2007

Prediction of Alternative Splice Sites in Human Genes

Douglas Simmons
San Jose State University

PREDICTION OF ALTERNATIVE SPLICE SITES IN HUMAN GENES

A Thesis

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Douglas A. Simmons

May 2007

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Sami Khuri

Dr. Soon Tee Teoh

Dr. Arthur Grossman, Carnegie Institution of Washington

APPROVED FOR THE UNIVERSITY

ABSTRACT

PREDICTION OF ALTERNATIVE SPLICE SITES IN HUMAN GENES

by Douglas A. Simmons

This thesis addresses the problem of predicting alternative splice sites in human genes. The most common way to identify alternative splice sites are the use of expressed sequence tags and microarray data. Since genes only produce alternative proteins under certain conditions, these methods are limited to detecting only alternative splice sites in genes whose alternative protein forms are expressed under the tested conditions.

I have introduced three multiclass support vector machines that predict upstream and downstream alternative 3' splice sites, upstream and downstream alternative 5' splice sites, and the 3' splice site of skipped and cryptic exons. On a test set extracted from the Alternative Splice Annotation Project database, I was able to correctly classify about 68% of the splice sites in the alternative 3' set, about 62% of the splice sites in the alternative 5' set, and about 66% in the exon skipping set.

Table of Contents

Introduction

The sequencing of the human genome was competed several years ago, sooner than was expected. Determining the sequence of protein coding regions within the human genome or any other eukaryotic genome is a very difficult problem, for which there is no reliable experimental or computational tools [6]. Computational methods are an important area of on going research for predicting genes. Gene prediction programs have been made that take several structural features of the gene into consideration. One of the most famous gene prediction programs developed in the last several years is GENSCAN by Chris Burge, while a graduate student at Stanford University [9]. GENSCAN uses Hidden Semi-Markov Models to identify promoters, exons, introns, and splice sites in genes. My thesis proposal is to take the output file from GENSCAN and use it to analyze the predicted splice junctions in order to predict which exons that may be involved in an alternative splicing.

The identification of alternative splice sites is important because it is thought that each gene codes, on average, three proteins [55]. Splice sites can usually be identified by four nucleotides at the splice junction, two nucleotides on each side. There are other features surrounding the splice junctions that need to be taken into consideration, therefore it is common to look at more nucleotides surrounding the splice junction. This work identifies some of the discriminative features between the exons that are involved in alternative splicing and the ones that are not. This allows my program to predict alternatively spliced exons, and thus multiple proteins that could be produced by a gene. In my preliminary literature research, I have not been able to find another software tool that predicts alternative spliced exons in a comprehensive manner. Most packages only identify splice sites in general, without differentiating between constitutive exons and alternative spliced exons. My work compares the results of my program against known alternative spliced exons taken from the Alternative Spliced Annotation Project database [57].

Motivation

Most of the current gene prediction programs only identify the most likely gene that could exist in a given sequence. It has been hypothesized that 35-42% of the genes in the human genome encode more than one protein [55]. It was initially predicted that the human genome encoded over 100,000 genes, but the latest estimates are now less than 30,000 genes. The large reduction in the estimated number of genes is due to the identification of alternately spliced genes. There are currently 11,717 alternately spliced human genes compiled in the Alternative Splicing Annotation Project database, second version, (ASAP II) at the University of California Los Angeles (UCLA) [57]. If these estimates are correct, then there is a need to be able to predict these alternative splice sites, and the proteins that may be produced from them. It is especially important for the development of drugs for treating diseases, since it is thought that genes that produce proteins to fight diseases commonly encode multiple proteins [55]. If we are able to predict these alternate proteins, we may be able to find drugs that enhance their production, or have insight into where to look for mutations that may have disabled the gene.

Biology Background

Living organisms carry genetic information in the form of DNA molecules. Recent advances in DNA sequencing technology have led to an explosion of genomic data. Information in cells passes through processes called transcription and translation. Each DNA molecule contains genes, which decide the structural components of cells, tissues, and enzymes for biochemical reactions essential for its survival and functioning. In the process of transcription, genes in the DNA sequences are converted into corresponding mRNA sequences. In the process of translation, the nucleotides in the coding regions are translated to synthesize proteins. In eukaryotic genomes, a gene is structured by a variety of biological features, such as the promoter, start codon, introns, exons, splice sites, stop codon, and the poly-adenylation signal.

The transcription process occurs in several parts. The DNA sequence that codes for a gene is converted to pre-mRNA. The pre-mRNA then moves from the nucleus of the cell to the cytoplasm. In the cytoplasm the introns are spliced out of the pre-mRNA and the mature mRNA is produced. The conversion from pre-mRNA to mRNA is more complex than just removing the introns. Alternate splicing is a widespread mechanism by which a single gene can encode two or more related proteins [35]. There are quite a few programs available that are useful in predicting genes, but none of them, to my knowledge, are able to predict which exons may be alternatively spliced, and the different proteins that may be produced by the gene.

Alternative Splicing in Eukaryotic Genes

In 2000 the initial estimates for the number of human genes was around 153,000 for making about 90,000 proteins. By 2002 after the initial completion of the Human Genome Sequence the estimates for the number of human genes had shrunk to between 30,000 and 35,000 protein coding genes. The current estimates are now less than 25,000 [55]. This means that many of the genes that code for proteins in not only the human genome, but also every eukaryotic organism, code for more than one protein. Some genes such as the Breast Cancer (BRCA1) gene are known to have splice variations that code for around 20 different proteins. It is thought that on average each human gene makes 3 alternatively spliced mRNA [55].

There are five types of alternate splicing variations that can occur in the transformation of pre-RNA to mRNA. These variations are exon skipping, alternate 3' splicing, alternate 5' splicing, mutually exclusive exclusion, and intron retention [7].

Exon skipping occurs when the splicing machinery is blocked, and the exon is left as part of its two bordering introns, forming one larger intron, which is spliced out as a single unit. Figure 1 shows an example of exon skipping.

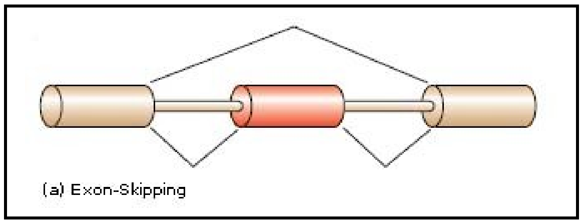


Figure 1. *Exon Skipping* [55]

Alternative 3’ splicing occurs when the exon has two possible sites that can signal the beginning of an exon. There are several additional factors that must be present for an alternative 3’ splice site to be possible. One major feature that must be present for a 3’ splice site is the branch point, which is 10 to 30 base pairs upstream of the splice site. Each 3’ splice site would need a branch point within this given region. See Figure 2.

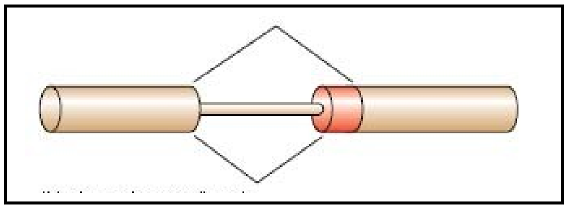


Figure 2. *Alternative 3’ Splice Site* [55]

Alternative 5’ splicing occurs when the exon has two possible sites that can signal the end of an exon. See Figure 3.

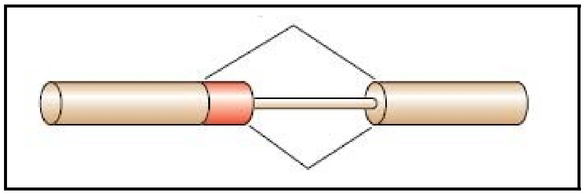


Figure 3. *Alternative 5’ Splice Site* [55]

Mutually exclusive exons are exons that are not both included in the same mRNA. They may be included separately in different mRNA’s or both may be excluded. See Figure 4.

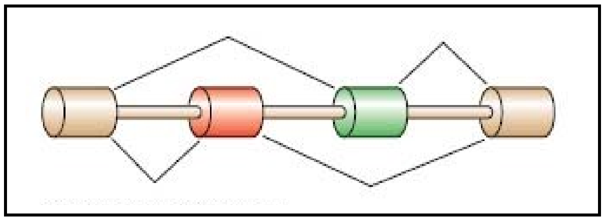


Figure 4. *Mutually Exclusive Exons* [55]

Intron Retention occurs when an intron is not spliced out of the coding sequence and is retained in the mRNA. This occurs very rarely in Mammalian genes, but is more prevalent in plant genes. See Figure 5.

