±0.58)	38.33 (±2.06)	51.32 (±2.86)	61.26 (±0.0)
A. thaliana					
$SVM_{S,T}$	24.21 (±3.41)	27.30 (±1.46)	**38.49 (±1.59)**	**49.75 (±1.46)**	**56.54 (±0.0)**
SVM_S+SVM_T	21.70 (±2.77)	**28.55 (±1.96)**	35.80 (±1.48)	44.07 (±2.99)	54.06 (±0.0)
SVM_{SxT}	**24.62 (±3.07)**	27.33 (±3.17)	38.20 (±1.32)	47.05 (±2.39)	53.60 (±0.0)
$SVM_{S\to T}$	17.09 (±6.79)	26.41 (±4.81)	36.83 (±1.74)	47.98 (±2.25)	55.99 (±0.0)
SVM_{S+T}	20.06 (±3.23)	24.71 (±3.25)	37.72 (±1.74)	47.31 (±2.55)	53.41 (±0.0)
SVM_S	14.07 (±0.46)	14.85 (±0.1)	14.23 (±0.53)	14.83 (±0.49)	14.33 (±0.0)
SVM_T	10.23 (±1.56)	19.07 (±2.53)	32.56 (±1.91)	45.34 (±2.83)	53.63 (±0.0)

Table F.1.: Area under the precision-recall curve for varying target training set sizes for four different target organisms. Shown are the median values over up to three different training set splits.

Tabular Multi-Source Results

	2500	6500	16000	40000	100000
C. remanei					
M-SVM$_{S,T}$	**69.45 (±0.17)**	71.44 (±1.5)	71.03 (±1.8)	**76.21 (±0.2)**	79.11 (±0.0)
M-SVM$_S$+SVM$_T$	68.51 (±2.95)	**72.69 (±0.86)**	**72.78 (±0.8)**	75.75 (±0.64)	79.06 (±0.0)
M-SVM$_{S\to T}$	63.23 (±5.61)	70.11 (±0.52)	72.09 (±0.19)	75.32 (±0.32)	**79.24 (±0.0)**
SVM$_T$	24.04 (±4.53)	46.45 (±2.29)	60.51 (±1.01)	70.50 (±0.85)	78.04 (±0.0)
P. pacificus					
M-SVM$_{S,T}$	61.38 (±2.05)	64.07 (±1.07)	**67.81 (±2.0)**	**71.05 (±1.26)**	**75.15 (±0.0)**
M-SVM$_S$+SVM$_T$	**62.88 (±0.3)**	**64.77 (±0.52)**	65.52 (±0.81)	70.50 (±0.94)	74.20 (±0.0)
M-SVM$_{S\to T}$	61.46 (±0.49)	62.48 (±0.41)	64.78 (±1.7)	70.14 (±0.56)	74.43 (±0.0)
SVM$_T$	20.36 (±3.94)	38.16 (±3.21)	57.28 (±3.46)	67.90 (±1.35)	74.10 (±0.0)
D. melanogaster					
M-SVM$_{S,T}$	46.32 (±0.39)	47.71 (±1.03)	53.17 (±0.45)	**57.56 (±1.54)**	**62.66 (±0.0)**
M-SVM$_S$+SVM$_T$	**46.61 (±3.27)**	**48.15 (±3.02)**	52.12 (±0.73)	57.01 (±1.64)	62.12 (±0.0)
M-SVM$_{S\to T}$	40.89 (±2.28)	44.61 (±1.51)	**53.29 (±1.47)**	56.35 (±1.57)	61.57 (±0.0)
SVM$_T$	14.59 (±1.02)	26.69 (±0.58)	38.33 (±2.06)	51.32 (±2.86)	61.26 (±0.0)
A. thaliana					
M-SVM$_{S,T}$	**30.90 (±2.12)**	**36.43 (±3.46)**	**43.63 (±1.92)**	**49.49 (±1.92)**	**56.57 (±0.0)**
M-SVM$_S$+SVM$_T$	26.61 (±2.29)	35.58 (±0.31)	39.43 (±1.98)	46.98 (±3.73)	54.11 (±0.0)
M-SVM$_{S\to T}$	27.17 (±1.33)	33.18 (±3.32)	39.32 (±2.07)	47.53 (±2.2)	55.50 (±0.0)
SVM$_T$	10.23 (±1.56)	19.07 (±2.53)	32.56 (±1.91)	45.34 (±2.83)	53.63 (±0.0)

Table F.2.: Area under the precision-recall curve for varying target training set sizes for four different target organisms. Shown are the median values over up to three different training set splits.

Bibliography

[1] A. Alexeyenko, I. Tamas, G. Liu, and E. Sonnhammer. Automatic clustering of orthologs and in-paralogs shared by multiple proteomes. *Bioinformatics*, 22(14):e9–15, 2006.

[2] L. A. Allison. *Fundamental Molecular Biology*. Wiley-Blackwell, 2007.

[3] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, 1990.

[4] Y. Altun and T. Hofmann. Large margin methods for label sequence learning. In *Proceedings of 8th European Conference on Speech Communication and Technology (EuroSpeech)*, 2003.

[5] Y. Altun, M. Johnson, and T. Hofmann. Investigating loss functions and optimization methods for discriminative learning of label sequences. In M. Collins and M. Steedman, editors, *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 145–152, 2003.

[6] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov support vector machines. In *Proceedings of the 20th International Conference on Machine Learning*, pages 3–10, 2003.

[7] S. Anderson. Shotgun DNA sequencing using cloned DNase I-generated fragments. *Nucleic Acids Res*, 9(13):3015–3027, 1981.

[8] O. T. Avery, C. M. Macleod, and M. McCarty. Studies on the chemical nature of the substance inducing transformation of pneumococcal types : Induction of transformation by a desoxyribonucleic acid fraction isolated from pneumococcus type III. *J Exp Med*, 79(2): 137–158, 1944.

[9] G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data (Neural Information Processing)*. The MIT Press, 2007.

[10] Y. Barash, J. A. Calarco, W. Gao, Q. Pan, X. Wang, O. Shai, B. J. Blencowe, and B. J. Frey. Deciphering the splicing code. *Nature*, 465(7294):53–59, 2010.

