JAVIER PÉREZ-RODRÍGUEZ

GENE PREDICTION BY USING ADVANCED TECHNIQUES OF COMPUTATIONAL INTELLIGENCE

Ph.D. Thesis

Supervised by Nicolás García-Pedrajas (Ph.D.)

University of Granada

Jan 2015

Editorial: Universidad de Granada. Tesis Doctorales
Autor: Javier Pérez-Rodríguez
ISBN: 978-84-9125-070-6
URI: http://hdl.handle.net/10481/40002

Ph.D. Thesis

Gene prediction by using advanced techniques of Computational Intelligence

Author: Javier Pérez Rodríguez

Supervisor: Nicolás García Pedrajas

MSc: Soft Computing and Intelligent Systems
Department of Computer Science and Artificial Intelligence
University of Granada

Prof. Dr. D. **Nicolás García Pedrajas**, Titular de Universidad en el Departamento de Informática y Análisis Numérico de la Universidad de Córdoba

INFORMA

que D. **Javier Pérez Rodríguez**, Ingeniero en Informática ha realizado en el Departamento de Ciencias de la Computacion e Inteligencia Artificial de la Universidad de Granada bajo su direccion el trabajo de investigacion correspondiente a la tesis doctoral titulada:

Gene prediction by using advanced techniques of Computational Intelligence

Revisado el mencionado trabajo, estima que puede ser presentado al tribunal que ha de juzgarlo y autoriza la defensa de esta Tesis Doctoral en la Universidad de Granada.

Granada, January 2, 2015

Fdo. Prof. Dr. D. Nicolás García Pedrajas.

El doctorando, **Javier Pérez Rodríguez**, y el director de la tesis, **Nicolás García Pedrajas**, garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección del director de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores al ser citados, cuando se han utilizado sus resultados o publicaciones.

Director de la Tesis	Doctorando
Fdo. D. Nicolás García Pedrajas	Fdo. D. Javier Pérez Rodríguez

Acknowledgments

I am specially grateful to my thesis supervisor, Nicolás. His help and support over these years are uncountable. He has infused me his research passion and has always uninterestingly shared his knowledge with me.

To the members of my research group, Computational Intelligence and Bioinformatics Research Group at the University of Córdoba. They are partners and friends above all. I definitely owe you for the support you have always offered me.

I want to express my gratitude to Mario Stanke and his research group, who has taught me so much during my short visit and that received me in Greifswald.

Finally, thank you so much to everyone and to so many others I left behind these short lines.

Resumen

La caída en los últimos años del coste de los procesos de secuenciación ha puesto a disposición de la comunidad investigadora los genomas de muchos organismos, convirtiéndose en necesaria la transformación de estos datos en información útil para disciplinas tales como la medicina, la biología o la agricultura. El primer paso, y uno de los más determinantes, en la interpretación de las secuencias de los genomas es la identificación y localización de los genes contenidos en ellas, así como la predicción de sus estructuras. Dicha tarea es conocida como predicción de genes, y el éxito de los pasos posteriores dentro del proceso dependen de la calidad de los resultados de esta. Sin embargo, no se trata de una tarea trivial. La dificultad de la labor queda ilustrada por el hecho de que a pesar de los grandes avances experimentados en la última década en las tecnologías aplicadas a ella, las técnicas actuales de predicción de la localización y estructura de los genes están todavía lejos de ser fiables.

Aunque inicialmente la predicción de genes se enfocó como una tarea experimental de laboratorio, la acumulación de datos hizo necesaria una metodología automatizada que integraran técnicas propias de la teoría de la información. De hecho, en la actualidad, incluso las anotaciones manuales realizadas sobre los genomas se valen de técnicas de predicción automáticas. En este momento, la tendencia es hacia el desarrollo de enfoques que integren ambas perspectivas. De este modo, el problema de la predicción de genes se puede abordar mediante el uso de técnicas de Aprendizaje Automático debido a que puede ser formulado en parte como una tarea de clasificación por un lado, y por otro como un proceso de optimización.

La predicción de genes consiste en identificar aquellas porciones de secuencias de ADN que contienen la información para codificar moléculas biológicamente funcionales, tales como proteínas u otros elementos como ncRNA. Desde un punto de vista computacional, una secuencia de ADN es una cadena sobre el alfabeto $\{A, T, G, C\}$, letras que se corresponden con las bases nitrogenadas que tienen los nucleótidos que la forman, Adenina, Timina, Guanina y Cistosina. El principal objetivo de la tarea sería asignar correctamente una etiqueta a cada uno de los elementos identificándolos como pertenecientes a una región determinada, codificante, no codificante, intergénica... Cada una de las regiones cuentan en sus extremos con un determinado marcador que las delimitan y los cuales son reconocidos por la maquinaria celular. La mayoría de las metodologías de predicción de genes cuentan con un componente que trata de identificar estos puntos frontera

denominados sitios funcionales. En este trabajo proponemos una serie de enfoques que abordan el reconocimiento de sitios funcionales mediante el uso de técnicas propias del Aprendizaje Automático y que tienen en consideración al mismo tiempo aspectos propios de la naturaleza biológica del problema. Concretamente, las metodologías presentadas para reconocimiento de puntos funcionales en la secuencia cuentan con las siguientes características:

- Desde una perspectiva pura de Aprendizaje Automático, se demuestra la naturaleza desequilibrada en las clases problema y se muestra la utilidad de aplicar métodos desarrollados para combatir esta característica. Del mismo modo, se hace un estudio que revela el beneficio de usar técnicas de selección de características en el problema en términos de mejora del rendimiento global de los clasificadores y de la interpretabilidad del problema.

- Se propone una nueva técnica para predicción de este tipo de sitios basada en la idea de que se consideren más de dos grupos en los patrones para el entrenamiento utilizados para la generación de modelos de clasificación. Esta premisa esta basada en el hecho de que los patrones presentan diferentes características biológicas con distintos niveles de importancia en el proceso de clasificación, abandonando el uso de la tradicional separación binaria positivo/negativo. Además, se considera el uso de más de un tipo de clasificador, apoyándose en la idea de que el comportamiento de un conjunto de clasificadores es mejor a uno individualmente, más si cabe con la existe diversidad en la naturaleza de los mismos.

- Por último, se introduce una nueva de metodología de clasificación basada en la creación de un modelo que combina tantos tipos de clasificadores como sea necesario y que consideran tantas fuentes de evidencia como sea posibles. Con fuentes de evidencia se refiere a la utilización de las secuencias de genomas de diferentes organismos como informantes.

Una vez identificados aquellos puntos que con mayor probabilidad representan un sitio funcional en la secuencia, el siguiente paso debe ser integrar toda la información disponible para presentar como resultado del proceso una estructura correcta de genes. Así pues, ortogonalmente a la predicción de sitios funcionales, se introduce las directrices para el diseño de un marco de trabajo general para predicción de genes que aborda el problema desde una perspectiva global. Dicho elemento considera el problema como un proceso de búsqueda en el que se integran tantas fuentes de información como sean necesarias para encontrar la estructura de genes más probables en una secuencia. El proceso de optimización se realiza a través del uso de los principios de la Computación Evolutiva. Una población de individuos, cada uno de los cuales representa una solución al problema, es utilizada para simular un proceso de evolución. En dicho proceso, una función basada en distintos sensores determina qué individuos se adaptan mejor a la solución del problema y poseen las características estadísticas reseñables propias de las regiones codificantes.

Los experimentos realizados en distintos cromosomas de varios organismos, principalmente en el genoma humano, muestran que las técnicas mencionadas son útiles a la hora de tratar el problema de la predicción de genes, mejorando los resultados de los mejores métodos actuales descritos en la bibliografía. El interés de los resultados obtenidos durante las investigaciones realizadas ha derivado en forma de presentaciones en diferentes congresos de ámbito internacional y en la publicación de varios artículos en contrastadas revistas del área de investigación.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In current molecular biology and genetics, the genome is the genetic material of an organism. It is encoded within the cells in form of chemical macromolecules called deoxyribonucleic acid (DNA). Genes are the information units within the genome that contain all the features necessary for the synthesis of the elements –including but not limited to proteins– that regulate the specific cellular functions related to the development and physiological operations of living beings.

The genomes of many organisms have been sequenced in recent years, and it has become necessary to transform these large raw datasets into useful knowledge. One of the most crucial steps in the interpretation of genome sequences is the identification of genes and the prediction of their structures. The success of all subsequent steps of biological and biomedical research that exploit these genomic sequences depends on the quality of this information. The difficulties of gene prediction can be illustrated by the fact that, despite significant improvements in gene prediction technologies, prediction of the location and the structure of genes is still far from reliable. According to current estimates, the complete structure of less than 50% of human genes is predicted correctly[30].

Gene prediction problem can be approached by using machine learning techniques because it can be formulated as a classification task plus an optimization process. In this chapter the problem of gene prediction will be presented. A quick review of the existing computer approaches will also be provided to better explain the key features of the machine learning methodology for gene prediction proposed in this thesis. Finally, the main objectives and the contribution of this dissertation will be summarized together with the organization of the thesis.

1.1 Motivation

Both, living organism features and the biological process produced within them, are determined by the chemical macromolecules – mainly proteins, but also non coding RNA[1] – that they synthesize. The instructions to complete the synthesis of these functional elements are contained in the cells, specifically in the DNA. The DNA segments that contain the required information to codify such functional elements are called genes. The term *genome* classically refers to the complete set of genes that are encoded by the sequence of nucleotides in a particular organelle. In the modern era, the term may refer to the sequence itself. In recent times, because the cost of sequencing the genome of an organism has decreased dramatically, an increase of the amount of data produced by genomic sequencers has driven an increase in the effort required for genome data analysis.

The term *genome annotation* denotes a two step procedure. First, *structural annotation*, the process of identifying genes, their locations in the sequence and their exon/intron structures. Second, *functional annotation* is the process of attaching biological information to these elements resulting in structural annotation. Thus, the aim of genome annotation is to determine the biochemical and biological function, if any, of each nucleotide in a genome. For this reason, genome annotation has become one of the most challenging tasks in bioinformatics.

Therefore, determining the complete set of genes in the whole genome of an organism is a crucial task for understanding the biological information held in such sequences. However, many aspects remain generating controversy and make this task non trivial. Although many genome projects corresponding to several species were completed years ago, the catalogs of the genes do not specify the exact number of genes that compose the genome, which is a quantity that can vary in individuals of the same species or even among cells of the same organism. Another controversial subject is the definition of the *gene* concept. Aside from recent investigations where overlapping genes

[1] *Non coding RNA* are functional RNA molecules that are transcribed from DNA but are not translated into proteins. In general ncRNAs function to regulate gene expression at the transcriptional and post-transcriptional levels.

in different strands of DNA have been shown [61], currently in bioinformatics, genes are discrete entities located linearly within chromosome sequences that hold potential information that is required to be transcribed into RNA. In this manner, the gene prediction problem can be stated as the effort to determine the boundaries of those sequences that are transcribed into RNA in a given genomic DNA target sequence, and this can refer to the search for both protein coding genes and non-coding RNA genes.

Complexity of the problem makes finding an accurate gene model a great bioinformatics challenge. An automatic annotation methodology is necessary because, among other aspects, even manual annotation uses computational techniques that search for evidence of the existence of genes in the absence of homologous sequences. In fact, the current tendency is to develop integrative systems that can incorporate diverse sources of information.

The human ENCODE Genome Annotation Assessment Project, EGASP [28], was a community study conducted to evaluate the state of gene prediction accuracy in the human genome. In this experiment, the gene predictor systems were divided into categories based on the information sources used to formulate their predictions. If we examine the presented results of such predictions with reference to a specific benchmark, it is easy to conclude that gene prediction is an unsolved problem, and hence every effort spent on it is still relevant. In spite of the great efforts produced in this field of research, the existing gene prediction programs have not provided a complete solution to the problem so far. Short exons are often difficult to locate because discriminative statistical characteristics are less likely to appear in short segments. In addition, the problem of alternative splicing, an important regulatory mechanism during gene expression in eukaryotes whereby a single gene codes for multiple proteins, has to be resolved in a more efficient way.

1.2 Biological background

Proteins are complex chains of chemical elements called amino acids. The instructions for building the proteins in an organism are maintained by another type of organic molecule called nucleotides, and groups of three nucleotides (codons) code for an amino acid. Nucleotides are joined together to form large chains in the core of cells called deoxyribonucleic acid or DNA. Nucleotides are organic molecules composed of a nitrogenous base, a five-carbon sugar, and at least one phosphate group. The four existing nitrogenous bases include Adenine, Thymine, Guanine and Cytosine, which are abbreviated A, T, G, and C, respectively. Therefore, the 64 different codons codify 20 amino acids (synonymous codons exist). A molecule of DNA is organized to form of two complementary chains of nucleotides wound in a double helix. In chromosomes, the sequence of each nucleotide in one strand is opposed by a complementary nucleotide in the other strand. A is the complement of T, and C and G are complementary bases. Therefore, the sequence of nucleotides of one strand is fully determined by the sequence of nucleotides on the opposite strand, and it is only necessary to know the sequence of one of the strands. This strand is called the forward strand, and its complementary strand is the reverse strand. Both strands have a (chemically) distinguished direction.

A gene is a DNA segment that codes for a specific functional element, i.e., a protein or non-coding RNA molecule. In eukaryotic cells, genes are linearly distributed along the two complementary strands of the chromosomes without overlapping in the same strand and are separated by intergenic regions, extensive areas of DNA between regions that have no currently known function. In a simplification of the central dogma of molecular biology, it can be stated that the information contained in the coding portion of the genes flows from DNA to RNA and then to proteins. These coding segments are called exons. However, in eukaryotic cells, these segments are interrupted by other segments called introns. Although introns are transcribed to RNA, they are removed when the RNA matures, and therefore such information is not translated into proteins. The intron boundaries are denoted as splice sites, which are called donor sites, and they include the beginning of the intron (i.e., the end of exon) or acceptor site and the end of the intron (i.e., where the next exon begins). Figure 1.1 illustrates the gene concept on a DNA strand, including the key functional sites on the strand, followed by the information flow in the biological processes of protein synthesis. Exons are joined together to become a mature RNA product, which is usually transformed into the corresponding protein through the process of translation. Although most gene finding software programs use the term exon to denote only the coding parts of the genome, in reality some exons (or parts of them) are non-coding. However, in this work, the commonly used terminology of gene prediction is followed, and the term exon only refers to the coding part of the genome.

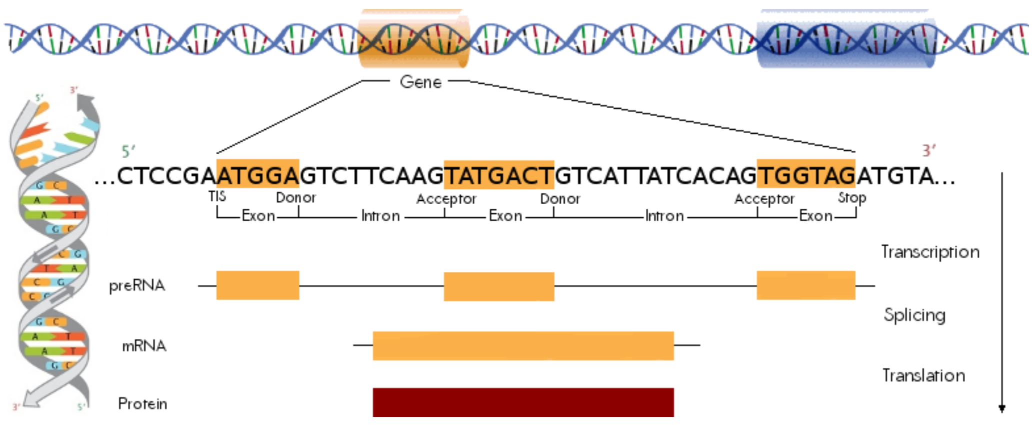


Figure 1.1: Information flow within the proteins synthesis process in a eukaryotic cell.

1.3 The gene prediction problem

As previously mentioned, the problem of gene prediction consists of identifying the portions of DNA sequences that hold the information that encodes the biologically functional elements. Although this concept mostly refers to protein-coding regions, it can also refer to other functional elements, e.g., ncRNA genes. In this sense, the main aim of gene prediction is to correctly label each element of the DNA sequence as belonging to a protein-coding region, RNA coding region, or non-coding or intergenic region. To represent the problem from a computational point of view, a DNA sequence can be established by its minimal dimension as a string over the alphabet A, T, G, C that correspond to the nitrogenous bases of the nucleotides, including Adenine, Thymine, Guanine and Cytosine, respectively.

Then, the gene prediction problem can be stated as:

Given a DNA sequence

$$X = (x_1, x_2, ..., x_n) \in \Sigma, \text{ where } \Sigma = \{A, C, G, T\} \tag{1.1}$$

the goal is to correctly label each element of X as belonging to a specific region.

All coding genes begin at the translation initiation site (TIS) and end at the stop codon. As previously mentioned, the coding segments are usually interrupted by introns in eukaryotic cells. Exon–intron and intron–exon boundaries are called the splice sites or donor and acceptor sites, respectively. Thus, in a correct gene structure:

- The exons do not overlap.
- The gene starts and finishes within an exon.
- An intron must be flanked by two exons.
- A gene can consist of only one exon.
- The complete set of coding exons must be frame compatible.
- Merging coding exons will not generate an in-frame stop codon.

Therefore, the problem consists of determining which parts of a DNA sequence code for the whole gene[2] from its start site to its stop codon [47, 10]. In the case of humans, it is believed that only 3% of the DNA sequence consists of coding regions. Gene prediction is relatively simple in prokaryotes due to their higher gene density and absence of introns. The main difficulty in prokaryote gene prediction is the presence of overlapping regions. The process is more complex for eukaryotes due to the large genome size and short exons that are bordered by large introns. Furthermore, eukaryote coding segments are subject to alternative splicing, i.e., a process of joining exons in different ways during RNA splicing. Indeed, it is estimated that more than 95% of human genes show evidence of at least one alternative splice site.

1.4 Current approaches

Most of the current gene prediction programs share a basic philosophy and have many common methods. However, there are differences among them that can be used to create a certain taxonomy of gene recognizing programs. In general, current gene recognition systems can be divided into three categories [27] depending on the type of information that they use.

[2]Some gene finding approaches consider information in addition to the coding parts of the genes. In our case, the gene is considered only the segment from the TIS to the stop codon.

- *Ab initio* predictors use only DNA sequences from the genome in which predictions are desired and are called target genome programs. Predictors such as GENSCAN[11] and CRAIG[7] belong to this category.
- *De novo* techniques additionally make use of aligned DNA sequences from other genomes, called informant genomes. Alignments with other genomes increase the prediction power because the functional parts of the genome show an outstanding level of conservation. ROSETTA[6] and CEM[2] were the first methods for predicting human genes using this type of information. More advanced de novo predictors include N-SCAN[26] and CONTRAST[27].
- *Homology based* A third class of predictors make use of the products of the genes themselves, which usually include alignments over known CDS, protein sequences, RNA-seq, expressed sequence tags (EST) or cDNA. Pairagon[1], N-SCAN_EST[70], GenomeWise[8] and EXOGEAN[18] are included in this category. These methods can provide highly accurate predictions for genes that are well covered by alignments of the expressed sequences.

However, some methods, such as AUGUSTUS[66], can work with ab initio, de novo or homology-based information.

Typically, all methodologies that face the gene prediction problem have three main components: signal detection, content sensing and optimal integration according to a global model of the gene structure. Signals are specific functional sites that are inside or at the boundaries of the various genomic regions and are involved in various levels of gene expression. This is the case for transcription factor binding sites, TATA boxes, splice sites (donor and acceptor sites), poly-A sites, translation initiation sites (TIS), and stop codons. Thus, signal detection is the task of evaluating and choosing all possible functionally relevant points in the sequence. Content sensors are measures that classify sequence regions into types by evaluating the set of nucleotides within them. Usually, these measures are discriminant factors between coding and non-coding regions, CpG islands, and ALU sequences. However, signal and content sensors cannot solve the problem by themselves, and an integrator mechanism becomes necessary to obtain the whole gene structure. This element can be considered as a framework where the other two main components must be integrated, resulting an accurate, efficient and consistent combination of information.

1.4.1 Functional site recognition

The functional sites that mark the transcriptional, splicing and translational boundaries in a gene consist of a sequence of nucleotides that are recognized by the cellular machinery. The minimal set of signals needed to define a coding sequence include the start, translation initiation site, and stop codons, and for genes with multiple exons, the splice sites and donor and acceptor sites for each intron are necessary. Other relevant signals include promoter sequences, transcription start and termination sites, and TATA boxes.

One of the first and most accurate attempts to model functional sites in the sequence consisted of the Positional Weight Matrices (PWM), known also as position specific scoring matrices or position specific probability matrices. PWMs are based on a matrix of the frequencies of nucleotides observed at each position. The likelihood that a sequence belongs to a certain class is derived from the product of the probabilities of the nucleotides in each position according to the PWM for each type of site. Dependencies between adjacent positions were incorporated in the Weight Array Matrix models (WAM). In this case, the probabilities within the matrix are calculated as conditional probabilities by examining the probability of previous positions. Thus, these models can be considered as Markov chains. Maximal dependence decomposition (MDD) based on decision trees, Inclusion-driven learned Bayesian Networks (idlBNs) or approaches based on Neural Networks have also been used for site prediction purposes. However, the improvements with respect to PWMs or WAMs tend to disappear when they are used inside a global strategy of prediction.

In any case, the methods considered most successful for addressing this task at this moment are the Suppor Vector Machines models (SVMs). The SVMs involve a machine learning method that attempts to discriminate two classes by separating them with as large of a margin as possible. They are trained by solving an optimization problem, and they employ similarity measures known as kernels. Specifically, the most useful kernel for site classification, the Weighted Degree kernel (WD) [58], compares pairs of sequences in terms of their substring matching. This approach, instead of performing a probability estimation, attempts to estimate a function that ranks the actual sites with as large of a margin as possible to all other potential sites.

1.4.2 Gene prediction frameworks

The most important evidence of the presence of genes are the TIS, stop codons, and splice sites, because they define the boundaries of coding regions, in addition to the content sensors, which determine the differences between regions according to their base composition. This evidence can be combined for predicting the whole gene structure, in contrast to the earlier approaches that identified individual exons. Dynamic Programming, Hidden Markov Models, Conditional Random Fields or Rule-based systems have been successfully used for these purposes, determining a global model of gene structure. The gene prediction problem can be addressed by choosing the optimal combination of potential exons. However, usually, the number of possible combinations of potential exons increases exponentially, and considering all combinations is not feasible. Dynamic Programming solves this problem because it determines the optimal solution without estimating all possibilities. In Hidden Markov Model approaches (HMM), different types of components (exon or introns) are labeled by a state, and the gene model is generated by a state machine. Each base is generated by an emission probability conditioned on the current state and a specific number of preceding bases. The transition from one state to another is determined by a transition probability that maintains constraints inherent to the biological problem. All of the parameters are learned from the training data set (annotated genes). Because the states are unknown (hidden), the Vitrebi algorithm is used to select the best set of consecutive states that holds the highest probability of any possible set of consecutive states for a given genomic sequence without actually having to enumerate all possible sets of consecutive states. More recent techniques[27] have been based on Conditional Random Fields (CRF), which are particularly suitable for gene prediction and include the semi Markov CRF. This model produces labels for segments over a target input sequence. Essentially, it is designed to find the most likely set of labels (states) that the model has been trained to determine over given a set of observations (input sequence). In this case, the probability of value-label pairs, the labels that are conditioned on the values, is learned directly. The observations are examined and not emitted.

1.5 Aims and Objectives

The main aim of this thesis is the design of machine learning methodologies to address different aspects of the gene prediction problem. We will use the most powerful soft-computing and data mining approaches to address the high complexity of the problem. All tasks that are involved in the gene prediction problem will be subject to study and improvement. Specifically, to achieve this aim, the following particular objectives have been established:

- The advanced techniques of metaheuristics will be used to design a gene prediction framework. Given the particularities of the problem, the principles of evolutionary computation will be taken under consideration to develop a gene prediction system that can contemplate a set of biological restrictions and support a correct gene model. It may be flexible enough to integrate several independent information sources, regardless of their origin and nature, with a search power sufficient to address the difficult and large solution space.

- The recognition of the different functional parts of the gene, such as promoters, translation initiation sites, donors, acceptors and stop codons, is a fundamental component of any gene finding system. The functional site recognition problem will be addressed by using machine learning techniques with additional biological information and/or introducing conservative principles from the evolution theory. This objective can be divided as follows:

  – To try the improvement of the performance of the current site recognition classifiers by introducing principles derived from information theory and biological aspects. In particular, we consider that training patterns can be separated, depending on their biological features, to build various models from each group that can be combined in an optimal way to obtain the best general classifier model.

  – A methodology for site recognition will be proposed by combining classifiers that consider as many different sources of evidence as possible from several genomes and as many different type of classifiers as needed. Previous studies have shown that the combination of different sources of evidence is fundamental for the success of any genomic recognition task. This idea has become feasible because the collection of large and complex genomes from different species has become routine.

– The imbalanced nature of the site recognition problem will be addressed. In genome sequences, the number of negative instances is much larger than the number of positive instances. In such cases, the classification problem is considered a class imbalance problem. Most learning algorithms expect a somewhat balanced distribution of instances among the different classes. It has been shown that learning algorithms suffer from a skewed distribution that is associated with class imbalance, resulting in negative effects on their performance.
– To study the relevance of the different features – i.e., the nucleotides and codons – that form the sequences used for recognizing functional sites by means of feature selection techniques. Our aim is to determine if this type of methods are useful for improving the performance of site recognition.

1.6 Proposed methodologies

Computational intelligence techniques have been shown to be suitable for solving several types of bioinformatics problems. In particular, such techniques have been widely and successfully applied to gene prediction tasks and at this moment the use of them is imperative. The presented methodologies in this work are focused on improving functional site recognition and building a global framework for gene prediction, making use of both the principles and strategies of computational intelligence and machine learning.

From the point of view of machine learning, site recognition is a class imbalance problem. Thus, in this work, we approach site recognition from this angle and apply different methods that have been developed to address imbalanced datasets. The proposed approach has two advantages. First, it improves the results using standard classification methods. Second, it broadens the set of classification algorithms that can be used because some of the class-imbalance methods, such as undersampling, are also useful as methods for scaling up data mining algorithms as they reduce the size of the dataset. In this manner, classifiers that cannot be applied to the whole dataset, due to long training time or large memory requirements, can be used when undersampling methods are applied. In the same sense, many classifiers have problems in site recognition tasks due to the large amount of features involved in the problem. Most previous studies have used an arbitrary window of nucleotides around the site. The methodology described in this work presents a more principled method of choosing the relevant features using feature selection techniques, first by selecting the most suitable window for each organism, and then by the selection of a subset of the most relevant features within that window.

At the same time, the most successful site recognition methods use powerful classifiers. All these methods use two classes, one for positive instances and another for negative instances, that are constructed using sequences from the whole genome. However, the features of the negative sequences differ depending on the origin of the negative samples. The negative sequences differ depending on whether they are located on exons, introns or intergenic regions. Thus, the positive class is fairly homogeneous as all of the sequences are located at the same part of the gene, but the negative class is composed of many different instances. The classifiers suffer from this problem. Herein, we propose training different classifiers with different negative classes and using a combination of these classifiers to improve accuracy.

Finally, due to the rapid evolution of our ability to collect genomic information, it has been shown that the combination of different sources of evidence is fundamental for the success of any recognition task. We present a methodology for combining hundreds of different classifiers to improve performance. Our approach can include almost a limitless number of sources of evidence. This approach can be used for any functional recognition task related to the gene prediction task.

Orthogonally, we present an approach that addresses the gene finding task from a general perspective, considering it as a search problem, where many evidence sources can be integrated and combined using the principles of evolutionary computation to produce a framework for gene structure modeling. A genetic algorithm is designed to evolve a population where the individuals consist of correct genetic structures of a given sequence. A fitness function based on several sensors will determine a solution for individuals that show outstanding statistical coding features. Additionally, this approach incorporates the best functional site recognizers to determine the most likely functional points along the genomic sequence, reducing the search space. One of the many advantages of this approach is the ability to incorporate the whole complexity of the problem. Other optimization methods used for this project lack this flexibility.

1.7 Contribution of the thesis

In this thesis, new approaches for the gene prediction problem based on advanced techniques of machine learning are presented. Specifically, we have focused our efforts on two main components of gene finding systems: improving the current approaches of functional site recognition and presenting a new integrative framework for gene prediction based on evolutionary computation principles.

The proposed methodologies have been successfully applied to gene recognition tasks. In the case of site recognition, an improvement has resulted from three different factors:

> First, from a pure machine learning point of view, the class imbalance nature of the problem has been revealed, and it has been shown that the methods developed to address this feature can be useful for approaching the site recognition problem. Equally, the usefulness of the application of feature selection techniques to improve the performance of the site classifiers by reducing the whole sequence to a smaller sequence and elaborating the ranking of the most interesting features in terms of bases or codons has been demonstrated.
>
> Second, a new approach has been introduced based on the idea of considering more than two groups in the training patterns used to generate the classification models and combining more than one type of classifier. The first premise is based on the biological feature differences presented by the patterns and allows for more than two groups with different importance levels in the classification process, instead of using the positive–negative binary separation. The second idea was motivated by the fact that the behavior of an ensemble of classifiers improves the performance of classifiers individually, and even more so if diversity exists between them.
>
> The third contribution to site recognition is a methodology based on the combination of classifiers that considers as many different informant genomes as possible and as many different classifiers as needed. This

approach opens a new research field for determining non canonical genes that most current gene recognizers tend to ignore.

All of these techniques were shown to be successful by experiments on the chromosomes of several organisms, mostly in the human genome, and were compared with the results of the best current methods for site prediction.

In the case of evolutionary gene recognition framework, the introduced paradigm has the ability to define accurate gene models while maintaining a set of constraints presented by the problem. At the same time, it is flexible enough to integrate the functional site recognition and content sensor mechanisms, integrating the latter as the search driver of the most likely gene structure in a given target sequence.

The research performed during the development of this thesis produced valuable results in the form of presentations at international congresses and the publication of several articles in outstanding journals in the specific research field.

1.8 Thesis organization

The present document is organized as follows. Chapter 2 describes the class imbalance nature of functional site recognition problem and presents a detailed study about the most useful classic machine learning techniques to address it. Equally, Chapter 3 shows the convenience of the application of feature selection methods to improve the performance of the site prediction classifiers. Chapters 4 and 5 provide the details of the design of two new approaches to address functional site recognition in DNA sequences by merging concepts from different backgrounds as information theory and biology. These chapters provide a discussion on the most relevant aspects in their implementations as well as the experimental setup and the study of the results. Chapter 6 presents in depth a gene prediction global framework based on evolutionary computation and includes a description of the results obtained by the methodology on human genome. Finally, Chapter 7 states the final conclusions of our work and a summary of the highlighted results of the described methodologies.