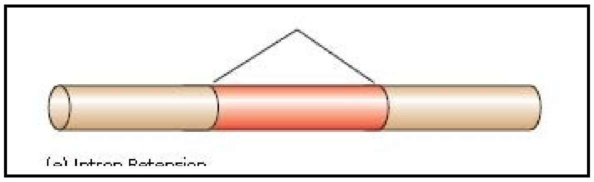


Figure 5. *Intron Retention* [55].

It is estimated that exons, or protein coding regions, only make up 1% to 2% of the entire human genome. There are on average 8.8 exons per gene producing an average of 3 alternate mRNA's.

There are currently several methods used to assemble databases of alternate spliced genes. The most common method is to use Expressed Sequence Tags (EST) to identify areas in the genome where a protein is coded [57].

The splicing of introns from the pre-RNA occurs when the splice machine known as the splicesome attaches before the 5' end of the exon. The splicesome is a highly conserved complex of five small nuclear uridine rich RNA molecules (snRNA – U1, U2, U4, U5, and U6) [56]. The table below shows the target sites and abundance of each of these snRNA's. See Table 1.

snRNP	Splice Target	Abundance
U1	5' junction	Many
U2	branch	1
U5	3' junction	1
U4	?	1
U6	?	1

Table 1. *Abundance of Target Sites* [56].

The cutting at the splice sites involves regulatory proteins called splicing regulator (SR) proteins. There are about 10 different known SR proteins identified so far [55]. These SR proteins can bind to two different nucleotide sequences called Exonic Splicing Enhancers (ESE) or Exonic Splicing Suppressors (ESS).

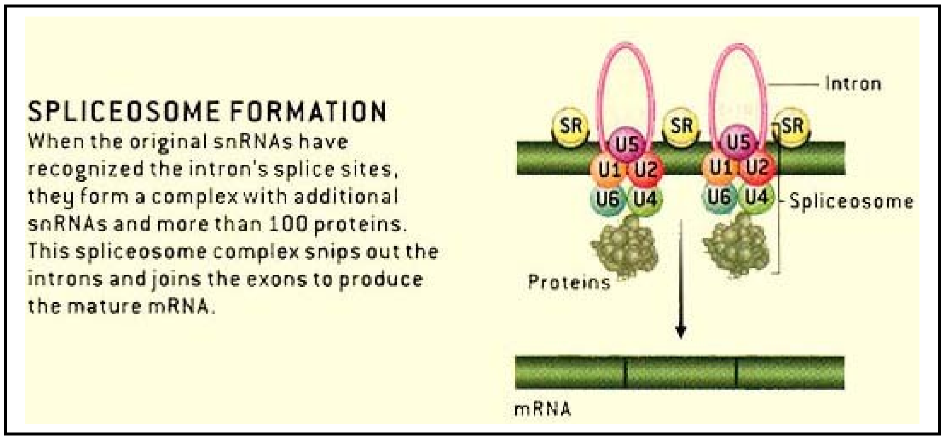

SPLICEOSOME FORMATION
When the original snRNAs have recognized the intron's splice sites, they form a complex with additional snRNAs and more than 100 proteins. This spliceosome complex snips out the introns and joins the exons to produce the mature mRNA.
Intron
SR
U5
U1
U2
U6
U4
Spliceosome
Proteins
mRNA

Figure 6. *Spliceosome Formation* [55].

Figure 6 shows the interactions of the snRNA's of the splicing machinery, and the binding of the SR proteins needed to splice introns from the pre-RNA to form the mRNA.

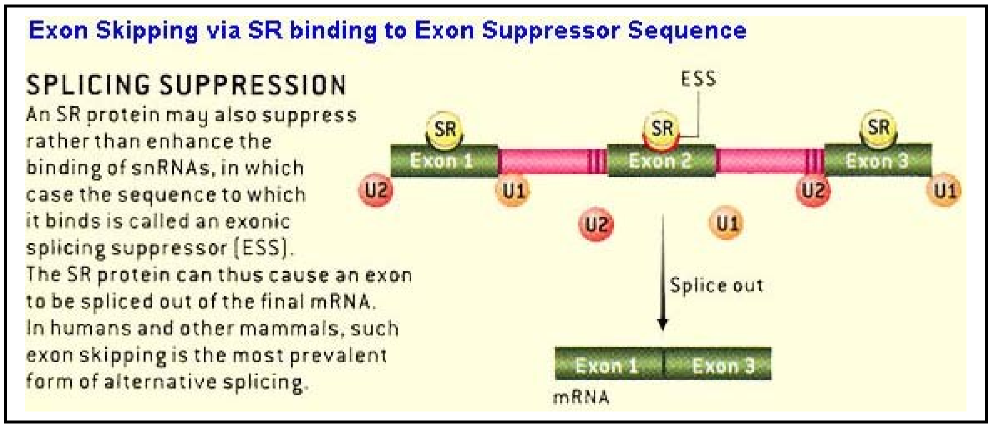

Exon Skipping via SR binding to Exon Suppressor Sequence
SPLICING SUPPRESSION
An SR protein may also suppress rather than enhance the binding of snRNAs, in which case the sequence to which it binds is called an exonic splicing suppressor (ESS). The SR protein can thus cause an exon to be spliced out of the final mRNA. In humans and other mammals, such exon skipping is the most prevalent form of alternative splicing.
ESS
SR
Exon 1
Exon 2
Exon 3
U2
U1
Splice out
Exon 1
Exon 3
mRNA

Figure 7. *Splicing Suppression* [55].

Figure 7 shows how exon skipping can occur when the SR protein in exon 2 binds to an ESS. The binding of the SR protein to the ESS on exon 2 blocks the splicesome machinery from binding near exon 2, thus skipping it and creating an alternate mRNA.

There is currently a lot of research being directed toward understanding the different biological conditions that cause alternate splicing to occur. There are many different factors that can cause the alternate splicing to occur, from tissue type, stress on the cell, fighting disease, etc. There are many factors that can influence which exons are spliced or where they are spliced. The attempt to answer these questions is beyond the scope of this thesis. This thesis predicts possible exons that may have a high probability of being involved in an alternatively spliced mRNA. The goal of this thesis is to predict these exons so the gene they occur in can be examined more extensively in order to analyze the possible proteins that may be produced from alternate splicing. I am attempting to produce a tool that can decrease the possible combinations of proteins that can be produced from alternative splicing.

Literature Review

Most of the literature review I did was to prepare for the topic I had originally proposed. My original thesis proposal was to implement GENSCAN with the addition of incorporating the detection of Transcription Factor Binding Sites (TFBS) into the model. During my literature research I realized that incorporating TFBS into the model would cause the algorithm to become computationally more complex without adding any additional benefits to the model. I will discuss these limitations in more detail in the literature conclusion section.

Through my literature research I discovered a more interesting and more useful tool that needed to be developed. Many of the papers that I read on the topic of TFBS prediction methods, could also be used to predict splice sites. By combining several methods, a useful tool could be built to predict where an alternate splice might occur in a predicted gene or a known gene.

The next section describes the Support Vector Machine algorithm and then reviews some papers that used SVMs and other methods for predicting splice sites in genes.

Support Vector Machine

A Support Vector Machine (SVM) is a supervised machine learning method based on statistical learning theory for classification and regression. This theory was initially proposed by Vapnik [49]. SVMs are useful for solving many biological problems, which involve high-dimensional noisy data, for which SVM's are known to behave well, compared to other statistical or learning machine methods. In contrast to most machine learning methods, Kernel methods like SVM can easily handle non-vector input, such as variable length sequences or graphs [44].

Kernels not only increase the flexibility by increasing the class of allowed similarity measures but also make it possible to work with non-vectorial data. This is due to the fact that kernels automatically provide a vectorial representation of the data in the feature space [44]. It has been shown that kernels can be used to construct generalizations of any algorithm that can be carried out in terms of dot products [44].

Data Representation

A set of objects can be denoted by $S = (\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)$, the set of n objects to be analyzed. If each object $\boldsymbol{x}_i$ is an element of a set X, then X could be a set of all possible images, which one wants to analyze, or the set of all possible molecules in a biological context. The first question that must be addressed is how to represent the data set S for further processing, in order to design the data analysis methods [44].

"The vast majority of data analysis methods, outside of kernel methods, have a natural answer to this question: first define a representation for each object, and then represent the set of objects by the set of their representations. This means that a representation $\phi(\boldsymbol{x}) \in \mathcal{F}$ is defined for each possible object $\boldsymbol{x} \in X$, where the representation, for example, be a real-valued vector ($\mathcal{F} = \mathbb{R}^p$), is a string of finite length, or a more complex representation that can be processed by an algorithm. The data set S is then represented as the set of individual object representations, $\phi(S) = (\phi(\boldsymbol{x}_1),\ldots,\phi(\boldsymbol{x}_n))$, and the algorithm is designed to process such data" [44].

