Transcript identification from deep sequencing data

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Diplom-Informatiker
Jonas Tahmoh Behr
aus Freiburg im Breisgau

Tübingen
2013

Tag der mündlichen Qualifikation:	22.04.2014
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Gunnar Rätsch
2. Berichterstatter:	Prof. Dr. Daniel Huson
3. Berichterstatter:	Prof. Dr. Mario Stanke

Abstract

Ribonucleic acid (RNA) sequences are polymeric molecules ubiquitous in every living cell. RNA molecules mediate the flow of information from the DNA sequence to most functional elements in the cell. Therefore, it is of great interest in biological and biomedical research to associate RNA molecules to a biological function and to understand mechanisms of their regulation. The goal of this study is the characterization of the RNA sequence composition of biological samples (*transcriptome*) to facilitate the understanding of RNA function and regulation. Traditionally, a similar task has been addressed by algorithms called *gene finding* systems, predicting RNA sequences (transcripts) from features of the genomic DNA sequence. Lacking sufficient experimental evidence for most of the genes, these systems learn sequence patterns on a few genes with direct evidence to identify many additional genes in the genome.

High-throughput sequencing of RNA (RNA-Seq) has recently become a powerful technology in studying the transcriptome. This technology identifies millions of short RNA fragments (reads of ≈100 letters length), holding direct evidence for a large fraction of the genes. However, the analysis of RNA-Seq data faces profound challenges.

Firstly, the distribution of RNA-Seq reads is highly uneven among genes, resulting in a considerable fraction of genes with very few reads and the stochastic nature of the technology leads to gaps even for well covered genes. To accurately predict transcripts in cases with incomplete evidence, we need to combine RNA-Seq evidence with features derived from the genomic DNA sequence. We therefore developed a method to learn the integration of both information sources and implemented this strategy as an extension of the gene finder *mGene*. The system, now called *mGene.ngs*, determines close approximations of potentially non-linear transformations for all features on the training set, such that the prediction performance is maximized. With this ability, which is to our knowledge unique among gene finding systems, *mGene.ngs* can not only learn complex relationships between the two mentioned information sources, but gains the flexibility to take many additional information sources into account. *mGene.ngs* has been independently evaluated within the context of an international competition (RGASP) for RNA-Seq-based reannotation and has shown very favourable performance for two out of three model organisms. Moreover, we generated and analyzed RNA-Seq-based annotations for 20 *Arabidopsis thaliana* strains, to facilitate a deeper understanding of phenotypic variation in this natural plant population.

A second major challenge in transcriptome reconstruction lies in the complexity of the transcriptome itself. A process called *alternative splicing* generates multiple mature RNA sequences from a single primary RNA sequence by cutting out so-called *introns*, typically in a tightly regulated manner. Inference algorithms of almost all gene finding systems are limited to predict transcripts not overlapping in their genomic region of origin. To overcome this limitation, purely RNA-Seq-based approaches have been developed. However, biologically implausible assumptions or the neglect of available information often led to unsatisfactory results. A major contribution of this study is the integer optimization-based transcriptome reconstruction approach *MiTie*. *MiTie* utilizes a biologically motivated loss function, can take advantage of *a priori* known genome annotations and gains predictive power by considering multiple RNA-Seq samples simultaneously. Based on simulated data for the human genome as well as on an extensive RNA-Seq data set for the model organism *Drosophila melanogaster* we show that *MiTie* predicts transcripts significantly more accurate than state-of-the-art methods like *Cufflinks* and *Trinity*.

Zusammenfassung

Ribonucleinsäuren (RNA) sind polymere Moleküle, die in jeder lebenden Zelle allgegenwärtig sind. RNA Moleküle übermitteln die in der genomischen DNA einer Zelle gespeicherte Information zur Erzeugung der meisten funktionalen Komponenten der Zelle. Daher ist das Entschlüsseln von Funktion und Regulationsmechanismen der RNA in vielen Bereichen der biologischen und biomedizinischen Forschung von großer Bedeutung. Ziel dieser Arbeit ist es die Zusammensetzung von RNA Molekülen in einer biologischen Probe zu charakterisieren, um die Erforschung von Funktion und Regulation der RNA zu unterstützen. Die herkömmliche Herangehensweise an ein sehr ähnliches Problem bedient sich sogenannter *gene finding* Systeme, die RNA Sequenzen (Transkripte) mit Hilfe der genomischen DNA Sequenz vorhersagen. Mangels hinreichender Evidenz für die meisten Gene, werden diese Systeme auf wenigen Genen mit direkter Evidenz trainiert, um viele weitere Gene im Genome vorherzusagen.

Hochdurchsatzverfahren zur Sequenzierung von RNA Molekülen (RNA-Seq) haben sich in jüngerer Vergangenheit als mächtige Werkzeuge zur Untersuchung des Transkriptoms etabliert. Diese Technologie ermöglicht es Millionen von kurzen RNA Sequenzfragmenten ($\approx$ 100 Zeichen Länge) zu entschlüsseln, die direkte Evidenz für die Mehrheit der Gene beinhalten. Allerdings hält die Analyse von RNA-Seq Daten auch große Herausforderungen bereit.

Zum einen sind die Sequenzfragmente sehr ungleich über die Gene verteilt, weshalb für eine bedeutende Anzahl von Genen nur sehr wenig Evidenz zu finden ist. Außerdem entstehen durch die stochastischen Eigenschaften der Technologie auch Lücken ohne Evidenz in Genen, für die insgesamt sehr viel Evidenz vorhanden ist. Um in Fällen mangelnder Evidenz Transkripte akkurat vorherzusagen, müssen zusätzliche Informationsquellen genutzt werden. Zu diesem Zweck haben wir eine Methode entwickelt, die die Integration von RNA-Seq und genomischen Merkmalen auf Trainingsbeispielen lernt. Wir haben diese Strategie im gene finding System *mGene* implementiert.

Das System, das nun *mGene.ngs* heißt, findet gute Approximationen für potenziell nicht lineare Transformationen für alle Merkmale, so dass die Vorhersagegenauigkeit maximiert wird. Diese Fähigkeit, die unseres Wissens unter gene finding Systemen einzigartig ist, ermöglicht es *mGene.ngs* nicht nur komplizierte Zusammenhänge zwischen genomischen und RNA-Seq basierten Merkmalen zu lernen, sondern bringt auch die Flexibilität mit sich, viele weitere Informationsquellen mit zu berücksichtigen.

mGene.ngs wurde im Rahmen des internationalen Wettbewerbs RGASP zur RNA-Seq basierten Genomannotation von einer unabhängigen Jury ausgewertet und war bei zwei von drei der untersuchten Modellorganismen sehr erfolgreich. Außerdem haben wir 20 Stämme des Modellorganismus *Arabidopsis thaliana* mit Hilfe von RNA-Seq Daten annotiert und analysiert, um die Charakterisierung Phänotypischer Variationen in dieser natürlichen Pflanzenpopulation zu unterstützen.

Eine weitere Herausforderung bei der Analyse von RNA-Seq Daten liegt in der Komplexität des Transkriptoms an sich begründet. Ein biologischer Prozess, das alternative *spleißen*, erzeugt verschiedene reife RNA Sequenzen aus der selben Ursprungssequenz, indem unterschiedliche Teile (*Introns*) der Sequenz meist auf genau regulierte Weise herausgeschnitten werden. Aufgrund der verwendeten Inferenzmethoden können gene finding Systeme meist nur eines dieser RNA Endprodukte vorhersagen. Um diese Einschränkung zu umgehen wurden Programme entworfen, die ausschließlich RNA-Seq Evidenz zur Vorhersage von Transkripten verwenden. Allerdings haben diese Programme entweder durch biologisch unplausible Annahmen oder durch die Vernachlässigung vorhandener Information oft keine zufriedenstellenden Resultate geliefert.

Ein wichtiger Beitrag der vorliegenden Arbeit ist das auf diskreten Optimierungsverfahren basierende Programm *MiTie* zur Rekonstruktion von Transkriptomen. *MiTie* verwendet eine biologisch begründete Kostenfunktion, kann Nutzen aus bekannten Genomannotationen ziehen und außerdem gleichzeitig mehrere biologische Proben berücksichtigen, um genauere Vorhersagen zu erzielen.

Mit Hilfe simulierter Daten für das menschliche Genom und auf einem umfangreichen RNA-Seq Datensatz für den Modellorganismus *Drosophila melanogaster* konnten wir zeigen, dass *MiTie* signifikant bessere Transkriptvorhersagen erzielt, als führende Programme wie *Cufflinks* und *Trinity*.

Acknowledgements

I would like to thank Gunnar for teaching me almost everything I know about scientific methods in general and machine learning in particular and for fixing the bugs in my own code via phone. I thank Gabriele and Georg for teaching me everything Gunnar did not teach me and Cheng for patiently explaining the SVM dual to me. I would like to thank all my colleagues at the FML in Tübingen for giving me a good time there and those at MSKCC in New York for the same.

I thank Claudia and Charly for patiently listening to my practice talks and trying to make sense out of what I said. Finally, I thank Nina for spending with me the 108 hours a week I did not work and for her patience during the rest of the time.

Contents

1. Introduction

> *The "gene" is nothing but a very applicable little word, easily combined with others, and hence it may be useful as an expression for the "unit-factors", "elements" or "allelomorphs" in the gametes, demonstrated by modern Mendelian researchers.*
> (Wilhelm Johannsen, 1911)

Remarkably, the terms "gene" as "elements of inheritance", "genotype" as the union of one organisms genes, and "phenotype", as the set of an organisms observable traits, have been defined by Wilhelm Johannsen already in 1909 [58], when the physical manifestation of inheritable elements was still unclear. In the light of modern biochemical research the abstract concept gene as a inheritable unit has changed into a biochemical term [89] describing a region on the genomic DNA sequence.[1]

Similar to Johannsen, but based on a modern gene definition, we may express the relationship between genotype and phenotype as follows. The phenotype of an organism is determined by the combined effect of environmental influences and inherited elements, including genes encoded in the genomic DNA sequence.[2]

However, to answer the question of how the genome contributes to the phenotype of an organism has turned out to be a substantial challenge, even if the entire sequence of the genome is known [e.g. 28, 34]. This can be attributed to the complexity of the processes that regulate the flow of *information* from the genome to functional molecules. The macro molecule ribonucleic acid (RNA) is the first step the information stored in a gene on the genomic DNA sequence takes. RNA molecules in a cell are essentially copies of the genomic sequence generated by a process called transcription. But these copies are then modified in many different ways, until a large fraction finally is translated into an amino acid sequence forming a protein. Proteins perform the vast majority of chemical reactions in a cell, build up the skeleton of the cell and perform essential tasks like metabolite transport. Therefore, proteins determine the phenotype of the organism to a large extend. Other types of RNA molecules serve as regulatory elements, catalyze chemical reactions or form

[1] An exact definition of the term gene is very difficult to give[89]. A working definition may be found in Section 2.1 and is intuitively apparent from Figure 1.1.

[2] A modern definition of the term gene and the implied definition of genotype excludes a large fraction of regulatory elements as well as epigenetic features, which are also known to be inherited "elements". We have to account for these additional inherited elements also influencing the phenotype.

complexes with proteins. The mechanisms by which the processing, translation and degradation of RNA is regulated is essential for understanding the connection of genotype and phenotype. Disentangling this relationship allows us to better understand diseases [e.g. 25, 93], to develop drugs that cure diseases in a personalized manner [e.g. 26] or breed plants that are more robust to pathogens [e.g. 5, 119].

The subject of this thesis is the characterisation of the transcriptional landscape based on diverse sources of information and using diverse machine learning and inference techniques. This task has been approached for many years [e.g. 2, 39, 133], but the complexity of the transcriptome and technical limitations pose profound challenges.

Transcripts in eukaryotic[3] genomes are not necessarily encoded continuously on the genomic DNA sequence. They may be interspersed by so called *introns*, that are *spliced* from the transcribed molecule (*pre-mRNA*[4]) at precisely defined boundaries. The final result of this process is called *mature mRNA*. The task of transcript identification is to find start and end of transcripts as well as the boundaries of introns, called *splice sites*. This task is further complicated by the presence of *alternative isoforms*. In cases where the same pre-mRNA can lead to multiple different mature mRNA molecules with distinct patterns of splicing events we call those mature mRNA molecules alternative isoforms of the gene. The basic terminology of genes and transcripts is illustrated in Figure 1.1.

The task of transcript identification has originally been addressed using *gene finding* techniques. Based on a set of known gene structures (often identified using low-throughput sequencing technologies) these systems train a model based on the sequence content of the genomic DNA, which is then used to find additional genes in the same or a closely related organism.

Such systems however have several shortcomings. Due to the training and inference algorithms, the prediction is often limited to a single isoform per locus. Moreover, since the information of the genomic DNA sequence is nearly identical in each cell of an organism, the gene prediction is static and cannot capture changes in the transcriptome depending on a given tissue or environmental condition.

The characterization of the comprehensive set of RNA-molecules being present at a given time point in the cell is essential for modeling the processes involved in transcriptional regulation and RNA processing. Many studies for human diseases, or in biological sciences rely on the measurement of transcription in a series of time points to measure the effects of a perturbation on the system. This perturbation can be for instance to administer a drug to a cancer patient or the treatment of the

[3]Organisms whos cells are organized in compartments, e.g. having a cell nucleus storing the genomic DNA, belong to the category of *Eukaryotes*

[4]The *m* stands for *messenger* RNA because it carries the information of the protein sequence from the cell nucleus to the *ribosomes* in the cytosol that synthesize the protein

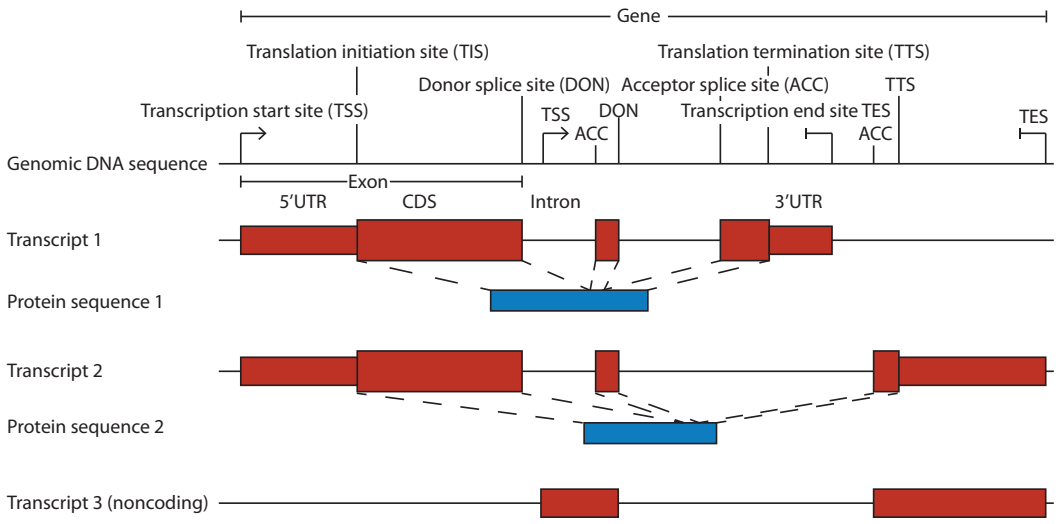


Figure 1.1.: Basic definitions on genes and parts of genes. We illustrate one locus on the genomic DNA sequence encoding a gene with three transcripts. Two transcripts encode the amino acid sequences for two different proteins and one transcript encodes a noncoding RNA molecule. Transcription starts at transcription start sites (TSS) and stops at transcription end sites (TES). Acceptor and donor splice sites precisely indicate boundaries of introns. The region between translation initiation site (TIS) and translation termination site (TTS) is translated into an amino acid sequence after removal of the introns. Parts of exons (or entire exons) that encode amino acid sequences are called coding exons (CDS). Red boxes indicate regions retained in the mature mRNA of a transcript (exons). Here we distinguish between exon parts that encode the amino acid sequence of a protein (high red boxes) and untranslated regions (UTRs)/ noncoding transcripts (low red boxes). Each strand of the genomic DNA sequence is directional. We distinguish between the 5' end and the 3' end for biochemical reasons. Transcription and translation always happen in direction 5' to 3', indicated by the arrow of the TSS sites.

model plant *Arabidopsis thaliana* with a heat shock. The measured changes on the transcriptional landscape may allow us to identify parts of the regulatory network [e.g. 59].

Recent advances in biotechnology have resulted in an immense growth in the availability and the quality of transcriptome measurements. The so called *Next Generation Sequencing* technology allows for sequencing of very large numbers of randomly selected fragments of RNA (termed *RNA-Seq*). Computational analysis of these fragments yields qualitative as well as quantitative information about the transcriptome. Nevertheless, the extensive amount, the significant error rates and biases observed in these measurements state profound challenges for the analysis of these data sets. Therefore, the demand for computational solutions to analyse these

data sets has never been as high as now and is expected to keep growing.

We propose two methods for this task that emphasize different aspects and complement one another. The first method called *mGene.ngs* extends the gene finding system *mGene* to account for different types of transcriptome measurements. *mGene.ngs* is a supervised machine learning approach, which needs examples of known genes to find additional genes in the same or a closely related genome. This method is motivated by the limited detection sensitivity of transcriptome measurements. Transcripts that are expressed at low levels might lack evidence from transcriptome measurements entirely, or are only partially confirmed. The strength of *mGene.ngs* is that it combines prior knowledge about the structure of genes with multiple sources of information. This can resolve ambiguities that emerge when considering individual sources of information independently. Ambiguities might originate from the lack of evidence in one data source or contradictory evidence from different data sources. The integration of the data, specifically, how does a certain level of evidence from a given source of information influence the prediction, is learned on a set of genes with known structure. Taking RNA-Seq measurements into account not only improves the transcript prediction for a given organism, but it allows us to capture the dynamics of transcriptome measurements at different time points or tissue. However, *mGene.ngs* still lacks the ability to predict multiple overlapping transcripts simultaneously.

To address this shortcoming was the main motivation to devise the second tool called Mixed Integer Transcript Identification (*MiTie*). *MiTie* is a pure inference technique. It identifies sets of transcripts and expression levels that maximizes the likelihood of the RNA-Seq data. In comparison to *mGene.ngs*, *MiTie* has two significant conceptual differences. First, it may predict several overlapping transcripts at the same time and second, it relaxes assumptions on the dependencies of output variables. This allows us to model the transcriptome measurements more accurately, while it comes with the downside of a considerable increase of computational complexity. Therefore, the data integration performed by *mGene.ngs* can currently not be learned using the *MiTie* model, although this is in principle possible [9].

MiTie is designed for the case, where a large amount of RNA fragment sequences is available, such that a more complex model with fewer assumptions is identifiable. *mGene.ngs* on the other hand is well suited for cases where partial information from RNA measurements has to be extended and complemented by information from the genomic DNA sequence to predict a single isoform.

Manuscript organization and contributions In the following three chapters, we define the basic terminology first for the genome biology (Chapter 2) and bioinformatics aspects of this thesis (Chapter 3) and then introduce the machine learning and optimization methods employed by *mGene.ngs* and *MiTie* in Chapter 4.

Chapter 5 covers the gene finding specific methodology, related work, extensions developed for *mGene* as part of this theses and applications. The development of *mGene* is a joint effort with several colleagues. The author's contributions are stated in the beginning of the chapter. The genome-based gene finder *mGene*, without the extensions for additional evidences is published in:

> [1] Schweikert G, Zeller G, Zien A, **Behr J**, Dieterich C, Ong C S, Philips P, Bohlen A, Hartmann L, Krüger N, Sonnenburg S, Rätsch G. **mGene: Accurate Computational Gene Finding with Application to Nematode Genomes**. *Genome Research*, 19:2133-2143, 2009.

Results for the RNA-Seq-based annotation of 20 *Arabidopsis thaliana* strains along with a basic description of the methods of *mGene.ngs* are available in the following publication and are discussed in detail in Section 5.5.2.

> [2] Gan X*, Stegle O*, **Behr J***, Steffen JG*, Drewe P*, Hildebrand KL, Lyngsoe R, Schultheiss SJ, Osborne EJ, Sreedharan VT, Kahles A, Bohnert R, Jean G, Derwent P, Kersey P, Belfield EJ, Harberd NP, Kemen E, Toomajian C, Kover PX, Clark RM, Rätsch G, Mott R. **Multiple reference genomes and transcriptomes for** *Arabidopsis thaliana*. *Nature* 477:419-423 doi:10.1038/nature10414, 2011.
> * These authors contributed equally to this work

A more technical description of the concepts of *mGene.ngs* and its relations to *MiTie* can be found in the book chapter:

> [3] **Behr J**, Schweikert G and Rätsch G. **Genome Annotation with Structured Output Learning**. In *S. Nowozin, P. Gehler, J. Jancsary, and C. Lampert, editors, Advanced Structured Prediction*, *The MIT Press*, in press.

SVM based splice site predictors are important components of the gene finding system *mGene*. A comparison of *mGene*'s splice site predictors with related approaches on several organisms may be found here:

> [4] Sonnenburg S, Schweikert G, Philips P, **Behr J**, and Rätsch G. **Accurate Splice Site Prediction using Support Vector Machines**. *BMC Bioinformatics*, 8(Suppl. 10):S7, 2007.

To facilitate the accessibility of the gene finding system *mGene.ngs* by non-computer-scientists we have developed a *Galaxy*-based web service [59]:

> [5] Schweikert G*, **Behr J***, Zien A, Zeller G, Ong C S, Sonnenburg S, Rätsch G. ***mGene.web*: A Web Service for Accurate Computational Gene Finding**. *Nucleic Acids Research*, 37 (suppl 2): W312-W316, 2009.
> * These authors contributed equally to this work

In Chapter 6 we describe the concepts and performance evaluation of *MiTie*. The software package and performance comparisons to competing methods have been published in:

> [6] **Behr J**, Kahles A, Zhong Y, Sreedharan VT, Drewe P and Rätsch G. **MiTie: Simultaneous RNA-Seq-based Transcript Identification and Quantification in Multiple Samples**. *Bioinformatics*, page btt442, 2013.

Parts of *MiTie* have been applied to identify potential *nonsense-mediated* decay targets in intergenic regions of the *Arabidopsis thaliana* genome. Results and experimental validations are accessible here:

[7] Drechsel G, Kahles A, Kesarwani AK, Stauffer E, **Behr J**, Drewe P, Rätsch G, and Wachter A. **Nonsense-mediated decay of alternative precursor mRNA splicing variants is a major determinant of the eukaryotic steady state transcriptome**. *The Plant Cell*, tpc-113, 2013.

SHOGUN is a general purpose machine learning toolbox with a particular focus on fast/large scale implementations of SVMs, string kernels and related sequence analysis tools. The author has contributed improvements to the dynamic programming implementation in *SHOGUN* used by *mGene*, and the interfaces to the different programming languages.

[8] Sonnenburg S, Rätsch G, Henschel S, Widmer C, **Behr J**, Zien A, De Bona F, Binder A, Gehl C, Vojtech F. **The SHOGUN Machine Learning Toolbox**. *Journal of Machine Learning Research*, 99:1799-1802, 2010.

2. Genome Biology

In the first part of this chapter we introduce the basic terms of genome biology with the aim of giving sufficient details such that non-biologists understand the problems we tackle in this thesis. Details not essential for this understanding were excluded from this description. The author is aware of the fact that this has lead to some simplifications. In Section 2.3, we introduce sequencing technologies, which are important driving forces in genome research.

2.1. Definitions

Genome The genome of an organism consists of polymeric molecules called Deoxyribonucleic acid (DNA). It stores essential information about functional molecules and regulatory elements. The four monomers building the linear structures of DNA sequences are guanine, adenine, thymine, and cytosine (abbreviated G, A T and C) collectively called nucleotides. Nucleotides consist of a base distinguishing the four nucleotides ligated to the five-carbon sugar 2-deoxyribose (ribose in case of RNA), which binds a phosphate group at the 5'-carbon. Nucleotides can form polymers by forming a bond between the 3' hydroxy group of the sugar and the phosphate group of another nucleotide. Two chains of DNA can form a double stranded molecule where the bases form hydrogen bonds and the strands wrap into a very stable helix structure. The two pairs G, C and A, T located on corresponding positions on the two strands are energetically highly preferred compared to other base pairings. Apart from the additional hydroxy group of the sugar, in RNA the nucleotide thymine is replaced by uracil. The double stranded state is less stable for RNA than for DNA, nevertheless, double stranded RNA-structures are biologically important. The ends of single stranded DNA and RNA molecules can be distinguished by the C-atom of the ribose which is free to bind another nucleotide. At the 5'-end (3'-end) the 5'-C-atom (3'-C-atom) of the sugar is free, respectively. In many cases the genome of a cell is present in separate polymers called chromosomes.

DNA Replication DNA replication is essential for cell devision. Both daughter cells receive a near identical copy of the original genome. Replication is initiated by separating the two strands. For both strands the enzyme DNA polymerase synthesizes the complementary strand by matching the two pairs of bases with very

high accuracy. DNA polymerase needs a small part of double stranded DNA, which it then extends by adding nucleotides to the 3' sugar of a already paired nucleotide. In order to initiate this process an enzyme called DNA primase synthesizes a short sequence of RNA (primer) complementary to a stretch of single stranded DNA [49].

Gene As previously mentioned, the exact definition of a gene is difficult to give [89]. Throughout this thesis we are going to work with the following definition. A gene is a contiguous part of one strand of DNA sequence encoding the information for a functional element of the cell. The functional element is either an amino acid sequence (protein) or an RNA molecule. In both cases the process of constructing the functional element is mediated by an RNA-copy of a contiguous DNA sequence called transcript. Here, we define a gene as the smallest part of contiguous genomic DNA sequence containing all overlapping transcripts.
In the following section, we will detail the mechanisms of creating RNA transcripts and proteins.

2.2. Gene Expression and Regulation

2.2.1. Transcription: Mechanisms and Regulation

Transcription refers to the process of generating an mRNA copy of a region in the genomic DNA sequence. It is initiated by the binding of the enzyme RNA-polymerase to the promoter region of a gene. The binding is mediated by transcription factors that recognize certain DNA motives called transcription factor binding sites (TFBS) upstream of the transcription start site (TSS).
Elongation of the RNA-molecule happens in the same manner as DNA-replication with the major distinction that the RNA copy quickly detaches from the DNA-template obtaining its single stranded state.
Termination of transcription in eukaryotes is not completely understood. It is likely to be induced by the secondary structure of the newly synthesized RNA and/or by RNA binding proteins and can occur far behind the cleavage site.
Recent studies [86] revealed that transcription start and termination are much more stochastic processes than formerly assumed.

2.2.2. RNA-processing and Posttranscriptional Modification

RNA-processing comprises several important steps that modify precursor mRNA (pre-mRNA) resulting in mature mRNA. Processing takes place in the nucleus of eukaryotic cells and begins during transcription as soon as the functional elements

are accessible.
During splicing particular segments called introns are excised from the pre-mRNA. The splice site at the 5'-end of an intron is called *donor splice site* and the splice site at the 3'-end of an intron is called *acceptor splice site*. After cutting the mRNA at the donor splice site a so called lariat structure is formed by covalent binding of the 5'-end of the intron to a intronic position called *branch point* (br) close to the acceptor splice site. Thereafter, the mRNA is cut at the acceptor splice site and the flanking exons are covalently bound. Accuracy in this step is critical for the functionality of the resulting mature mRNA, because only one nucleotide more or less at the joining of the exons will corrupt the reading frame (see below) for translation. The chemical reaction is preformed by the spliceosome, an assembly of so called small nuclear ribonucleo-particles (snRNPs) and larger proteins. The snRNP's bind the pre-mRNA at specific places around the splice sites and thus contribute highly to their correct recognition.
Splice sites can be classified into at least two subtypes [101]: U2 and U12 dependent splice sites. These terms are justified by different assemblies of snRNPs building the spliceosome. U2-dependent introns comprise the vast majority of introns. U12-type splice sites are found in vertebrates, insects, jellyfish and plants only [101].

Alternative Splicing Alternative splicing is mechanism, that causes variation of mature mRNA representations from the same pre-mRNA. This is accomplished by excluding one or more exons (exon skipping), moving exon/intron boundaries or by retention of introns. The rate of each variant is typically regulated by a complex machinery [16] depending on the given environment. Regulation of acceptor and donor splice sites is due to exonic/intronic splicing enhancers/silencers (ESE, ESS, ISE, ISS). These short sequences flank the splice sites and are recognized by snRNPs [128]. This process is not jet fully understood and it remains an open challenge to accurately predict the rate of splicing given the flanking genomic sequence alone [6, 35].

Posttranscriptinal Modification During transcription GTP is ligated to the 5'-end of the mRNA. The modification called 5'-cap has several functional implications. It renders the 5'-end of the mRNA insensitive to 5'exonucleases[1] and is essential for mRNA recognition, transport and ribosome binding [67].

Rare cases of posttranscriptional modification include the exchange or chemical modification of individual nucleotides. This process is called RNA-editing [66].

Most mature transcripts are endowed with a poly-adenyl tail by an enzyme called Polyadenylate Polymerase (PAP). Polyadenylation is tightly connected to the cleav-

[1]Exonucleases are enzymes degrading RNA by cutting short pieces from either the 3' or the 5'-end.

age process. The Cleavage and Polyadenylation Specific Factor (CPSF) binds to a consensus sequence typically 11 to 23 bases upstream of the cleavage site. About 10-30 nucleotides downstream of the cleavage site the Cleavage stimulation Factor (CstF) binds to the RNA-strand at a weak pattern often referred to as U rich element (URE) and causes cleavage and poly-adenylation together with CPSF. This complex is completed by PAP and further cleavage factors. Sequence of CstF binding site and mentioned distances vary in large ranges between genes and species [56, 73].

2.2.3. Translation: Mechanisms and Regulation

Mature mRNA is transported actively out of the nucleus. Translation into amino acid sequence is done by protein-RNA complexes called ribosomes. Ribosomes bind to the 5'-UTR, which is determined by the start codon (ATG). The start codon defines the reading frame. All triplets of nucleotides following the start codon map to exactly one amino acid except the three so called stop codons (TAG, TAA, TGA) that cause translation to stop. This mapping is mediated by RNA-oligomers called tRNA, that reveal a three nucleotide antisense pattern and carry one specific amino acid according to that pattern. This construction guarantees that one mature mRNA-template determines the exact composition of amino acids to build a functional protein. On the other hand the redundancy of the triplet code leads to the possibility that more than one template codes for one amino acid sequence.
More than 99.9% of the start codons are ATG. But there exist cases where non-ATG codons are used for translation initiation [118]. In most of these cases downstream ATG codons are used for alternative initiation as well. According to Tikole et al. [118] many of the resulting proteins seem to be involved in regulatory and signaling mechanisms (DNA/RNA-binding, kinases, growth factors and immune response).
Termination of translation, when a ribosome is reaching one of the canonical stop codons, TAG, TAA and TGA, is mediated by two release factors (eRF1 and eRF2). Beside the canonical mechanism three other events may occur: According to the context of the stop codon, the codon itself and concentrations of release factors and particular tRNAs the stop codon can be read through (inserting an arbitrary amino acid), the frame can be shifted or re-initiation can occur leading to two disjoint peptides [31].
Read-through can be evoked either by near-cognate or by suppressor tRNA. Suppressor tRNAs are tRNAs matching to one of the three stop codons. This enables tissue specific regulation of three autonomous read-through events, one for each stop codon.
In many viruses (e.g. tobacco mosaic virus) read-through mechanism is employed for regulation of the balance between envelope proteins and polymerases for genome

amplification. In *S. cerevisiae* reading frames only separated by a stop codon with weak termination contexts were identified, indicating that this mechanism does indeed have physiological meaning [31].

2.3. Sequencing Technologies

Advances in genetics research are often driven by advances in sequencing technology. The basis of the whole field of research is the knowledge of the genome of an rapidly increasing number of organisms. In addition sequencing of RNA-transcripts enables qualitative and quantitative measurements of the process of transcription. In this section, we introduce the most important sequencing technologies.

2.3.1. Sanger Sequencing

The Sanger sequencing technology [96], named after the inventor Frederick Sanger, poses the first technology allowing to sequence DNA with relatively low costs and time consumption [1]. Using purified DNA polymerase enzymes and specific primers of length k DNA replication is conducted *in vitro* if 4 different experiments. In addition to endogenous deoxynucleotide triphosphates (dATP, dGTP, dCTP, and dTTP), modified nucleotide [2] are added in low concentration. Whenever a dideoxynucleotide is added to the synthesized DNA sequence the DNA polymer is terminated. Gel or capillary tube electrophoresis [1] determines the length distributions of the synthesized sequences in all four experiments with single base resolution. If capillary tube electrophoresis is used to determine the sequence length then this technique is often referred to as *capillary sequencing*. If each given length $l >= k$ is observed in only one of the four experiments then the sequence of the original DNA is exactly determined up to the first gap. With read lengths of 400 to 900 bases the accuracy of Sanger sequencing is approximately 99.9 percent and costs are estimated to be 2400 US-dollars per million bases [50].

2.3.2. Pyrosequencing

The pyrosequencing technique, also known as 454 sequencing, monitors the incorporation of nucleotides based on the release of pyrophosphate as a side product of forming the phosphodiester bond. The chemiluminescent enzyme luciferase emits light upon the detection of free pyrophosphate. Adding one type of nucleotide triphosphate at a time the emitted light is detected and the intensity is proportional to

[2] dideoxynucleotides lacking the 3' hydroxyl group which is needed to form the phosphodiester bond forming the backbone of the DNA polymer

the number of incorporated nucleotides. Pyrosequencing allows for sequencing of 1 million 700 bp reads within 24 hours at costs of 10 US-dollars per million bases and an accuracy of 99.9% [50].

2.3.3. Illumina Genome Analyzer

The illumina sequencing technology is based on the clonal amplification of DNA on a glass surface. Starting with a large variety of sequence fragments being immobilized on the glass, polymerase amplification results in localized clusters of monoclonal fragments [50]. In each round fluorescently labeled nucleotides with 3' terminal blockers are added to the solution. This leads to incorporation of at most one nucleotide at a time. Then unbound nucleotides are washed away, images are taken and finally, blockers are removed. Illumina sequencing can generate up to 3 billion reads of length 30 to 150 bases in a run taking 5-10 days. Estimated costs per million bases are 0.05 to 0.15 US-dollars [50].

2.3.4. Single-molecule Real-Time Sequencing

Single-molecule real-time sequencing (SMRT) is able to generate sequences of up to 15 kb in length with an average of 2.4 to 2.9 kb [50, 53]. DNA polymerase enzymes are immobilized in perforations of a metal film (zero-mode waveguides). This allows to observe the incorporation of fluorescently labeled nucleotides separately, but highly parallel [65]. Nucleotides are labeled with four different fluorescent dyes being permanently present in the solution. The longer sequences allow to resolve low complexity genomic regions that could not be sequenced unambiguously with previous techniques [19]. The relatively low accuracy of SMRT (87%) can be increased with a protocol called circular consensus sequencing. Here, the same molecule is sequenced several times leading to considerably higher accuracy (99%) at the cost of shorter reads and lower read numbers [50]. The costs are estimated at two US-dollars per million bases.

3. Bioinformatics Background

In the beginning of this chapter, we introduce the terminology and formalizations of bioinformatics problems we discuss in this thesis. We then briefly introduce general concepts of bioinformatics algorithms addressing these problems.

3.1. Formalization of the Problem

The following problem definitions appear mostly identical in [9].

The genome sequence $S = \Sigma^L$ of length L, with $\Sigma = \{A, C, G, T\}$, contains regions that encode G genes. The boundaries (p_g^s, p_g^e) of genes are typically unknown. The genomic sequence $S_g \in \Sigma^{L_g}$ associated with a gene $g = 1, \ldots, G$ of length L_g encodes all transcripts $t_i, i = 1, \ldots, T_g$, that can be produced at any time by any cell. A transcript t is given by its start and end positions, p_t^s and p_t^e respectively, translation start and stop positions, p_t^t and p_t^o, as well as a list of J_t acceptor/donor splice site positions $(p_t^{d_j}, p_t^{a_j}), j = 1, \ldots, J_t$ (see Figure 1.1 for illustration).

Definition 3.1 (*Ab initio* Gene Prediction) *The aim of* ab initio *gene prediction is to predict the boundaries* (p_g^s, p_g^e) *of* all *genes* $g = 1, \ldots, G$ *and* all *transcripts* $t_i, i = 1, \ldots, T_g$, *that are encoded in these genes in the given genome sequence* S. *Hence, we seek a prediction algorithm as follows:*

$$\mathcal{A}_{ai} : S \mapsto \left\{ (p_g^s, p_g^e), \{t_i\}_{i=1}^{T_g} \right\}_{g=1}^{G}.$$

While it would be nice to predict the transcript abundance from the genomic sequence, it is very hard and, to our knowledge, has not been done with reasonable accuracy. Also, most algorithms are not able to predict multiple transcripts and only report one transcript per gene (with very few exceptions[1]).

Experimental evidence is typically associated with a cell (or mixture of cells) in one or more specific condition. In *de novo* gene prediction the experimental evidence is used to improve gene prediction:[2]

[1] Exceptions include work from M. Stanke [e.g. 115] and V. Solovyev [54]. We discuss these approaches in Section 5.4.1.

[2] The term "*de novo* gene finding" has typically been used for gene finding using the genome sequence and conservation. We use it in a broader sense.

Definition 3.2 (*De novo* Gene Prediction) *As in Problem 3.1 one attempts to predict* all *genes and transcripts. In addition to the genome sequence, there is genome-wide experimental evidence X (possibly of different type and of heterogeneous origin) that can be taken advantage of to solve the problem. Hence, we seek a prediction algorithm as follows:*

$$\mathcal{A}_{dn} : (S, X) \mapsto \left\{ (p_g^s, p_g^e), \{t_i\}_{i=1}^{T_g} \right\}_{g=1}^{G}.$$

Ultimately, one is interested in reconstructing the transcriptome and proteome of a single cell or of a collection of cells in a specific state. Here it is important to predict not the transcripts that could potentially be produced, but to predictively reconstruct transcripts that are present in a given cell and to determine their abundance:

Definition 3.3 (Transcriptome Reconstruction) *As in Problem 3.2 one attempts to predict genes and transcripts, however, in a sample-specific manner. In transcriptome reconstruction one additionally has to predict the abundance α_i of transcripts t_i in a biological sample. Hence, we seek a prediction algorithm as follows:*

$$\mathcal{A}_{tr} : (S, X) \mapsto \left\{ (p_g^s, p_g^e), \{t_i, \alpha_i\}_{i=1}^{T_g} \right\}_{g=1}^{G}.$$

If the set of genes and transcripts is already known (for instance, by solving Problem 3.1 or 3.2), then it will be sufficient to solve a simpler problem:

Definition 3.4 (Transcriptome Quantification) *Given a complete set of genes and transcripts, in transcriptome quantification one predicts the abundance α_i of transcripts t_i in a biological sample. Hence, we seek a prediction algorithm as follows:*

$$\mathcal{A}_{tr} : \left(S, X, \left\{ (p_g^s, p_g^e), \{t_i\}_{i=1}^{T_g} \right\}_{g=1}^{G} \right) \mapsto \left\{ \{\alpha_i\}_{i=1}^{T_g} \right\}_{g=1}^{G}.$$

3.2. Computational Assembly

Computational assembly is defined as the inference of long sequences from multiple measurements of fragments of the sequences. The general idea is to find fragments that share a common sequence and are consistent with each other. Two reads are consistent, if they can be represented as substrings of a single sequence. To account for errors in sequencing technology and biological modifications of the sequence the consistency definition is often relaxed in different ways depending on the peculiarities of the sequencing technology and requirements of the computational method.

3.2.1. Genome Assembly

Genome assembly is a computational task aiming to construct chromosome sequences from DNA sequencing measurements. Low complexity sequences are contiguous parts of DNA sequence consisting of repeated occurrences of short sequence patterns. Low complexity regions and multiple copies of identical or similar DNA sequence pose difficulties for genome assembly. The shorter the sequence reads, the higher the repeat content of the genome, and the higher the error rate of the sequencing technique the more ambiguities remain for the sequence assembly.

3.2.2. Methods for Transcriptome Reconstruction

The transcriptome reconstruction task is often addressed using assembly strategies. Then, the problem is related to genome assembly, but it entails two major additional difficulties. While each part of the genome is available at exactly the same abundance in a cell, transcript abundance depends on the cells need for a given function and differs in a wide range between different transcripts, cells (e.g. in separate tissues of a multicellular organism) and time points. The second difficulty originates from alternative splicing. One segment of RNA sequence may be continued with several distinct portions of the same pre-mRNA. The former problem leads to the technical difficulty of observing all transcripts in different ranges of expression, while the latter leads to a conceptual difference between genome and transcriptome assembly. In genome assembly the ground truth is known to be a linear structure. Instead, overlapping transcripts sharing identical sequence parts can be represented as a directed acyclic graph called splicing graph [45]. Lacking the information of long range dependencies, a splicing graph defined on reads from only a few transcripts may encode an enormous set of biologically implausible transcripts in addition. The number of encoded potential transcripts (paths in the graph) is in the worst case exponential in the number of nodes (segments or exons) in the graph.

3.3. Methods for *ab initio* Gene Prediction

While the prediction algorithm only takes the sequence S as input, parameters of the model typically need to be trained on known examples. There are two main sources for training examples. These are, related organisms which have already been annotated, and transcriptome sequencing data [135].

Training a model based on related organisms is conceptually straightforward. However, the success critically depends on the evolutionary distance. Changes in parts of the transcription and splicing regulatory system may reflect on the level of the genomic DNA sequence and thus violates the assumption that training exam-

ples and prediction examples are samples from the same distribution. To account for this protein sequence obtained from the annotated organism may be aligned to the organism of interest, such that the model can be trained directly on the organism of interest. However, this approach assumes a high sequence conservation on the protein level, which may also be violated.

Traditionally, gene prediction efforts were mainly based on relatively long (100-900bp) sequence fragments (Expressed Sequence Tags, ESTs) obtained from sequencing random fragments of cDNA[3] libraries and *full-length cDNAs* [21], ideally corresponding to entire transcript sequences. ESTs and full-length cDNAs were aligned to the genome and ESTs subsequently assembled to transcripts. Due to their length this is relatively straightforward, but because of technical constraints of the sequencing technologies the coverage is relatively low and most of the information is limited to a small fraction of highly expressed genes. To identify additional genes that could not be directly observed using these sequencing technologies, computational models are trained on a reliable subset of assembled transcripts. These models (gene finding systems) infer statistical properties of the gene structures in a training set and try to identify additional gene structures in other parts of the genome. The accuracy of the models can be estimated on a hold out set of assembled transcripts not used for training.

More recent advances in sequencing technology (foremost Illumina sequencing cf. Section 2.3) allow for direct measurements of a much larger fraction of genes. The downside of this technique is the significantly shorter read length posing substantial computational challenges in transcript assembly. In the following, we will give an overview of methods for gene prediction using high throughput sequencing data. We will distinguish between methods primarily based on features obtained from the genomic DNA sequence and methods primarily aiming to assemble sequences from mRNA fragments.