Chapter 2

Class imbalance methods for site recognition

TIS, splice site or stop codon recognition consists of identifying the places which mark the boundaries of the coding segments in the genes. Most previous approaches have focused on recognizing sites in transcripts. However, recognizing sites in genomic sequences is a different, more difficult task. Full length or partial transcripts usually contain a few of sites, and no introns. On the other hand, in a generic genetic sequence, we can find decoy codons, and thus a putative TIS, in any place. In this work we consider the most difficult case of analyzing genomic sequences that contain intergenic DNA, exons, introns and untranslated terminal regions (UTRs). The different characteristics of recognizing site in transcripts and genomic sequences are illustrated in the different performance of the predictors in each problem. For example, TisMiner [44] is one of the best performing programs for TIS recognition in transcripts, able to achieve a specificity of 98% at a sensitivity level of 80%. However, when tested in genomic sequences, its performance at the same level of sensitivity drops to a specificity of 55%.

One of the most important characteristics of site prediction in genome sequences is the fact that negative instances greatly outnumber positive instances. In machine learning theory, this is called the class imbalance problem [3] [45]. Most learning algorithms expect a somewhat balanced distribution of instances among the different classes. It has been shown that learning algorithms suffer from the skewed distribution that is associated with class imbalance. Most of the previous works in site recognition have not considered addressing this problem from the point of view of class imbalance methods. However, the problem can be highly imbalanced. In our TIS test sets we have a positive/negative ratio of 1:25, 1:93, and 1:123. In sequences with low level of codification, such as human chromosome 21, it can reach a ratio of 1:4912. The cases of stop codon and splice site prediction are even worse.

In this chapter we approach the recognition of TIS as a class imbalance problem. We test whether class imbalance methods are able to achieve the same performance of the methods designed specifically for TIS recognition from the biological point of view. This work also tests some of the most widely used class-imbalance methods in a hard real-world task. Thus, it allows an interesting evaluation of those methods in difficult problems.

2.1 Class Imbalance Problems

It has been repeatedly shown that most classification methods suffer from an imbalanced distribution of the training instances among the classes [13]. Most learning algorithms expect an approximately even distribution of the instances among the different classes and suffer, in different degrees, when that is not the case. Dealing with the class imbalance problem is a difficult task, but a very relevant one as many of the most interesting and challenging real-world problems have a very uneven class distribution, such as gene recognition, intrusion detection, web mining, etc.

In most cases this problem appears in two class datasets. There is a class of interest, the positive class, which is highly underrepresented in the dataset, together with a negative class which accounts for most of the instances. In highly imbalanced problems the ratio between the positive and the negative class can be as high as 1:1000 or 1:10000. Many algorithms and methods have been proposed to ameliorate the effect of class imbalance on the performance of the learning algorithms. There are mainly three different approaches [67] [21]:

- Internal approaches acting on the algorithm. These approaches modify the learning algorithm to deal with the imbalance problem. They can adapt the decision threshold to create a bias towards the minority class or introduce costs in the learning process to compensate the minority class.

- External approaches acting on the data. These algorithms act on the data instead of on the learning method. They have the advantage of being independent from the classifier used. There are two basic approaches, oversampling the minority class and undersampling the majority class.

- Combined approaches which are based on *boosting* [20] taking into account the imbalance in the training set. These methods modify the basic boosting method to account for the minority class underrepresentation in the dataset.

There are two principal advantages of sampling against cost sensitive methods. Firstly, sampling is more general as it does not depend on the possibility of adapting a certain algorithm to work with classification costs. Secondly, the learning algorithm is not modified, which can be in many cases a difficult task and also adds additional parameters to be tuned.

Data driven algorithms can be broadly classified into two groups. Undersampling the majority class and oversampling the minority class. There are also algorithms that combine both processes. Both undersampling and oversampling can be made randomly or with a more complicated process searching for least/most useful instances. Previous works have shown that undersampling the majority class usually leads to better results than oversampling the minority class [13], at least, when oversampling is performed using sampling with replacement from the minority class. Furthermore, combining undersampling on the majority class with oversampling of the minority class has not yield to better results than undersampling of the majority class alone [43]. One of the possible sources of this worse performance of oversampling is the fact that there is no new information introduced in the training set, as oversampling must rely on adding new copies of the minority class instances already in the dataset.

2.1.1 Undersampling and Exploratory Undersampling

The first method for balancing a dataset is undersampling the majority class until both classes have the same number of instances. We have not used oversampling methods because most previous works agree that undersampling performs better than oversampling [42]. However, a few works have found the opposite [19]. Additionally, as we are dealing with very large datasets, oversampling would make the datasets almost twice in size, preventing the use of some of the most interesting classifiers, such as support vectors machines.

Random undersampling consists of randomly removing instances from the majority class until a certain criterion is reached. In most works, instances are removed until both classes have the same number of instances. Several studies comparing sophisticated undersampling methods with random undersampling [34] have failed to establish a clear advantage of the formers. Thus, in this work we consider first random undersampling. However, the problem with random undersampling is that many, potentially useful, samples from the majority class are ignored. In this way, when the majority/minority class ratio is large the performance of random undersampling degrades [21]. Furthermore, when the number of minority class samples is very small, we will also have problems due to small training sets.

Liu et al. [46] propose two ensemble methods combining undersampling and boosting to avoid that problem. This methodology is called exploratory undersampling. The two proposed methods are called `EasyEnsemble` and `BalanceCascade`. We describe these two methods in more detail as we have tested both experimentally. `EasyEnsemble` consists of applying repeatedly the standard ensemble method AdaBoost [5] to different samples of the majority class. Algorithm 1 shows `EasyEnsemble` method. The idea behind `EasyEnsemble` is generating T balanced subproblems sampling from the majority class.

`EasyEnsemble` is an unsupervised strategy to explore the set of negative instances, $\mathcal{N}$, as the sampling is made without using information from the classification performed by the previous members of the ensemble. On the other hand, `BalanceCascade` method explores $\mathcal{N}$ in a supervised manner, removing from the majority class those instances that have been correctly classified by the previous classifiers added to the ensemble. `BalanceCascade` is shown in Algorithm 2.

2.1.2 SMOTE-N

One of the problems with oversampling is that merely making copies of the minority class samples does not add new information to the dataset, and the learning method is not able to significantly improve the classification of the minority class. To overcome this problem, Chawla et al. [13] proposed a method called SMOTE, which combines undersampling of the majority class with oversampling of the minority class. However, instead of oversam-

```
Data    : A minority training set $\mathcal{P}$ and a majority training set $\mathcal{N}$,
          $|\mathcal{P}| \ll |\mathcal{N}|$, a number of subsets $T$ to sample from $\mathcal{N}$ and $s_i$
          the number of iterations to train the AdaBoost ensemble
          $H_i$.
Result  : The final ensemble:
          $H(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{T}\sum_{j=1}^{s_i}\alpha_{i,j}h_{i,j}(\mathbf{x}) - \sum_{i=1}^{T}\theta_i\right)$.
for $i = 1$ to $T$ do
    1: Randomly sample a subset $\mathcal{N}_i$ from $\mathcal{N}$, $|\mathcal{N}_i| = |\mathcal{P}|$
    2: Learn $H_i$ using $\mathcal{P}$ and $\mathcal{N}_i$. $H_i$ is an AdaBoost ensemble with $s_i$ weak classifiers $h_{i,j}$ and corresponding weights $\alpha_{i_j}$. The ensemble threshold is $\theta_i$. $H_i$ is given by: $H_i(\mathbf{x}) = \text{sgn}\left(\sum_{j=1}^{s_i}\alpha_{i,j}h_{i,j}(\mathbf{x}) - \theta_i\right)$.
end
```

Algorithm 1: `EasyEnsemble` method.

```
Data    : A minority training set $\mathcal{P}$ and a majority training set $\mathcal{N}$,
          $|\mathcal{P}| \ll |\mathcal{N}|$, a number of subsets $T$ to sample from $\mathcal{N}$ and $s_i$
          the number of iterations to train the AdaBoost ensemble
          $H_i$.
Result  : The final ensemble:
          $H(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{T}\sum_{j=1}^{s_i}\alpha_{i,j}h_{i,j}(\mathbf{x}) - \sum_{i=1}^{T}\theta_i\right)$.
1: $f = \sqrt[T-1]{|\mathcal{P}|/|\mathcal{N}|}$. $f$ is the false positive rate that $H_i$ should achieve.
for $i = 1$ to $T$ do
    2: Randomly sample a subset $\mathcal{N}_i$ from $\mathcal{N}$, $|\mathcal{N}_i| = |\mathcal{P}|$
    3: Learn $H_i$ using $\mathcal{P}$ and $\mathcal{N}_i$. $H_i$ is an AdaBoost ensemble with $s_i$ weak classifiers $h_{i,j}$ and corresponding weights $\alpha_{i,j}$. The ensemble threshold is $\theta_i$. $H_i$ is given by: $H_i(\mathbf{x}) = \text{sgn}\left(\sum_{j=1}^{s_i}\alpha_{i,j}h_{i,j}(\mathbf{x}) - \theta_i\right)$.
    4: Adjust $\theta_i$ such that $H_i$'s false positive rate is $f$.
    5: Remove form $\mathcal{N}$ all examples that are correctly classified by $H_i$.
end
```

Algorithm 2: `BalanceCascade` method.

pling the minority class just making copies of the minority class samples, SMOTE generates synthetic instances from actual instances of the minority class. Synthetic samples are generated in the following way: take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. In this way, we can generate new samples that share the main characteristics of actual instances for a more dense dataset.

The original algorithm is proposed for numerical attributes. However, an algorithm, called SMOTE-N, is also proposed for nominal attributes as it is our case. The procedure for generating this subset of synthetic samples, SMOTE-N, is shown in Algorithm 3. SMOTE-N differs from standard SMOTE in using a modified version of the value difference metric (VDM), proposed by Cost and Salzberg [15], instead of the Euclidean distance, as our instances, DNA sequences, have only nominal attributes.

```
Data   : A minority training set $\mathcal{P} \subset \mathbb{R}^D$, the number of synthetic
         instances to generate for each instance $N_s$, and the number
         $k$ of nearest neighbors.
Result : $\mathcal{P}$ with the synthetic instances added.
for $t = 1$ to $|\mathcal{P}|$ do
    1: Obtain $k$ nearest neighbors of $\mathbf{x}_t$ using modified VDM distance
    for $i = 1$ to $N_s$ do
        for $j = 1$ to $D$ do
            2: $\mathbf{x}^j_{new}$ = select randomly a value from the corresponding
            feature $j$ within the set of $k$ nearest neighbors of the same
            class as $\mathbf{x}_t$.
        end
        3: Add $\mathbf{x}_{new}$ to $\mathcal{P}$
    end
end
```

Algorithm 3: SMOTE-N algorithm.

2.1.3 Evaluation measures in class imbalanced problems

Accuracy is not a useful measure for imbalanced data, specially when the number of instances of the minority class is very small compared with the majority class. If we have a ratio of 1:100, a classifier that assigns all instances to the majority class will have a 99% accuracy. Several measures [67] have been developed to take into account the imbalance nature of the problems. Given the ratio of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) we can define several measures. Perhaps the most common are the true positive rate (TP_{rate}), recall (R) or sensitivity (Sn):

$$TP_{rate} = R = Sn = \frac{TP}{TP + FN}, \tag{2.1}$$

which is relevant if we are only interested in the performance on the positive class; and the true negative rate (TN_{rate}) or specificity (Sp)[1]:

$$TN_{rate} = Sp = \frac{TN}{TN + FP}. \tag{2.2}$$

From these basic measures, others have been proposed, such as the F-measure [41] or, if we are concerned about the performance on both negative and positive classes the $G - mean$ measure [36]: $G - \text{mean} = \sqrt{Sp \cdot Sn}$.

Many classifiers are subject to some kind of threshold that can be varied to achieve different values of the above measures. For that kind of classifiers receiver operating characteristic (ROC) curves can be constructed. A ROC curve, is a graphical plot of the TP_{rate} (sensitivity) against the FP_{rate} (1 - specificity or $FP_{rate} = \frac{FP}{TN+FP}$) for a binary classifier system as its discrimination threshold is varied. The perfect model would achieve a true positive rate of 1 and a false positive rate of 0. A random guess will be represented by a line connecting the points $(0, 0)$ and $(1, 1)$. ROC curves are a good measure of the performance of the classifiers. Furthermore, from this curve a new measure, area under the curve (AUC), can be obtained, which is a very good overall measure for comparing algorithms. AUC is a useful metric for classifier performance as it is independent of the decision criterion selected

[1]In Bioinformatics, sometimes an alternative form of specificity is used, $Sp = \frac{TP}{TP+FP}$, as the large number of negative instances may yield to an unrealistic high specificity value.

and prior probabilities. The AUC comparison can establish a dominance relationship between classifiers. If the ROC curves are intersecting, the total AUC is an average comparison between models [40].

In our experiments we will use as main comparison tool ROC curves. As a numerical measure we have chosen the AUC value obtained by means of a trapezoid numerical integration. Saeys et al. [59] developed the best current model and compared their proposal with other methods using the obtained specificity at a sensitivity level of 80%. To allow a fair comparison with their method, we will also use a sensitivity of 80% and show the corresponding specificity.

2.2 Experimental setup

Among the different methods for dealing with class imbalance problems we have selected some of the most successful ones in the literature, which we have described in previous sections. We have used random undersampling, SMOTE-N, `BalanceCascade` and `EasyEnsemble`. Other methods were tried with worse results. We must take into account that the two former are single classifier methods, while the two latter, are ensemble methods. Due to their increased complexity, `BalanceCascade` and `EasyEnsemble` must achieve a significantly better performance than undersampling and SMOTE-N to be competitive.

As base learners we used a C4.5 decision tree [53], a support vector machine (SVM) [62] and a k-nearest neighbors (k-NN) classifier. These are some of the most powerful learning algorithms available. Although many other classification methods can be used, these three are usually the best ones in most works devoted to classification. There are also other reasons in their favor. We have used decision trees because they can deal with nominal values, are fast and achieve good results, and are good base learners for ensembles. SVMs are included because overall they usually tend to be the best methods in classification tasks, are specifically designed to two-class problems and are very efficient in problems with many inputs. Finally, k-NN method is used because it is simple and fast, and achieves very good results in other real-world applications such as computer vision [64] and other areas of Bioinformatics [52].

We have performed our experiments using k-fold cross-validation for setting the values of the parameters, with $k = 10$. For each one of the classifiers used, we have obtained the best parameters from a set of different values. For SVMs we tried a linear kernel with $C \in \{0.1, 1, 10\}$, and a Gaussian kernel with $C \in \{0.1, 1, 10\}$ and $\gamma \in \{0.0001, 0.001, 0.01, 0.1, 1, 10\}$, testing all the 21 possible combinations. For C4.5 we tested 1 and 10 trials and softening of thresholds trying all the 4 possible combinations. For k-NN the value of k is obtained by cross-validation in the interval $[1, 100]$. The cross-validation process used for setting the parameters is as follows: each time a classifier has to be trained the *training* set is divided into k parts. Then each set of parameters is evaluated using these k parts as in standard k-fold cross-validation. That is, for evaluating each subset of parameters, each one of the k parts is used in turn for testing the performance of the parameters and the remaining $k-1$ parts are used to train the classifier. After all the sets of parameters have been evaluated, the best set is selected and the classifier is trained with the whole training set and that set of parameters. G-mean is used as the evaluation measure of each set of parameters. The whole process for evaluating the testing error and obtaining the best parameter set is shown in Figure 2.1 for $k = 10$.

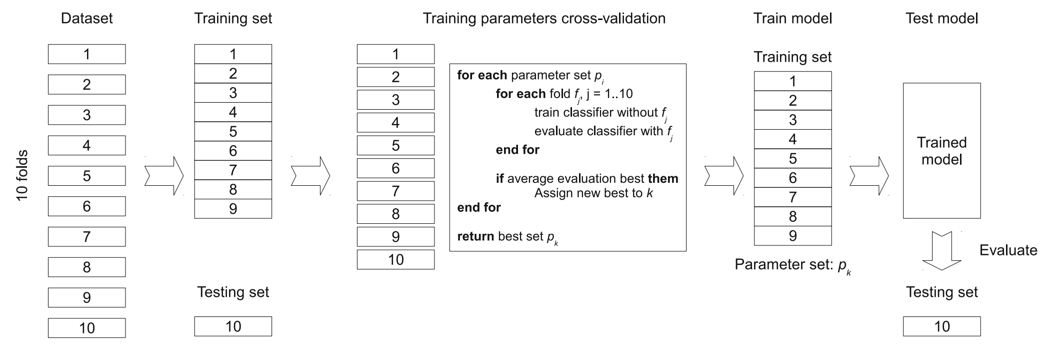

Dataset
Training set
Training parameters cross-validation
Train model
Test model
10 folds
1
2
3
4
5
6
7
8
9
10
Testing set
for each parameter set p_i
for each fold f_j, j = 1..10
train classifier without f_j
evaluate classifier with f_j
end for
if average evaluation best them
Assign new best to k
end for
return best set p_k
Trained model
Parameter set: p_k
Evaluate

Figure 2.1: Procedure for evaluating the testing error using k-fold cross-validation, with $k = 10$ for one of the folds, together with the procedure of obtaining the best set of parameters also using 10-fold cross-validation in the training set.

The source code, in C and licensed under the GNU General Public License, used for all methods as well as the datasets are freely available upon request from the authors. SVMs were implemented using `libsvm` [12] library.

2.2.1 Datasets

We have used three datasets for testing the performance of the described methods. The CCDS dataset was compiled by Saeys et al. [59] from the consensus CDS database. The CCDS project is a collaborative effort of compiling and identifying a core of human genes that are accurately annotated. The annotation is a mix of manual curation and automatic annotation. CCDS contains 350,578 negative samples and 13,917 positive samples with a positive/negative ratio of 1:25. *Ustilago* dataset is a set of coding and non-coding regions of genomic sequences from the sequencing of the fungus *Ustilago maydis*. The sequences are first obtained from the Broad Institute[2] and then completed with the information of the Munich Information Center for Protein Sequences (MIPS)[3]. *Ustilago* dataset contains 607,696 negative samples and 6,515 positive samples with a ratio of 1:93. *Arabidopsis* dataset comprises coding and non coding regions of the plant *Arabidopsis thaliana* obtained from "The *Arabidopsis* Information Resource" (TAIR)[4]. This dataset contains 27,342 positive instances and 3,369,875 negatives instances with an imbalance ratio of 1:123.

For estimating the testing error we used a k-fold cross-validation method. In this method the available data is divided into k approximately equal subsets. Then, the method is learned k times, using, in turn, each one of the k subsets as testing set, and the remaining $k-1$ subsets as training set. The estimated error is the average testing error of the k subsets. We have used a fairly standard value for k which is $k=10$.

Our aim was to test the proposed methods in very different datasets to study whether the problem is more difficult depending on the organism. Thus, we used these three datasets because they correspond to very different species. CCDS dataset contains human DNA, thus it has long genes, with many exons and potentially long introns. On the other hand, *Ustilago* dataset contains shorter genes, with very few exons, usually only one or two, and few and shorter introns. *Arabidopsis* complexity is between these two organisms. With these datasets, we can study the behavior of the proposed methods in different environments.

For all datasets we consider a sequence of 500 bps upstream and downstream of every ATG codon. For the classifiers that can deal with nominal

[2]`http://www.broadinstitute.org/annotation/genome/ustilago_maydis/`

[3]`http://www.helmholtz-muenchen.de/en/mips/home/index.html`

[4]`http://www.arabidopsis.org/`

attributes, C4.5 and k-NN, we used the original sequence. For classifiers that need numerical attributes, SVMs, we used a 1 out of 4 codification of each element of the sequence for a total of 4012 inputs. For k-NN classifier we used Hamming distance, d_H, and the votes of each neighbor were weighted. Thus, for a given query instance x, the vote for i-th neighbor, n_i, receives a weight w_i, according to:

$$w_i = \frac{\exp(-d_H(x, n_i))}{\sum_{j=1}^{k} exp(-d_H(x, n_j))}. \tag{2.3}$$

2.3 Experimental results

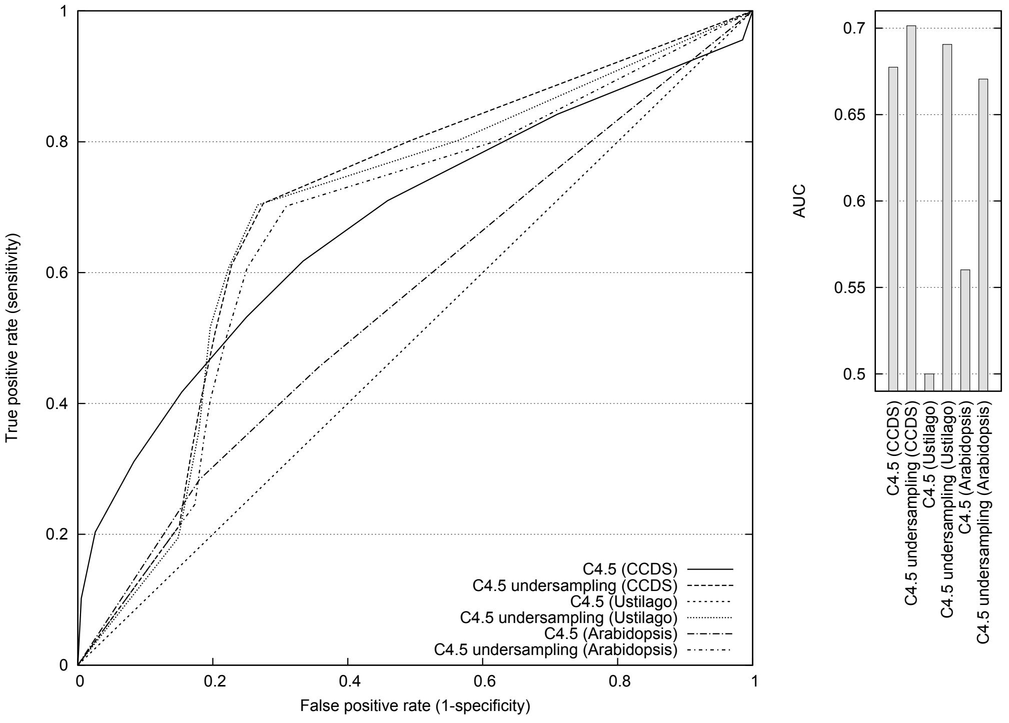
In a first step we wanted to establish the usefulness of undersampling. Figures 2.2, 2.3 and 2.4 show ROC curves for C4.5, k-NN and SVM classifiers respectively for all datasets. These curves show a comparison between the classification method using the whole dataset and the same classifier using random undersampling.

ROC curves show the problems encountered by standard classification methods when facing class imbalance datasets. For *Ustilago* dataset, C4.5 was not able to achieve useful values, classifying all the instances in either class, depending on the used threshold. For CCDS, C4.5 obtained better results, although not competitive with other methods that we will show. For *Arabidopsis* the results are better than for *Ustilago*, but again undersampling is able to show a large improvement. Undersampling was very effective in improving the results for *Ustilago* and *Arabidopsis*. The results for CCDS were also better, although the differences were not so marked, as C4.5 without undersampling had a better performance.

ROC curves for k-NN show more marked differences. Undersampling improved the performance of k-NN consistently. For CCDS the differences are clear, achieving higher sensitivity. For *Ustilago* the behavior is even better. Undersampling improved the performance of k-NN from a very poor level to be one of the best performing algorithms of all for this dataset. The same is observed for *Arabidopsis*, with a very marked improvement.

For SVM the performance of SVM with all the instances is better, but still undersampling provides a significant improvement, especially for CCDS and *Arabidopsis*. The performance of SVM with *Ustilago* is remarkably well, almost matching undersampling in terms of AUC.

Figure 2.2: ROC curve and AUC for C4.5 and CCDS, *Ustilago* and *Arabidopsis* datasets, with and without undersampling.



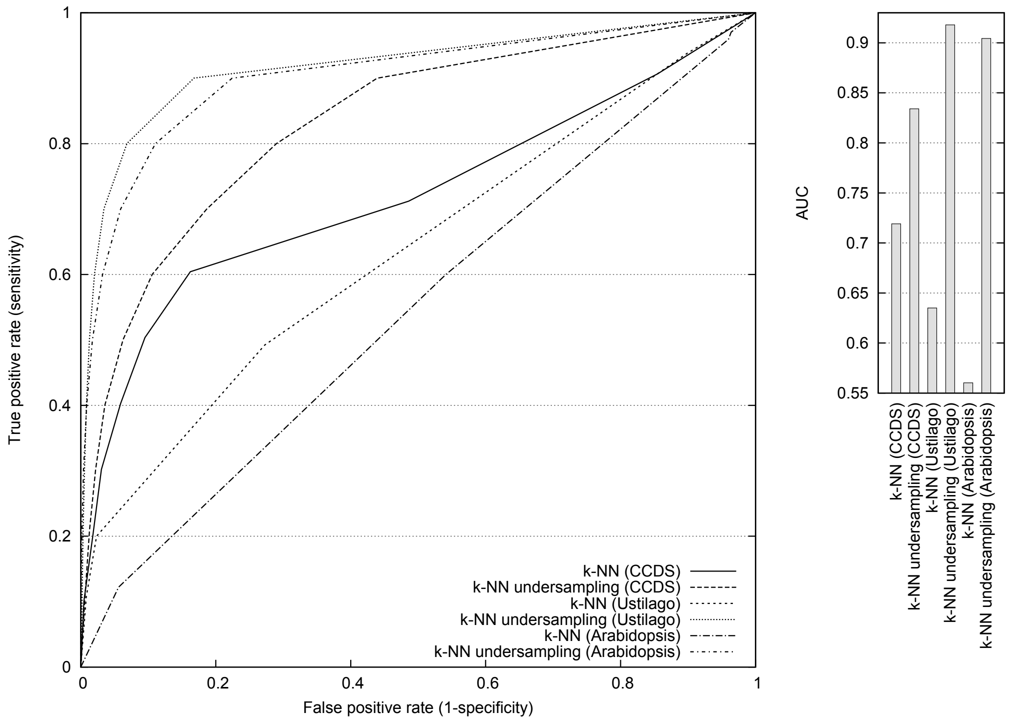


Figure 2.3: ROC curve and AUC for k-NN and CCDS, *Ustilago* and *Arabidopsis* datasets, with and without undersampling.

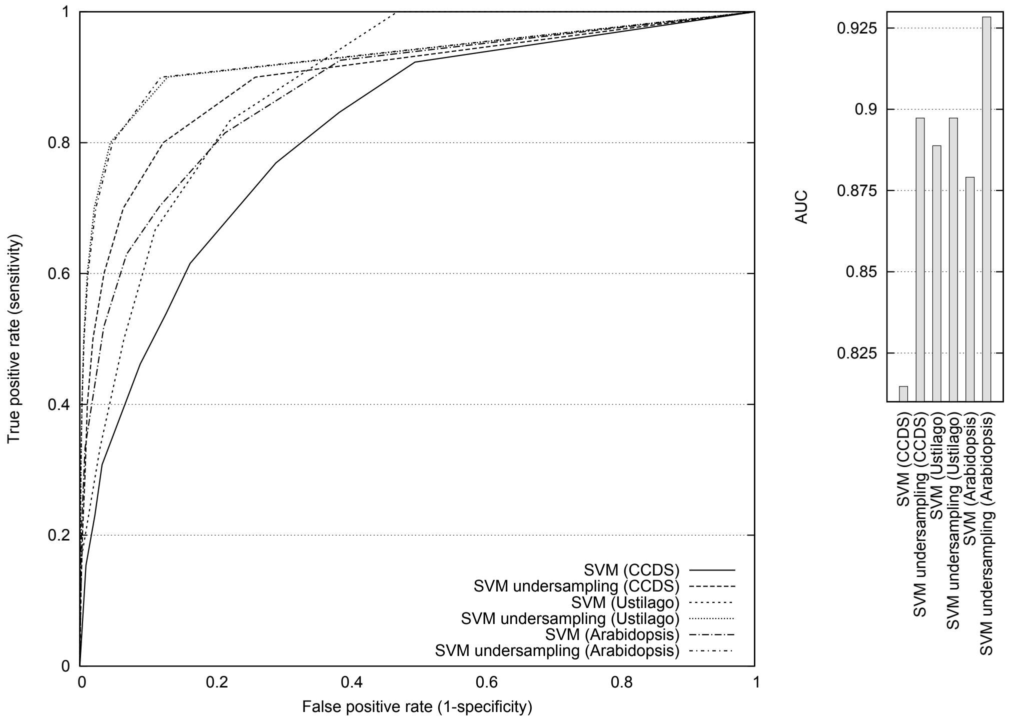


Figure 2.4: ROC curve and AUC for SVM and CCDS, *Ustilago* and *Arabidopsis* datasets, with and without undersampling.

Our second step was the comparison of the performance of undersampling against the more sophisticated method, SMOTE-N, described above. We wanted to check whether the added complexity of SMOTE-N paid off in terms of improved performance. Figures 2.5, 2.6, and 2.7 compare the results using undersampling and SMOTE-N methods for the three classifiers. The behavior using C4.5 and SVM is similar. The results for these two classifiers show small differences using undersampling and SMOTE-N, with a better performance of each one of the methods depending on the dataset and the classifier.

For k-NN, the results are somewhat different, as SMOTE-N is always worse than undersampling, although the differences are not large. Thus, as a general rule, SMOTE-N was not able to significantly improve the results of standard undersampling. It does not mean that SMOTE is not a useful method, although we can conclude that SMOTE-N, the version for nominal attributes, is less efficient.

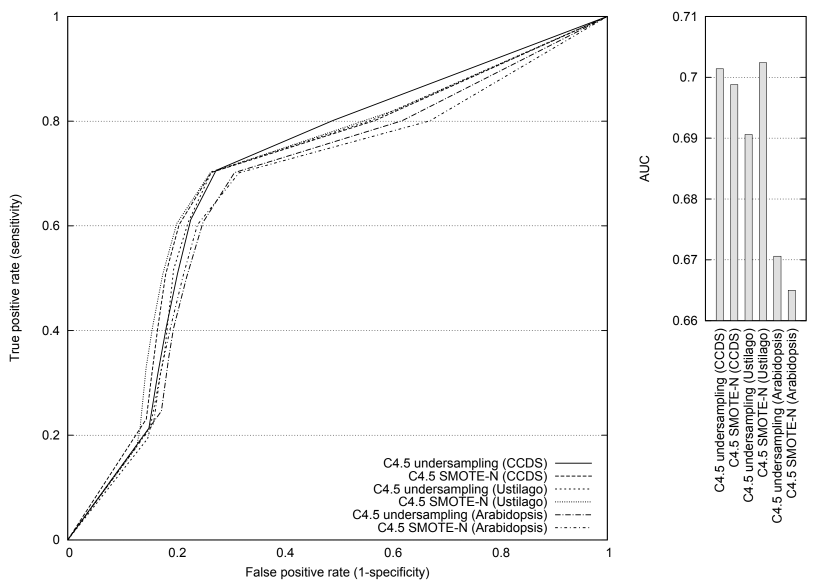


Figure 2.5: ROC curve and AUC for C4.5 and *Arabidopsis*, CCDS and *Ustilago* datasets using undersampling and SMOTE-N methods.