Kernel methods offer a radically different answer to the question of how to represent data. The data can be represented through a set of pairwise comparisons, and does not need to be represented individually anymore. Instead of using a mapping $\phi : X \rightarrow \mathcal{F}$ to represent each object $\boldsymbol{x} \in X$ by $\phi(\boldsymbol{x}) \in \mathcal{F}$, a real-valued "comparison function" $k : X \times X \rightarrow \mathbb{R}$ is used, and the data set S is represented by an $n \times n$ matrix of pairwise comparisons $k_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ [44]. All kernel methods are designed to process such square matrices. The difference between both approaches is represented in the Figure 8.

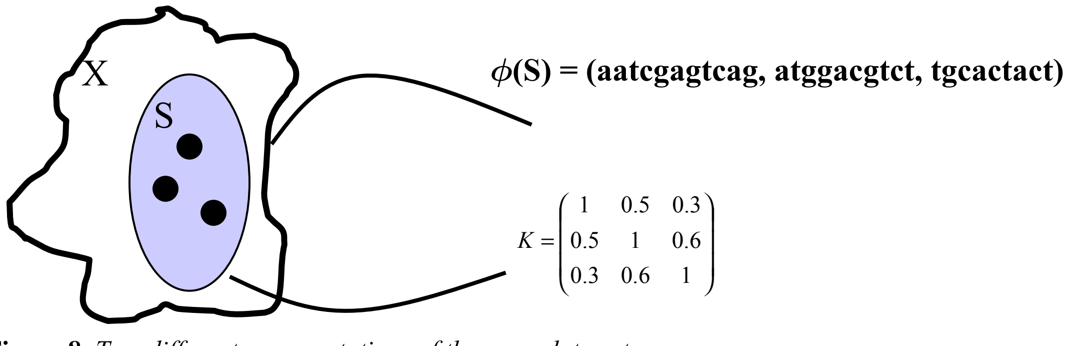

X
S
φ(S) = (aatcgagtcag, atggacgtct, tgcactact)
$K = \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0.5 & 1 & 0.6 \\ 0.3 & 0.6 & 1 \end{pmatrix}$

Figure 8. *Two different representations of the same data set.*
The classic way to represent S is first to define a representation $\phi(\boldsymbol{x})$ for each element of $\boldsymbol{x} \in X$, and then to represent S as the set $\phi(S)$ of representations of its elements (upper part). Kernel methods are based on a different representation of S, as a matrix of pairwise similarity between it elements (lower part) [44].

Second, the nature or the complexity of the objects does not affect the size of the matrix used to represent a dataset of n objects. The size of the matrix is always $n \times n$. For example, if you had microarray data for a set of ten tissues, and each are characterized by thousands of gene expression levels, the kernel matrix would be represented by a 10×10 matrix, no matter how many genes where involved. If you have a small number of objects to be processed this is computationally attractive [44].

Third, there are many cases where finding a way to explicitly represent each object is more difficult than comparing the objects that a given algorithm can process. As an example, many data analysis algorithms, such as least squares regression or neural networks, require an explicit representation of each object $\boldsymbol{x}$ as a vector $\phi(\boldsymbol{x}) \in \mathbb{R}^p$. There are some objects, such as protein sequences, where there is no obvious way to represent the data as vectors in a relevant way, however, there are meaningful pairwise sequence comparison methods which exist [44].

Formal Definition

"Suppose that the data set S consists of a series of objects $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n \in X$, together with a series of labels $\boldsymbol{y}_1,\ldots,\boldsymbol{y}_n \in Y$ associated with the objects. SVMs are kernel methods to learn a function $f: X \rightarrow Y$ from S, which can be used to predict the label of any new object $\boldsymbol{x} \in X$ by $f(\boldsymbol{x})$" [44]. The basis of the SVM is that it takes a set of training vectors $x_{i,}$ and maps them into a higher dimensional feature space by the function ϕ.

The definition of the SVM states that there exists a Kernel function that is equivalent to the dot product of the vectors in the Feature space. The kernel similarity matrix is equivalent to the dot product of the vectors in the feature space [44]:

$$K(x_i, x_j) \equiv \phi(x_i) \bullet \phi(x_j)$$

One of the most important concepts of the SVM is that all that is required to find the linear relations are inner products, so we do not need an explicit representation of the mapping ϕ, nor do we need to know the nature of the feature space. We only need to be able to evaluate the Kernel function.

Evaluating the kernel on all pairs of data items yields a symmetric positive definite matrix K known as the kernel matrix. The kernel matrix can be regarded as a matrix of generalized similarity measures among the data points. Below is a list of four general kernels that are used most often [32]:

1. Linear:

   $K(x_i, x_j) = x_i \bullet x_j$
2. Polynomial:

   $K(x_i, x_j) = (\gamma x_i \bullet x_j + r)^d, \gamma > 0$
3. RBF (Radial basis function):

   $K(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2), \gamma > 0$
4. Sigmoid:

   $K(x_i, x_j) = \tanh(\gamma x_i \bullet x_j + r)$

Training an SVM

Given a training set of instance-label pairs (x_i, y_i), i = 1,…,l where $x_i \in \mathbb{R}^n$ and $y \in \{1, -1\}^l$, the support vector machine requires the solution of the optimization problem.
The following optimization problem needs to be solved:

$$\min \tfrac{1}{2} \|\omega\|^2 + C \sum_i \xi_i \tag{i}$$

such that, $y_i(\omega \bullet \phi(x_i) + b) \geq 1 - \xi_i$, and $\xi_i \geq 0$.
In equation (i), ω is a weight vector that points perpendicular to the separating hyperplane, C is a regularization parameter, b is an offset parameter, and x_i are slack variables [44].

The training vectors, $\boldsymbol{x}_i$, are mapped into a higher (sometimes infinite) dimensional space by the function $\phi(x_i)$. $C > 0$ is the penalty parameter of the error term.
Equation (i) can be solved by solving the following dual problem:

$$\min_{\alpha} F(\alpha) = \frac{1}{2} \alpha \bullet Q\alpha - e \bullet \alpha$$

subject to $0 \leq \alpha_i \leq C$, i = 1, …, l, $\quad y \bullet \alpha = 0$,
where e is the vector of all ones and Q is an l by l positive semi-definite matrix. The (i, j)-th element of Q is given by $Q_{ij} \equiv y_i y_j K(x_i, x_j)$ where $K(x_i, x_j) \equiv \phi(x_i) \bullet \phi(x_j)$ is called the kernel function. Then

$$\omega = \sum_{i=1}^{l} \alpha_i y_i \phi(x_i) \text{ and}$$

$$\operatorname{sgn}(\omega \bullet \phi(x) + b) = \operatorname{sgn}(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b)$$

is the decision function [36].

SVMs are unique in that they focus more on the confidence of the classification than on the number of misclassifications. One way to formalize the comparison between the confidence of the classification and the number of misclassifications is shown in the Figure 9 [44]. "The linear function $f(\boldsymbol{x}) = \omega \bullet x + \mathrm{b}$ defines two half-spaces of points classified positively and negatively with large confidence. The distance between these two half-spaces, called the margin, is exactly equal to $2/\|\omega\|$. If all of the points in the training set S were to be correctly classified with strong confidence by a linear function f with the largest possible margin, this would correspond to the problem of maximizing $2/\|\omega\|$ under the constraint $y_i(\omega \bullet x + \mathrm{b}) \geq 0$ for $i = 1, \ldots, n$" [44].

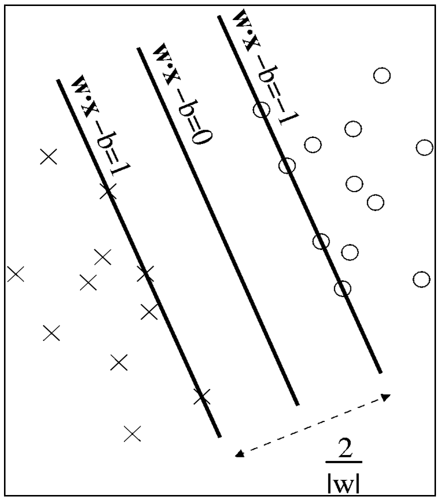


Figure 9. *Half-spaces of Linear Function f*

Scaling Input Vectors

The main advantage of scaling the input is to prevent attributes in greater numeric ranges from dominating those in smaller numeric ranges. Another advantage is that it helps reduce the chance of numerical problems when the inner product is taken on vectors with large attribute values. Scaling each attribute to the range of [-1, 1] or [0, 1] is recommended [32].