[11] A. Baten, B. Chang, S. Halgamuge, and J. Li. Splice site identification using probabilistic parameters and SVM classification. *BMC Bioinformatics 7(Suppl. 5):S15*, 2006.

[12] J. Behr. Improving gene finding accuracy with tiling array and RNA-Seq data. *Diplom* thesis, University of Tübingen, Germany, 2008.

[13] A. Ben-Hur, C. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch. Support vector machines and kernels for computational biology. *PLoS Computational Biology*, 4(10):e1000173, 2008.

[14] K. Bennett and O. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[15] B. Benoff, H. Yang, C. L. Lawson, G. Parkinson, J. Liu, E. Blatter, Y. W. Ebright, H. M. Berman, and R. H. Ebright. Structural basis of transcription activation: the CAP-alpha CTD-DNA complex. *Science*, 297(5586):1562–1566, 2002.

[16] A. Bernal, K. Crammer, A. Hatzigeorgiou, and F. Pereira. Global discriminative learning for higher-accuracy computational gene prediction. *PLoS Comput Biol*, 3(3):e54, 2007.

[17] M. Boguski and T. L. C. Tolstoshev. dbEST–database for "expressed sequence tags". *Nature Genetics*, 4(4):332–3, 1993.

[18] M. Borodovsky, A. Lomsadze, N. Ivanov, and R. Mills. Eukaryotic gene prediction using GeneMark.hmm. *Curr Protoc Bioinformatics*, Chapter 4:Unit4.6, 2003.

[19] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proc. COLT*, pages 144–152. ACM Press, 1992.

[20] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[21] V. Brendel and J. Kleffe. Prediction of locally optimal splice sites in plant pre-mRNA with applications to gene identification in *Arabidopsis thaliana* genomic DNA. *Nucleic Acids Res*, 26(20):4748–4757, 1998.

[22] S. Brenner. The genetics of *Caenorhabditis elegans*. *Genetics*, 77(1):71–94, 1974.

[23] M. Brent. Steady progress and recent breakthroughs in the accuracy of automated genome annotation. *Nat Rev Genet*, 9(1):62–73, 2008.

[24] R. Brown, S. Gross, and M. Brent. Begin at the beginning: Predicting genes with 5'UTRs. *Genome Res.*, 15(5):742–747, 2005.

[25] M. Brudno, C. B. Do, G. M. Cooper, M. F. Kim, E. Davydov, E. D. Green, A. Sidow, and S. Batzoglou. LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res*, 13(4):721–731, 2003.

[26] C. Burge. *Identification of genes in human genomic DNA*. PhD thesis, Stanford University, 1997.

[27] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268:78–94, 1997.

[28] Cancer Genome Atlas Research Network. Comprehensive genomic characterization defines human glioblastoma genes and core pathways. *Nature*, 455(7216):1061–1068, 2008.

[29] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[30] T.-M. Chen, C.-C. Lu, and W.-H. Li. Prediction of splice sites with dependency graphs and their expanded bayesian networks. *Bioinformatics*, 21(4):471–482, 2005.

[31] Y. Cheng, R. M. Miura, and B. Tian. Prediction of mRNA polyadenylation sites by support vector machine. *Bioinformatics*, 22(19):2320–2325, 2006.

[32] R. M. Clark, G. Schweikert, C. Toomajian, S. Ossowski, G. Zeller, P. Shinn, N. Warthmann, T. T. Hu, G. Fu, D. A. Hinds, H. Chen, K. A. Frazer, D. H. Huson, B. Schölkopf, M. Nordborg, G. Rätsch, J. R. Ecker, and D. Weigel. Common sequence polymorphisms shaping genetic diversity in *Arabidopsis thaliana*. *Science*, 317(5836):338–342, 2007.

[33] L. Cochella and R. Green. Wobble during decoding: more than third-position promiscuity. *Nat Struct Mol Biol*, 11(12):1160–1162, 2004.

[34] A. Coghlan, T. J. Fiedler, S. J. McKay, P. Flicek, T. W. Harris, D. Blasiar, The nGASP Consortium, and L. D. Stein. nGASP–the nematode genome annotation assessment project. *BMC Bioinformatics*, 9:549, 2008.

[35] M. Collins. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 1–8, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[36] M. Collins. Parameter estimation for statistical parsing models: theory and practice of distribution-free methods. In *New developments in parsing technology*, pages 19–55, Norwell, MA, USA, 2004. Kluwer Academic Publishers.

[37] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[38] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, 2002.

[39] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991, 2003.

[40] F. H. Crick. On protein synthesis. *Symp Soc Exp Biol*, 12:138–163, 1958.

[41] F. H. Crick. Codon–anticodon pairing: the wobble hypothesis. *J Mol Biol*, 19(2):548–555, 1966.

[42] F. H. Crick. How to live with a golden helix. *sciences-New York*, 19, 1979.

[43] R. Dahm. Discovering DNA: Friedrich Miescher and the early years of nucleic acid research. *Hum Genet*, 122(6):565–581, 2008.

[44] H. Daume III. Frustratingly easy domain adaptation. In *Conference of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic, 2007.

[45] K. Davies. *Cracking the Genome: Inside the Race to Unlock Human DNA*. The Free Press, 2001.

[46] J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *ICML*, pages 233–240, 2006.

[47] D. DeCaprio, J. Vinson, M. Pearson, P. Montgomery, M. Doherty, and J. Galagan. Conrad: Gene prediction using conditional random fields. *Genome Res*, 17:1389–1398, 2007.

[48] S. Degroeve, Y. Saeys, B. D. Baets, P. Rouzé, and Y. V. de Peer. Splicemachine: predicting splice sites from high-dimensional local context representations. *Bioinformatics*, 21(8):1332–8, 2005.

[49] C. Dieterich, S. Clifton, L. Schuster, A. Chinwalla, K. Delehaunty, I. Dinkelacker, L. Fulton, R. Fulton, J. Godfrey, P. Minx, M. Mitreva, W. Roeseler, H. Tian, H. Witte, S. Yang, R. Wilson, and R. Sommer. The *Pristionchus pacificus* genome provides a unique perspective on nematode lifestyle and parasitism. *Nat Genet*, 40(10):1193–1198, 2008.