3.3.1. Genome based Methods

Computational methods for gene finding that are primarily based on the genomic DNA sequence belong to the broad category of label sequence learning methods. The goal is to segment a given sequence by assigning each nucleotide to a finite set of states, e.g. the exonic, intronic or intergenic state. Different approaches for this method have quite distinct strengths and weaknesses, but the limitation common to almost all of these methods is that they can predict only one transcript per locus. We will detail the different approaches in Chapter 5.

[3]RNA-molecules are relatively instable and can be degraded rapidly by ubiquitously abundant RNAse enzymes. To facilitate manageability in the laboratory RNA sequences are often converted to DNA sequences (complementary DNA or cDNA) using an enzyme called reverse transcriptase.

3.3.2. RNA-Sequencing based Methods

Most of the gene prediction methods that are mainly based on deep sequencing data do not have the limitation of predicting only a single transcript per locus. Many methods first generate a splicing graph from the read information and then infer transcripts as paths in this graph. However, the strategies of dealing with the exponential number of paths in the graph differ greatly. We present an overview of the methods in Section 6.2.

4. Machine Learning and Optimization

In the following chapter we introduce basic mathematical definitions, algorithms for solving several types of optimization problems and finally a class of machine learning algorithms for binary classification called *Support Vector Machines*.

4.1. Mathematical Notation

4.1.1. General Terms and Notations

We denote the set of real and natural numbers by $\mathbb{R}$ and $\mathbb{N}$, respectively. For a given set of numbers $\mathbb{X}$, $\mathbb{X}_+$ denotes the subset of non-negative numbers and $\mathbb{X}_{++}$ denotes the subset of strictly positive numbers.

In the following, we will list basic definitions used throughout the text.

Definition 4.1 (Domain) *Let function $f : R^p \to \mathbb{R}^q$ be defined on a subset of $\mathbb{R}^p$, then we call this subset the "domain" of f. We will use the notation $dom(f)$ to refer to the domain of f.*

4.1.2. Linear Algebra

Definition 4.2 (Norm) *We call a function $p : \mathbb{R}^n \to \mathbb{R}$ a "norm" if (1) for all $u \in \mathbb{R}^n$ $p(u) \geq 0$ and $p(u) = 0$ if and only if u is the zero vector, (2) for all $u, v \in \mathbb{R}^n$ $p(u+v) \leq p(u) + p(v)$ and (3) $|a| \cdot p(u) = p(a \cdot u)$ for all $a \in \mathbb{R}$ (positive scalability).*

For $u \in \mathbb{R}^n$ and $p \in \mathbb{N}_{++}$ we define the l_p-*norm* as

$$\|u\|_p = \left[\sum_{i=1}^{n} |u_i|^p \right]^{\frac{1}{p}}.$$

Moreover, we adopt the term l_0-*norm* commonly used in the machine learning community to refer to

$$\|u\|_0 = \sum_{i=1}^{n} I(u_i \neq 0),$$

although this is not a norm according to Definition 4.2 (it does not satisfy the positive scalability criterion). To simplify notation here and throughout this thesis we use the *indicator function* $I(.)$. Given an expression (in the common computer science sense) this function returns a one if the expression evaluates to *true* and zero otherwise.

The scalar product $\langle u, v \rangle$ of vectors $u, v \in \mathbb{R}^n$ is defined as

$$\langle u, v \rangle = \sum_{i=1}^{n} u_i \cdot v_i.$$

Definition 4.3 (Affine Function) *A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is called affine if it is a sum of a linear function and a constant, i.e., if it has the form $f(x) = Ax + b$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ [15].*

Definition 4.4 (Hyperplane) *The set $h(w, b) \subset \mathbb{R}^n$ with $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$*

$$h(w, b) = \{x | \langle w, x \rangle + b = 0\}$$

is called hyperplane. w can be accounted for a normal vector on the hyperplane and b (bias) is the distance of the hyperplane from the origin.

4.2. Optimization

General Optimization Problem Without loss of generality we define an optimization problem over variable $x \in \mathbb{R}^N$, with m inequality constraints and n equality constraints as follows [15]:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, i \in 1, ..., m \\ & h_i(x) = 0, i \in 1, ..., n \end{aligned} \tag{4.1}$$

This formalizes the goal to find a value for vector x such that $f(x)$ is minimal among all vectors x satisfying the m inequality and n equality constraints. For simplicity of notation we assume that functions f, g_i and h_i are defined on $\mathbb{R}^N$. The *feasible set* $\mathcal{F}$ of the optimization problem is defined as the set of vectors satisfying all constraints, i.e. $\mathcal{F} = \{x | g_i(x) \leq 0 \ \forall i \in \{1, ..., m\}, h_i(x) = 0 \ \forall i \in \{1, ..., n\}\}$. If $\mathcal{F}$ is nonempty we call the optimization problem *feasible* and *infeasible* otherwise.

Convex Functions

Definition 4.5 (Convex Function) *A function $f : \mathbb{R}^n \to \mathbb{R}$ is called "convex" if for any two vectors $x \in dom(f)$ and $y \in dom(f)$ and any $0 \leq \theta \leq 1$:*

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$

f is called "strictly convex" if for $x \neq y$ equality only holds if $\theta \in \{0, 1\}$. f is called "concave" if $-f$ is convex.

Based on Definition 4.5 it is straightforward to see that affine functions are convex. It can easily be shown that a weighted sum of convex functions is convex if all weights are nonnegative [15].

Duality A general optimization problem as defined in 4.1 can be viewed from either of two perspectives [15]. There are different approaches to change from the initial optimization problem, referred to as the *primal optimization problem* or just the *primal*, to the *dual optimization problem* or *dual*. Here we will introduce the Lagrange dual problem. The basic idea of the Lagrange dual is to substitute the constraints of an optimization problem into the objective function, to obtain a lower bound (*Lagrangian function* or *Lagrangian*) of the original objective for all $x \in \mathcal{F}$. Given an optimization problem in standard form (4.1) the Lagrangian can be defined as

$$L(x, \alpha, \beta) = f(x) + \sum_{i=1}^{m} \alpha_i \cdot g_i(x) + \sum_{i=1}^{n} \beta_i \cdot h_i(x), \tag{4.2}$$

with $\alpha \in \mathbb{R}_+^m$ and $\beta \in \mathbb{R}^n$. We will call x the vector of primal variables and α and β vectors of dual variables. Using the Lagrangian we define the *Lagrange dual function*

$$\mathcal{L}(\alpha, \beta) = \min_x L(x, \alpha, \beta). \tag{4.3}$$

The dual optimization problem corresponding to 4.1 can then be defined as

$$\begin{aligned} \max_{\alpha,\beta} \quad & \mathcal{L}(\alpha, \beta) \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i \in 1, ..., m. \end{aligned} \tag{4.4}$$

Once we have found the optimal values for the dual function $\tilde{\alpha}$ and $\tilde{\beta}$ we can compute the optimal value $\tilde{x}$ for the primal as

$$\tilde{x} = \operatorname*{argmax}_x L(x, \tilde{\alpha}, \tilde{\beta}) \tag{4.5}$$

In general this problem can be solved if we are able to compute the dual function.

Example In this example we demonstrate how to solve a small constrained optimization problem using the Lagrange dual. Figure 4.1 shows a plot of the objective function, the Lagrangian and Matlab/octave code for creating the plot.

Given the optimization problem

$$\begin{aligned} \min_x \quad & x^3 - 6x \\ \text{s.t.} \quad & x^2 \leq 1 \end{aligned}$$

we obtain the Lagrangian

$$L(x, \alpha) = x^3 - 6x + \alpha(x^2 - 1).$$

Figure 4.1 shows the Lagrangian for different values of α. We then define the dual function and solve analytically for x by setting $\frac{\partial}{\partial x} L(x, \alpha) = 0$ and solving for x:

$$\begin{aligned} \mathcal{L}(\alpha) &= \min_x L(x, \alpha) \\ &= \min_x x^3 - 6x + \alpha(x^2 - 1) \\ &= \tilde{x}^3 - 6\tilde{x} + \alpha(\tilde{x}^2 - 1) \end{aligned}$$

with $\tilde{x} = -\frac{1}{3}\alpha + \sqrt{\frac{1}{9}\alpha^2 + 2}$. We then perform a line search maximizing $\mathcal{L}(\alpha)$ and substitute into the Lagrangian. The Lagrangian with optimal α has a minimal value for x corresponding to the minimal value of $f(x)$ $\forall x \in \mathcal{F}$.

Applicability For various reasons an optimization problem may be easier to solve in the dual than in the primal. If the number of variables in the primal is large, the dual might be favourable, where the number of variables corresponds to the number of inequality constraints. Moreover, in some very important cases the functional form of the dual is easier tractable. In the case of Support Vector Machines the functional form of the dual does not rely on explicit representations of vectors in the primal optimization space. This allows to solve optimization problems with extremely large (even infinite) numbers of primal variables using *kernel functions* (cf. Section 4.4).

```
% range of values
x = -2:0.01:2;
alphas = 0:0.01:3;
% objective function
y = x.^3 - 6*x;
% compute the Lagrangian for different values
% of Lagrange multipliers alpha
for alpha = alphas,
   L = x.^3 - 6*x + alpha * (x.^2 - 1);
   plot(x, L); hold on;
end
plot(x, y, 'r');
% compute the dual function, set derivative of the
% Lagrangian with respect to x to zero and solve
cnt = 0;
for alpha = alphas
   cnt = cnt+1;
   x1 = -1/3*alpha + sqrt((1/3*alpha)^2+2);
   D1(cnt) = x1^3 + -6*x1 + alpha*(x1^2 -1);
end
% plot Lagrangian for maximal alpha
[m, idx] = max(D1);
alpha = alphas(idx);
L = x.^3 - 6*x + alpha * (x.^2 - 1);
plot(x, L, 'k');
```

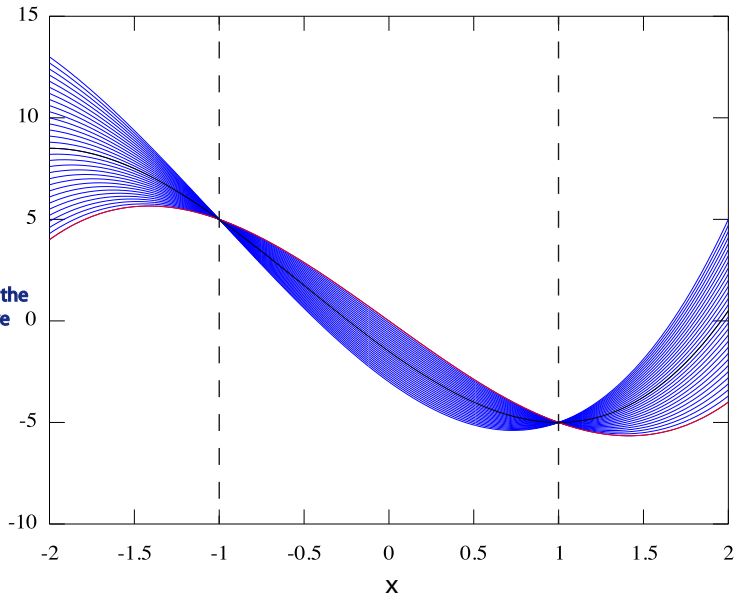


Figure 4.1.: Example: Solving a constrained optimization problem in the dual with matlab/octave. The objective function is shown in red and the Lagrangian for different multiplier values is shown in blue. In black is shown the Lagrangian with optimal α (having the maximal lower bound of all blue lines). The feasible set is indicated with dashed lines.

The dual of an optimization problem is always concave, even if the original optimization problem in non-convex [15]. To show this, we consider the functional form of the Lagrangian for a fixed primal vector $\tilde{x}$. Let $\Phi_i = g_i(\tilde{x}), i = 1, ..., m$, $\Psi_i = h_i(\tilde{x}), i = 1, ..., n$, and $\gamma = f(\tilde{x})$, then we can rewrite the Lagrangian as

$$L(\tilde{x}, \alpha, \beta) = \alpha^T \Phi + \beta^T \Psi + \gamma, \tag{4.6}$$

which is clearly an affine function. Thus, the dual function is the minimum over a set of affine functions (being convex and concave). The maximum of a set of convex functions is convex, since this operation corresponds to the intersection of convex sets, which is known to be convex. The same clearly holds true for the minimum of concave functions. We present efficient algorithms to solve convex optimization problems in Section 4.2.1.

Thus, we can obtain a lower bound for the original optimization problem by solving the dual.

4.2.1. Solving Convex Optimization Problems

In this thesis, we formulate all optimization problems as convex optimization problems. In the following, we will introduce numerical strategies to solve convex optimization problems. This description is largely based on Chapter 10 and Chapter 11 of [15]. Firstly, we will present Newton's method for solving convex optimization problems with linear equality constraints and then demonstrate how to approximate a convex optimization problem with inequality constraints using interior point methods.

Newton's Method with Equality Constraints We aim to solve an optimization problems of the form:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & Ax = b. \end{aligned} \tag{4.7}$$

We assume we are given a feasible starting point x_0. The algorithm iteratively increments x_t such that for each step t the increment Δx_t satisfies

$$A\Delta x_t = 0.$$

With $Ax_t = b$ we have $A(x_t + \Delta x_t) = b$, guaranteeing that each step satisfies the equality constraints. Similar to the unconstrained Newton method steps are performed by minimizing the second order Taylor approximation of the objective function at the current solution x_t:

$$f(x_t + \Delta x_t) \approx \hat{f}(\Delta x_t) = f(x_t) + \nabla f(x_t)^T \Delta x_t + \frac{1}{2}\Delta x_t^T \nabla^2 f(x_t)\Delta x_t \tag{4.8}$$

We find the optimal step Δx_t by solving the optimization problem:

$$\begin{aligned} \min_{\Delta x_t} \quad & \hat{f}(\Delta x_t) \\ \text{s.t.} \quad & A\Delta x_t = 0 \end{aligned} \tag{4.9}$$

Using the Lagrangian

$$L(\Delta x_t, \alpha) = \quad f(x_t) + \nabla f(x_t)^T \Delta x_t + \frac{1}{2}\Delta x_t^T \nabla^2 f(x_t)\Delta x_t + (A\Delta x_t)^T \alpha \tag{4.10}$$

and

$$\begin{aligned} & \frac{\partial L(\Delta x_t, \alpha)}{\partial \Delta x_t} = 0 \\ \Leftrightarrow \quad & \nabla^2 f(x_t)\Delta x_t + A^T\alpha = -\nabla f(x_t) \end{aligned} \tag{4.11}$$

we obtain the system of equations

$$\begin{bmatrix} \nabla^2 f(x_t) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \alpha \end{bmatrix} = \begin{bmatrix} -\nabla f(x_t) \\ 0 \end{bmatrix} \tag{4.12}$$

which can be solved analytically [15, page 526].

Approximation of Convex Optimization Problems with Inequality Constraints
A wide spread method to numerically solve convex optimization problems with inequality constraints is called interior point method or barrier method. Optimization problems of the form

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, i = 1, ..., m \\ & Ax = b \end{aligned} \tag{4.13}$$

can be approximated with arbitrary accuracy ϵ by substituting the inequality constraints into the objective function as follows:

$$\hat{f}(x) = f(x) - \frac{1}{t}\sum_{i=1}^{m} log(-g_i(x)) \tag{4.14}$$

The domain of $\hat{f}$ is limited to feasible points of the original optimization problem. As soon as x approached the boundaries of the feasible set $\hat{f}(x)$ approaches infinity. Parameter t is closely related to the approximation accuracy ϵ[1]. With $t \to \infty$ $\hat{f}(x)$ approaches the step function with $\frac{1}{t}log(-g_i(x)) \approx 0$ if constraint i is strictly satisfied and $\frac{1}{t}log(-g_i(x)) \to \infty$ if $g_i(x) \to 0$. If f and g_i are twice continuously differentiable the approximate optimization problem

$$\begin{aligned} \min_x \quad & \hat{f}(x) \\ \text{s.t.} \quad & Ax = b \end{aligned} \tag{4.15}$$

$$\tag{4.16}$$

[1] $\epsilon = m/t$ proof: [15, page 566]

can be solved using Newton's method. For large values of t the gradient becomes very small, causing numerical difficulties with Newton's method. Therefore, the optimization is started with moderate values for t and iteratively increases t using the solution of the previous iteration as a starting point for the next iteration.

4.2.2. Combinatorial Optimization

Optimization problems with discrete output spaces cannot be solved with the previously described methods. Under certain assumptions on the dependencies of output variables such problems can be solved efficiently using *Dynamic Programming*. Most importantly, dynamic programming is advantageous if we can split the objective function into parts, that may be computed independently. We describe the theory and application of such models in Chapter 5.

Branch and Bound In Chapter 6 we make use of quadratic optimization techniques where a subset of the variables is binary. In this case the dependency structure of output variables suggests that dynamic programming based solution strategies are suboptimal. Instead the optimal solution has to be found by trying combinations of choices for the integer variables. Given n binary variables an exhaustive search needs to check all 2^n combinations. A search tree can be defined by choosing an order on the binary variables and then fixing these variables to one of the two values. Once we entered a branch of the tree, by fixing one binary variable, this variable will be treated as a constant in all computations in this branch. A node at depth k in the tree is then defined by the values of the first k variables that have been fixed.

Branch and bound techniques can be used to avoid an exhaustive search of this tree by pruning parts of the search space that cannot yield the optimal solution. This is based on the following observation. The *relaxed* version of the optimization problem, where we allow intermediate values also for the binary variables ($x_i \in [0, 1]$), will have an optimal objective value smaller or equal (in the case of minimization problems) to the optimal objective value of the original problem. The solution of the relaxed problem $\hat{x}_s$ at node s therefore serves as a lower bound for the optimal solution that may be found in the branch associated with s.

An upper bound for the global optimal solution can be derived by performing a depth first search for a feasible integer solution. The currently best integer solution we denote by x^*. Whenever we reach a feasible leave node (a node at depth n) we update x^* if necessary.

The branch and bound algorithm for solving mixed integer optimization problems searches through the tree and performs the following steps at each node s: (1) it checks the constraints and prunes the branch below s, if the so far fixed variables violate a constraint and (2) it solves the relaxed optimization problem with k fixed

variables to obtain $\hat{x}_s$. If the $\hat{x}_s$ is larger or equal to x^*, this branch cannot lead to a better solution and therefore it can been pruned. Otherwise, the algorithm starts searching the branch associated with node s.

Since the lower bound generated by the relaxed version of the optimization problem might not be very tight, there is no guarantee that the runtime is better than an exhaustive search. We will discuss the relationship of problem formulation and runtime of the branch and bound algorithm in Section 6.

Branch and Cut The result of the relaxed optimization problem determines, whether the current branch has to be searched or may be pruned. Therefore, a variety of strategies has been developed to improve the lower bound obtained from the relaxed solution [78]. The general idea is to find constraints that prune parts of the feasible set of the relaxed optimization problem without excluding feasible integer solutions. Ideally, one could identify constraints such that the feasible set of the relaxed optimization problem is restricted to the convex hull of the feasible integer solutions. Then, the result of the relaxed solution (using the simplex algorithm) would also provide the optimal integer feasible solution [78] (see Figure 4.2 for an illustration). However, enumerating all necessary constraints is in general not feasible [78]. Nevertheless, strategies for generating constraints to restrict the feasible set of the relaxed optimization problem have been successful in practice [78]. General purpose combinatorial optimization solvers often include one or several such strategies, such that the user of such packages does not necessarily need to reimplement them. We will discuss one branch and cut method to illustrate the idea.

We discuss the branch and cut strategy introduced by Crowder and Johnson [24], which may be applied for binary optimization problems with N variables and M constraints of the form:

$$\begin{aligned} \max_{x \in \{0,1\}^N} \quad & F(x) \\ s.t. \quad & Ax \geq b \end{aligned}$$

for given $A \in \mathbb{R}^{M \times N}$ and $b \in \mathbb{R}^M$. If there is any constraint i

$$\sum_{j=1}^{N} A_{i,j} x_j \geq b_i \tag{4.17}$$

with coefficients

$$A_{i,j} \geq 0 \quad \forall j = 1, ..., N, \tag{4.18}$$

then we may replace any coefficient $A_{i,j} > b_i$ with b_i and thereby reduce the feasible

set of the relaxed problem, while not excluding any feasible integer solution [24]. This can be easily seen, since setting x_j to one will already satisfy the constraint, independent of all other variables. On the other hand, if we set the variable x_j to zero, the coefficient does not matter. Moreover, it it possible to convert all constraints into the form with only positive coefficients by substituting x_j with $(1 - x'_j)$ whenever needed [24].

Example Figure 4.2 illustrates this strategy based on the following optimization problem:

$$\begin{aligned} \max_{x_1,x_2\in\{0,1\}} \quad & 2x_1 + x_2 \\ s.t. \quad & x_1 + x_2 \leq 1.5 \end{aligned}$$

We now transform the only constraint by substituting $x_1 = 1 - x'_1$ and $x_2 = 1 - x'_2$:

$$\begin{aligned} -x_1 - x_2 \quad & \geq -1.5 \\ \Leftrightarrow x'_1 + x'_2 \quad & \geq 0.5 \end{aligned}$$

Setting the coefficients larger than the right hand side to that value leads to:

$$\begin{aligned} 0.5x'_1 - 0.5x'_2 \quad & \geq 0.5 \\ \Leftrightarrow x'_1 + x'_2 \quad & \geq 1 \end{aligned}$$

and back transformation:

$$\Leftrightarrow x_1 + x_2 \quad \leq 1$$

Including the *box* constraints $x_i \in [0, 1]$, this constraint reduces the feasible set of the relaxed optimization problem directly to the convex hull of the feasible integer solutions and the result of the relaxed problem will be the optimal integer solution.

4.3. General Concepts in Machine Learning

Machine learning is the automatic extraction of information from data by means of statistical methods [129]. There are two major subclasses of machine learning algorithms, supervised and unsupervised methods. Unsupervised machine learning methods identify statistical properties in unlabeled data. Clustering algorithms represent the most well known class of unsupervised learning algorithms. All methods discussed in the following belong to the class of supervised learning algorithms. Therefore, we will focus on supervised learning methods in our description.

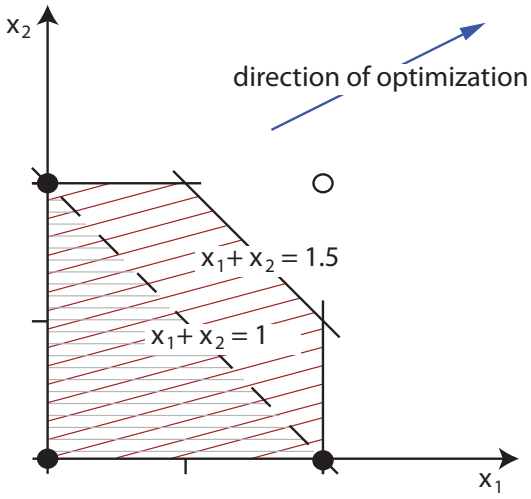


Figure 4.2.: Branch and cut (see Section 4.2.2 for problem definition and details). We illustrate a simple binary optimization problem with two variables. The solid dots indicate the feasible integer solutions, the gray shaded area is the convex hull of the integer feasible solutions and the area shaded in red indicates the feasible set of the relaxed problem (including box constraints $x_i \in [0, 1]$). The newly generated constraint (dashed line) restrict the feasible set of the relaxed problem, without excluding any feasible integer solutions.

Supervised machine learning algorithms aim to identify parameters θ of a function $f_\theta : X \to Y$ given a set of m training examples $(x_i, y_i) \in X \times Y, 0 \leq i < m$. Once the parameters θ are determined the function f_θ can be used to predict on new instances $x \in X$.

$$\hat{y} = f_\theta(x)$$

The performance of the prediction can be evaluated on instances from the training set (training error) or a test set (test error) using an evaluation function $e : Y \times Y \to \mathbb{R}$. A test set is a set of pairs $(x_i, y_i) \in X \times Y, 0 \leq i < n$ independently drawn from the same distribution as the training set, but not overlapping with the training set. If Y is finite, the machine learning task is referred to as a classification problem otherwise it is called a regression problem. Structured Output Learning (SOL) comprises learning problems (classification or regression), where Y is arbitrarily complex. If Y is the set of sequences on a finite alphabet, the problem is referred to as Label Sequence Learning (LSL).

4.3.1. Decision Theory

In the following, we will present three distinct ways of taking the class decision in supervised classification. Taking a probabilistic point of view our training data

$(x_i, y_i) \in X \times Y, 0 \leq i < m$ are drawn from a joint probability distribution with probability $P(x, y)$. We will assume that each observation was drawn independently from this distribution. The joint probability $P(x, y)$ of observing data point x with label y can be divided into the conditional probability $P(y|x)$ of observing label y given the data point x and the *marginal probability* $P(x)$:

$$P(x, y) = P(y|x) \cdot P(x) \tag{4.19}$$

Equivalently, we can express the joint probability as a product of the class prior distribution $P(y)$ and class densities $P(x|y)$:

$$P(x, y) = P(x|y) \cdot P(y) \tag{4.20}$$

Equalizing 4.19 and 4.20 we obtain

$$P(y|x) = \frac{P(x|y) \cdot P(y)}{P(x)} \tag{4.21}$$

known as *Bayes' theorem*.

Intuitively, we only need to know the probabilities $P(y|x)$ for each $y \in Y$ to take the decision of class assignment to minimize the rate of misclassification. The simple proof for this intuition can be found in [10].

Generative Learning Generative learning methods aim to estimate $P(y|x)$ by estimating all quantities on the right hand side of 4.21. While the class prior $P(y)$ can be estimated as the fraction of observations with label y, estimation of the class densities $P(x|y)$ is generally a very hard problem needing large data sets for training [10]. If $P(x|y)$ and $P(y)$ are known, the marginal probability $P(x)$ can be computed as:

$$P(x) = \sum_{y \in Y} P(x|y) \cdot P(y) \tag{4.22}$$

Discriminative Learning Discriminative learning methods directly model the class probabilities $P(y|x)$ without estimating the individual class densities. This is considerably easier, especially when there is complex structure in the different class densities in areas not important for the decision [10]. On the other hand models for class densities are able to quantify the uncertainty in the decision in terms of the marginal probability. If a new data point x has low marginal probability it represents an outlier or a novel type of observation and therefore we will be less confident in the prediction [10].

Decision Functions Finally, there are methods directly deriving decision functions without modelling the class probabilities. Among those are linear models for binary classification separating the feature space X into two half spaces associated with the two classes. While these methods have been proven very powerful in many applications [e.g. 108] the ability of quantifying the uncertainty in the prediction is lost [10].

4.3.2. Regularization

A general concern of supervised learning strategies is how well a trained model performs on the test set, i.e. how well the model *generalizes*. A measure for the generalization capability of a model can be obtained by comparing the training and the test performance of the model. If the training error is much smaller than the test error (the model has a low generalization capability) we call the model *overtrained*.

Important questions in supervised learning are when does overtraining happen and how can we avoid it. The generalization capabilities of a model depend on the size of the training set and the *complexity* of the model.

A widely applied measurement for the complexity of a model is the Vapnik-Chervonenkis (VC) dimension [125]. Here we use the definition of the VC dimension from [11] for an M dimensional Euclidean space X. A *concept* corresponds to a region of X and a *concept class* is a nonempty set of concepts, e.g. the set of half spaces in X.

Definition 4.6 (VC-dimension) *Given a nonempty concept class $C \subseteq 2^X$ and a set of points $S \subset X$, $\Pi_C(S)$ denotes the set of all subsets of S that can be obtained by intersecting S with a concept in C, that is, $\Pi_C(S) = \{S \cap c : c \in C\}$. If $\Pi_C(S) = 2^S$, then we say that S is shattered by C. The Vapnik-Chervonenkis (VC) dimension of C is the cardinality of the largest finite set of points $S \subset X$ that is shattered by C. If arbitrarily large finite sets are shattered, the VC dimension of C is infinite.*

Intuitively, the VC dimension is the maximal number of points where any binary labeling of the points can be correctly reproduced by the model.

Application to Machine Learning The rate of misclassification of training examples or *empirical risk* $R_{\text{emp}}(\theta)$ of a model $f_\theta : X \to \{-1, 1\}$ for binary classification of examples $x_1, ..., x_N \in X$ is computed as:

$$R_{\text{emp}}(\theta) = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{1}{2}(y_i - f_\theta(x_i)) \tag{4.23}$$

Based on the empirical risk and the VC dimension of the model a probabilistic upper bound on the misclassification rate on new data can be given [e.g. 76]. With probability $1 - \eta$ the rate of misclassification on new samples from the same distribution as the training set is

$$R(\theta) \leq R_{\text{emp}}(\theta) + \sqrt{\frac{h \cdot \left(log\left(\frac{2N}{h}\right) + 1\right) - log\left(\frac{\eta}{4}\right)}{N}}. \tag{4.24}$$

Where N is the number of training examples, $0 < \eta < 1$ and h is the VC dimension of the model. Based on this formulation the general concept of *regularization* arises. According to Equation 4.24 we need to minimize training error and model complexity at the same time to have good performance on the test set with high probability.

A *regularization* function $\Omega : \mathbb{R}^M \rightarrow \mathbb{R}$ maps points in parameter space to real values. Low values of a regularization function correspond to low complexity of model f measured e.g. as the VC dimension of f_θ. Using a regularization function the trade off between training error and model complexity can be formalized in the following way [117]:

$$R(\theta) = \lambda\Omega(\theta) + R_{\text{emp}}(\theta) \tag{4.25}$$

The parameter $\lambda > 0$ has to be determined using model selection. By varying measurements of the empirical error (loss functions) and regularization functions, a large class of widely used machine learning methods arise [117]. We will see in Section 4.4 that a very popular class of machine learning methods called Support Vector Machines (SVMs) are a special case of this formulation.

4.3.3. Model Selection

Model selection is a technique to optimize meta parameters of an algorithm A. Using a single value performance measurement $f_A(\theta)$ for A and meta parameters θ standard optimization techniques like grid search, simulated annealing[2] and gradient descent[3] methods are applicable in the meta-parameter space. The choice of the

[2] Simulated annealing is an optimization technique that performs random jumps in parameter space. The direction of movement is controlled by rejecting jumps according to particular criteria. Jumps that improve f are always accepted, whereas jumps impairing f in the optimization sense are accepted if the loss is below a threshold t. t is decreased during training. Simulated annealing is applicable for optimization in discrete, continuous and mixed spaces, where some dimensions are continuous and others are discrete, but there is no guarantee that either a global or a local optimum is found.

[3] Gradient decent methods calculate a direction D depending on ∇f_A and then change the parameters by $s \cdot D$. s is the step size calculated heuristically using for example quadratic interpolation of f in direction D.

method depends largely on the number and type of meta parameters and on A. Grid search, the most simple technique, uses discretized sets of values for each meta parameter. For each combination of meta parameter values f_A is computed. The run time of a grid search is therefore given by

$$T = \mathcal{O}(e^{\text{number of meta parameters}} \cdot T(A)).$$

Where $T(A)$ is the run time of A. Independent of $T(A)$ grid search is not applicable if the number of meta parameters is large.
If f_A is defined such that

$$\frac{\partial f_A}{\partial p}$$

for any meta parameter p can be calculated, gradient descent methods are applicable.
In machine learning one has to estimate f using an additional set of examples called validation set. Just like the error rates on the training set the error rate on the validation set are of interest but do not constitute an estimation of the overall performance of the algorithm.

4.3.4. Cross Validation

Cross validation is a technique to derive test predictions for all available data. From a single machine trained on one part of the data we can only obtain test predictions for the rest of the data. Using n-fold cross validation one splits the data into n parts and trains n machines on $n-1$ parts such that for each part there is one machine that has not seen this particular part in training.
The smaller n is the less machines have to be trained and the bigger n is the more data is left for training of each machine. Setting n is therefore a trade off between runtime and accuracy.

4.4. Binary Classification with Support Vector Machines

Support Vector Machines (SVMs) are a powerful machine learning technique for classification problems. We will show how the SVM formulation arises as a special case of the regularized risk minimization (cf. [117]). In this section, will focus on binary classification and discuss multi class classification in Section **??**.

Empirical Risk of Linear Separators Given the M dimensional Euclidean space X, a hyperplane can be written as

$$S(w,b) = \{x \in X | \langle w, x\rangle + b = 0\} \quad , w \in X, b \in \mathbb{R}.$$

w can be accounted for an orthogonal vector to the hyperplane. S is invariant to scaling w and b with the same nonzero factor. We define the parameter vector $\theta \in \mathbb{R}^{M+1}$ as a concatenation of w and b and obtain the decision function:

$$f_\theta(x) = sgn(\langle w, x\rangle + b).$$

The empirical risk of f_θ is:

$$R_{\text{emp}}(\theta) = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{1}{2}(y_i - f_\theta(x_i)) \tag{4.26}$$

$$= \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{1}{2}(y_i - sign(\langle w, x_i\rangle - b)) \tag{4.27}$$

Hinge Loss The empirical risk computed in Equation 4.27 is a step function also referred to as the *0-1-loss function* (cf. Figure 4.3). Minimizing 4.25 using the 0-1-loss function to compute the empirical risk is a non-convex optimization problem and computationally very challenging. To enable the application of SVMs for large training sets a convex upper bound for the 0-1-loss called *hinge loss* is used in SVMs (cf. Figure 4.3). The hinge loss is computed as:

$$l_\theta(x_i) = max(0, y_i \cdot (\langle w, x_i\rangle - b) + 1) \tag{4.28}$$

Using the hinge loss and $\Omega(w) = \frac{1}{2}\|w\|_2^2$ we obtain the formulation of an SVM from 4.25:

$$\begin{aligned} \min_{w \in X, \zeta} \quad & r(\boldsymbol{w}, \boldsymbol{\zeta}) = \frac{1}{2}||w||_2^2 + \frac{C}{N}\sum_{i=1}^{N}\zeta_i, \quad C \in \mathbb{R}_+ \\ s.t. \quad & y_i(\langle w, x_i\rangle + b) \geq 1 - \zeta_i, \quad i = 1, ..., N, \\ & \zeta_i \geq 0, \quad i = 1, ..., N \end{aligned} \tag{4.29}$$

C is a hyper-parameter of the SVM. The choice of the l_2 norm as regularizer is justified by learning theoretic results we review in Section C. The so called *slack variables* ξ_i quantify misclassification of sample i by implementing the hinge loss. It determines the trade off between performance on the training set and generalization

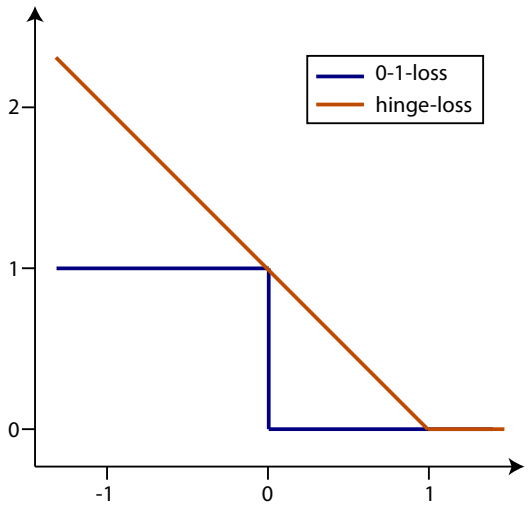


Figure 4.3.: **Hinge loss.** 0-1-loss counts the number of misclassified examples. The hinge-loss is an upper bound of the 0-1-loss.

ability. For each trained classifier C has to be determined by model selection.

4.4.1. Solving the SVM Optimization Problem

The optimization problem 4.29 can be solved directly in the primal formulation using general purpose solvers for quadratic optimization problems with linear constraints (cf. 4.2.1). The primal optimization problem has $M+N$ variables and N constraints.

If the number of feature dimensions M is significantly larger than the number of examples N it is more efficient to solve the SVM in the *dual* space. The resulting quadratic optimization problem has N variables and N constraints.

The SVM dual As described in Section 4.2 we can create a dual representation of the SVM by first substituting the constraints into the objective function using Lagrange multipliers $\alpha \geq \mathbf{0} \in \mathbb{R}^M$.

$$L(w, b, \zeta, \alpha, \nu) = \frac{1}{2}||w||^2 + \frac{C}{N}\sum_{i=1}^{N}\zeta_i - \sum_{i=1}^{N}\alpha_i(y_i(\langle w, x_i\rangle + b) - 1 + \zeta_i) - \sum_{i=1}^{N}\nu_i\zeta_i \quad (4.30)$$

Minimizing the Lagrangian with respect to the primal variables w, b and ζ and maximizing with respect to the dual variables $\alpha \geq 0$ and $\nu \geq 0$ is equivalent to minimizing the primal 4.29. We can find the stationary point by setting the partial derivatives to zero:

$$\begin{aligned} & \tfrac{\partial}{\partial w} L(w,b,\zeta,\alpha,\nu) &&= 0 \\ \Leftrightarrow \quad & w - \textstyle\sum_{i=1}^{N} \alpha_i y_i x_i &&= 0 \\ \Leftrightarrow \quad & \textstyle\sum_{i=1}^{N} \alpha_i y_i x_i &&= w \end{aligned} \tag{4.31}$$

$$\begin{aligned} & \tfrac{\partial}{\partial b} L(w,b,\zeta,\alpha,\nu) &&= 0 \\ \Leftrightarrow \quad & \textstyle\sum_{i=1}^{N} \alpha_i y_i &&= 0 \end{aligned} \tag{4.32}$$

$$\begin{aligned} & \tfrac{\partial}{\partial \zeta_i} L(w,b,\zeta,\alpha,\nu) &&= 0 \\ \Leftrightarrow \quad & \alpha_i + \nu_i &&= \frac{C}{N} \end{aligned} \tag{4.33}$$

Substituting 4.31 into 4.30 we obtain:

$$\begin{aligned} L(w,b,\zeta,\alpha,\nu) \quad &= \frac{1}{2}\langle \sum_{i=1}^{N} \alpha_i y_i x_i, \sum_{j=1}^{N} \alpha_j y_j x_j \rangle - \sum_{i=1}^{N} \alpha_i (y_i \langle \sum_{j=1}^{N} \alpha_j y_j x_j, x_i \rangle) \\ &+ \sum_{i=1}^{N} \alpha_i - b \cdot \sum_{i=1}^{N} \alpha_i y_i + \frac{C}{N} \sum_{i=1}^{N} \zeta_i - \sum_{i=1}^{N} \alpha_i \zeta_i - \sum_{i=1}^{N} \nu_i \zeta_i \\ &= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ &+ \sum_{i=1}^{N} \alpha_i \underbrace{-b \cdot \sum_{i=1}^{N} \alpha_i y_i}_{=0 \text{ (using 4.32)}} + \underbrace{\frac{C}{N} \sum_{i=1}^{N} \zeta_i - \sum_{i=1}^{N} (\alpha_i + \nu_i) \zeta_i}_{=0 \text{ (using 4.33)}} \\ &= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_{i=1}^{N} \alpha_i \end{aligned} \tag{4.34}$$

Using 4.33 and $\nu_i > 0$ we get constraints $0 \leq \alpha_i \leq \frac{C}{N}$ and obtain the dual optimization:

$$\begin{aligned} \min_{\alpha} L(\alpha) \quad &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ s.t. \quad & 0 \leq \alpha_i \leq \frac{C}{N} \end{aligned} \tag{4.35}$$

Those observations with associated multipliers $\alpha_i > 0$ are called *support vectors*.

4.4.2. Kernels

To solve 4.35 the training examples never have to be considered individually, but only in terms of inner products of pairs of examples. Therefore we can rewrite 4.35 using *kernel function* $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$.

$$\begin{aligned} \min_{\alpha} L(\alpha) \quad &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ s.t. \quad & 0 \leq \alpha_i \leq \frac{C}{N} \end{aligned} \tag{4.36}$$

With $\phi(x_i) = x_i$ this is equivalent to the original problem, but using the kernelized version we gain additional flexibility and under certain circumstances drastic reductions in runtime and memory consumption. First, if the training examples are not embedded in a dot product space we can choose a *feature mapping function* $\phi : X \to \mathcal{F}$ mapping the original examples into the dot product space $\mathcal{F}$. Second, since no explicit representation of $\phi(x_i)$ is needed the dimensionality of the dot product space may be arbitrarily large. Note that in many cases the kernel function $K(x_i, x_j)$ can be computed significantly more efficient than $\langle \phi(x_i), \phi(x_j) \rangle$.

In the following, we will present several kernel functions that will be used later in this work.

String Kernels Problems arising in many areas in computational biology are classification tasks on strings. Using suitable kernels we can train an SVM also on data that does not come from a dot product space, like the set of strings on a given alphabet. In this section, we will introduce kernels defined on strings.

Weighted Degree Kernel

Consider two strings s_i and s_j of length L over an alphabet Σ. The weighted degree (WD) kernel of order K counts all exact k-mer ($1 \leq k \leq K$) matches of two strings s_i, s_j. The number of exact k-mer matches is multiplied by a specific weight and added up to derive the result of the kernel function as follows:

$$k(s_i, s_j) = \sum_{k=1}^{K} \beta_k \sum_{l=1}^{L-k+1} I(u_{k,l}(s_i) = u_{k,l}(s_j))$$

Where $u_{k,l}(s)$ denotes the k-mer starting at position l of string s. The indicator function $I(.)$ equals 1 if its argument is true and 0 otherwise. The weights β_k were

chosen to be [106]:

$$\beta_k = \frac{2(K-k+1)}{K(K+1)}$$

The weights decrease with increasing k. This is counter intuitive, but it can be seen by considering that a match of length k encloses two matches of length $k-1$, which also contribute to the final sum. The feature space of the WD-kernel has

$$d_K = \sum_{k=1}^{K}(L-k)\cdot|\Sigma|^k$$

dimensions, where L is the length of the sequences. This illustrates the impossibility of an explicit calculation of the mapping into the feature space for even moderate values of K[4]. The WD kernel is used to detect positional arrangements of sequence patterns around genomic signal sites.

Spectrum Kernel

The spectrum kernel detects co-occurring k-mers in the pair of sequences. The kernel function is given by

$$k(s_i, s_j) = \sum_{k=1}^{K}\beta_k \sum_{\sigma\in\Sigma^k} |\{l|u_{k,l}(s_i)=\sigma\}|\cdot|\{l|u_{k,l}(s_j)=\sigma\}|,$$

i.e. the product of the number of occurrences of all substrings in each of the sequence [107]. Positional information is not taken into account.