Prediction

An unknown vector can be classified by the support vector by the given formula:

$$f(x) = \omega \bullet x + b = \sum_{i=1}^{n} y_i \alpha_i x_i \bullet x + b \qquad \text{(ii)}$$

Equation (ii) will predict the class of x as -1 or $+1$ depending on the sign of this function. This can be simplified by the following decision rule:

$\hat{c} = 1$ if $\omega \bullet x + b \geq 0$ or -1 if $\omega \bullet x + b \leq 0$ [36].

Multi-class Problems

Many real real-world problems require the classification of more than two classes even though the basic SVM algorithm for pattern recognition is define for the classification of objects into two classes. The three SVM models used in this project each use four classes to classify the input sequences. One way to use SVMs in this context is to apply an implementation that specifically solves multiclass problems [44].

"The most widely used method for multiclass problems involves reformulating them as a number of binary classification problems, and solving these problems with binary SVMs. The resulting SVMs must then be combined to form a multiclass prediction algorithm. The most common way to perform and split this combination is called the one-against-all scheme. It consists of first finding discrimination between each class and all the others, thus transforming a problem with N classes into N binary problems. The scores output by each SVM are then combined by a max rule: an object assigned to the class corresponding to the SVM that outputs the largest score" [44].

For example, consider a three-class classification problem with the following training set labels:

$$y = (1,1,1,2,2,2,3,3,3).$$

In the one-against-all scheme this problem is decomposed into three vectors:

$$y_1 = (1,1,1,-1,-1,-1,-1,-1,-1)$$
$$y_2 = (-1,-1,-1,1,1,1,-1,-1,-1)$$
$$y_3 = (-1,-1,-1,-1,-1,-1,1,1,1)$$

Three SVMs are trained on the three class labels respectively. An unknown sample can be classified by combining the outputs of SVMs, and than the sample is assigned to the class with the largest output [44].

The method used in the implementation of the three SVM models was the pairwise classification, where one classifier is learned for each possible pair of classes. It consists in first finding a discrimination between each pair of classes, thus transforming a problem with N classes into (N choose 2 or $N!/2!(N-2)!$) binary problems. The scores output by each SVM are then combined by a max rule: an object assigned to the class corresponding to the SVM that outputs the largest score.

As an example, consider a four-class classification problem with the following training set labels:

$y = (1,1,1,2,2,2,3,3,3,4,4,4).$

In the pairwise classification scheme this problem is decomposed into six vectors:

$y_1 = (1,1,1,-1,-1,-1)$, classes 1, 2
$y_2 = (1,1,1,-1,-1,-1)$, classes 1, 3
$y_3 = (1,1,1,-1,-1,-1)$, classes 1, 4
$y_4 = (1,1,1,-1,-1,-1)$, classes 2, 3
$y_5 = (1,1,1,-1,-1,-1)$, classes 2, 4
$y_6 = (1,1,1,-1,-1,-1)$, classes 3, 4

Six SVMs are trained on the six class labels respectively. When an unknown sample is classified, the outputs of SVMs are combined and the sample is assigned to the class with the largest output [44].

Parameter Setting

In order to use a basic SVM for binary classification, two kinds of parameters have to be determined [44]:

- The regularization parameter C of the SVM.
- The kernel and its parameters.

It is crucial that a proper choice of these parameters is selected to ensure good performance of the algorithm. Overfitting is likely to occur if the parameters are set based on only the performance of the training set used on the SVM. Overfitting occurs when the performance increases on the training set used, but decreases on new samples [44].

"A standard way to fix parameters is to use cross-validation. Given specific values C and γ the k-fold cross-validation error is calculated by randomly dividing the training set into k relatively equal size subsets. The SVM is trained on k-1 subsets and its error rate on the remaining subset is computed. Repeating this process k times, such that each subset is tested once, the cross-validation error is determined by the average of the test errors" [44].

Another procedure is Grid search, where C and γ are selected to minimize the cross-validation error. In this scheme a set of candidate values are chosen both for C and γ, and the cross-validation error is computed for every possible combination of them. If n_c and

n_γ are the number of candidate values, then the cross-validation error is computed $n_c n_\gamma$ times, which means the SVM is trained $k n_c n_\gamma$ times in total [44].

Procedures for Training

The following procedure for creating the models for the SVM as recommended by C. Hsu [32]:

1. Transform data to the format of an SVM software.
2. Conduct simple scaling on the data.
3. Consider the RBF kernel $K(\mathrm{x}, \mathrm{y}) = e^{-\gamma\|\mathrm{x} - \mathrm{y}\|^2}$.
4. Use cross-validation to find the best parameter C and g.
5. Use the best parameter C and g to train the whole training set.
6. Test the models.

This is a general procedure that has been found to give new users of SVMs good results, and avoid some of the common mistakes that can cause lower than expected results.

Methods for predicting DNA Splice Sites

Machine learning approaches to the recognition of DNA Splice Junctions.

Several machine learning approaches have been proposed for recognizing DNA splice junctions. Three of these approaches that were introduced in “Recognition of DNA Splice Junction via Machine Learning Approaches” [34], are Kohonen’s Self-Organizing Maps (KSOM), Back-propagation Neural Networks (BNN), and Support Vector Machines (SVM). There are two variations of the splice sites that need to be identified, the intron/exon border going from 5’ to 3’, also know as the “acceptor” site, and the exon/intron border, known as the “donor” site. Splice site recognition is an important component in the prediction of genes, because there are usually multiple splice sites in eukaryotic genes. It is important to be able to accurately identify these splice sites in order to predict the protein coding region of the gene.

The three machine learning approaches introduced above are divided into two categories, supervised and unsupervised learning. The KSOM is an unsupervised machine learning algorithm. This means that the result is unknown by the algorithm when it is trained by a training set. The Back-propagation Neural Network and the Support Vector Machine are supervised algorithms, meaning that the algorithm knows what the results are for each item in the training set [34].

The methods described above were trained on a training set that contained 1,000 sequences, and tested on a set of 424 sequences. The sequences in the sets were each 32 base pairs in length, containing 15 nucleotides upstream and 15 nucleotides downstream the dinucleotide splice site. The DNA nucleotides were converted into four digit binary code. The nucleotides adenine (A), cytosine (C), guanine (G), thymine (T) were represented as 0001, 0010, 0100, 1000 respectively [34], increasing the length of the input sequences to 128 descriptors (32*4).

The KSOM is an unsupervised learning neural network developed by Kohonen. KSOM transforms input data from a high-dimensional space into a lower-dimensional space in such a way that the topology of the input data is preserved.

Support Vector Machines (SVM) are learning techniques, developed by Vapnik, based on Statistical Learning Theory. SVM is usually used for binary classification where the output can have two possible values (e.g. 0 or 1, -1 or 1). SVM learning comprises of two essential steps. The first involves the use of kernel functions to linearly or non-linearly transform input data from a low-dimensional space to a high-dimensional space. Second, numerous hyperplanes are generated that segregates the data objects into distinctive regions based on the output binary value [34].

The results of the test sets on the three machine learning approaches showed that the supervised learning models were more accurate at predicting the three classes of junction sites, gene to non-gene, non-gene to gene, and no transition. In particular the SVM out performed both the BNN and KSOM in prediction accuracy for each of the three classes, though the BNN had the same prediction accuracy for the no transition class. The SVM seems to be the method that warrants further investigation. The SVM could possibly be used for the application in the prediction of other signals common in gene prediction, such as the promoter region, Transcription Start Site (TSS), Translation Initiation Sites (TIS), and poly-adenylation signals.

Maximum Entropy Modeling

The Maximum Entropy Modeling (MEM) method can be used to predict RNA splicing signals. The maximum entropy method has been used in natural language processing, amino acid sequence analysis, and as a weighting scheme for database searches [11].

The model is based on a 9 nucleotide 5’ splice site (donor), and a 23 nucleotide at the 3’ splice site (acceptor).

The distribution with the largest Shannon entropy, H, is the best approximation of the true distribution, given by the expression:

$$H(\hat{p}) = -\sum \hat{p}(x)\log_2(\hat{p}(x))$$

Let X be a sequence of λ random variables $X = \{X_1, X_2, \ldots, X_\lambda\}$.
Let $x = \{x_1, x_2, \ldots, x_\lambda\}$ represent a specific DNA sequence.

Let p(X) be the joint probability distribution p($X_1 = x_1, X_2 = x_2, \ldots, X_\lambda = x_\lambda$), and P($X = x$) denote the probability of a state in this distribution [11].