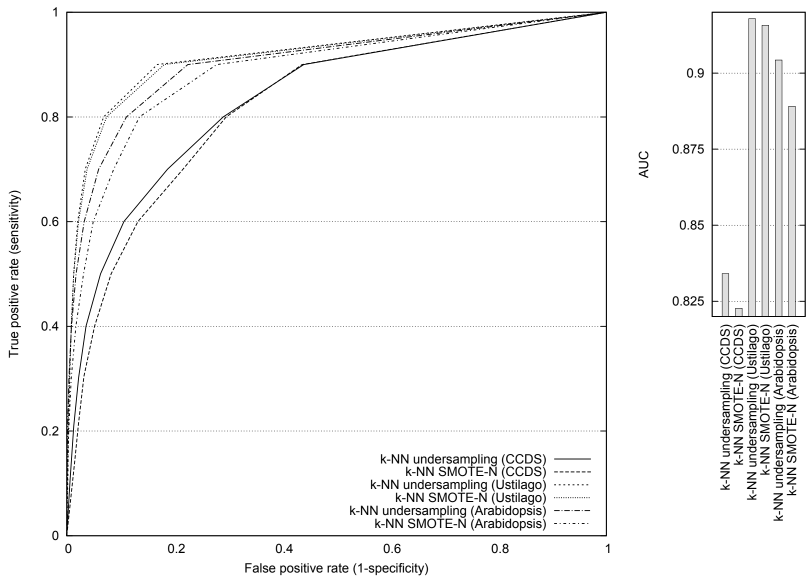


Figure 2.6: ROC curve and AUC for *k*-NN and *Arabidopsis*, CCDS and *Ustilago* datasets using undersampling and SMOTE-N methods.

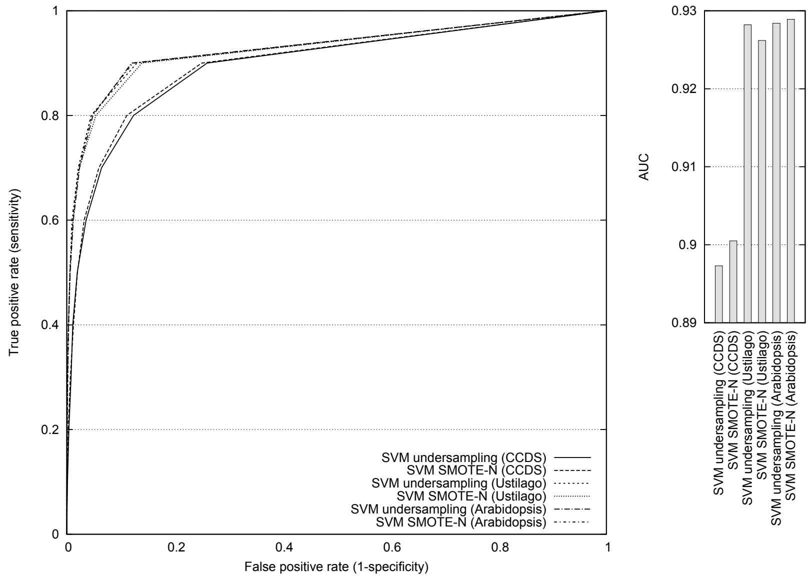


Figure 2.7: ROC curve and AUC for SVMs and *Arabidopsis*, CCDS and *Ustilago* datasets using undersampling and SMOTE-N methods.

2.3.1 Ensemble methods

The next step in our experiments was devoted to ensemble methods. In the previous experiments we have shown the efficiency of undersampling and SMOTE-N in improving the performance of the classifiers. In this section, we show the results of using the two ensemble methods described above. The parameters are chosen following the recommendations of the authors [46]. For easy and balance cascade ensembles we constructed 4 AdaBoost ensembles, $T = 4$, of 10 classifiers, $s_i = 10$.

These two methods are applied using C4.5 and SVM as base learners. k-NN is not used as it has been shown that this classifier is not efficient as a member of an ensemble [24].

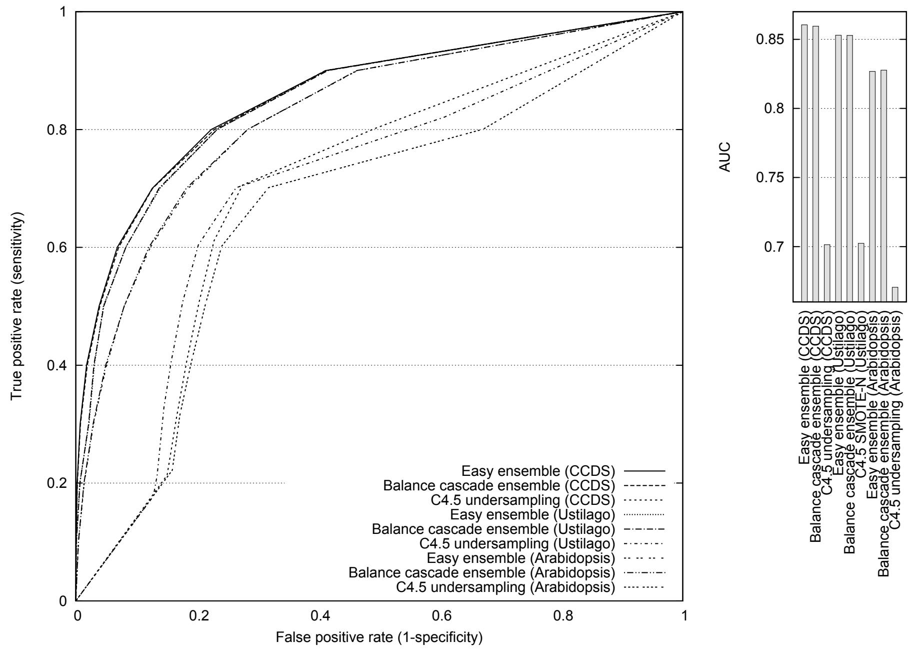
Figure 2.8 shows the ROC curve for easy and balance cascade ensembles. We have included in the plots the performance of C4.5 and SVM with the best class imbalance method, undersampling or SMOTE-N, for each dataset for comparison purposes.

For C4.5, the performance of both methods is very similar in the three datasets. We can see that the performance of these ensemble methods is significantly better than the performance of C4.5 using undersampling or SMOTE-N as imbalance method. However, we must also take into account that their complexity is higher, as these ensembles are made of 40 classifiers. We tried larger ensembles, but the addition of classifiers did not increase the performance in a significant way. It is a known fact [23] that the performance of ensembles does not improve after the first few classifiers are trained even if we add many more classifiers.

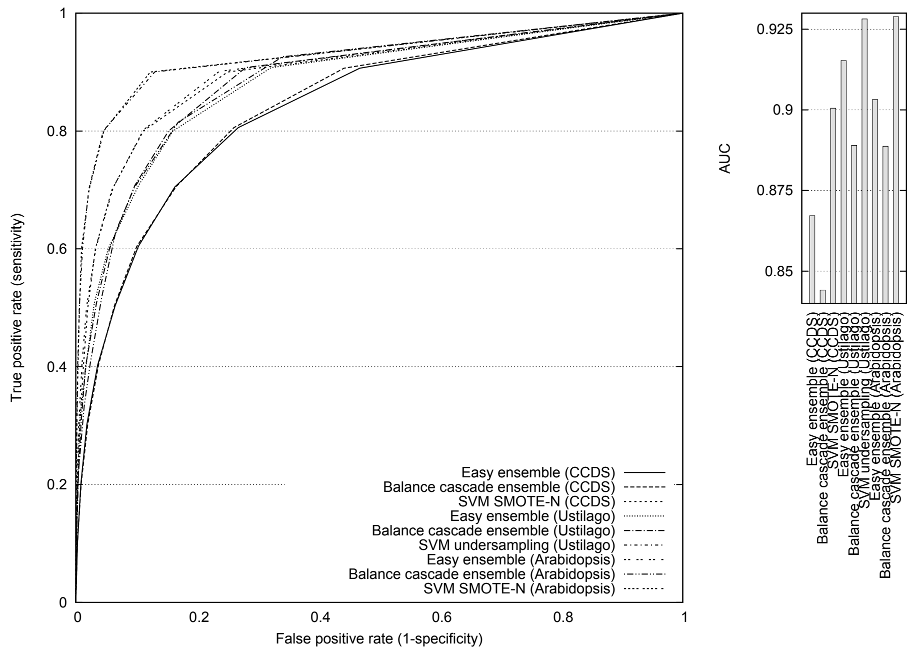
For the case of SVM, the results are different, EasyEnsemble is always better than BalanceCascadeEnsemble. However, both ensemble methods are markedly worse than undersampling or SMOTE-N. It has been shown that SVMs do not usually perform as well as decision trees as members of an ensemble [63]. Our results corroborate that fact.

2.3.2 Comparison with state-of-the-art methods

Once we had established the utility o undersampling and SMOTE-N, we wanted to compare whether these methods were able to improve the results of the best performing method so far, the stop codon method [59]. This method consists of looking at the stop codon frequencies downstream of the TIS. The rationale for this approach is the following: TIS are characterized by the fact



C4.5 as base learner



SVM as base learner

Figure 2.8: ROC curve for ensemble methods and CCDS, *Ustilago* and *Arabidopsis* datasets.

that they represent the start of the first exon, so we know the reading frame of the first exon. In general, the first exon will have a minimal length, and thus there will be a minimal amount of sequence downstream of the TIS that does not contain an in-frame stop codon. On the other hand, pseudo TIS will not have this constraint, so the presence of in-frame stop codons can be used to discriminate between true and pseudo TIS. A very simple predictor can now be constructed that looks at the region following a putative TIS for the occurrence of in-frame stop codons. The earlier an in-frame stop codon occurs in this region, the less likely it is that the putative TIS is a true TIS. To obtain a simple scoring function that constructs a classifier out of this observation, Saeys et al. calculate the (cumulative) probability of observing an in-frame stop codon for the positive examples in the training set. It turns out that there is a significant difference in the cumulative distributions of the in-frame stop codons in both datasets. Then, for each testing example, the method scans the downstream part of the sequence until it finds an in-frame stop codon. For this first occurrence of an in-frame stop codon, the position x is recorded, and the model checked to find the probability of having a first in-frame stop codon at position x following a true TIS.

This last comparison shows the results of the proposed method with the method based on stop codon frequencies as baseline method. We also wanted to test whether the rationale of stop codon method applied to genomes as different from human genome as *Ustilago maydis* and *Arabidopsis thaliana*. Figures 2.9, 2.10, and 2.11 show the comparison between the baseline stop codon method and the best performing method for each classifier.

We made the comparison using two numerical values. Firstly, we consider the testing error. To compare the different methods, we set a sensitivity of 80% and measure the specificity at that level of sensitivity. We have chosen this sensitivity level as it is the one used by Saeys *et al.*[59]. Figure 2.12 shows a bar plot and numerical values for that specificity for all the methods. As a second method, we used the area under the curve (AUC) of the ROC curves shown above.

As described, AUC is a good value for comparing the overall behavior of the different algorithms. Numerical values and a bar plot of AUC results are shown in Figure 2.13. The first interesting result shows that stop codon method performed very well for *Ustilago* and *Arabidopsis* datasets, in the same way it obtains good results for CCDS dataset.

Both values show the same behavior. We see that stop codon method is among the best performing ones, which is a great achievement if we consider

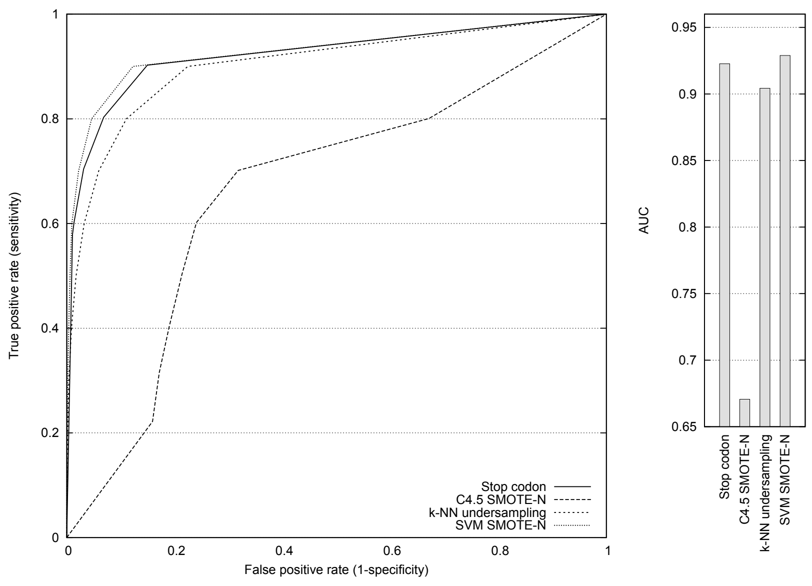


Figure 2.9: ROC curve for C4.5, k-NN and SVM and *Arabidopsis* dataset using the best performing method for dealing with imbalanced datasets and stop codon method.

that it is very simple. These values also corroborate that the classifiers applied without class imbalance methods perform poorly. On the other hand, the application of even a simple method as undersampling improves the performance very significantly. In fact, SVM with undersampling or SMOTE-N is able to beat stop codon method for all three datasets in terms of AUC values. k-NN with either undersampling or SMOTE-N beats stop codon method for *Ustilago* dataset. Although undersampling is also useful for C4.5 classifier, the poor performance of C4.5 without undersampling prevents the results of C4.5 and undersampling or SMOTE-N of being useful, even after the improvement obtained.

Ensemble methods are especially useful for improving the performance of its base classifier when we use a decision tree as base learner. Both, easy and balance cascade ensembles, obtain a very good performance that match the performance of stop codon method for CCDS and *Ustilago* datasets, and its clearly better that the performance of their base classifier, C4.5.

In addition to the comparison using the AUC we have also performed a statistical test to assure whether the observed differences are statistically

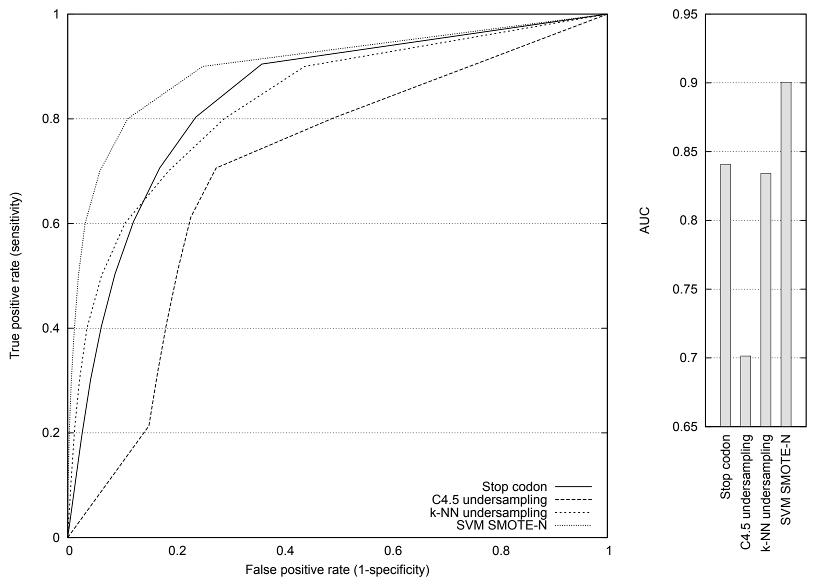


Figure 2.10: ROC curve for C4.5, k-NN and SVM and CCDS dataset using the best performing method for dealing with imbalanced datasets and stop codon method.

significant. As we are comparing results of different methods on the same problems using 10-fold cross-validation, we have chosen the corrected resampled t-test [49]. We have compared the specificity results of the different methods at a sensitivity level of 80%. Table 2.1 shows the results of this comparison at a confidence level of 95%. The table shows that the proposed class imbalance methods are able to significantly improve the results of the base method. In fact, SVM with either undersampling or SMOTE-N, is able to perform significantly better than stop codon method for all three datasets. The differences are specially marked for CCDS and *Ustilago* datasets.

However, there is second important issue in any real-world task, execution time. Methods must not only be efficient for the solution of the problems, they also have to solve the problem in a reasonable amount of time. Table 2.2 shows the training and testing time for all the used methods. We have separated the execution time in training and testing time because testing time is usually more relevant than training time, as training is performed off-line. The table shows that although the training time is always longer for the class imbalance methods the total time is within reasonable bounds.

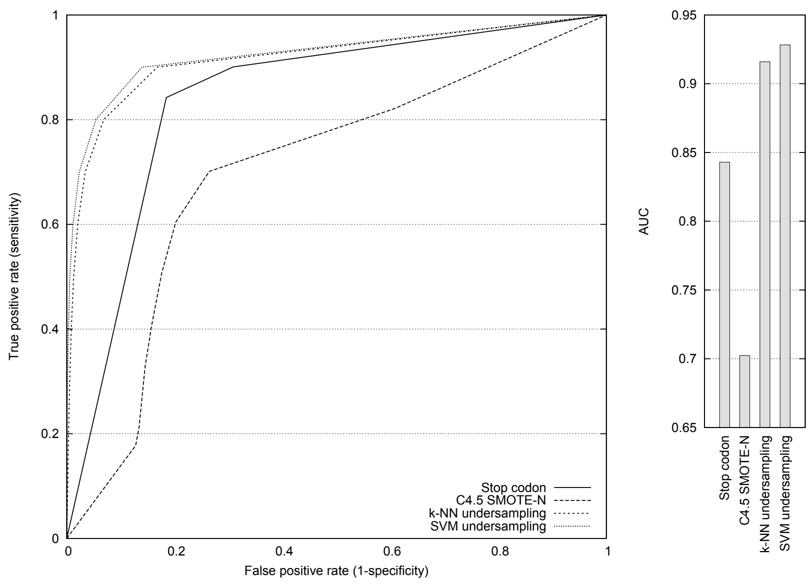


Figure 2.11: ROC curve for C4.5, *k*-NN and SVM and *Ustilago* dataset using the best performing method for dealing with imbalanced datasets and stop codon method.

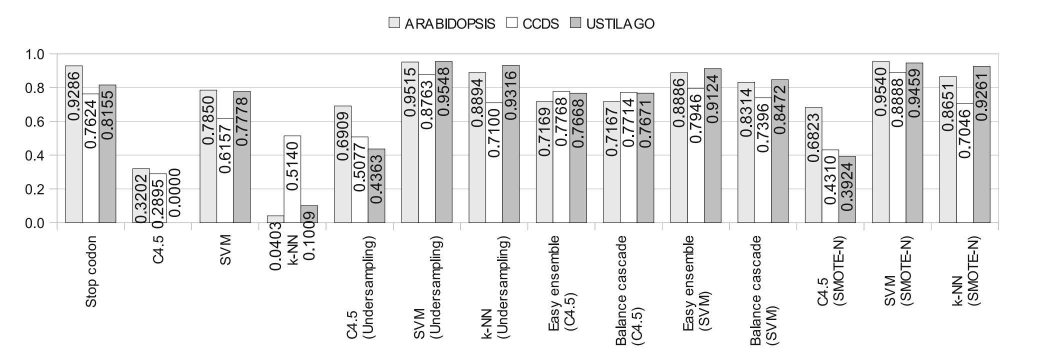


Figure 2.12: Specificity at 80% sensitivity for *Arabidopsis*, CCDS and *Ustilago* dataset using all the methods.

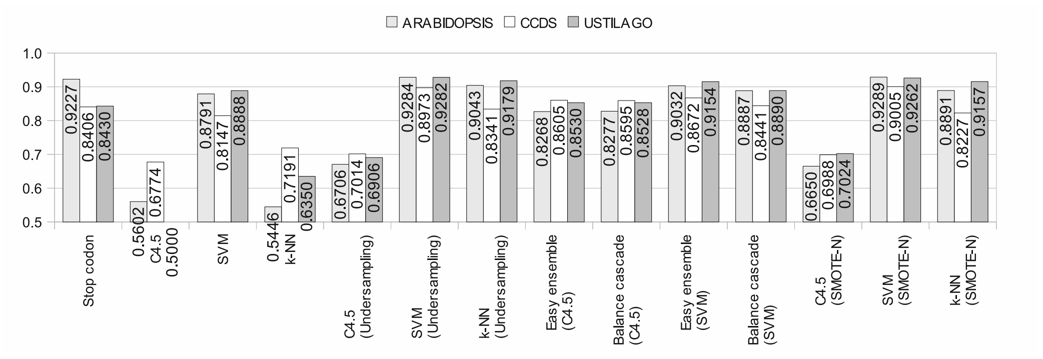


Figure 2.13: Area under the curve (AUC) for ROC curves for *Arabidopsis*, CCDS and *Ustilago* dataset using all the methods.

Table 2.1: Comparison between stop codon method and the methods studied.

Stop codon method vs.	CCDS	*Ustilago*	*Arabidopsis*
C4.5	✗	✗	✗
C4.5 + undersampling	✗	✗	✗
C4.5 + SMOTE	✗	✗	✗
k-NN	✗	✗	✗
k-NN + undersampling	✗	✓	✗
k-NN + SMOTE	✗	✓	✗
SVM	✗	✗	✗
SVM + undersampling	✓	✓	✓
SVM + SMOTE	✓	✓	✓
EasyEnsemble (C4.5)	—	✗	✗
BalanceCascade (C4.5)	—	✗	✗
EasyEnsemble (SVM)	✓	✓	✗
BalanceCascade (SVM)	✗	✓	✗

A ✓ marks results significantly better that stop codon method, and ✗ marks results significantly worse, at a confidence level of 95%. A — means no significant differences.

The worst case took less than 20 hours, which is a good result for these huge datasets. Furthermore, testing time is even shorter. For SVM, the best overall method, the worst case is 35,384 seconds. If we take into account that this value is for a test set of 339,721 instances, each sequence is evaluated fast and the application to the gene recognition task of this classifier is not prevented by its computational cost.

Table 2.2: Average training and testing times in seconds for all the methods and the three datasets.

Method	CCDS		*Ustilago*		*Arabidopsis*	
	Training time	Testing time	Training time	Testing time	Training time	Testing time
Stop codon	1	1	1	1	1	1
C4.5 + undersampling	7	1	3	2	28	9
C4.5 + SMOTE	276	1	72	1	959	9
k-NN + undersampling	1256	1916	276	1503	5171	37289
k-NN + SMOTE	20851	7828	4456	6190	4230	43145
SVM + undersampling	3453	2561	751	2054	11832	35384
SVM + SMOTE	61967	8601	9006	3008	32456	31234

2.4 Summary

In this chapter we have shown that the methods developed to cope with class imbalance problems can be useful for approaching TIS recognition problems. We have tested four methods designed to deal with class imbalance problems. These methods are able to improve the results of stop codon method, which it is the best reported in the literature [59]. The achieved results are very interesting, as TIS recognition is a difficult and relevant task within the field of gene structure prediction.

The results show that simple random undersampling is a very competitive method when compared with more complex ones. SMOTE-N achieved a good performance, but the improvement over random undersampling is only marginal if it exists at all. On the other hand, the performance of `EasyEnsemble` and `BalanceCascade` is remarkably good. These two ensemble methods improve the results obtained with undersampling and SMOTE-N, obtaining a performance comparable with stop codon method even from a base learner as C4.5 whose results as a single classifier were poor.

Furthermore, these methods, undersampling and SMOTE-N, are also useful for allowing the application of complex learning methods, such as SVMs, to large problems. Although stop codon method is very efficient and accurate for TIS recognition, it cannot be extended to the recognition of other sites, such as splice sites or stop codons. On the other hand, the proposed methodology can be applied to any of these problems in a straightforward manner.

Chapter 3

Feature selection for site recognition

Functional site prediction in genome sequences has two important characteristics from the machine learning perspective: firstly, as presented in Chapter 2, negative instances outnumber positive instances by many times, i.e., it is a class imbalance problem, and secondly, there are usually many features to describe each sequence. This Chapter explains the dealing with the second aspect and study the effect of the number of features on the classification accuracy. Additionally, the results of the application of feature selection techniques and their effect on the performance of the model are shown.

Many classifiers find problems in site recognition task due to the large amount of features involved. Most previous works used an arbitrary window of nucleotides around the putative site. In this part, we present a more principled way of selecting the relevant features using feature selection techniques. The method is designed to proceed in a two step procedure. First, it selects the most suitable window for each organism, and then it is carried out a selection of a subset of the most relevant features within that window.

This chapter summarizes the most important aspects of feature selection, shows the experimental setup of applying feature selection for site recognition and the results obtained on it.

3.1 Feature selection

Feature selection has been a fertile field of research and development since 1970's in statistical pattern recognition, machine learning, and data mining, and widely applied to many fields such as text categorization, image retrieval, customer relationship management, intrusion detection, and genomic analysis. Feature selection can be defined as the selection of a subset of M features from a set of N features, $M < N$, such that the value of a criterion function is optimized over all subsets of size M [50]. The objectives of feature selection are manifold, the most important ones being [59]:

- To avoid over-fitting and improve model performance, e.g. prediction performance in the case of supervised classification and better cluster detection in the case of clustering.
- To provide faster and more cost-effective models.
- To gain a deeper insight into the underlying processes that generated the data.

However, the advantages of feature selection techniques come at a certain price, as the search for a subset of relevant features introduces an additional layer of complexity in the modeling task.

In the context of classification, feature selection techniques can be organized into different categories. The output type divides feature selection algorithms into two groups: ranked list and minimum subset. The real difference between the two is about the order among the selected features. There is no order among the features in a selected subset. One cannot easily remove any more features from the subset, but one can do so for a ranked list by removing the least important one. If we focus on how they combine the feature selection search with the construction of the classification model we can identify three categories [59]: filter methods, wrapper methods, and hybrid/embedded methods:

1. Filter techniques rely on the intrinsic properties of the data to evaluate and select feature subsets without involving any mining algorithm. Advantages of filter techniques are that they easily scale to very high-dimensional datasets and that they are computationally simple, fast, and independent of the classification algorithm. As a result, feature

selection needs to be performed only once, and then different classifiers can be evaluated.

2. Wrapper methods embed the model hypothesis search within the feature subset search. In this setup, a search procedure in the space of possible feature subsets is defined, and various subsets of features are generated and evaluated. To search the space of all feature subsets, a search algorithm is then *wrapped* around the classification model. However, as the space of feature subsets grows exponentially with the number of features, heuristic search methods are used to guide the search for an optimal subset. The evaluation of a specific subset of features is obtained by training and testing a specific classification model. Their advantages include the interaction between feature subset search and model selection, and the ability to take into account feature dependencies. A common drawback is that they have a high risk of over-fitting and are very computationally intensive.

3. Hybrid/embedded techniques attempt to take advantage of the two models by building the search for an optimal subset of features into the classifier construction. Just like wrappers, they are specific to a given learning algorithm. Embedded methods have the advantage that they include the interaction with the classification model, while at the same time being far less computationally intensive than wrapper methods.

3.1.1 Feature selection for functional site recognition

In order to choose an appropriate algorithm for our problem we have to bear in mind the special characteristics of site recognition. First, its class imbalance nature. To avoid the problems derived from this fact, we have performed an undersampling step before applying any learning algorithm. We applied undersampling as it was the overall best performing method in our previous study. Then, we must consider the large number of features in the datasets. This size prevents the use of wrapper approaches due to their computational cost. Then, we must choose a filter approach. Furthermore, the method used must be able to cope with many features. With all these constraints we have selected SVM-RFE as the best choice.

Support vector machine recursive feature elimination (SVM-RFE) method is well-studied for use in gene expression problems [29]. This algorithm conducts feature selection in a sequential backward elimination manner, which starts with all the features and discards one feature at a time –it is a greedy algorithm –. Just like SVM, SVM-RFE was initially proposed for binary problems. The squared coefficients: $w_j^2(j = 1, ...; p)$ of the weight vector w are employed as feature ranking criteria. Intuitively, those features with the largest weights are the most informative. Thus in an iterative procedure of SVM-RFE one trains the SVM classifier, computes the ranking criteria w_j^2 for all features, and discards the feature with the smallest ranking criterion. One can consider that the removed variable is the one which has the least influence on the weight vector norm. The procedure provides as output a ranking of the features.

3.2 Experimental setup and results

As learning algorithm we have used a support vector machine. In a previous work we found SVM to achieve the best overall results [22] when compared with other widely used classification methods. SVM is very sensitive to its learning parameters, especially C, and γ in case of a Gaussian kernel. Thus, we have carried out a cross-validation procedure for obtaining values for these two parameters. We tried a linear kernel with $C \in \{0.1, 1, 10\}$, and a Gaussian kernel with $C \in \{0.1, 1, 10\}$ and $\gamma \in \{0.0001, 0.001, 0.01, 0.1, 1, 10\}$, testing all the 21 possible combinations. As the optimal values of these parameters depends on the training set, the cross-validation process is repeated each time a SVM is trained.

As previously stated in Chapter 2, site recognition is a class-imbalance problem and accuracy is not a useful measure. We used the developed measures that take into account the imbalanced nature of the problem described on Section 2.1.3.

3.2.1 Datasets

We have used three datasets for testing the performance of the described methods. The CCDS dataset was compiled by Saeys et al. [59] from the consensus CDS database. The CCDS project is a collaborative effort of compiling and identifying a core of human genes that are accurately annotated. The annotation is a mix of manual curation and automatic annotation. The CCDS dataset contains 350,578 negative samples and 13,917 positive samples with a positive/negative ratio of 1:25. The *Ustilago* dataset is a set of coding and non-coding regions of genomic sequences from the sequencing of the fungus *Ustilago maydis*. The sequences are first obtained from the Broad Institute[1] and then completed with the information of the Munich Information Center for Protein Sequences (MIPS)[2]. The *Ustilago* dataset contains 607,696 negative samples and 6, 515 positive samples with a ratio of 1:93. The *Arabidopsis* dataset comprises coding and non coding regions of the plant *Arabidopsis thaliana* obtained from "The *Arabidopsis* Information Resource" (TAIR)[3]. This dataset contains 27,342 positive instances and 3,369,875 negatives instances with an imbalance ratio of 1:123.

Our aim was to test the proposed methodology on very different datasets to study whether the problem is more difficult depending on the organism. Thus, we used these three datasets because they correspond to very different species. The CCDS dataset contains human DNA, thus it has long genes, with many exons and potentially long introns. On the other hand, the *Ustilago* dataset contains shorter genes, with very few exons, usually only one or two, and few and shorter introns. *Arabidopsis* complexity is between these two organisms. With these datasets, we can study the behavior of the proposed methodology in different environments.

In this case, the collected data is just related with the Translation Initiation Site, althought the proposed methodology is enough general as to be applied to any other functional site prediction problems. For all datasets we consider a sequence of 500 bps upstream and downstream of every ATG codon. The SVM classifier needs numerical attributes, thus we used a 1 out of 4 codification of each element of the sequence for a total of 4012 inputs.

Our first task was to obtain a working window for classifying the sequences. We chose an initial window large enough to assure that all needed information is contained in it. We selected sequences with 500 bps upstream and downstream of the actual or putative TIS. With these sequences we applied SVM-RFE method to rank the inputs. However, we must take into account that SVM-RFE works with decoded inputs. So we have to find a method that processes the 4012 original inputs in a way that allows us to measure the relevance of the 1003 corresponding nucleotides. To obtain that measure, we assigned to each nucleotide in the sequence a ranking equal to the average of the rankings of each of the 4 variables that codify the

[1]`http://www.broadinstitute.org/annotation/genome/ustilago_maydis/`

[2]`http://www.helmholtz-muenchen.de/en/mips/home/index.html`

[3]`http://www.arabidopsis.org/`

nucleotide. To make the information obtained manageable we grouped the relevance of the variables in 40 groups. These results are shown in Figure 3.1.