Weighted Degree Kernel with Shifts

The WD kernel with shifts (WDS kernel) is an extension of the WD kernel such that not only matches on equivalent positions are considered, but also matches that are shifted to a certain degree. The kernel function is given by

$$k(s_i, s_j) = \sum_{k=1}^{K}\beta_k \sum_{l=1}^{L-k+1}\sum_{s=0,s+l<L}^{S(l)} \delta_s(I(u_{k,l+s}(s_i)=u_{k,l}(s_j)) + I(u_{k,l}(s_i)=u_{k,l+s}(s_j))).$$

Every position l of the string can be assigned with a maximal shift $S(l)$. β and $I(.)$ are defined as for the WD kernel. The shift is penalized through the parameter $\delta_s > 0$, such that a perfect match, that is shifted, will not score as high as a non-shifted match. This kernel is computational more expensive than the WD kernel, but it is still feasible for up to ten million training sequences of reasonable length [108].

For numerical reasons, the WD as well as the WDS-kernel are normalized accord-

[4] with $L = 100$ and $K \geq 18$ storage of a single example explicitly will exceed main memory capacities on a *large* compute server

ing to the following formula:

$$K(x, x') = \frac{k(x, x')}{\sqrt{k(x, x)k(x', x')}}$$

Kernel Combinations *Let K_1 and K_2 be kernels with respect to the mapping functions ϕ_1 and ϕ_2. Then K_c with*

$$K_c(x_i, x_j) = K_2(x_i, x_j) + K_1(x_i, x_j)$$

is also a kernel since we can define the mapping function ϕ_c as

$$\phi_c = \left\langle \begin{pmatrix} \phi_1(x_i) \\ \phi_2(x_i) \end{pmatrix}, \begin{pmatrix} \phi_1(x_j) \\ \phi_2(x_j) \end{pmatrix} \right\rangle.$$

A function K_t with

$$K_t(x_i, x_j) = t \cdot K_1(x_i, x_j) \tag{4.37}$$

is also a Kernel since the corresponding mapping function ϕ_t can be defined as

$$\phi_t(x) = t \cdot \phi_1(x), \tag{4.38}$$

mapping into the same feature space as ϕ_1.
Applying 4.37 and 4.38 iteratively one can define new kernels as weighted sums of known kernels.

5. RNA-Seq-based gene finding (mGene.ngs)

Before the invention of high throughput expression measurements, direct evidence for the existence of transcripts originated mainly from Sanger sequencing of cDNA libraries. This resulted in long sequence fragments (ESTs) and full-length cDNAs that gave relatively accurate information on the structure of transcripts. However, the limited throughput of the Sanger sequencing technology resulted in a very low fraction of transcripts for which direct evidence was available. Gene finding systems were employed to predict the structure of transcripts without direct evidence, being trained on labels generated e.g. from ESTs.

The establishment of high throughput expression measurements (first based on microarrays, then using high throughput sequencing platforms) put us into a situation where expression measurements were available for a majority of genes in a genome. Due to biases, high error rates, and the extensive dynamic range of transcript expression levels the inference of transcript structures purely based on expression measurements has proven to be challenging [e.g. 8, 136]. The motivation to develop *mGene.ngs* was to overcome this limitation by integrating information from the genomic DNA sequence with expression measurements. This entails the challenge to weight the different, potentially contradictory sources of information based on varying levels of confidence for each information source.

In order to learn the trade-off between the different information sources from the data we extended the gene finding system *mGene* to take additional sources of information into account. In the first part of this chapter we will describe the methodical background of gene finding systems in general and then the important details of *mGene* in Section 5.3. We then introduce the changes and extensions we implemented to take additional information sources into account in Section 5.4. In Section 5.5 we present the results of *mGene.ngs* in an international competition, for the reannotion of 19 *Arabidopsis thaliana* genomes and we assess the performance of *mGene.ngs* to recover known genes of model organisms without using the genome annotation for training. We compare our results to two well-known RNA-Seq-based approaches.

5.1. Contributions

The gene finding system *mGene* uses discriminative classification as well as structured output learning methods to identify protein coding and noncoding genes based on a training set and a large variety of information sources. Components of this system have been developed by several people, which we want to acknowledge here to the best of our knowledge.

The learning algorithm of *mGene* was first applied to this problem in *mSplicer* [92] to predict the splice structure of a transcript with known start and end sites. This model was then extended to the full gene finding system *mGene* [100]. Signal predictors were mainly developed by Sören Sonnenburg, Alexander Zien, Petra Philips, Gabriele Schweikert, Jonas Behr and Gunnar Rätsch [107, 108]. Integration of signal predictions into a common cross validation framework was done by Gabriele Schweikert and Jonas Behr. Extension of *mSplicer* to account for all additional signals was done by Gabriele Schweikert and is described in detail in [99]. Speedup of *mGene* to be applicable to mammalian genomes was done by Jonas Behr, Gunnar Rätsch and Peter Niermann [84]. Extensions to account for expression measurements and additional biological measurements resulting in *mGene.ngs* were done by Jonas Behr.

5.2. Background

Protein coding as well as noncoding genes consist of a small set of distinct types of segments that are organized according to an underlying grammar. The grammar arises from biological mechanisms acting on the genomic DNA as well as the RNA sequence. The general mechanisms are known (see Chapter 2) and this knowledge can be used to construct models enforcing the grammatical rules of genes. We frequently refer to the different types of genomic segments as content types (e.g. "exon", "intron", "5'UTR", "intergenic region", ...).

The aim of gene finding systems is to segment a genomic DNA sequence into these distinct segments according to the predefined grammar. This is an instance of *label sequence learning* (LSL). One of the most well-known algorithms to solve LSL tasks is called *Hidden Markov Models* (HMMs).

In the following, we formally introduce HMMs and then discuss benefits, limitations and several extensions and enhancements.

5.2.1. Markov Models

Markov Models (MMs) are probabilistic models for distributions of sequences on finite alphabets. We assume an unknown distribution Λ of sequences defined on

alphabet Σ. Given a set of sequences S drawn independently from Λ, then a MM of order n models Λ by estimating the probability of observing symbol $\sigma_i \in \Sigma$ succeeding a substring l of length n. The model can be represented as a directed graph $G = (V, E)$, where the set of nodes V corresponds to all possible sequences of length n. Edges $e \in E$ exist between node v_1 and v_2 if and only if $s(v_1)_{2,n}$ is equal to $s(v_2)_{1,n-1}$, where $s(v)$ denotes the sequence associated with node v and $s_{j,k}$ denotes the substring starting at position j and ending at position k.

The length of sequences is modeled by including explicit start and end states in the set of nodes V. Each node has an incoming edge from the start state and an outgoing edge to the end state. Each edge e has an associated weight $\theta_e \in \mathbb{R}_+$. Weights of outgoing edges of each node are required to sum to one.

Maximum likelihood estimates for parameter θ_e, frequently termed *transition probability* associated with edge e from node v_1 to v_2 can be computed from the training set as follows:

$$\theta_e = \frac{N(s(v_1)_{2,n}s(v_2)_{n,n})}{\sum_{i=1}^{|\Sigma|} N(s(v_1)_{2,n}\sigma_i)}$$

$N(s)$ is defined as the number of occurrences of substring s in the training set. Transition weights from start states and to end states can be computed accordingly. For better generalization capabilities it is advisable to assume a uniform prior on transition probabilities and compute the posterior using *pseudo count* $k > 0$.

$$\theta_e = \frac{N(s(v_1)_{2,n}s(v_2)_{n,n}) + k}{\sum_{i=1}^{|\Sigma|} N(s(v_1)_{2,n}\sigma_i) + k}$$

Inhomogeneous MMs are a generalization of MMs where the transition probabilities are not independent from the position in the sequence.

Sequence Classification with Markov Models The posterior probability of sequence s for given parameters θ of a MM can be computed as

$$P(s|\theta) = \prod_{e=\Xi(s)} \theta_e,$$

where $\Xi(s)$ denotes the sequence of edges needed to generate s from the model.

Training two MMs on different classes we can obtain a decision function for classification:

$$D(s) = sign\left(log\left(\frac{P(s|\theta^+)}{P(s|\theta^-)}\right)\right).$$

5.2.2. Hidden Markov Models

Hidden Markov Models (HMMs) are a generalization of MMs, where the one-to-one mapping of graph nodes and symbols in the alphabet may be violated. An emission probability $e_v(\sigma)$ for symbol σ is associated with node v. Thus, using HMMs we distinguish sequences of states (label sequences) from sequences of emitted symbols (observation sequences). The transition probabilities p_{v_1,v_2} is independent of the sequence of symbosl, when conditioned on the previous state v_1. With HMMs one can model segmentations of sequences where each segment consists of the same set of symbols (e.g. $\Sigma = \{A, C, G, T\}$), but originates from different source distributions. Since many paths through the graph $G = (N, E)$ of an HMM can produce the same sequence of observations, the true path that generated a given sequence cannot be directly inferred from the observation sequence. In the following, we present the *Viterbi* algorithm to decode the path that has most likely produced the observed sequence under a given model parametrization.

5.2.3. The Viterbi Algorithm

The Viterbi algorithm [38] solves the inference problem to find the path $\pi_1, ..., \pi_{|s|}$ maximizing the likelihood $P(s|\pi, \theta)$ for any observation sequence s. The Viterbi algorithm is based on a dynamic program defined on a matrix V with $|s|$ columns and $|N|$ rows. The entries of V are given recursively by

$$V(i,j) = \begin{cases} e_i(s_j) \cdot max_{k \in N}(V(k, j-1) \cdot p_{k,i}) & ,\forall\ 2 \leq j \leq |s| \\ e_i(s_1) & ,else \end{cases}$$

Using a matrix of trace-back pointers T

$$T(i,j) = \underset{k \in N}{argmax}\, V(k, j-1) \cdot p_{k,i}, \forall 2 \leq j \leq |s|$$

one can infer π.

$$\pi_j = \begin{cases} T(\pi_{j+1}, j+1) & ,\forall\ 1 \leq j < |s| \\ argmax_{k \in N}\, V(k,j) & ,else \end{cases}$$

5.2.4. Application of HMMs to Gene Finding

As discussed in the beginning of this section, biological constraints define a grammar on genes encoded by a genome. This grammar is common knowledge and conserved to a very large degree among eukaryotes. HMMs are well suited to model sequence segmentations incorporating strict grammatical rules originating from bi-

ological prior knowledge. Therefore, HMMs are a widely used formalism to solve LSL problems in computational biology and are the basis of many well-known gene finders (e.g. GenScan [18] and Augustus [111][1]).
However, though the commonly used maximum likelihood estimation of parameters is computationally very efficient, it has been shown to be a major reason for the limited performance commonly observed in HMM based gene finding systems [134]. A second limiting factor is the assumption of HMMs stating that an observed symbol is independent of the state sequence, when conditioned on the state that generated the symbol. This assumption limits the set of features that can be used to model certain genomic signals. The method we present in the Section 5.2.5 addresses these shortcomings of HMMs.

5.2.5. Hidden Markov Support Vector Machines

Altun et al. [3] proposed a method called Hidden Markov-SVMs (HM-SVMs) combining the above mentioned advantages of HMMs with a discriminative training procedure. Let X and Y be the set of possible observation sequences and the set of label sequences, respectively. To simplify the notation we assume all observation and label sequences having the same length l. Given a training set of tuples $(x_i, y_i), i \in 1, ..., m$ with observation sequences $x_i \in X$ and label sequences $y_i \in Y$ the goal is to learn a discriminant function $F_\theta : X \times Y \to \mathbb{R}$ such that the label sequence y_i scores higher than all other sequences $y \in Y, y \neq y_i$ for each training example i. This can be formalized in terms of an optimization problem:

$$\min_{\theta \in \mathbb{R}^n, \xi \in \mathbb{R}^m_+} \quad \Omega(\theta) + C \sum_i \xi_i \tag{5.1}$$

$$\text{s. t.} \quad F_\theta(x_i, y_i) - F_\theta(x_i, y) > 1 - \xi_i \tag{5.2}$$

$$\forall \quad i = 1, \ldots, m, \quad y \in Y \bigwedge y \neq y_i \tag{5.3}$$

$\Omega(.)$ is a regularization function. A typical choice is the l_1 or l_2 norm.

To enable efficient decoding of the highest scoring label sequence via a dynamic program (using the Viterbi algorithm; Section 5.2.3), F is required to be linear in θ, i.e.

$$F_\theta(x, y) = \langle \theta, \phi(x, y) \rangle$$

Function $\phi : X \times Y \to \mathbb{R}^n$ maps tuples of observation and label sequences into the n-dimensional euclidean space. The choice of the *joint feature map* ϕ depends on the problem at hand. It is clearly crucial for the performance of the algorithm that ϕ encodes the features necessary to distinguish true from wrong examples. The slack

[1] Augustus provides an option to train parameters using a conditional random field

variables ξ allow misclassification of training examples. This problem is an instance of regularized empirical risk minimization approaches discussed in Section 4.3.2. Like in other instances of this class of problems the hyper-parameter C has to be determined using model selection.

Given the optimal value $\hat{\theta}$, predictions are done by maximizing F over Y, i.e.

$$f(x) = \underset{y \in Y}{\operatorname{argmax}} F_{\hat{\theta}}(x, y)$$

Due to the number of wrong label sequences, resulting in an exponential number of constraints, this problem cannot be solved directly. In the following, we introduce a method called column generation that can be applied to optimization problems, where the enumeration of inequality constraints is not feasible.

Column Generation

Column generation is a method for solving optimization problems, where the enumeration of inequality constraints is not feasible. This technique in generally guarantees optimality of the solution and converges in polynomial time under conditions discussed below. The main idea is to exploit the fact that at the optimal solution most of the constraints are inactive. Column generation is applicable if, for a given solution, we can guarantee to find violated constraints. The simple procedure iterates between (1) solving the original optimization problem with respect to a subset of constraints and (2) finding constraints violated by the current solution. (3) The currently violated constraints are appended to the subset of constraints. If step (2) does not return any violated constraints, then the current solution is the optimal solution also with respect to all constraints. If we can further guarantee that step (2) finds the constraint that is maximally violated by the current solution, then Tsochandaridis et al. [122] have shown a polynomial bound on the number of constraints that have to be considered until the algorithm convergences.

For the optimization problem 5.4, we can easily see, that the maximally violated constraint is given by the highest scoring label sequence $\tilde{y}$:

$$\tilde{y} = \underset{y \in Y}{\operatorname{argmax}} F_{\theta}(x, y)$$

As mentioned previously, $\tilde{y}$ can be efficiently computed using the Viterbi algorithm.

Margin Rescaling

The optimization problem 5.4 treats every wrong label sequence the same way. The constraints enforce a margin of size 1 if $f(x_i)$ is different from y_i and 0 otherwise.We

can equivalently formulate the optimization problem as:

$$\min_{\theta\in\mathbb{R}^n,\xi\in\mathbb{R}^m_+} \quad \Omega(\theta) + C\sum_i \xi_i \tag{5.4}$$

$$\text{s. t.} \quad F_\theta(x_i, y_i) - F_\theta(x_i, y_j) > \Delta(y_i, y_j) - \xi_i \tag{5.5}$$

$$\xi_i \geq 0 \tag{5.6}$$

$$\forall \quad i = 1, \ldots, m, \quad y_j \in Y \bigwedge y_j \neq y_i \tag{5.7}$$

using

$$\Delta(y_i, y) = \begin{cases} 0 & \text{if } y_i \text{ and } y \text{ are equal} \\ 1 & \text{else} \end{cases}$$

When encountering complicated output structures like in gene finding this choice of the Δ function might be too stringent. Each example we cannot predict exactly correct will incur the same loss, independent of how close the prediction to the label is. We can generalize the formulation using other choices for $\Delta(.,.)$ that reflect our intuition of how good a prediction is. We then enforce a larger margin if the prediction is very far from the label. This technique is called margin rescaling. The only property $\Delta(.,.)$ has to fulfill is that it has to be decomposable over the sequence, to allow for an efficient maximisation with dynamic programming. To find the maximal margin violator y^* for training example i we have to maximize:

$$y^* = \operatorname*{argmax}_y F_\theta(x_i, y) + \Delta(y_i, y)$$

One example for an appropriate Δ function would be the Hamming distance.[2] However, the choice of the distance measurement should be similar to the evaluation metric we intend to optimize.

Semi-Markov Extension

Hidden semi-Markov Models (HSMMs) are an extension to HMMs that is closely related to generalized HMMs [62]. The system is allowed to stay in one state for a non unit number of observations. The observation sequence generated during that time is not necessarily treated as a Markov chain.
Rätsch et al. [91] transferred the semi-Markov extension to HM-SVMs. This has large effects on the Viterbi decoding. In the original form one had to consider transitions between all pairs of states in every step. The set of states we denote by Υ. Since all steps were of length one the run time was given by $\mathcal{O}(|\Upsilon|^2 \cdot l(x_i))$.

[2]The Hamming distance of two sequences of the same length counts the number of positions where the sequences differ.

With the semi-Markov extension we need to consider all transitions from previous states of distance less than some maximal lookback parameter κ. Thus the overall run time of the semi-Markov Viterbi decoding is given by $\mathcal{O}(|\Upsilon^2| \cdot l(x_i) \cdot \kappa)$. The major advantage of this extension is that arbitrary features depending on the entire sequence generated from one state can be taken into account. For example it is possible to learn length distributions of individual states in a nonparametric way.

5.3. The Gene Finding System *mGene*

In this section, we present how the strategies introduced in Section 5.2 are realized in the two layers of *mGene*. We point out details of the implementation, whenever this does not trivially follow from the formal description. This is important for two reasons. First, the applicability of *mGene* to large genomes (like mammalian genomes) is a computational challenge and heavily depends on an efficient implementation. Several contributions of this thesis improving the runtime of training and prediction made it possible to apply *mGene* to mammalian genomes. Second, the flexibility to deal with a wide variety of information sources is due to the flexible design and attentive implementation. This allows us to take advantage of an information source to improve genome annotation even without prior information on whether and in which sense (enrichment or depletion in one or several genomic segment types) this information source is informative for transcript structure prediction.

5.3.1. Genomic Signal Prediction

Genomic signal predictions (foremost splice site predictions) are the most informative genome-based features. *mGene* performs genome-wide signal predictions in a separate step preceding the data integration and gene finding step itself. In this section, we describe the pipeline for genome-wide cross validated signal prediction. Implementation of the genomic signal prediction pipeline was part of a previous thesis of the author [7]. Details of label generation and modeling of the signals can be found in the Supplemental Section A.2.

Training

For the positive and negative example sets we cut out genomic DNA sequences from windows around the example site. For a given signal all example sequences are of the same length. Figure A.1 visualizes positive and negative example sequences for the acceptor splice site, by encoding each nucleotide with a different color. We performed a five fold cross validation to obtain test predictions for each consensus position in the genome. Each classifier undergoes model selection using training

and validation sets nested into the outer cross validation loop to tune the SVM regularization parameter.

Genome Wide Predictions

SVM predictions from different cross validation splits may vary significantly in scale. *mGene* computes empirical estimates of posterior probabilities[3] for each SVM and then fits a piecewise linear function to transform SVM predictions into posterior probabilities. This procedure is described in detail in [108] and recapitulated in Section A.2.2.

5.3.2. Sequence Based Content Prediction

For the different content types our model accounts for, we train sequence-based content predictors. Content predictors are used to classify the DNA sequence into the four classes *intergenic*, *UTR*, *exon* and *intron*. This is an instance of multi class classification. There are several possible solutions for multi class classification with SVMs and we decided to use an one-versus-rest training procedure.[4] This is motivated by the observation that during decoding we know for any transition between two states which content type we should see in between. Therefore, we need a score for a given sequence being of a given content type. As features we use counts of substrings. We train a linear SVM on feature vectors having one dimension for each possible sequence of length one to k. Similar to signal predictions we train content predictors in a five fold cross validation scheme with nested model selection to tune the SVM hyper-parameters.

In addition to these content predictors, we train one predictor to discriminate frame 0 versus other frames in coding regions. The feature vector consists of substring counts for substrings beginning at every third position. Coding regions of spliced mRNA sequences serve as positive training examples. Two sets of negative examples are generated by removing 1 or 2 leading nucleotides, respectively. In addition, we discriminate against all three frames of the reverse complement of the positive examples.

5.3.3. Transcript Structure Prediction

In this section, we introduce the second layer of the gene finding system *mGene*. We describe how the general methods presented in Section 5.2 are applied to solve the

[3] This is the probability of observing a positive example given the SVM prediction value.

[4] training four SVMs using one out of four classes as positive labels and the rest as negative labels

transcript structure prediction problem: Informally, we seek to find a discriminant function $F(x, y)$ such that for a majority of training examples x_i, $f(x_i)$ with

$$f(x_i) = \underset{y \in Y}{\operatorname{argmax}} F_\theta(x, y)$$

is equal to the true label sequence y_i.

Features

The usage of HM-SVMs allows us to construct features depending on a single position as well as features depending on the start and stop position of a segment. We denote the set of features depending on a single position by $\mathcal{S}$. Additionally to these signal features, there are three types of features associated with transitions: Length features $\mathcal{L}$, content features $\mathcal{C}$ and transition counters. Since there is no one-to-one mapping between the states Υ and the signal features $\mathcal{S}$ we introduce the function $\Xi_s : \Upsilon \to \mathcal{P}(\mathcal{S})$ returning the set of features associated with a given state. $\mathcal{P}(.)$ denotes the power set. In the same way we define $\Xi_c : \Upsilon \times \Upsilon \to \mathcal{P}(\mathcal{C})$ and $\Xi_l : \Upsilon \times \Upsilon \to \mathcal{L}$ for transitions. The set of allowed transitions we denote by $\mathcal{T} \subseteq \Upsilon \times \Upsilon$.

Joint Feature Map

The joint feature map Φ projects an observation-label sequence pair (x, y) into the parameter space $\mathbb{R}^n$. To give consideration to the semi-Markov property of our model, we switch notation of the label sequence equivalently from a nucleotide based to a segment-based one:

$$y = \{(\psi_1, \upsilon_1), ..., (\psi_s, \upsilon_s)\}$$

where ψ_k is the start of segment k, υ_k is the label of segment k and s is the number of segments. We define the joint feature map $\Phi(x, y)$ as a concatenation of four parts:

$$\Phi(x, y) = \begin{pmatrix} \left(\sum_{k=1}^{s-1} C_{c,k} \cdot \phi_c(x_{\psi_k ... \psi_{k+1}})\right)_{(\sigma,\rho)\in\mathcal{T}, c\in\Xi_c(\sigma,\rho)} \\ \left(\sum_{k=1}^{s} S_{s,k} \cdot \phi_s(x_{\psi_k \pm w})\right)_{\sigma\in\Upsilon, s\in\Xi_s(\sigma)} \\ \left(\sum_{k=1}^{s-1} L_{l,k} \cdot \phi_l(\psi_{k+1} - \psi_k)\right)_{(\sigma,\rho)\in\mathcal{T}, l=\Xi_l(\sigma,\rho)} \\ \left(\sum_{k=1}^{s-1} I(\upsilon_k = \sigma \wedge \upsilon_{k+1} = \rho)\right)_{(\sigma,\rho)\in\mathcal{T}} \end{pmatrix}$$

$C_{c,k}$, $S_{s,k}$ and $L_{l,l}$ are an indicator variable defined as

$$\begin{aligned} C_{c,k} &= I(\upsilon_k = \sigma \wedge \upsilon_{k+1} = \rho \wedge c \in \Xi_c(\sigma, \rho)) \\ S_{s,k} &= I(\upsilon_k = \sigma \wedge s \in \Xi_s(\sigma)) \\ L_{l,k} &= I(\upsilon_k = \sigma \wedge \upsilon_{k+1} = \rho \wedge l = \Xi_l(\sigma, \rho)) \end{aligned}$$

using the indicator function $I(.)$. $x_{\psi_k \ldots \psi_{k+1}}$ denotes the DNA-sequence between position ψ_k and ψ_{k+1}. Assuming that all signal features are only dependent on a symmetric window of size $2 \cdot w$, $x_{\psi \pm w}$ denotes the DNA-sequence $x_{\psi-w} \ldots x_{\psi+w}$.

ϕ is a set of functions mapping features f to $\mathbb{R}^b$. These functions are designed individually for the different features. The functions ϕ have the common form:

$$\phi_k(x) = \pi_k(\tau_k(x))$$

where $\tau_k : X \to \mathbb{R}$ is a feature specific function (e.g. a trained classification function) and $\pi_k : \mathbb{R} \to \mathbb{R}^b$. The constant b defines the number of support points of the piece wise linear feature transformation functions. b is defined in advance, typical values range between 10 and 100. Intuitively, π_k computes a smoothed histogram of a single value according to predefined bins[5] (limits) $l_1, ..., l_b$ with $l_1 < l_2 < ... < l_b$:

$$\pi_{k_i}(v) = \begin{cases} \frac{v - l_i}{l_{i+1} - l_i} & \text{if } l_1 < v < l_b \text{ and } l_i = max(\{l_j | l_j < v, 1 \leq j < b\}) \\ \frac{l_i - v}{l_i - l_{i-1}} & \text{if } l_1 < v < l_b \text{ and } l_i = min(\{l_j | l_j > v, 1 < j \leq b\}) \\ 1 & \text{if } v \geq l_b \text{ and } i = b \\ 1 & \text{if } v \leq l_1 \text{ and } i = 1 \\ 0 & \text{else} \end{cases}$$

Taking all the different feature types together, $\Phi(x, y)$ has the dimensionality

$$n = (|\mathcal{C}| + |\mathcal{S}| + |\mathcal{L}|) \cdot b + |\mathcal{T}|,$$

corresponding to the number of parameters we determine in training.

Piecewise Linear Functions

Given the training parameters (weights) $\theta_s \in \mathbb{R}^b$ corresponding to feature $s \in \{\mathcal{S}, \mathcal{C}, \mathcal{L}\}$ we define the piecewise linear function (PLiF):

$$\Psi_s(r) = \langle \theta_s, \pi_s(r) \rangle$$

[5] the bins are computed as percentiles of the feature distributions

This way we decompose the large scalar product of the discriminant function F to a sum of PLiFs over all features. PLiFs transform any real valued feature into a different range such that the feature is most useful in discriminating true from wrong paths. The parameters defining the PLiFs are learned during training in a globally optimal way.
The PLiFs also offer possibilities to interpret the training results. Uninformative features for example will be transformed to 0, whereas PLiFs of highly informative features will have large output ranges and therefore contribute more to the discriminant function F than less informative features.

Since all PLiF parameters are learned during training, *mGene* is robust to transformations of its input data and new features can be included without defining the weighting relative to the old features in advance. Choosing the number of bins b, PLiFs can approximate any kind of transformation as accurate as needed. But since b is a linear factor in the number of training parameters, large values for b increase the time and space requirements for training. However, using smoothness constraints[6] and monotonicity constraints[7] the learned transformations are appropriately regularized and transcript predictors with larger values for b are expected to generalize equally well (see Section 5.3.3 for details).

Dynamic Program

With the Dynamic Program we compute the label sequence maximizing the discriminant function F_θ for a given input sequence x.

The Viterbi matrix V has size $|x| \times |\Upsilon|$ and is recursively defined by:

$$\begin{aligned} V(\sigma,1) = \quad & 0, && \forall 1 \le i \le |\Upsilon| \\ V(\sigma,\psi_2) = \quad & max_{\rho,\psi_1} && V(\rho,\psi_1) + \sum_{s\in\Xi_s(\sigma)} \Psi_s(\tau_s(x_{\psi_2\pm w})) + \\ & && \sum_{c\in\Xi_c(\rho,\sigma)} \Psi_c(\tau_c(x_{\psi_1\ldots\psi_2})) + \theta_{trans_{\rho,\sigma}} \\ & s.t. && 1 \le \rho \le |\Upsilon| \\ & && max(\psi_2-\kappa,1) \le \psi_1 < \psi_2 \end{aligned}$$

Where $\theta_{trans_{\rho,\sigma}}$ is the parameter associated with transition count $\rho \to \sigma$ and κ is the maximal length allowed for a single segment. For positions ψ where we have no prediction for signal s, we define $\Psi_s(\tau_s(x_{\psi\pm w})) = -\infty$. For reasons of efficiency we use a sparse representation of V by removing columns with only $-\infty$ values. After

[6] we use an l_1 or l_2 penalty on the difference of parameters corresponding to neighboring bins
[7] If it is known in advance that higher values of a given feature should correspond to higher (lower) scores, we strictly enforce monotonic increasing (decreasing) PLiF transformations

computing V we decode the corresponding label sequence using trace back pointers. Computing τ for the signal features can be expensive since one needs to predict using SVMs with combinations of complex string kernels. However, signal predictions can be precomputed once genome-wide prior to the training of the HSM-SVM training.

Optimization Problem

We map the problem 5.4 to the following definition of a quadratic optimization problem:

$$\min_{z \in \mathbb{R}^{n+m+o}} \quad \frac{1}{2} z^T Q z + f^T z \tag{5.8}$$

$$\text{subject to} \quad Az \leq c$$

$$l \leq z \leq u \tag{5.9}$$

The vector z determined by the optimization problem is composed of three different parts:

$$z = \begin{pmatrix} \theta \in \mathbb{R}^n \\ \xi \in \mathbb{R}^m \\ S \in \mathbb{R}^o \end{pmatrix}$$

Where θ is the vector of model parameters, ξ is the vector of slacks for each of the m training examples and S is a vector of auxiliary variables enforcing a smooth shape of the Ψ-functions. o is given by $(|\mathcal{C}| + |\mathcal{S}| + |\mathcal{L}|) \cdot (b-1)$, i.e. for each pair of neighboring bins of the Ψ-functions we penalize the absolute difference of the corresponding model parameters. The constraint matrix A has $n+m+o$ columns. The number of rows of A increases during training. A encloses three different types of constraints. First, $2 \cdot o$ smoothness constraints enforcing smooth Ψ-functions:

$$\theta_i - \theta_{i+1} \leq S_i$$

$$\theta_{i+1} - \theta_i \leq S_i$$

We favour smooth transformations by incorporating $\gamma_s \sum_i S_i$ into the objective function. γ_s is a hyperparameter of the method and has to be determined using model selection. Second, A comprises a set of o constraints enforcing monotonicity of the Ψ-functions for monotonically increasing with $\theta_i \leq \theta_{i+1}$ and monotonically decreasing Ψ-functions using $\theta_i \geq \theta_{i+1}$. And third, we define a constraint for training example i to enforce that the score of the true label sequences is higher than the score of another label sequence y_j:

$$\langle \theta, \Phi(x_i, y_j) - \Phi(x_i, y_i) \rangle \leq \Delta(y_i, y_j) - \xi_i$$

To derive initial model parameters, we generate a set of constraints by constructing wrong label sequences from the true label sequence with a simple heuristic skipping some gene fragments.[8]

Q is a $(n+m+o)\times(n+m+o)$ diagonal matrix. Linear regularization parameters are determined by $f \in \mathbb{R}^{n+m+o}$. While the Ψ-parameters are regularized quadratically, we regularize the slacks linearly to promote sparsity.

5.3.4. Evaluation of Transcript Structures

Evaluation of gene predictions on real world data entails the difficulty of uncertainty in the labels. A large fraction of the genes in any genome annotation are predicted by a gene finding system and were never experimentally confirmed. To account for this, we assess gene prediction methods following an evaluation scheme that is similar to the strategy described in [23, 100]: If a transcript confirmation score for annotated transcripts is available, we assess the sensitivity on the confirmed transcripts only, while all annotated transcripts were used to determine the methods' specificity. Furthermore, we report the F-score as a single value performance estimate. Given the sensitivity SN and the specificity SP, the F-score is defined as $\frac{2*SN*SP}{SN+SP}$. We distinguish performance metrics on three different *levels*: nucleotide, exon and transcript level. Considering the evaluation of coding regions as a special case, the number of true positive predictions (TP) on nucleotide level is the number of predicted nucleotides that are part of an coding exon with respect to the prediction as well as the annotation. On exon level we require the entire predicted coding exon to exactly match an annotated coding exon and on transcript level we require an exact one to one relationship of all exons of a predicted coding transcripts to all coding exons of an annotated transcript. All predicted features, that are not counted as true positives are false positives (FP) and correspondingly we call all annotated features that are not matched false negatives (FN).

5.4. mGene.ngs Methods

In this section, we will first present related approaches for transcript prediction using expression measurements. We then present our contributions to the gene finding system *mGene* that allow us to (1) integrate various types of information sources, (2) train on examples generated from RNA-Seq data, (3) speedup computations to be applicable to mammalian genomes, and (4) predict noncoding genes in addition to protein-coding genes.

[8] Note that the heuristic only influences the starting point, but not the solution of the optimization problem, which is globally optimal in any case.

5.4.1. Related Work

In the following, we discuss tools for gene finding using RNA-Seq evidence as well as information from the genomic DNA sequence. Tools for transcript prediction only based on RNA-Seq data are discussed in Section 6.2. A detailed comparison of RNA-Seq based gene finding systems and transcriptome assembly tools has been the goal of the RGASP competition [116].

Fgenesh++R

Fgenesh [95] is an HMM based gene finding system. For participation in the rGASP competition it has been extended to take evidence from RNA-Seq read alignments into account. This version of *Fgenesh* is called *Fgenesh++R*. RNA-Seq information is considered during the prediction by forcing the prediction to include a previously defined set of compatible introns [54]. Alternative transcripts are found by iteratively predicting transcripts for a given gene locus using an intron not yet explained by previously predicted transcripts.

Augustus

Similar to *Fgenesh*, *Augustus* [110–113] (in its extended version *Augustus+* [114]) considers RNA-Seq information not during training but only during the prediction step of the model. The scoring function of the dynamic program is modified in the following way. For each transition of the dynamic program corresponding to an exon or intron *Augustus* checks whether there is confirmation from RNA-Seq alignments. If their is confirmation a predefined *bonus* value > 0 is added to the score, otherwise a predefined *malus* value > 0 is subtracted. Bonus and malus values do not depend on the amount of RNA-Seq confirmation, but the RNA-Seq data is filtered prior to the prediction such that lowly confirmed exons and introns are discarded from the RNA-Seq evidences being passed to *Augustus*. The bonus and malus values are determined using model selection for exon and intron evidences separately.

In contrast to *Fgenesh++R* *Augustus*' bonus/malus system does not force the prediction to comply with all hints. Therefore, contradictory hints may be used in a single prediction run. On the other hand, contradictory hints may indicate alternative splicing variants. In [115] multiple prediction runs are performed where only one out of a group of incompatible hints is considered at a time. Similar predictions are collapsed in a post-processing step.

Interestingly, *Augustus* proposes a second strategy for alternative transcript prediction based on sampling from the posterior distribution of transcript structures [48]. Instead of reporting only the transcript with the largest posterior probability, multiple predictions may be sampled with frequencies proportional to their poste-

rior probabilities. From those predictions feature specific (exon and intron) posterior probabilities may be estimated. Thresholds on the minimal and mean (geometric mean across the transcript) feature specific posterior probabilities may be used to control the sensitivity/specificity trade-off. This strategy allows *Augustus* to report multiple overlapping transcripts and still control for the specificity in a methodically sound way. However, not all information (in particular quantitative information) provided by RNA-Seq data is taken into account.

Discussion

To our knowledge *mGene.ngs* is the only gene finding system that learns the scoring of additional evidence during the training of the system. While in principle a model selection strategy like the one employed by *Augustus* can achieve similar results, this strategy is limited to few parameters and therefore cannot learn complex relationships between varying levels of confidences from different sources. Even when restricting the number of parameters to two for each additional source of information (like in the case of *Augustus*) it is computationally challenging to find optimal parameters for more than one additional information source. Moreover, the data processing for hint generation is critical for the performance and needs (at least once for each new data type) expert knowledge.

Fgenesh does not take exon coverage into account and therefore cannot benefit from read coverage e.g. in cases of single exon transcripts. On the other hand the idea, that each well confirmed intron should be explained by at least one transcript is reasonable and it is sensible to use a gene finding system to propose transcripts based on partial information. However, enforcing combinations of introns at the same time may lead to wrong predictions, which could have been avoided by iteratively enforcing the prediction of one not yet explained candidate intron per locus (which is nontrivial to define). The latter strategy might be beneficial if (1) combined with an appropriate filtering of predicted transcripts and (2) additional RNA-Seq-based features are taken into account during the prediction. Otherwise, ambiguities of parts of transcripts that are not enforced, but could be resolved using RNA-Seq data, result in suboptimal predictions. In this context, the probabilistic treatment of RNA-Seq data by *Augustus* seems more favourable.

The performance of *Augustus*, *Fgenesh++R* and *mGene.ngs* were compared among others in an international competition. We present the results in Section 5.5.1. We now present the strategies *mGene.ngs* employs to overcome the limitations discussed above.

5.4.2. Integration of Additional Evidence

An HMM can model an arbitrarily large sets of output sequences F by estimating emission probabilities $e_k^f(s)$ for state $k \in N$, symbol s in the alphabet $\Sigma(f)$ of output sequence $f \in F$. Each output sequence f is modeled to be dependent on the current state k only. Therefore, it is assumed that output sequences are conditionally independent from each other, given the current state. In practice this assumption can be violated, resulting in an inaccurate estimation of the posterior probabilities.

Discriminative approaches like Conditional Random Fields [64] and HM-SVMs do not make this assumption. Therefore, these models are well suited to integrate sources of evidence in addition to the genomic DNA sequence. In Section 5.4 we introduce the strategy to integrate various features based on the genomic sequence into HM-SVMs. We make use of the same strategy to integrate features based on other information sources like RNA-Seq measurements.

We distinguish between two types of evidences. Evidences associated with a single genomic position and evidences associated with a continuous segment on the genome. In the following section, we describe how the integration is is implemented using existing strategies of *mGene*.

5.4.3. Pice-Wise Linear Functions to Integrate Heterogeneous Sources of Evidence

Similar to features predicted based on the genomic DNA sequence we implemented generic interfaces to integrate features from other sources. For features f associated with a single genomic position we define piece-wise linear function and associate them with segment types like exons and intron. Such that for each transition (p_1, p_2) of type c we include $\sum_f \sum_{x \in X_{p_1,p_2}} \Psi_{c,f}(\tau_{c,f}(x))$ in the scoring function F, where X_{p_1,p_2} is the set of observations of type f between genomic position p_1 and p_2.

Features f associated with a pair of positions in the genome, e.g. evidence for introns from spliced reads, are also associated with a segment type c. For a transition (p_1, p_2) we add $\sum_f \Psi_{c,f}(\tau_{c,f}(x))$ to the scoring function with observation x being associated with p_i, p_2. Missing observations are set to zero.

5.4.4. Extension for noncoding Genes

One of the difficulties gene finding systems face when applied to RNA-Seq data is the enormous number of reads mapping outside of protein coding genes. These reads may be true biological measurements of noncoding genes or originate from false alignments to possibly non-functional duplications of coding genes. Based on RNA-Seq data one cannot decide between these different sources. Therefore, when only considering protein coding genes, the evidence from RNA-Seq data is very often

misleading. We solved this difficulty by extending the model of mGene described in [100] to also predict noncoding genes at the same time. Figure 5.1 shows how the new states for noncoding genes are integrated with the existing model for coding genes. We integrate the noncoding gene model as a *bypass* to the coding gene model, such that for each genic locus the prediction algorithm has to determine wether this locus is coding or noncoding. To avoid an excessive increase in the number of parameters that have to be trained the model for noncoding genes shares many parameters with the model for coding genes. All signal sites (TSS, TTS, acceptor and donor slice sites) share all features with the corresponding signal sites of the coding region. Moreover, the exonic transitions (TSS $\rightarrow$ donor and acceptor $\rightarrow$ TTS) share all features and parameters with the 5'-UTR exon transitions of coding genes. Introns in noncoding genes share the features and parameters of introns in the coding region. The only additional parameters that are learned specifically for noncoding genes are transition scores for each new transition. This way differences in the frequence of introns in noncoding genes can be captured.

Since it has been difficult to obtain reliable annotations for long noncoding intergenic transcripts, we train this model using a fraction of candidate transcripts from *Transcript Skimmer* (see Section 5.4.6) that had very short open reading frames as labels for noncoding genes and transcripts with long open reading frames as label for coding genes. We assume that a considerable fraction of the "noncoding" genes labels are in fact coding genes, where the transcript skimmer prediction failed to recover the correct transcript from the splicing graph. Therefore, we do not expect to obtain confident calls for noncoding genes, instead, we use the model for noncoding genes to absorb coverage that could otherwise only be explained by predicting a coding gene. We found that the predictions for protein coding genes become considerably more specific, when using the model for noncoding genes.

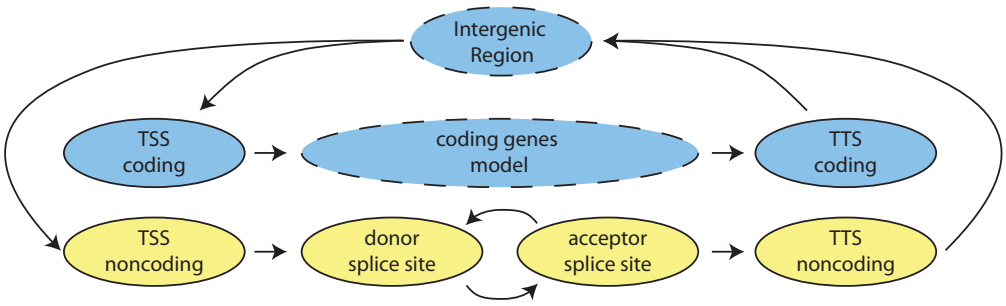


Figure 5.1.: Extension of the state-transition model for noncoding genes. Four new nodes (yellow) extend the existing model (in light blue; only scematic). States with dashed lines are conceptual and do not correspond to real states of the model.

5.4.5. Scaling up for Mammalian Genomes

Transcript predictions on mammalian genomes are computationally more challenging than for species like *C. elegans*, *D. melanogaster* and *A. thaliana* because the maximal segment sizes are one order of magnitude longer. Introns in the human genome are up to 200,000 nt long (20,000 nt for *C. elegans*) and therefore increase the runtime of the semi Markov Viterbi algorithm by a factor of 10 per nucleotide. With genome sizes also being a factor of 10 larger and a higher number of iterations needed for training the original model used for smaller genomes became infeasible. Genome wide transcript predictions for *C. elegans* require approximately 500 CPU hours. Extrapolating the runtime requirements for the same model to the human genome, we expect a runtime of approximately 38 CPU years. Using a lookup table for very long transitions we could achieve a speedup by a factor of 100, resulting in a runtime of 3300 CPU hours, which is feasible given the opportunity to distribute computations over many CPUs.

We extended the method of *mGene* by five strategies to speedup the training and prediction, which we first introduce individually and then provide an algorithm that combines all extensions for training and prediction.

Approximate Viterbi Prediction During the training of *mGene.ngs* the generation of constraints iterates with solving the quadratic program. Since the semi-Markov Viterbi algorithm is very costly, we implemented an approximation by restricting the positions considering in the Viterbi decoding to a relatively small subset. This subset includes only positions with high confidence values for any of the signal predictions and positions used in the label sequence. This reduction in the position list yields a 10 to 20-fold speedup of the Viterbi decoding, but may potentially exclude the highest scoring prediction. However, we can still guarantee optimality if we find that the final parameter set satisfies the constraints 5.2 defined on the full Viterbi algorithm.