There are two categories of constraints:

- Complete constraints, which specifies sets of position dependencies.
- Specific constraints, which are constraints on oligo-nucleotide frequencies at a subset position.

The Maximum Entropy Model (MEM) is specified with a set of "complete" constraints, and has two distributions:

- The signal probability distribution – ($p^+(X)$).
- The decoy probability distribution – ($p^-(X)$).

The MEM can be used to distinguish true from false signals from by the likelihood ratio [11], L,

$$L(X = x) = \frac{P^+(X = x)}{P^-(X = x)}$$

Permuted Markov Models

Short DNA motifs can be found using Permuted Variable Length Markov Models (PVLMM). PVLMM can be used to predict both transcription factor binding sites as well as splice sites.

"By permuting the signal sequence, strongly dependent positions are brought close enough together to permit distant dependencies to be captured in low-order non-stationary Markov models" [32].

Given a DNA sequence of length L: where $\mathbf{x} = x_1, \ldots, x_L$, it is possible to calculate the probability. This probability can be factored into a product of conditional probabilities as follows [33]:

$$P(x) = P(X_1 = x_1)\prod_{l=2}^{L} P(X_l = x_l \mid X_1^{l-1} = x_1^{l-1} = x_1^{l-1}) \qquad \text{(iii)}$$

Equation (iii) shows that the current position in the sequence depends on all its predecessors, however only some of the contexts may be relevant [32].

Equation (iii) can be simplified by assigning a context function c_l to the context x_1^{l-1} that is only relevant in the past:

$$P(x) = P(X_1 = x_1)\prod_{l=2}^{L} P(X_l = x_l \mid c_l(X_1^{l-1}) = c_l(x_1^{l-1})) \qquad \text{(iv)}$$

A permutation, π, can be applied to position 1*, … ,* L*,* by the permuted variable length Markov models (PVLMM) to simplify the component terms in

$$P(x) = P(X_{\Pi(1)} = x_{\Pi(1)}) \times \prod_{l=2}^{L} P(X_{\Pi(l)} = x_{\Pi(l)} \mid X_{\Pi(1)}^{\Pi(l-1)} = x_{\Pi(1)}^{\Pi(l-1)}) \qquad \text{(v)}$$

Equation (v) can then be simplified to equation (vi) by using a PVLMM.

$$P(x) = P(X_{\Pi(1)} = x_{\Pi(1)}) \times \prod_{l=2}^{L} P(X_{\Pi(l)} = x_{\Pi(l)} \mid c_l(X_{\Pi(1)}^{\Pi(l-1)}) = c_l(x_{\Pi(1)}^{\Pi(l-1)})) \qquad \text{(vi)}$$

Permuting the positions in the signal brings important non-adjacent positions together, while at the same time keeping the important local dependence [33].

Methods for Detecting Discriminative Features of Alternative Splicing

Identifying Regulatory Sequence Patterns

A method has been developed for finding the regulatory sequence patterns specific to three of the types of alternative splicing events, alternative 5' splicing, alternative 3' splicing, and exon skipping. This method introduced by H.Sakai [43] identifies regulatory sequence patterns on the alternative exon and its flanking introns.

The method of finding regulatory sequence patterns is performed on DNA sequences and considers the following various patterns; *l*-mers with some mismatches, strings over IUPAC nucleic acid codes (degenerate patterns), and nucleic acid indexing. Conjunctions and disjunctions can be dealt with since the patterns are formulated as binary functions [43]. This method categorizes the constitutive exons as negative examples and all the alternative exons as positive examples.

The method considers four kinds of search regions for alternative 5' and 3' exons. These search regions include the upstream, overlapped exonic, non-overlapped exonic, and downstream regions. The length of the upstream and downstream regions are constrained to 100 nucleotides. The exon skipping and constitutive search regions include the upstream, exonic, and downstream regions. Figure 10 shows where each region is defined.

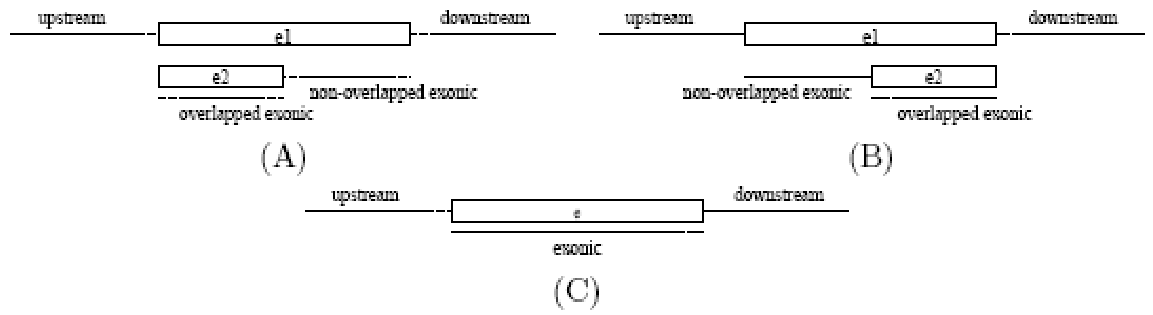


Figure 10. *Search Regions Used for the Discriminative Feature Detection Method* [43].

Pattern Models Used

There are three patterns used in the method for finding regulatory sequence patterns. These patterns are the substring pattern, mismatch pattern, and the degenerate pattern. Some of the details of these patterns are described by H. Sakai [43] below:

- A substring pattern over Σ is a string *p* over Σ. The substring pattern matcher returns true if given a string *t* over Σ if there is at least one occurrence of p in *t*, and false otherwise.
- A mismatch pattern over Σ is a pair of a string *p* over Σ and a nonnegative integer *k*. The mismatch pattern matcher returns true if there is at least one substring of a give string *t* identical to *p* except at most *k* positions, false otherwise.
- A degenerate pattern over Σ is a sequence of subsets of Σ. For a degenerate pattern $p = p_1 p_2 \ldots p_n$ with $p_i \subseteq \Sigma$ for $i = 1,2,\ldots,n$, the degenerate pattern matcher returns true if there is at least one substring $s = s_1 s_2 \ldots s_n$ ($s_i \in \Sigma$) of a given string *t* such that s_i is included in p_i for each $i = 1,2,\ldots,n$, and false otherwise.

A score function *F* of patterns is based on the binary values depending on whether there exists an occurrence of a specified pattern or not [43].

Conclusions From Literature Review

There have been several methods introduced for predicting splice sites, from machine learning methods such as Support Vector Machines, Back-propagation Neural Networks, and Kohonen Self-Organizing Maps, to the Maximum Entropy Modeling and Permuted Variable Length Markov Models. All of the previously mentioned methods only attempt to identify splice sites in general. They do not take into consideration any of the subtle differences that may exist between constitutive splice sites and those involved in alternate splicing. The Sakai [43] paper on searching for these discriminative features in alternatively spliced exons can be used to train machine learning models to recognize different splice sites.

It is just as important to be able to determine the correct combination of exons and alternatively spliced exons that are used to produce a protein. It is the goal of my project to create a tool that can be used to predict useful alternative splice variants in the human genome.

Materials and Methods

Procedures for Alternative Splice Site Prediction

Due to the existence of exon splice site enhancers (ESE) and exon splice site silencers (ESS) around alternative splice site junctions, there should be unique motifs on the DNA strand that signals the location for them to bind to. These motifs near the splice site junctions coupled with weak splice junctions [47] found in alternative splice sites would give enough data differences to classify alternative splice sites from constitutive and consensus splice sites.

Based on this theory I could train a SVM to separate a given length of DNA sequence into one of four classifications. These four classifications could be a constitutive splice site, the upstream alternative splice site, the downstream alternative splice site, or a non-splicing site. The upstream alternative splice site is an alternative splice site that is spliced upstream, or the left most splice site, given the DNA strand is oriented from the 5' to 3' end. The downstream alternative splice site is the alternative splice site that is spliced downstream, or the right most splice site, given the DNA strand is oriented from the 5' to 3' end. I selected competing pairs of upstream and downstream splice sites by the number of Expressed Sequence Tags (EST) and mRNA that supported the two competing splice sites. I used the condition that at least one of the competing splice sites had to have 10 or more ESTs support and the other needed to have 3 or more EST support.

I trained three separate SVM's to classify alternative splice sites. The three alternative splice sites that the SVM's were trained to classify were the alternative 5' splice site, the alternative 3' splice site, and exon skipping. The classes in the exon skipping model were a little different than those in the alternative 5' and 3' models. Instead of classifying an upstream and downstream alternative site, it classified a major and minor exon. A major or silenced exon was one that was included in mature mRNA a majority of the time, but was skipped under certain cell conditions. A minor or cryptic exon was one that was excluded in the mature mRNA a majority of the time, but under certain cell conditions it was included in the mature mRNA. As above the two classifications of exons were supported by the EST and mRNA data in the database.