The first thing showed by the results is that the relevance of the variables is highly dependent on the organisms. It is very clear that the profile of the plots is very different for human, *Arabidopsis thaliana* and *Ustilago maydis* genomes.

The profile of *Ustilago maydis* shows that most of the downstream part is relevant. The ustilago dataset has genes with few exons and introns. Due to the fact that the are few introns, the downstream sequence is usually an exon with no introns, or at most one intron. Thus, it is a coding region and significant for the classification of the sequence. For the human genome, CCDS dataset, the situation is different. Human genome has more complex genes, and the first exon is usually disrupted by an intron after a short sequence. In this way, the downstream part is not all coding, and after the first 50 codons, the significance of the features decreases. For *Arabidopsis thaliana*, we found a behavior between these two.

With this first results we determined a sequence window to proceed with our second experiment. This window must consider a smaller sequence and at the same time include most of the significant features. We chose a sequence of 250 bps for the three genomes. For CCDS and *Ustilago maydis* we selected a window of 100 bases downstream and 150 upstream, and for *Arabidopsis thaliana* of 50 bases upstream and 200 downstream. With these windows we performed a second round of feature selection to study the relevancy of each base in the sequence. The results are shown in Figure 3.2 for each base in the sequence window.

For the CCDS dataset we found interesting results. First, most of the useful information is concentrated in the coding part, as should be expected. Furthermore, the relevance of the features follows a very marked sequence of period three, always assigning less relevancy to the middle base in the codon than the first base and the most relevant is always the last base. This is very interesting and further research is ongoing for a complete explanation of this fact. The ustilago has a more even distribution of the relevance of the features but it also exhibits the periodicity of CCDS in the coding part. However, for *Ustilago maydis* the first base of the codon is always the less informative, and the middle one the most relevant. For *Arabidopsis thaliana*, the downstream sequence is almost irrelevant. The periodicity in the coding region is also observed with the middle base as the most important. A comparison of the

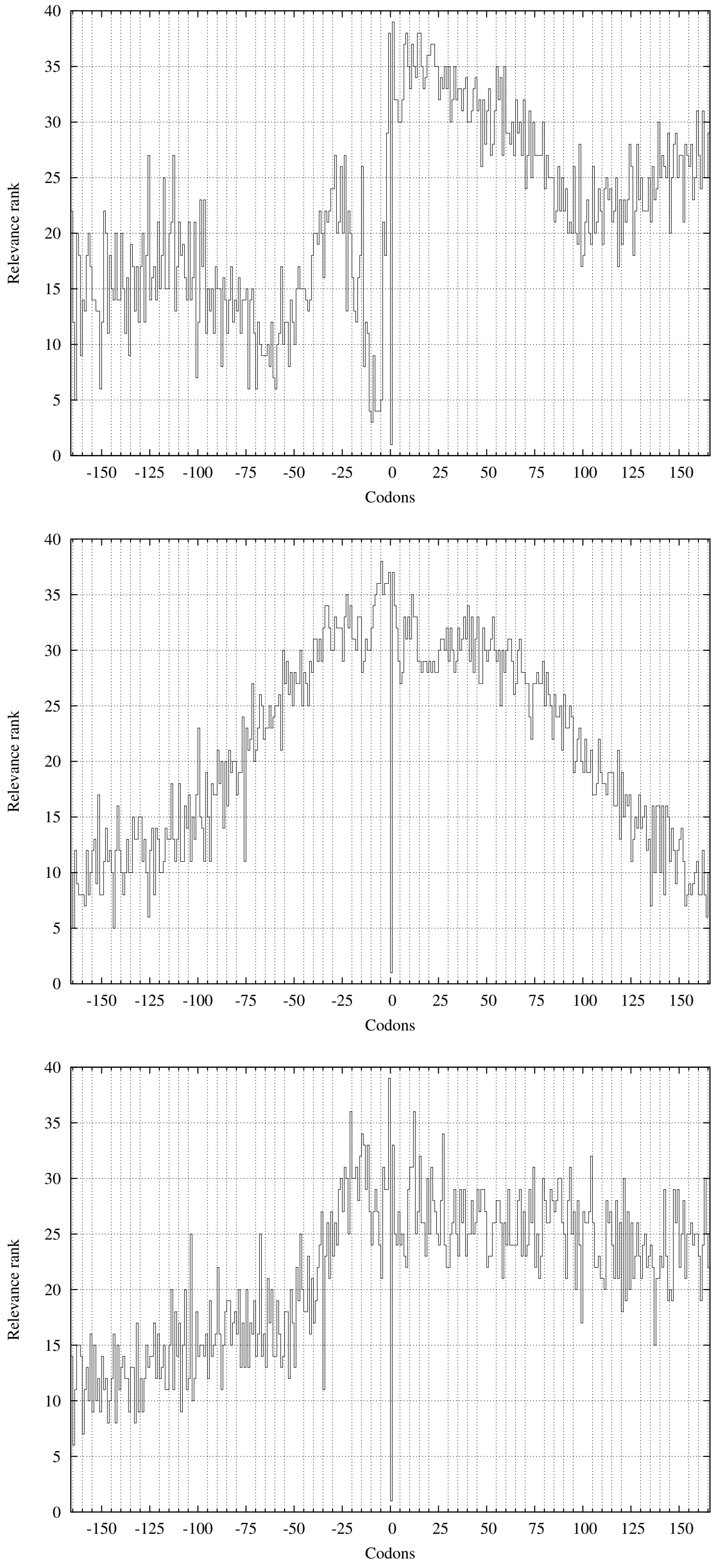


Figure 3.1: Codon relevance for a sequence of 1000 bps for, from top to bottom, *Arabidopsis*, CCDS and *Ustilago*

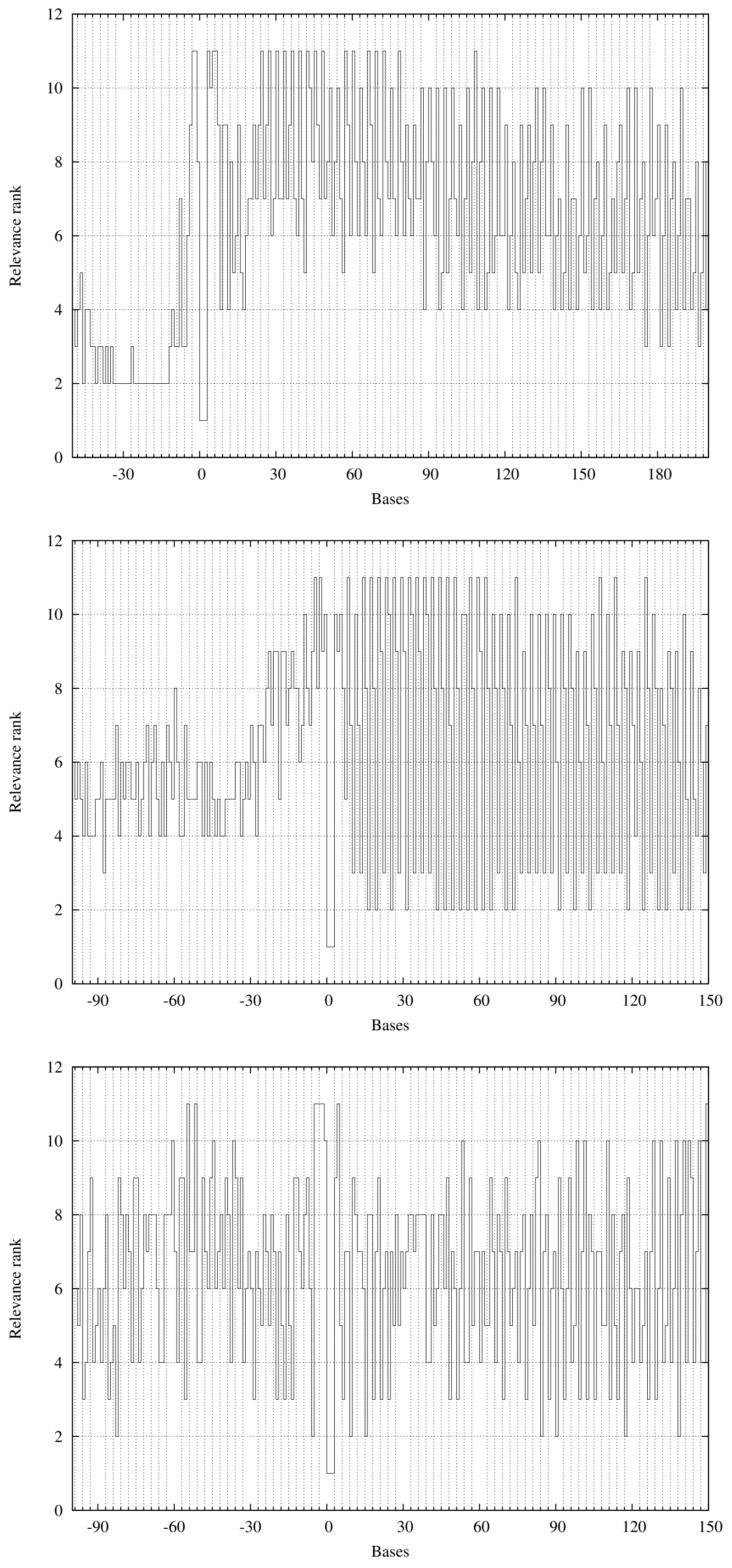


Figure 3.2: Base relevance for a sequence of 250 bps around the TIS for, from top to bottom, *Arabidopsis*, CCDS and *Ustilago*

performance using this window and the whole sequence is shown in Figure 3.3. It is clear that using the reduced window the classification is improved.

[htbp]

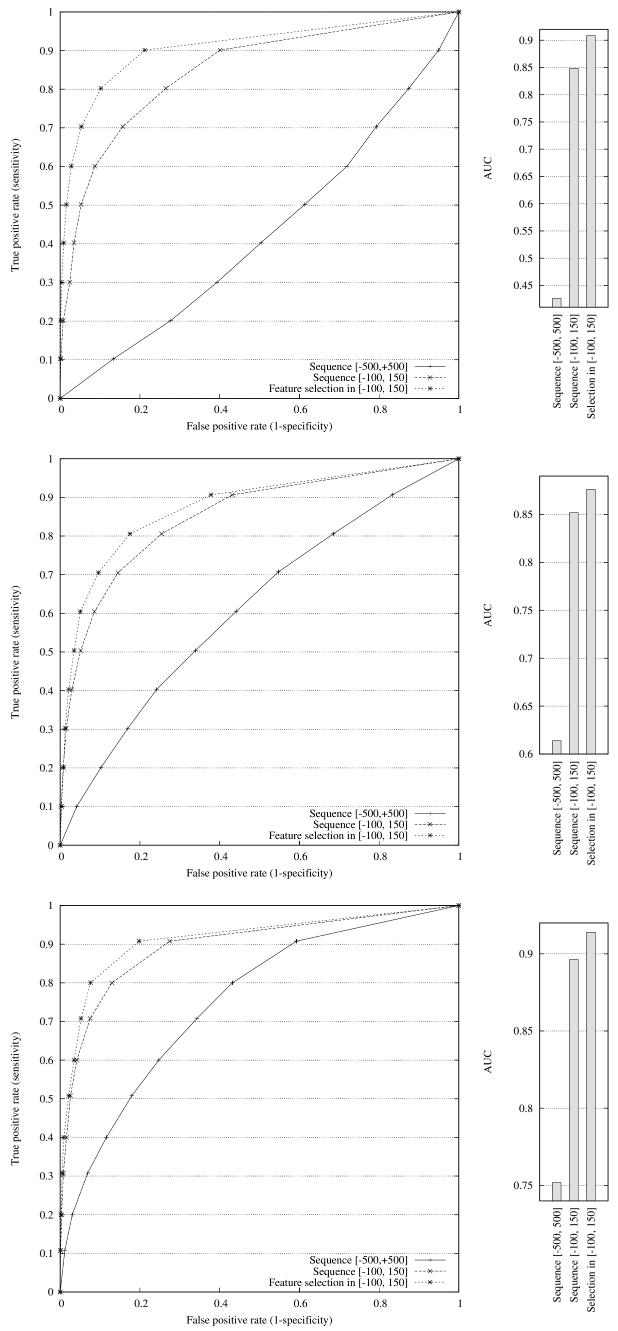


Figure 3.3: ROC curves and AUC values for, from top to bottom, *Arabidopsis*, CCDS and *Ustilago*

Our last step was to obtain the subset of most relevant features within the window and test its performance. However, SVM-RFE outputs a ranking of the variables and not a selection. As we want to compare the performance of feature selection with no feature selection, we performed a cross-validation approach for selecting the most appropriate number of retained features according to the ranking order provided by SVM-RFE in the classification process. A comparison of the three different groups of features is shown in Figure 3.3 for the three datasets. The figure shows the ROC curves and corresponding AUC. The results show the improvement in performance of feature selection against using all the variables. AUC is improved for the three problems with fewer features.

3.3 Summary

In this chapter we have shown that feature selection methods are a useful tool for improving the performance of TIS recognition. In a first step we have improved the performance of the classifier reducing the whole sequence to a smaller sequence. Then, we have carried out a second feature selection process to select the most interesting features within that smaller sequence. The performed selection has achieve better results in terms of AUC. In this way, SVM-RFE has shown is ability to, first select the correct window for learning the classifiers, and them to obtain a subset of features able to improve the results of all of them.

Our current research is centered in the explanation of the periodicity exhibited by the coding part of the sequences in the relevancy of the features.

Chapter 4

Site prediction using more than two classes

As previously mentioned, the site recognition is one of the most critical tasks for gene structure prediction. Most successful current gene recognizers first implement a step of site recognition [27], which is followed by a process of combining the sites into meaningful gene structures. This first step is of the utmost importance because the program cannot find genes whose functional sites are not identified. Furthermore, a large number of false positives might inundate the second step of the programs, making it difficult to predict accurate gene structures.

The best current approaches use powerful classifiers, namely support vector machines (SVMs), and moderately large sequences around the functional site [73, 17, 4, 65]. In accordance with common practices in machine learning, these methods construct a positive instance set using sequences that contains true sites and a negative instance set. In the case of TIS, in the negative instance set, the sequences centered around an ATG triplet are not TISs. In the stop codon recognition, the negative set are the sequences centered around TAG, TAA or TGA triplets are not stop codons. The negative sequences are obtained from all the available information or are randomly selected when sampling is used [25]. Thus, negative sequences can be part of intergenic regions, introns, exons, UTRs, etc.

However, the negative sequences from these different regions have different features. Therefore, the negative class, which the classifier must learn, is highly non-homogeneous. This inhomogeneity is an unnecessary difficulty that the learning algorithm must face and that might damage its performance. In this chapter, we show how the performance of the classifier can be actually improved if the negative instances are divided into different classes based on their position in the gene; subsequently, different classifiers are learned for each pair of positive and negative instance sets.

Some previous works have also consider the idea of differentiating between functional sites before proceeding to their recognition. TriTISA [32] is a method for detecting TISs in microbial genomes that classifies all candidate TISs into three categories based on evolutionary properties, and characterizes them in terms of Markov models. Also, other methods [11], have developed different models depending on the structure and composition of the sequences to recognize. However, these approaches are different from ours, as these models are trained and used separately instead of combined as in our proposal.

4.1 Site recognition method by using more than two classes

As explained in the previous section, our approach is based on separating the negative sequences based on their position in the gene. The same methodology was used for TIS and stop codon recognition. Thus, five different sets are created. First, we create a set containing all sequences that contain positive instances. Then, four additional sets are created containing negative sequences; these sets vary based on the position in the gene of those negative sequences. One set was created for each of the three following types of sequences: exons, introns and intergenic regions. A fourth set of negative sequences was created using sequences from noncoding regions, that is, from introns and intergenic regions together. As stated, the aim of this partitioning of the negative set is to obtain more homogeneous negative sets.

In the second step, we must decide how to use these five sets of instances. A straightforward approach would be to use any classifier that can handle more than two classes. However, as mentioned, SVMs are the best performing classifiers for both TIS and stop codon prediction. Although multi-class methods have been developed for SVMs [31], two-class approaches usually outperform those methods [55]. Thus, we chose to train four different classifiers, with each classifier trained to differentiate between the positive class and one of the four different negative classes. This approach has the additional advantage that overwhelming evidence in the machine learning literature indicates that a combination of different learners very frequently outperforms methods using only one classifier [57].

Additionally, we use another method in our approach. This last method is the stop codon method. This method is chosen because it only uses positive sequences; thus, it is not affected by the problem of mixing in the different instances of the negative classes. The stop codon method [59] looks at either the stop codon frequencies downstream of the TIS for TIS recognition or the stop codon frequencies upstream of the actual stop codon for stop codon recognition.

After these two steps, we have five trained classifiers that must be combined to obtain a single value that tells us whether a certain sequence is a true site or not. To classify a new sequence, we obtain the output of each classifier and use those five outputs to predict the class of the sequence. There are many ways of combining the outputs of different classifiers [37], some with high complexity. However, in most cases, simple methods are not beaten by the most complex ones, and those simple methods are faster and less prone to over-fitting. The most common of these simple methods include the sum of the outputs, majority voting and the maximum output.

Given the outputs of the five classifiers, $c_1, \ldots, c_5$, and a threshold for each one of these classifiers, $t_1, \ldots, t_5$, the final answer of the classifier, $C(\mathbf{x})$, for a given sequence $\mathbf{x}$, is defined as follows. For the sum of outputs, the final answer is given by:

$$C(\mathbf{x}) = \sum_i c_i(\mathbf{x}) - t_i. \tag{4.1}$$

The threshold term, t_i, corrects for the different ranges of the classifiers. The majority voting approach is given by:

$$C(\mathbf{x}) = \arg \max_{y \in \{-1,+1\}} \sum_{y : c_i(\mathbf{x}) = y} 1. \tag{4.2}$$

Finally, the maximum is given by:

$$C(\mathbf{x}) = c_i(\mathbf{x}) : i = \arg\max_j |c_j(\mathbf{x}) - t_j|. \tag{4.3}$$

Once $C(\mathbf{x})$ is obtained by any of these methods, a general threshold T should be fixed to decide whether a certain sequence is an actual site or not. One of the problems we have when choosing the combination method is model selection as we do not know *a priori* whether any of the three methods would be consistently better that the other two for all the chromosomes and all the three evaluation measures we used as performance measures. Thus, the best combination was chosen for each case using cross-validation. This cross-validation method is explained in the next section.

As a final remark, we should note that our approach is also general enough to be used with any other classifier. Because this approach is based on modifying the number of classifiers and the training sets, it can be used with any other classification method. Furthermore, this method can also be applied if a classifier uses other types of data besides the raw sequence if the information used by the classifier is extracted using the datasets described above.

4.2 Evaluating the approach

To evaluate our approach, we chose five different human chromosomes, namely chromosomes 1, 3, 13, 19 and 21 for testing purposes, and chromosome 16 for model selection. For each chromosome, we trained the classifiers with all the remaining chromosomes excepting 16, then we chose the best combination method using chromosome 16 and tested the chosen model with all the true TIS or stop codons and the negative samples of the given chromosome. That is, for chromosome 1, we trained the models with chromosomes 2 to 22 and X and Y excepting 16. Then, we chose the best combination method using chromosome 16 and tested this model using chromosome 1. A summary of these datasets is shown in Table 4.1. The chromosomes were selected with the aim of choosing chromosomes of different lengths and codification density. Chromosome 16 was chosen as validation set as it is a chromosome of average length and coding density. For SVMWD, as no model selection is needed, chromosome 16 was added to the training set. We used all TISs and stop codons of the CCDS Update Released for Human of September 7, 2011. This update uses Human NCBI build 37.3 and includes a total of 26,473 CCDS IDs that correspond to 18,471 GeneIDs.

One of the key aspects of the evaluation of any new proposal is the set of previous methods used in the comparison. Many different methods have been proposed for recognizing TISs and stop codons [73, 72, 69, 59]. However, these previous works and our own research [25] have shown that a SVM with

Table 4.1: Summary of the training and testing sets

Dataset		Training data	Testing data	
		Positives/Negatives	Positives	Negatives
Chr. 1	TIS	17,638	2,156	8,074,590
	STOP	17,404	2,154	23,573,031
Chr. 3	TIS	18,631	1,163	7,291,951
	STOP	18,444	1,114	21,522,500
Chr. 13	TIS	19,454	340	3,664,164
	STOP	19,225	333	10,878,302
Chr. 19	TIS	18,383	1,411	1,698,891
	STOP	18,136	1,422	4,665,804
Chr. 21	TIS	19,561	233	1,303,634
	STOP	19,558	237	3,726,959

Random undersampling was used for training; thus, the number of negative instances was equal to the number of positive instances.

a string kernel is the best state-of-the-art method not only for TISs and stop codons but also for splice sites [65]. To assure the general advantage of SVMs with string kernels we performed a preliminary study of the different available methods that included position weight matrices, decision trees, k-nearest neighbors, stop codon method [59], Wang et al.'s method [69], Salzberg's method [60] and SVMs with linear and Gaussian kernels and three different string kernels: the locality improved (LI) kernel, the weighted degree kernel (WD) and the weighted degree kernel with shifts [58] (WDS). SVMs with WD kernel obtained consistently the best results and thus was chosen as the method to be compared with our proposal. WDS obtained marginally better results than WD but with a far higher computational complexity. We will refer throughout the chapter to the SVM with a WD kernels as SVMWD. The same WD kernel was used for the classifiers in our proposal. However, we must bear in mind that our method, as it works on the design of the datasets, can be used with any other classification method.

Another key parameter of the learning process is the window around the functional site that is used to train the classifiers. A further advantage of our approach is that it allows the use of a suitable window for each type

Table 4.2: Summary of the window cross-validation

Data	Positives vs. Negatives in					
(chr)	All	Exons	Introns	Intergenic regions	Noncoding regions	Stop codon
TIS						
1	[-50,50]	[-50,50]	[-50,50]	[-50,50]	[-50,50]	[0,500]
3	[-25,75]	[-50,50]	[-25,75]	[-25,75]	[-25,75]	[0,500]
13	[-50,50]	[-50,50]	[-10,40]	[-10,40]	[-10,40]	[0,500]
19	[-25,75]	[-50,50]	[-25,75]	[-25,75]	[-25,75]	[0,500]
21	[-50,50]	[-50,50]	[-25,75]	[-10,40]	[-25,75]	[0,500]
STOP						
All	[-90,10]	[-90,10]	[-90,10]	[-90,10]	[-90,10]	[-500,0]

The window obtained around the functional site is shown for each classifier.

of sequence. The value of the window for each classifier was obtained by cross-validation. We considering the site as offset 0 and did not count the TIS or the stop codon, and we tested the performance of the following windows $[-100, 0]$, $[-75, 25]$, $[-50, 0]$, $[-50, 50]$, $[-25, 0]$, $[-25, 25]$, $[-25, 75]$, $[-10, 15]$, $[-10, 40]$, $[-10, 90]$, $[0, 25]$, $[0, 50]$ and $[0, 100]$. For each trained classifier, the best window was chosen. Table 4.2 shows the window obtained by cross-validation for all the classifiers. For the stop codon method, we used the additional window values of $[0, 200]$, $[0, 300]$, $[0, 400]$ and $[0, 500]$ for TIS recognition and the window values of $[-200, 0]$, $[-300, 0]$, $[-400, 0]$ and $[-500, 0]$ for stop codon recognition.

Table 4.2 shows interesting results. First, the window for TIS recognition depended on the classifier and the chromosome. However, the window for stop codon prediction was the same for all cases with only one exception. Second, this table also shows that the different classifiers used for TIS recognition had different values; this finding supports our previous claim that using different classifiers has the advantage of allowing better fine tuning of the learning parameters.

Furthermore, SVMs are very sensitive to the learning parameters; thus, we also performed a cross-validation to obtain their values. The WD kernel has two parameters: the standard C parameter of any SVM and the window width of the string kernel. We tested values of $1, 10, 100$ and 1000 for C and 12 and 24 for the window width. All 8 combinations were evaluated using 10-fold cross-validation, and the best one was chosen. Although it may be argued that this method might result in suboptimal parameters, this method is a good compromise between the performance of the SVM and the high computational cost of evaluating each set of parameters. This same procedure was used for both SVMWD and our approach.

For training the models, we used random undersampling [33] because previous studies have shown its usefulness for TIS recognition [25]. For random undersampling, we used a ratio of 1, which means that the majority class was randomly undersampled until both classes had the same number of instances.

To evaluate the obtained classifiers, we used the standard measures for imbalanced data explained on Section 2.1.3 . Given the number of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN), we used the sensitivity and the specificity. The geometric mean of these two measures will be our first classification metric. As a second measure we used the area under the receiver operating characteristic (ROC) curve (auROC). However, auROC is independent of class ratios and that it can be less meaningful when we have very unbalanced datasets [65]. In such cases, area under the precision recall curve (auPRC) can be used. This measure is specially relevant if we are mainly interested in the positive class. However, it can be very sensible to subsamplig. In our results we use all the positive and negative instances for each one of the five chromosomes tested, so no subsampling is used. This also yields to small auPRC values.

We use these three metrics because they provide two different views of the performance of the classifiers. The auROC and auPRC values describe the general behavior of the classifier. However, when used in practice, we must establish a threshold for the classification of a query pattern. G-mean provides the required snapshot of the performance of the classifier when we set the needed threshold.

4.3 Results and discussion

The first step of our experiments was devoted to studying the usefulness of the five different classifiers that we considered. As stated, we have five classifiers that are trained with the same positive class and a negative class consisting on negative instances from exons, introns, intergenic regions and noncoding regions. We had a fifth classifier using only positive instances. Thus, we tested the performance, as measured by the auROC, of the combined approach with all five classifiers and then removed one classifier at a time. A negative value means that the classifier had a negative effect on the performance of the model and thus should not be used.

The results showed that the worst performing classifier was the one trained using negative instances extracted form exons. For this classifier, the positive and negative instances were the most similar; thus, the training algorithm had more difficulties in differentiating between the positive and negative instances. In fact, the overall effect of this classifier was harmful to the performance of the method. Thus, this classifier was removed and was not considered in the subsequent experiments.

The results also showed that the stop codon method classifier had the most important contribution. This finding is interesting because this classifier was the worst when considered alone. The explanation for this difference may be found in the behavior of the ensembles of classifiers. It is well known [38] that a diverse ensemble of classifiers improves the performance of the set of classifiers. The stop codon method differs from the other four classifiers, which are all based on SVMs; thus, although its performance is worse than the performance of those four classifiers individually, the diversity it introduces improves the performance of the set of classifiers.

The next step was the comparison of the performances of our approach and SVMWD. A summary of the results for TIS recognition of the five studies chromosomes is shown in Table 4.3. The first interesting result is that the proposed approach beat SVMWD for all measures and all chromosomes with only one exception. The improvements in specificity, sensitivity, geometric mean, auROC and auPRC are shown in Figure 4.1.

The second remarkable result shown in Table 4.3 is the significant reduction in the FN rate. The reduction in the number of FNs was 9.3% in the worst case and 44.5% in the best case. This reduction means that 284 TISs that were inaccurately classified as negatives by SVMWD were correctly identified by our method. Most current gene recognizers rely heavily on the classification of TISs; therefore, it is very likely that those genes would be completely missed by any gene recognizer. Thus, our approach has the potential to improve the accuracy of any annotation system by 6.4%.

Furthermore, our method was also able to improve the true negative rate. In total, 145,780 false positives from SVMWD were correctly classified as negatives using our approach. Therefore, any annotation system that uses our metric would have a significantly reduced set of putative TISs and better expected performance.

Table 4.3: Summary of the results for TIS recognition.

Dataset	SVMWD						
	Sp	Sn	TP	FN	TN	FP	auROC/PRC
Chr. 1	.9155	.8326	1795	361	7392644	681946	.9481/.1001
Chr. 3	.9024	.8203	954	209	6580426	711525	.9357/.0891
Chr. 13	.9398	.8294	282	58	3443428	220736	.9522/.0818
Chr. 19	.8961	.8703	1228	183	1522340	176551	.9522/.1358
Chr. 21	.8965	.8326	194	39	1168732	134902	.9387/.0689
Dataset	**Proposed approach**						
	Sp	Sn	TP	FN	TN	FP	auROC/PRC
Chr. 1	.9209	.9003	1941	215	7435495	639095	.9693/.1351
Chr. 3	.9066	.9003	1047	116	6611176	680775	.9628/.1229
Chr. 13	.9457	.8824	300	40	3465327	198837	.9695/.1207
Chr. 19	.9077	.8824	1245	166	1542012	156879	.9551/.1321
Chr. 21	.9200	.8755	204	29	1199340	104294	.9691/.1203

The table shows the specificity (Sp), sensitivity (Sn), true positives (TP), true negatives (TN), false negatives (FN), false positives (FP) and area under the ROC and PRC curves (auROC/PRC) for both methods and the five studied chromosomes.

The improvement for auROC and auPRC[1] values are also shown in Figure 4.1. The actual ROC and PRC curves are shown in Figures 4.2–4.6. These figures show that our approach improved the auROC and auPRC for all five studied chromosomes. These results demonstrate that the overall performance of the proposed method was better than the performance of SVMWD. The actual ROC and PRC curves shown in Figures 4.2–4.6 show that the curves corresponding to our proposal are always above than the curves of SVMWD. This result indicates the better performance for all the possible thresholds of classification.

[1]We always performed the testing of all the methods with all the negative samples. That means that the ratio minority/majority class is almost 1:11000 for the worst case yielding to low auPRC values. We must take into account that with only a few thousands FPs among several millions of TNs we would obtain a very low precision value. The situation for stop codon recognition is even worse as the number of TNs is multiplied by three.

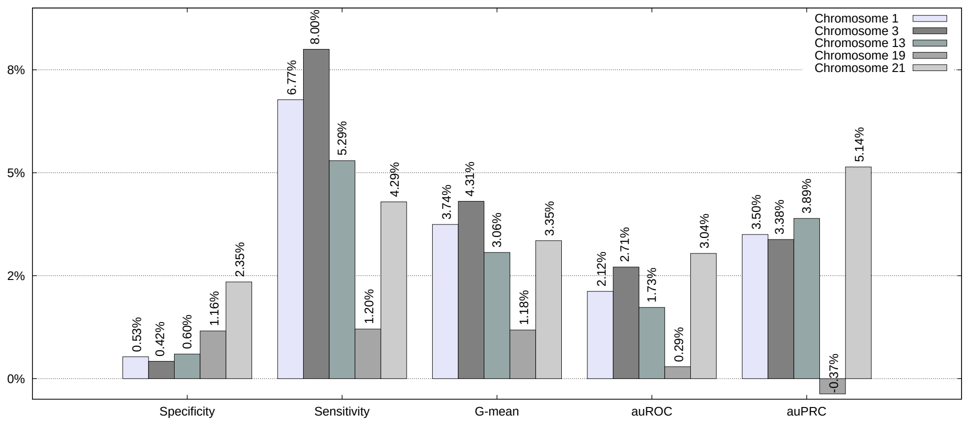


Figure 4.1: Absolute improvement for TIS recognition in specificity, sensitivity, G-mean, auROC and auPRC of our approach compared with SVMWD.