In each iteration we flag a training example for computing the full Viterbi with a given probability ρ or if the training example with approximate decoding did not generate a constraint. The latter is the case if the true path scores highest among all possible paths. As soon as a given fraction of examples are flagged for exact decoding, we perform a single round of exact decoding for all examples. If once in the training process the number of new constraints drops below a given threshold, we turn off the approximation for all iterations to come. This is motivated by the observation that towards the end of the training the approximate decoding does not generate constraints that will be active at the final solution and also produces very few constraints and therefore does not advance the training process significantly. This strategy is detailed in Algorithm 1. The same convergence guarantee of the

original training algorithm clearly also holds for this strategy, because we only add easy to compute negative examples to an otherwise unchanged procedure.

Parallel Constraint Generation We may reduce the number of times we need to solve the optimization problem by generating a large batch of constraints with the same parameter vector θ_t in the t-th column generation iteration. This allows us to distribute the viterbi prediction for parallel computation at the cost that some of the generated constraints might not be informative. The parallel constraint generation is mentioned here for completeness. It is part of the original *mGene* publication [100] and has not been developed by the author of this thesis.

```
function C, A = generate_constraints(X, Y, A, θ)
// X = {x_i|i ∈ {1, ..., m}} training examples
// Y = {y_i|i ∈ {1, ..., m}} training labels
// A = {a_i|i ∈ {1, ..., m}} approximation flags
// θ current parameters
{
  // determine if approximation is appropriate
  τ = decide(mean(A))
  // compute Viterbi decoding in parallel threads
  for all i=1, ..., m do
    // precompute segment features
    f_i = precompute_features(x_i, θ)
    // Viterbi decoding
    ŷ_i = Viterbi(x_i, f_i, τ)
    // Update approximation flag
    if rand ≤ ρ or ŷ_i == y_i then
      a_i = 0
    end if
  end for
  Ŷ = {ŷ_i|i ∈ {1, ..., m}}
  C = compute_constraints(X, Y, Ŷ);
  return C, A
}
```

Algorithm 1: Generation of constraints. Function $decide : \mathbb{R} \rightarrow \mathcal{B}$ is implemented as a simple thresholding function (we use threshold 0.5) and ρ is in our case fixed to 0.05.

Reduction of Constraints During the training the number of constraints constantly increases. We have noted that most of the constraints generated early in the training process are not active at later stages and at the final solution. However, the runtime of the quadratic program solver increases significantly, the more constraints are accumulated. To speed up the solver we compute the margin of each constraint, by which it is inactive given the previoius solution. We discard temporarily exclude constraints which are inactive by a margin larger than ϵ. This includes by definition all new constraints. After solving the QP on this subset of constraints, we compute the fraction of all constraints that are violated. If this fraction exceeds a given threshold $1 - \rho, \rho > 0$ we iterate with an increased ϵ.

Algorithm 2 illustrates this strategy. While in the beginning of the training we frequently need to increase ϵ, typically considering up to 20% of the constraints, we find that towards the end of the training the solution obtained on 1% of the most

active constraints satisfies all constraints. Therefore, this simple extension saves a significant amount of computing time, while still guaranteeing optimality.

```
function θ_t = solve_QP(qp, θ_{t-1})
// qp: structure with fields: Q, f, A, c, l, u (cf. 5.8)
// θ_{t-1}: solution of previous iteration, empty if t == 0
{
  if is_empty(θ_{t-1}) then
    return solve(qp)
  end if
  while true do

    // select ε percentile of most active constraints
    qp_tmp = select_constraints(qp, θ_{t-1}, ε)

    // solve qp with standard QP solver
    θ_t = solve(qp_tmp)

    // compute fraction of satisfied constraints
    η = frac_satisfied(qp, θ_t)

    if η > 1 − ρ then
      return θ_t
    end if
    increase(ε)
  end while
}
```

Algorithm 2: Solving the quadratic program with reduced constraint lists

Lookup Table This strategy has been partially implemented by Peter Niermann and described in his Diplom thesis [84]. We change the model by defining a set of transitions for which the maximal length is computationally infeasible. We set the maximal length for these transitions to a significantly smaller value, but allow longer transitions using a lookup table. For each state t being a potential start state of one such transition we store the best so far seen score and the corresponding position p. In each step of the Viterbi algorithm where we computed the score for being in state t we compare this score to the value in the table (initialized with $-\infty$) and replace the value in the table if the current score is larger. Whenever we compute the score for a state which is a potential end state of a transition starting at t, we also compute the score we would obtain using a transition from p. While the change to the model is not equivalent, the implementation guarantees to find the best path given this model.

Pre-computing Segment Features The transformed scores of each feature f associated with genomic positions can precomputed to speedup the Viterbi algorithm. For each content type c and each position p we store

$$T(f,p) = \sum_{x \in X_{1,p}} \Psi_{c,f}(\tau_{c,f}(x)).$$

During the Viterbi algorithm we then use

$$\sum_{x \in X_{p_1,p_2}} \Psi_{c,f}(\tau_{c,f}(x)) = T(f,p_2) - T(f,p1).$$

Split Chromosomes for Prediction The memory consumption of the Viterbi decoding grows linear with the length of the sequence for predictions. For mammals this often exceeds the available amount of main memory. Therefore, we computed an extension that allows us to split large chromosomes into overlapping chunks and perform predictions independently. We choose the overlap large enough such that it is unlikely that any single gene exceeds the length of the overlapping region. Given two chunks, we then combine predictions by discarding possibly truncated genes in the overlap and replacing them by genes from the other trunk accordingly. This strategy does not only reduce the memory consumption, but also allows us to distribute the genome-wide prediction more effectively on a compute cluster.

Combining the Extensions In Algorithm 3, we show how the extensions are combined to speedup the training of *mGene.ngs*. The speedup for the Viterbi decoding using a lookup table for long transitions reduces the runtime during training as well as for genome-wide predictions. This is not mentioned explicitly in the following algorithm.

```
function θ = train(X, Y)
// X = {x_i | i ∈ {1, ..., m}} training examples
// Y = {y_i | i ∈ {1, ..., m}} training labels
{
  // according to Equation 5.4
  qp_0 = initialize_QP()

  // initialize approximation settings
  A_0 = {1}^m

  for t = 1, 2, ... do
    // Solve using Algorithm 2
    θ_t = solve_QP(qp_{t-1})
    // Generate constraints using Algorithm 1
    [C, A_t] = generate_constraints(X, Y, A_{t-1}, θ_t)
    // Termination
    if |C| == 0 and mean(A_{t-1}) ≥ 0.5 then
      break
    end if
    if |C| == 0 and mean(A_{t-1}) < 0.5 then
      A_t := {0}^m
    end if
    // Add constraints
    qp_t = add_constraints(qp_{t-1}, C)
  end for
}
```

Algorithm 3: Training of *mGene.ngs*

5.4.6. Label Generation based on RNA-Seq Data

mGene is based on supervised learning techniques, where statistical properties of known transcripts are observed to make predictions of undiscovered genes. The core system therefore requires a training set of annotated genes. To generate such a training set from RNA-Seq reads, we developed a simple *transcript skimming* technique that constructs preliminary RNA transcripts from read alignments. We first apply a filter to remove reads containing a high number of mismatches, very short exons, or very long introns, which are indicative of false alignments. Using a simple greedy approach we then select consecutive exons and introns until the coverage drops below a given threshold.

The algorithm starts with searching for regions in the genome that are covered above a certain threshold. To avoid reporting of incomplete transcripts gaps in

the coverage up to a certain length are tolerated in this step. Regions without spliced read mappings are reported as single exon transcripts, if the mean coverage is above a given threshold and there are no gaps in the coverage. For each remaining region we then select the intron having maximal coverage and extend the transcript into both directions. We continue the transcript by selecting the nearest intron, if this is covered above a certain threshold and if there is no nearby intron having (1) higher coverage and (2) being connected to our current intron by an potential exonic region without gaps in the coverage. We terminate transcripts if there is no nearby intron at the first position the coverage drops below a threshold. To avoid reporting incomplete transcripts, we discard transcripts as soon as the coverage drops below a threshold, but there is coverage nearby in the current search direction.

We then detect the longest open reading frame and mark transcripts as being likely coding transcripts if the fraction of the transcript, which is part of the ORF exceeds a given threshold. Therefore, in this step, only transcripts with high coverage that pass very stringent quality criteria are considered. This results in a small set of transcripts at comparably high specificity that is sufficiently large for training subsequent steps of the system.

5.5. Results

5.5.1. The rGASP Competition

In 2009 The Wellcome Trust Sanger Institute in Hinxton, Great Britain and the Spanish Center for Genomic Regulation in Barcelona, Spain launched the RNA-Seq Genome Annotation Assessment Project (rGASP). The goal of the competition was to assess and further the current state-of-the-art in genome annotation with RNA-Seq data. Participants were allowed to use any external source of information for their predictions, including genome annotations for the three organisms. Predictions were evaluated against the ENCODE genome annotation.

Data Sets There were two phases of the rGASP transcript recognition competition. In phase 1 the organizers provided a total of 21 RNA-Seq data sets for *C. elegans*, *D. melanogaster* and *H. sapiens*. In phase 2 additional 3 data sets were provided. An overview of the data sets can be found in Table A.1.

Strategy We generated spliced read alignments using *PALMAPPER* [55] using splice site predictions trained on the genome annotation and Illumina quality scores as features. We discarded all alignments but the one with the best alignment score. We filtered the alignments such that there were at least 5 nucleotides of the aligned read on each side of an intron and each read had at most 1 mismatch.

We trained *mGene.ngs* using the genomic DNA sequence and one RNA-Seq data set for each organism as additional evidence. From the RNA-Seq data set we extracted one track with read coverage, one track with RNA-Seq intron evidence and lists of RNA-Seq confirmed introns. As training examples, a set of annotated genes with read support was used.

In addition to the single transcript per locus that was discovered by *mGene.ngs* we predicted alternative transcripts using a splicing graph based approach. We complemented the prediction for each locus with intron evidence from RNA-Seq data. Moreover, potential exon skips were inferred where they substantially increased the length of the open reading frame. From the splice graphs we generated all paths if the number of paths did not exceed 100. Otherwise we randomly sampled 100 paths. The resulting candidate transcripts were quantified with *rQuant* [14] and filtered according to their predicted expression level.

Finally, an SVM based filtering approach was applied. Transcript predictions which appeared implausible due to short transcript length, low read coverage, or short open reading frames were filtered out using SVMs with RBF kernels that were trained on annotated genes.

For each organism, we used *mGene.ngs* system trained on one of the data sets to predict all data sets individually.

Results The predictions have been evaluated by the organizers and results are available from `http://www.nature.com/nmeth/journal/v10/n12/extref/nmeth.2714-S1.pdf`. Figure 5.2 shows scatter plots for the sensitivity and specificity of submissions[9] on coding exon and coding transcript level. Results for phase 1 of the competition are not yet available from the rGASP organizers. Therefore, all results we show are on the phase 2 data sets.

Discussion The evaluation results provided by the organisers suggest, that *mGene.ngs* provides accurate predictions for *C. elegans* and *D. melanogaster*. Predictions for *H. sapiens* are comparable in specificity to predictions of Stanke et al., but significantly less sensitive. We discuss potential reasons for this in Section 5.5.3. One significant inaccuracy of the evaluation is that the training set for the different systems was not defined by the organizers. Each method used a different set of annotated genes for training, potentially varying significantly in size. Therefore, the evaluation for each method is a mixture of training and test error, with unknown shares. Any comparison between the different methods has to be treated with caution and is only valid if we assume that each participant used a training set size much smaller than the whole genome. Our own predictions were trained on 3000 genes for each organism

[9] We discarded the submissions of Jie et al., which was only available for *H. sapiens* and were neither in terms of sensitivity nor specificity among the top performers.

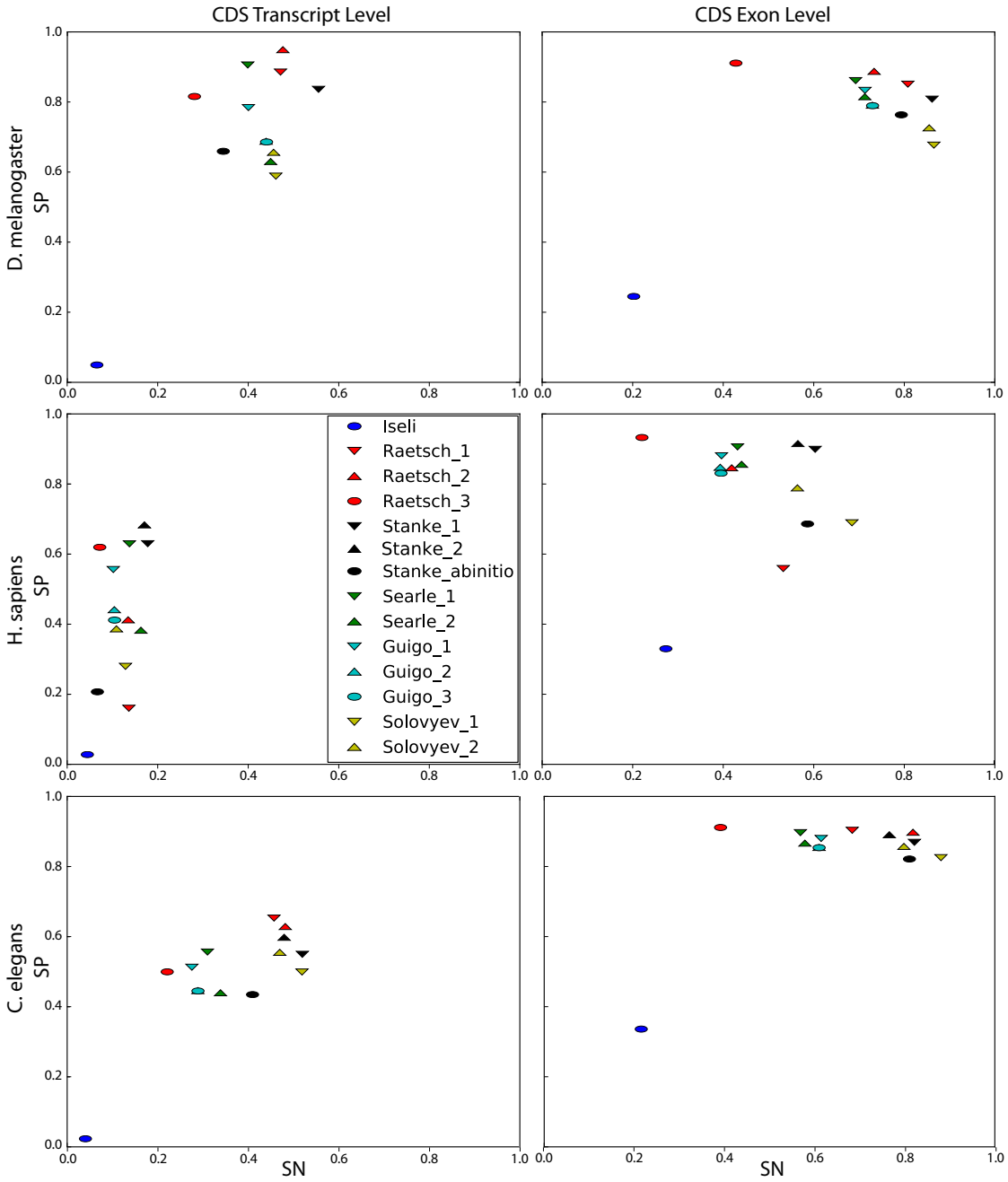


Figure 5.2.: Sensitivity versus specificity of submissions to the rGASP competition for organisms *D. melanogaster*, *H. sapiens* and *C. elegans*. The evaluation has been performed by the organizers. Submissions of Jie et al. are not shown to keep the Figure uncluttered. Submissions *Raetsch_1* and *Raetsch_2* were performed using *mGene.ngs* with different choices to transcript filters. For submission *Raetsch_3* *mTim* was used.

and predictions from Stanke et al. were trained on even fewer genes (Katharina Hoff, personal communication). Exon and transcript level sensitivity and specificity

for the submissions with *mGene.ngs* ("Raetsch_1" and "Raetsch_2") are among the best performing submissions for *C. elegans* and *D. melanogaster*.

5.5.2. Multiple Reference Genomes and Transcriptomes for *Arabidopsis thaliana*

In this section, we describe a collaborative study aiming to characterize sequence variation in *A. thaliana* and associate the sequence variation with phenotypic variation within a set of 19 natural *A. thaliana* accessions (strains). The project is divided into two phases. In the first phase genomes of 18 strains have been sequenced and assembled. Furthermore, transcriptomes were sequenced for all 18 strains plus the reference strain *Col-0* and genomes were (re-)annotated using computational methods we describe in the following. Main results of the first phase are published in Gan et al. [39]. The second phase is still ongoing. For this, recombinant lines with multiple parents (MAGIC lines) were created from the 19 original strains in four generations. These lines were then selfed for six generations to obtain a high degree of homozygosity. The resulting *mosaic* genomes were genotyped and transcriptomes for more than 700 lines were sequenced. Preliminary results show that this data set allows for cis and trans association of gene expression levels (eQTLs) with very high resolution.

The analysis of the phase two of the project depends on correct genome sequence and genome annotations for the 19 founder strains of the MAGIC lines. Therefore, the major goal of phase one was to assemble the genomes and characterize the transcriptome of the founder strains as accurate as possible using all available information. In the following, we describe the strategies for genome assembly and annotation we applied in phase one of the project.

Genome Assembly

The genome assembly has been carried out by Xiangchao Gan and colleagues at the laboratory of Richard Mott at Oxford University. The genomic DNA sequence of all 18 strains was sequenced with Illumina paired-end reads, with two libraries with 200-bp and 400-bp inserts and reads of 36 and 51 bp, respectively. The coverage varied per strain between 27-fold and 60-fold. The goal of the genome assembly was to obtain full length chromosome sequences with an assembly quality sufficient for a sensible re-annotation using gene finding systems.

In a first step reads were aligned to the TAIR reference genome using STAMPY [74] to detect a set of high confidence sequence variations. These variations were then integrated into the genome to generate a new genomic sequence. This procedure was iterated 5 times. Finally, de novo assemblies were generated using *SOAP denovo*

[70] and aligned to the TAIR reference genome to generate a list of variations. This approach called DENOM generates variant lists that potentially include complex variants being difficult to detect by the iterative approach. Results of the iterative approach were integrated with a rule based strategy.

Quality assessment of the resulting pseudo chromosome sequences using capillary sequencing and previously determined SNP calls revealed a high quality of the assemblies with less than one assembly error per gene [39]. We conclude that this quality is sufficient to sensibly attempt the *de novo* genome annotation for the 18 genomes using *mGene*.

An important side product of the reference guided assembly strategy is a mapping between coordinates of the TAIR reference sequence and the strain genomes. We used this mapping to project the TAIR genome annotation and gene predictions between the *Col-0* reference sequence and the accessions' genomes.

Projection of the TAIR Annotation to the Strain Genomes

We projected the TAIR10 genome annotation to the accessions' genomes and assessed the changes to the gene models. We consider a gene structure being disrupted if the consensus sequence for a splice site was changed, the consensus for translation start and termination sites was changed, a frame shift was detected or an additional in-frame stop codon (premature stop) was introduced. Table 5.1 gives an overview of the number of disruptions for the different accessions.

Altogether, out of 27,206 protein-coding genes from *Col-0* a naïve projection to the accessions' genomes predicts a severe disruption of 32% of the genes in at least one accession. This observation raises the question of how many disrupted gene structures are still functional with potential minor modifications. To address this question we re-annotated all genomes using computational methods to find gene models in the affected accessions that could potentially replace the disrupted *Col-0* gene models. We use the similarity on amino acid level as an indicator for similar function.

Genome Annotation

Parts of the following description are adopted from the Supplemental Information from [39].[10]

We used a variety of different strategies to annotate the 18 newly assembled genomes. We then evaluated these strategies based on two different sources of information. Firstly, we measured the consistency with the projected TAIR10 genome annotation, and secondly we generated an additional RNA-Seq data set for accession

[10] available from `http://www.nature.com/nature/journal/v477/n7365/extref/nature10414-s1.pdf`

Table 5.1.: Numbers of large-effect disruptions observed in annotated genes (TAIR10) on the strains' genomes. We count the total number of disruptions of translation start and stop sites, introductions of premature stop codons, splice site disruptions (separately for UTR and CDS splice sites and for acceptor (acc) and donor (don) splice site). We also count how often more than one disruption type occurs (column "multiple disruptions") and additionally report the number of disruptions when no other type of disruption was determined. In the second last column we report the number of genes per accession with more than 50% in deletions or polymorphic regions. The last column shows the number of disrupted miRNA gene stem sequences. (Adapted from [39] Supplementary Table 13)

Accession	translation start consensus disruption (single/mult)	translation stop consensus disruption (single/mult)	premature stop introduced (single/mult)	frame shift introduced (single/mult)	Splice site consensus disruption UTR [Acc (sngl/mlt) Don (sngl/mlt)]	Splice site consensus disruption CDS [Acc (sngl/mlt) Don (sngl/mlt)]	multi disruptions	>50% CDS in deletion or PRs (union 1,675)	miRNA disr.
Bur-0	307/766	173/487	3552/4285	2182/2794	186/187 142/143	245/251 172/176	770	780	11
Can-0	323/896	157/515	5054/6161	2587/3308	214/215 142/142	284/286 197/198	907	726	18
Ct-1	280/697	169/428	3094/3805	2026/2559	173/175 103/105	220/226 142/143	666	768	13
Edi-0	263/675	146/426	3410/4292	2230/2765	175/175 126/126	236/237 145/146	676	768	12
Hi-0	276/632	137/361	2644/3372	2105/2567	161/166 115/115	218/220 156/158	568	832	14
Kn-0	254/688	161/467	3683/4304	2354/2962	174/174 116/117	250/253 161/162	724	766	15
Ler-0	264/716	155/467	4022/4858	2324/2907	204/204 138/139	274/276 157/160	744	774	10
Mt-0	256/614	131/402	2928/3685	2132/2646	171/171 115/116	223/226 140/143	624	765	15
No-0	296/741	150/436	3704/4441	2139/2704	198/200 123/124	230/234 158/158	716	778	11
Oy-0	263/685	162/435	2512/3102	2064/2607	157/159 109/109	214/216 132/135	679	759	10
Po-0	329/729	161/402	2621/3373	2217/2732	168/170 107/107	233/235 142/146	629	827	14
Rsch-4	257/694	139/413	3455/4341	2117/2668	192/192 125/125	222/224 145/146	688	794	9
Sf-2	305/759	172/494	3886/5063	2390/2984	219/219 150/150	246/249 151/154	766	765	14
Tsu-0	249/677	152/464	4135/5298	2124/2685	180/181 115/115	214/215 141/143	716	776	17
Wil-2	263/771	155/497	3804/4748	2222/2892	178/178 131/132	244/246 187/189	830	762	15
Ws-0	294/788	149/467	3715/4711	2316/2958	221/221 138/139	240/245 156/158	798	770	13
Wu-0	269/727	166/424	3109/3879	2127/2706	178/179 126/126	234/235 133/137	705	776	11
Zu-0	286/765	148/410	3352/4139	2217/2793	200/201 129/130	235/237 156/159	716	776	13

Can-0. This data set was a) significantly deeper, b) from three different tissues and c) used a different protocol for library preparation (strand specific). We therefore assume the technical noise of this RNA-Seq data set to be to a large degree independent from the RNA-Seq data sets used for annotation, which justifies the usage for evaluation. We discuss the different gene prediction approaches later in this section.

Finally, we consolidated (cf. Section 5.5.2) the best performing prediction with the TAIR10 genome annotation to remove technical noise from the gene prediction method. The goal of this rule based approach was to use the RNA-Seq evidence to decide on gene models, only accepting new predictions if either the TAIR10 gene model was disrupted in a given strain, or the newly predicted features has sufficient RNA-Seq support. With this strategy we intend to counteract the tendency of overestimating the variability between accessions which we expect when directly comparing the independent gene predictions of each accession.

Signal Predictions We trained *mGene* signal and content predictors (cf. Section A.2.1 and Section 5.3.2) based on the genome of the *Col-0* strain (version TAIR9) and the TAIR9 annotation. All predictions were performed in a five-fold cross validation scheme. We then predicted these features on the genome assemblies of all 18 strains. To avoid training/test overlaps we mapped the cross validation splits to the genomes and used a predictor which has not used the corresponding position in *Col-0* for training.

Gene Predictions We trained two different *mGene* gene predictors on 7,500 randomly selected TAIR9 genes. The first predictor (*ab initio*) was only based on genomic features while the second (*de novo*) used RNA-Seq features as additional evidence. The RNA-Seq data set for *Col-0* consisted of 12 million aligned reads. Both predictors were used to annotate the 18 strains and re-annotate *Col-0* and were then evaluated with respect to the TAIR9 annotation mapped to the respective genome. The *ab initio* predictor predicted 26,292 genes on average per strain with an average transcript level F-score of 58%. The *de novo* predictor predicted 24,681 genes on average per strain with an average transcript level F-score of 63%. We also performed predictions using *Cufflinks*[121] and inferred coding regions using the maximal open reading frame for each transcript. Ignoring transcripts with a maximal open reading frame of less than 300bp (likely non coding transcripts or mis-predictions) we obtained an F-score on transcript level of 37.5%. Table 5.2 shows the exon and coding transcript level for the different predictions on *Col-0*. The increase in prediction accuracy by 5% between the *ab initio* and the *de novo* mGene prediction has two main reasons. Firstly, RNA-Seq reads help to determine ambiguous cases and secondly the prediction is more biased to higher expression levels. Higher expressed genes are generally detected with higher accuracy, even if no expression evidence is used for the prediction. See Section 5.5.3 for a discussion on the expression bias in gene prediction and evaluation.

Consolidation of Gene Predictions and the TAIR Annotation

In the following, we describe the pipeline consolidating the gene predictions with the TAIR10 annotation. This text appeared identically in [39] Supplementary Information Section 10.4. We first identified "units" of orthologous annotated genes across the accessions and TAIR10. We use the word unit rather than gene because it is possible that a TAIR10 gene might be split into several transcriptionally independent units (but no unit contains more than one TAIR10 gene). The coordinates of each accession's predicted transcripts were translated to TAIR10. All overlapping transcripts were treated as a unit, and duplicates (with identical exon/intron structure) were removed. The TAIR10 gene classification (protein-coding, transposable ele-

Table 5.2.: Comparison of three annotation strategies on the reference accession *Col-0*: a) *mGene*, which predicts protein-coding genes *ab initio* and uses the genome sequence only, b) *mGene.ngs*, which uses the genome sequence as well as RNA-Seq read alignments to predict protein-coding genes *de novo*, and c) *Cufflinks*, which only uses the RNA-Seq read alignments to predict transcripts to which we assigned open reading frames (ORF) of length at least 100nt and 300nt (the 590 and 2,884 transcripts, respectively, for which we could not identify an open reading frame were omitted from this analysis). Reported are the coding (CDS) exon and transcript level sensitivity (SN), specificity (SP) and F-score (F). (Adopted from [39] Supplementary Table 17)

	CDS Exon				CDS Transcript			
	# predicted	SN (%)	SP (%)	F (%)	# predicted	SN (%)	SP (%)	F (%)
mGene	146,241	84.8	85.5	85.2	26,649	56.5	63.0	59.6
mGene.ngs	136,391	84.3	91.2	87.6	25,077	57.6	75.5	65.2
Cufflinks	94,921	53.6	88.1	66.7	16,811	28.6	52.0	36.9
Cufflinks ORF≥100	90,176	52.5	90.7	66.5	14,517	27.5	58.8	37.5
Cufflinks ORF≥300								

ment gene, pseudogene, etc.) was passed onto the unit. We then back-transformed each unit to the original accession coordinates and restored lost accession-specific features (e.g., exons absent in TAIR10). We removed transcripts from a unit if splice site consensus sequences in an accession's genome were absent, or if they overlapped with more than one TAIR10 gene. We incorporated any TAIR10 annotated transcripts into their corresponding units and also mapped these back to each accession, removing invalid transcripts where a splice site consensus was missing in the accession. If the resulting unit contained at least one valid TAIR10 annotated transcript/isoform, it was merged with transcripts in the unit provided that all introns were confirmed by RNA-Seq alignment or were part of an annotated TAIR10 transcript, and the transcript contained at least one intron different from the annotated introns. If the unit did not contain a TAIR10 transcript then the predicted transcript with the highest number of confirmed introns was chosen. In rare cases there was no transcript in the unit (either they were all removed or the gene was missed in the *de novo* gene prediction). However, in most cases (99%) there was at least one consolidated transcript for each TAIR10 gene in each accession's genome. We noticed a number of remaining RNA-Seq-confirmed introns that were not part of any annotated transcript. With the aim to integrate the introns with highest confidence, we constructed a strictly filtered set of introns from spliced RNA-Seq read alignments. We required the intron to be confirmed by at least two reads with the split position at least 12nt from the read boundary and matching with at most 1 mismatch. For each such intron, we identified exons in transcripts with boundaries at most 50nt away from the intron boundary. For each case a new

transcript including this intron (or otherwise the previous transcript structure) was generated. The coding sequences of protein-coding transcripts were determined as follows: The longest region without stop codon in the mRNA was identified. We then checked whether the 5' and 3' ends of this region coincide with the transcript boundaries. If this condition was violated, we used the first in-frame occurrence of the translation initiation signal (TIS) consensus ATG as start and the first stop codon (TAA/TAG/TGA) as end of the coding sequence. If the 3' end of the region coincided with the transcript end, we extended the 3' end of the last exon by at most 300nt in order to terminate the open reading frame. This modification of the transcript was necessary, as the transcript ends were often predicted or annotated too short to include a suitable in-frame stop codon. If the 5' end of the region coincided with the transcript start, we checked the region 300nt upstream of the transcript for a suitable alternative TIS consensus signal and used it, if the length of the implied coding region increased by at least 100nt. For consistency, we also applied this strategy to the annotations of the reference accession Col-0. The statistics of the resulting consolidated accession-specific gene annotation are given in Table 5.3. Results for the additional strand specific data for *Col-0* and *Can-0* show that a significantly larger number of transcripts predicted by *mGene.ngs* can be confirmed using deeper sequencing. Thus, many of the predictions conservatively discarded, because there was not sufficient evidence from the smaller data set of non strand specific reads, are likely correct.

Analysis of Gene Variability between Strains

We performed a comprehensive comparison of amino acid sequences of all protein coding genes based on the consolidate gene sets. To relate the diversity between *A. thaliana* strains to an outside group we included annotated genes for *A. lyrata* in this comparison. Furthermore, we show the diversity of paralogous loci of *Col-0* genes. We measured the distance between two amino acid sequences s_1 and s_2 as $d(s_1, s_2) = 1 - \frac{m}{\max(l(s_1),l(s_2))}$, where m is the number of matches in a global sequence alignment and $l(.)$ returns the sequence length. For one pair of organisms a and b we perform an amino acid sequence alignment for each pair of protein coding transcripts. For each transcript in a we report the minimal distance to any transcript in b. We summarize distances for genes taking the minimum over all transcripts. Figure 5.3 shows the distribution of gene distances for different pairs of organisms. We computed the gene distances for all pairs of the 19 strains (Figure 5.3 black). Moreover, we computed for each gene in each strain the minimum and the maximum gene distance over all strains (Figure 5.3 yellow and Figure 5.3 red, respectively). Additionally, we computed the minimal distance of each *Col-0* gene to any other *Col-0* gene.

Table 5.3.: Features of the consolidated annotation of the 18 genomes based on RNA-Seq-based gene predictions and the TAIR10 reference annotation (excluding novel genes). The upper part describes the consolidated annotations used for most analyses (NSS). The lower part is based on additional, independent, strand-specific (SS) validation RNA-Seq data only available for *Col-0* and *Can-0*, leading to a larger number of predicted novel transcripts and introns.(Adopted from [39] Supplementary Table 18)

Acces-sion	Genes	Protein-coding transcript	Non-coding transcript	Novel transcript	Introns	Novel introns	TIS sites	Novel TIS sites	Stop codon sites	Novel stop codon sites	genes with modifications (union=8,757)
Col-0	33,295	41,303	1,395	1,687	129,368	1,143	33,710	323	34,204	351	1,604
Bur-0	32,842	40,526	1,368	2,152	127,344	1,262	33,201	406	33,635	405	1,977
Can-0	32,741	40,332	1,362	2,149	127,038	1,289	33,090	423	33,555	467	2,100
Ct-1	32,858	40,765	1,368	2,384	127,813	1,402	33,271	425	33,776	501	2,232
Edi-0	32,847	40,623	1,366	2,209	127,570	1,317	33,252	443	33,705	455	2,105
Hi-0	32,968	40,857	1,381	2,325	127,934	1,421	33,376	450	33,840	474	2,199
Kn-0	32,833	40,696	1,372	2,307	127,601	1,441	33,235	441	33,723	477	2,191
Ler-0	32,852	40,839	1,376	2,559	127,585	1,592	33,289	518	33,756	542	2,406
Mt-0	32,849	40,513	1,372	2,080	127,469	1,205	33,193	370	33,661	399	1,911
No-0	32,817	40,415	1,366	1,975	127,475	1,209	33,162	389	33,630	422	1,920
Oy-0	32,866	40,428	1,367	1,841	127,383	1,112	33,217	386	33,677	399	1,835
Po-0	32,987	40,648	1,377	2,021	127,605	1,251	33,326	376	33,817	413	1,945
Rsch-4	32,867	40,346	1,364	1,802	127,237	1,050	33,194	341	33,620	358	1,742
Sf-2	32,806	40,319	1,383	1,980	127,109	1,115	33,138	377	33,569	399	1,875
Tsu-0	32,832	40,506	1,372	2,030	127,443	1,176	33,211	410	33,658	407	1,941
Wil-2	32,776	40,363	1,369	1,987	127,197	1,208	33,109	383	33,584	439	1,948
Ws-0	32,844	40,217	1,365	1,727	127,097	1,030	33,160	355	33,586	357	1,725
Wu-0	32,879	40,545	1,377	2,041	127,605	1,249	33,241	410	33,670	392	1,984
Zu-0	32,883	40,745	1,383	2,300	127,699	1,456	33,303	482	33,770	489	2,217
Col-0 SS	33,295	43,546	1,402	4,420	131,341	3,101	33,656	269	34,066	213	2,959
Can-0 SS	32,741	42,617	1,369	4,937	129,045	3,276	33,107	437	33,477	386	3,566

We observe that for 99% of the genes in each strain there is at least one other strain, that has nearly exactly the same amino acid sequence (Figure 5.3 yellow line). On the other hand about 30% of all genes show a significant diversity of at least 10% amino acid changes in at least one other strain. Comparing to the amino acid distance of genes within *Col-0* (Figure 5.3 "Col-0 vs Col-0 (different locus)") we can observe that for more than 99% of all genes we find a better match in at least one different strain than to a paralogous[11] gene of the same strain. While in general our distance measurement cannot distinguish between orthologous[12] and paralogous genes this comparison shows that in most cases we correctly find the orthologous genes in the respective organisms. Less than 4% of the genes in *Col-0* have paralogous genes that are not distinguishable on amino acid sequence level.

Variability of Disrupted Genes We analyzed the changes in accessions' gene structures when the TAIR10 gene model was disrupted and frequently observed that disrupted splice sites were "rescued" by nearby splice sited, resulting in minor changes of amino acid sequences (e.g. Figure 5.4a). In order to assess if this switch of splice

[11]Paralogous genes are homologous genes separated by a gene duplication event.
[12]Orthologous genes are homologous genes separated by a speciation event.

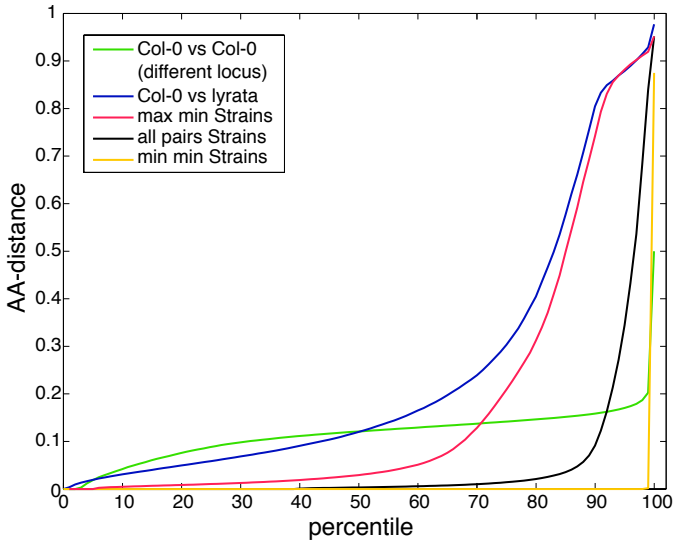


Figure 5.3: Variability of amino acid sequences. For one pair of organisms we computed all pairwise alignments of protein sequences of all protein coding genes. This plot shows the distribution of minimal pairwise distances for all genes. See text for details.

sites is a common evolutionary mechanism we were interested in the extend of similar events in the *A. thaliana* population. Based on the consolidated gene predictions we analyzed the variability of amino acid sequences of TAIR10 coding transcripts that were disrupted in at least one accession. For any predicted gene structure replacing a disrupted TAIR10 gene structure we computed the mean amino acid distance to all other accessions. Figure 5.5a shows the distribution of mean distances for different sets of disruptions. We observe that predicted replacement transcripts for splice site disruptions are on average more similar to the original transcript than any other type of disruptions. Performing a hierarchical clustering we identified groups of accessions having small within amino acid distance for a given gene. Figure 5.5b shows that for genes with splice site disruptions we find more often other accessions with the same or very similar allel than for other disruptions. This adds evidence that the proteins might still be functional. Using these measurement, the most severe types of coding region disruptions are premature stop codons and frame shifts.

Surprisingly, combinations of multiple disruptions are on average not more severely affecting the amino acid sequence than single disruptions. Therefore, we analyzed the amino acid similarity also for the most frequently occurring combinations of disruptions (Figure 5.6). We find that the majority of combinations of disruptions can be attributed to only two combinations of disruptions: 1) frame shift and translation terminations site disruptions and 2) frame shift and translation start site disruptions. In both cases the change in amino acid sequence is less sever than when observing a frame shift alone. This could potentially be explained by compensatory mechanisms, where a second mutation restores to a large extend the amino acid sequence after a frame shift occurred. In case 2) this could for example be the disruption of a translation start site consensus in favor of another in-frame translation start site downstream of the frame shift.

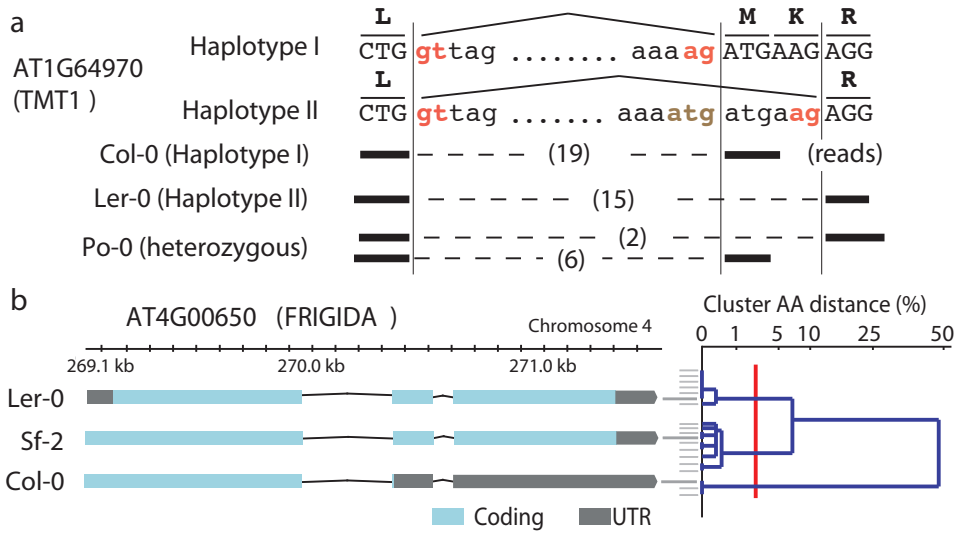


Figure 5.4.: **Transcript variation**. **a**, Example of a splice site change between two haplotypes for the gene AT1G64970. Haplotype I (Col-0) is spliced with an intron 6 bp (two amino acids) shorter than haplotype II (Ler-0); Po-0 (heterozygous) shows allele-specific expression of both. **b**, Re-annotation of the FRIGIDA locus showing annotations for accessions Sf-2 (functional), and Col-0 (truncated by a premature stop) and Ler-0 (non-functional) (Supplementary Figs 18 and 42). Right: the 19 accessions are shown clustered on the basis of the AA distance between their FRIGIDA amino-acid sequences. Common isoform clusters (at distance 2% or less; red line) are shown, leading to three clusters with three, seven and nine accessions. (Figure was created by the author, it appeared identically in [39])

GO-term Analysis

To further characterize the sets of genes that were mostly unchanged between strains and those with significant variability we analysed varying frequencies of functional annotations between the sets. As functional annotations we used the GO-annotation available form the TAIR website (`arabidopsis.org`).

We created three sets of protein coding genes based on amino acid sequence alignments between the *Col-0* gene model and the consolidated gene models in the strains.

- Gene set 1: *Recovered Genes*: This set consists of 940 genes that had at least one of the four signals acceptor splice site, donor splice site, translation start site or stop codon disrupted in at least one of the strains. But there was a gene prediction at the same locus that resulted in an amino acid sequence with more that 95% alignment identity to the TAIR annotation.

- Gene set 2: *Disrupted Genes*: This set consists of 312 genes that have at least one of the four signals disrupted in at least one strain and there was no gene

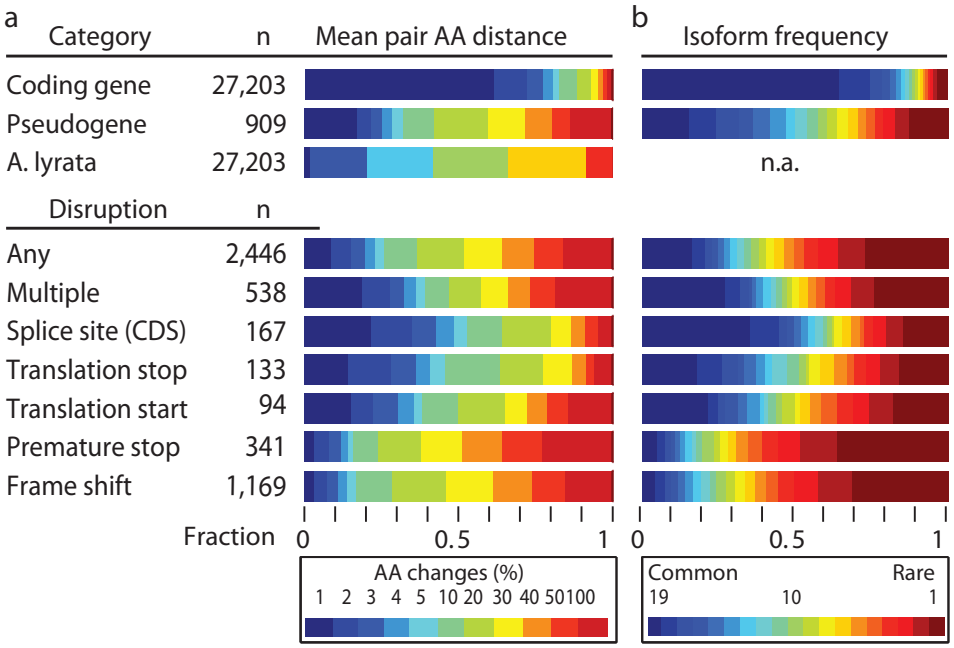


Figure 5.5.: **Protein sequence variation**. **a** Proteome diversity for coding genes, pseudogenes and A. lyrata genes (top) and for genes with disruptions (bottom). Reported is the fraction of genes with relative AA distance to other accessions (average over pairs) in the given colour-coded interval (Supplementary Information section 10.7). **b**, Frequency of isoforms of coding genes and pseudogenes (top), and those associated with different disruptions (bottom). (Figure was created by the author, it appeared identically in [39])

prediction in that strain, or the gene prediction had an amino acid similarity of less than 80%.