Another important distinction I used in selecting data for the major and minor exons in the exon skipping model, was that only sequences where only one exon was skipped were considered. The number of occurrences where multiple exons were skipped was rare. I felt that trying to differentiate between a single skip and a multiple skip would complicate the prediction, and was beyond the scope of what was trying to accomplish. In the case where multiple exons were skipped, the range of consecutive exons excluded was from two to nineteen exons.

Model Selection

I chose to use the Radial Basis Function (RBF) kernel over the polynomial and sigmoid kernels. The accuracy results were very close for all three kernels, but the RBF kernel was recommended by C. Hsu [32] for the following reasons. First, I did not consider the linear kernel, since it is not considered for use in pattern recognition [12]. Unlike the linear kernel, the RBF kernel can nonlinearly map samples into a higher dimensional space, allowing it to handle the case when the relation between class labels and attributes is nonlinear. The sigmoid kernel behaves like the RBF for certain parameters. The second reason for choosing the RBF is the number of hyperparameters, which influences the complexity of model selection. The polynomial kernel has more hyperparameters than the RBF kernel. Finally, the RBF kernel has less numerical difficulties. One key difference is that in the RBF kernel, the kernel values are always in the range, $0 < K_{ij} \leq 1$, but in the polynomial kernels, the kernel values may go to infinity ($x_i^T x_j + r > 1$) or zero ($x_i^T x_j + r < 1$) when the degree is large. There exists problems with the sigmoid kernel in which some parameters can cause the kernel values in be invalid. In other word the kernel value is not the inner product of two vectors [32].

I took the nucleotide regions with the highest accuracy scores from each model that was obtained with the RBF kernel using the default settings. I then trained and tested these samples with the polynomial and sigmoid kernels for the three SVMs. Figure 11 shows how similar the accuracies are between the three kernel functions for a given nucleotide range. The decision to use the RBF kernel came down to the arguments presented in [32].

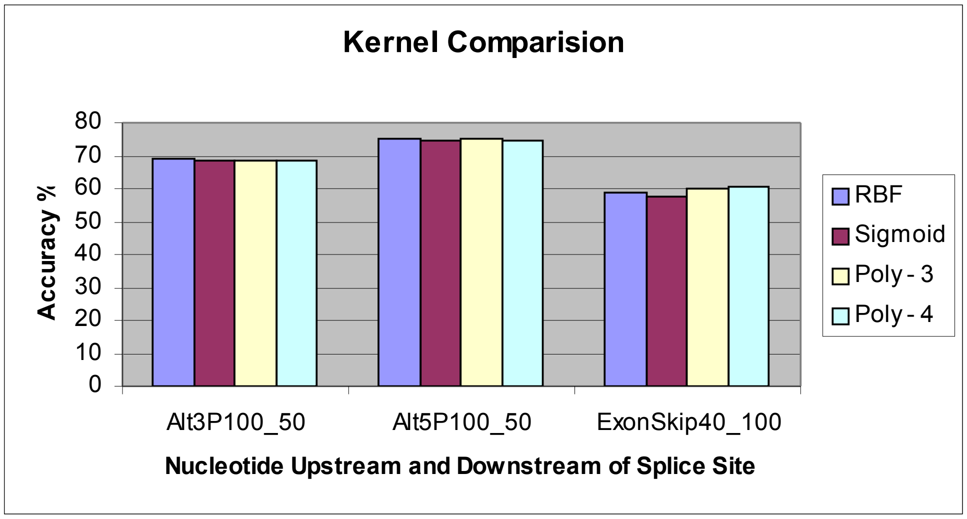


Figure 11. *Kernel Comparison*

Dataset

Human sequences of alternative splice sites were extracted from the Alternative Splice Annotation Project II (ASAP II) database [57], publicly available at "http://www.bioinformatics.ucla.edu/ASAP2". Sequences for alternative 5' splice sites, alternative 3' splice sites, exon skipping, and constitutive splice sites were extracted from the database along with the number of ESTs that supported the splice sites. The splice sites for each of the alternative forms were separated into two classes, the consensus class and the alternative class. The consensus class was the class for the alternative splice site that occurred a majority of the time based on more support from the ESTs. The alternative class was the class for the alternative splice site that occurred a minority of the time based on support from the number of ESTs for that splice site. The same procedure was used for exon skipping. The majority for an exon that was not spliced out a majority of the time and remained in the mRNA after splicing was completed. The minority exon for an exon that was spliced out a majority of the time and was removed in the mRNA after splicing was completed.

A total of 1,238 alternative 5' upstream and downstream splice sites were extracted from the ASAP II database for use. A total of 984 alternative 3' upstream and downstream splice sites were extracted from the database for use. A total of 5,257 consensus skipped exons and 4,075 cryptic exons were extracted for use. Constitutive splice sites were also extracted from the database, for both the 5' and 3' splice sites. For the constitutive 5' splice site, a total of 6,104 splice sites for the U1/U2 snRNA spliceosome binding was extracted. For the constitutive 3' splice site, a total of 6,124 splice sites for U1/U2 snRNA spliceosome binding was extracted. For the constitutive 5' and 3' splice site with the U11/U12 snRNA spliceosome binding, a total of 515 splice sites for each were extracted from the database.

	Upstream	Downstream	Constitutive U1/U2	Constitutive U11/U12	Non-Splicing
5' SS					
Training set	564	554	914	24	954
Test set	674	684	921	25	954
3' SS					
Training set	444	432	920	25	959
Test set	540	552	928	25	955
Exon Skipping	Major	Minor			
Training set	911	723	955 - 24	24	951
Test set	908	712	945 – 24	24	954

Table 2. *Training and Testing Set Distributions*

Non-splicing data was extracted from the database by taking random subsequences of length of 200 nucleotides in the intron and exon regions that did not span any part of a known splice site. I compiled a total of 11,827 non-splicing subsequences.

The training and testing sets for the constitutive, exon skipping, and non-splicing sets were randomly selected from the pool of extracted data corresponding to their set. The training and testing sets for the alternative 3' and 5' upstream and downstream splice site sets, were randomly divided into two somewhat equal sets. Table 2 shows the numbers of each feature used in the training and testing sets.

Feature Selection

Nucleotide composition is the basic feature of the splice site sequence. Each of the four nucleotides adenine (A), cytosine (C), guanine (G), and thymine (T) can be represented as a 4-bit string code. I used the following 4 bit codes to represent each of the four nucleotides, A-1000, C- 0100, G-0010, and T-0001. It is also common for an N or X to be used to represent an unknown nucleotide. I used the 4-bit string code of N/X-0000 to handle this case, however I did not include this case in my training and testing sets. This has the effect of scaling each feature of the input vector to the range of [0,1].

For each case, alternative 5' splice sites, alternative 3' splice sites, and exon skipping, I analyzed the region of 30 to 100 nucleotides upstream and downstream of the splice site in 10 nucleotide increments. Therefore, there were 64 different length combinations for each of the three cases. This set of possible vectors contained 16 different dimensional vectors. The overall length range is from 60 to 200 nucleotides by 10 nucleotide increments, so the dimension of each vector is 4L-D, where L is 60 to 200 by 10. The smallest dimensional vector is 4 x 60 = 240, and the largest is 4 x 200 = 800 dimensions.

A subset of each of the length combinations was trained and tested with default settings. The subsets were made up of upstream lengths of 30 to 100 nucleotides with a downstream length of 100 nucleotides, and upstream length of 100 nucleotides with downstream lengths of 30 to 100 nucleotides. The subset size was 15 as opposed to 64. The subset case with the highest accuracy was then run through the grid.py tool that came with the LIBSVM package [59]. The grid.py tool is an application that performs an exhaustive grid search on a set of penalty parameters, *C*, and gamma values. It is used to find the best penalty parameter, *C*, and gamma values. The best *C* and gamma values for the alternative 3' SVM was $C = 2.0$ and $\gamma = 0.0078125$. The best *C* and gamma values for the alternative 5' SVM was $C = 1.414213$ and $\gamma = 0.015625$. The best *C* and gamma values for the exon skipping were $C = 8$ and $\gamma = 0.0078125$. These new values improved the accuracy of all three SVMs over the default values. A more extensive description of the grid search process can is covered in the Performance Assessment section later in this chapter.