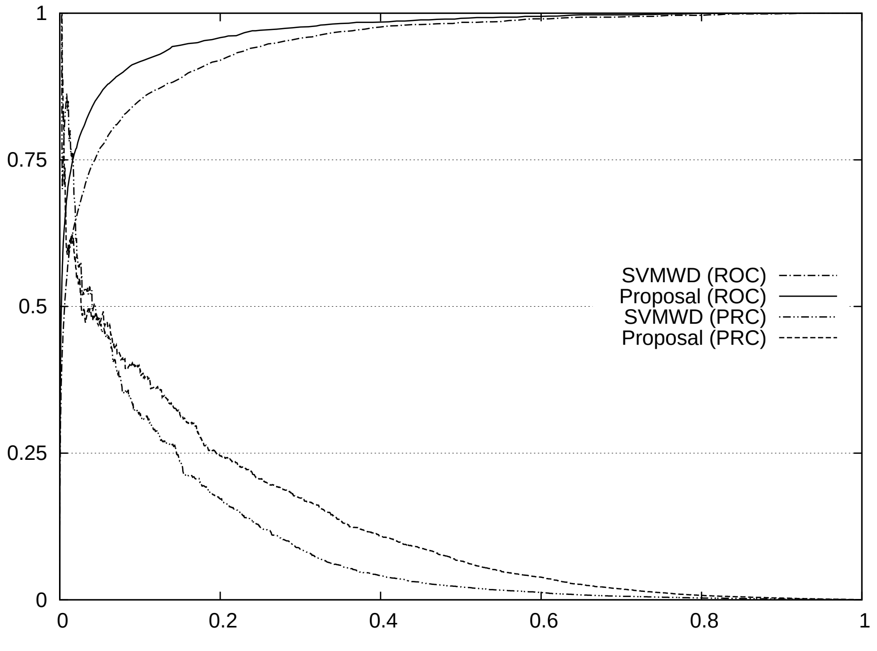


Figure 4.2: ROC/PRC curves for TIS prediction for chromosome 1.

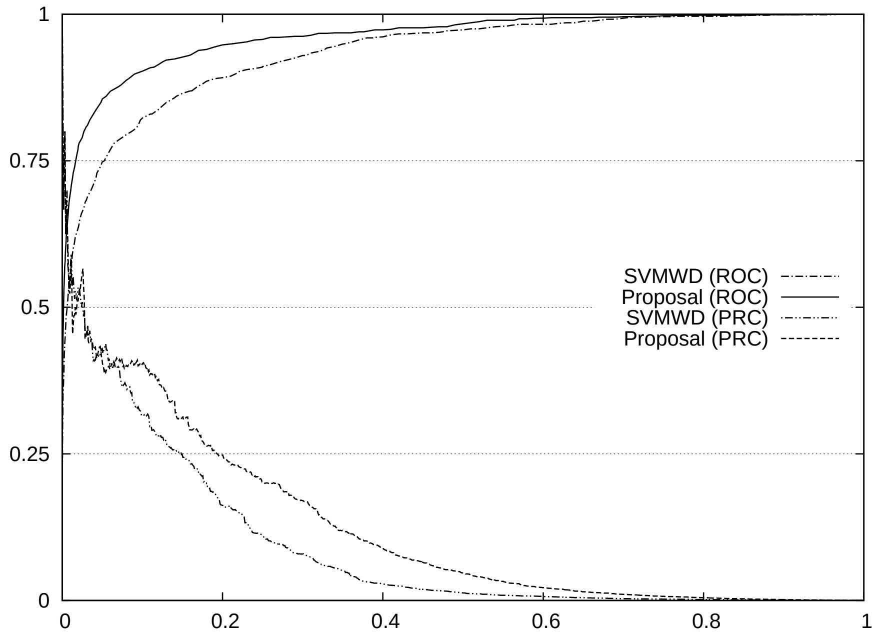


Figure 4.3: ROC/PRC curves for TIS prediction for chromosome 3.

It is interesting to study how the proposed method achieved its good performance. For both methods, Table 4.4 shows the distribution of the false positives according to the part of the gene to which the TIS sequences belong. The behavior is clear. Separating the negatives samples into four classes improves the discrimination between positive instances and negative instances from introns and intergenic regions. However, the number of false positives for instances from exons increases but to a lesser extent than the decrease in the number of false positives from introns and intergenic regions. Furthermore, the false positives from exons many be reduced using other sources of information, such as content measures.

The second part of our experiments was devoted to stop codon recognition. Stop codon recognition is a more difficult task because the achieved accuracy is less than that for TIS recognition. One of the major sources of this increased complexity is the number of negative instances. There are three different stop codons rather than just one as it is the case for TIS recognition; therefore, the number of negative instances is three times the number of negative instances for TIS prediction. For instance, using the same five

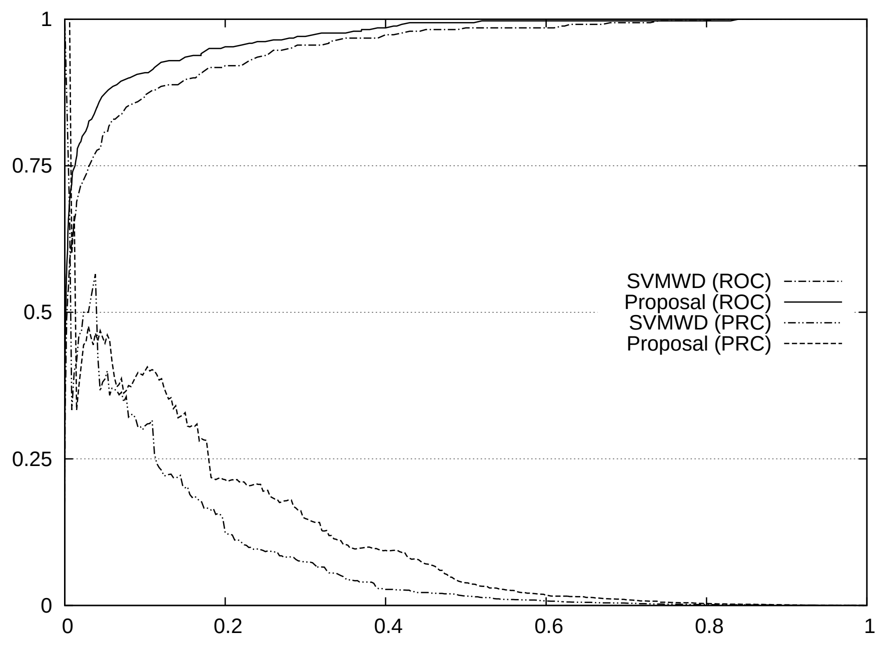


Figure 4.4: ROC/PRC curves for TIS prediction for chromosome 13.

Table 4.4: Distribution of false positives for both methods.

Dataset	SVMWD			Proposed approach		
	Exon	Intron	Intergenic regions	Exon	Intron	Intergenic regions
Chromosome 1	0.15%	1.39%	6.91%	0.21%	0.71%	3.72%
Chromosome 3	0.12%	1.58%	8.06%	0.13%	0.56%	4.24%
Chromosome 13	0.06%	0.79%	5.17%	0.07%	0.37%	2.79%
Chromosome 19	0.58%	1.55%	8.26%	0.71%	1.23%	6.54%
Chromosome 21	0.10%	1.30%	8.95%	0.13%	0.63%	4.05%
Average	0.20%	1.32%	7.47%	0.25%	0.70%	4.27%

The table shows the type of genome region for each false positive.

chromosomes from the previous experiments, the best current method found more than 11.5 million false positive stop codons. This amount of incorrectly predicted stop codons might be able to mar any annotation system,

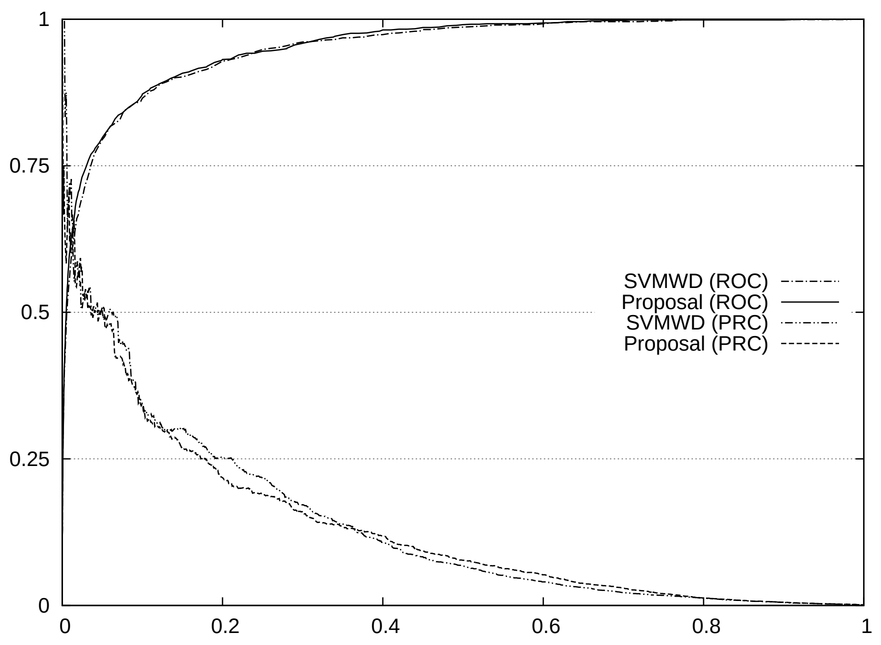


Figure 4.5: ROC/PRC curves for TIS prediction for chromosome 19.

indicating that there is ample room for improvement.

Our approach for stop codon prediction used the same classifiers as for TIS recognition. Table 4.3 shows a summary of results for stop codon recognition. The first remarkable result is the large improvement in the number of both FNs and FPs. The FNs were reduced by 26.8% in the worst case and by 57.9% in the best case. This result indicates that the number of total FNs was reduced from 823 with SVMWD to 443 with our method. As for TIS recognition, an annotation program may not be able to recognize a gene when the stop codon is missing. Furthermore, this improvement was achieved along with a significant improvement in the FPs. The FP improvement was also large with a best result of 46.9%. As a whole, 2.7 million FPs from SVMWD were accurately classified as negatives by our method. This quantity of false positives may overwhelm any annotation system; thus, the improvement should have a significant impact on automatic annotation.

Figure 4.7 shows the absolute improvement of our method in the specificity, sensitivity, G-mean, auROC and auPRC. The improvement for auROC is particularly relevant. The proposed approach improved the auROC

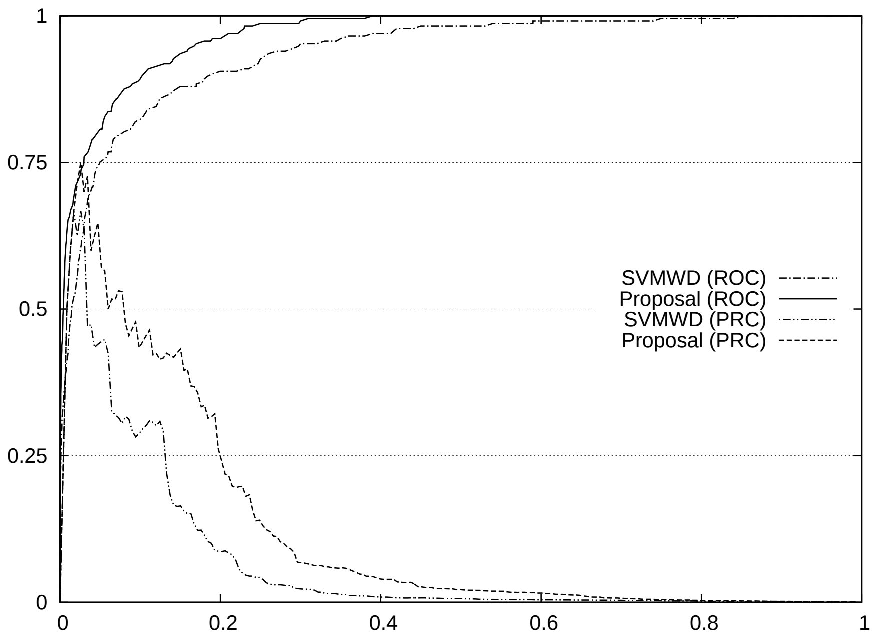


Figure 4.6: ROC/PRC curves for TIS prediction for chromosome 21.

by 3.7% in the worst case and 7.2% in the best case. Figures 4.8–4.12 show the ROC/PRC curves for the five chromosomes. As for TIS recognition, the ROC/PRC curves of our approach not only achieved a better auROC and auPRC but were also always above than the curves of SVMWD.

In Section 4.1, we stated that our approach could be applied to any type of classifier. In the previous experiments, we used SVMs because they achieved the best performance in the literature. Now, we present the results of another experiment that was conducted to demonstrate the applicability of our method to other classifiers. We used a decision tree using the C4.5 learning algorithm [54] instead of SVMs. We tested a decision tree using the standard approach of only one training set and one classifier and using our method with the four classifiers that were used for SVMs. To avoid repeating all the experiments, we only performed experiments for chromosome 13. The results for both TIS and stop codon recognition are shown in Table 4.6. For TIS recognition, the improvement was remarkable; the G-mean improved by 8%, and the auROC increased from 0.8189 to 0.9372. For stop codon classification, the improvement was even better. The standard approach had

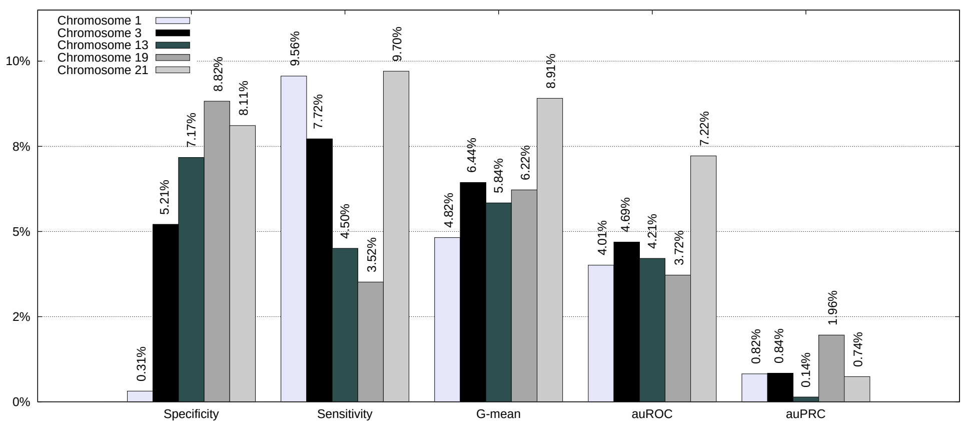


Figure 4.7: Absolute improvement for stop codon recognition in the specificity, sensitivity and G-mean of our approach compared with SVMWD.

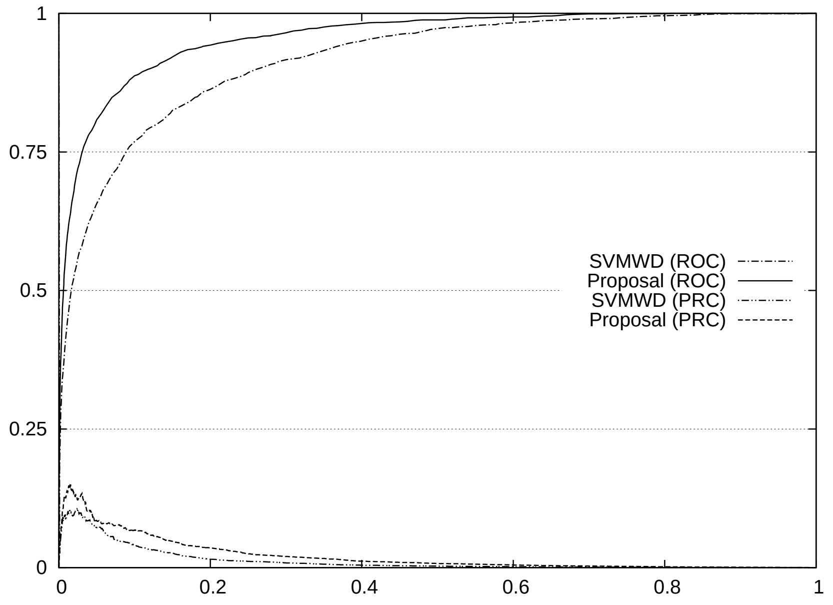


Figure 4.8: ROC/PRC curves for stop codon prediction for chromosome 1.

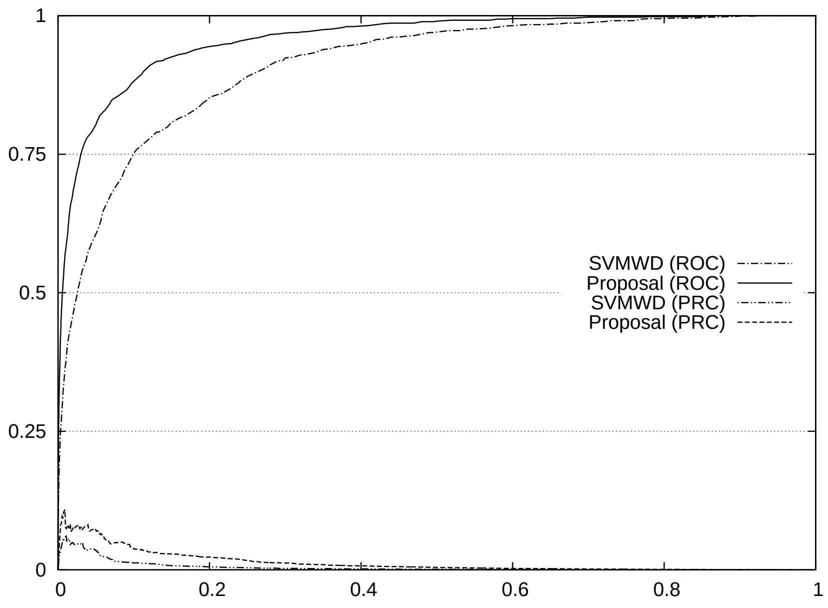


Figure 4.9: ROC/PRC curves for stop codon prediction for chromosome 3.

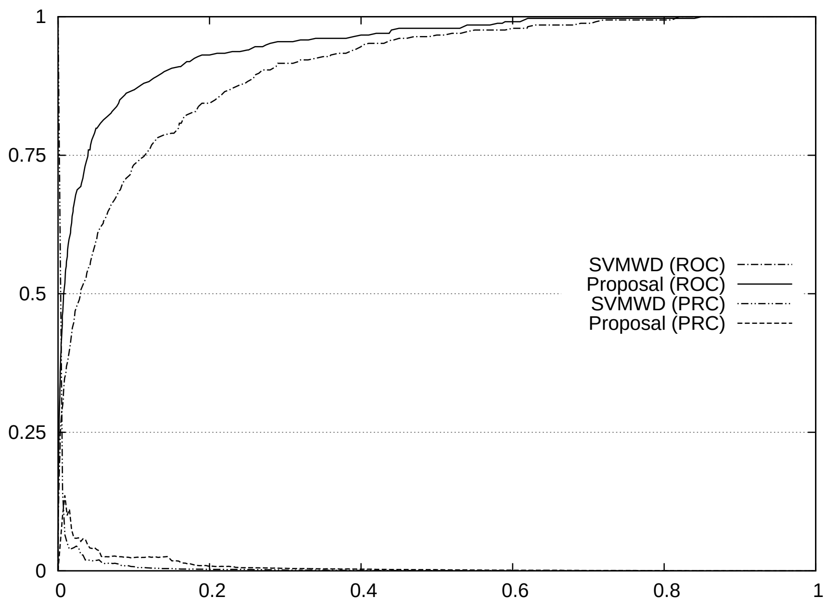


Figure 4.10: ROC/PRC curves for stop codon prediction for chromosome 13.

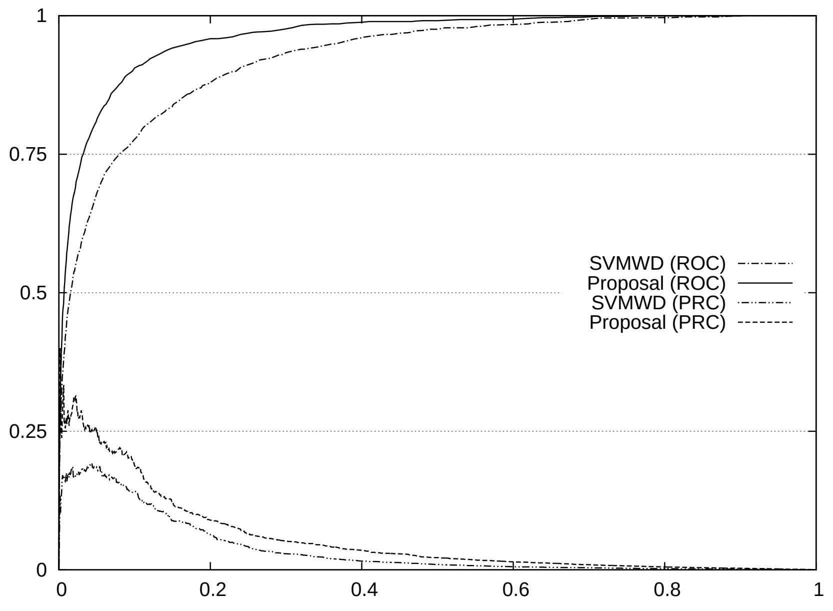


Figure 4.11: ROC/PRC curves for stop codon prediction for chromosome 19.

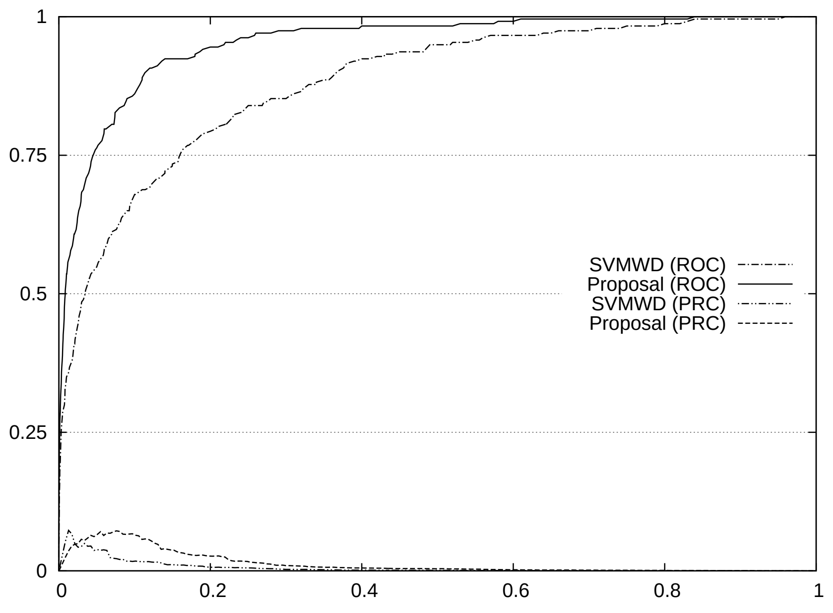


Figure 4.12: ROC/PRC curves for stop codon prediction for chromosome 21.

Table 4.5: Summary of the results for STOP codon recognition.

Dataset	SVMWD						
	Sp	Sn	TP	FN	TN	FP	auROC/PRC
Chr. 1	.8361	.8347	1798	356	19710568	3862463	.9182/.0127
Chr. 3	.8109	.8402	936	178	17452195	4070305	.9115/.0049
Chr. 13	.8183	.8318	277	56	8901404	1976898	.9073/.0067
Chr. 19	.8117	.8706	1238	184	3787466	878338	.9258/.0357
Chr. 21	.8089	.7932	188	49	3014762	712197	.8811/.0058
Dataset	Proposed approach						
	Sp	Sn	TP	FN	TN	FP	auROC/PRC
Chr. 1	.8393	.9304	2004	150	19784595	3788436	.9583/.0209
Chr. 3	.8630	.9174	1022	92	18573710	2948790	.9584/.0133
Chr. 13	.8900	.8769	292	41	9681647	1196655	.9494/.0081
Chr. 19	.9000	.9058	1288	134	4199221	466583	.9630/.0553
Chr. 21	.8900	.8903	211	26	3316990	409969	.9533/.0132

The table shows the specificity (Sp), sensitivity (Sn), true positives (TP), true negatives (TN), false negatives (FN), false positives (FP) and area under the ROC and PRC curves (auROC/PRC) for both methods and the five studied chromosomes.

an auROC of 0.6853, whereas our approach achieved an auROC of 0.9260. As it was the case for the previous experiments auPRC was very low for all experiments due to the huge number of negative instances.

4.4 Summary

In this chapter, we presented a new approach for TIS and stop codon recognition. This approach uses more than one classifier, divides the negative class into four different groups and trains one classifier for each type of negative class. This approach was applied to the recognition of TIS and stop codons in five human chromosomes. The approach was compared with the best current method for TIS and stop codon prediction. The proposed approach also has the advantage of its simplicity, which makes it easily applicable to any program for TIS or stop codon recognition.

The reported results show that the proposed method shows improved sen-

Table 4.6: Results for TIS and stop codon prediction for chromosome 13 using a decision tree as the classifier.

Dataset	Method	auROC/auPRC	Sp	Sn	G-mean
TIS	C4.5	0.8183/0.0005	0.7756	0.7706	0.7731
	Proposed	0.9372/0.0154	0.9053	0.8000	0.8510
Stop codon	C4.5	0.6853/0.0001	0.6489	0.6426	0.6458
	Proposed	0.9260/0.0097	0.8468	0.8709	0.8587

The table shows the values of the specificity (Sp), sensitivity (Sn), geometric mean of the specificity and sensitivity and the area under the ROC/PRC curves (auROC/auPRC).

sitivity, specificity, auROC and auPRC compared with SVMWD. The results show a remarkable improvement in the ratio of FNs and FPs achieved over those of SVMWD. Because state-of-the-art annotation systems rely heavily on the accurate prediction of the functional sites of the gene, the proposed method is an effective way of improving current gene recognizers.

Chapter 5

Stepwise approach for combining many of sources of evidence for site recognition

With the rapid evolution of our ability to collect genomic information, it has been shown that combining differences sources of evidence is fundamental to the success of any recognition task in Bioinformatics. The advent of next-generation sequencing has made possible the number of available genomes is increasing very rapidly. Thus, methods for making use of such large amounts of information are needed. In this chapter, we present a methodology for combining tens or even hundreds of different classifiers for an improved performance. Our approach can include almost a limitless number of sources of evidence. We can use the evidence for the prediction of sites in a certain species, such as human, or other species as needed. This approach can be used for any of the functional recognition tasks cited above, although in this work we have tested our approach in two functional recognition tasks: translation initiation site and stop codon recognition.

5.1 Background

As previously stated, the recognition of functional sites within the genome is one of the most important problems in Bioinformatics research. Determining where different functional sites, such as the promoters, translation start sites, translation initiation sites (TISs), donors, acceptors and stop codons, are located provides useful information for many tasks. For instance, the recognition of translation initiation sites, donor, acceptors and stop codons [73] is one of the most critical tasks for gene structure prediction. Most successful gene recognizers currently in use first implement a step of site recognition [27], which is followed by a process of combining the sites into meaningful gene structures. This first step is of the utmost importance because the program cannot find genes whose functional sites are not identified. Furthermore, a large number of false positives might inundate the second step, making it difficult to predict accurate gene structures. The best current approaches use powerful classifiers, namely support vector machines (SVMs), and they consider moderately large sequences around the functional site [73, 17, 4, 65].

Recent approaches [27] for human gene recognition also make use of the information available for other species to improve the recognition of the functional sites. However, the combination is carried out in a heuristic way. The species used for comparison are arbitrarily chosen, using the widely assumed hypothesis that we must consider moderately distant evolutionary relatives. Furthermore, the classifiers used for recognition of the sites in each species are also arbitrarily chosen. It is unlikely that such a process would produce the best possible result. Due to the large number of available species and the large number of different classifiers that can be applied to make use of such information, a systematic method for obtaining the best possible combination is highly desirable.

In this work, we propose a principled approach in which we can consider as many different sources of evidence as possible and use as many different classifiers as needed. A rapid validation process constructs a near-optimal combination that achieves a better performance than any of its members. To obtain a method that can be scaled up to as many sources of information as needed, we use a greedy stepwise approach. Two alternatives are designed, one based in a constructive approach beginning with an empty model and another based on a destructive approach beginning with a model considering all available sources of evidence. Then, a stepwise procedure is applied until no further improvement is observed in the obtained model.

5.2 Methods

Our aim is to develop a methodology for combining tens or even hundreds of classifiers for site recognition. From a machine learning perspective, such a problem is usually approached differently depending on the computational cost of the available solutions. The optimum approach is the exhaustive evaluation of all possible combinations of classifiers. However, if we have N trained classifiers, the number of possible combinations is $2^N - 1$, which is prohibitive even for moderate values of N. Thus, we must resort to optimization algorithms that will perform a guided search in the space of possible solutions. For the problem of finding the optimal solution, any of the many metaheuristics available in the machine learning literature, such as evolutionary computation [51], particle swarm optimization [35], ant colonies [14] or differential evolution [16], could be used. However, all of these methodologies require the repetitive evaluation of many solutions to achieve their optimization goal. In the problem of site recognition, the evaluation of a possible solution is a costly process due to the large datasets involved. Thus, these metaheuristics are not feasible.

To avoid the computational cost of these metaheuristics, we developed a different approach. We used a stepwise greedy approach in both a constructive and a destructive way, which requires evaluating significantly fewer solutions. The process for obtaining the best combination of classifiers for different species is composed of two main stages: training stage and validation stage. Before starting the learning process, we need the training datasets, the testing dataset and the validation dataset. Without loss of generality and to provide the necessary focus for our description, we will use here the same setup of the reported experiments below. We will address the problem of site recognition in the human genome. To solve this problem, we will use as a test set the sites of a certain chromosome, C. The training set will be all the remaining human chromosomes and the genomes of all the species we want. As validation, we will chose one of the human chromosomes in the training set, V, and remove it from the training set.

For the training stage, we select as many species as could be useful for our problem. We need not select the most appropriate ones because the stepwise validation stage will discard the useless classifiers. Once we have selected the set of species whose genomes we are going to use, we train as many classifiers as we want from those species. For every organism, we can train different classifiers, such as support vector machines (SVMs), neural networks (NNs), decision trees (DTs) of k-Nearest Neighbor (k-NN) rule, or the same classifier methods with different parameters. Because the validation stage can consider hundreds of classifiers, any method of potential interest can be used. Again, the validation stage will remove unneeded classifiers.

Once we have the trained classifiers, we will perform the validation stage, whose aim is to obtain the best possible combination of classifiers. For that purpose, we designed two different approaches. Both of these approaches are stepwise greedy approaches. We developed a constructive incremental approach and a destructive decremental approach. In the incremental approach, we begin by evaluating all the classifiers in the validation set V. The best one, c_1, is added to the set of selected classifiers, which was empty. Then, the evaluation is conducted again using c_1 together with all the remaining classifiers. The best combination is chosen, and a second classifier, c_2, is added. The process is repeated until the addition of a new classifier does not improve the validation accuracy.