[50] Z. Dominski and R. Kole. Selection of splice sites in pre-mRNAs with short internal exons. *Mol Cell Biol*, 11(12):6075–6083, 1991.

[51] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[52] ENCODE Project Consortium. The ENCODE (ENCyclopedia Of DNA Elements) Project. *Science*, 306(5696):636–640, 2004.

[53] Encyclopdia Britannica. DNA: molecule. World Wide Web electronic publication, 2007. URL `http://www.britannica.com/EBchecked/topic-art/528804/40216/DNA-molecule`.

[54] T. Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.

[55] P. Flicek, B. L. Aken, B. Ballester, K. Beal, E. Bragin, S. Brent, Y. Chen, P. Clapham, G. Coates, S. Fairley, S. Fitzgerald, J. Fernandez-Banet, L. Gordon, S. Graf, S. Haider, M. Hammond, K. Howe, A. Jenkinson, N. Johnson, A. Kahari, D. Keefe, S. Keenan, R. Kinsella, F. Kokocinski, G. Koscielny, E. Kulesha, D. Lawson, I. Longden, T. Massingham, W. McLaren, K. Megy, B. Overduin, B. Pritchard, D. Rios, M. Ruffier, M. Schuster, G. Slater, D. Smedley, G. Spudich, Y. A. Tang, S. Trevanion, A. Vilella, J. Vogel, S. White, S. P. Wilder, A. Zadissa, E. Birney, F. Cunningham, I. Dunham, R. Durbin, X. M. Fernandez-Suarez, J. Herrero, T. J. P. Hubbard, A. Parker, G. Proctor, J. Smith, and S. M. J. Searle. Ensembl's 10th year. *Nucleic Acids Res*, 38(Database issue):D557–62, 2010.

[56] S. Foissac and T. Schiex. Integrating alternative splicing detection into gene prediction. *BMC Bioinformatics*, 6:25, 2005.

[57] R. E. Franklin and R. G. Gosling. Molecular configuration in sodium thymonucleate. *Nature*, 171(4356):740–741, 1953.

[58] K. Fukumizu, A. Gretton, X. Sun, and B. Schölkopf. Kernel measures of conditional dependence. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 489–496. MIT Press, Cambridge, MA, 2008.

[59] T. Gao, Z. Yang, Y. Wang, and L. Jing. Identifying translation initiation sites in prokaryotes using support vector machine. *J Theor Biol*, 262(4):644–649, 2010.

[60] B. Gassend, C. W. ODonnell, W. Thies, A. Lee, M. van Dijk, and S. Devadas. Predicting secondary structure of all-helical proteins using hidden Markov support vector machines. In *Pattern Recognition in Bioinformatics*, number 4146 in LNCS, pages 93–104. Springer Berlin/Heidelberg, 2006.

[61] D. S. Gerhard, L. Wagner, E. A. Feingold, C. M. Shenmen, L. H. Grouse, G. Schuler, S. L. Klein, S. Old, R. Rasooly, P. Good, M. Guyer, A. M. Peck, J. G. Derge, D. Lipman, F. S. Collins, W. Jang, S. Sherry, M. Feolo, L. Misquitta, E. Lee, K. Rotmistrovsky, S. F. Greenhut, C. F. Schaefer, K. Buetow, T. I. Bonner, D. Haussler, J. Kent, M. Kiekhaus, T. Furey, M. Brent, C. Prange, K. Schreiber, N. Shapiro, N. K. Bhat, R. F. Hopkins, F. Hsie, T. Driscoll, M. B. Soares, T. L. Casavant, T. E. Scheetz, M. J. Brown-stein, T. B. Usdin, S. Toshiyuki, P. Carninci, Y. Piao, D. B. Dudekula, M. S. H. Ko, K. Kawakami, Y. Suzuki, S. Sugano, C. E. Gruber, M. R. Smith, B. Simmons, T. Moore, R. Waterman, S. L. Johnson, Y. Ruan, C. L. Wei, S. Mathavan, P. H. Gunaratne, J. Wu, A. M. Garcia, S. W. Hulyk, E. Fuh, Y. Yuan, A. Sneed, C. Kowis, A. Hodgson, D. M. Muzny, J. McPherson, R. A. Gibbs, J. Fahey, E. Helton, M. Ketteman, A. Madan, S. Rodrigues, A. Sanchez, M. Whiting, A. Madari, A. C. Young, K. D. Wetherby, S. J. Granite, P. N. Kwong, C. P. Brinkley, R. L. Pearson, G. G. Bouffard, R. W. Blakesly, E. D. Green, M. C. Dickson, A. C. Rodriguez, J. Grimwood, J. Schmutz, R. M. Myers, Y. S. N. Butterfield, M. Griffith, O. L. Griffith, M. I. Krzywinski, N. Liao, R. Morin, D. Palmquist, A. S. Petrescu, U. Skalska, D. E. Smailus, J. M. Stott, A. Schnerch, J. E. Schein, S. J. M. Jones, R. A. Holt, A. Baross, M. A. Marra, S. Clifton, K. A. Makowski, S. Bosak, and J. Malek. The status, quality, and expansion of the NIH full-length cDNA project: the Mammalian Gene Collection (MGC). *Genome Res*, 14(10B):2121–2127, 2004.

[62] U. Gerischer, editor. *Acinetobacter Molecular Biology*. Caister Academic Press, 2008.

[63] B. Giardine, C. Riemer, R. Hardison, R. Burhans, L. Elnitski, P. Shah, Y. Zhang, D. Blankenberg, I. Albert, J. Taylor, W. Miller, W. Kent, and A. Nekrutenko. Galaxy: a platform for interactive large-scale genome analysis. *Genome Res*, 15(10):1451–1455, 2005.

[64] R. Giegerich, C. Meyer, and P. Steffen. A discipline of dynamic programming over sequence data. *Sci. Comput. Program.*, 51(3):215–263, 2004.

[65] J. Graber, J. Salisbury, L. Hutchins, and T. Blumenthal. *C. elegans* sequences that control *trans*-splicing and operon pre-mRNA processing. *RNA*, 13, 2007.

[66] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems 19*, Cambridge, MA, 2007. MIT Press.