- Gene set 3: *Conserved Genes*: This gene set consists of 18768 genes that were never disrupted in any of the strains.

We then performed statistical tests for all GO-terms associated with TAIR10 genes and all pairs of the three sets defined above.

The null-hypothesis for the statistical test was that a given GO-term was randomly distributed between the gene sets with probability equal to the proportions of the gene set sizes (hypergeometric distribution). The statistical test was performed two sided detecting over and under represented GO-terms. Table 5.4 shows the GO-terms significantly enriched in the pairwise comparison of all sets.

Discussion In comparison to the set of disrupted genes we find genes associated with essential housekeeping related GO terms like *chloroplast/plastid*, *peroxisome*, *plasma membrane* and *transport* in both other gene sets. The GO-term which is

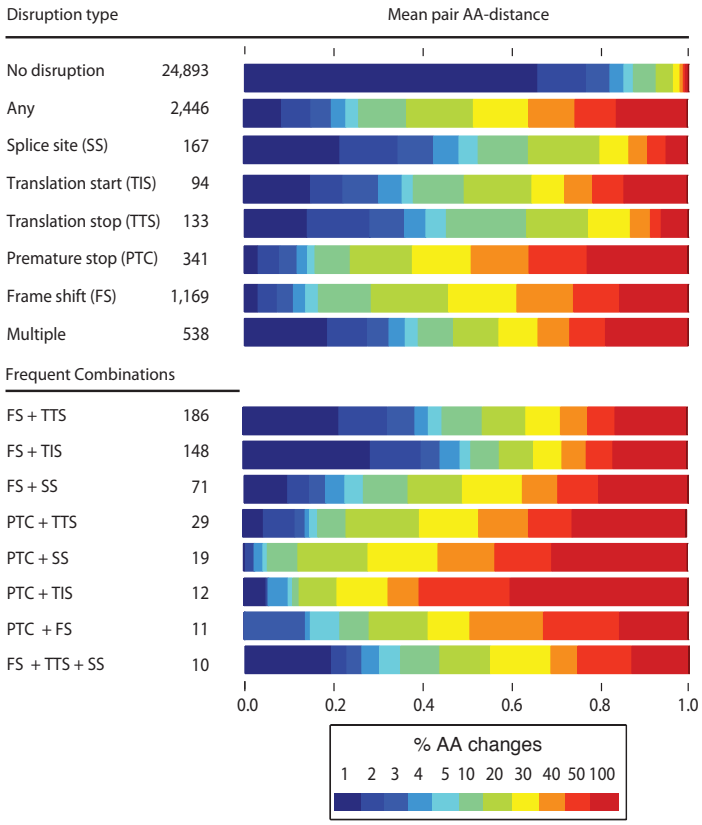


Figure 5.6.: **Influence of large effect disruptions on AA sequence difference**. Displayed is the fraction of genes with different degrees of AA sequence change between accessions with different combinations of disruptions and other accessions. We observe that splice site disruptions lead to least and FSs/PTCs lead to most severe AA-sequence changes. Among genes with multiple disruptions, combinations of FSs/PTCs with other disruptions, in particular TTS and TIS, are frequent. Surprisingly, the AA sequence differences for multiple disruptions are often smaller than one would expect, in particular smaller than the individual disruptions' sequence changes. (Figure was created by the author, it appeared identically in [39])

overrepresented in the set of disrupted genes with highest significance against both other gene sets is *transmembrane receptor activity*. Transmembrane receptors detect extracellular cues and transmit the detection signal to intracellular components. Reflecting the variety of extracellular ligands like phytohormones [81], metabolites and pathogen effectors [61] gene duplication is very common among transmembrane receptors, where the transduction mechanism is shared among members of the family while ligand binding domain and intracellular effector domain vary. Due to varying environments and exposure to different pathogens it is expected that a significant number of transmembrane receptors become unimportant in a given natural *A.*

thalina accession. The same argument holds true for GO terms *defense response* and *innate immune response* also significantly enriched in the set of disrupted genes. GO terms related to defense and pathogen response are also enriched in the group of *Recovered Genes* accompanied by the terms *chitin binding* and *chitinase activity* which are also associated with the defense against pathogens [88].

In the test between *Conserved Genes* and *Recovered Genes* we expect to identify genes where any change in amino acid sequence leads to significant reduction of the accessions fitness. This clearly holds true for genes being part of plastides playing crusial roles in photosynthesis and metabolism storage. Photosynthesis proteins are shared among green plants with extremely little sequence divergence and can be expected to be highly optimized such that any change in the amino acid chain will result in reduction of the plant fitness. The GO-terms *chloroplast* and *plastid* are among the most significantly enriched GO-terms in the *Conserved Genes* set. Interestingly, we find that *response to auxin stimulus* is enriched in the *Conserved Genes* set while *response to cytokinin stimulus* is enriched in the *Recovered Genes* set. *Auxin* is a phytohormone playing an essential role in embryonic development and growth [77]. Similarly, cytokinins are hormones regulating growth and development [60]. Cytokinin receptors are a family of proteins with high sequence similarity to bacterial histidine kinases. We hypothesize that cytokinin receptors are either partially redundant or specific to signaling of biotic stresses not present in all accessions environments. Therefore, there is freedom to evolutionary changes, while genes responding to auxin (which is just a single substance), are apparently under high stabilizing selection pressure.

5.5.3. De novo Gene Finding on various Organisms Using RNA-Seq Data

We investigate the ability of using *mGene.ngs* to provide genome annotations for newly sequenced genomes based on RNA-Seq data. Similar to solely RNA-Seq-based methods like *Cufflinks* and *Scripture* we restricted the source of information to the genomic sequence (used for alignments in the competing methods) and the RNA-Seq reads.

In order to obtain labels to train the gene finding system we devised a simple method called *Transcript Skimmer* (Section 5.4.6) for generating transcript structures from RNA-Seq data alone. We then trained the gene finding system on a subset of these labels and perform genome-wide predictions. This set of genes has a strong bias towards high expression. To increase sensitivity of low expressed genes we subsampled the amount of RNA-Seq for the training examples. We adjusted the subsampling strategy and label generation parameters for *C. elegans* and then used the same parameters for *A. thaliana* and *D. melanogaster*. This shows that the

Table 5.4.: Statistical GO-term enrichment analysis for all pairs of the three sets *Recovered Genes*, *Disrupted Genes* and *Conserved Genes* defined in the text. We report all GO-terms with a p-value smaller than 10^{-9}. The log_{10} of all p-values is given in the second column. (abbreviations: a: activity, m: membrane)

Recovered Genes vs. Disrupted Genes

Enriched in Disrupted Genes		**Enriched in Recovered Genes**	
transmembrane receptor a.	-42.46	plasma membrane	-21.40
defense response	-41.88	transport	-15.03
intrinsic to membrane	-38.02	transferase a.	-11.94
innate immune response	-28.31	peroxisome	-9.94
apoptosis	-17.93		
signal transduction	-17.02		
molecular function	-13.75		
serine-type endopeptidase inhibitor a.	-13.59		
cellular component	-12.97		
biological process	-12.33		
nucleoside-triphosphatase a.	-11.39		

Conserved Genes vs. Disrupted Genes

Enriched in Disrupted Genes		**Enriched in Conserved Genes**	
transmembrane receptor a.	-60.72	plasma membrane	-20.91
intrinsic to membrane	-59.02	chloroplast	-20.45
defense response	-58.65	transport	-16.12
innate immune response	-43.34	plastid	-10.43
cellular component	-34.01	cytoplasm	-9.94
apoptosis	-26.28		
serine-type endopeptidase inhibitor a.	-18.83		
nucleoside-triphosphatase a.	-18.50		
molecular function	-18.17		
biological process	-14.80		
signal transduction	-12.67		
specification of carpel identity	-9.14		
cytidine deamination	-9.00		

Conserved Genes vs. Recovered Genes

Enriched in Recovered Genes		**Enriched in Conserved Genes**	
peroxisome	-20.06	plastid	-13.58
chitin binding	-17.56	response to auxin stimulus	-10.89
homogalacturonan biosynthetic process	-15.38		
chitinase a.	-14.97		
glucosinolate catabolic process	-14.67		
response to cytokinin stimulus	-14.36		
microtubule motor a.	-14.34		
GDP-mannose 3,5-epimerase a.	-14.26		
small nucleolar ribonucleoprotein complex	-13.08		
trans-Golgi network transport vesicle m.	-12.75		
protein kinase a.	-12.11		
gibberellin 20-oxidase a.	-10.39		
glycolipid binding, transport	-10.39		

approach developed on *C. elegans* generalizes to a large range of organisms.

However, it might be beneficial to perform a separate model selection for these parameters for each organism and data set. In a scenario, where no annotation is available, the model selection can be performed based on similar evaluation measurements on protein sequence alignments from related organisms or on ESTs. The latter will prefer predictors with strong expression bias and therefore model selection based on protein alignments are preferable.

For each of the organisms we adjusted model parameters like the maximal intron length and for *M. musculus* we used the mammalian genome speedup described in Section 5.4.5. These parameters were fixed prior to the training of the gene finding system and were not adjusted in model selection.

In order to evaluate this approach we performed predictions on a variety of well annotated model organisms and compared the predicted structures to the annotation. In the following we present the results for four different model organisms from four different phyla (*C. elegans* kingdom *Animalia*, phylum *Nematoda*; *A. thaliana* kingdom *Plantae*, *Angiosperms* order *Brassicales*; *D. melanogaster* kingdom *Animalia*, phylum *Arthropoda*; *M. musculus* kingdom *Animalia*, phylum *Chordata*).

C. elegans Figure 5.7A shows the evaluation results of several transcript predictions compared to the *C. elegans* genome annotation (wormbase WS199). We split genes into ten equally sized expression bins using rQuant for quantification results. Similarly we split the annotation into bins. For each bin we computed the specificity using all annotated genes, but only the predicted genes in the corresponding expression bin and we computed the sensitivity using all predicted genes, but only the annotated genes in the corresponding bin. We can thus measure the performance of each method as a function of the expression level of genes.

We observe that for each method the performance generally increases with higher expression levels. This holds true even for predictions solely bases on the genomic sequence, ignoring expression data entirely. The reasons for this are manifold. First, higher expressed genes have generally more accurate annotations because annotation is often based on EST alignments and ESTs are more abundantly available for highly expressed genes. Secondly, very highly expressed genes tend to have fewer exons in *C. elegans* than medium to lowly expressed genes and are thus more easy to detect.

For the *de novo* predictions using RNA-Seq data (Figure 5.7 red) we observe a very strong expression bias. The performance for the lower 20% drops drastically, while the performance for the 70% highest expressed genes is favourable. We wanted to know to which extent this bias can be attributed to the expression bias in the *Transcript Skimmer* training set and to the level of RNA-Seq evidence. We therefore trained two gene finding systems 1) using annotated genes and RNA-Seq data for training (Figure 5.7 blue) and 2) using the *Transcript Skimmer* training set, but

ignoring the RNA-Seq data (Figure 5.7 cyan). Setting 1) achieves very good overall results and has a relatively weak expression bias, while setting two has a much stronger expression bias and overall relatively poor results, first and foremost when compared to the mGene system trained on the annotation without RNA-Seq data (Figure 5.7 green).

We conclude that a large part of the expression bias of the *de novo* prediction using RNA-Seq data can be attributed to the strong expression bias in the training set, while subsampling the RNA-Seq data, in order to fit a realistic genome-wide read coverage distribution also in the training set, could yield improvements in transcript level F-score of at most 20 percentage points for the 20% of the genes with lowest expression.

The *de novo* prediction using RNA-Seq data compare favourable to *Cufflinks* predictions. Moreover, we noticed that the performance of *Cufflinks* critically depends on the filtering of the RNA-Seq alignments. Unfiltered RNA-Seq alignments resulted in very poor performance (data not shown). We thus, hand tuned filter settings to improve *Cufflinks*' performance (Figure 5.7 yellow). Finally, we evaluated the accuracy of spliced RNA-Seq alignments using all annotated introns. We optimized the filter settings to maximize the F-score of candidate introns from RNA-Seq alignments and annotated introns. This resulted again in significantly better *Cufflinks* predictions. It is important to note that it is not possible to perform the latter strategy on unannotated organisms. We performed this analysis to exclude the possibility that the limited performance of *Cufflinks* is due to suboptimal RNA-Seq filter settings.

We investigated the set of genes the *ab initio mGene* system could not identify, when trained on only highly expressed genes, but identified correctly when trained on an unbiased selection from the entire genome annotation. We found that the largest fraction of these genes belong to the family of *G-protein coupled receptors*. This is a family of proteins which exhibits a highly conserved structure consisting of seven transmembrane α-helices. Members of this family of proteins detect extracellular cues and transduce the signal to intracellular signaling cascades. Only the relatively small intracellular and the extracellular domains vary, while the signal transduction mechanism is highly conserved between members of the family. The family members are known to be expressed at very low levels mainly in neuronal tissue [109], therefore none of the members of this large family appeared in the training set with strong expression bias. Figure 5.8 shows the coding transcript length distributions for the *C. elegans* genome annotation separate for transcripts with and without expression evidence. A large fraction of transcripts with exactly 1000 nt length originates from the family of *G-protein coupled receptors*.

We conclude that members of this gene family exhibit genomic features distinct from other gene families and can only be recognized if they are part of the training

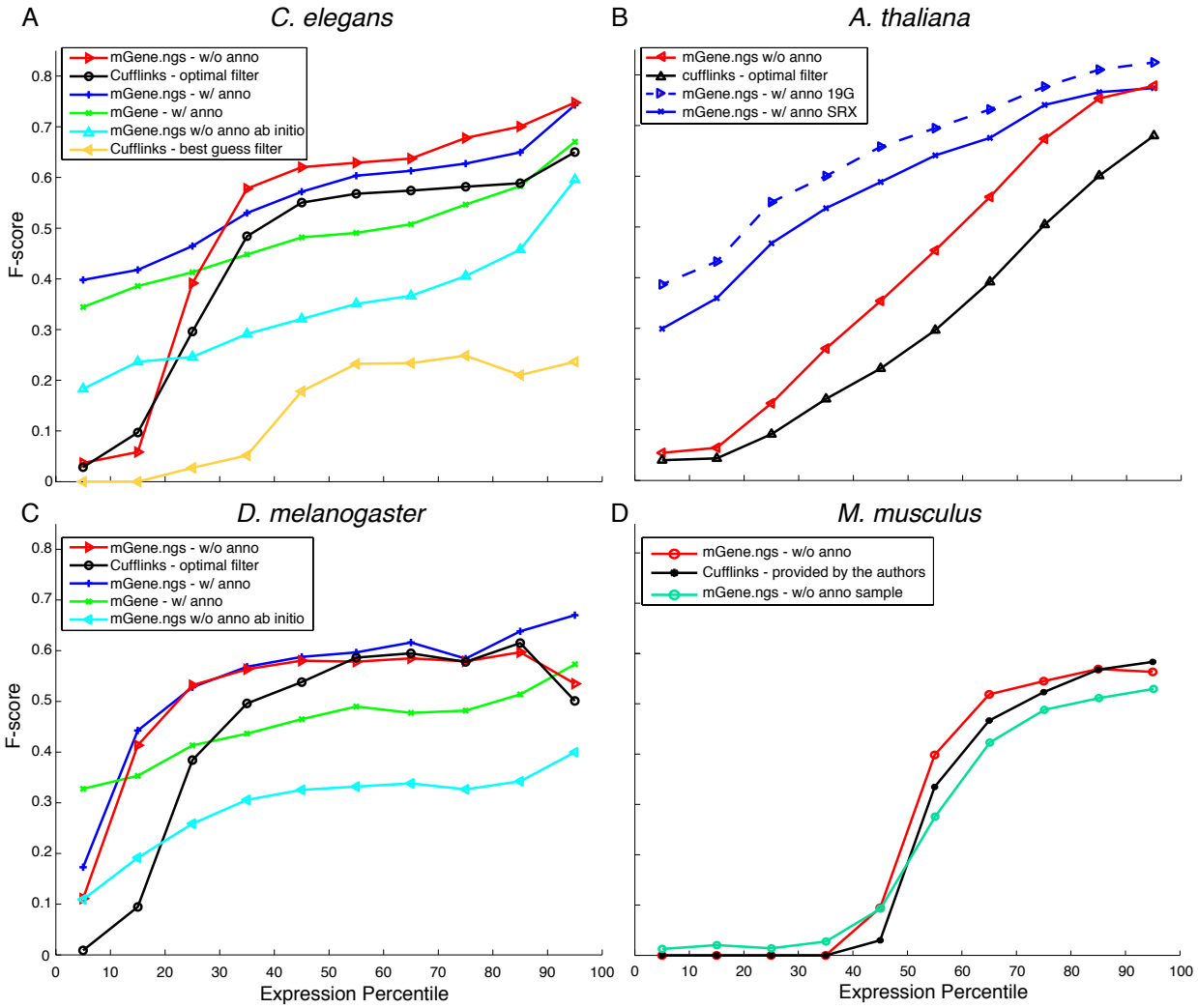


Figure 5.7.: Evaluation of gene predictions for the *C. elegans* genome. The evaluation is separated for genes from different equally sized expression level bins. Measured is the F-score on coding transcript level. See text for evaluation details.

set. We assume that similar effects may be observed for other protein families, but become less apparent because either, the family has at least a few members being sufficiently high expressed, or the family has significantly fewer members.

A. thaliana For *A. thaliana* we observe a strong expression bias for the *de novo* predictions as well as for *Cufflinks*. The *mGene.ngs* predictor using the annotation (blue solid line) performs significantly better. This can be likely attributed to relative low quality RNA-Seq data on one hand and rather simple gene structures on the other hand. The latter acts in favour of gene predictors trained on the annotation. All these predictions were generated based on seven lanes of RNA-Seq data (Short Read Archive accessions SRX006192, SRX006681, SRX006682, SRX006688,

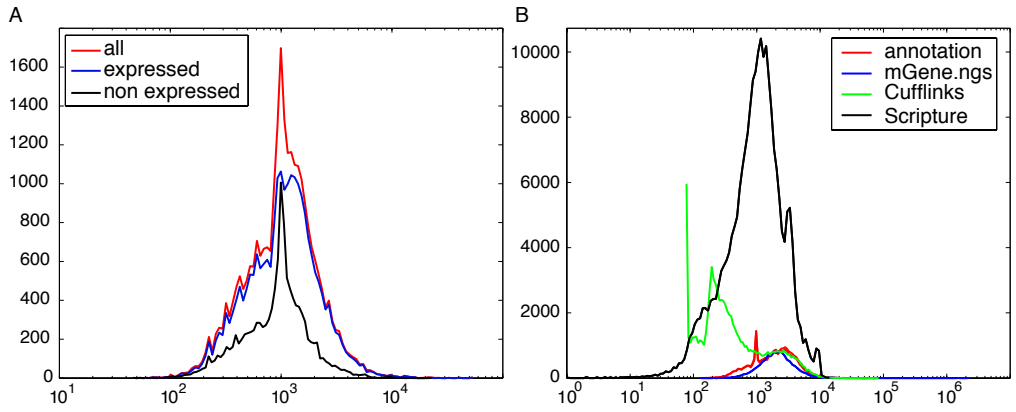


Figure 5.8.: **A** Length distribution of coding transcripts in the *C. elegans* WS199 genome annotation. **B** Length distribution of transcript predictions for *M. musculus* in comparison to the MM9 genome annotation.

SRX006690, SRX006692, SRX006704). To estimate the effect of the RNA-Seq data we also evaluated the predictions performed for the 19G project (Section 5.5.2). We observe significant improvements for the *mGene.ngs* predictor trained on the annotation when using more reliable RNA-Seq information (Figure 5.7B blue dashed line).

D. melanogaster Using the same setup we performed *de novo* gene predictions for *D. melanogaster*. We observe that the performance of the *de novo* prediction performs very well on the 90% highest expressed. Only for the lowest 10% the performance significantly drops. This is most likely attributed to a larger fraction of expressed genes. The performance of the *ab initio* predictions are similar to those of *C. elegans*, but especially the one trained on the transcript skimmer training set shows a less severe expression bias. Likely, the larger fraction of expressed genes results in a training set which is less biases to highly expressed genes.

The performance of *mGene.ngs* predictors trained on the genome annotation and on the *Transcript Skimmer* gene set are comparable. *Cufflinks* predictions for genes expressed above median expression perform very similar to the *mGene.ngs* prediction trained on the *Transcript Skimmer* gene set, but the latter approach significantly outperforms *Cufflinks* predictions of genes expressed below median.

M. musculus Improvements described in Section 5.4.5 allowed us to perform the same type of analysis on the mouse genome. We used RNA-Seq alignments provided by [121] and compared our results to the predictions also provided by [121]. Gene finding on mammalian genomes is significantly more challenging than on *C. elegans*, *D. melanogaster* and *A. thaliana* (see e.g. Section 5.5.1 and [42]). Therefore, the

contribution of the genomic sequence based features to the recognition of transcripts is relatively small compared to the other organisms. Moreover, the large number of exons of a gene and the considerably more complex splice graphs pose difficulties for the very simple label generation strategy, aiming to identify the path with highest coverage in the splicing graph. Therefore, improvements with respect to the *Cufflinks* predictions are moderate. The best performing *mGene.ngs* prediction and the *Cufflinks* prediction identify very few transcripts with expression level below median. When tuning *mGene.ngs* to rely more on genomic features by subsampling RNA-Seq evidence in the training set, we observe small improvements in the prediction of lowly expressed genes and the overall performance decreases. For *M. musculus* we readjusted the parameters for read subsampling and label generation. Therefore, the gene prediction has not been carried out independent of the genome annotation. We still think the comparison to *Cufflinks* predictions is fair for two reasons. First, the genome annotation was only used to tune the label generation, not for training itself. Therefore the flow of information from the annotation to the classifier is very limited and we can expect this to generalize to other mammal genomes. Second, the *Cufflinks* predictions were provided by the organizers as part of the original *Cufflinks* publication. Therefore, we can expect that the predictions are tuned to this data set. This became evident when we tried to regenerate the transcript predictions. Based on the same alignment files and the published parameters we obtained significantly weaker results. With nearly the same sensitivity (25.42% vs. 25.40%) the specificity on transcript level decreased from 44.30% to 37.46%. We contacted the authors, but were unable to resolve this issue.

Figure 5.8B shows the length distribution of predictions from *mGene.ngs*, *Cufflinks*, and *Scripture* in comparison to the mouse genome annotation. *Scripture* reports an extremely large number of transcripts resulting in an near zero transcript level F-score (not shown). The majority of Cufflinks predictions is very short and no open reading frame of at least 141nt could be found. Therefore, these transcripts are ignored in the evaluation. We note that inclusion of these transcripts in an evaluation scheme not relying on ORF detection would drastically lower the specificity of cufflinks predictions in comparison to the annotation. However, knowing that noncoding transcripts are highly underrepresented in the annotation we are limited to evaluate the coding portion of the genome.

5.6. Conclusion

A detailed comparison of the gene finding performance across organisms is difficult for various reasons. The quality and amount of RNA-Seq data vary and the quality of annotations, which we cannot directly assess here, is expected to vary significantly. Although the behaviour of all predictions varies drastically between organisms, we

consistently observe a favourable performance of *de novo mGene.ngs* predictions. Nevertheless, we expect further significant improvements when a) employing a more elaborate method for label generation like *MiTie*, b) using e.g. *MiTie* to find additional alternative transcripts for a given *mGene.ngs* prediction and c) adjusting the RNA-Seq subsampling strategy to generate a training set fitting the RNA-Seq coverage distribution of the entire genome more accurately.

In this study we limited the sources of information to RNA-Seq data and genomic sequence in order to compare results to methods not being able to take more information into account. However, the real strength of *mGene.ngs* is its flexibility to incorporate a large variety of information sources. In a pilot study on mouse chromosome 1 we observed an improvement in transcript level F-score from 27.89 (SN: 22.96, SP:35.53) to 30.10 (SN 23.03, SP:43.42) when considering measurements of nucleosome occupancy[13] as features in addition to RNA-Seq data. This finding supports our theoretical understanding that HSM-SVMs can be used to integrate multiple heterogeneous information sources.

Recently, the mass spectrometry technology, allowing for high throughput sequencing of peptides, reached a genome-wide coverage seeming sufficient for incorporating this source of information into genome annotation approaches. So far this is the only direct biological measurement allowing us to distinguish coding from noncoding region of transcripts. We assume that proteome measurements will be very useful to incorporate in *mGene.ngs* together with transcriptome measurements. As far as we know *mGene.ngs* is the only method which is ready to perform the integration of such various types of information sources. Approaches like *Augustus* relying on model selection to tune the weighting of *external evidence* fail as soon as the number of tuning parameters prohibits efficient model selection.

[13] available from `ftp://ftp.ncbi.nih.gov/pub/geo/DATA/supplementary/samples/GSM717nnn/GSM717558/GSM717558%5Fnucleosomes%2Ebed%2Egz`

6. RNA-Seq Based Alternative Transcript Identification (MiTie)

6.1. Introduction

Most of the complexity of higher eukaryotic transcriptomes can be attributed to the encoding of multiple transcripts at a single genic locus by means of alternative splicing, transcription start and termination [e.g., 85, 87]. A comprehensive catalog of all transcripts encoded by a genomic locus is essential for downstream analyses that aim at a more detailed understanding of gene expression and RNA processing regulation.

As introduced in Section 2.3, RNA-Seq is a powerful strategy to measure transcription in a high-throughput manner. In recent years the amount of sequence data obtained from a single sequencing run has drastically increased from a few million reads with length 30-50nt to 150-200 million reads of length 50-150nt. This wealth of information now supports tens of thousands of transcripts in full length, many of them being alternative splice variants of the same gene. Unfortunately, the gene finding methods we have discussed in the previous chapter are limited to predict a single transcript per genomic locus. This is mainly due to the inference using dynamic programming, which can in principle be extended to predict several transcripts at the same time, but this extension is computationally very demanding.

To allow for the prediction of several overlapping transcripts we replaced the inference via dynamic programming by a mixed integer optimization problem (MIP). This framework is more flexible than dynamic programming and allows us to take long range dependencies into account. Nevertheless, the runtime for solving the MIP is on average significantly larger than predicting a single transcript using dynamic programming. This renders the training procedure of a gene finding system with a MIP as inference method computationally very costly. Therefore we decided to implement *MiTie* as a pure inference strategy based solely on RNA-Seq data. In the following we discuss in detail the design decisions of *MiTie*.

Typical RNA-Seq Transcript Prediction Work-Flows In many cases, the RNA-Seq reads are first aligned to a reference genome using an alignment tool that identifies possible read origins within the genome. Contiguous regions covered with read

alignments (possibly with small gaps) are candidates for exonic segments. Alignment tools for RNA-Seq reads, such as *PALMapper* [27, 55], *TopHat* [120], *MapSplice* [126] or *Gsnap* [131], are typically able to identify new exon-exon junctions which are candidates for introns. This information can be compiled into a segment or splicing graph [45], a directed acyclic graph, where the nodes correspond to exonic segments and the edges correspond to intron candidates (cf. Figure 6.4 for an illustration). Assuming complete coverage, an expressed transcript corresponds to a path in the graph. Similar graphs are produced during *de novo* transcript assembly with the difference that the graph can potentially be cyclic and the segments are not explicitly associated with a genomic location. In genome- and assembly-based transcript reconstruction, tools such as *Scripture* [43], *Cufflinks* [121], *Trans-ABySS* [94], *Trinity* [40], and *OASES* [98] select a subset of paths through the graph as transcript predictions. For simplicity, we will focus on genome-based transcript reconstruction when describing the approach and discuss *de novo* assembly whenever necessary.

Challenges in Transcript Prediction with RNA-Seq Data Due to the nature of the RNA-Seq reads, the information obtained from the alignments is of local nature only, even when considering paired-end sequencing [e.g., 103]. The splicing graph representation implicitly assumes independence of local events. Hence, it will typically contain more paths than expressed transcripts. This is also true in the ideal case when the graph is a) complete in the sense that it contains all vertexes and edges and b) accurate in the sense that it only contains expressed exonic segments as vertexes and edges that correspond to introns of expressed transcripts. For instance, the 183,807 splice variants annotated in any of the four human genome annotations [Ensembl, HAVANNA, ENCODE, Vega, see 22, 37, 44] define splicing graphs that encode 707,386 paths, with less than 5% of the loci contribute more than 60% of the paths. Thus, we encounter a few particularly complex cases that contribute most to transcriptome complexity. Defining splicing graphs based on RNA-Seq data entails the additional difficulty that inaccurate or ambiguous read alignments can substantially increase the size of these graphs. While this problem can be addressed by filtering the read alignments, we find that strict filtering often leads to a reduced sensitivity of transcript prediction and introduces artifacts, for instance, in the presence of unknown genomic variations.[1]

[1] In the case of a genomic variation all reads spanning a given genomic position will have at least the number of mismatches of that genomic variation. If this number exceeds the filter criterion, the graph will inevitably have a gap in RNA-Seq coverage and abundance estimates will be significantly biased.

Ideal Design of an RNA-Seq-based Transcript Prediction Methods Based on a detailed analysis of the problem and of previous work, we identified three important requirements that a transcript inference algorithm based on RNA-Seq data should meet. First, following the arguments in [132], we note that it is important to simultaneously identify and quantify transcripts in order to identify long range dependencies. In Section 6.5.1, we illustrate how quantitative information can perfectly deconvolve contributions from multiple transcripts, while ignoring quantitative information leads to inaccurate predictions. Second, enumeration of all paths defined by a splicing graph is often not tractable. For instance, for $\approx 13\%$ and $\approx 3\%$ of human genes, the number of paths in splicing graphs generated from the annotation and RNA-Seq reads (see Section B.5) is greater than 1,000 and 1,000,000, respectively (see Suppl. Figure 6.3). This large number is the result of a combinatorial explosion of possible combinations of alternative segments and edges. This effect is even more severe in case of *de novo* assembly where several loci are merged into cluster of connected segments if sequence is repeated. Therefore, a generally applicable approach should avoid explicit enumeration, ideally still guaranteeing optimality. Third, we show that multiple RNA-Seq samples help to solve the ill-posed problem of transcript identification [see e.g., 63, 72]. By sharing information between samples, while still considering them separately, we can often exactly determine the correct set of expressed transcripts. We provide illustrative examples where neither merging data of multiple samples nor the independent analysis of data from each sample can solve the problem.

We describe an approach called *MiTie* (Mixed Integer Transcript IdEntification) that meets the aforementioned requirements. The main idea of *MiTie* is to report a small, optimal set of transcripts that can well explain the observed RNA-Seq data in multiple samples. It does not require an explicit enumeration of all paths to find the optimal set of transcripts. This is achieved by employing branch-and-bound algorithms that prune parts of the combinatorial search tree that cannot yield the optimal solution.

MiTie consists of two main parts: A data processing part generates a splicing graph from RNA-Seq alignments in BAM-format, the annotation in GTF-format or both (Section 6.4.2). The second part solves the core optimization problem and expects a graph decorated with quantitative information as input (Section 6.4.1ff). The design enables the flexible use of *MiTie* in existing RNA-Seq pipelines. For example, we can use the output of *Trinity*'s *inchworm* tool [40] as input to the second part of *MiTie* and thereby solve the transcript reconstruction task, also solved by *Trinity*'s *butterfly* tool.

6.1.1. Ambiguities in Read Assignment

The goal of RNA-Seq-based quantification tools is to implicitly or explicitly assign reads to transcripts. Regardless of sequencing errors ambiguities remain in two cases. First, if genes are duplicated in the genome reads will map to several genomic locations. The smaller the sequence divergence and the larger the sequencing error rates, the ambiguities of read assignment will increase. Read with several alignment positions are termed *multi-mappers*. The second source of ambiguity stems from transcripts sharing genomic segments. Reads entirely falling into such segments cannot be assigned to a specific transcript without considering information from reads yielding transcript specific features.

Multi-mapper Resolution Assignment of multi mapper reads to one of the locations (or generating expected values for assignment) relies on specific assumptions. A popular assumption is that multi-mappers will more likely come from transcripts with high expression. Many tools compute expression priors based on unique mappers and already assigned multi-mappers and assign reads according to this prior. The second assumption is based on smoothness of local coverage. Here, the flanking regions of a duplicated region (or a more diverged part) are used to determine an expected coverage level and multi-mappers are assigned according to the extent of deviation from this expectation. This strategy heavily depends on the loss function quantifying the deviation, since the choice of the loss function reflects assumptions of the variance of the coverage.

We will describe as strategy to combine both assumptions in Section 6.4.7.

Overlapping Transcripts Reads falling entirely into regions that are part of more than one transcript can only be assigned based on quantification priors. Similar to multi mapper resolution these priors will mainly be determined based on reads yielding transcript specific features.

6.1.2. Splicing Graphs

Based on alignments of RNA-Seq data against a reference genome one can define an acyclic directed graph called *splicing graph*. Nodes in the graph correspond to continuous genomic segments covered by RNA-Seq alignments. Edges in the graph connect neighboring segments and distant segments if there are spliced read alignments indicating a potential intron between the respective segments. In *MiTie* we slightly vary this definition to increase the robustness of the method (see Section 6.4.2 for details. Similar graphs can be defined by overlaps (or identical kmers) of RNA-Seq reads without prior alignment to a reference gene [e.g. 40, 98]. The additional difficulty is that repeats in the transcript sequence lead to cycles in the

graph that cannot be resolved unambigiously. Heuristical solutions include pruning of the entire cycle or cutting the edge of a cycle with the lowest coverage.

6.1.3. Mathematical Models for Read Count Data

In the following we will assume to have R samples of RNA-Seq data, corresponding to different experimental conditions. Sample i may have N_i biological or technical replicates. In this context a technical replicate refers to several sequencing runs using the same RNA-material, whereas a biological replicate is obtained by taking independent biological samples under the same condition.

Variability in RNA-seq measurements can be attributed to different sources. Differences between technical replicates can be attributed to the stochastic process during the sequencing itself. We denote the variance of read counts X for a given genomic region between technical replicates as $\mathbb{V}\mathrm{ar}_t(X)$. The variance between biological replicates is determined by the technical variance and variability in the regulatory system of an organism, termed biological variability. We denote the biological variability as $\mathbb{V}\mathrm{ar}_b(X)$. Finally, by $\mathbb{V}\mathrm{ar}_c(X)$ we denote the variability between samples originating from different biological conditions. Assuming an unobserved read count X_c depending only on the biological condition without biological and technical noise, we can decompose the total variance [cf. 51] with $E(X|X_c) = X_c$.

$$\begin{aligned} \mathbb{V}\mathrm{ar}(X) &= E(V(X|X_c)) + \mathbb{V}\mathrm{ar}(E(X|X_c)) && (6.1)\\ &= \mathbb{V}\mathrm{ar}_b(X|X_c) + \mathbb{V}\mathrm{ar}(X_c) && (6.2) \end{aligned}$$

In a typical RNA-Seq experiment we are interested in finding transcripts that are expressed in a condition dependent manner. Thus, we would like to estimate $\mathbb{V}\mathrm{ar}(X_c)$ based on $\mathbb{V}\mathrm{ar}(X)$ by computing estimates for X_c and $\mathbb{V}\mathrm{ar}_b(X|X_c)$.

Several approaches [e.g. 127] propose to model the read count distribution with a Poisson distribution. The probability density function of the Poisson distribution with parameter $\lambda > 0$ is given by:

$$P(X = k) = \frac{\lambda^k}{k!} * e^{-\lambda} \quad \forall k \in \mathbb{N}_0^+$$

The mean and variance of a Poisson distribution with parameter λ is equal to λ. The Poisson distribution naturally arises when we assume reads are sampled independently with uniform distribution from each position of each copy of a transcript in the biological sample [127]. This model fails as soon as the observed variance is larger than the mean value. To account for biological variability e.g. Anders and Huber [4] propose to use the negative binomial distribution. The negative binomial

distribution arises as a continuous mixture of Poisson distribution where the mixture rate is Gamma distributed with parameters r and p [52]:

$$P_{NB}(X = k|r, p) = \int_0^\infty P_{Pois}(X = k|\lambda) \cdot P_{Gamma}(\lambda|r, p)\ d\lambda \tag{6.3}$$

$$= \frac{\Gamma(r+k)}{k!\Gamma(r)}(1-p)^r \cdot p^k \tag{6.4}$$

The mean and variance of the negative binomial distribution are given by $\frac{pr}{1-p}$ and $\frac{pr}{(1-p)^2}$, respectively. This derivation of the negative binomial distribution motivates the application to model read counts emerging from mixtures of samples. The mean values of the different samples are then assumed to be gamma distributed. This model allows us to adjust the variance $\sigma^2 \geq \mu$ independent of the mean μ, when modeling read count observations.

Estimation of within Transcript Variability Methods for read assignment based on local smoothness assumptions need to model the read distribution within a single transcript. Assuming that the sequencing is a random process selecting reads uniformly along the transcript we could accurately model this process using a Poisson distribution. However, several studies [e.g. 14] found significant biases in read distribution along transcripts, depending e.g. on the position and the sequence of the mRNA (see also Figure 6.1). Assuming that these biases act independently on each read and assuming the summarized effect of all biases to come from an unknown distribution A with mean 1 and variance η, we expect to see $X = \theta_s \cdot X_s$ reads in a given segment s, with $\theta_s \sim A(1, \eta)$ and $X_s \sim P(\lambda)$. Assuming independence of θ_s and X_s we obtain $E(X) = \lambda$ and

$$\mathbb{V}\mathrm{ar}(X) = \mathbb{V}\mathrm{ar}(\theta_s)E(X_s)^2 + E(\theta_s)^2\mathbb{V}\mathrm{ar}(X_s) + \mathbb{V}\mathrm{ar}(\theta_s)\mathbb{V}\mathrm{ar}(X_s) \tag{6.5}$$

$$= \eta\lambda^2 + (1+\eta)\lambda \tag{6.6}$$

We note that under these assumptions we expect the variance of read counts to be a quadratic function of the mean. In Section 6.4.4 we propose a method to estimate the parameters of this model from the data.

6.2. Related Work

6.2.1. Transcript Quantification

Basic Read Counting Methods The simplest strategy for RNA-Seq quantification is to count the number of reads that aligned to the genomic location of a gene

[e.g. 82]. This strategy has several shortcomings. First, reads mapping to multiple locations in the genome cannot be unambiguously assigned to one genomic location. Workarounds like a) ignoring multi-mappers altogether, b) random assignment with uniform distribution and c) uniform partial assignment (add $1/l$ to all l mapping locations) introduce significant biases [68]. Moreover, the read coverage depends on local sequence context and the position of the read in the transcript [14].

Models based on the Poisson Distribution Jiang et al. [57] model read counts using the Poisson distribution. Based on maximum likelihood estimates of the transcript abundance vector a sampling strategy provides confidence intervals for the transcript abundance. The model does not take sequencing biases into account.

rQuant *rQuant* casts the quantification task into a quadratic optimization problem. It was the first quantification tool estimating positional biases (cf. Figure 6.1a) along the length of the transcripts to improve quantification results [13]. Moreover, it takes sequence content biases into account (Figure 6.1b). A significantly simplified version of the *rQuant* optimization problem can be formalized as follows:

$$\min_{w,\beta,\theta} \quad \sum_{p \in G} \|C_p^{exp} - C_p^{obs}\|_2^2 + \gamma_1 \|w\|_1 + \gamma_2 \Omega_2(\beta) + \gamma_3 \Omega_3(\theta) \tag{6.7}$$

$$\text{with} \quad C_p^{exp} = \sum_{t \in T} \mathcal{B}_\beta(p) \cdot \Theta_\theta(p,t) \cdot w_t \tag{6.8}$$

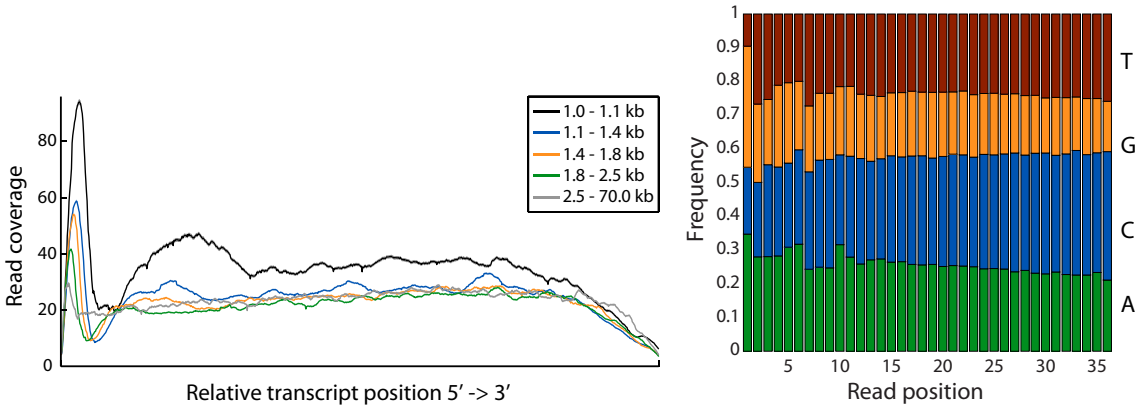
$$\text{s.t.} \quad w_t \geq 0 \quad \forall t \in T \tag{6.9}$$

C_p^{exp} and C_p^{obs} and the expected and the observed coverage at position p of gene G. w_t is the relative abundance of transcript t and functions $\mathcal{B}_\beta(\cdot)$ and $\Theta_\theta(\cdot,\cdot)$ model the content and transcript length bias, respectively. Parameters β and θ for bias estimation are determined within the same optimization problem. This renders the optimization problem to be non-convex, while it is still convex with respect to w, β and θ individually. *rQuant* solves this problem by iteratively solving the convex optimization problem associated with the three parameter types.