For the destructive approach, we start with a model with all the available classifiers, n, $\{c_1, c_2, \ldots, c_n\}$. One by one, every classifier is removed from the set, and the set is reevaluated using the validation set. If all of the classifiers have a positive effect on the validation accuracy, the process is stopped. Otherwise, the worst performing classifier is removed and the process is repeated until the stop criterion is met.

Another issue must be considered for our approach. We must determine how the different classifiers are combined. In the machine learning literature, combining different sources of evidence for a classification problem is a common task [37]. Although various sophisticated methods have been developed for combining many classifiers [71, 48, 39, 56], in a practical sense, none of them are able to beat the simpler methods on a regular basis. Thus, we have considered three commonly used simple methods to combine the classifiers: sum of outputs, majority voting and maximum output. These methods are fairly straightforward. The combination using the sum of outputs simply adds together the outputs of all the models. The majority voting scheme counts the classification given by every model and outputs the most common case. The maximum approach uses only the classifier whose output has the highest absolute value.

For these three methods to be useful, we must consider the different ranges of their outputs and the different optimal decision thresholds of the five classification method we will use. To account for the different ranges, all the outputs of the methods were scaled to the interval $[-1, 1]$. To account for the different thresholds, we obtain the optimal threshold for each method by cross-validation, $th_{otpimal}$, and we obtain the effective output of every classifier, which is given by $y - th_{opimal}$, where y is the actual output of the classifier.

With the three combination methods and the two stepwise algorithms, we have for any performance measure selected six different combinations of models. For any recognition task and any performance measure, we will obtain these six models and return as a final result of our methodology the best combination in terms of cross-validation performance.

5.2.1 Experimental setup

To test our model, we chose the human genome together with other 20 species. Our aim was to test whether any species, regardless of its closeness with the human genome, could be useful. The following species were considered:[1] Anolis carolinensis (AC), Bos primigenius taurus (BT), Caenorhabditis elegans (CE), Callithrix jacchus (CJ), Canis lupus familiaris (CLF), Danio rerio (DR), Drosophila melanogaster (DM), Equus caballus (EC), Ficedula albicollis (FA), Gallus gallus (GG), Homo sapiens (HS), Macaca mulatta (MaM), Monodelphis domestica (MD), Mus musculus (MM), Ornithorhynchus anatinus (OA), Oryctolagus cuniculus (OC), Pan troglodytes (PT), Rattus norvegicus (RN), Schistosoma mansoni (SM), Sus scrofa (SS) and Takifugu rubripes (TR). These genomes were selected to have a wide variety of organisms whose genomes are fully annotated.

Five classifiers were trained from every dataset: the stop codon method [59], a decision tree, a k-nearest neighbor rule, a positional weight matrix and a support vector machine with a string kernel. The parameters for every classifier were obtained using 10-fold cross-validation.

To evaluate our approach, we used five different human chromosomes for testing purposes, chromosomes 1, 3, 13, 19 and 21, and we used chromosome 16 for validation purposes. For each chromosome, we trained the classifiers with all of the remaining chromosomes except 16, and we obtained the best combination method by using our approach and by using chromosome 16 for validation. We tested the chosen models with all of the true TIS and stop codons and all the negative samples of the given chromosome. That is,

[1]The acronyms in parentheses will be used across the chapter to refer to the corresponding species.

for chromosome 1, we trained the models with chromosomes 2 to 22 and X and Y, leaving out chromosome 16. Then, we chose the best combination method using chromosome 16, and we tested this combination of models using chromosome 1. A summary of these datasets is shown in Table 5.1. The chromosomes were selected with the aim of choosing chromosomes of different lengths and coding density. Chromosome 16 was chosen as a validation set because it is a chromosome of average length and coding density. We used all TISs and stop codons of the CCDS Update Released for Human of September 7, 2011. This update uses Human NCBI build 37.3 and includes a total of 26,473 CCDS IDs that correspond to 18,471 GeneIDs.

Table 5.1: Random undersampling was used for training; thus, the number of negative instances was equal to the number of positive instances.

Dataset		Training data	Testing data	
		Positives/Negatives	Positives	Negatives
Chr. 1	TIS	17,638	2,156	8,074,590
	STOP	17,404	2,154	23,573,031
Chr. 3	TIS	18,631	1,163	7,291,951
	STOP	18,444	1,114	21,522,500
Chr. 13	TIS	19,454	340	3,664,164
	STOP	19,225	333	10,878,302
Chr. 19	TIS	18,383	1,411	1,698,891
	STOP	18,136	1,422	4,665,804
Chr. 21	TIS	19,561	233	1,303,634
	STOP	19,558	237	3,726,959

One of the key aspects of the evaluation of any new proposal is the set of previous methods used in the comparison. Many different methods have been proposed for recognizing TISs and stop codons [73, 72, 69, 59]. However, these previous works and our own research [25] have shown that an SVM with a string kernel is the best state-of-the-art method not only for TISs and stop codons but also for splice sites [65]. To assure the general advantage of SVMs with string kernels, we performed a preliminary study of the different available methods that included position weight matrices, decision trees, k-nearest neighbors, stop codon method [59], Wang et al.'s method [69], Salzberg's method [60] and SVMs with linear and Gaussian kernels and three different string kernels: the locality improved (LI) kernel, the weighted degree kernel (WD) and the weighted degree kernel with shifts [58] (WDS). SVMs with WD kernels consistently provided the best results, so we chose them for the method to be compared with our proposal. WDS provided marginally better results than WD, but with a far higher computational complexity. To assure a fair comparison, we considered not only these methods but also all others used in classifiers. Then, for every experiment, we compared our approach to the best performing method in terms of validation performance. In fact, SVM with WD kernel was always the best classifier.

Another key parameter of the learning process is the window around the functional site that is used to train the classifiers. A further advantage of our approach is that it allows the use of a suitable window for each dataset. The value of the window for each classifier was obtained by cross-validation. We considered the site to be offset 0, and we did not count the TIS or the stop codon. We tested the performance of the following windows: $[-100, 0]$, $[-75, 25]$, $[-50, 0]$, $[-50, 50]$, $[-25, 0]$, $[-25, 25]$, $[-25, 75]$, $[-10, 15]$, $[-10, 40]$, $[-10, 90]$, $[0, 25]$, $[0, 50]$ and $[0, 100]$. For each trained classifier, the best window was chosen. For the stop codon method, we used the additional window values of $[0, 200]$, $[0, 300]$, $[0, 400]$ and $[0, 500]$ for TIS recognition and the window values of $[-200, 0]$, $[-300, 0]$, $[-400, 0]$ and $[-500, 0]$ for stop codon recognition.

Furthermore, SVMs are very sensitive to the learning parameters; thus, we also performed a cross-validation to obtain their values. The WD kernel has two parameters: the standard C parameter of any SVM and the window width of the string kernel. We tested values of $1, 10, 100$ and 1000 for C and 12 and 24 for the window width. All 8 combinations were evaluated using 10-fold cross-validation, and the best one was chosen. Although it may be argued that this method might result in suboptimal parameters, it represents a good compromise between the performance of the SVM and the high computational cost of evaluating each set of parameters. For PWM and C4.5, there are no parameters with a significant effect on their performance. For k-NN, the number of neighbors, k, was chosen by cross-validation in the interval $[1, 100]$.

To train the models, we used random undersampling [33] as stated in previous chapter, because it have been shown its usefulness for TIS recognition [25]. For random undersampling, we used a ratio of 1, which means that the majority class was randomly undersampled until both classes had the same number of instances. To avoid any contamination of the experiments, for every training set, regardless of the species, we removed the genes that were shared with the test chromosome.

To evaluate the obtained classifiers, we used the standard measures for imbalanced data , specificity and sensitivity, described on Chapter 2. The geometric mean of these two measures, $G - \text{mean} = \sqrt{Sp \cdot Sn}$, will be our first classification metric. As a second measure, we used the area under the receiver operating characteristic (ROC) curve (auROC). However, auROC is independent of class ratios, and it can be less meaningful when we have very unbalanced datasets [65]. In such cases, area under the precision recall curve (auPRC) can be used. This measure is especially relevant if we are mainly interested in the positive class. However, it can be very sensitive to subsampling. In our results, we use all the positive and negative instances for each of the five chromosomes tested, so no subsampling is used. This also yields small auPRC values.

We use these three metrics because they provide two different views of the performance of the classifiers. The auROC and auPRC values describe the general behavior of the classifier. However, when used in practice, we must establish a threshold for the classification of a query pattern. G-mean provides the required snapshot of the performance of the classifier when we set the required threshold.

5.3 Results and discussion

As stated, we performed experiments for the recognition of TISs and stop codons to provide the necessary focus. However, our approach is applicable to any recognition task. The experiments had two different objectives. We wanted to know which species were more useful for the recognition of the two functional sites. We challenged the general heuristic method of selecting a species based on biological considerations alone. We also wanted to compare the results using our method with the standard procedure of selecting the best performing model, which is the common approach in the literature. In the following two sections, we discuss the results for TIS and stop codon recognition.

5.3.1 Results for TIS recognition

One of the advantages of our approach is that we can optimize for the performance measure that we are interested in, which can be the G-mean, the auROC, the auPRC or any other measure useful for our application. Thus, we conducted our experiments using three performance measures: G-mean, auROC and auPRC. The first relevant result is that the combination of best models obtained for each measure was different. This means that, depending on the aim of the work, different combinations of classifiers are needed.

For each of the five studied chromosomes, we obtained three different combinations of models, each optimized for one of the three measures mentioned above. As a general rule, the constructive method always outperformed the destructive method. The latter always obtained combinations of many more models that exhibited over-fitting and worse performance. It is also interesting to note the homogeneous behavior across the different chromosomes. For all of the five chromosomes, the combination that achieved the best results was the sum for auROC and auPRC and majority for G-mean. The combination based on the maximum output was never the best-performing one. In this latter combination method, the effect of a bad classifier was too harmful to obtain good performance. In this work, for brevity's sake, only the best models are reported. However, all the results can be found in the supplementary material.

Once we established the best stepwise method and the best combination, we examined the results in terms of the species involved in the best combinations. Table 5.2 shows the models selected for the best combination for each measure and each chromosome. Regardless of the optimized measure, there was only one species that never appeared in the best combination: CE. This result indicates that, although the contribution of certain species is more relevant than others, the information of many genomes was useful for the prediction of human TISs, even those species that are very distant relatives of humans. Another interesting result is the fact that, for the three different measures, auROC, auPRC and G-mean, the obtained combinations of models were quite different. This result indicates that we must consider our aims before designing our classifier. In most previous works, that is not taken into account.

Regarding the classification models, PWM was never chosen. The stop codon method was chosen for EC and SM. The decision tree trained with the C4.5 algorithm was selected several times, but the k-NN rule and the SVM with a string kernel were the most frequently selected methods. The case of k-NN is remarkable in that it is not usually used for this task [73, 72, 69, 59]. It appears that the diversity that k-NN introduced in the models was useful for the overall performance of the combinations, despite of the fact that k-NN

			CE					AC					DM					OA					HS					OC					BT				
Chr	Obj.	#	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W
1	auROC	6																																			
	auPRC	28								X					X					X							X			X					X		
	G	2																																			
3	auROC	7																																			
	auPRC	18								X					X												X			X							X
	G	2																																			
13	auROC	6																																			
	auPRC	16																		X										X					X		
	G	2																																			
19	auROC	7																																			X
	auPRC	20													X					X							X			X					X		
	G	2																																			
21	auROC	6																																			X
	auPRC	29								X					X															X		X		X	X		X
	G	2																																			

			CJ					FA					EC					CLF					GG					MD					SM				
Chr	Obj.	#	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W
1	auROC	6					X						X		X					X																	
	auPRC	28			X		X			X			X		X		X			X					X					X			X				
	G	2																																			
3	auROC	7					X						X				X			X					X												
	auPRC	18			X					X					X					X					X								X				
	G	2																																			
13	auROC	6											X				X			X																	
	auPRC	16								X			X		X					X					X												
	G	2																																			
19	auROC	7					X													X																	
	auPRC	20								X		X	X		X					X					X					X							
	G	2																																			
21	auROC	6											X		X										X												
	auPRC	29		X	X		X			X			X		X		X		X	X					X					X			X				
	G	2																																			

			MM					DR					MaM					TR					PT					SS					RN				
Chr	Obj.	#	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W
1	auROC	6															X										X										
	auPRC	28		X	X		X			X				X	X					X		X		X	X					X							X
	G	2															X										X										
3	auROC	7															X										X										
	auPRC	18			X									X	X					X										X					X		X
	G	2															X										X										
13	auROC	6															X										X			X							
	auPRC	16					X			X				X	X		X			X		X			X												
	G	2															X										X										
19	auROC	7													X		X										X	X									
	auPRC	20			X									X						X		X							X	X					X		X
	G	2															X										X										
21	auROC	6															X										X										
	auPRC	29			X					X				X	X					X		X			X					X					X		X
	G	2															X										X										

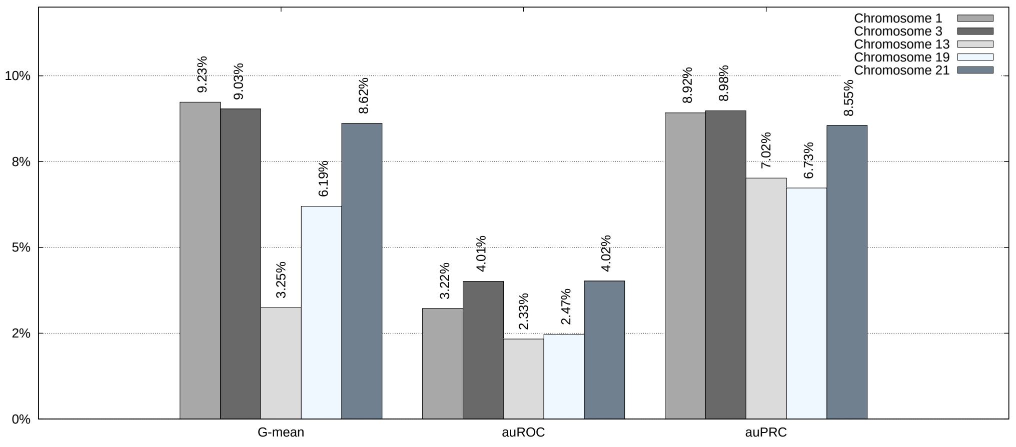
Table 5.2: The table shows the models selected for all methods and the five studied chromosomes for TIS recognition. S stands for stop codon method, C for C4.5, K for k-NN, P for PWM and W for a SVM with a WD string kernel.

alone showed worse performance than an SVM alone. The explanation for this may be found in the behavior of the ensembles of classifiers. It is well known [38] that a diverse ensemble of classifiers improves the performance of the set of classifiers.

With respect to the three different objectives, optimizing the G-mean showed the most stable results. For the five chromosomes, the selected models were always the SVM method for MaM and PT. For auROC, six or seven models were selected. The SVM method was always chosen for MaM and PT, but the remaining methods depended on the chromosome. This is another interesting result because most TIS recognition programs mainly rely on common models for any task. Finally, for auPRC, significantly more models were selected, from 16 to 29, with a significant variation between the chromosomes. Here, the large number of negative samples made this task harder than optimizing the other two criteria.

The next step was to compare the performances of our approach and the standard method of choosing the best performing classifier. A summary of the results for TIS recognition of the five studies chromosomes is shown in Table 5.3. The first interesting result is that the proposed approach beat the standard approach for all measures and all chromosomes. The improvements in auROC, auPRC and G-mean are shown in Figure 5.1.

Figure 5.1: Improvement for TIS recognition.
The figure shows the improvement of our approach with respect to the standard approach of using the best performing classifier.



The differences were significant. For G-mean, in the worst case, the

Chrom	Obj.	Method	Combin.	auROC	auPRC	G	Sp	Sn	TP	FN	TN	FP	#models
Chr. 1	-	Std	-	0.9473	0.0929	0.8430	0.9528	0.7458	1,608	548	7,693,177	381,413	-
	auROC	Cons	Sum	**0.9795**	0.1478	0.8732	0.9891	0.7709	1,662	494	7,986,476	88,114	6
	auPRC	Cons	Sum	0.9698	**0.1821**	0.8449	0.9860	0.7240	1,561	595	7,961,915	112,675	28
	G	Cons	Majority	0.9445	0.0074	**0.9353**	0.9599	0.9114	1,965	191	7,750,802	323,788	2
Chr. 3	-	Std	-	0.9334	0.0831	0.8335	0.9584	0.7248	843	320	6,988,324	303,627	-
	auROC	Cons	Sum	**0.9735**	0.1371	0.8418	0.9929	0.7137	830	333	7,240,279	51,672	7
	auPRC	Cons	Sum	0.9552	**0.1729**	0.7925	0.9897	0.6346	738	425	7,216,813	75,138	18
	G	Cons	Majority	0.9315	0.0049	**0.9238**	0.9664	0.8831	1,027	136	7,046,822	245,129	2
Chr. 13	-	Std	-	0.9506	0.0844	0.8692	0.9479	0.7971	271	69	3,473,147	191,017	-
	auROC	Cons	Sum	**0.9739**	0.1281	0.8256	0.9946	0.6853	233	107	3,644,322	19,842	6
	auPRC	Cons	Sum	0.9679	**0.1546**	0.8067	0.9921	0.6559	223	117	3,635,161	29,003	16
	G	Cons	Majority	0.9099	0.0029	**0.9017**	0.9699	0.8382	285	55	3,553,896	110,268	2
Chr. 19	-	Std	-	0.9456	0.1277	0.8727	0.9068	0.8398	1,185	226	1,540,551	158,340	-
	auROC	Cons	Sum	**0.9703**	0.1741	0.8752	0.9737	0.7867	1,110	301	1,654,211	44,680	7
	auPRC	Cons	Sum	0.9542	**0.1950**	0.8520	0.9646	0.7527	1,062	349	1,638,692	60,199	20
	G	Cons	Majority	0.9436	0.0181	**0.9346**	0.9555	0.9142	1,290	121	1,623,271	75,620	2
Chr. 21	-	Std	-	0.9353	0.0726	0.8409	0.9415	0.7511	175	58	1,227,429	76,205	-
	auROC	Cons	Sum	**0.9755**	0.1137	0.8512	0.9873	0.7339	171	62	1,287,068	16,566	6
	auPRC	Cons	Sum	0.9694	**0.1581**	0.8288	0.9879	0.6953	162	71	1,287,849	15,785	29
	G	Cons	Majority	0.9362	0.0043	**0.9271**	0.9582	0.8970	209	24	1,249,122	54,512	2

Table 5.3: The table shows the specificity (Sp), sensitivity (Sn), true positives (TP), true negatives (TN), false negatives (FN), false positives (FP) and area under the ROC and PRC curves (auROC/PRC) for all methods and the five studied chromosomes for TIS recognition.

improvement was 3.25%, and in the best case, it was 9.23%. For auPPRC, the results were even better, from 6.73% to 8.92%. For auROC, the improvement was less significant, but it still ranged from 2.33% to 4.02%.

Table 5.4 shows the relative improvement of our approach in terms of the numbers of true positive, false negatives, true negatives and false positives. In the table, we can see how our approach was able to improve the false negative results in the worst case by 20% and in the best case by 65%. This reduction is relevant because most current gene recognizers rely heavily on the classification of TISs; therefore, it is very likely that those genes would be completely missed by any gene recognizer. Our approach has the potential to significantly improve the accuracy of any annotation system.

Chromosome	True positive	False negative	True negative	False positive
Chr.1	22.20%	65.15%	0.75%	15.11%
Chr. 3	21.83%	57.50%	0.84%	19.27%
Chr. 13	5.17%	20.29%	2.32%	42.27%
Chr. 19	8.86%	46.46%	5.37%	52.24%
Chr. 21	19.43%	58.62%	1.77%	28.47%

Table 5.4: Relative improvement for true positives, false negatives, true negatives and false positive of our approach over the best method for TIS recognition.

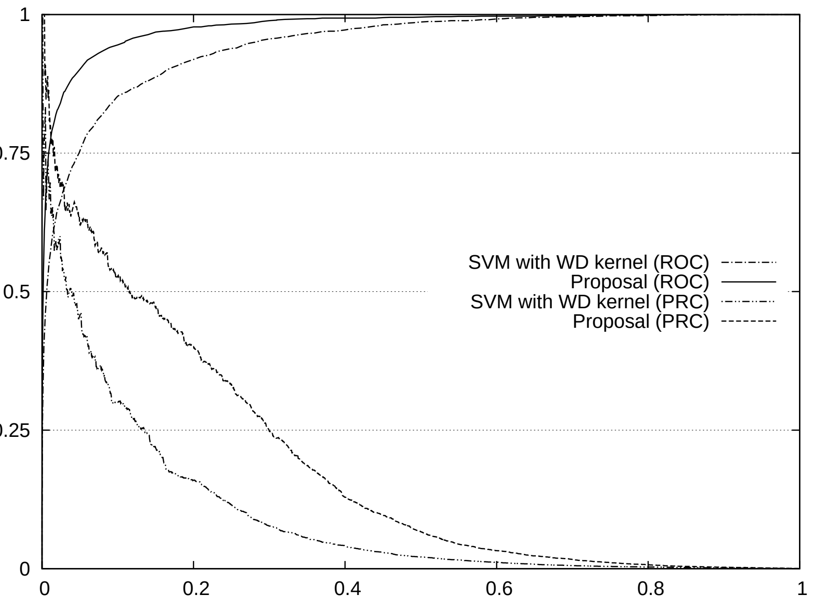
Furthermore, our method was also able to improve the true negative rate, from 15% to 52% depending on the chromosome. Therefore, any annotation system that uses our approach would have a significantly reduced set of putative TISs and better expected performance. This would be especially true when the large amount of false positives found by the standard approach is an actual problem for any automatic annotation system.

The improvements in the auROC and auPRC[2] values are also shown in Figure 5.1. The actual ROC and PRC curves are shown in Figures 5.2–5.6. These figures show that our approach improved the auROC and auPRC for

[2]We always performed the testing of all the methods with all the negative samples. That means that the ratio minority/majority class is almost 1:11000 for the worst case yielding to low auPRC values. We must take into account that with only a few thousands FPs among several millions of TNs we would obtain a very low precision value. The situation for stop codon recognition is even worse as the number of TNs is multiplied by three.

all five studied chromosomes. These results demonstrate that the overall performance of the proposed method was better than the performance of the best model. The actual ROC and PRC curves shown in Figures 5.2–5.6 show that the curves corresponding to our proposal are always above the curves of the best model. This result indicates better performance for all the possible thresholds of classification.

Figure 5.2: ROC/PRC curves for TIS prediction for chromosome 1.



5.3.2 Results for stop codon recognition

The second part of our experiments was devoted to stop codon recognition. Stop codon recognition is a more difficult task because the achieved accuracy is less than that of TIS recognition. One of the major sources of this increased complexity is the number of negative instances. There are three different stop codons rather than just one, as in the case of TIS recognition. Therefore, the number of negative instances is approximately three times the number of negative instances for TIS prediction. For instance, using the same five chromosomes from the previous experiments, the best current method found more than six million false positive stop codons across all the chromosomes.

Figure 5.3: ROC/PRC curves for TIS prediction for chromosome 3.

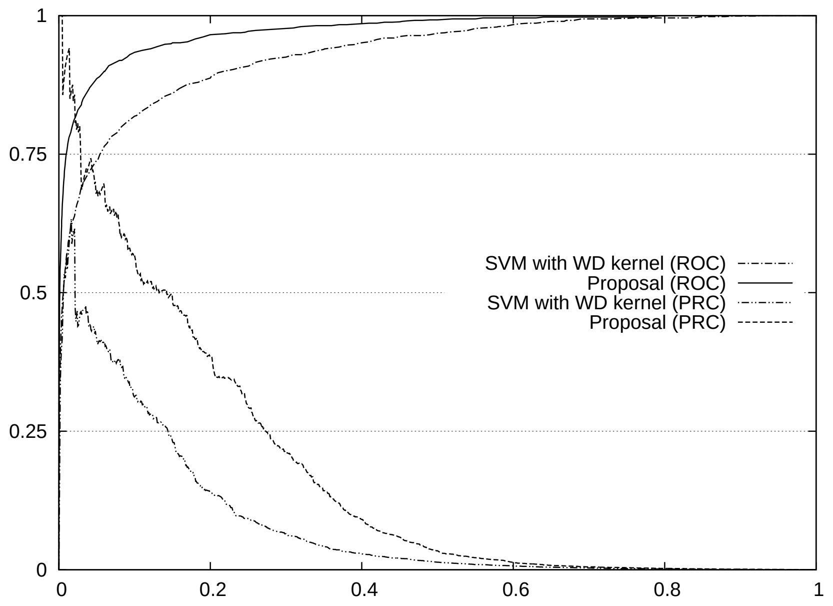


Figure 5.4: ROC/PRC curves for TIS prediction for chromosome 13.

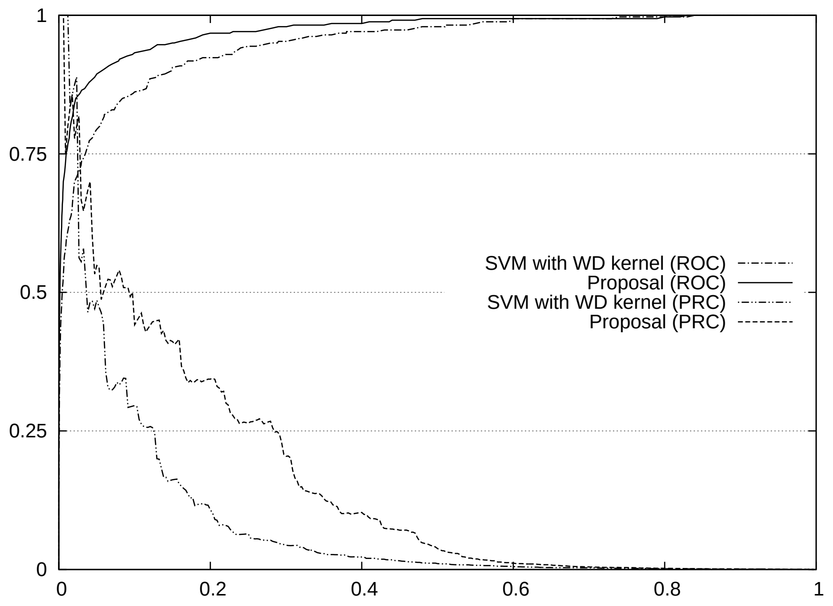


Figure 5.5: ROC/PRC curves for TIS prediction for chromosome 19.

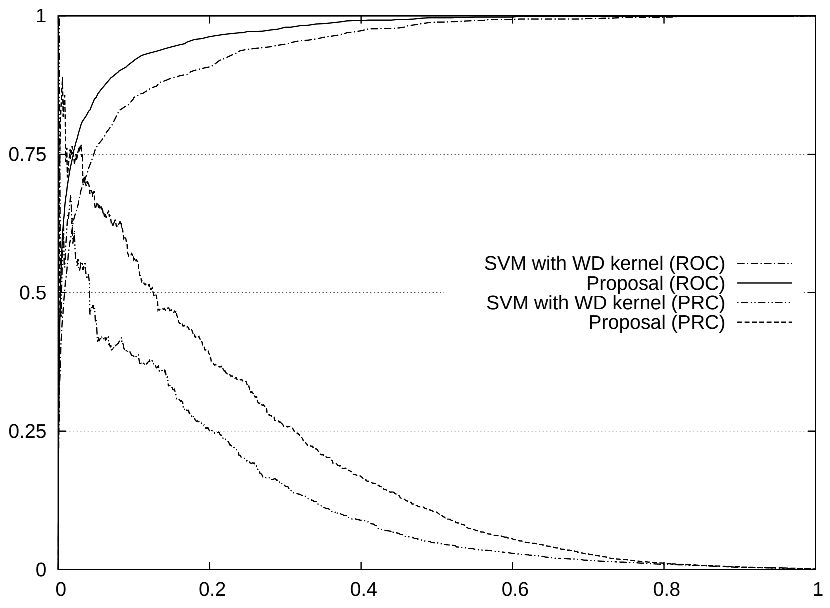
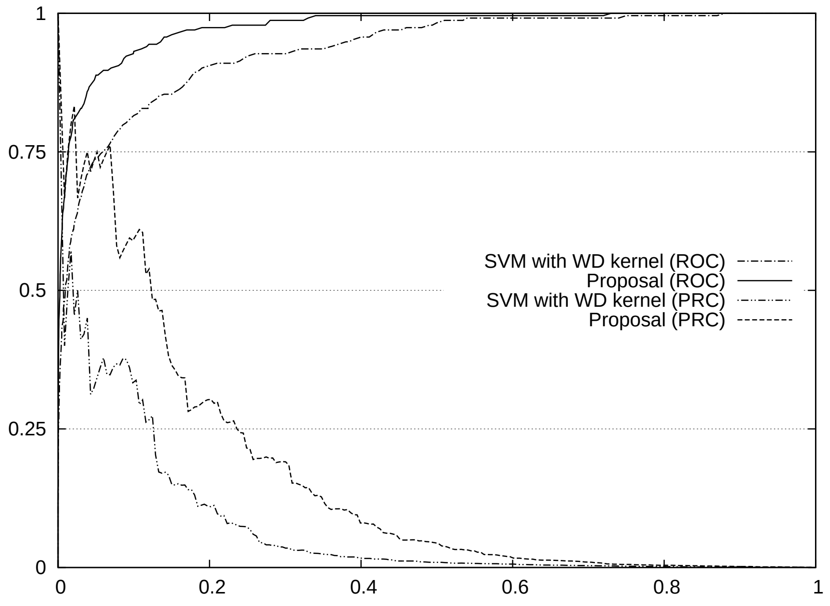


Figure 5.6: ROC/PRC curves for TIS prediction for chromosome 21.



This number of incorrectly predicted stop codons might be enough to mar any annotation system, indicating that there is ample room for improvement.

As stated, one of the advantages of our approach is that we can optimize for the performance measure we are interested in, whether it is G-mean, auROC, auPRC or any other useful metric. Thus, as for TIS recognition, we carried out experiments using three performance measures: G-mean, auROC and auPRC. Again we found that the combination of best models obtained for each measure was different. In fact, more variation was found for stop codons than for TIS recognition.

For each of the five studied chromosomes, we obtained three different combinations of models, each one aiming at optimization of one of the three measures mentioned above. Table 5.5 shows the models selected for the best combination for each measure and each chromosome. As it did for TIS recognition, the constructive method always outperformed the destructive method. The latter always obtained combinations of more models that yielded to over-fitting and worse performance. It is also interesting to note the homogeneous behavior across the different chromosomes. For all five chromosomes, the combination that achieved the best results was the sum for auROC and auPRC and the majority for G-mean. The combination based on the maximum output was never the best performing combination.

Regardless of the optimized measure, there were only three species that never appeared in the best combination: AC, HS and FA. As was the case for TIS recognition, although the contributions of certain species were more relevant than others, the information from many genomes was useful for the prediction of human stop codons, even those species with a large distance from the human genome. It is interesting to note that classifiers trained on the human genome were never used. The analysis of the behavior showed that the information found in the human genome was redundant after a few other species were added and then its inclusion did not improve the overall performance.