[67] A. Gretton, A. Smola, J. Huang, M. Schmittfull, and B. Schölkopf. Covariate shift by kernel mean matching. In J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, editors, *Dataset Shift in Machine Learning*. MIT Press, Cambridge, MA, 2008.

[68] F. Griffith. The significance of pneumococcal types. *J Hyg (Lond)*, 27(2):113–159, 1928.

[69] S. Gross and M. Brent. Using multiple alignments to improve gene prediction. *J Comput Biol*, 13(2):379–393, 2006.

[70] S. Gross, C. Do, M. Sirota, and S. Batzoglou. Contrast: A discriminative, phylogeny-free approach to multiple informant de novo gene prediction. *Genome Biol*, 8(12):r269, 2007.

[71] R. Guigó, J. Flicek, P. Abril, A. Reymond, J. Lagarde, F. Denoeud, S. Antonarakis, M. Ashburner, V. Bajic, E. Birney, R. Castelo, E. Eyras, C. Ucla, T. Gingeras, J. Harrow, T. Hubbard, S. Lewis, and M. Reese. EGASP: The human ENCODE genome annotation assessment project. *Genome Biology*, 7(S2), 2006.

[72] S. Hahn. Structure and mechanism of the RNA polymerase II transcription machinery. *Nat Struct Mol Biol*, 11(5):394–403, 2004.

[73] T. W. Harris, I. Antoshechkin, T. Bieri, D. Blasiar, J. Chan, W. J. Chen, N. De La Cruz, P. Davis, M. Duesbury, R. Fang, J. Fernandes, M. Han, R. Kishore, R. Lee, H.-M. Muller, C. Nakamura, P. Ozersky, A. Petcherski, A. Rangarajan, A. Rogers, G. Schindelman, E. M. Schwarz, M. A. Tuli, K. Van Auken, D. Wang, X. Wang, G. Williams, K. Yook, R. Durbin, L. D. Stein, J. Spieth, and P. W. Sternberg. Wormbase: a comprehensive resource for nematode research. *Nucleic Acids Res*, 38(Database issue):D463–7, 2010.

[74] P. M. Harrison, N. Echols, and M. B. Gerstein. Digging for dead genes: an analysis of the characteristics of the pseudogene population in the *Caenorhabditis elegans* genome. *Nucleic Acids Res*, 29(3):818–830, 2001.

[75] R. Hettich and K. Kortanek. Semi-infinite programming: Theory, methods and applications. *SIAM Review*, 3:380–429, 1993.

[76] L. W. Hillier, V. Reinke, P. Green, M. Hirst, M. A. Marra, and R. H. Waterston. Massively parallel sequencing of the polyadenylated transcriptome of *C. elegans*. *Genome Res*, 19(4): 657–666, 2009.

[77] T. J. P. Hubbard, B. L. Aken, S. Ayling, B. Ballester, K. Beal, E. Bragin, S. Brent, Y. Chen, P. Clapham, L. Clarke, G. Coates, S. Fairley, S. Fitzgerald, J. Fernandez-Banet, L. Gordon, S. Graf, S. Haider, M. Hammond, R. Holland, K. Howe, A. Jenkinson, N. Johnson, A. Kahari, D. Keefe, S. Keenan, R. Kinsella, F. Kokocinski, E. Kulesha, D. Lawson, I. Longden, K. Megy, P. Meidl, B. Overduin, A. Parker, B. Pritchard, D. Rios, M. Schuster, G. Slater, D. Smedley, W. Spooner, G. Spudich, S. Trevanion, A. Vilella, J. Vogel, S. White, S. Wilder, A. Zadissa, E. Birney, F. Cunningham, V. Curwen, R. Durbin, X. M. Fernandez-Suarez, J. Herrero, A. Kasprzyk, G. Proctor, J. Smith, S. Searle, and P. Flicek. Ensembl 2009. *Nucleic Acids Res*, 37(Database issue):D690–7, 2009.

[78] S. Hunter, R. Apweiler, T. Attwood, A. Bairoch, A. Bateman, D. Binns, P. Bork, U. Das, L. Daugherty, L. Duquenne, R. Finn, J. Gough, D. Haft, N. Hulo, D. Kahn, E. Kelly, A. Laugraud, I. Letunic, D. Lonsdale, R. Lopez, M. Madera, J. Maslen, C. McAnulla, J. McDowall, J. Mistry, A. Mitchell, N. Mulder, D. Natale, C. Orengo, A. Quinn, J. Selengut, C. Sigrist, M. Thimma, P. Thomas, F. Valentin, D. Wilson, C. Wu, and C. Yeats. InterPro: the integrative protein signature database. *Nucleic Acids Res*, 2008.

[79] International HapMap Consortium. A haplotype map of the human genome. *Nature*, 437 (7063):1299–1320, 2005.

[80] J. T. Kadonaga. Regulation of RNA polymerase II transcription by sequence-specific DNA binding factors. *Cell*, 116(2):247–257, 2004.

[81] S. Keerthi and S. Sundararajan. CRF versus SVM-struct for sequence labeling. Technical report, Yahoo Research, 2007.

[82] W. J. Kent. BLAT—the BLAST-like alignment tool. *Genome Res*, 12(4):656–664, 2002.

[83] W. J. Kent and A. M. Zahler. Conservation, regulation, synteny, and introns in a large-scale *C. briggsae — C. elegans* genomic alignment. *Genome Res*, 10(8):1115–1125, 2000.

[84] G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Appl.*, 33:82–95, 1971.

[85] K. Kiontke and W. Sudhaus. Ecology of *Caenorhabditis* species. *WormBook*, pages 1–14, 2006.

[86] I. Korf. Gene finding in novel genomes. *BMC Bioinformatics*, 5:59, 2004.

[87] I. Korf, P. Flicek, D. Duan, and M. R. Brent. Integrating genomic homology into gene structure prediction. *Bioinformatics*, 17 Suppl 1:S140–8, 2001.

[88] A. Krogh. Using database matches with HMMGene for automated gene detection in *Drosophila*. *Genome Res*, 10(4):523–528, 2000.

[89] D. Kulp, D. Haussler, M. Reese, and F. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. *ISMB 1996*, pages 134–141, 1996.

[90] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.

[91] C. H. Lampert. Kernel methods in computer vision. *Found. Trends. Comput. Graph. Vis.*, 4(3):193–285, 2009.