While it is appealing to optimize parameters for bias estimation together with transcript abundance in a unified framework, this comes at the cost of significantly increased computing time. As discussed in [75] the l_1-norm on transcript abundance does not lead to solutions as sparse as it would be desirable. This is mainly accredited to the positive nature of the transcript abundance and the fact that the sum over all abundances is more or less fixed.[2] In other words, independently of how the

[2]Significant deviations of the optimal sum over all transcripts will lead to a general under or overestimation and will be strongly penalized by the data fit term of the objective function.

abundance is distributed among transcripts, $\|w\|_1$ will have the same value. The l_2-norm on the goodness-of-fit term 6.7 implies a Gaussian distribution of the read coverage (cf. [10] Chapter 1) with constant variance for all expression levels. This model fails to explain the dependency of mean and variance commonly observed in read count data (cf. Section 6.1.3).



(a) Transcript profiles (b) Read content

Figure 6.1.: **Sequencing biases.** Biases for the *C. elegans* SRX001872 data set [47]. (a) Biases in read coverage estimated by *rQuant* for transcripts of different length ranges. (b) Nucleotide frequency as a function of the position within the reads. Figure from [13].

RSEM *RSEM* [68] builds on a statistical model for observing a read with a given sequence and quality score based on the abundance of all transcripts. The likelihood of observing N RNA-Seq reads in the data set r is given by:

$$P(r|\theta) = \prod_{n=0}^{N} \sum_{i=0}^{M} P(r_n|G_n = i) \cdot P(G_n = i|\theta) \tag{6.10}$$

$\theta \in \mathbb{R}^{M+1}$ corresponds to the relative abundance of M given transcripts and one *noise transcript* explaining unmappable reads. The variable G_n determines the assignment of read r_n to one of the $M + 1$ transcripts. The maximum likelihood values for the transcript abundance vector θ and model parameters are then computed using the EM algorithm [29]. In the E-step of iteration t, expected values for transcript assignments are computed using the $\theta^{(t)}$ as prior ($\theta^{(0)} = \{\frac{1}{M+1}\}^{M+1}$). In the M-step $\theta^{(t+1)}$ is determined based on the transcript assignment. The formulation is guaranteed to find the maximal likelihood solution with arbitrary proximity [68, 69].

Transcript length biases and sequencing errors are incorporated into the model for $P(r_n|G_n = i)$, i.e. the probability of observing read r_n given that the read is assigned to transcript i.

Initially, the *RSEM* procedure assigns multi mapper reads with equal probability to each location. Transcript abundance values for transcripts with high sequence similarity will differ if there are reads mapping to unique portions of the transcripts. Those unique mappers then change the prior probabilities for the next E-step and transcripts with more uniquely mapped reads will accumulate more multi-mappers over time until the fraction of multi mapper assignments fits the fraction of unique mappers.

While being intuitive this strategy fails to determine transcript abundances in cases where one transcript is a short version of another transcript. In this case the short transcript will have only multi mapper reads assigned and its expression will be underestimated. This limitation can be overcome by modeling the spacial distribution of reads along the transcript sequences as implemented in *rQuant*. The homogeneous treatment of multi-mappers between transcript at the same genomic locus and between different loci is conceptually appealing (cf. Section 6.4.7).

6.2.2. Genome Guided Assembly

In the following we will discuss transcript identification methods building on quantification approaches. These methods first generate a set of potential transcripts from a splicing graph and then quantify them. Finally, a subset of transcripts is reported based on the quantification values.

iReckon [75] and *NSMAP* are capable of finding new transcripts but limit the search to transcripts having the same transcription start and termination site as known transcripts. While this significantly reduces the search space it is a biologically implausible restriction. Moreover, the applicability of both methods is restricted to cases where a subset of transcripts is already known.

Scripture [43] enumerates all potential transcripts from a splicing graph and reports them in the result file. While this approach guarantees maximal sensitivity in the case of unfiltered data it is in general not feasible and alignments have to be filtered stringently. The approach does not aim to achieve specific results. *IsoLasso* [71] uses the l_1-norm to regularize transcript abundance. This approach significantly reduces the number of reported transcripts, but the choice of the regularizer is suboptimal given that all abundance values are positive and the sum is fixed. Thus, the regularizer does not penalize a solution explaining the coverage with two very similar transcripts compared to a solution with only one transcript [compare 75].

CLIIQ [72] addresses this problem by applying an integer linear programming (ILP) approach to limit the number of isoforms expressed in any sample combined with

an l_1 loss on the difference of observed and expected coverage. While this is conceptually similar to the *MiTie* optimization problem with respect to the integration of multiple samples the formulation has significant disadvantages. The number of integer variables in the *CLIIQ* ILP depends on the number of potential isoforms which increases exponentially with the number of exons. Thus, given S exonic segments the theoretical runtime of the algorithm is $\mathcal{O}(2^{2^S})$ and therefore stringent filters on the read data and on the enumerated transcripts have to be applied to prevent a combinatorial explosion.

The following approaches avoid the expensive enumeration of transcripts using diverse techniques. *Cufflinks* reports the minimal number of transcripts such that each read alignment is explained by at least one transcript. While this parsimony assumption reduces the computations significantly it is violated by many known genes and is not robust to noise from read alignment. We will discuss the benefits and drawbacks of *Cufflinks* in more detail in Sections 6.5.1 and 6.5.2. *Montebello* [46] uses a probabilistic model to score sets of transcripts and implements a probabilistic search strategy to generate and modify transcript sets until a certain criterion is reached. While this strategy allows for a wide range of functions to quantify the quality of a solution it does not provide any guarantee of optimality. *MiTie* instead guides the search using the branch and bound strategy and can therefore avoid regions in the search space that cannot yield the optimal solution.

6.2.3. De novo Assembly

De novo transcript assemblers have been proven useful in cases where the reference genome is missing or of poor quality. They have the additional advantage of treating alternative transcripts and paralogous genes (resulting in multiple mappings for reads in genome alignment) naturally the same way. The optimization problem formalized by *MiTie* generalizes to solve the transcript prediction task also in the *de novo* setting and we show in Section 6.5.5 that the *MiTie* strategy is superior to the dynamic programming-based strategy of *Trinity*. *OASES* follows a different heuristic which has been shown by the authors to be more sensitive but less specific than the *Trinity* approach. *Trans-ABySS* extends the genome assembly method *ABySS* [102] to cope with the high variation in local read densities observed in RNA-Seq data. Like *Cufflinks* and *OASES*, *Trans-ABySS* does not aim to explain the read data quantitatively during the transcript prediction.

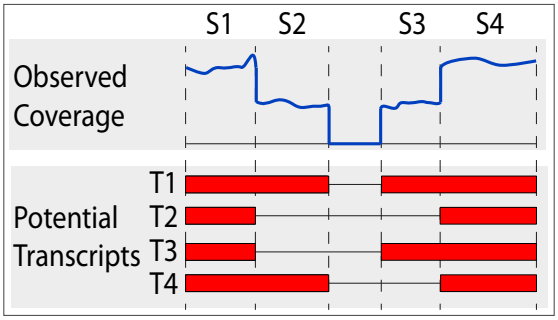


Figure 6.2: The observed read coverage alone cannot unambiguously determine the transcripts present in the sample. Transcript sets {T1, T2}, {T3, T4} and {T1, T2, T3, T4} can perfectly explain the observed coverage. The quantities of splices reads spanning between the segments can exactly resolve this case, even if all four transcripts are present.

6.3. Concepts and Design of MiTie

In this section, we will discuss the design and conceptual ideas that lead to the development of *MiTie*.

Mixture Model As discussed in the previous section it is beneficial to include the spacial structure of transcripts into account when trying to assign reads to transcripts overlapping on the genome. This lead us to model the read information as a mixture of transcripts with unknown abundance including the spacial information similar to *rQuant*.

$$C^{obs} \approx C^{exp} = U^T \cdot W \tag{6.11}$$

$C^{obs} \in \mathbb{R}^P$ and $C^{exp} \in \mathbb{R}^P$ are observed and expected read coverage values for each of the P positions of the locus. All K possible transcripts are encoded in a binary matrix $U \in \{0,1\}^{K \times P}$ and $W \in \mathbb{R}^K$ corresponds to the transcript abundance (cf. Figure 6.5).

Spliced Reads In many cased the coverage information cannot determine the set of expressed transcripts without ambiguities. Such a case is illustrated in Figure 6.2. In this case the quantities of splices reads spanning between any pair of segments S_1 - S_4 can determine the transcripts present in the sample and their relative abundance level. Therefore, we decided to integrate another term into the objective function modeling the abundance of spliced reads as a sum over all transcripts sharing a given intron. This idea has been previously realized in *rQuant*.

Paired-end Reads Paired-end reads have the potential to determine the correct set of transcripts, where coverage and spliced reads fail. Read pairs from one fragment falling into two separate segments indicate transcripts using both segments. If the

fragment size distribution is known, then read pairs can offer information about the splice structure between those segments.

Multi Mapper Optimization We extended the idea of *RSEM* to use an EM algorithm for multi mapper assignment to the case, where the transcript structure is taken into account. Based on an initial random assignment of multi-mappers to genomic loci we compute expected coverages for each genomic locus and then reassign multi-mappers to fit the genome-wide expected coverage. This combines the strength of the mixture model approach to implicitly assign reads to transcripts within a genomic locus and the conceptual advantage of *RSEM* over *rQuant* to model all read assignments (between genomic loci and between different transcripts of the same locus) in a single objective function.

Sparsity As discussed in Section 6.2.1 the l_1-norm does not give sufficiently sparse solutions and has unfavorable stability properties. We decided to use the l_0-norm on transcript weights. This norm directly penalizes the number of transcripts needed to explain the data. Due to the introduction of integer variables this modification causes an increase in computational costs, even in the pure quantification scenario, where all transcripts are assumed to be known.

Multiple Samples Integration of multiple RNA-Seq samples within a single optimization problem has been previously done by assuming similarity of transcript abundance values across samples. *rQuant* for example uses an l_2-norm penalty for deviations of transcript abundance between samples. While this strategy increases stability in abundance estimation, it aims to decrease real differences in abundance, which is not favourable from a biological point of view. We decided to integrate multiple samples without this assumption by extending the concept of the l_0-norm to multiple samples. Whenever a transcript has non-zero predicted expression in one sample it will not incur additional penalty when predicted in another sample.[3] The rationality behind this is to aim for a small set of transcripts explaining the read data in all samples. This is biologically plausible, since in general not all paths in the splicing graph will encode functional products, but our goal is to find exactly those.

Extension for Complex Graphs The number of paths in a splicing graphs with S nodes is in $\mathcal{O}(2^S)$. In the vast majority of cases the adjacency matrix of the graphs is extremely sparse, but we encountered several cases where the number of paths

[3]This idea has been presented at the NIPs workshop "Machine Learning in Computational Biology" in December 2011. It has been developed in parallel by Lin et al. and published [72] in September 2012

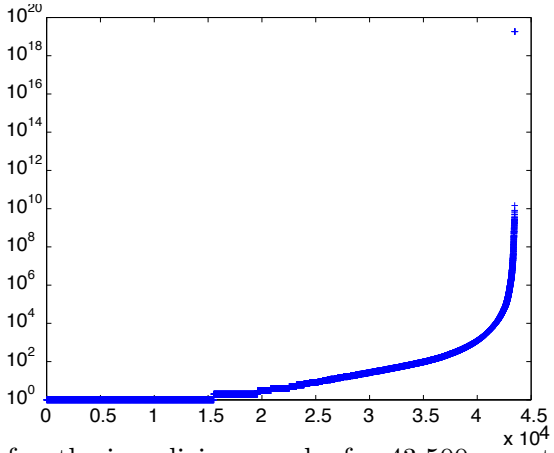


Figure 6.3.: Number of paths in splicing graphs for 43,500 annotated human genes. We merged the four human genome annotations [Ensembl, HAVANNA,ENCODE, Vega, see 22, 36, 37, 44] by concatenating the transcripts and generated a splicing graph. We then extended the splicing graph by adding evidence from two RNA-Seq libraries for cell lines HepG2 (wgEncodeCshlLongRnaSeqHepg2CellLongnonpolyaAlnRep2.bam) and K562 (wgEncodeCshlLongRnaSeqK562CellPapAlnRep1.bam) from `http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeCshlLongRnaSeq/` [36].

prohibits an explicit enumeration. A distribution of the number of paths occurring in graphs constructed from the human genome annotation combined with one RNA-Seq sample obtained from the ENCODE project [36] can be found in Figure 6.3.

To cope with this complexity we limit the number of transcripts we aim to predict to a predefined constant k. This is motivated by two observations. First, as we show in Section 6.5.1, the quantification problem is underdetermined even for much fewer paths in the graph. Therefore, we can have little confidence in the quantification results for transcripts with low relative abundance. Second, introducing a limit for the number of transcripts allows us to compute the optimal quantification result with up to k transcripts without enumerating all possible transcripts. This is related to the concept of *approximation set coding* [e.g. 17].

Modular Design We designed *MiTie* in a modular fashion with a clear interface between data processing and optimization. Different data processing pipelines produce directed acyclic graphs (DAGs) decorated with quantitative values for nodes and edges. The optimization procedure only takes those graphs into account. This allows a flexible integration of *MiTie* into different pipelines, e.g. for genome-based and *de novo* transcriptome assembly.

6.4. Methods

6.4.1. The Core Optimization Formulation

Preliminaries We define segments as sets of neighboring genomic positions corresponding to minimal entities of paths in the splicing graph $G = (\mathcal{S}, \mathcal{I})$ with nodes (segments) $\mathcal{S}$ and edges (introns) $\mathcal{I}$. A segment s can be allowed to be used as the initial or terminal segment in a transcript. This information is assumed to be given as $\iota(s) = \begin{cases} 1 & s \text{ is initial} \\ 0 & \text{otherwise} \end{cases}$ and $\tau(s) = \begin{cases} 1 & s \text{ is terminal} \\ 0 & \text{otherwise} \end{cases}$. The transcript matrix U is defined as a $S \times k$ binary matrix, where $S = |\mathcal{S}|$ and k is a parameter determining the maximal number of transcripts returned by the algorithm. Paths through the splicing graph G can be represented as a binary vector of length S. Let $\mathcal{P}$ be the set of all valid paths, R the number of RNA-Seq samples, and $W_r \in [0,1]^k$ the (normalized) abundance estimates for the k transcripts and sample r. Moreover, $C_r^{exp} \in \mathbb{R}^S$ and $I_r^{exp} \in \mathbb{R}^{|\mathcal{I}|}$ are the *expected* segment read counts and intron confirmation values (from spliced reads) for sample r under our model, respectively. Analogously, C_r^{obs} and I_r^{obs} correspond to *observed* segment and intron counts for sample r.

Using these definitions, the core of *MiTie* is an optimization problem that can be formalized as:

$$\begin{aligned} \min_{W,U} \quad & \sum_{r=1}^{R} (L(C_r^{exp}, C_r^{obs}) + \gamma_1 L(I_r^{exp}, I_r^{obs})) + \gamma_2 \|W\|_0 \\ \text{s.t.} \quad & U_t \in \mathcal{P} \quad \forall t \in \{1, .., k\}, \\ & W \in [0,1]^{k \times R} \\ & U \in \{0,1\}^{S \times k}. \end{aligned} \tag{6.12}$$

For technical reasons we use the normalized transcript abundance $W_r \in [0,1]^k$. However, without loss of generality we can define the maximal observed count as $c_{g,r} = \max(C_r^{obs}, I_r^{obs})$ and then the expected counts can be computed from W_r and U as $C_r^{exp} = c_g U^T W_r$. Similarly, we can compute the expected number of reads $I_{(s_1,s_2),r}^{exp}$ from sample r that span from segment s_1 to segment s_2 for all $(s_1, s_2) \in \mathcal{I}$ as $I_{(s_1,s_2),r}^{exp} = c_g \left(\sum_{t=1}^{k} i_{s_1,s_2}^t \times W_{r,t} \right)$, where i_{s_1,s_2}^t is a binary variable indicating whether intron (s_1, s_2) is part of transcript t. L is a loss function (see Section 6.4.3) and $\|W\|_0$ is defined as the number of non-zero rows in W, hence we only count transcripts that are quantified above zero in any of the samples. γ_1 and γ_2 are hyper-parameters determining the trade-off between the different terms.[4] A version of this optimization problem is illustrated in Figure 6.5.

[4]The hyper-parameters have to be tuned by model selection in order to obtain the best performance. We provide useful default settings (cf. Section B.9).

6.4.2. Constructing the Splicing Graph

We start by defining the boundaries of a region either based on annotated genes or read coverage. If gene/transcript annotations are available, we define regions within each annotated genic locus (see Figure 6.4). Otherwise, we define *islands* by identifying genomic regions that are connected by fragment alignments (cf. Section B.4). Each region may contain exonic and intronic segments and the splicing graph generation is performed independently from other regions. This process is illustrated and described in more detail in Figure 6.4. The main emphasis of the graph generation is completeness. False information can be tolerated to some extent, since the loss function includes a model for noise e.g. originating from wrong alignments.

6.4.3. The Loss Function

A commonly used loss function is the sum of squared deviations between expected and observed values [ℓ_2-loss, see for instance, 14, 71], i.e., $\sum_{s=1}^{S}(C_s^{exp} - C_s^{obs})^2$. The choice of the loss function, however, reflects assumptions on the variance of the measurements [e.g. 83]. The underlying assumption of penalizing the quadratic de-

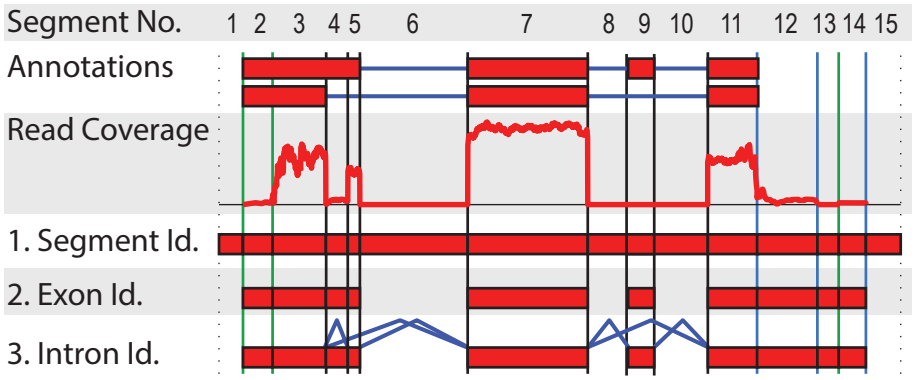


Figure 6.4.: Splicing graph generation from aligned RNA-Seq reads: 1. *Segment identification:* Given a genomic region, we construct splicing graphs by generating a list of segment boundaries. Boundaries are either splice sites (SS) depicted as black vertical lines, potential transcription start sites (TSS; green) and end sites (TES; blue). Potential SS positions can originate from spliced reads (e.g., between segments 4 and 5) or annotated transcripts. Analogously, TSS and TES sites can stem from annotated transcripts or from potential transcript end positions (e.g., between 2 and 3 as well as 13 and 14). See B.4 for more details. 2. *Exon identification:* We keep a) segments that have more than 5% of their nucleotides covered, b) are part of annotated transcripts, or c) if the removal of segment s does not leave any path between two segments connected by paired-end reads (if available). 3. *Intron identification:* We connect segments based on spliced reads and annotated introns.

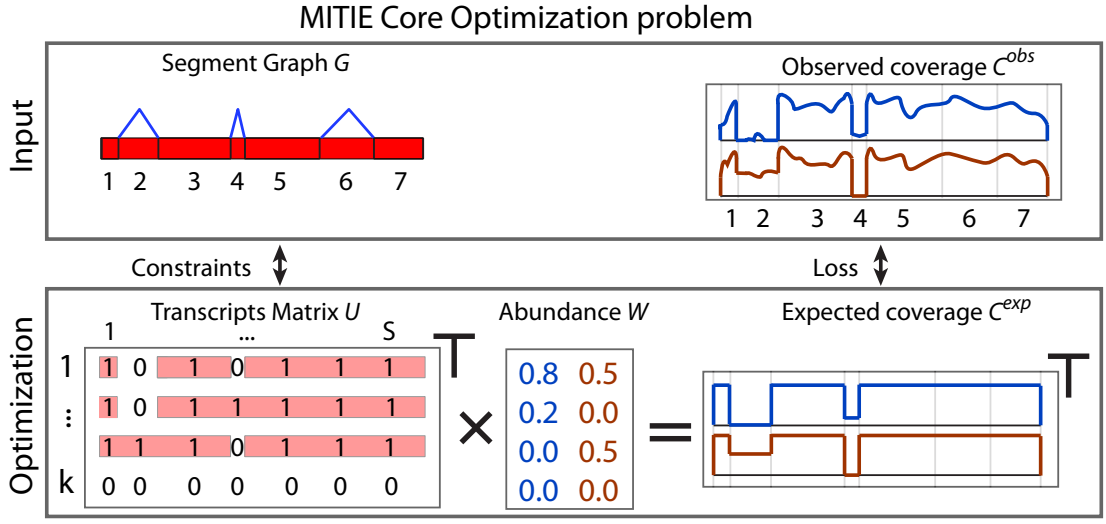


Figure 6.5.: Illustration of the core optimization problem of MiTie. The transcript matrix U (bottom left) and abundance matrix W (bottom center) will be optimized such that the implied expected read coverage of the k valid transcripts (bottom right) matches the observed coverage (top right) well. Validity of the transcripts is ensured by appropriate constraints derived from the segment graph (top left). We illustrate the case of two samples. For each sample we have abundance estimates W for each of the $k = 4$ transcripts. The identity of the transcripts, i.e., the rows of U are shared among the samples. By *Occam's razor* principle, we implement a trade-off between loss between the observed and expected coverages and the number of used transcripts, i.e., number of rows in W with non-zero abundances.

viation is that the measurement is Gaussian distributed with mean equal to the true abundance of the mRNA and variance constant for all expression levels. Following the arguments in Section 6.1.3 and [4, 33] we employ a negative binomial distribution with a standard deviation dependent on the mean of the observation to model the distribution of read count data. We make use of this distribution to define the log-likelihood-based loss function. In addition, we model background noise stemming from false alignments or incomplete RNA processing using a Poisson distribution with fixed mean λ.

We define the likelihood of observing a count V in dependence of the unknown expected count V^* as follows

$$p_M(V|V^*) = \sum_{x=0}^{V} p_P(x|\lambda) \times p_N\left(V - x|V^*, (1+\eta_1)V^* + \eta_2 {V^*}^2\right),$$

where $p_P(\cdot|\lambda)$ is the probability under the Poisson distribution with mean λ and $p_N(\cdot|V^*, (1+\eta_1)V^* + \eta_2 \times {V^*}^2)$ is the likelihood under the negative binomial distribution with mean V^* and variance $(1+\eta_1)V^* + \eta_2 {V^*}^2$. The choice for parameters $\eta_1, \eta_2 \geq 0$ depends on the extent of biases present in the RNA-Seq library. In the

Section 6.4.4, we propose a method for estimating the parameters η_1 and η_2 of this model. We optimize the parameter λ using model selection. In Section 6.5.2 we investigate the impact of different choices for these parameters on sensitivity and specificity of the prediction results.

We assume deviations being independent between the segments and can thus define the negative log-likelihood $\hat{L}(C^{exp}, C^{obs})$ for all segments in sample r as follows:

$$-\log\left(\prod_{s=1}^{S} p_M(C_{r,s}^{obs}, C_{r,s}^{exp})\right) = -\sum_{s=1}^{S} \log(p_M(C_{r,s}^{obs}, C_{r,s}^{exp})).$$

6.4.4. Estimation of Read Count Variability

To estimate the variability of read counts falling into segments we investigated human annotated single transcript genes using one RNA-Seq library from the ENCODE project[5] [36]. We counted read starts falling into 20 randomly selected segments for each transcript and computed mean and variance. Figure 6.6 shows a scatter plot of mean versus variance. As discussed in Section 6.4.3 we model the relationship of mean μ and variance σ^2 with $\sigma^2 = (1 + \eta_1)\mu + \eta_2\mu^2$. We estimate parameters η_1 and η_2 by computing a weighted least squares fit, where the standard least squares is modified using a gaussian distributed weighting (mean zero and standard deviation 5,000) to increase robustness. For $\eta_1 = 0$ we find that $\eta_2 = 0.42$ gives the best fit for the observed data (see Figure 6.6 red dots). We repeated the analysis for one sample obtained from the TCGA RNA-Seq data collection [6] and obtained an optimal value of $\eta_2 = 0.50$. We conclude that we consistently observe a significantly higher variability between segments in the same sample than previously observed for the same segment in different samples (cf. [4] and [33], where values for η_2 ranging from 0.1 to 0.2 were estimated). For the data sets simulated using the Flux Simulator we estimate $\eta_2 = 0.38$.

6.4.5. Implementation as Mixed Integer Quadratic Program

In this section, we detail the implementation of the optimization problem in terms of a quadratic objective function and linear equality and inequality constraints.

[5] library for cell line K562 (wgEncodeCshlLongRnaSeqK562CellPapAlnRep1.bam) from `http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeCshlLongRnaSeq/`

[6] `https://wiki.nci.nih.gov/display/TCGA/RNASeq+Data+Format+Specification` TCGA-DK-A3IM-01A-11R-A20F-07.d0cbd85a-e244-4f99-9adf-71f7afa5f6ec aligned with *Star aligner* [30]

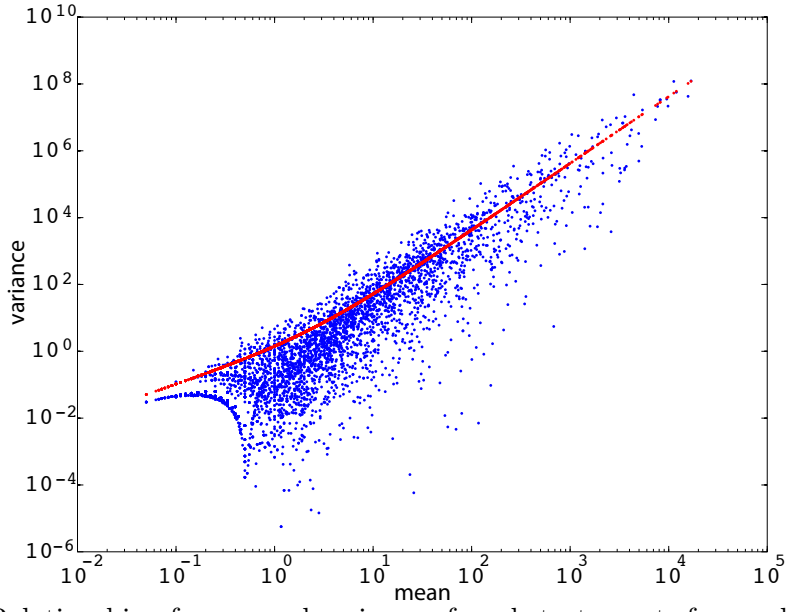


Figure 6.6.: Relationship of mean and variance of read start counts for randomly selected segments in 10,820 single transcript genes (human hg19). We counted read starts for up to 20 non-overlapping regions of length 30nt and computed mean and variance of read start counts. In red we show the weighted least squares fit.

Fitting the Expected Read Counts The read count loss term of the objective function is computed as:

$$\sum_{r=1}^{R}\sum_{s=1}^{S} L\left(\sum_{t=1}^{k} C_{s,t,r}^{exp}, C_{s,r}^{obs}\right)$$

With $L(.)$ being a piecewise quadratic approximation of the negative log likelihood under the negative binomial distribution with mean $C_{s,r}^{obs}$. We implement the piecewise approximation using indicator variables $\mathcal{I}_s$:

$$\mathcal{I}_{s,r} = \begin{cases} 1, & if \ \ \sum_t C_{s,t,r}^{exp} < C_{s,r}^{obs} \\ 0, & else \end{cases}$$

We then equivalently formulate this term of the objective function as

$$\sum_{r=1}^{R}\sum_{s=1}^{S} H_{s,r}$$

with

$$H_{s,r} = \sum_t (q_{left} \cdot \mathcal{I}_{s,r} \cdot \Delta_{s,r}^2 + q_{right} \cdot (1 - \mathcal{I}_{s,r}) \cdot \Delta_{s,r}^2 \quad (6.13)$$

$$+ \; l_{left} \cdot \mathcal{I}_{s,r} \cdot \Delta_{s,r} + l_{right} \cdot (1 - \mathcal{I}_s) \cdot \Delta_{s,r}) \quad (6.14)$$

and

$$\Delta_{s,r} = |\sum_t C_{s,t,r}^{exp} - C_{s,r}^{obs}|.$$

We formulate this relationship with the following linear constraints:

$$\Delta_{s,r} \geq \sum_t C_{s,t,r}^{exp} - C_{s,r}^{obs}$$

$$\Delta_{s,r} \geq C_{s,r}^{obs} - \sum_t C_{s,t,r}^{exp}$$

With binary variables $U_{s,t}$ encoding participation of segment s in transcript t and the relative abundance values $W_{t,r}$of transcript t in sample r, the expected segment count C^{exp} can be computed as $C_{s,t,r}^{exp} = c_g \cdot U_{s,t} \cdot W_t$. c_g is a constant scaling factor for gene g, which is sufficiently large such that optimal $W_{t,r}$ values reside in the interval $[0, 1]$. We equivalently reformulate this equality in terms of linear constraints:

$$C_{s,t,r}^{exp} \cdot \frac{1}{c_g} \leq U_{s,t} \quad (6.15)$$

$$C_{s,t,r}^{exp} \cdot \frac{1}{c_g} \leq -U_{s,t} - W_{t,r} + 1 \quad (6.16)$$

$$C_{s,t,r}^{exp} \cdot \frac{1}{c_g} \geq U_{s,t} + W_{t,r} - 1 \quad (6.17)$$

Reduction of the Search Space We require the transcript abundance $W_{t,r}$ to be zero for each sample r if transcript t does not have any segments:

$$W_{t,r} \leq \sum_s U_{s,t} \quad \forall 1 \leq r \leq R \quad (6.18)$$

Moreover, we sort the transcript in order to reduce the search space by eliminating equivalent solutions corresponding to permutations of transcripts.

$$W_{t,1} \geq W_{t+1,1}, \quad \forall 1 \leq t < k \quad (6.19)$$

l_0 Regularization The transcript indicator variables $I_t = \begin{cases} 1 & \sum_{r=0}^{R} W_{t,r} > 0 \\ 0 & \text{else} \end{cases}$ can be computed as:

$$\sum_{r=1}^{R} W_{t,r} \leq R \cdot I_t \tag{6.20}$$

$\gamma_2 \cdot \sum_{t=1}^{k} I_t$ is part of the objective function.

Prior Knowledge from the Splice Graph We enforce all predicted introns to correspond to connections in the graph $G = (\mathcal{S}, \mathcal{I})$. This can be done by firstly enforcing that, if segment s is used in transcript t (i.e., $U_{s,t} > 0$) any of the segments preceding s in the graph (and being directly connected) is used as well:

$$U_{s,t} \leq \sum_{x \in \{x|(s,x) \in \mathcal{I}\}} U_{xt} \tag{6.21}$$

$$U_{s,t} \leq \sum_{x \in \{x|(x,s) \in \mathcal{I}\}} U_{xt} \tag{6.22}$$

Secondly, we need to make sure, that no intron (s_1, s_2) is used which is not in G:

$$U_{s_1,t} + U_{s_2,t} <= 1 + \sum_{i=s_1+1}^{s_2-1} U_{i,t} \tag{6.23}$$

Since constraints (6.22) force any of the segments directly preceding s_1 to be used, there is no need to exclude intron (s_1, s_3) for any $s_3 > max(\{i|(s_1, i) \in \mathcal{I}\})$. If segment s is a potential TSS or TES site, we allow a connection to conceptual *source* and *sink* nodes similar to introns, for which we do not expect to see any evidence.

Fitting Spliced Alignment Counts To compute expected intron counts $C^{I,exp}_{s_1,s_2,t}$, we first determine if the intron (s_1, s_2) is used in transcript t and if so, we set $C^{I,exp}_{s_1,s_2,t}$ be equal to W_t. Otherwise we set $C^{I,exp}_{s_1,s_2,t}$ to zero. This can be formulated as follows:

$$C^{I,exp}_{s_1,s_2,t} = W_t \cdot U_{s_1,t} \cdot U_{s_2,t} \cdot \prod_{i=s1+1}^{s_2-1} (1 - U_{i,t}) \tag{6.24}$$

This relationship can be expressed in terms of linear constraints as follows:

$$C^{I,exp}_{s_1,s_2,t} \cdot \frac{1}{c_g} \leq U_{s_1,t} \tag{6.25}$$

$$C^{I,exp}_{s_1,s_2,t} \cdot \frac{1}{c_g} \leq U_{s_2,t} \tag{6.26}$$

$$C^{I,exp}_{s_1,s_2,t} \cdot \frac{1}{c_g} \leq 1 - U_{i,t} \quad \forall s_1 < i < s_2 \tag{6.27}$$

$$C^{I,exp}_{s_1,s_2,t} \cdot \frac{1}{c_g} \leq W_t - U_{s_1,t} - U_{s_2,t} + 2 + \sum_{i=s_1+1}^{s_2-1} U_{i,t} \tag{6.28}$$

$$C^{I,exp}_{s_1,s_2,t} \cdot \frac{1}{c_g} \geq W_t + U_{s_1,t} + U_{s_2,t} - 2 - \sum_{i=s_1+1}^{s_2-1} U_{i,t} \tag{6.29}$$

Paired-end Information For all pairs of segments (s_1, s_2) that have confirmation from paired end alignments to be part of a single transcript, we compute binary variables $P_{s_1,s_2,t}$ indicating that segment pair (s_1, s_2) is part of an expressed transcript t:

$$P_{s_1,s_2,t} = U_{s_1,t} \cdot U_{s_2,t} \cdot I_t \tag{6.30}$$

In terms of linear constraints this can be rewritten as:

$$P_{s_1,s_2,t} \leq 1/3(U_{s_1,t} + U_{s_2,t} + I_t) \tag{6.31}$$

$$P_{s_1,s_2,t} \geq U_{s_1,t} + U_{s_2,t} + I_t - 2 \tag{6.32}$$

Then, we compute binary variables $P^n_{s1,s2}$ indicating weather segments s_1 and s_2 confirmed by paired-end reads do not occur together in any predicted transcript with nonzero expression:

$$P^n_{s_1,s_2} \geq -\sum_{t=1}^{k} P_{s_1,s_2,t} + 1 \tag{6.33}$$

The term $N_{s_1,s_2} \cdot P^n_{s_1,s_2}$ is part of the objective function, where N_{s_1,s_2} is the number of paired-end fragments supporting the connection between segments s_1 and s_2.

6.4.6. Confidence Quantification for Transcript Calls

Given a set of k transcripts we are interested in the importance of each transcript for explaining the total RNA-Seq data. We make use of a likelihood-ratio test to

quantify the confidence in each predicted transcript t. We compute the test statistic:

$$T = -2ln\left(\frac{\widetilde{p}(D|M)}{\widetilde{p}(D|M_t)}\right) \tag{6.34}$$

Where $\widetilde{p}(D|M)$ is the approximate likelihood of observing the read data D under our model M based on all k transcripts and $\widetilde{p}(D|M_t)$ is the approximate likelihood when restricting the quantification value of transcript t to zero. To compute this we solve the quantification task k times using all transcripts from the transcript inference step and set the quantification value of transcript t to zero. We compute the objective function setting all regularization parameters to zero. We assume the test statistic to be χ^2- distributed with $df = k - (k-1) = 1$ degrees of freedom and compute a p-value for each transcript. This strategy allows us for example to estimate the probability that a newly predicted transcript explains features of RNA-Seq data that cannot be explained by known annotated transcripts.

6.4.7. MiTie+MMO: Joint-Optimization of Transcript Abundance, Structure and Multiple Mapping Locations

Mitie seeks to maximize the approximated likelihood of the observed data being generated by the predicted transcripts. Ambiguities in read alignment result in uncertainties in the *observed* data which are impossible to resolve without knowing the transcripts and their abundance. This interdependence of finding the correct transcripts and finding the correct alignments can be accounted for by treating not only the transcripts and their abundance as variables, but also the alignments. We make use of the statistical model introduced in Section 6.4.3 and maximize the likelihood also with respect to the read alignments.

Starting with alignments where only the alignment with the highest alignment score was present we predicted transcript structures and abundances. Given the transcript structures and abundances, we can compute the expected read counts for segments and exon-exon junctions (C_r^{exp} and I_r^{exp}; see Section 6.4.1). For each read with multiple alignments of similar quality we selected the alignment that maximized the likelihood (approximated with the $\widetilde{NB}$-loss function). We define the alignment quality here as the number of edit operations[7] of the alignment. We consider all alignments to have similar quality if the number of edit operations differs by less than a predefined constant r.[8] We then recompute transcript structure and abundance

[7] The total number of insertions, deletions and mismatches

[8] We found that a very stringent cutoff with $r = 0$ works best in the cases we observed. This is expected since the chance that an alignment with r more edit operations than the best alignment is the correct alignment is expected to decreases rapidly with r, given error rates of about one percent in Illumina reads and an even lower chance for genomic alterations.

predictions and iterate until very few alignments change. This EM-like algorithm can find a local maximum of the approximated likelihood function.

6.5. Results

6.5.1. An Illustrative Simulation Study

We start by considering a specific case of transcript inference to illustrate the limits of transcript identification from a single sample and to show how multiple samples can help identifying commonly expressed transcripts. In Figure 6.7A, we consider a splicing graph encoding three exons skips leading to eight possible transcripts. The task is to determine which transcripts are expressed. We consider multiple samples and assume that the same small set of transcripts are expressed in all samples but with different abundances (including the possibility of zero abundance).

This problem can be reduced to solving systems of linear equations [63]. If a system of equations is solvable, then the corresponding set of transcripts can fully explain the observed read coverage (see Section B.6; for simplicity, we ignore statistical fluctuations and use exact quantities). In case of multiple samples, we identify the sets of transcripts that are consistent with all samples (intersection of the sets of sets of transcripts). If only one such set of transcripts remains, then we can be sure to have found the correct solution ("exactly one solution"). If several sets remain, the best strategy is to randomly select one set out of the possible ones ("optimal strategy").

It turns out inference for the considered example becomes increasingly more difficult the more transcripts are expressed (Figure 6.7B). If only one transcript is expressed, all strategies always find the correct answer. If two of the eight transcripts are expressed, it is theoretically always possible to identify them correctly. Also, *Cufflinks* and *MiTie* often identify the correct set of transcripts (see Figure 6.7B top). Repeating the same experiment for three expressed transcripts, the observations change completely. Only in 16% and 60% of the cases, there is exactly one solution or the optimal algorithm identifies the correct one (one sample), respectively. The success rate increases significantly with the number of samples (88% and 95%). The accuracy of *MiTie* is close to the optimum and better than the optimal conservative algorithm. *Cufflinks* finds the correct 3 transcripts in only 2% of the runs, which comes close to randomly guessing three out of eight (1.78%) (see Figure 6.7B middle). If four out of eight transcripts are expressed *Cufflinks* never finds the correct solution, while *MiTie* performs comparable to the optimal strategy (cf. Figure 6.7B bottom).[9]

[9] *Cufflinks* was run on merged samples since the *Cufflinks*/*Cuffmerge* combination as described in Section 6.5.2 did perform worse.

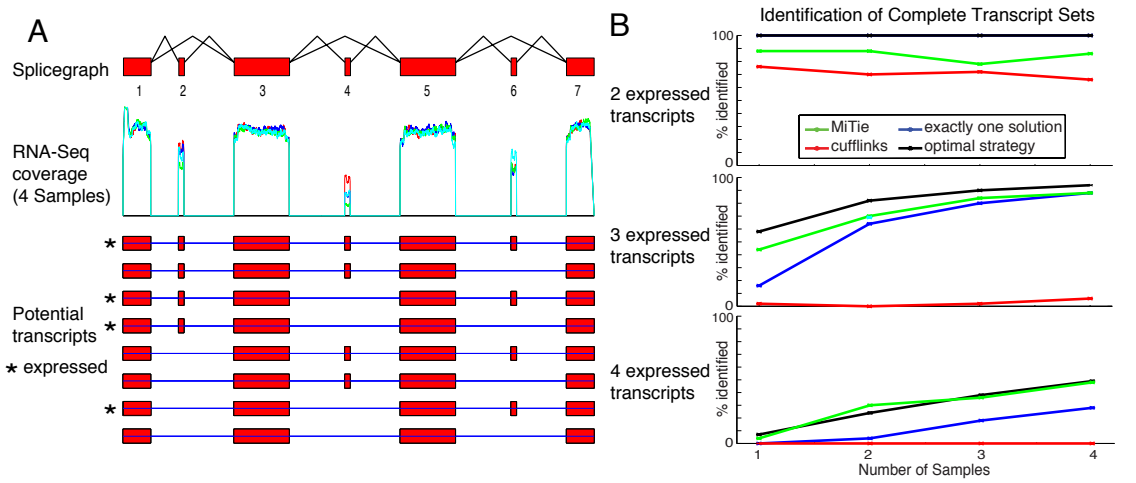


Figure 6.7.: **A** Example with four samples of simulated reads. All four samples express the same 4 transcripts (marked with asterisks) with different relative abundances. The different relative abundances lead to distinct coverage patterns in the alternative regions. **B** We randomly selected 2 (top), 3 (middle), and 4 (bottom) transcripts and simulated 4 samples RNA-Seq reads each. For each sample, we uniformly redistributed the abundance between the selected transcripts. We then predicted transcripts with different methods. The prediction was counted as correct if all transcripts were exactly matched and no additional transcripts were predicted. To obtain more robust measurements we repeated the whole procedure 50 times and report the mean number of correct predictions for each method.

6.5.2. Simulated Data for *H. sapiens*

Read Simulation

A major obstacle for the evaluation of tools for transcriptome reconstruction is the lack of a gold standard set of RNA-Seq libraries and known expressed transcripts. Simulated reads have the advantage that we can evaluate different aspects of predictions which we would not be able to observe in reality. They are therefore an important part in evaluating many RNA-Seq-based algorithms. To obtain realistic RNA-Seq read alignments, we a) randomly draw the transcript abundances in multiple samples, b) used the FluxSimulator [41] to incorporate typical biases from library preparation and sequencing, and c) introduced errors into the generated reads and d) mapped the generated reads against the whole genome (see Section B.2 for more details). For this study, we generated simulated reads for a set of $1,000$ human genes with $8,592$ transcripts in total. The first 500 genes were used to tune hyper-parameters for all compared methods. Reported results correspond to the performance on the second set of 500 genes.