For the three different measures, auROC, auPRC and G-mean, the obtained combinations of models are quite different. That means that we must consider which our aim before designing our classifier. This same behavior was observed for TIS recognition. However, here the situation is less stable, with more variations among chromosomes.

Regarding the classification models, PWM was never chosen. The stop codon method was chosen for EC and SM. The decision tree trained with the C4.5 algorithm was selected several times, but the k-NN rule and the

			CE					AC					DM					OA					HS					OC					BT				
Chr	Obj.	#	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W
1	auROC	5																																			
	auPRC	16																													X					X		
	G	2																																			
3	auROC	5																																			
	auPRC	10																		X										X					X		
	G	4	X																																		
13	auROC	5																																			
	auPRC	22																		X															X		
	G	4	X																																		
19	auROC	5																																			
	auPRC	27	X										X							X									X	X					X		
	G	2																																			
21	auROC	5																																			
	auPRC	21											X							X										X			X		X		
	G	2																																			

			CJ					FA					EC					CLF					GG					MD					SM				
Chr	Obj.	#	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W
1	auROC	5					X													X																	
	auPRC	16		X											X			X												X			X				
	G	2																																			
3	auROC	5					X													X																	
	auPRC	10																															X				
	G	4					X										X																				
13	auROC	5					X								X																						
	auPRC	22		X	X								X	X	X					X										X			X				
	G	4													X																						
19	auROC	5					X								X																						
	auPRC	27		X	X									X	X			X		X		X											X				
	G	2																																			
21	auROC	5					X								X																						
	auPRC	21		X													X		X						X								X				
	G	2																																			

			MM					DR					MaM					TR					PT					SS					RN				
Chr	Obj.	#	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W	S	C	K	P	W
1	auROC	5															X										X						X				
	auPRC	16					X						X	X	X					X		X		X						X					X		
	G	2															X										X										
3	auROC	5															X										X						X				
	auPRC	10					X								X					X					X					X			X				
	G	4															X																				
13	auROC	5															X										X						X				
	auPRC	22			X		X	X		X				X	X							X		X			X	X		X			X				
	G	4															X										X										
19	auROC	5															X										X						X				
	auPRC	27		X	X		X			X				X						X		X		X	X				X	X			X		X		X
	G	2															X										X										
21	auROC	5															X										X						X				
	auPRC	21			X		X							X	X					X		X		X	X		X			X			X		X		
	G	2															X										X										

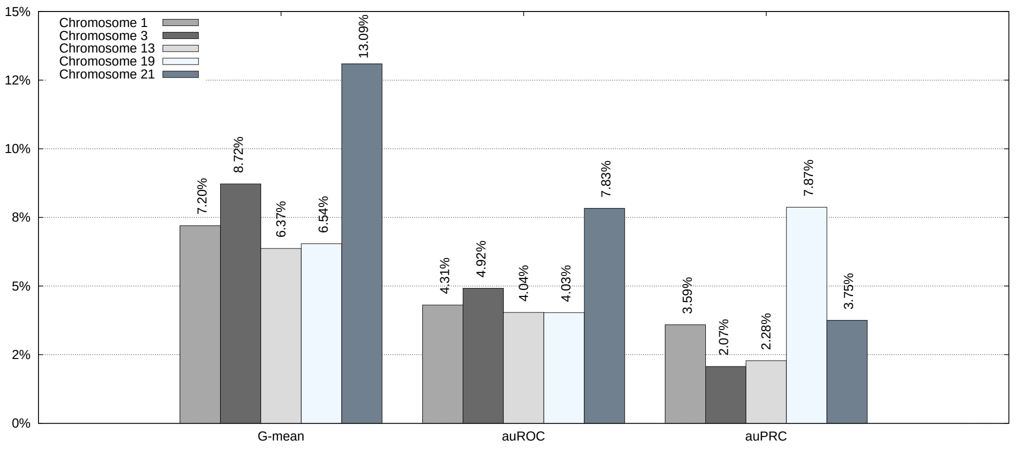
Table 5.5: The table shows the models selected for all methods and the five studied chromosomes for stop codon recognition. S stands for stop codon method, C for C4.5, K for k-NN, P for PWM and W for a SVM with a WD string kernel.

SVM method with a string kernel were the most frequently selected methods. These results are similar to the ones obtained for TIS recognition.

With respect to the three different objectives, optimizing the G-mean showed the most stable results. For the five chromosomes, the SVM method for MaM and PT was always selected, with the exception of chromosome 3. However, for chromosomes 13 and 19, two additional models were selected. Surprisingly, CE was selected in both cases, despite its large evolutionary distance to human. This result supports the idea that selecting the genomes in an intuitive way is not optimal. For auROC, five models were always selected, although not the same models for every chromosome. The SVM method for MaM and PT was always chosen, but the remaining methods depended on the chromosome. This is another interesting result because most stop codon recognition programs rely on common models for any task. Finally, for auPRC, significantly more models were selected, from 10 to 27, with a significant variation between the chromosomes.

The next step was to compare the performances of our approach and the standard method of choosing the best performing classifier. A summary of the results for stop codon recognition of the five studies chromosomes is shown in Table 4.5. The first interesting result is that the proposed approach beat the standard approach for all measures and all chromosomes. The improvements in auROC, auPRC and G-mean are shown in Figure 5.7.

Figure 5.7: Improvement for stop codon recognition.
The figure shows the improvement of our approach with respect to the standard approach of using the best performing classifier.



Chrom.	Objective	Method	Combination	auROC	auPRC	G	Sp	Sn	TP	FN	TN	FP	#models
Chr. 1	-	Std	-	0.9280	0.0142	0.8487	0.8942	0.8055	1,735	419	21,077,932	2,495,099	-
	auROC	Cons	Sum	**0.9711**	0.0376	0.8961	0.9695	0.8282	1,784	370	22,853,736	719,295	5
	auPRC	Cons	Sum	0.9553	**0.0501**	0.8259	0.9718	0.7019	1,512	642	22,909,075	663,956	16
	G	Cons	Majority	0.9307	0.0013	**0.9207**	0.9426	0.8993	1,937	217	22,218,892	1,354,139	2
Chr. 3	-	Std	-	0.9233	0.0083	0.8256	0.9159	0.7442	829	285	19,711,889	1,810,611	-
	auROC	Cons	Sum	**0.9725**	0.0276	0.9086	0.9661	0.8546	952	162	20,793,633	728,867	5
	auPRC	Cons	Sum	0.9478	**0.0290**	0.7887	0.9747	0.6382	711	403	20,979,038	543,462	10
	G	Cons	Majority	0.9584	0.0015	**0.9128**	0.9462	0.8806	981	133	20,364,397	1,158,103	4
Chr. 13	-	Std	-	0.9185	0.0071	0.8150	0.9103	0.7297	243	90	9,902,079	976,223	-
	auROC	Cons	Sum	**0.9589**	0.0165	0.8780	0.9723	0.7928	264	69	10,577,345	300,957	5
	auPRC	Cons	Sum	0.9458	**0.0299**	0.7978	0.9812	0.6486	216	117	10,673,614	204,688	22
	G	Cons	Majority	0.9311	0.0010	**0.8787**	0.9489	0.8138	271	62	10,321,997	556,305	4
Chr. 19	-	Std	-	0.9328	0.0379	0.8515	0.8664	0.8368	1,190	232	4,042,574	623,230	-
	auROC	Cons	Sum	**0.9731**	0.0847	0.9153	0.9336	0.8973	1,276	146	4,356,213	309,591	5
	auPRC	Cons	Sum	0.9615	**0.1166**	0.8750	0.9435	0.8115	1,154	268	4,402,031	263,773	27
	G	Cons	Majority	0.9346	0.0026	**0.9169**	0.9002	0.9339	1,328	94	4,199,984	465,820	2
Chr. 21	-	Std	-	0.8890	0.0083	0.7778	0.9191	0.6582	156	81	3,425,375	301,584	-
	auROC	Cons	Sum	**0.9673**	0.0319	0.8853	0.9625	0.8143	193	44	3,587,201	139,758	5
	auPRC	Cons	Sum	0.9571	**0.0458**	0.8123	0.9712	0.6793	161	76	3,619,627	107,332	21
	G	Cons	Majority	0.9199	0.0007	**0.9087**	0.9320	0.8861	210	27	3,473,463	253,496	2

Table 5.6: The table shows the specificity (Sp), sensitivity (Sn), true positives (TP), true negatives (TN), false negatives (FN), false positives (FP) and area under the ROC and PRC curves (auROC/PRC) for all methods and the five studied chromosomes for stop codon recognition.

Chromosome	True positive	False negative	True negative	False positive
Chr.1	11.64%	48.21%	5.41%	45.73%
Chr. 3	18.34%	53.33%	3.31%	36.04%
Chr. 13	11.52%	31.11%	4.24%	43.01%
Chr. 19	11.60%	59.48%	3.89%	25.26%
Chr. 21	34.62%	66.67%	1.40%	15.95%

Table 5.7: Relative improvement for true positives, false negatives, true negatives and false positive of our approach over the best method for stop codon recognition.

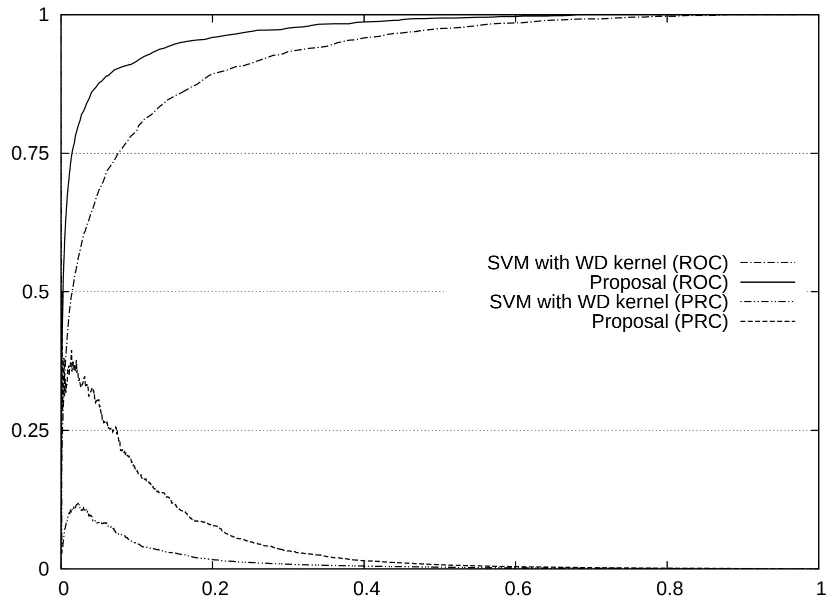
The differences were significant. For G-mean, in the worst case, the improvement was 6.37%, and in the best case, it was 13.09%. For auPPRC, the results showed an improvement from 2.07% to 7.87%. For auROC, the improvement was also significant, ranging from 4.03% to 7.83%.

Table 5.7 shows the relative improvement of our approach in terms of true positives, false negatives, true negatives and false positives. From the table, we can see how our approach was able to improve the false negative results in the worst case by 31% and in the best case by 66%. This reduction is relevant, as most current gene recognizers rely heavily on the classification of stop codons; therefore, it is very likely that those genes would be completely missed by any gene recognizer. Our approach has the potential to significantly improve the accuracy of any annotation system.

Furthermore, our method was also able to improve the true negative rate, from 1% to 5% depending on the chromosome. Therefore, any annotation system that uses our approach would have a significantly reduced set of putative TISs and better expected performance. This is especially true when a large amount of false positives is found the by the standard approach, which is an actual problem for any automatic annotation system.

The improvements for auROC and auPRC values are also shown in Figure 5.7. The actual ROC and PRC curves are shown in Figures 5.8–5.12. These figures show that our approach improved the auROC and auPRC for all five studied chromosomes. These results demonstrate that the overall performance of the proposed method was better than the performance of best model. The actual ROC and PRC curves shown in Figures 5.8–5.12 show that the curves corresponding to our proposal are always above the curves of the best model. This indicates better performance for all the possible thresholds of classification.

Figure 5.8: ROC/PRC curves for stop codon prediction for chromosome 1.



5.4 Summary

In this chapter, we presented a new approach for functional site recognition in genomic sequences. The approach consists of a stepwise procedure that can combine tens or hundreds of classifiers trained on different sequences and using genomic information from different species. The approach is rapid and can be used for the recognition of any type of functional site. Our method substitutes the current approach of selecting the species to be used heuristically based on biological considerations. Our results have proven that that methodology is suboptimal because species that are not considered in previous works have been shown useful in our experiments.

Although we have focused our experiments on the case of the combination of multiple species, we can also use the proposed approach for combining classifiers trained on different sequences of the same species, or classifiers trained using different parameters or learning procedures.

Figure 5.9: ROC/PRC curves for stop codon prediction for chromosome 3.

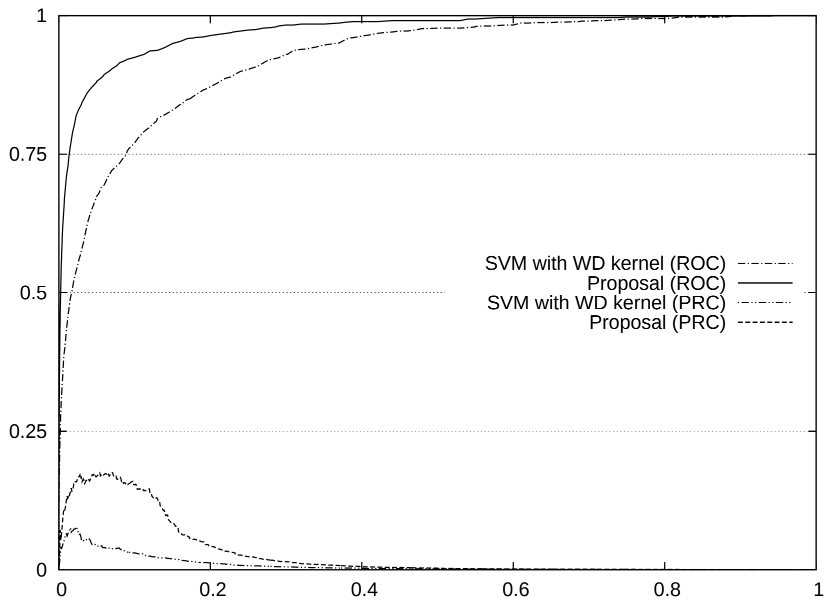


Figure 5.10: ROC/PRC curves for stop codon prediction for chromosome 13.

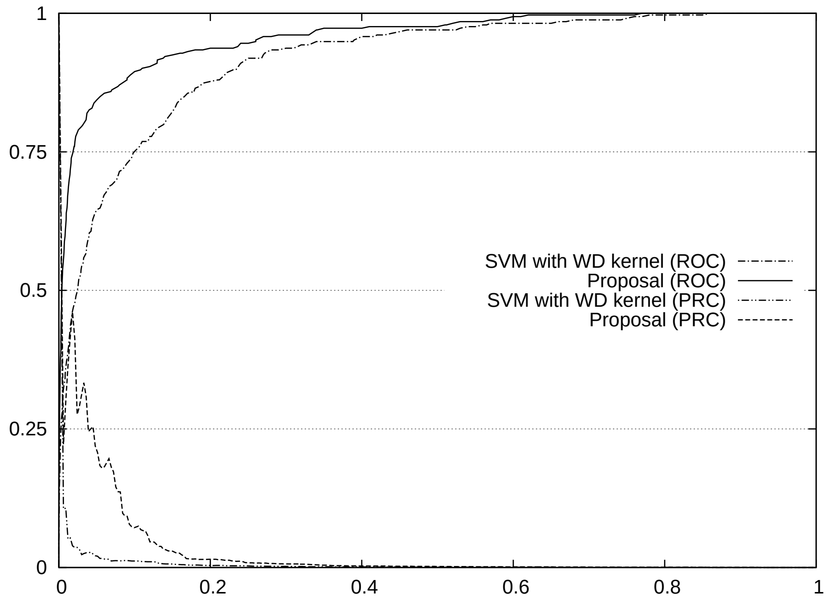


Figure 5.11: ROC/PRC curves for stop codon prediction for chromosome 19.

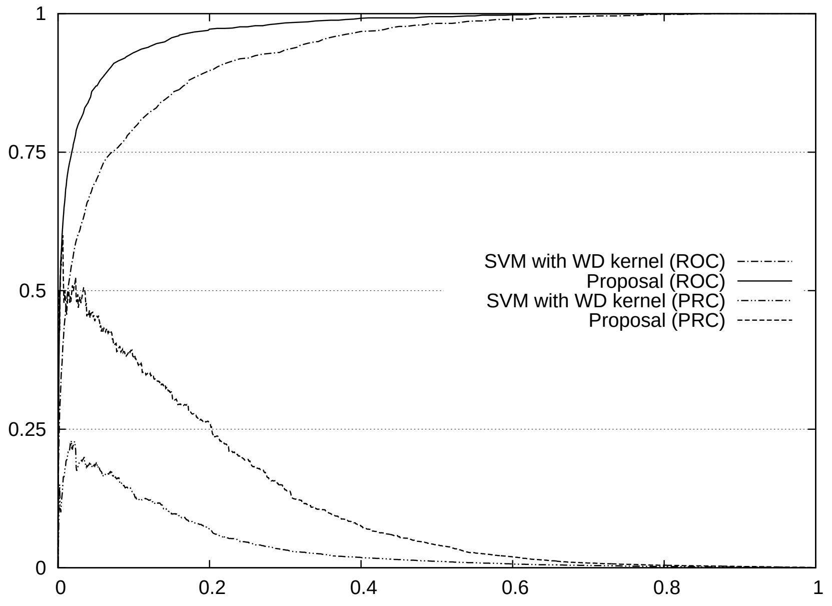
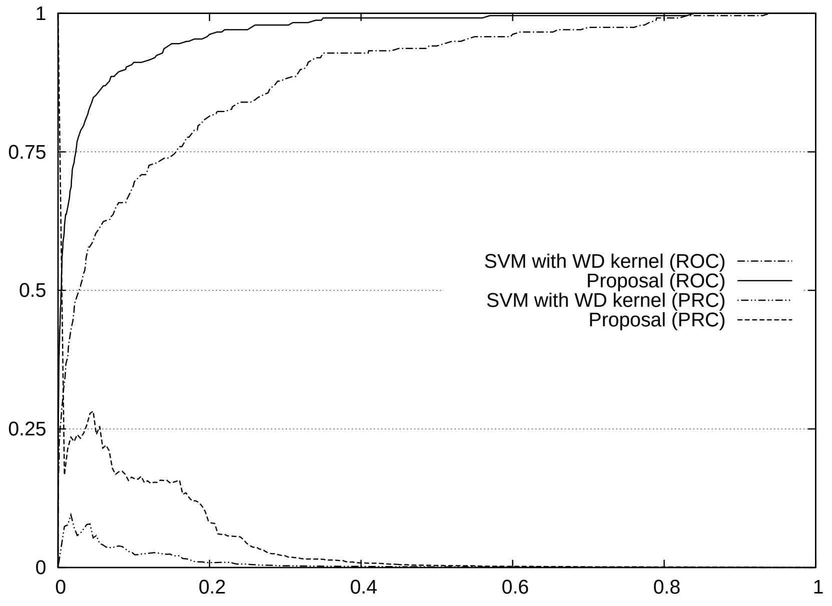


Figure 5.12: ROC/PRC curves for stop codon prediction for chromosome 21.



Furthermore, with our method, we can optimize any measure we are interested in. For instance, in the reported experiments, we have shown how we can focus on the optimization of G-mean, auROC or auPRC measures. The results have shown that the combination of classifiers that optimizes each one of these measures can be very different, supporting our separate approach.

To provide the necessary focus, we restrict the experimental study of our method to TIS and stop codon recognition. The reported results show that the proposed method exhibits improved sensitivity, specificity, auROC and auPRC compared with the standard approach of using the best available classifier. The results show a remarkable improvement in the G-mean, auROC and auPRC measures. Because state-of-the-art annotation systems rely heavily on the accurate prediction of functional sites of genes, the proposed method is an effective way of improving current gene recognizers.

Chapter 6

Evolutionary computation as a gene structure prediction framework

A complete and accurate gene set for each sequenced genome is still perhaps the single most important resource for genomic research after the genome sequence itself. Accurate annotation of the human genome and other species is an essential element in support of current drug discovery efforts, and validating potential drug targets against genomic sequence requires accurate annotation from the start to make this procedure worthwhile.

Automatic gene recognition continues being a difficult task, mainly due to the absence of clear signals that can be used to recognize and the growing complexity of the gene structure that its being revealed with the increased knowledge we are gaining. Many flimsy pieces of evidence must be combined to determine the correct structure of the gene. Regarding it as a search problem, where many evidence sources can be combined, this paper proposes an evolutionary computation framework for gene structure prediction. Additionally, we use functional site prediction techniques presented in previous chapter to localize the functional sites along the genomic sequence to reduce the search space. Evolutionary computation is used to evolve a population where the individuals are correct genetic structures. A fitness function based on content sensors determines which individuals hold outstanding statistical features. Evolutionary computation is a powerful paradigm that will allow capturing all the complexities of the structure of genes when other optimization measures fail.

The application of evolutionary algorithms to gene recognition will open a new field of research where its flexibility can be used to address the complexities of the problem, which are growing as our knowledge of the molecular processes of transcription and translation deepens.

6.1 Background

Bioinformatics is among the areas of major development in the field of machine learning. The vast amount of data that are constantly being produced, as well as the high difficulty inherent to the problems in this field, poses a challenge to the machine learning community. A notably relevant and attractive topic in bioinformatics is gene structure prediction[9], due to the enormous potential information hidden in the genomes of the different species.

The genome of a living organism is the set of instructions that determine its biological processes and features. These instructions, collected in DNA in nucleus of the cells as sequences of nucleotides, are processed to be transcripted in RNA segments, which in great part, are used to synthesize the proteins. Even though the importance of the played role by non-coding RNA has been demonstrated recently, the proteins continue being the main elements for the cell from a functional and structural point of view. Therefore, to determine the complete set of genes composing the whole genome of an organism is a crucial task to understand the biological information held in such sequence. However, it remains many aspects which generate controversy and convert this task in a not trivial one. Although many Genome Projects corresponding to several species were completed years ago, the catalogs of genes have not set a clearly exact number that compose them, quantity that can be variable in individuals of the same specie, even between DNAs of the same organism.

At this chapter, genes will be consider as discrete entities linearly located within chromosome sequences that they have the potential information required to transcript in RNA. In such sense, gene prediction problem can be stated as the try of determining of the boundaries of structures of those parts that are transcripted in RNA in a given genomic DNA target sequence and it can make reference to the search both protein coding genes as non-coding RNA genes.

The current approaches (see Section 1.4) have been able to achieve a peak performance of about a 50% of all the human genes. This is an impressive results if we considered the state of the problems just a decade back. However, the current methods seems to have reached a standstill situation, where no significant advances have been reported in the last few years. In this context, we are persuaded that evolutionary computation offers the perfect tool to the necessary significant improvement of gene recognizers. Evolutionary computation is not only a very powerful heuristic to address very difficult optimization problems, but also it has the flexibility we need to deal with all the non-canonical structures of the genes and the new situation that arise with our increasing knowledge of the biological processes.

However, although evolutionary computation has been used to solve some aspects of gene recognition, no general approach mainly based on evolutionary computation has been developed. In this chapter we show how we can use evolutionary computation for gene recognition with a performance comparable with gene predictors of much more complexity. Specifically, we present a novel approach to implement a framework where different sources of evidences for gene recognition from diverse nature are combined. This approach takes the advantages that evolutionary computation characteristics hold to develop a robust and open framework to face the problem from a heuristic point of view. The most likely exonic structure is searched by using the available information and while maintaining the inherent biological restrictions.

6.1.1 Gene prediction problem

As stated on Section 1.3, gene prediction is the problem of identifying the biologically functional portions in DNA sequences and assembling these parts to obtain the complete structure of the gene. Despite this concept mostly refers to protein-coding regions, it can also be used to make mention to other functional elements, e.g. ncRNA genes. The approach presented on this work will address the protein coding gene prediction problem in eukariotic cells, although the methodology is general enough to be useful in the problem of non coding gene prediction.

Then, the gene prediction problem can be stated as:

Let be X a DNA sequence, a string over the alphabet A, T, G, C,

$$X = (x_1, x_2, ..., x_n) \in \Sigma, \text{ where } \Sigma = \{A, C, G, T\} \tag{6.1}$$

the goal is to correctly label each element of X as belonging to a specific region. All coding genes begin at the TIS and end at the stop codon. The coding segments in certain cases, can be interrupted by introns. The boundaries Exon–intron intron–exon boundaries are called the splice sites, donor and acceptor respectively. Thus, in a correct gene structure:

- The exons do not overlap.
- The gene starts and finishes within an exon.
- An intron must be flanked by two exons.
- A gene can consist of only one exon.
- The complete set of coding exons must be frame compatible.
- Merging coding exons will not generate an in-frame stop codon.

Therefore, it consists of determining which parts of a DNA sequence are constructing the whole gene from its start site to its stop codon.

As it is the general case in most works, we are only concerned in recognizing the coding part of the gene. However, as it is usual in the literature, we will use the term gene as synonym of the coding part of the gene. In such cases when we want to refer to the whole biological gene we will state that explicitly.

6.2 Evolutionary framework for gene recognition

The approach presented in this paper considers the gene recognition task as an optimization problem, where, given a DNA sequence, the goal is finding on it the most likely correct gene structure, by using the different measures or hints described in the literature that show the difference in characteristics between the types of regions. Each putative gene structure is modeled as a series of not overlapped intervals in the sequence. Thus, the problem can be formalized as a process of identifying a set of intervals in an input sequence, where the intervals represent putative coding regions, and the set the gene structure.

Formally, given a genomic DNA sequence *seq*, we propose to compute a set of segmentations over it. Each segment of such set can be represented by a sequence of intervals

$$(b, e, S, S', C) \tag{6.2}$$

characterized by start b and end e coordinates in seq, the types of the site S and S' at these positions, and the class C of the interval. The next segment must begin at $e + 1$ position downstream from previous one. The signal at the beginning of any segment needs to be identical to that at the end of the preceding segment. The set of intervals should covering the whole length of seq.

However, the huge size of the resulting space of solutions make it not possible the exhaustive search process. At this point, we consider to use suitable computational intelligence methods to deal with the task which can provide great approximations of the results of the problem under consideration and also work well in situations of incompleteness and uncertainty data. In particular, the approach presented in this paper combines two basic methods. First, following the ideas introduced in Chapters 2, 4 and 5, functional site recognition models are used to localize the functional sites along the genomic sequence. We have shown that such models, trained using more than two class, combining several classifiers from several sources of evidences, and the undersampling method are the best current method for predicting the functional sites of a genomic sequence. By using these models, it possible to predict the most probable putatives funcional sites in a target DNA fragment, and consequently, to limit the search space. Specifically, to obtain the site recognition models by a supervised training process, we consider every actual sites and canonical but false sites that were found in the set as positive and negative patterns for training respectively. Because we had a class imbalance problem (ratio 1:1000 between positive and negative patterns), an random undersampling process was employed to deal with the situation. By applying theses models with the setting a certain threshold experimentally determined, a set of most likely start, stop and splice sites is obtained.

In theory, we can determine that each consistent pair of predicted sites defines a potential region since its boundaries are defined. If only this set of sites are taken into account to make up an new set of potential exons, the number of potential complete gene structures decreases immensely. Additionally, the biological constraints should be beared in mind, such as the rules of a correct gene structure previously described, the maximal and minimal exon/intron lengths or the maintenance of the ORF, so the reduction is even bigger.

Once the space of possible solutions is reduced, the search to identify the most likely gene structure could be carried on. At this point, a second computational intelligence technique was used in order to complete such task.

Actually, it could be considered the main part of the presented approach due to it is a framework where many evidences of the presence of genes are combined. Based on Evolutionary Computation (EC), this framework works with a population (set) of individuals (solutions to the problem faced) which are codified following a code similar to the genetic code of plants and animals. This population of solutions is evolved (modified) over a certain number of generations (iterations) until the defined stop criterion is fulfilled. Each individual is assigned a real value that measures its ability to solve the problem, which is called its *fitness*. In each iteration new solutions are obtained combining individuals (crossover operator) or randomly modifying one individual (mutation operator). After applying these two operators a subset of individuals is selected to survive to the next generation, either by sampling the current individuals with a probability proportional to their fitness, or by selecting the best ones (elitism). The repeated processes of crossover, mutation and selection are able to obtain increasingly better solutions for many problems of artificial intelligence.

The choice of using EC comes from the flexibility and simpleness of the paradigm and its power for solving complex tasks. The search is carried out considering many sources of evidence, as the signals that identify gene regions are subtle and must be combined to accumulate enough information to assure with a high probability that an exon or an intron are found. However, the problem is not multi-objective in the usual way. The combination of evidence is used, but each single source is not useful by itself. Thus, the framework was designed under the guideline of a standard genetic algorithm, whose fitness function is a combination of different measures.

At each generation the standard steps of a generational genetic algorithm are carried out:

- **Selection.** Selection is performed using binary tournament, to avoid too much selective pressure, and taking care of maintaining the balance among the number of exons of the individuals. The subpopulations with different number of exons are kept with the same number of individuals. Elitism is applied to avoid loosing the best solutions so far.

- **Crossover.** Crossover is carried out randomly recombining the exons of two parents to obtain two offspring. The offspring substitutes their parents.

- **Mutation.** This operator consists of randomly removing, adding or exchanging an exon. After mutation the individual is checked to assure its viability.

The algorithm is a standard generational genetic algorithm [68] with the particularity that we have forced an even distribution of the lengths of the genes, in terms of number of exons, to avoid a premature convergence to a suboptimal solution.

When the stop criterion is reached, a fine tuning process is carried out by performing a local search over the individual with the best fitness. It consists, first, on evaluating the fitness of such individual incorporating every possible exon from the exon set suitable to be added. Secondly, on evaluating the fitness of the individual when each exon that composes it is removed. If a better solution is found, that new structure is considered as the output of the process.

Finally, a layer based on the content sensors principles is included. Such layer considers a binary task in which each exon of the final individual is evaluated by a SVM model with a numeric kernel. This model uses the frequency of the codon contained in the exon to be generated.

6.3 Experimental setup

In this section we show the details of the application of our method to human genome gene prediction.

6.3.1 Human gene prediction

To test the accuracy of our system, we generated predictions for the chromosomes 1, 13 and 21 from August 2014 build of the human genome (NCBI build 104/ Assembly GRCh38). We used the consensus coding sequence (CCDS) annotations of such chromosomes as our set of known genes. This set contained the features specified on Table 6.1. A three-fold cross-validation procedure was used to estimate how well are the predicts genes not present in its training set. The chromosome 16 was used to validate several aspects in the system components. The complementary genomic data besides human we used on various layers of the system, came from other species: Bos primigenius taurus (BT), Callithrix jacchus (CJ), Canis lupus familiaris (CLF), Danio rerio (DR), Equus caballus (EC), Gallus gallus (GG), Macaca mulatta (MaM), Pan troglodytes (PT) and Sus scrofa (SS).