[92] C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *J. Mach. Learn. Res.*, 5:1435–1455, 2004.

[93] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. In R. Altman, A. Dunker, L. Hunter, K. Lauerdale, and T. Klein, editors, *PSB*, pages 564–575. River Edge, NJ, World Scientific, 2002.

[94] B. Lewin. *Genes IX*. Johns and Bartlett Publishers, 2008.

[95] H. Li and T. Jiang. A class of edit kernels for SVMs to predict translation initiation sites in eukaryotic mRNAs. *J Comput Biol*, 12(6):702–718, 2005.

[96] Q. Liu, A. Mackey, D. Roos, and F. Pereira. Evigan: A hidden variable model for integrating gene evidence for eukaryotic gene prediction. *Bioinformatics*, 24(5):597–605, 2008.

[97] A. V. Lukashin and M. Borodovsky. GeneMark.hmm: new solutions for gene finding. *Nucleic Acids Res*, 26(4):1107–1115, 1998.

[98] W. Majoros. *Methods for Computational Gene Prediction*. Cambridge University Press., 2007.

[99] W. Majoros, M. Pertea, C. Antonescu, and S. L. Salzberg. GlimmerM, Exonomy and Unveil: three ab initio eukaryotic genefinders. *Nucleic Acids Res*, 31(13):3601–3604, 2003.

[100] C. P. Manning, P. Raghavan, and H. Schtze. *Evaluation in information retrieval*. Cambridge University Press, 2008.

[101] E. H. Margulies and E. Birney. Approaches to comparative sequence analysis: towards a functional view of vertebrate genomes. *Nat Rev Genet*, 9(4):303–313, 2008.

[102] J. Medenbach. Alternative splicing. World Wide Web electronic publication, 2005. URL `http://de.wikipedia.org/wiki/Alternatives_Spleißen`.

[103] C. E. Metz. Basic principles of ROC analysis. *Seminars in Nuclear Medicine*, VIII(4), 1978.

[104] F. Miescher. Ueber die chemische Zusammensetzung der Eiterzellen. *Medicinisch-chemische Untersuchungen*, 4:441–460, 1871.

[105] R. C. Moore, W.-t. Yih, and A. Bode. Improved discriminative bilingual word alignment. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 513–520, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[106] A. Mortazavi, B. A. Williams, K. McCue, L. Schaeffer, and B. Wold. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nat Methods*, 5(7):621–628, 2008.

[107] A. Mounsey, P. Bauer, and I. A. Hope. Evidence suggesting that a fifth of annotated *Caenorhabditis elegans* genes may be pseudogenes. *Genome Res*, 12(5):770–775, 2002.

[108] P. Niermann. Machine-learning based gene finding for human sequences. *Diplom* thesis, University of Tübingen, Germany, 2009.

[109] D.-K. Niu. Exon definition as a potential negative force against intron losses in evolution. *Biol Direct*, 3:46, 2008.

[110] F. Odronitz and M. Kollmar. Comparative genomic analysis of the arthropod muscle myosin heavy chain genes allows ancestral gene reconstruction and reveals a new type of 'partially' processed pseudogene. *BMC Mol Biol*, 9:21, 2008.

[111] G. Parra, P. Agarwal, J. F. Abril, T. Wiehe, J. W. Fickett, and R. Guigo. Comparative gene prediction in human and mouse. *Genome Res*, 13(1):108–117, 2003.

[112] M. Pertea and S. L. Salzberg. Between a chicken and a grape: estimating the number of human genes. *Genome Biol*, 11(5):206, 2010.

[113] M. Pertea, X. Lin, and S. L. Salzberg. GeneSplicer: a new computational method for splice site prediction. *Nucleic Acids Res*, 29(5):1185–1190, 2001.

[114] A. Pires-DaSilva and R. Sommer. Conservation of the global sex determination gene tra-1 in distantly related nematodes. *GENES & DEVELOPMENT*, 18(10):1198–1208, 2004.

[115] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[116] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Readings in speech recognition*, pages 267–296, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[117] G. Rätsch. *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Germany, Am neuen Palais, 14469 Potsdam, Germany, 2001.

[118] G. Rätsch and S. Sonnenburg. Accurate splice site detection for *Caenorhabditis elegans*. In K. T. B. Schölkopf and J.-P. Vert, editors, *Kernel Methods in Computational Biology*. MIT Press, 2004.

[119] G. Rätsch and S. Sonnenburg. Large-scale hidden semi-Markov SVMs. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS'06)*, volume 19, pages 1161–1168, Cambridge, MA, 2007. MIT Press.

[120] G. Rätsch, S. Sonnenburg, and B. Schölkopf. RASE: Recognition of alternatively spliced exons in *C. elegans*. *Bioinformatics*, 21(Suppl. 1):i369–i377, 2005.

[121] G. Rätsch, S. Sonnenburg, and C. Schäfer. Learning interpretable SVMs for biological sequence classification. *BMC Bioinformatics*, 7(Suppl 1):S9, 2006.

[122] G. Rätsch, S. Sonnenburg, J. Srinivasan, H. Witte, K.-R. Müller, R.-J. Sommer, and B. Schölkopf. Improving the *Caenorhabditis elegans* genome annotation using machine learning. *PLoS Comput Biol*, 3(2):e20, 2007.

[123] B. Rhead, D. Karolchik, R. M. Kuhn, A. S. Hinrichs, A. S. Zweig, P. A. Fujita, M. Diekhans, K. E. Smith, K. R. Rosenbloom, B. J. Raney, A. Pohl, M. Pheasant, L. R. Meyer, K. Learned, F. Hsu, J. Hillman-Jackson, R. A. Harte, B. Giardine, T. R. Dreszer, H. Clawson, G. P. Barber, D. Haussler, and W. J. Kent. The ucsc genome browser database: update 2010. *Nucleic Acids Res*, 38(Database issue):D613–9, 2010.

[124] B. L. Robberson, G. J. Cote, and S. M. Berget. Exon definition may facilitate splice site selection in RNAs with multiple exons. *Mol Cell Biol*, 10(1):84–94, 1990.