Quantification, Loss Functions and Multi-Mapper Resolution

Figure 6.8A illustrates the effect of different loss functions on the Pearson correlation of predicted and ground truth transcript abundances. Similar to *MiTie*'s loss function ($\widetilde{NB}$-loss) we implemented a quadratic proxy function for the negative log-likelihood under the model of Poisson-distributed reads ($\widetilde{P}$-loss). The $\widetilde{NB}$-loss generally gives more accurate results than the $\widetilde{P}$-loss and the ℓ_2-loss. Both the $\widetilde{P}$-loss and $\widetilde{NB}$-loss are significantly more robust to erroneous data (for instance, spurious alignments when allowing more mismatches) than the ℓ_2-loss.[10]

We investigated the impact of the three hyper-parameters on the ability of *MiTie* to detect expressed transcripts. Figure 6.9 shows sensitivity and specificity for varying loss parameter η_1, η_2 and λ, respectively. We switch to a binary evaluation measurement here, because the correlation prefers non-sparse, and therefore less specific solutions. The binary representation of the predicted abundance allows us to distinguish between sensitivity and specificity.

We observe that increasing η_2, corresponding to a model expecting higher read count variability for segments with high levels of RNA-Seq evidence, leads to more

[10]For independence of hyper-parameters, we only use the exon coverage.

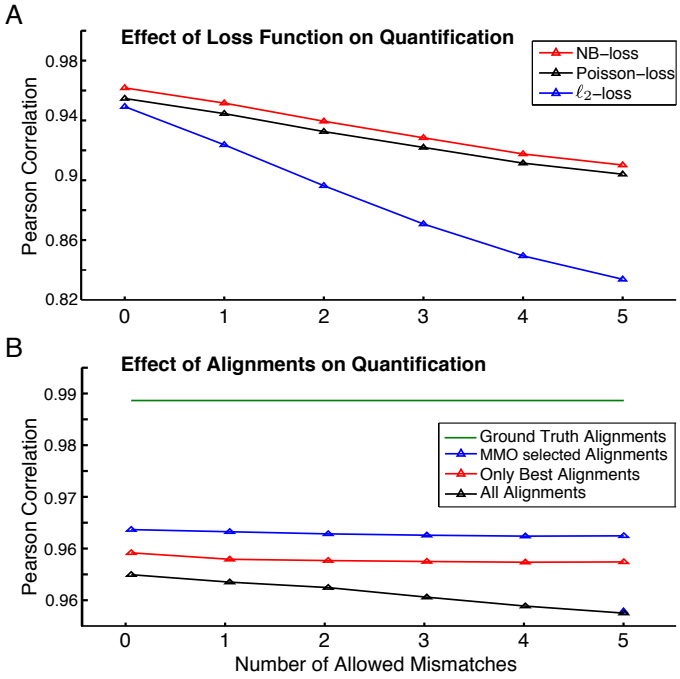


Figure 6.8.: **Assessment of quantification results. A** MiTie quantification results for the three different loss functions $\widetilde{NB}$, Poisson, and ℓ_2-loss. We consider stringent (0 mismatches) and liberal read alignments (up to 5 mismatches), leading to fewer or more multi-mapping reads, respectively. **B** MiTie quantification results with $\widetilde{NB}$-loss, when considering ground truth alignments, all multiple alignments, or after multi-mapper handling with *MMO* (see Section 6.4.7).

specific, but less sensitive results. The higher expected variance allows us to explain a larger fraction of the RNA-Seq data with a single transcript. Therefore, the threshold for "triggering" an additional transcript is increased. Similarly, increasing λ results in more specific results, since an increased fraction of RNA-Seq data can be explained as noise (Figure 6.9C). Increasing the parameter η_1 also results in more specific results, but similarly to the effect when increasing λ the increase in specificity saturates quickly and higher values only result in a lower sensitivity. In both cases we allow more transcripts to be quantified as zero, and therefore loose sensitivity, while segments with higher expression levels are essentially unaffected.

We also investigated the effect of multi-mapper handling on the quantification performance (Figure 6.8B). For this experiment, we used the full set of *MiTie* features as described above and the $\widetilde{NB}$-loss. We observe that using all features, the quantification is much more robust with respect to noise in the reads. Moreover, by using *MMO* (see Section 6.4.7) one can significantly improve the quantification. After resolving multi-mappers with *MMO*, the quantification improves even beyond less stringent filtering.

Finally, we evaluated how indicative the confidence value based on the likelihood-ratio test (Section 6.4.6) is for a transcript to be expressed. We find that among the 4718 nonzero quantified transcripts with p-value smaller than 0.1 we have 86% correct transcripts with nonzero simulated expression, whereas out of 326 transcripts with p-value larger or equal to 0.1 we find 44% correct predictions. This result shows that the confidence values accurately indicate cases with possible alternative explanations. We argue that it is more favourable to use the confidence values to filter transcripts than the frequently employed filtering based on relative or absolute abundance estimates, because ambiguities might originate from the topology of the

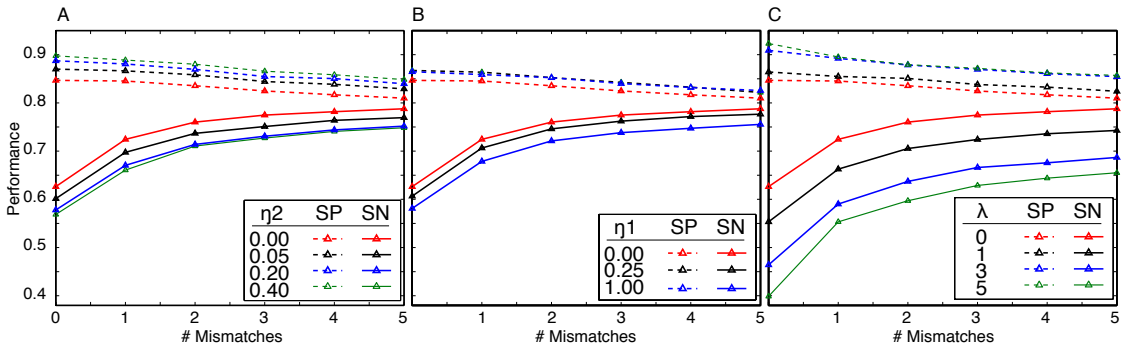


Figure 6.9.: **Loss function parameter selection.** We assess the three parameters of the loss function in terms of sensitivity and specificity of the quantification results of annotated transcripts and simulated reads. Simulated and predicted transcripts were binarized to compute sensitivity and specificity.

splicing graph and are therefore independent of the expression level. This is supported by a relatively low Pearson correlation between predicted relative transcript abundance and p-values of only 0.18, indicating that the likelihood-ratio test adds additional information about the topology of the graph which cannot be retrieved from the predicted abundance alone.

Accuracy of Transcript Prediction (MiTie and *Cufflinks*)

For most genes of many organisms we know a subset of transcripts in advance. The known transcripts are likely the ones with the highest expression level as those are easiest to identify by traditional annotation strategies. To test the accuracy for this realistic scenario, we omit the information of all transcripts, except the one that has the highest simulated abundance. We ran *MiTie* and *Cufflinks* both given only this one annotated transcript and the RNA-Seq reads from a larger set of transcripts to predict transcripts. We compare transcript-level sensitivity and specificity of the predictions relative to all known transcripts. We counted transcripts as being correct, if the intron structure matched the one of an annotated transcript. Single exon transcripts were counted as being correct if they overlapped with an annotated single exon transcript. Each prediction was matched to at most one annotated transcript and each annotated transcript was associated to at most one predicted transcript.

The results for one to five samples are shown in Figure 6.10A. For *Cufflinks* we optimized the hyper-parameters (see Section B.7) and used two different strategies to perform predictions. The first strategy merged all RNA-Seq alignments and the second strategy merged individual *Cufflinks* predictions for each of the samples with *Cuffmerge*. We observe that the latter strategy outperforms the data merge strategy, but both strategies can not benefit from additional samples. This is mostly attributed to a drastically decreasing specificity, while the sensitivity improves with more samples (cf. Suppl. Figure B.1). *MiTie* with *MMO* outperforms the best *Cufflinks* prediction on average by 6.7 percentage points in F-score. *MiTie*/*MMO* on five samples is 2.4% more accurate than with one sample. We observe that the significant improvements *MMO* contributes in quantification accuracy to not translate to similarly high transcript recognition improvements. We attribute this to the robustness of the loss function.

6.5.3. Runtime comparison

In Figure 6.11 we compare the runtime of *Cufflinks* and *MiTie* on 1000 genes with simulated RNA-Seq reads. We discarded the time spent on regions not overlapping the respective genes. For each method we only recorded the time needed in the core algorithmic part, disregarding data input, output and processing. For *MiTie*

we recorded the CPU-time needed to solve the mixed integer optimization problem using the MATLAB function *cputime*. For *Cufflinks* we recorded the CPU-time spend in the *assemble_bundle* method in the *cufflinks.cpp* file using the *boost::chrono* library. We recorded the time for each *bundle* and stored it with bundle start and stop coordinates in a separate output file. For each gene we searched for overlapping regions (bundles in the case of *Cufflinks*) and added the CPU-time spent on the region. Regions overlapping with more than one gene contributed only to the runtime of one gene.

We note that the runtime of *Cufflinks* is favourable for filtered alignment files. However, we were not able to compute *Cufflinks* predictions for unfiltered alignment files. This is likely caused by additional alignments (potentially a single alignment) merging large graphs resulting in an explosion of runtime and memory consumption. This observation was consistent over several *Cufflinks* releases.

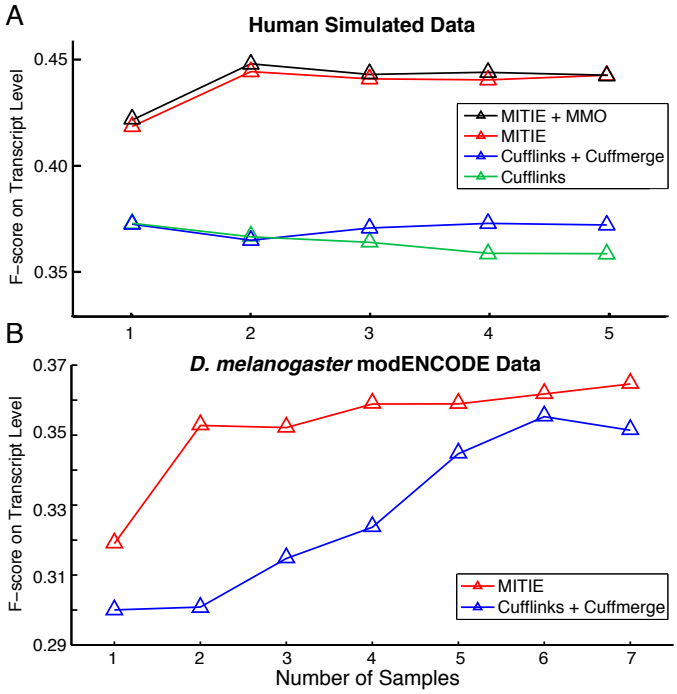


Figure 6.10.: *MiTie*/*Cufflinks* **transcript prediction evaluation. A** Transcript-level F-score as a function of the number of samples for the simulated human data set. **B** Transcript-level F-score as a function of the number of modENCODE samples for up to seven developmental stages of *D. melanogaster*.

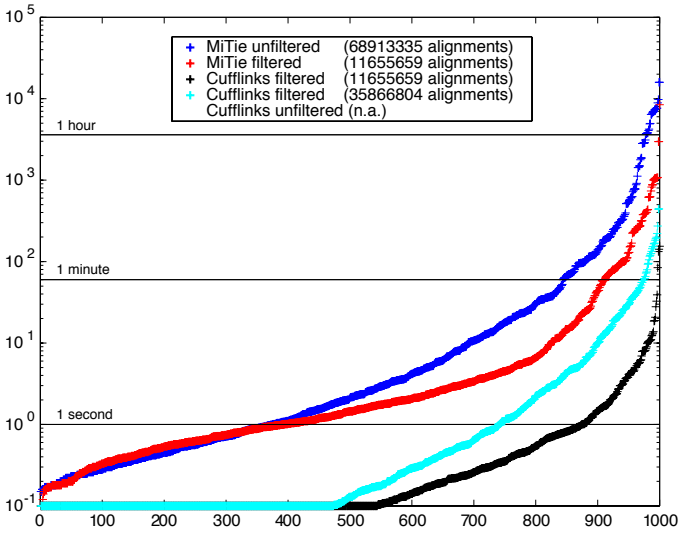


Figure 6.11.: *MiTie* **runtime assessment.** Runtime for the prediction of 1000 genes with simulated RNA-Seq reads. We show the runtime for identical parameter settings evaluated in the main paper. *MiTie* uses unfiltered read alignments. In several attempts we failed to compute a *Cufflinks* run with unfiltered read alignments on a 64 core machine with 512GB memory. Instead we computed *Cufflinks* runs with increasing number of alignments by allowing more mismatches per alignment. Number of alignments is given in brackets. Values smaller than 0.1 were set to 0.1.

6.5.4. Predictions for Developmental Stages of D. melanogaster

Setup

To show that the performance improvements we have seen on simulated data translate to large scale, experimental data sets, we applied *MiTie* to a data set of seven developmental stages of *Drosophila melanogaster* (550M alignments from 38 RNA-Seq libraries for 7 developmental stages). We filtered the modENCODE *D. melanogaster* genome annotation for genes with at least two annotated transcripts. We then randomly removed one transcript variant (a transcript differing in splice structure to all other transcripts) which had a non-zero *Cufflinks* quantification value. We discarded genes where no such transcript could be found. From the remaining genes we randomly selected $1,000$ genes for tuning the hyper-parameters and $1,000$ genes for testing. This setup retrospectively simulates the identification of new transcripts in already well-annotated genomes (as in Section 6.5.2).

Results

We evaluated the sensitivity of *MiTie* and *Cufflinks* based on the omitted transcripts and the specificity with respect to all annotated transcripts. Figure 6.10B shows a comparison of the F-score as a function of the number of samples. *MiTie*

outperforms the best (in terms of F-score) *Cufflinks* prediction in sensitivity and specificity. Similar to the simulated data, the merge of the *Cufflinks* predictions using *Cuffmerge* significantly outperforms the *Cufflinks* prediction on merged data (not shown). While having a similar performance for large sample numbers, *MiTie* has a much higher F-score on up to five samples. This is mostly due to a higher sensitivity at a similar specificity.

Since we used the alignment files provided by [20], we had no control over the quality or sensitivity of the alignments and multi-mapper resolution. Our results on simulated data let us expect an even higher performance for more sensitive alignments and appropriate multi-mapper handling.

6.5.5. Application to De Novo Assembly

Comparison to *Trinity*

On the same set of simulated reads we also compared the core optimization of *MiTie* to the transcript calling method *Butterfly*, which is part of the *Trinity* pipeline. We ran the entire *Trinity* pipeline and then generated *MiTie* predictions based on the graphs reported by the *Trinity* component *Chrysalis*.

We evaluated the performance of both methods by aligning predicted mRNA sequences to the annotated mRNA sequences. A prediction was counted to be correct if a) it was $\leq 1\%$ longer than the annotated transcript and b) the region 20nt upstream of the first exon-exon junction to 20nt downstream of the last exon-exon junction aligned with at most 5 edit operations to the reference sequence.[11] In its current implementation, *Trinity* is not capable of integrating multiple samples. Therefore, we compared the results using only a single sample. We performed a model selection to tune hyper-parameters of *Trinity* and observed that the parameter determining the merging/splitting behaviour of components (*–min_glue*) strongly influences the performance of *Trinity*. If *–min_glue=1* predictions are more sensitive but approximately 15 percentage points less specific compared to the performance with *–min_glue=2* (default). For both sets of predictions we selected Pareto-optimal predictions and ran *MiTie* on the corresponding graphs. The *MiTie* core optimization problem outperforms *Butterfly* significantly in terms of sensitivity while having similar or higher specificity (cf. Figure 6.12).

Since *Trinity* applies stringent filtering in the *Inchworm* step, the obtained segment graph does not contain all true transcripts. From our results on genome based assembly we expect even higher performance gains with a more sensitive graph generation algorithm. Furthermore, we expect improvements from multiple samples, which can theoretically be utilized in the same way as in genome based assembly.

[11] For efficiency reasons, we ran the entire experiment for each gene separately on a FASTA file only containing the simulated reads as they were simulated from this genic locus without mismatches.

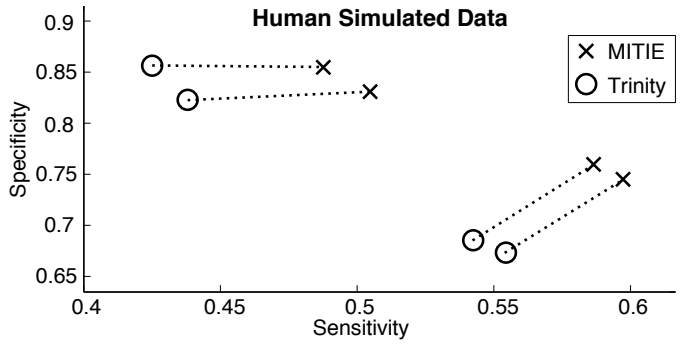


Figure 6.12.: *MiTie Trinity* **comparison.** Cycles show a subset of *Trinity* model selection runs. We selected the pest performing predictions for different trade-offs of sensitivity and specificity. We ran *MiTie* predictions on the De Bruijn graphs generated by trinity. Dotted lines connect the corresponding predictions.

6.6. Conclusion

The transcript prediction problem is typically under-determined. One important consequence of this observation is that deeper sequencing only helps to reduce the variance of abundance estimation and to close gaps in the splicing graph, but it does not solve the transcript identification problem as such. The proposed method reduces the set of solutions by leveraging quantitative information and multiple RNA-Seq samples combined with mild, biologically plausible assumptions. Furthermore, prior information can be taken into account in a direct way within a single optimization problem, which we think will turn out particularly advantageous for integrating long reads from third-generation sequencing platforms [97] with RNA-Seq data.

Our results highlight the importance of a well-motivated loss function to penalize the read count deviation. The application of the $\widetilde{NB}$-loss significantly improved our quantification and transcript recognition results, while it comes at nearly zero additional computational cost.

The underlying assumption of previously published transcript calling strategies like *Cufflinks* and *Trinity* is correctness and completeness of the graphs. Achieving both at the same time is challenging and typically not possible. This results in either wrong transcript predictions that have to be filtered out heuristically or in fragmented transcript predictions. *MiTie* assumes completeness of the graph, but not correctness. Completeness can often be achieved by not filtering the input alignments or not pruning the assembly graph. The decision of filtering can be deferred to the optimization problem that may choose to discard information in a context-dependent way. This is conceptually more attractive than global and uninformed filtering as a pre-processing step.

MiTie finds a solution which is compatible to the overall observed read data. As

observed on simulated and real world RNA-Seq data, *MiTie* pushes the boundaries of what can be observed from RNA-Seq data towards more complex mixtures of transcripts by leveraging variability between samples. These improvements come with the downside of higher computational costs, however, the vast majority of cases can be optimally computed within seconds and our implementation provides options for approximations in cases where exact computations are too expensive. Furthermore, our experiments clearly show that we can obtain the same performance as competing methods with only a fraction of the data. Which in turn can save the time, money, and storage capacity of deeper sequencing.

MiTie allows us to pool information from different samples in an effective way. This conceptual improvement will further future RNA-Seq studies; rather than spending efforts into very deep sequencing of a few samples, future studies will have the choice to investigate a larger variety of samples at a lower depth. The combination of these samples allows us to obtain more confident transcript predictions in each sample and more insights into the biological questions at the same time.

7. Conclusion

Investigation of the transcriptional landscape provides clues to connect genotype to phenotype. However, estimating the transcriptome state in a qualitative and quantitative manner from current high-throughput measurements poses profound computational challenges.

We devised two strategies for improved characterization of the transcriptome complementing each other in the applicability. *mGene.ngs* combines evidence from RNA-Seq data with information from the DNA sequence in a structured output learning framework to predict a single transcript per genomic locus.

This design was motivated by the high variability and error rates in expression measurements and has proven to be very powerful and flexible also with respect to other high throughput information sources. The major limitation however is that *mGene.ngs* can only predict a single transcript per genomic locus. Given the coverage levels in early RNA-Seq experiments this limitation was less severe than it is nowadays, where tens of thousands of transcripts may be fully covered with a single lane of RNA-Seq data.

For such genomic loci with large amounts of RNA-Seq evidence *MiTie* detects transcripts using a mixed integer optimization approach solely based on the RNA-Seq data. While the performance for those cases is decent, it is difficult to judge whether a given transcript prediction is full length or fragmented by a gap in the coverage. This decision should be taken using features from the genomic DNA sequence and prior knowledge about the structure of genes like the existence of open reading frames and length distributions of exons and introns.

In the following, we describe additional ideas and strategies that go beyond this work, but will likely lead to additional improvements.

Combination of Methods We provide indirect means to integrate such prior knowledge by accounting for known gene structures during the prediction step. However, it is non-trivial to take the confidence of these gene structures into account and find new genes structures with similar features. The most promising approach is to combine the powerful learning strategy of *mGene.ngs* with the MIP based inference strategy of *MiTie*. In this case the MIP would replace the dynamic programming based Viterbi algorithm. This would allow us to find transcripts that explain the RNA-Seq coverage as good as possible and exhibit genomic features similar to genes

in a training set. We have proposed this idea in the book chapter [9]. In order to train such a system a training set with accurately annotated alternative transcripts is needed. We think that using RNA-Seq data and a high quality genome annotation it should be possible to compile such a training set, at least for commonly used model organisms. The main challenge remaining is the computational complexity for training such a system.

The implementation of the combination of both strategies lies beyond the scope of this thesis. But we think that this is a very promising direction for further research, given that both methods individually were shown to further the state-of-the-art in transcriptome annotation.

RNA-Seq Library Normalization The extensive dynamic range of transcript expression levels renders the detection of lowly expressed transcripts very difficult. For the mouse transcriptome data (cf. Section 5.5) we observed that 40% of the genes have less than a hand full of reads mapped to them, although we can expect that a considerable fraction of those genes are expressed at low levels.

In principle there are two possibilities to directly measure the expression of these transcripts. First, the depletion of highly expressed transcripts from the cDNA library by designing specific probes that capture those transcripts. Using duplex-specific nucleases the captured transcripts can be degraded in solution [12]. This can be accomplished because it is known that very few genes contribute the vast majority of messenger RNAs in a given sample [12]. Second, the enrichment of lowly expressed transcripts by designing specific probes against these transcripts. This could be done cost efficiently using standard exome capture kits designed for DNA-exome-sequencing. The rationality behind this strategy is that highly abundant transcripts saturate the capture probes and remaining transcripts of the same type will be washed away.

The benefit of the first technique is that the identification of transcripts is not limited to the set of known and predicted transcripts, but the depletion will likely be less effective than the enrichment strategy. However, with both techniques we will likely miss lowly expressed isoforms of highly expressed genes. Both methods will distort the quantitative information of the RNA-Seq data, but for the depletion strategy we can expect relative transcript abundance values of genes not targeted by the depletion probes to be largely conserved. In the enrichment scenario, relative transcript abundance are only expected to be conserved between transcripts that share the same probe target sites. Under the assumption that only full length transcripts are removed from the library, *MiTie* is still applicable. We believe that library normalization is a very promising approach, as it has already proven to be effective in EST sequencing [105]. It may allow us to correct the transcript structures of many lowly expressed genes and find a large number of so far uncharacterized

transcripts even for well studied model organisms.

Third Generation Sequencing So called third-generation sequencing technologies (3GS) [97] allow for direct measurement of large fractions or even full length RNA molecules. But the throughput of these technologies is significantly lower than for the second-generation sequencing. Therefore, a promising approach to obtain more accurate transcript predictions for a large fraction of genes is to integrate both types of measurements. There are several possible solutions for integrating 3GS reads with *MiTie*. A straightforward and computational efficient way is to use the paired end feature of *MiTie* to account for 3GS reads. By doing so, we impose a fixed penalty on a solution that does not contain a pair of segments together in at least one transcript, which has been confirmed by a 3GS reads to be in a single transcript. The drawback of this solution is that we cannot guarantee that the predicted transcripts are consistent with all confirmed fragments, although they might fulfill all the pairwise segment occurrences.

A more accurate but also more expensive approach is to fix partial transcripts, fully confirmed by 3GS reads, in the transcript matrix and let *MiTie* extend those transcripts to full length. The penalty of those partial transcripts will be set lower than for entirely newly predicted transcripts, to favor the selection of the fragments. This strategy becomes computational costly, as soon as a large number of incompatible or non-overlapping 3GS reads is available for a given gene. Moreover, non-overlapping fragments, that originate from the same transcripts are not ideal handled, because it might turn out to be favourable to use them in different transcripts to avoid the expensive l_0 regularization of completely new transcripts. This undermines the idea of the l_0 regularization.

This could again be overcome by a third approach to integrate 3GS reads. Here, we would keep a list of 3GS fragments during the optimization and introduce binary variables if one fragment has been explained by at least one transcript. This can be accomplished by a single constraint per fragment. The l_0 regularization would then favour solutions where many compatible fragments are explained by a single transcript. Regarding the computational costs of this solution, we expect positive effects due to the reduction in search space by induction of long range dependencies as well as negative effects due to the additional binary variables. Therefore, the additional computational cost of this approach will have to be determined empirically.

Interpretation of Results Even if the prediction of a given transcript is accurate, it remains unclear whether the transcript has a biological function or appears as a result of inaccurate splicing regulation. We have hypothesized [32] that the degradation of non functional transcripts known as *nonsense mediated decay* plays a major role in shaping the transcriptome distribution of the model plant *A. thaliana*. The

underlying assumption is that the splicing process is stochastic and splicing regulators influence the outcome not in a deterministic fashion, but rather change the odds of splicing event. This results in a fraction of transcripts that is degraded shortly after biosynthesis. Very sensitive transcript prediction approaches might identify those biologically irrelevant transcripts and there is no direct way to avoid this based on expression measurements only.

But the question of biological function can be answered by identifying phenotypic consequences upon removal of a transcript or the encoded protein from a given cell, tissue or individual. Various low and high throughput techniques exist for this purpose. RNA interference screens for example deplete a given transcript from a set of cells and observe phenotypic changes like growth or drug sensitivity. In addition association of varying transcript expression with phenotypic changes can provide hints to biological function.

In summary, we note that many biological and biomedical studies critically depend on the accurate characterization of RNA compositions of biological samples. However, the RGASP competition made apparent that gene finding systems using RNA-Seq data as well as purely RNA-Seq-based methods have clear limitations in terms of prediction accuracy. At the time of the RGASP competition, gene finding systems outperformed pure RNA-Seq-based systems on average across the genome, mainly because the performance of gene finding systems depends less on the expression level of genes. Interestingly, the performance gap is smallest for the human genome, although we observe the largest fraction of uncovered genes for mammalian genomes (cf. Figure 5.7). As discussed in Section 5.5.3, we can attribute this to the increased difficulty of *ab initio* gene prediction in mammals. Since the RGASP competition technological improvements (resulting in longer reads, less severe biases and higher coverage) and methodological improvements, like the ones implemented in *MiTie*, have substantially improved the performance of RNA-Seq-based methods, while we expect gains of gene finding systems (still mainly relying on the genomic DNA sequence) to be milder.

Therefore, we perceive the field to be shifting more towards purely RNA-Seq-based methods, where we can soon expect even further improvements as discussed previously in this section. The development of MiTie contributed to this positive trend in two ways. First, by being applied in biological studies and second, we observe that ideas implemented in *MiTie* fertilize other method developments in the field.

Nevertheless, we do not expect that technological and methodological improvements will result in accurate predictions for all transcripts purely based on RNA-Seq data. For a considerable fraction of transcripts, we still have to rely on predictions from gene finding systems, in particular, in cases where a gene is only expressed

under specific environmental conditions. *mGene* has shown very favourable results using genome and RNA-Seq-based features and we reckon that the flexibility of *mGene* to take many types of high-throughput measurements simultaneously into account will prove invaluable in future, with rapidly increasing amounts and types of measurements becoming available.

Concluding, we think that the developments of this thesis contributed important ideas to the transcriptome bioinformatics community and lead to more accurate transcriptome reconstructions. We thereby facilitate a deeper understanding of RNA regulatory mechanisms and further efforts to associate genotype and phenotype.

Appendices

A. mGene.ngs: Supplemental Information

A.1. Availability and Documentation

mGene is available for download from `www.mGene.org`. The standalone version contains several introductory command line scripts on small sized sample data guiding the user through the most common use-cases.

A.2. Genomic Signal Prediction

mGene.ngs identifies several types of gene features based on the genomic DNA sequence content. In the following we describe how these signals are trained in practice.

Label Generation

The first step of creating genome-wide signal predictions is to create a training set for each signal type. These training examples are then used to train binary classifiers.

If a genome annotation is available for training we compile training labels for all signals directly from the genome annotation. In general we use the whole annotation for signal training in a five fold cross-validation scheme to obtain unbiased predictions.

In the case of full *de novo* predictions, we generate training examples directly from RNA-Seq alignments if possible, or from transcripts predicted by *Transcript Skimmer*. In the following we describe the details of this process.

Splice Sites We observed previously [108] that computational splice site predictions greatly benefit from large training sets. We therefore decided to generate splice site labels directly from RNA-Seq alignments, instead of using only splice sites being part of *Transcript Skimmer* predictions. We extract candidate splice sites from splices alignments and select a confident subset using several filter criteria. We then extract negative training examples from potentially exonic and intronic portions of RNA-Seq alignments. For each alignment satisfying certain quality criteria, we extract all consensus positions from the first to the last alignment position. We then

exclude the intersection of candidate positive and negatively labeled positions from both sets and use the remaining positions for training.

Transcription and Translation Start and Termination We obtain training examples for TIS and TTS sites by computing maximal open reading frames for transcripts predicted by transcript skimmer. We filter out cases where the maximal open reading frame is either very short or not significantly longer than the second longest open reading frame. Transcripts passing this filter are also used to generate labels for transcription start and termination sites.

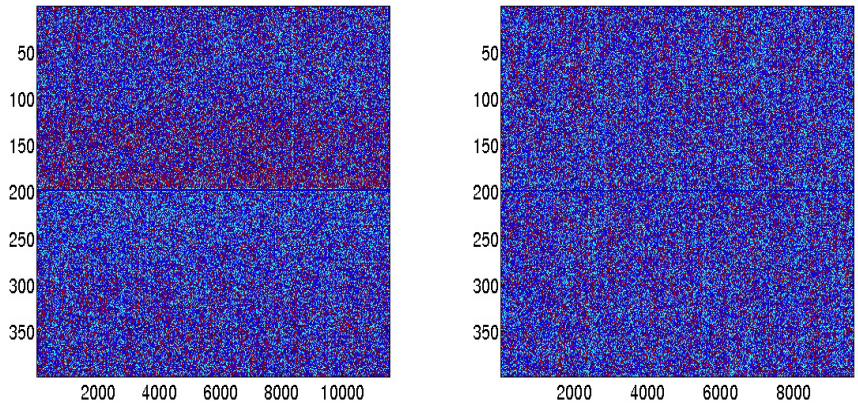


Figure A.1.: Visualization of positive (left) and negative (right) example sequences for the acceptor splice site signal. Each column corresponds to one example sequence. The nucleotides are encoded by dots of different colors. The sequences are all of length 400 and the positive as well as the negative examples exhibit the acceptor consensus 'AG' at position 199.

A.2.1. Modeling of the Signals

Splice Sites Most U2-dependent introns are bound by GU-AG and GC-AG (donor-acceptor site) splice sites, but at a very low rate also AU-AC bounded U2-dependent splice sites can be found. The terminal nucleotides of U12-type introns are AU-AC, GU-AG and few others at very low rates.
We modeled the splice sites using the canonical consensus GU-AG and GC-AG only. A window of size 141 around the intron boundaries, with 80 nucleotides inside the intron and 61 nucleotides of exonic sequence was used for training and prediction. Accurate prediction can be done with a single weighted degree kernel or a weighted degree kernel with shifts applied to the whole window, but additional spectrum kernels for intronic and exonic content prediction improve the accuracy [108].

Start and Stop Codons Our tool considers the canonical start (AUG) and stop codons (UAG, UAA, UGA) and models these sites with a spectrum kernel on each side of the consensus and a weighted degree kernel with shifts from -30 to 110 relative to the consensus position. The spectrum kernels reach from -200 to 0 and from 0 to 180. The spectrum kernels are both weighted by 0.5 and the WD-kernel by 1.0.

Transcription Start Sites RNA polymerase II binds to the promoter region upstream of 5'-UTR and begins transcription not necessarily at a specific place but rather within a range of about 40 nucleotides. Therefore, no specific consensus site exists and often no single TSS can be pinned down [107]. We model the TSS sites with a spectrum kernel from -600 to 100, a weighted degree kernel with shift from -70 to 70 and another spectrum kernel from 0 to 900. The kernels are weighted equally. We note that explicit modeling of specific elements of the promoter region like the TATA-box and the cap-site is not necessary in this approach because the spectrum kernels will catch such elements independently of their position and the SVMs will regard them if they are discriminative.

Cleavage Sites The cleavage of an mRNA does not accurately occur at a specific place [79], but usually 13 to 30 nucleotides downstream of a poly-A site. Thus like for TSS sites, no strong consensus can be found. The cleavage site is modeled with two spectrum kernels with windows from -400 to 100 and 0 to 800, respectively. A WDS-kernel is used from -60 to 60. The kernels are weighted equally.

A.2.2. Estimation of posterior probabilities

In order to provide an interpretable and comparable confidence score of the SVM predictions, we estimated the conditional likelihood $P(y = 1|f(\boldsymbol{x}))$ of the true label y being positive for a given SVM output value $f(\boldsymbol{x})$. To do this, we applied a piecewise linear function which was determined on the validation set (the same used for the classifier model selection). We used the $N = 50$ quantiles taken on the SVM output values as supporting points ϕ_i, $i = 1, \ldots, N$. For convenience, denote $\phi_0 = -\infty$. For each point ϕ_i the corresponding $\hat{\pi}_i$-value, which represents the empirical probability of being a true positive, was computed as $\hat{\pi}_i = \frac{n_i^{TP}}{n_i}$, where n_i $(i = 1, \ldots, N)$ is the number of examples with output values $\phi_{i-1} \leq f(\boldsymbol{x}) < \phi_i$ and n_i^{TP} is the number of true splice sites in the same output range. Additionally, we determined the empirical cumulative probability as follows $\hat{\pi}_i^c = \left(\sum_{j=i}^{N} n_j^{TP}\right) / \left(\sum_{j=i}^{N} n_j\right)$. In order to obtain a smooth and strictly monotonically increasing probability estimate, we solve the

following quadratic optimization problem:

$$\begin{aligned}\min_{\boldsymbol{\pi},\boldsymbol{\pi}^c\in\mathbb{R}_+^N} \quad & \sum_{i=1}^{N}(s_i(\boldsymbol{\pi})+t_i(\boldsymbol{\pi}^c)) \\ \text{s.t.} \quad & \pi_i \le \pi_i^c, \text{for all } i=1,\dots,N \\ & \pi_i \le \pi_{i+1}-\epsilon, \text{for all } i=1,\dots,N-1 \\ & \pi_i^c \le \pi_{i+1}^c-\epsilon, \text{for all } i=1,\dots,N-1,\end{aligned}$$

where $\epsilon = 10^{-4}$ is a small constant ensuring that the functions are *strictly* monotonically increasing and $s_i(\boldsymbol{\pi}) = \frac{n_i}{\sum_{j=1}^N n_j}(\pi_i - \hat{\pi}_i)^2$ and $t_i(\boldsymbol{\pi}^c) = \frac{\sum_{j=i}^N n_j}{\sum_{j=1}^N n_j}(\pi_i^c - \hat{\pi}_i^c)^2$ ensuring that big differences between the final and empirical estimates in ranges with many outputs are penalized stronger. Using the newly computed values $\pi_1,\dots,\pi_N$, we can compute for any output value $f(\boldsymbol{x})$ the corresponding posterior probability estimate $P(y=1|f(\boldsymbol{x}))$ by linear interpolation

$$P(y=1|f(\boldsymbol{x})) \approx \begin{cases} \pi_1 & \text{for } f(\boldsymbol{x}) < \phi_1 \\ r(\phi_i,\phi_{i+1}) & \text{for } \phi_i \le f(\boldsymbol{x}) < \phi_{i+1} \\ \pi_N & \text{for } f(\boldsymbol{x}) \ge \phi_N \end{cases},$$

where $r(\phi_i,\phi_{i+1}) = \frac{\pi_{i+1}(f(\boldsymbol{x})-\phi_i)+\pi_i(\phi_{i+1}-f(\boldsymbol{x}))}{\phi_{i+1}-\phi_i}$. The cumulative posterior probability $P^c(y=1|f(\boldsymbol{x}))$ is computed analogously. The above estimation procedure was performed separately for every classifier.

Performance Measurements

In hypothesis testing two kinds of errors are known. A type I error is to reject the null hypothesis (in our case: the position is a decoy site) if the null hypothesis is in fact true. A type II error occurs if the alternative hypothesis (the position is a true signal site) is rejected, when it is true. Positions that underlie the type I error are called false positives (FP=#false positives), whereas those underlying the type II error are referred to as false negatives (FN=#false negatives). Correctly classified examples are named equivalently as true positives (TP = #true positives) and true negatives (TN = #true negatives).

The true positive rate (TPR, =sensitivity, =recall) is defined as $\frac{TP}{TP+FN}$. The false positive rate (FPR) is defined as $\frac{FP}{FP+TN}$. The specificity is defined by $1-FPR$. Setting the threshold for classification to different values leads to different (TPR, FPR) pairs. Plotting these values against each other leads to the receiver operating characteristic (ROC).

The precision is defined as $1-FDR = \frac{FP}{FP+TP}$. Plotting precision against sensitivity (recall) leads to the precision-recall curve (PRC). The areas under these

curves (auROC, auPRC) are derived by linear interpolation serve as single-number performance measurements.

A.3. RGASP

Table A.1.: Overview of rGASP data sets. Data sets marked with P2 were provided in the second round of the rGASP competition. Sequencing parameters: single-end (s), paired-end (p), strand-specific (ss) read length in base pairs (bp)

Tissue	**RNA extraction**	**Sequencing technology**	**Sequencing parameters**
Homo sapiens			
Cell line K562	total RNA	Illumina	s-33bp
Cell line K562	total RNA	Illumina	p-75bp
Cell line GM12878	total RNA	Illumina	s-33bp
Cell line GM12878	total RNA	Illumina	p-75bp
Cell line K562	cytosolic RNA	Illumina	s-ss-36bp
Cell line K562	cytosolic RNA	SOLiD	36bp
Cell line GM12878	cytosolic RNA	SOLiD	36bp
Cell line K562	cytosolic RNA	Helicos	31bp
Cell line HepG2 (P2)	total RNA	Illumina	p-75bp
Drosophila melanogaster			
Cell line S2-DRSC	total RNA	Illumina	p-37bp
Cell line S2-DRSC	total RNA	Illumina	s-75bp
Cell line CME-W1-CI	total RNA	Illumina	p-37bp
Cell line Kc167	total RNA	Illumina	p-37bp
Cell line Kc167	total RNA	Illumina	s-36bp
Cell line ML-DmBG3-c2	total RNA	Illumina	p-37bp
L3 stage larvae (P2)	total RNA	Illumina	p-75bp
Caenorhabditis elegans			
Early embryo	total RNA	Illumina	s-36bp
Late embryo	total RNA	Illumina	s-36bp
mid-L1	total RNA	Illumina	p-36bp
mid-L2	total RNA	Illumina	s-36bp
mid-L3	total RNA	Illumina	s-36bp
mid-L4	total RNA	Illumina	s-36bp
Young adult	total RNA	Illumina	s-36bp
L3 live stage (P2)	total RNA	Illumina	p-76bp

B. MITIE: Supplemental Methods

B.1. Availability and Documentation

The *MITIE* code including documented scripts on sample data are available from `git@github.com:ratschlab/MiTie.git`. Additional information is available from `www.bioweb.me/mitie`.

B.2. Read simulation

The reads were simulated using the flux simulator [41]. The gene expression values c_g (sum over transcript mRNA copy numbers) for gene g was given by

$$c_g = \sqrt{\frac{m}{x_g}} \times e^{-\frac{x_g}{x_1} - \frac{x_g}{x_1}^2}$$

Where $m = 10^8$, $x_1 = 2,000$ and x_g is randomly chosen without replacement from the natural numbers from 1 to x_1. Therefore, c_g values range from 60 to $9,980$ with an average of 567. Gene expression values were then distributed over the transcripts based on a stick breaking process resulting in an exponential decay of the number of reads per transcript. Read length was set to 75bp, library preparation simulation parameters were chosen to be "random priming" and "chemical fragmentation".

We simulated sequencing errors by estimating an error model based on an Illumina sequencing run [HepG2 Encode cell lines, 36] The error model computes a mutation probability based on read quality scores, while error positions are assumed to be independent. A set of read quality strings was sampled from the same Illumina run and randomly assigned to a read. Thus, we obtain a read error distribution similar to a given Illumina run. This strategy is implemented in the *PALMapper* package [27, 55].

B.3. Read Alignment

Reads were aligned against the human reference genome *hg19* using *PALMapper* [55]. We performed very sensitive alignments allowing up to 10 mismatches and 2 gaps

(but at most 10 edit operations in total). Splice site predictions were made with the *mGene toolbox* [92, 100, 108]. Reads for which no full length match could be found were trimmed in steps of 5 nt until they could be matched or a minimum length of 40 nt was reached. Junction remapping was allowed with a junction coverage greater than 2. All further options are summarized below:

```
-l 10 -L 20 -K 12 -C 30 -I 200000 -NI 2 -SA 100 -CT 50 -a -S
-seed-hit-cancel-threshold 1000 -report-splice-sites 0.95
-filter-splice-region 5 -qpalma-use-map-max-len 2000
-qpalma-prb-offset-fix -min-spliced-segment-len 8
-report-splice-sites-top-perc 0.01
```

B.4. Splicing Graph Generation

We generate the splicing graphs in four major steps from aligned RNA-Seq reads:

1. *Genomic region identification:* For genes where at least one transcript was known we used genomic regions starting 50000 bases upstream of the transcript start to 50000 based downstream of the transcript end to account for potentially significantly longer transcript. We then trimmed the regions if we found a gap in read coverage of more than 100 bases that was also not overlapped by spliced reads. Other parts of the genome were segmented into regions based on a map adding per position coverage, number of spliced reads spanning a position and number of read-pairs spanning a position. Whenever the value of this map exceeds a user defined threshold (default 2) we call a region. We join neighboring regions with a distance of less than 50 bases and finally, we discard regions with fewer than a user defined number of reads (default 50).

2. *Segment identification:* Given a genomic region, we construct splicing graphs by generating a list of segment boundaries. Boundaries are either splice sites (SS), potential transcription start sites (TSS) and termination sites (TTS). Potential SS positions can originate from spliced reads or annotated transcripts. Analogously, TSS and TTS sites can stem from annotated transcripts or from potential transcript start and end positions inferred from RNA-Seq coverage. We identify possible start and end positions as a) drop of the read coverage to zero or b) steep drops in read coverage. The latter we find by applying a statistical test as follows. For each segment, we use a sliding window of length 60 and compare the number of read starts (ends) in the first half of the window to the corresponding number in the second half of the window in case of TSS

(TTS). We apply a binomial test on the obtained counts and call a TSS/TTS site, if the p-value is smaller than 10^{-4}.