I then used these new values to run all 64 different length combinations for each of the three SVM cases. Figure 12 shows the accuracy for each of the three SVMs against the upstream and downstream nucleotide ranges.

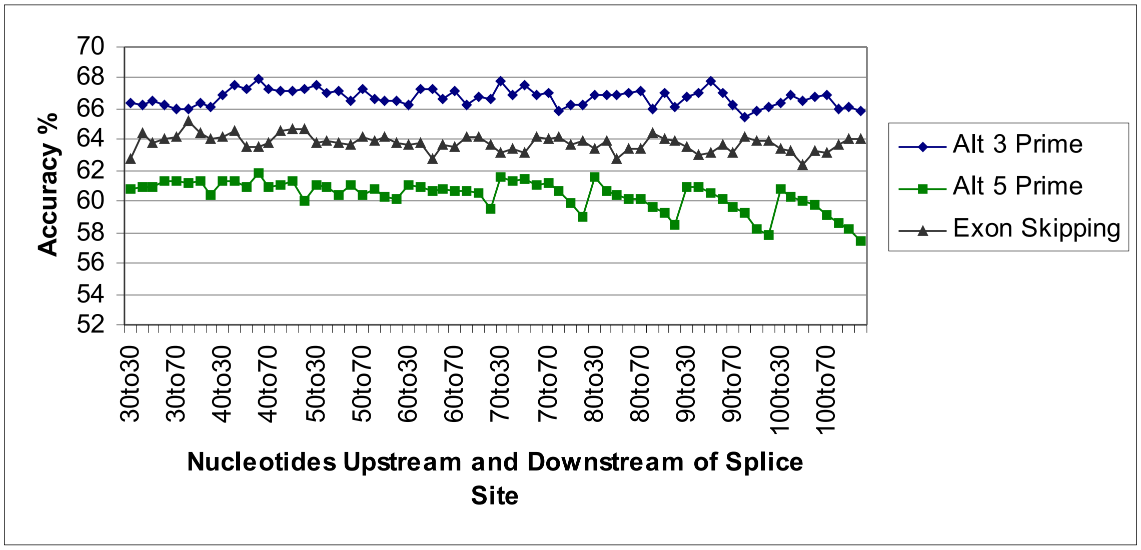


Figure 12. *Accuracy of Nucleotide Ranges for the Three Models*

It can be seen that the Alternative 3' SVM had the best accuracy followed by the exon skipping SVM and than the 5' SVM. The accuracy of the alternative 3' SVM was in range of 65% to 68%. The accuracy of the alternative 5' SVM was between 59% and 62% in range, and the exon skipping was in the range of 62% to 66%.

Figures 13, 14, and 15 show the accuracy of each of the individual SVM's accuracy against nucleotide distribution. I then took the five highest accuracy scores across the nucleotide distributions from each of the three SVM models. These five highest distribution scores were then compared against the sensitivity for each class. The nucleotide distributions with the five highest accuracy scores for the alternative 3' model were: 40 upstream to 60 downstream, 70 upstream to 30 downstream, 90 upstream to 50 downstream, 40 upstream to 40 downstream, and 50 upstream to 30 downstream of the 3' splice site. The nucleotide distributions with the five highest accuracy scores for the alternative 5' model were: 40 upstream to 60 downstream, 70 upstream to 30 downstream, 80 upstream to 30 downstream, 70 upstream to 50 downstream, and 70 upstream to 40 downstream of the 5' splice site. The nucleotide distributions with the five highest accuracy scores for the exon skipping model were: 30 upstream to 80 downstream, 40 upstream to 100 downstream, 40 upstream to 90 downstream, 40 upstream to 40 downstream, 40 upstream to 80 downstream of the 3' splice site for the skipped exon.

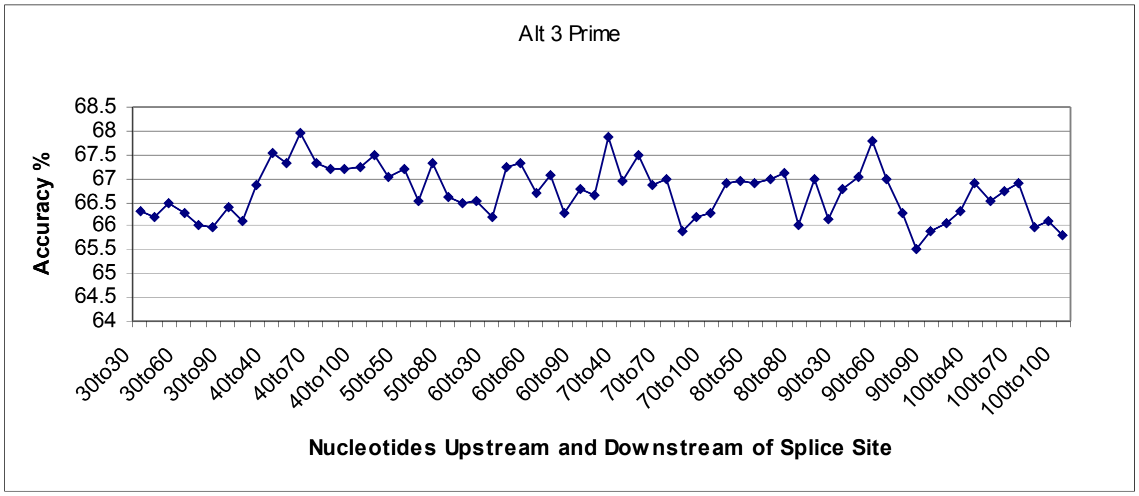


Figure 13. *Alternative 3' Model Accuracy Across Nucleotide Distributions*

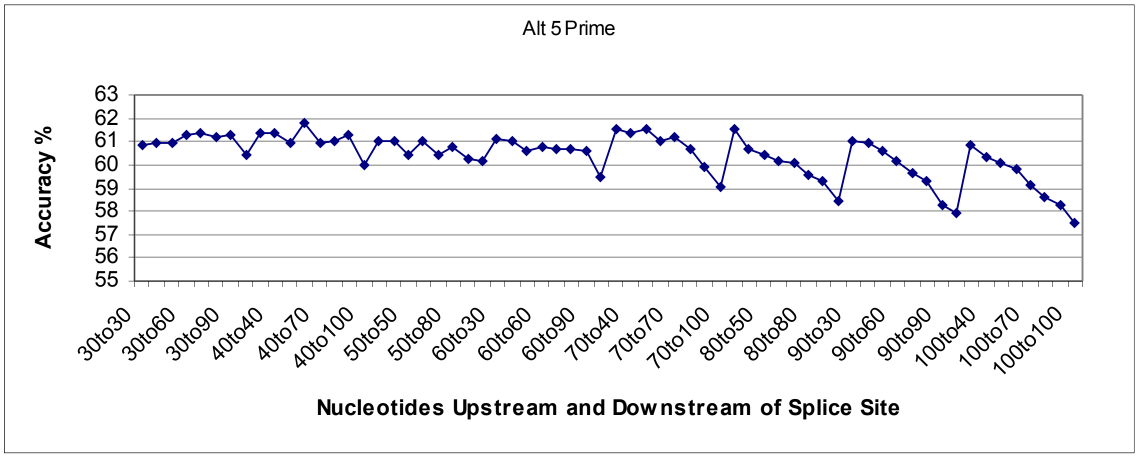


Figure 14. *Alternative 5' Model Accuracy Across Nucleotide Distributions*

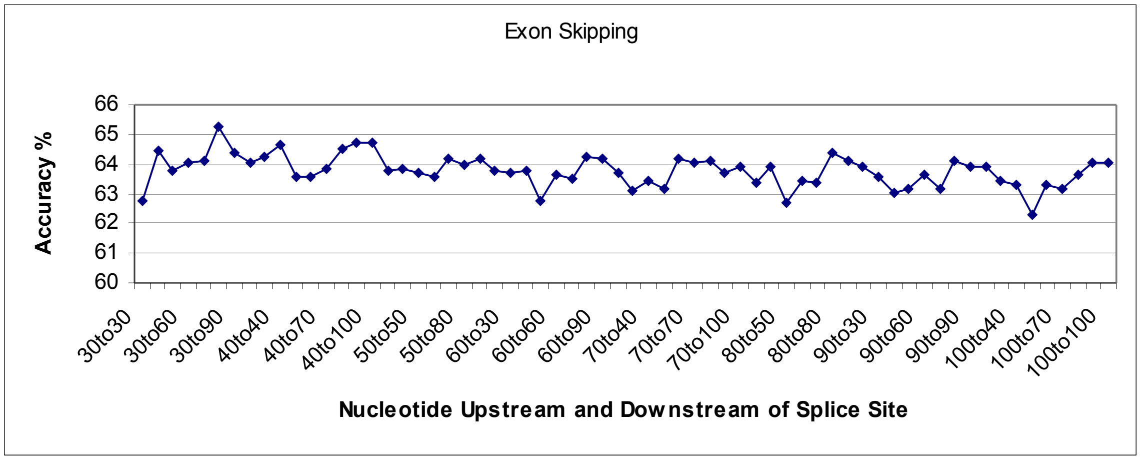


Figure 15. *Exon Skipping Model Accuracy Across Nucleotide Distributions*

Feature Selection for Alternative 3' Splice Site Model

I compared the sensitivity scores for each of the four classes in the alternative 3' model against the nucleotide distributions with the five highest accuracy scores for the alternative 3' model. These nucleotide distributions are described in the previous section. The four classes in the alternative 3' model are the upstream alternative 3' splice site, the downstream alternative 3' splice site, the constitutive 3' splice site, and the non-splicing region respectively. See Figure 16.