[125] S. Rozen and H. Skaletsky. Primer3 on the WWW for general users and for biologist programmers. In S. Krawetz and S. Misener, editors, *Bioinformatics Methods and Protocols: Methods in Molecular Biology*, pages 365–386. Humana Press, Totowa, NJ, 2000.

[126] T. Sakurai, M. Satou, K. Akiyama, K. Iida, M. Seki, T. Kuromori, T. Ito, A. Konagaya, T. Toyoda, and K. Shinozaki. RARGE: a large-scale database of RIKEN *Arabidopsis* resources ranging from transcriptome to phenome. *Nucleic Acids Res*, 33(Database issue): D647–50, 2005.

[127] A. Salamov and V. Solovyev. Ab initio gene finding in *Drosophila* genomic DNA. *Genome Res.*, 10(4):516–522, 2000.

[128] S. Salzberg, A. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, 26(2):544–548, 1998.

[129] S. L. Salzberg, M. Pertea, A. L. Delcher, M. J. Gardner, and H. Tettelin. Interpolated Markov models for eukaryotic gene finding. *Genomics*, 59(1):24–31, 1999.

[130] F. Sanger and A. R. Coulson. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *J Mol Biol*, 94(3):441–448, 1975.

[131] F. Sanger, G. M. Air, B. G. Barrell, N. L. Brown, A. R. Coulson, C. A. Fiddes, C. A. Hutchison, P. M. Slocombe, and M. Smith. Nucleotide sequence of bacteriophage phi x174 dna. *Nature*, 265(5596):687–695, 1977.

[132] F. Sanger, S. Nicklen, and A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci U S A*, 74(12):5463–5467, 1977.

[133] S. Sarawagi and W. Cohen. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems 17 (NIPS'04)*, 2004.

[134] S. Sarawagi and R. Gupta. Accurate max-margin training for structured output spaces. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 888–895, New York, NY, USA, 2008. ACM.

[135] J. Schmitz, D. Prufer, W. Rohde, and E. Tacke. Non-canonical translation mechanisms in plants: efficient in vitro and in planta initiation at auu codons of the tobacco mosaic virus enhancer sequence. *Nucleic Acids Res*, 24(2):257–263, 1996.

[136] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[137] U. Schulze, B. Hepp, C. S. Ong, and G. Rätsch. PALMA: mRNA to genome alignments using large margin algorithms. *Bioinformatics*, 23(15):1892–1900, 2007. Project URL: http://www.fml.mpg.de/raetsch/projects/palma.

[138] S. H. Schwartz, J. Silva, D. Burstein, T. Pupko, E. Eyras, and G. Ast. Large-scale comparative analysis of splicing signals and their corresponding splicing factors in eukaryotes. *Genome Res*, 18(1):88–103, 2008.

[139] G. Schweikert, J. Behr, A. Zien, G. Zeller, C. S. Ong, S. Sonnenburg, and G. Rätsch. mGene.web: a web service for accurate computational gene finding. *Nucleic Acids Res*, 37(Web Server issue):W312–6, 2009.

[140] G. Schweikert, C. Widmer, B. Schölkopf, and G. Rätsch. An empirical analysis of domain adaptation algorithms for genomic sequence analysis. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1433–1440, 2009.

[141] G. Schweikert, A. Zien, G. Zeller, J. Behr, C. Dieterich, C. S. Ong, P. Philips, F. De Bona, L. Hartmann, A. Bohlen, N. Krüger, S. Sonnenburg, and G. Rätsch. mGene: accurate SVM-based gene finding with an application to nematode genomes. *Genome Res*, 19(11): 2133–2143, 2009.

[142] F. Sha. *Large margin training of acoustic models for speech recognition*. PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 2007. Adviser-Saul, Lawrence K.

[143] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[144] A. Smit, R. Hubley, and P. Green. RepeatMasker Open-3.0. World Wide Web electronic publication, 1996-2004. URL `http://www.repeatmasker.org`.

[145] S. Sonnenburg. *Machine Learning for Genomic Sequence Analysis*. PhD thesis, Fraunhofer Institute FIRST, 2008. supervised by K.-R. Müller and G. Rätsch.

[146] S. Sonnenburg, A. Zien, and G. Rätsch. ARTS: accurate recognition of transcription starts in human. *Bioinformatics*, 22(14):e472–80, 2006.

[147] S. Sonnenburg, G. Rätsch, and K. Rieck. Large-scale learning with string kernels. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large-Scale Kernel Machines*, chapter 4, pages 73–104. MIT Press, Cambridge, MA, 2007.

[148] S. Sonnenburg, G. Schweikert, P. Philips, J. Behr, and G. Rätsch. Accurate splice site prediction using support vector machines. *BMC Bioinformatics*, 8 Suppl 10:S7, 2007.

[149] S. Sonnenburg, G. Rätsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. de Bona, A. Binder, C. Gehl, and V. Franc. The SHOGUN machine learning toolbox. *Journal of Machine Learning Research*, 11:1799–1802, 2010.

[150] R. Staden. A strategy of DNA sequencing employing computer programs. *Nucleic Acids Res*, 6(7):2601–2610, 1979.

[151] R. Staden. Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Res*, 12(1 Pt 2):505–519, 1984.

[152] M. Stanke and S. Waack. Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics*, 19 Suppl 2:ii215–25, 2003.

[153] M. Stanke, O. Schoffmann, B. Morgenstern, and S. Waack. Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. *BMC Bioinformatics*, 7(1):62, 2006.

[154] M. Stapleton, J. Carlson, P. Brokstein, C. Yu, M. Champe, R. George, H. Guarin, B. Kronmiller, J. Pacleb, S. Park, K. Wan, G. M. Rubin, and S. E. Celniker. A *Drosophila* full-length cDNA resource. *Genome Biol*, 3(12):research 0080, 2002.

[155] M. Stapleton, G. Liao, P. Brokstein, L. Hong, P. Carninci, T. Shiraki, Y. Hayashizaki, M. Champe, J. Pacleb, K. Wan, C. Yu, J. Carlson, R. George, S. Celniker, and G. M. Rubin. The Drosophila gene collection: identification of putative full-length cDNAs for 70% of D. melanogaster genes. *Genome Res*, 12(8):1294–1300, 2002.