	Size	Genes	Gene Density (x1000)
Chr 1	226.280.621 bps	2807 genes	.012
Chr 13	95.589.878 bps	413 genes	.004
Chr 21	35.158.702 bps	312 genes	.009

Table 6.1: Features of data used as testing dataset.

6.3.2 Components

As previously stated, the core features of our algorithm can be grouped in three classes: functional site recognition, the estimation of sequence composition of a given segment, and the definition of a gene structure by a global evolutionary framework.

Due to layered architecture, our approach is a very flexible system. One advantage of this is that separate datasets can be used for training its individual parts (e.g., the site detectors). In contrast to less modular systems, it can thus exploit the available data to a fuller extent. As another benefit, the architecture readily allows the integration of additional information from diverse data sources.

Site recognition

To recognize sites, we developed a mechanism according to the principles described on Chapter 4 and Chapter 5 that computed the set corresponding putative site found in the target sequence. Combining models based on considering as many different sources of evidence as possible and as many different classifiers as needed were used to TIS and stop codon recognition because it has been shown to perform with superior accuracy compared with the best methods described in the bibliography. In the case of splice site recognition, the recognition models were built up by using more than one SVM classifier, dividing the negative class into four different groups and trains one classifier for each type of negative class and then, combining their output in the most efficient way. Each of such site model was used to go over the sequence and determining as an independent binary classification task completed by the models. All detail about model construction set up is specified on Table 6.2. All models were trained using undersampling method to address the class-imbalance nature of the datasets.

Content sensors

Content sensors are designed to discriminate sequence segment class by analyzing the base composition of the segments. In this sense, we have used two different approaches to use this type of information. First, we have taken the resultant scores of BLAST alignments to drive the evolutionary algorithm search. Secondly, we have construct a SVM model with a gaussian kernel to make the discrimination between coding and non coding segments in base on the frequencies of the codons that they are formed. With this SVM, we set up a binary classification problem once the gene structure evolutionary framework is finished, in order to carry on a fine tune of the given solution.

To avoid the influence of different length distributions of the segments on the discrimination, both techniques have a mechanism to deal with it. In the case of the BLAST score, the actual used score is obtained from the expression:

$$scoreBLAST(seq) = blastn(seq) * \frac{length(blastn(Aligment(seq)))}{length(seq)} \tag{6.3}$$

In the case of SVM, the model results from patterns of 64 dimensions, where each dimension represents the relative frequency of the corresponding codon on the region. A standard centering and normalization on the unit dispersion procedure is applied, i. e., $\widetilde{x_{is}} = \frac{x_{is}-m_s}{\sigma_s}$, where $\widetilde{x_{is}}$ is the value of the sth coordinate of the ith point after normalization, m_s is the mean value of the sth coordinate, and σ is the standard deviation of the sth coordinate. The training data came from the predictions made over chromosome 16, where the codon frequencies of each correctly predicted exons were used as positive patterns and the codon frequencies of each wrong predicted exons used as negative patterns.

The BLAST alignments were performed over a database composed by the genes CDS of the genomes of three organisms: Canis Lupus Familiaris, Macaca Mulata and Pan Troglodites.

Test	Site	Approach	Combin.	Classifier	Organism	Window	Setup
Chr1	TIS	Stepwise	Sum	kNN	CLF	[-50:50]	k: 25
				kNN	EC	[-50:50]	k: 25
				StopCodon	EC	[0:203]	
				SVM WD	CJ	[-50:50]	d: 24
				SVM WD	MaM	[-50:50]	d: 24
				SVM WD	PT	[-50:50]	d: 24
	Stop Codon	Stepwise	Sum	kNN	CLF	[-50:50]	k: 25
				StopCodon	RN	[-203:0]	
				SVM WD	CJ	[-50:50]	d: 24
				SVM WD	MaM	[-50:50]	d: 24
				SVM WD	PT	[-50:50]	d: 24
	Acceptor	MoreThanTwo	Sum	SVM WD	HS – Pos vs Intergenic Neg.	[-25:75]	d: 6
				SVM WD	HS – Pos vs Intron Neg.	[-25:75]	d: 6
				SVM WD	HS – Pos vs Intergenic+Intron Neg.	[-25:75]	d: 6
	Donor	MoreThanTwo	Sum	SVM WD	HS – Pos vs Intergenic Neg.	[-75:25]	d: 6
				SVM WD	HS – Pos vs Intron Neg.	[-75:25]	d: 6
				SVM WD	HS – Pos vs Intergenic+Intron Neg.	[-75:25]	d: 6
Chr13	TIS	Stepwise	Sum	kNN	CLF	[-50:50]	k: 25
				kNN	SS	[-50:50]	k: 25
				StopCodon	EC	[0:203]	
				SVM WD	EC	[-50:50]	d: 24
				SVM WD	MaM	[-50:50]	d: 24
				SVM WD	PT	[-50:50]	d: 24
	Stop Codon	Stepwise	Sum	kNN	EC	[-50:50]	k: 25
				StopCodon	RN	[-203:0]	
				SVM WD	CJ	[-50:50]	d: 24
				SVM WD	MaM	[-50:50]	d: 24
				SVM WD	PT	[-50:50]	d: 24
	Acceptor	MoreThanTwo	Sum	SVM WD	HS – Pos vs Intergenic Neg.	[-25:75]	d: 6
				SVM WD	HS – Pos vs Intron Neg.	[-25:75]	d: 6
				SVM WD	HS – Pos vs Intergenic+Intron Neg.	[-25:75]	d: 6
	Donor	MoreThanTwo	Sum	SVM WD	HS – Pos vs Intergenic Neg.	[-75:25]	d: 6
				SVM WD	HS – Pos vs Intron Neg.	[-75:25]	d: 6
				SVM WD	HS – Pos vs Intergenic+Intron Neg.	[-75:25]	d: 6
Chr21	TIS	Stepwise	Sum	kNN	EC	[-50:50]	k: 25
				kNN	GG	[-50:50]	k: 25
				StopCodon	EC	[0:203]	
				SVM WD	BT	[-50:50]	d: 24
				SVM WD	MaM	[-50:50]	d: 24
				SVM WD	PT	[-50:50]	d: 24
	Stop Codon	Stepwise	Sum	kNN	EC	[-50:50]	k: 25
				StopCodon	RN	[-203:0]	
				SVM WD	CJ	[-50:50]	d: 24
				SVM WD	MaM	[-50:50]	d: 24
				SVM WD	PT	[-50:50]	d: 24
	Acceptor	MoreThanTwo	Sum	SVM WD	HS – Pos vs Intergenic Neg.	[-25:75]	d: 6
				SVM WD	HS – Pos vs Intron Neg.	[-25:75]	d: 6
				SVM WD	HS – Pos vs Intergenic+Intron Neg.	[-25:75]	d: 6
	Donor	MoreThanTwo	Sum	SVM WD	HS – Pos vs Intergenic Neg.	[-75:25]	d: 6
				SVM WD	HS – Pos vs Intron Neg.	[-75:25]	d: 6
				SVM WD	HS – Pos vs Intergenic+Intron Neg.	[-75:25]	d: 6

Table 6.2: Models setup used for site recognition.

Gene structure genetic algorithm framework

We performed the described evolutionary processes to establish the global gene structure framework where the rest of the components take sense. The evolution to obtain the gene structure of each test sequence was performed for 1000 generations where populations of 200 individuals were evolved. The codification of each individual is a string of integers that represents the sites of the gene, i.e. the boundaries of the exons that compose the gene expressed by intervals (see form 6.2). The initial population is randomly obtained from the possible exons set satisfying the biological constraints suitable to a correct gene structure. For example, the minimum and maximum allowed lengths for each class of interval are controlled, as well as the ORF of the CDS defined by the solution is keeping in mind. The initial population is divided into a number of subpopulations where the individuals are placed depending on their number of exons. An individual may migrate to another population when a mutation modifies its length. Some other details, e.g. mutation or crossover probabilities, are specified under author request.

Fitness function

In this approach we have used a fitness function that is as simple as possible. It must be borne in mind that our main objective is developing a system to prove the validity of evolutionary computation as a tool for gene recognition. We are not creating a system competitive with current gene recognizers, which make use of very complex measures.

The fitness of the possible solutions represented by the individuals are calculated evaluating each segment regarding its type and the global CDS defined by the intervals. In this way, exons and introns are evaluated differently. In order to test two type of bias, one of them tends to be prone to over-predicting and the other to be more restrictive, we have defined two fitness function. On one hand, the more prone expression was defined as:

$$Fitness(ind) = scoreBLAST(CDS(ind)) + \sum_{i=1}^{n} scoreBLAST(exon_i) \quad (6.4)$$

where n is the total number of fragments that ind represents in its genotype as exons. This fitness function allows as good the inclusion of exons with at least a bit of significance. Introns are not specifically valued at themselves in the expression. However, if CDS contains non coding fragments they will

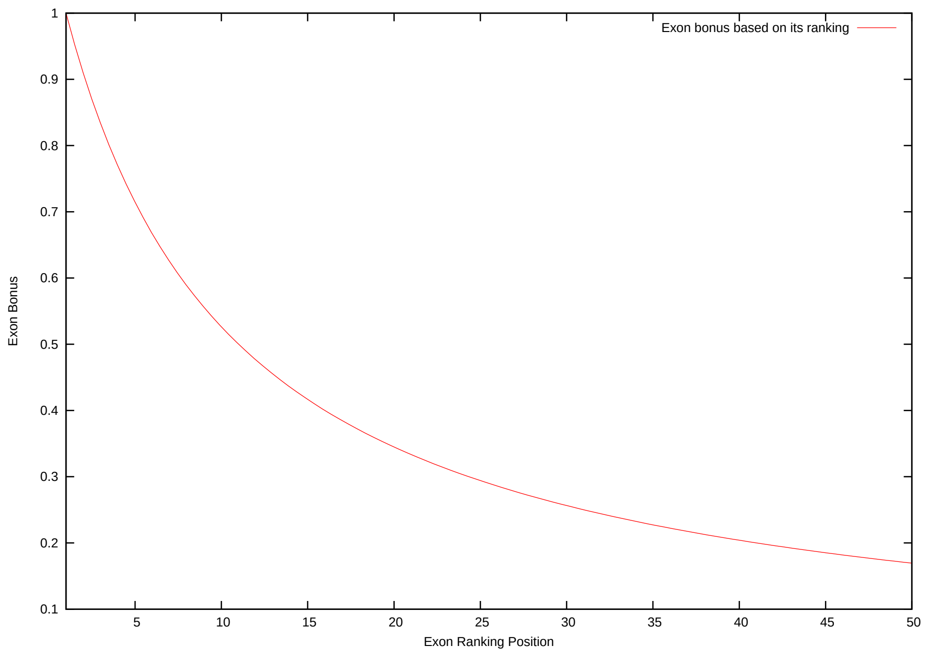


Figure 6.1: Exon bonus obtained by an exon depending on its ranking.

be penalized by the fact that the expression 6.3 is dependent of the length of the sequence.

On the other hand, the fitness function that limits the inclusion of coding regions in the prediction was defined as:

$$Fitness(ind) = scoreBLAST(CDS(ind)) + \sum_{i=1}^{n} scoreBLAST(exon_i) * \frac{10}{9+n+i} \tag{6.5}$$

where *exon* is a ordered array that contains the set the exons hold by *ind*. The order of such array is determined by the $scoreBLAST$. This fitness function give better score to the individual that include exons with a consistent $blastScore$, and restricting the inclusion of exons with low significance.

6.3.3 Evaluation

In general, one of the most important difficulties in research on computational methods is to evaluate their performance in comparison to previously published works with as little bias as possible. This situation is particularly prominent on gene preditors due to even the definition of a gene is still under discussion. Moreover, tiny differences in evaluation procedures can cause important differences in the result of the comparison between predictor systems. Finally, although standardized datasets for training and evaluation are usually used to measure the accuracy, they are often insufficient.

To evaluate the gene predictors performance over a test sequence, the predicted gene structure is compared with the annotated gene structure on the target sequence. The accuracy is evaluated at different levels of resolution. Commonly, these levels are the nucleotide, exon and gene levels. Due to the imbalance nature of the problem, the two useful measures that can offer useful views of the performance at each level are Sensitivity and Specificity. They are based on the concept of True Positive (TP), the total number of coding elements correctly predicted, True Negative (TN), the number of correctly predicted non coding elements, False Positive (FP), the number of non coding elements predicted as coding, and False Negative (FN), the number of coding elements predicted as non coding. Sensitivity (Sn) is defined as:

$$Sn = \frac{TP}{TP + FN} \tag{6.6}$$

This measure is relevant if we are interested only in the performance on the positive class. Specificity (Sp) is defined as:

$$Sp = \frac{TN}{TN + FP} \tag{6.7}$$

However, neither sensitivity nor specificity by themselves constitute a measure of global accuracy. A good measure that summarizes both at nucleotide level is the correlation coefficient (CC):

$$CC = \frac{(TP)(TN) - (FP)(FN)}{\sqrt{(PP)(PN)(AP)(AN)}} \tag{6.8}$$

where PP are the predicted positives, AP the actual positives, PN the predicted negatives and AN the actual negatives.

At exon level, an exon is considered correctly predicted when both boundaries are correctly predicted. At gene level, a gene is considered as correctly predicted if all its coding exons are precisely identified. A global summary measure at these level could be the geometric mean of sensitivity and specificity. The performance at nucleotide level expresses a measure of prediction

Fitness function	Exon Filter based on Codon Frequency	Label
1:1	✓	GA:BLAST11+CF
10:9	✓	GA:BLAST109+CF
1:1		GA:BLAST11
10:9		GA:BLAST109

Table 6.3: Labels of system configurations.

in terms of content capacity. At exon level, the signal and content prediction ability is measured. Finally, the gene level is the measure to evaluate the general performance of the system and to maximize this parameter is the actual goal.

6.4 Results

The performance of four configurations of the system were tested. The details of each one of them are summarized on Table 6.3. The label 1:1 denotes the usage of expression 6.4 as fitness function. On the other hand, 10:9 label denotes the usage of expression 6.5 for such purpose. The application of the SVM model as exon filter based on codons frequency at the final stage of the system is marked by a ✓ in the pertinent column.

The results obtained for each configuration in the experimental process are shown in Table 6.4, Table 6.5, Table 6.6 and Table 6.7. Tables show the correlation coefficient at nucleotide level, sensitivity and specificity at nucleotide and at exon levels, and the number of genes completely predicted for each configuration in predictions made over chromosomes 1, 13 and 21 of the human genome.

The results at nucleotide level are very promising if we take into account the fact that our program is using less than two thousand sequences to learn all its parameters. To get a clearer idea of the performance of our system, Figure 6.8 shows the results published in Gross et al.[27]. At nucleotide level our program in competitive, if we consider that CONTRAST is trained using 11 complete genomes.

	GA:BLAST11+CF			
	Chr1	Chr13	Chr21	Summary
Complete Genes	279	44	41	364
Exon Sn	.389	.346	.519	.418
Exon Sp	.665	.506	.535	.569
Nuc Sn	.512	.618	.633	.587
Nuc Sp	.978	.954	.966	.966
Nuc CC	.534	.480	.547	.520

Table 6.4: Results of GA:BLAST11+CF configuration for chromosomes 1, 13 and 21 at different levels.

	GA:BLAST109+CF			
	Chr1	Chr13	Chr21	Summary
Complete Genes	268	45	40	353
Exon Sn	.354	.403	.460	.406
Exon Sp	.816	.603	.725	.715
Nuc Sn	.500	.565	.613	.559
Nuc Sp	.980	.971	.972	.974
Nuc CC	.550	.526	.580	.552

Table 6.5: Results of GA:BLAST109+CF configuration for chromosomes 1, 13 and 21 at different levels.

	GA:BLAST11			
	Chr1	Chr13	Chr21	Summary
Complete Genes	232	42	39	313
Exon Sn	.383	.268	.364	.338
Exon Sp	.444	.503	.553	.500
Nuc Sn	.539	.587	.636	.587
Nuc Sp	.879	.844	.886	.870
Nuc CC	.442	.407	.464	.438

Table 6.6: Results of GA:BLAST11 configuration for chromosomes 1, 13 and 21 at different levels.

	GA:BLAST109			
	Chr1	Chr13	Chr21	Summary
Complete Genes	231	41	40	312
Exon Sn	.380	.374	.449	.401
Exon Sp	.594	.491	.599	.561
Nuc Sn	.509	.519	.586	.538
Nuc Sp	.895	.845	.885	.875
Nuc CC	.476	.445	.507	.476

Table 6.7: Results of GA:BLAST109 configuration for chromosomes 1, 13 and 21 at different levels.

	N-SCAN (mouse)	CONTRAST (mouse)	CONTRAST (11 informats)
Gene Sn	.356	.508	.586
Gene Sp	.251	.293	.355
Exon Sn	.842	.908	.928
Exon Sp	.646	.705	.725
Nucleotide Sn	.908	.960	.969
Nucleotide Sp	.679	.700	.720

Table 6.8: Results reported in Gross et al.[27].

6.5 Summary

In this work we have shown that EC can be efficiently used for gene prediction. We achieved performance comparable with the best in the literature with a complexity far from the commonly used in gene prediction programs, showing that a prediction program based on EC can be very efficient with a significantly simpler setup. Once that is proved, all the power of EC techniques can be used to improve the current gene recognition programs.

These promising results and the flexibility of the methodology will provide a tool that can deal with alternative splicing, non-canonical functional sites, ignored stop codons, pseudo-genes, and any other issue that need to be addressed in the search process.

The proposed methodology opens a new field of application of genetic algorithms to gene structure prediction. Many new sources of evidence can be added to the system, as well as more sophisticated evolutionary methods. Additionally, we are also studying a multi-objective approach to the gene finding problem.

Chapter 7

Conclusions

By means of the work presented in this thesis we have shown the suitability of state-of-the-art machine learning techniques to address different aspects of the gene prediction problem. We have used the most powerful soft-computing and data mining approaches to address the high complexity of the problem. All tasks that are involved in the gene prediction problem have been subject to study and improvement. Specifically, to achieve this aim, the following particular objectives have been reached:

- Advanced techniques of metaheuristics have been used to design a gene prediction framework. Given the particularities of the problem, the principles of evolutionary computation have been taken under consideration to develop and implement a gene prediction system that has been able to contemplate the set of biological restrictions. It has support a correct gene model that is flexible enough to integrate several independent information sources, regardless of their origin and nature, with a search power sufficient to address the difficult and large solution space. Our results show that with a very simple program we are able to achieve very good accuracies in the recognition of genes in human chromosomes. This is the first evolutionary computation approach to be used for full scale gene prediction.

- We have faced the recognition of the different functional parts of the gene, such as translation initiation sites, donors, acceptors and stop codons, a fundamental component of any gene finding system. The functional site recognition problem has been addressed by using machine learning techniques with additional biological information and/or introducing conservative principles from the evolution theory. This step was particularly divided as follows:

1. The imbalanced nature of the site recognition problem has been addressed. In genome sequences, the number of negative instances is much larger than the number of positive instances. Most learning algorithms expect a somewhat balanced distribution of instances among the different classes and it has been shown that they suffer from a skewed distribution that is associated with class imbalance, resulting in negative effects on their performance. With the studies presented in this work, it has been demonstrated an advantage of class imbalance methods with respect to the same methods applied without considering the class imbalance nature of the problem. The applied methods are also able to improve the results obtained with the best methods in the literature. Finally, the results show that simple random undersampling is a very competitive method when compared with more complex ones.
2. We have carried out a study of the relevance of the different features, the nucleotides and codons that form the sequences, used for recognizing functional sites by means of feature selection techniques, and we valued this kind of methods as they are a useful tool for improving the performance of site recognition. In this way, in our experiments using sequences from various genomes, SVM-RFE has shown is ability to, first select the correct window for learning the classifiers, and them to obtain a subset of features able to improve the results of all of them.
3. We got a improved performance of the current site recognition classifiers by introducing principles derived from information theory and biological aspects. In particular, we have proposed the training of different classifiers with different negative, more homogeneous, classes and the combination of these classifiers for improved accuracy. Such method, tested on recognizing both translation initiation sites and stop codons on the whole human genome, achieves better accuracy than the best state-of-the-art method in the rates of both false negatives and false positives.

4. A methodology for site recognition was proposed by combining classifiers that considers as many different sources of evidence as possible from several genomes and as many different classifiers as needed. Previous studies have shown that the combination of different sources of evidence is fundamental for the success of any genomic recognition task. This idea was feasible because the collection of large and complex genomes from different species has become routine. Our approach has proven to be a powerful tool for improving the performance of functional site recognition, and it is a useful method for combining many sources of evidence for any recognition task in Bioinformatics. The results also show that the common approach of heuristically choosing the species to be used as source of evidence can be improved because the best combinations of genomes for recognition were those not usually selected. In fact, we have shown that species not usually used for site prediction are useful for improving the overall performance of the human genome gene prediction.

References

[1] Arumugam M, Wei C, Brown R, Brent M. Pairagon+N-SCAN-EST: a model–based gene annotation pipeline. Genome Biol. 2006;7(Suppl 1):S5.

[2] Bafna V, Huson DH. The conserved exon method for gene finding. Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology. pp. 3–12. 2000.

[3] R. Barandela, J. L. Sánchez, V. García, and E. Rangel. Strategies for learning in class imbalance problems. *Pattern Recognition*, 36:849–851, 2003.

[4] A. Baten, B. Chang, S. Halgamuge, and J. Li. Splice site identification using probabilistic parameters and svm classification. *BMC Bioinformatics*, 7, 2006.

[5] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1/2):105–142, July/August 1999.

[6] Batzoglou S, Pachter L, Mesirov JP, Berger B, Lander ES. Human and mouse gene structure: comparative analysis and application to exon prediction. Genome Res.; 10(7):950-8. 2000.

[7] A. Bernal, K. Crammer, A. Hatzigeorgiou, and F. Pereira. Global Discriminative Training for Higher-Accuracy Computational Gene Prediction. PLoS Computational Biology 3(3):e54. 2007.

[8] Birney E, Clamp M, Durbin R. GeneWise and Genomewise. Genome Res. 14:988. 2004;

[9] M. Brent. Steady progress and recent breakthroughs in the accuracy of automated gene annotation. *Nature Reviews Genetics*, 9:62–73, 2008.

[10] M. R. Brent and R. Guigó. Recent advances in gene structure prediction. *Current Opinion in Structural Biology*, 14:264–272, 2004.

[11] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268:78–94, 1997.

[12] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[14] O. Cordón, F. Herrera, and T. Stützle. A review of the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware & Soft Computing*, 9:141–175, 2002.

[15] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57–78, 1993.

[16] S. Das and P. N. Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15:4–31, 2011.

[17] S. Degroeve, Y. Saeys, B. D. Baets, P. Rouzé, and Y. V. de Peer. SpliceMachine: predicting splice sites from high-dimensional local context representations. *Bioinformatics*, 21(8):1332–1338, 2005.

[18] Djebali S, Delaplace F, Crollius H. Exogean: a framework for annotating protein-coding genes in eukaryotic genomic DNA. Genome Biol. 7(Suppl 1):S7. 2006.

[19] A. Estabrooks, T. Jo, and N. Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

[20] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc. of the Thirteenth International Conference on Machine Learning*, pages 148–156, Bari, Italy, 1996.

[21] S. García and F. Herrera. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary Computation*, 17(3):275–306, 2009.

[22] N. García-Pedrajas, J. Pérez-Rodríguez, M. García-Pedrajas, D. Ortiz-Boyer, and C. Fyfe. Class imbalance methods for translation initiation site recognition in dna sequences. *Knowledge Based Systems*, 2010. (submitted).

[23] N. García-Pedrajas and D. Ortiz-Boyer. A cooperative constructive method for neural networks for pattern recognition. *Pattern Recognition*, 40(1):80–99, 2007.

[24] N. García-Pedrajas and D. Ortiz-Boyer. Boosting k-nearest neighbor classifier by means of input space projection. *Expert Systems with Applications*, 36(7):10570–10582, September 2009.

[25] N. García-Pedrajas, J. Pérez-Rodríguez, M. D. García-Pedrajas, D. Ortiz-Boyer, and C. Fyfe. Class imbalance methods for translation initiation site recognition in dna sequences. *Knowledge-Based Systems*, 25(1):22–34, 2012.

[26] S. S.Gross, M. R.Brent. Using multiple alignments to improve gene prediction. Proceedings of the Ninth Annual International Conference on Research in Computational Molecular Biology (RECOMB 2005) 2005.

[27] S. S. Gross, C. B. Do, M. Sirota, and S. Batzoglou. CONTRAST: a discriminative, phylogeny-free approach to multiple informant de novo gene prediction. *Genome Biology*, 8(12):R269.1–R269.16, 2007.

[28] Guigó R, Flicek P, Abril JF, Reymond A, Lagarde J, Denoeud F, Antonarakis S, Ashburner M, Bajic VB, Birney E, Castelo R, Eyras E, Ucla C, Gingeras TR, Harrow J, Hubbard T, Lewis SE, Reese MG. EGASP: the human ENCODE Genome Annotation Assessment Project. Genome Biol. 2006;7 Suppl 1:S2.1-31. 2006.

[29] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, March 2002.

[30] Harrow J, Nagy A, Reymond A, Alioto T, Patthy L, Antonarakis SE and Guigó R Identifying protein coding genes in genomic sequences.Genome Biol 10, 201. 2009.

[31] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *TNN*, 13(2):415–425, March 2002.

[32] G. Q. Hu, X. Zheng, H. Q. Zhu, and Z. S. She. Prediction of translation initiation site for microbial genomes with tritisa. *Bioinformatics*, 25:123–125, 2009.

[33] J. V. Hulse, T. M. Khoshgoftaar, and A. Napolitano. An empirical evaluation of repetitive undersampling techniques. *International Journal of Software Engineering and Knowledge Engineering*, 20(2):173–195, 2010.

[34] N. Japkowicz. The class imbalance problem: significance and strategies. In *Proceedings of the 200 International Conference on Artificial Intelligence (IC-AI'2000): Special Track on Inductive Learning*, volume 1, pages 111–117, Las Vegas, USA, 2000.

[35] A. Khare and S. Rangnekar. A review of particle swarm optimization and its applications in solar photovoltaic system. *Applied Soft Computing*, 13:2997–3006, 2013.

[36] M. Kubat, R. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30:195–215, 1998.

[37] L. Kuncheva. A theoretical study of six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281–286, 2002.

[38] L. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, May 2003.

[39] L. I. Kuncheva. Combining classifiers: Soft computing solutions. In S. K. Pal and A. Pal, editors, *Pattern Recognition: From Classical to Modern Approaches*, pages 427–451. World Scientific, 2001.

[40] S. Lee. Noisy replication in skewed binary classification. *Computational Statistics and Data Analysis*, 34(2):165–191, 2000.

[41] D. Lewis and W. Gale. Training text classifiers by uncertainty sampling. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information*, pages 73–79, New York, USA, 1998.

[42] X. Li, Y. Yan, and Y. Peng. The method of text categorization on imbalanced datasets. In *Proceedings of the 2009 International Conference on Communication Software and Networks*, pages 650–653, 2009.

[43] C. Ling and G. Li. Data mining for direct marketing problems and solutions. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 73–79, New York, USA, 1998. AAAI Press.

[44] H. Liu, H. Han, J. Li, and L. Wong. Using amino acids patterns to accurately predict translation initiation sites. *In Silico Biology*, 4:255–269, 2004.

[45] J. Liu, Q. Hu, and D. Yu. A comparative study on rough set based class imbalance learning. *Knowledge-Based Systems*, 21(8):753–763, 2008.

[46] X.-Y. Liu, J. Wu, and Z.-H. Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 39(2):539–550, April 2009.

[47] C. Mathé, M.-F. Sagot, T. Schiex, and P. Rouzé. Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Research*, 30(19):4103–4117, 2002.

[48] C. J. Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 36(1):33–58, July 1999.

[49] C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52:239–281, 2003.

[50] P. Narendra and K. Fukunaga. Branch, and bound algorithm for feature subset selection. *IEEE Transactions Computer*, C-26(9):917–922, 1977.

[51] S. K. Pal, S. Bandyopadhyay, and S. S. Ray. Evolutionary computation in bioinformatics: A review. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 36:601–615, 2006.

[52] R. M. Parry, W. Jones, T. H. Stokes, J. H. Phan, R. Moffitt, H. Fang, L. Shi, A. Oberthuer, M. Fischer, W. Tong, and M. D. Wang. k-nearest neighbor models for microarray gene expression analysis and clinical outcome prediction. *The Pharmacogenomics Journal*, 10:292–309, 2010.

[53] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, 1993.

[54] J. R. Quinlan. Bagging, boosting, and c4.5. In *Proceedings if the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI Press and the MIT Press, 1996.

[55] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[56] J. J. Rodríguez and J. Maudes. Boosting recombined weak classifiers. *Pattern Recognition Letters*, 29:1049–1059, 2008.

[57] L. Rokach. Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics & Data Analysis*, 53(12):4046–4072, 2009.

[58] G. Rätsch, S. Sonnenburg, and B. Schölkopf. RASE: Recognition of alternative spliced exons in *c. elegans*. *Bioinformatics*, 21(Suppl 1):i369–i377, 2005.

[59] Y. Saeys, T. Abeel, S. Degroeve, and Y. V. de Peer. Translation initiation site prediction on a genomic scale: beauty in simplicity. *Bioinformatics*, 23:418–423, 2007.

[60] S. L. Salzberg. A method for identifying splice sites and translational start sites in eukaryotic mrna. *Computational Applied Bioscience*, 13:365–376, 1997.

[61] C. R .Sanna, W. Li and L. Zhang. Overlapping genes in the human and mouse genomes. BMC Genomics 2008, 9:169.

[62] B. Schölkopf, A. Smola, R. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

[63] Sh.-J., A. D. Mathew, Y. Chen, L.-F. X. L. Ma, and J. Lee. Empirical analysis of support vector machine ensemble classifiers. *Expert Systems with Applications*, 36(3):6466–6476, 2009.

[64] G. Shakhnarovich, T. Darrell, and P. Indyk, editors. *earest-Neighbor Methods in Learning and Vision: Theory and Practice.* Neural Information Processing. MIT Press, 2006.

[65] 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.

[66] Stanke M, Waack S. Gene prediction with a hidden Markov model and a new intron submodel. Bioinformatics. 19:2. 2003.

[67] Y. Sun, M. S. Kamel, A. K. Wong, and Y.Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[68] G. Syswerda. A study of reproduction in generational and steady-state genetic algorithms. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann, 1991.

[69] Y. Wang, J. Liu, T. Zhao, and Q. Ji. Recognizing translation initiation sites of eukaryotic genes based on the cooperatively scanning model. *Bioinformatics*, 19:1972–1977, 2003.

[70] Wei C, Brent M. Using ESTs to improve the accuracy of de novo gene prediction. BMC Bioinformatics. 2006.

[71] K. Woods, W. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:405–410, 1997.

[72] F. Zeng and R. H. C. Yap. Using feature generation and feature selection for accurate prediction of translation initiation sites. *Genome Bioinformatics*, 13:192–200, 2002.

[73] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering support vector machines kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.