3. *Exon identification:* We keep segments that a) have more than 5% of their nucleotides covered, b) are part of annotated transcripts, or c) if the removal of segment s does not leave any path between two segments connected by paired-end reads (if available).

4. *Intron identification:* We connect segments based on spliced reads and annotated introns.

See Figure 6.4 for more details.

B.5. Number of Paths in Human Segment Graph

We merged the four human genome annotations [Ensembl, HAVANNA, ENCODE, Vega, see 22, 36, 37, 44] by concatenating the transcripts and generated a splicing graph. We then extended the splicing graph by adding evidence from two RNA-Seq libraries for cell lines HepG2 (wgEncodeCshlLongRnaSeqHepg2CellLongnonpolyaAlnRep2.bam) and K562 (wgEncodeCshlLongRnaSeqK562CellPapAlnRep1.bam) from `http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeCshlLongRnaSeq/` [36].

B.6. Transcript identification with exact information

The regions in Figure 6.7A summarize genomic positions sharing the same composition of overlapping transcripts. If we assume to know the number of expressed transcripts in advance (3 transcripts in this case), then we know that at least five of the unknowns are equal to zero. If we randomly select 5 unknowns and set them to zero, then this results in a system of four equations and three unknowns which is either infeasible or has exactly one solution. If it is infeasible, we know that the three remaining transcripts cannot explain the read coverage. Otherwise, we found one possible solution. We iterate this by setting all possible permutations of transcripts to zero and count the number of cases the corresponding system of equations has a solution.

B.7. Transcript prediction with *Cufflinks*

Model selection for MITIE optimized the F-score on transcript level. Doing the same for *Cufflinks* results in sub-optimal predictions when samples are merged with *Cuffmerge*. Thus, the main text shows the *Cufflinks+Cuffmerge* combination, where

the mean of sensitivity and specificity was optimized for *Cufflinks*. Figure B.1 shows the sensitivity and specificity for all predictions also shown in Figure 6.10A and in addition the result for the F-score optimized version. We note that if we optimize the same criterion for *Cufflinks* and *MiTie* then *MiTie* predictions significantly outperform *Cufflinks* predictions in sensitivity as well as in specificity. Furthermore, we note that ranking methods based on the mean of sensitivity and specificity favors trivial and meaningless solutions. One example for such a trivial solution is to select only a single high confidence transcript prediction for the whole genome which if correct results in an mean(SN, SP) of 50%, but a F-score close to zero. For each of the eight optimized *Cufflinks* parameters we tried seven values. Thus we performed 56 predictions for each optimization criterion (112 in total). Tested values were equally distributed in log-space from default value divided by five to default value times five. All optimized parameters are shown in Table B.1

Table B.1.: Optimized *Cufflinks* parameters

Parameter name	Default value	Optimized F-score	Optimized mean(SN, SP)
–min-isoform-fraction	0.1	0.1	0.29
–pre-mrna-fraction	0.15	0.26	0.26
–junc-alpha	0.001	0.001	0.001
–small-anchor-fraction	0.09	0.09	0.09
–min-frags-per-transfrag	10	10	50
–overhang-tolerance	8	8	8
–trim-3-avgcov-thresh	10	10	10
–trim-3-dropoff-frac	0.1	0.1	0.1
Cuffmerge:			
–min-isoform-fraction	0.25	0.25	0.25

We combined the different cufflinks predictions using *Cuffmerge* and also optimized the hyper-parameter "–min-isoform-fraction" of the *Cuffmerge* tool. Sensitivity and specificity of the predictions are shown in (Suppl. Figure B.1).

For the *D. melanogaster* data set we performed *Cufflinks* predictions with default parameters. The parameter values optimized for the human artificial data are not likely to perform better in this setting, since data as well as the evaluation criterion have changed. Performing the same model selection on genome wide data was computationally prohibitive.

B.8. Transcript prediction based on *Trinity* graphs

We parsed the graphs and read counts for all components from files "comp*.out" and collapsed linear portions of the graph. We then resolved cycles by removing

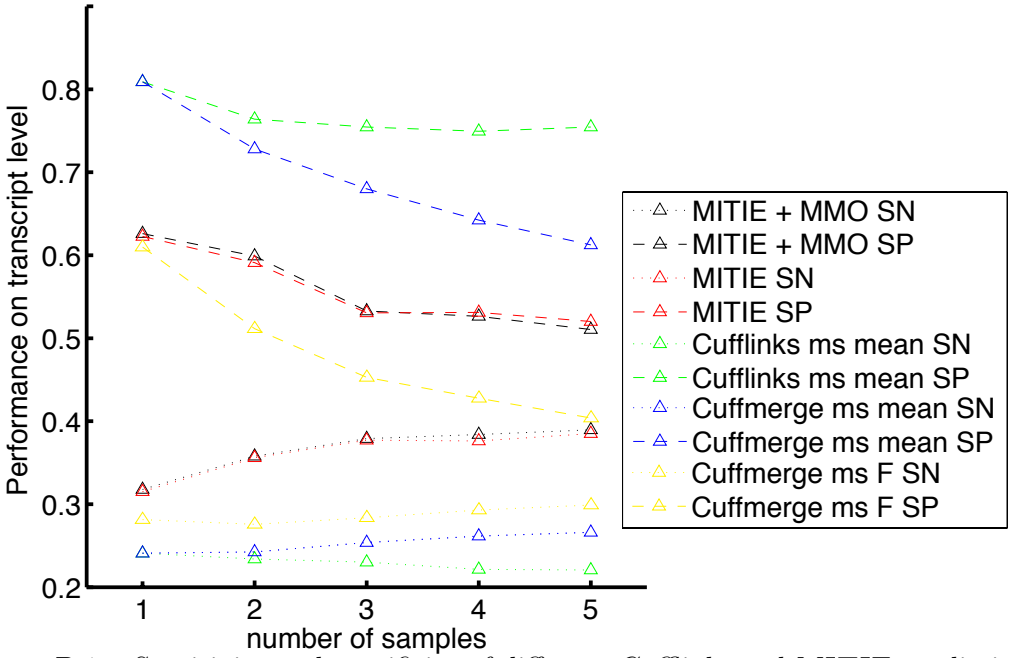


Figure B.1.: Sensitivity and specificity of different *Cufflinks* and MITIE predictions.

self loops and cutting each larger cycle at the first node on the path from an initial segment which is also part of the cycle. We then defined a total order of nodes and ran the *MiTie* optimization. For simple cases with less than 9 paths we did not run *MiTie*, but reported all possible paths instead.

B.9. Model selection

Following ideas from [104], we used Gaussian Processes (GP) to find optimal hyper-parameters for *MiTie* on the respective training sets. We employed the GP implementation provided by [90]. We trained a GP to predict the performance of our algorithm by randomly choosing initial parameter vectors and 20 training examples from the training set. As target values for the GP, we chose the F-score on transcript level. In each iteration, we randomly sampled parameter vectors and selected the one for evaluation. As selection criterion, we used the maximal upper confidence bound *ucb*:

$$ucb = \mu_y + \gamma\sigma_y$$

Where μ_y and σ_y are the mean and standard deviation of the predictive distribution of the GP. γ is a hyper-parameter of the model selection strategy and was chosen to be 2. We iterated until we had 100 data points. Finally, we selected the parameter combination with maximal lower confidence bound (*lbc*) to predict on the test set.

The *lbs* can be computed as:

$$lcb = \mu_y - \gamma\sigma_y$$

The rationale behind this strategy is to explore the space and choose new parameter vectors that might lead to good performance (vectors with high predicted mean and high variance), but finally to select a vector with a high mean and low variance to achieve good performance with high confidence. We optimized the regularization parameters for (1) the number of transcripts (2) the intron coverage fit, (3) the paired-end penalty term and (4) the exonic segment read count fit. We did not include parameters η_1 and η_2 into the model selection. We found that choices of 1.2 and 0.0 worked consistently well in all experiments.

B.10. Transcript Evaluation

We evaluate against a set of annotated genes by first finding all transcripts overlapping with a given gene. We then compute a binary match score for each pair of transcripts according to the criteria described in Section 6.5.2. We then computed a maximal matching to find pairs of annotated and predicted transcripts such that the total number of correct matches is maximal. The number of matching pairs is taken as the number of true positive predictions, when computing sensitivity and specificity of the methods. Clearly, no annotated and no predicted transcript is allowed to be part of more than one such pair. All predicted transcripts not being part of a matching pair are counted as false positives and all annotated transcripts not being part of a matching pair are counted as false negatives.

C. Learning Theory of Support Vector Machines

In this section, we recapitulate learning theoretic results for SVMs. In the following, we show that a linear separator in the M dimensional space has VC-dimension $h = M + 1$ [cf. 80].

VC-dimension of Linear Separators in $\mathbb{R}^M$ First we proof a lower bound $h \geq M + 1$ and finally we proof the upper bound $h < M + 2$. We choose the $M + 1$ data points as: $x_{M+1} = \{0\}^M$ and for $i = 1, ..., M$ x_i is the point where all but the ith coordinate are zero and the ith coordinate is equal to one. Given a labeling $y_1, ..., y_{M+1}$ we choose the parameters as $\mathbf{w}_i = y_i \forall i = 1, ..., M$ and $b = 0.5 \cdot y_{M+1}$. Substituting into 4.27 we obtain a misclassification rate equal to zero.

Using Radon's theorem [130] we can proof that the VC-dimension of a linear separator in an M-dimensional space is less than $M+2$. Given a set of $M+2$ points $x_1, ..., x_{M+2} \in \mathbb{R}^M$ we can find multipliers $a_1, ..., a_{M+2} \in \mathbb{R}$ such that:

$$\sum_{i=0}^{M+2} a_i \cdot x_i = \mathbf{0} \tag{C.1}$$

$$\sum_{i=0}^{M+2} a_i = 0 \tag{C.2}$$

We solve this system of $M + 1$ equations and $M + 2$ variables and fix any solution where not all multipliers are zero. This ensures that there is a non empty set of positive multipliers I and a non empty set of negative multipliers J. We can split

the sum in equation C.1:

$$\sum_{i=0}^{M+2} a_i \cdot x_i = \mathbf{0} \tag{C.3}$$

$$\sum_{i \in I} a_i \cdot x_i + \sum_{j \in J} a_j \cdot x_j = \mathbf{0} \tag{C.4}$$

$$\sum_{i \in I} a_i \cdot x_i = \sum_{j \in J} -a_j \cdot x_j \tag{C.5}$$

$$\sum_{i \in I} \frac{a_i}{\sum_{i \in I} a_i} \cdot x_i = \sum_{j \in J} \frac{-a_j}{\sum_{j \in J} a_j} \cdot x_j \tag{C.6}$$

Equation C.6 proofs the existence of a point that is element of the convex hull of both sets I and J and therefore the sets cannot be separated linearly. Thus, for any set of $M+2$ points we have found a labeling of the points corresponding to the sign of the multipliers such that the points are not linearly separable. q.e.d.

Substituting $h = M + 1$ into 4.24 leads to poor generalization bounds especially when we apply kernel functions with very large or even infinite dimensional feature spaces. In the following section, we will discuss results for significantly better generalization bounds. By taking into account that the SVM does not choose any separating hyperplane, but the one with maximal margin and allows misclassification of training data (even if the data is separable) significantly better generalization bounds have been proven.

VC Dimension of Support Vector Machines Vapnik [124] showed that a hyperplane separating classes with margin ρ has a VC dimension of:

$$h = min(\frac{D^2}{\rho^2}, M) + 1 \tag{C.7}$$

Where D is the diameter of the smallest sphere in feature space that encloses all training examples. With $\rho = \frac{2}{\|w\|}$ this result justifies the SVM regularizer minimizing the l_2 norm of w. This result has strong implications when considering kernels associated with very high dimensional, but sparse feature spaces, like string kernels. Considering the weighted degree kernel with degree K on strings of length L on alphabet Σ. The dimensionality of the feature space is in $\sum_{k=1}^{K}(L-k) \cdot |\Sigma|^k$. We know that each example x has the same l_2 norm $\|x\| < K \cdot L$, which gives:

$$D < K \cdot L \ll M = \sum_{k=1}^{K}(L-k) \cdot |\Sigma|^k$$

This result can be used to proof an upper bound on the risk R [76]:

$$E[R] \leq E\left[\frac{D^2 \cdot \sum_{i=1}^{N} \zeta_i}{\rho^2 \cdot N}\right] \tag{C.8}$$

Where $\sum_{i=1}^{N} \zeta_i$ is a measure for the training error. This result could be further improved by Vapnik and Chapelle [123]:

$$E[R] \leq E\left[\frac{S \cdot max(D, \frac{1}{\sqrt{C}}) \cdot A + m}{N}\right] \tag{C.9}$$

Where S is the maximal distance of one support vector to a linear combination of all other support vectors. A is the sum over all Lagrange multipliers α_i with $\alpha_i < C$ and m is the number of Lagrange multipliers with $\alpha_i = C$. Note that for all $i = 1, ..., N$ $0 \leq \alpha_i \leq C$.

Bibliography

[1] J. Adams. Dna sequencing technologies. *Nature Education*, 1(1), 2008.

[2] M. D. Adams, J. M. Kelley, J. D. Gocayne, M. Dubnick, M. H. Polymeropoulos, H. Xiao, C. R. Merril, A. Wu, B. Olde, and R. F. Moreno. Complementary dna sequencing: expressed sequence tags and human genome project. *Science*, 252(5013):1651–1656, 1991.

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

[4] S. Anders and W. Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11(10):R106, 2010.

[5] S. Atwell, Y. S. Huang, B. J. Vilhjalmsson, G. Willems, M. Horton, Y. Li, D. Meng, A. Platt, A. M. Tarone, T. T. Hu, R. Jiang, N. W. Muliyati, X. Zhang, M. A. Amer, I. Baxter, B. Brachi, J. Chory, C. Dean, M. Debieu, J. de Meaux, J. R. Ecker, N. Faure, J. M. Kniskern, J. D. G. Jones, T. Michael, A. Nemri, F. Roux, D. E. Salt, C. Tang, M. Todesco, M. B. Traw, D. Weigel, P. Marjoram, J. O. Borevitz, J. Bergelson, and M. Nordborg. Genome-wide association study of 107 phenotypes in arabidopsis thaliana inbred lines. *Nature*, 465(7298): 627–631, 2010.

[6] 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.

[7] J. Behr. Implementation of a generic pipeling for genomic signal prediction. student research project, 2007.

[8] J. Behr, A. Kahles, Y. Zhong, V. T. Sreedharan, P. Drewe, and G. Rätsch. MITIE: Simultaneous RNA-Seq-based transcript identification and quantification in multiple samples. *Bioinformatics*, page btt442, 2013.

[9] J. Behr, G. Schweikert, and G. Rätsch. Genome annotation with structured output learning. In S. Nowozin, P. Gehler, J. Jancsary, and C. Lampert, editors, *Advanced Structured Prediction*. The MIT Press, 2014. in press.

[10] C. Bishop. *Pattern Recognition and Machine Learning.pdf*. Springer Science+Business Media, Boston, 2006.

[11] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the vapnik chervonenkis dimension. *Journal of the ACM*, 36(4):925–965, 1989.

[12] E. A. Bogdanova, I. Shagina, E. V. Barsova, I. Kelmanson, D. A. Shagin, and S. A. Lukyanov. Normalizing cDNA libraries. *Current Protocols in Molecular Biology*, pages 5–12, 2010.

[13] R. Bohnert. *Computational Methods for High-Throughput Genomics and Transcriptomics*. PhD thesis, Eberhard Karls Universität Tübingen, 2011.

[14] R. Bohnert, J. Behr, and G. Rätsch. Transcript quantification with RNA-Seq data. *BMC Bioinformatics*, 10(Suppl 13):P5+, 2009.

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

[16] R. K. Bradley, J. Merkin, N. J. Lambert, and C. B. Burge. Alternative Splicing of RNA Triplets Is Often Regulated and Accelerates Proteome Evolution. *PLoS Biol*, 10(1): e1001229+, 2012.

[17] J. M. Buhmann, M. H. Chehreghani, M. Frank, and A. P. Streich. Information theoretic model selection for pattern analysis. In *ICML Workshop on Unsupervised and Transfer Learning*, 2011.

[18] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268:78–94, 1997.

[19] M. Carneiro, C. Russ, M. Ross, S. Gabriel, C. Nusbaum, and M. DePristo. Pacific biosciences sequencing technology for genotyping and variation discovery in human data. *BMC Genomics*, 13(1):375, 2012.

[20] S. Celniker, L. Dillon, M. Gerstein, K. Gunsalus, S. Henikoff, G. Karpen, M. Kellis, E. Lai, J. Lieb, D. MacAlpine, G. Micklem, F. Piano, M. Snyder, L. Stein, K. White, R. Waterston, and modENCODE Consortium. Unlocking the secrets of the genome. *Nature*, 459(7249): 927–30, 2009.

[21] A. Chenchik, L. Diachenko, F. Moqadam, V. Tarabykin, S. Lukyanov, and P. D. Siebert. Full-length cdna cloning and determination of mrna 5' and 3' ends by amplification of adaptor-ligated cdna. *Biotechniques*, 21(3):526–534, 1996.

[22] A. J. Coffey, F. Kokocinski, M. S. Calafato, C. E. Scott, P. Palta, E. Drury, C. J. Joyce, E. M. Leproust, J. Harrow, S. Hunt, A.-E. Lehesjoki, D. J. Turner, T. J. Hubbard, and A. Palotie. The gencode exome: sequencing the complete human exome. *Eur J Hum Genet*, 19(7):827–31, 2011.

[23] 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.

[24] H. P. Crowder, E. L. Johnson, and M. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31:803–834, 1983.

[25] A. K. Daly. Genome-wide association studies in pharmacogenomics. *Nat Rev Genet*, 11(4): 241–246, 2010.

[26] A. K. Daly. Using genome-wide association studies to identify genes important in serious adverse drug reactions. *Annu Rev Pharmacol Toxicol*, 52:21–35, 2012.

[27] F. De Bona, S. Ossowski, K. Schneeberger, and G. Rätsch. Optimal spliced alignments of short sequence reads. *Bioinformatics*, 24(16):i174–80, 2008.

[28] C. DeLisi. The human genome project: The ambitious proposal to map and decipher the complete sequence of human DNA. *American Scientist*, 76(5):pp. 488–493, 1988.

[29] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Stat. Soc. B*, 39:1–38, 1977.

[30] A. Dobin, C. A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T. R. Gingeras. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 2012.

[31] V. A. Doronina and J. D. Brown. Non-canonical decoding events at stop codons in eukaryotes. *Molecular Biology*, 40(4):645–663, 2006.

[32] G. Drechsel, A. Kahles, A. K. Kesarwani, E. Stauffer, J. Behr, P. Drewe, G. Rätsch, and A. Wachter. Nonsense-mediated decay of alternative precursor mRNA splicing variants is a major determinant of the arabidopsis steady state transcriptome. *The Plant Cell Online*, pages tpc–113, 2013.

[33] P. Drewe, O. Stegle, L. Hartmann, A. Kahles, R. Bohnert, A. Wachter, K. Borgwardt, and G. Rätsch. Accurate detection of differential RNA processing. *Nucleic acids research*, 41 (10):5189–5198, 2013.

[34] R. Dulbecco. A turning point in cancer research: sequencing the human genome. *Science*, 231(4742):1055–1056, 1986.

[35] J. Eichner, G. Zeller, S. Laubinger, and G. Ratsch. Support vector machines-based identification of alternative splicing in arabidopsis thaliana from whole-genome tiling arrays. *BMC Bioinformatics*, 12(1):55, 2011.

[36] ENCODE Project Consortium, B. E. Bernstein, E. Birney, I. Dunham, E. D. Green, C. Gunter, and M. Snyder. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57–74, 2012.

[37] P. Flicek, M. R. Amode, D. Barrell, K. Beal, S. Brent, D. Carvalho-Silva, P. Clapham, G. Coates, S. Fairley, S. Fitzgerald, L. Gil, L. Gordon, M. Hendrix, T. Hourlier, N. Johnson, A. K. Kähäri, D. Keefe, S. Keenan, R. Kinsella, M. Komorowska, G. Koscielny, E. Kulesha, P. Larsson, I. Longden, W. McLaren, M. Muffato, B. Overduin, M. Pignatelli, B. Pritchard, H. S. Riat, G. R. S. Ritchie, M. Ruffier, M. Schuster, D. Sobral, Y. A. Tang, K. Taylor, S. Trevanion, J. Vandrovcova, S. White, M. Wilson, S. P. Wilder, B. L. Aken, E. Birney, F. Cunningham, I. Dunham, R. Durbin, X. M. Fernández-Suarez, J. Harrow, J. Herrero, T. J. P. Hubbard, A. Parker, G. Proctor, G. Spudich, J. Vogel, A. Yates, A. Zadissa, and S. M. J. Searle. Ensembl 2012. *Nucleic Acids Res*, 40(Database issue):D84–90, 2012.

[38] G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[39] X. Gan, O. Stegle, J. Behr, J. G. Steffen, P. Drewe, K. L. Hildebrand, R. Lyngsoe, S. J. Schultheiss, E. J. Osborne, V. T. Sreedharan, A. Kahles, R. Bohnert, G. Jean, P. Derwent, P. Kersey, E. J. Belfield, N. P. Harberd, E. Kemen, C. Toomajian, P. X. Kover, R. M. Clark, G. Rätsch, and R. Mott. Multiple reference genomes and transcriptomes for Arabidopsis thaliana. *Nature*, 477(7365):419–423, 2011.

[40] M. G. Grabherr, B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng, Z. Chen, E. Mauceli, N. Hacohen, A. Gnirke, N. Rhind, F. di Palma, B. W. Birren, C. Nusbaum, K. Lindblad-Toh, N. Friedman, and A. Regev. Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nature Biotechnology*, 29(7):644–652, 2011.

[41] T. Griebel, B. Zacher, P. Ribeca, E. Raineri, V. Lacroix, R. Guigó, and M. Sammeth. Modelling and simulating generic RNA-Seq experiments with the flux simulator. *Nucleic Acids Res*, doi:10.1093/nar/gks666, 2012.

[42] R. Guigó, P. Flicek, J. F. Abril, A. Reymond, J. Lagarde, F. Denoeud, S. Antonarakis, M. Ashburner, V. B. Bajic, E. Birney, R. Castelo, E. Eyras, C. Ucla, T. R. Gingeras, J. Harrow, T. Hubbard, S. E. Lewis, and M. G. Reese. EGASP: the human ENCODE genome annotation assessment project. *Genome Biol*, 7 Suppl 1:S2.1–S231, 2006.

[43] M. Guttman, M. Garber, J. Z. Levin, J. Donaghey, J. Robinson, X. Adiconis, L. Fan, M. J. Koziol, A. Gnirke, C. Nusbaum, J. L. Rinn, E. S. Lander, and A. Regev. Ab initio reconstruction of cell type-specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincRNAs. *Nature Biotechnology*, 28(5):503–510, 2010.

[44] J. Harrow, F. Denoeud, A. Frankish, A. Reymond, C.-K. Chen, J. Chrast, J. Lagarde, J. G. R. Gilbert, R. Storey, D. Swarbreck, C. Rossier, C. Ucla, T. Hubbard, S. E. Antonarakis, and R. Guigo. Gencode: producing a reference annotation for encode. *Genome Biol*, 7 Suppl 1: S4.1–9, 2006.

[45] S. Heber, M. Alekseyev, S.-H. Sze, H. Tang, and P. A. Pevzner. Splicing graphs and EST assembly problem. *Bioinformatics*, 18(suppl 1):S181–S188, 2002.

[46] D. Hiller and W. Wong. Simultaneous isoform discovery and quantification from RNA-Seq. *Statistics in Biosciences*, pages 1–19, 2012.

[47] D. Hiller, H. Jiang, W. Xu, and W. H. Wong. Identifiability of isoform deconvolution from junction arrays and RNA-Seq. *Bioinformatics*, 25(23):3056–9, 2009.

[48] `http://augustus.gobics.de/binaries/README.TXT`. Augustus documentation web-site, 2014.

[49] `http://en.wikipedia.org/wiki/DNA_replication`. DNA replication, 2013.

[50] `http://en.wikipedia.org/wiki/DNA_sequencing`. DNA sequencing, 2013.

[51] `http://en.wikipedia.org/wiki/Law_of_total_variance`. Law of total variance, 2013.

[52] `http://en.wikipedia.org/wiki/Negative_binomial_distribution`. Negative binomial distribution, 2013.

[53] `http://en.wikipedia.org/wiki/Single_molecule_real_time_sequencing`. Single molecule real time sequencing, 2013.

[54] `http://linux1.softberry.com/berry.phtml?topic=transomics`. Fgenesh++R - gene prediction using RNASeq data, 2013.

[55] G. Jean, A. Kahles, V. Sreedharan, F. De Bona, and G. Rätsch. RNA-Seq read alignments with PALMapper. *Curr. Protoc. Bioinform.*, 32:11.6.1–11.6.38, 2010.

[56] G. Ji, J. Zheng, Y. Shen, X. Wu, R. Jiang, Y. Lin, J. C. Loke, K. M. Davis, G. J. Reese, and Q. Q. Li. Predictive modeling of plant messenger RNA polyadenylation sites. *BMC Bioinformatics*, 8:43, 2007.

[57] H. Jiang and W. Wong. Statistical inferences for isoform expression in RNA-Seq. *Bioinformatics*, 25(8):1026–1032, 2009.

[58] W. Johannsen. *Elemente der exakten Erblichkeitslehre*. Gustav Fischer, Jena, 1909.

[59] J. J. B. Keurentjes, J. Fu, I. R. Terpstra, J. M. Garcia, G. van den Ackerveken, L. B. Snoek, A. J. M. Peeters, D. Vreugdenhil, M. Koornneef, and R. C. Jansen. Regulatory network construction in arabidopsis by using genome-wide gene expression quantitative trait loci. *Proc Natl Acad Sci U S A*, 104(5):1708–1713, 2007.

[60] J. J. Kieber. Cytokinins. *The Arabidopsis Book*, 1(e0063), 2002.

[61] C. Knepper and B. Day. From perception to activation: The molecular-genetic and biochemical landscape of disease resistance signaling in plants. *The Arabidopsis Book*, 1(e012), 2010.

[62] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. A generalized hidden markov model for the recognition of human genes in DNA. *ISMB-96*, pages 134–141, 1996.

[63] V. Lacroix, M. Sammeth, R. Guigo, and A. Bergeron. Exact transcriptome reconstruction from short sequence reads. In *Proceedings of the 8th international workshop on Algorithms in Bioinformatics*, WABI '08, pages 50–63, Berlin, Heidelberg, 2008. Springer-Verlag.

[64] 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.

[65] M. Levene, J. Korlach, S. Turner, M. Foquet, H. Craighead, and W. Webb. Zero-mode waveguides for single-molecule analysis at high concentrations. *Science*, 299(5607):682–686, 2003.

[66] B. Lewin. *Genes VII*. Oxford University Press, New York, 2000.

[67] J. D. Lewis and E. Izaurralde. The role of the cap structure in RNA processing and nuclear export. *Eur J Biochem*, 247(2):461–469, 1997.

[68] B. Li and C. Dewey. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC bioinformatics*, 12(1):323, 2011.

[69] B. Li, V. Ruotti, R. Stewart, J. Thomson, and C. Dewey. RNA-Seq gene expression estimation with read mapping uncertainty. *Bioinformatics*, 26(4):493–500, 2010.

[70] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, et al. De novo assembly of human genomes with massively parallel short read sequencing. *Genome research*, 20(2):265–272, 2010.

[71] W. Li, J. Feng, and T. Jiang. IsoLasso: A LASSO regression approach to RNA-Seq based transcriptome assembly. In V. Bafna and S. Sahinalp, editors, *Research in Computational Molecular Biology*, volume 6577 of *Lecture Notes in Computer Science*, pages 168–188. Springer Berlin Heidelberg, 2011.

[72] Y.-Y. Lin, P. Dao, F. Hach, M. Bakhshi, F. Mo, A. Lapuk, C. Collins, and S. Sahinalp. Cliiq: Accurate comparative detection and quantification of expressed isoforms in a population. In B. Raphael and J. Tang, editors, *Algorithms in Bioinformatics*, volume 7534 of *Lecture Notes in Computer Science*, pages 178–189. Springer Berlin Heidelberg, 2012.

[73] J. C. Loke, E. A. Stahlberg, D. G. Strenski, B. J. Haas, P. C. Wood, and Q. Q. Li. Compilation of mRNA polyadenylation signals in *Arabidopsis* revealed a new signal element and potential secondary structures. *Plant Physiol*, 138(3):1457–1468, 2005.

[74] G. Lunter and M. Goodson. Stampy: A statistical algorithm for sensitive and fast mapping of illumina sequence reads. *Genome research*, 21(6):936–939, 2011.

[75] A. M. Mezlini, E. J. Smith, M. Fiume, O. Buske, G. Savich, S. Shah, S. Aparicion, D. Chiang, A. Goldenberg, and M. Brudno. iReckon: Simultaneous isoform discovery and abundance estimation from RNA-seq. *Genome Research*, 2012.

[76] X. Miao and L. Liao. Vc-dimension and its applications in machine learning `http://www.liaolin.com/courses/vc-dimension.pdf`, 2003.

[77] M. Michniewicz, P. B. Brewer, and J. Friml. Polar auxin transport and asymmetric auxin distribution. *The Arabidopsis Book*, 1(e0108), 2007.

[78] J. E. Mitchell. *Branch-and-Cut Algorithms for Combinatorial Optimization Problems*. Handbook of Applied Optimization, 2002.

[79] R. R. Monarez, C. C. MacDonald, and B. Dass. Polyadenylation proteins CstF-64 and tCstF-64 exhibit differential binding affinities for RNA polymers. *Biochemestry*, pages 651–658, 2007.

[80] A. W. Moore. Vc-dimension for characterizing classifiers. `http://www.autonlab.org/tutorials/vcdim08.pdf`, 2001.

[81] E. Moriyama, P. Strope, S. Opiyo, Z. Chen, and A. Jones. Mining the arabidopsis thaliana genome for highly-divergent seven transmembrane receptors. *Genome Biology*, 7(10):R96, 2006.

[82] A. Mortazavi, B. A. Williams, K. McCue, L. Schaeffer, and B. Wold. Mapping and quantifying mammalian transcriptomes by rna-seq. *Nat Methods*, 5(7):621–8, 2008.

[83] J. Nelder and R. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society*, 1972.

[84] P. Niermann. Gene prediction with mgene on the human genome. Master's thesis, Eberhard Karls Universität Tübingen, 2011. supervised by Gunnar Rätsch.

[85] T. W. Nilsen and B. R. Graveley. Expansion of the eukaryotic proteome by alternative splicing. *Nature*, 463:457–463, 2010.

[86] U. Ohler. Identification of core promoter modules in drosophila and their application in accurate transcription start site prediction. *Nucleic Acids Res*, 34(20):5943–5950, 2006.

[87] Q. Pan, O. Shai, L. J. Lee, B. J. Frey, and B. J. Blencowe. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature genetics*, 40(12):1413–1415, 2008.

[88] P. A. Passarinho and S. C. de Vries. Arabidopsis chitinases: a genomic survey. *The Arabidopsis Book*, 1(e0023), 2002.

[89] H. Pearson. Genetics: what is a gene? *Nature*, 441(7092):398–401, 2006.

[90] C. E. Rasmusen and H. Nickisch. Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research*, 11:3011–3015, 2010.

[91] 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 19*, pages 1161–1168. MIT Press, Cambridge, MA, 2007.

[92] 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.

[93] J. D. Rioux, R. J. Xavier, K. D. Taylor, M. S. Silverberg, P. Goyette, A. Huett, T. Green, P. Kuballa, M. M. Barmada, L. W. Datta, Y. Y. Shugart, A. M. Griffiths, S. R. Targan, A. F. Ippoliti, E.-J. Bernard, L. Mei, D. L. Nicolae, M. Regueiro, L. P. Schumm, A. H. Steinhart, J. I. Rotter, R. H. Duerr, J. H. Cho, M. J. Daly, and S. R. Brant. Genome-wide association study identifies new susceptibility loci for crohn disease and implicates autophagy in disease pathogenesis. *Nat Genet*, 39(5):596–604, 2007.

[94] G. Robertson, J. Schein, R. Chiu, R. Corbett, M. Field, S. D. Jackman, K. Mungall, S. Lee, H. M. Okada, J. Q. Qian, M. Griffith, A. Raymond, N. Thiessen, T. Cezard, Y. S. Butterfield, R. Newsome, S. K. Chan, R. She, R. Varhol, B. Kamoh, A.-L. Prabhu, A. Tam, Y. Zhao, R. A. Moore, M. Hirst, M. A. Marra, S. J. M. Jones, P. A. Hoodless, and I. Birol. De novo assembly and analysis of RNA-seq data. *Nature Methods*, 7:909–912, 2010.

[95] A. Salamov and V. Solovyev. Ab initio gene finding in *Drosophila* genomic DNA. *Genome Res.*, 10(4):516–522, 2000.

[96] F. Sanger, S. Nicklen, and A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467, 1977.

[97] E. E. Schadt, S. Turner, and A. Kasarskis. A window into third-generation sequencing. *Human molecular genetics*, 19(R2):R227–R240, 2010.

[98] M. H. Schulz, D. R. Zerbino, M. Vingron, and E. Birney. Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. *Bioinformatics (Oxford, England)*, 28(8):1086–1092, 2012.

[99] G. Schweikert. *Accurate Prediction of Protein-Coding Genes with Discriminative Learning Techniques*. PhD thesis, Technischen Universität Berlin, 2010.

[100] G. Schweikert, A. Zien, G. Zeller, J. Behr, C. Dieterich, C. Ong, P. Philips, F. De Bona, L. Hartmann, A. Bohlen, et al. mGene: accurate SVM-based gene finding with an application to nematode genomes. *Genome research*, 19(11):2133–2143, 2009.

[101] N. Sheth, X. Roca, M. Hastings, T. Roeder, A. R. Krainer, and R. Sachidanandam. Comprehensive splice-site analysis using comparative genomics. *Nucleic Acids Research*, 34(14): 3955–3967, 2006.

[102] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. M. Jones, and I. Birol. ABySS: A parallel assembler for short read sequence data. *Genome Research*, 19(6):1117–1123, 2009.

[103] L. Smith, L. Hartmann, P. Drewe, R. Bohnert, A. Kahles, C. Lanz, and G. Rätsch. Multiple insert size paired-end sequencing for deconvolution of complex transcriptomes. *RNA biology*, 9, 2012.

[104] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. Technical Report arXiv:1206.2944, arXiv, 2012.

[105] M. B. Soares, M. de Fatima Bonaldo, J. D. Hackett, and D. Bhattacharya. Expressed sequence tags: Normalization and subtraction of cDNA libraries. In *Expressed Sequence Tags (ESTs)*, pages 109–123. Springer, 2009.

[106] S. Sonnenburg, G. Rätsch, and B. Schölkopf. Large scale genomic sequence SVM classifiers. *Procedings ICML'05, ACM Press*, 2005.

[107] S. Sonnenburg, A. Zien, and G. Rätsch. ARTS: accurate recognition of transcription starts in human. *Bioinformatics*, 22(14):e472–80, 2006.

[108] 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.

[109] W. C. Spencer, G. Zeller, J. D. Watson, S. R. Henz, K. L. Watkins, R. D. McWhirter, S. Petersen, V. T. Sreedharan, C. Widmer, J. Jo, et al. A spatial and temporal map of c. elegans gene expression. *Genome research*, 21(2):325–341, 2011.

[110] M. Stanke and B. Morgenstern. Augustus: a web server for gene prediction in eukaryotes that allows user-defined constraints. *Nucleic Acids Res*, 33(Web Server issue):W465–W467, 2005.

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

[112] M. Stanke, R. Steinkamp, S. Waack, and B. Morgenstern. Augustus: a web server for gene finding in eukaryotes. *Nucleic Acids Res*, 32(Web Server issue):W309–W312, 2004.

[113] M. Stanke, O. Keller, I. Gunduz, A. Hayes, S. Waack, and B. Morgenstern. Augustus: ab initio prediction of alternative transcripts. *Nucleic Acids Res*, 34(Web Server issue):W435–W439, 2006.

[114] M. Stanke, O. Schffmann, B. Morgenstern, and S. Waack. Gene prediction in eukaryotes with a generalized hidden markov model that uses hints from external sources. *BMC Bioinformatics*, 7:62, 2006.

[115] M. Stanke, M. Diekhans, R. Baertsch, and D. Haussler. Using native and syntenically mapped cdna alignments to improve de novo gene finding. *Bioinformatics*, 24(5):637–644, 2008.

[116] T. Steijger, J. Abril, P. Engström, K. F., R. Consortium, T. Hubbard, R. Guigo, J. Harrow, and P. Bertone. Assessment of transcript reconstruction methods for rna-seq. *Nature Methods*, page accepted, 2013.

[117] C. H. Teo, S. Vishwanthan, A. J. Smola, and Q. V. Le. Bundle methods for regularized risk minimization. *The Journal of Machine Learning Research*, 11:311–365, 2010.

[118] S. Tikole and R. Sankararamakrishnan. A survey of mRNA sequences with a non-AUG start codon in RefSeq database. *Journal of Biomolecular Structure and Dynamics*, 24(1):33–41, 2006.

[119] M. Todesco, S. Balasubramanian, T. T. Hu, M. B. Traw, M. Horton, P. Epple, C. Kuhns, S. Sureshkumar, C. Schwartz, C. Lanz, R. A. E. Laitinen, Y. Huang, J. Chory, V. Lipka, J. O. Borevitz, J. L. Dangl, J. Bergelson, M. Nordborg, and D. Weigel. Natural allelic variation underlying a major fitness trade-off in arabidopsis thaliana. *Nature*, 465(7298): 632–636, 2010.

[120] C. Trapnell, L. Pachter, and S. L. Salzberg. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, 25(9):1105–1111, 2009.

[121] C. Trapnell, B. A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M. J. van Baren, S. L. Salzberg, B. J. Wold, and L. Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, 2010.

[122] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6: 1453–1484, 2005.

[123] V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9):2013–2036, 2000.

[124] V. N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5), 1999.

[125] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

[126] K. Wang, D. Singh, Z. Zeng, S. J. Coleman, Y. Huang, G. L. Savich, X. He, P. Mieczkowski, S. A. Grimm, C. M. Perou, J. N. MacLeod, D. Y. Chiang, J. F. Prins, and J. Liu. MapSplice: Accurate mapping of RNA-seq reads for splice junction discovery. *Nucleic Acids Res*, 38(18): e178, 2010.

[127] L. Wang, Z. Feng, X. Wang, X. Wang, and X. Zhang. DEGseq: an R package for identifying differentially expressed genes from RNA-seq data. *Bioinformatics*, 26(1):136–138, 2010.

[128] Z. Wang and C. B. Burge. Splicing regulation: from a parts list of regulatory elements to an integrated splicing code. *RNA*, 14(5):802–813, 2008.

[129] Wikipedia. Machine learning. http://en.wikipedia.org/wiki /machine_learning, 2008.

[130] Wikipedia. Randon's theorem. http://en.wikipedia.org/wiki/radon's_theorem, 2013.

[131] T. D. Wu and S. Nacu. Fast and snp-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*, 26(7):873–81, 2010.

[132] Z. Xia, J. Wen, C. C. Chang, and X. Zhou. NSMAP: A method for spliced isoforms identification and quantification from RNA-Seq. *BMC Bioinformatics*, 12(1):162+, 2011.

[133] K. Yamada, J. Lim, J. M. Dale, H. Chen, P. Shinn, C. J. Palm, A. M. Southwick, H. C. Wu, C. Kim, M. Nguyen, P. Pham, R. Cheuk, G. Karlin-Newmann, S. X. Liu, B. Lam, H. Sakano, T. Wu, G. Yu, M. Miranda, H. L. Quach, M. Tripp, C. H. Chang, J. M. Lee, M. Toriumi, M. M. H. Chan, C. C. Tang, C. S. Onodera, J. M. Deng, K. Akiyama, Y. Ansari, T. Arakawa, J. Banh, F. Banno, L. Bowser, S. Brooks, P. Carninci, Q. Chao, N. Choy, A. Enju, A. D. Goldsmith, M. Gurjal, N. F. Hansen, Y. Hayashizaki, C. Johnson-Hopson, V. W. Hsuan, K. Iida, M. Karnes, S. Khan, E. Koesema, J. Ishida, P. X. Jiang, T. Jones, J. Kawai, A. Kamiya, C. Meyers, M. Nakajima, M. Narusaka, M. Seki, T. Sakurai, M. Satou, R. Tamse, M. Vaysberg, E. K. Wallender, C. Wong, Y. Yamamura, S. Yuan, K. Shinozaki, R. W. Davis, A. Theologis, and J. R. Ecker. Empirical analysis of transcriptional activity in the arabidopsis genome. *Science*, 302(5646):842–846, 2003.

[134] K. Yamada, J. Lim, J. M. Dale, H. Chen, P. Shinn, C. J. Palm, A. M. Southwick, H. C. Wu, C. Kim, M. Nguyen, P. Pham, R. Cheuk, G. Karlin-Newmann, S. X. Liu, B. Lam, H. Sakano, T. Wu, G. Yu, M. Miranda, H. L. Quach, M. Tripp, C. H. Chang, J. M. Lee, M. Toriumi, M. M. H. Chan, C. C. Tang, C. S. Onodera, J. M. Deng, K. Akiyama, Y. Ansari, T. Arakawa, J. Banh, F. Banno, L. Bowser, S. Brooks, P. Carninci, Q. Chao, N. Choy, A. Enju, A. D. Goldsmith, M. Gurjal, N. F. Hansen, Y. Hayashizaki, C. Johnson-Hopson, V. W. Hsuan, K. Iida, M. Karnes, S. Khan, E. Koesema, J. Ishida, P. X. Jiang, T. Jones, J. Kawai, A. Kamiya, C. Meyers, M. Nakajima, M. Narusaka, M. Seki, T. Sakurai, M. Satou, R. Tamse, M. Vaysberg, E. K. Wallender, C. Wong, Y. Yamamura, S. Yuan, K. Shinozaki, R. W. Davis, A. Theologis, and J. R. Ecker. Empirical analysis of transcriptional activity in the *Arabidopsis* genome. *Science*, 302(5646):842–846, 2003.

[135] M. Yandell and D. Ence. A beginner's guide to eukaryotic genome annotation. *Nature Reviews Genetics*, 13(5):329–342, 2012.

[136] G. Zeller, S. R. Henz, S. Laubinger, D. Weigel, and G. Rätsch. Transcript normalization and segmentation of tiling array data. In *Pacific symposium on biocomputing*, volume 13, pages 527–538, 2008.