[156] A. Stark, M. F. Lin, P. Kheradpour, J. S. Pedersen, L. Parts, J. W. Carlson, M. A. Crosby, M. D. Rasmussen, S. Roy, A. N. Deoras, J. G. Ruby, J. Brennecke, E. Hodges, A. S. Hinrichs, A. Caspi, B. Paten, S.-W. Park, M. V. Han, M. L. Maeder, B. J. Polansky, B. E. Robson, S. Aerts, J. van Helden, B. Hassan, D. G. Gilbert, D. A. Eastman, M. Rice, M. Weir, M. W. Hahn, Y. Park, C. N. Dewey, L. Pachter, W. J. Kent, D. Haussler, E. C. Lai, D. P. Bartel, G. J. Hannon, T. C. Kaufman, M. B. Eisen, A. G. Clark, D. Smith, S. E. Celniker, W. M. Gelbart, and M. Kellis. Discovery of functional elements in 12 *Drosophila* genomes using evolutionary signatures. *Nature*, 450(7167):219–232, 2007.

[157] L. Stein, Z. Bao, D. Blasiar, T. Blumenthal, M. Brent, N. Chen, A. Chinwalla, L. Clarke, C. Clee, A. Coghlan, A. Coulson, P. D'Eustachio, D. Fitch, L. Fulton, R. Fulton, S. Griffiths-Jones, T. Harris, L. Hillier, R. Kamath, P. Kuwabara, E. Mardis, M. Marra, T. Miner, P. Minx, J. Mullikin, R. Plumb, J. Rogers, J. Schein, M. Sohrmann, J. Spieth, J. Stajich, C. Wei, D. Willey, R. Wilson, R. Durbin, and R. Waterston. The genome sequence of *Caenorhabditis briggsae*: a platform for comparative genomics. *PLoS Biol*, 1(2):E45, 2003.

[158] L. Stein, D. Blasiar, A. Coghlan, T. Fiedler, S. McKay, and P. Flicek. nGASP gene prediction challenge, 2007. `http://www.wormbase.org/wiki/index.php/Gene_Prediction`.

[159] P. Sternberg, R. Waterston, J. Speith, S. Eddy, and R. Wilson. Genome sequence of additional *Caenorhabditis* species: enhancing the utility of *C. elegans* as a model organism. *National Human Genome Research Institute White Paper*, 2003.

[160] M. Sugiyama, M. Krauledat, and K.-R. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:1027–1061, 2007.

[161] S. Sugiyama and K.-R. Müller. Input-dependent estimation of generalization error under covariate shift. *Statistics and Decisions*, 23(4):249–279, 2005.

[162] M. Sultan, M. H. Schulz, H. Richard, A. Magen, A. Klingenhoff, M. Scherf, M. Seifert, T. Borodina, A. Soldatov, D. Parkhomchuk, D. Schmidt, S. O'Keeffe, S. Haas, M. Vingron, H. Lehrach, and M.-L. Yaspo. A global view of gene activity and alternative splicing by deep sequencing of the human transcriptome. *Science*, 321(5891):956–960, 2008.

[163] O. Svensson, L. Arvestad, and J. Lagergren. Genome-wide survey for biologically functional pseudogenes. *PLoS Comput Biol*, 2(5):e46, 2006.

[164] B. Taskar. *Learning structured prediction models: a large margin approach.* PhD thesis, Stanford University, Stanford, CA, USA, 2005.

[165] B. Taskar, D. Klein, M. Collins, D. Koller, and C. D. Manning. Max-margin parsing. In *EMNLP*, pages 1–8, 2004.

[166] The *C. elegans* Sequencing Consortium. Genome sequence of the Nematode *Caenorhabditis elegans*: A platform for investigating biology. *Science*, 282:2012–2018, 1998.

[167] The White House. Press release, june 26, 2000.

[168] J. Thomas. Adaptive evolution in two large families of ubiquitin-ligase adapters in nematodes and plants. *Genome Res*, 16(8):1017–1030, 2006.

[169] J. Thomas and H. Robertson. The Caenorhabditis chemoreceptor gene families. *BMC Biol*, 6:42, 2008.

[170] M. C. Thomas and C.-M. Chiang. The general transcription machinery and general cofactors. *Crit Rev Biochem Mol Biol*, 41(3):105–178, 2006.

[171] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the International Conference on Machine Learning*, 2004.

[172] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005.

[173] A. Ureta-Vidal, L. Ettwiller, and E. Birney. Comparative genomics: Genome-wide analysis in metazoan eukaryotes. *Nature Reviews Genetics*, 4(4):251–262, 2003.

[174] H. van Bakel, C. Nislow, B. J. Blencowe, and T. R. Hughes. Most "dark matter" transcripts are associated with known genes. *PLoS Biol*, 8(5):e1000371, 2010.

[175] V. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian].* Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).

[176] V. Vapnik. *The Nature of Statistical Learning Theory*. New York, Springer Verlag, 1995.

[177] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24, 1963.

[178] H. Wakaguri, R. Yamashita, Y. Suzuki, S. Sugano, and K. Nakai. DBTSS: database of transcription start sites, progress report 2008. *Nucleic Acids Res*, 36(Database issue):D97–101, 2008.

[179] E. T. Wang, R. Sandberg, S. Luo, I. Khrebtukova, L. Zhang, C. Mayr, S. F. Kingsmore, G. P. Schroth, and C. B. Burge. Alternative isoform regulation in human tissue transcriptomes. *Nature*, 456(7221):470–476, 2008.

[180] Z. Wang, M. Gerstein, and M. Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nat Rev Genet*, 10(1):57–63, 2009.

[181] J. D. Watson and F. H. Crick. Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953.

[182] J. Weston and C. Watkins. Multi-class support vector machines. Technical report, Department of Computer Science, Royal Holloway, University London, 1998.

[183] D. L. Wheeler, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, D. M. Church, M. DiCuccio, R. Edgar, S. Federhen, W. Helmberg, D. L. Kenton, O. Khovayko, D. J. Lipman, T. L. Madden, D. R. Maglott, J. Ostell, J. U. Pontius, K. D. Pruitt, G. D. Schuler, L. M. Schriml, E. Sequeira, S. T. Sherry, K. Sirotkin, G. Starchenko, T. O. Suzek, R. Tatusov, T. A. Tatusova, L. Wagner, and E. Yaschenko. Database resources of the national center for biotechnology information. *Nucleic Acids Res*, 33(Database issue):D39–45, 2005.

[184] C. Widmer. Domain adaptation in sequence analysis. *Diplom* thesis, University of Tübingen, Germany, 2008.

[185] C. Widmer. Empirical analysis of domain adaptation algorithms. Talk at the TU Berlin, Seminar on Methods for Domain Adaptation, 2008.

[186] R. Wilson. pers. comm. World Wide Web electronic publication, 2009. URL `www.genome.wustl.edu`.

[187] WormBase consortium. nGASP. World Wide Web electronic publication, 2006. URL `http://www.wormbase.org/wiki/index.php/NGASP`.

[188] C. Wu, R. Apweiler, A. Bairoch, D. Natale, W. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. Martin, R. Mazumder, C. O'Donovan, N. Redaschi, and B. Suzek. The Universal Protein Resource (UniProt): an expanding universe of protein information. *Nucleic Acids Res*, 34(Database issue):D187–91, 2006.

[189] Y. Xue, A. Daly, B. Yngvadottir, M. Liu, G. Coop, Y. Kim, P. Sabeti, Y. Chen, J. Stalker, E. Huckle, J. Burton, S. Leonard, J. Rogers, and C. Tyler-Smith. Spread of an inactive form of caspase-12 in humans is due to recent positive selection. *Am J Hum Genet*, 78(4):659–670, 2006.

[190] R. F. Yeh, L. P. Lim, and C. B. Burge. Computational inference of homologous gene structures in the human genome. *Genome Res*, 11(5):803–816, 2001.

[191] E. M. Zdobnov and R. Apweiler. InterProScan–an integration platform for the signature-recognition methods in InterPro. *Bioinformatics*, 17(9):847–848, 2001.

[192] J. Zhang, X. Gao, J. Xu, and M. Li. Rapid and accurate protein side chain prediction with local backbone information. In *RECOMB'08: Proceedings of the 12th annual international conference on Research in computational molecular biology*, pages 285–299, Berlin, Heidelberg, 2008. Springer-Verlag.

[193] M. Q. Zhang and T. G. Marr. A weight array method for splicing signal analysis. *Comput Appl Biosci*, 9(5):499–509, 1993.

[194] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering support vector machine kernels that recognize translation initiation sites. *BioInformatics*, 16(9):799–807, 2000.

Publications

Publications related to this thesis are marked with •, others with ◦.

Journal Articles

• **G. Schweikert**, A. Zien, G. Zeller, J. Behr, C. Dieterich, C. S. Ong, P. Philips, F. De Bona, L. Hartmann, A. Bohlen, N. Krüger, S. Sonnenburg, and G. Rätsch. mGene: accurate SVM-based gene finding with an application to nematode genomes. *Genome Research*, 19(11):2133-43, 2009

• **G. Schweikert***, J. Behr*, A. Zien, G. Zeller, C. S. Ong, S. Sonnenburg, and G. Rätsch. mGene.web: a web service for accurate computational gene finding. *Nucleic Acids Research*, 1;37(Web Server Issue):W312-6, 2009.

• A. Coghlan, T. J. Fiedler, S. J. McKay, P. Flicek, T. W. Harris, D. Blasiar, **The nGASP Consortium**, and L.D. Stein. nGASP: the nematode genome annotation assessment project. *BMC Bioinformatics*, 9:549, 2008

• S. Sonnenburg*, **G. Schweikert***, P. Philips*, J. Behr, and G. Rätsch. Accurate splice site prediction using support vector machines. *BMC Bioinformatics*. 8 Suppl 10:S7, 2007.

◦ R. M. Clark, **G. Schweikert***, C. Toomajian*, S. Ossowski*, G. Zeller*, P. Shinn, N. Warthmann, T. T. Hu, G. Fu, D. A. Hinds, H. Chen, K. A. Frazer, D. H. Huson, B. Schölkopf, M. Nordborg, G. Rätsch, J. R. Ecker, and D. Weigel. Common sequence polymorphisms shaping genetic diversity in *Arabidopsis thaliana*. *Science*, 317(5836)338–342, 2007.

◦ V. Lucic, T. Yang, **G. Schweikert**, F. Förster, W. Baumeister. Morphological characterization of molecular complexes present in the synaptic cleft. *Structure*, 13:423-34, 2005.

Papers at International Conferences

• **G. Schweikert**, C. Widmer, B. Schölkopf and G. Rätsch. An empirical analysis of domain adaptation algorithms for genomic sequence analysis. *NIPS '09: Advances in Neural Information Processing Systems 21*, pp. 1433–1440, 2009.

Oral Presentations at International Meetings

• **G. Schweikert**, G. Zeller, A. Zien, J. Behr, S. Sonnenburg, P. Philips, C. S. Ong and G. Rätsch. mGene: A novel discriminative gene finder. *Worm Genomics and Systems Biology meeting*, Cambridge, USA, 07 2008

*contributed equally

◦ G. Zeller, **G. Schweikert**, R. Clark, S. Ossowski, P. Shin,, B. Schölkopf and G. Rätsch. Machine Learning Algorithms for Polymorphism Detection. *NIPS '07:* Workshop on New Problems and Methods in Computational Biology, Whistler, Canada, 12 2007

• **G. Schweikert**, A. Zien, G. Zeller, C. S. Ong, F. de Bona, S. Sonnenburg, P. Phillips and G. Rätsch. Ab-initio gene finding using machine learning. *NIPS '06:* Workshop on New Problems and Methods in Computational Biology, Whistler, Canada, 12 2006

◦ **G. Schweikert**, G. Zeller, R. Clark, S. Ossowski, N. Warthmann, P. Shinn, K. Frazer, J. Ecker, D. Huson, D. Weigel, B. Schölkopf and G. Räsch. Machine Learning Algorithms for Polymorphism Detection. *2nd ISCB Student Council Symposium*, Fortaleza, Brazil, 08 2006

◦ **G. Schweikert**, U. Luecken, G. Pfeifer, W. Baumeister and J. Plitzko. The benefit of liquid Helium cooling for cryo-electron tomography: A quantitative comparative study. *The thirteenth European Microscopy Congress*, Antwerp, Belgium, 08 2004