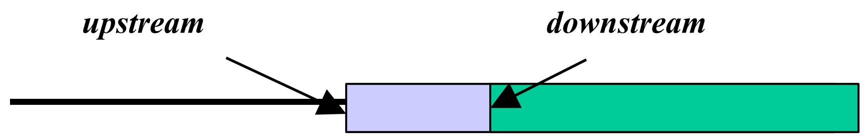


Figure 16. *Upstream and Downstream Positions for Alternative 3' Splice Site*

The sensitivity remained fairly constant across all classes over the nucleotide distribution. Classes 1 and 2, the upstream alternative 3' and downstream alternative 3' respectively, showed a slight improvement as the downstream length decreased, witnessed by the saw tooth form in the figure below. See Figure 17. From the comparison, I chose to use the nucleotide range of 40 upstream to 60 downstream of the 3' splice site. This range had the best model accuracy at 67.955% and good average balance of sensitivity scores for the four classes.

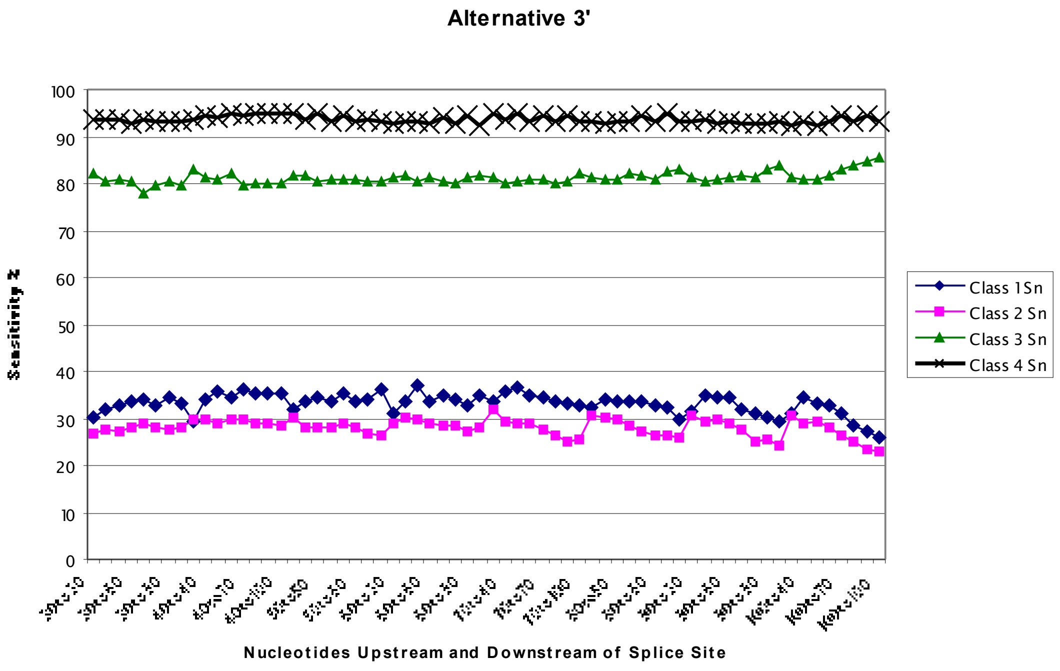


Figure 17. *Sensitivity Scores Across Nucleotide Distributions for Each Class in the Alternative 3' Model*

Feature Selection for Alternative 5' Splice Site Model

I compared the sensitivity scores for each of the four classes in the alternative 5' model against the nucleotide distributions with the five highest accuracy scores for the alternative 5' model. These nucleotide distributions are described in the previous section. The four classes in the alternative 5' model are the upstream alternative 5' splice site, the downstream alternative 5' splice site, the constitutive 5' splice site, and the non-splicing region respectively. See Figure 18.

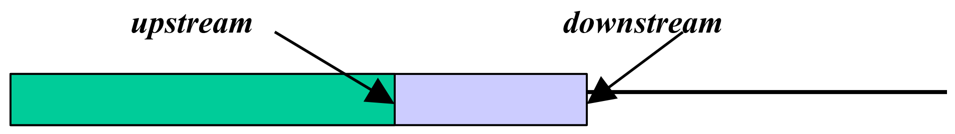


Figure 18. *Upstream and Downstream Positions for Alternative 5' Splice Site*

The sensitivity remained fairly constant across all classes over the nucleotide distribution. Class 3, the constitutive class, showed a slight improvement as the downstream length increased. However classes 1 and 2, the upstream alternative 3' and downstream alternative 3' respectively, showed a slight improvement as the downstream length decreased, witnessed by the saw tooth form in Figure 19.

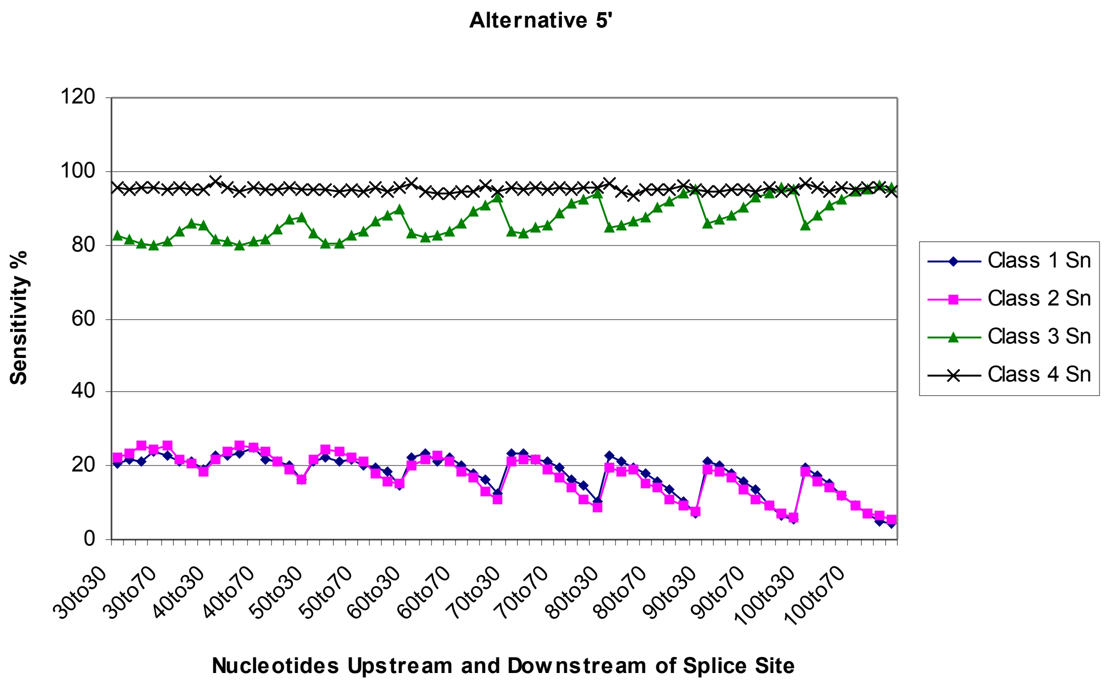


Figure 19. *Sensitivity Scores Across Nucleotide Distributions for Each Class in the Alternative 5' Model*

From the comparison, I chose to use the nucleotide range of 40 upstream to 60 downstream of the 5' splice site. This range had the best model accuracy at 61.817% and good average balance of sensitivity scores for the four classes.

Feature Selection for Exon Skipping Model

I compared the sensitivity scores for each of the four classes in the exon skipping model against the nucleotide distributions with the five highest accuracy scores for the exon skipping model. These nucleotide distributions are described in the previous section. The four classes in the exon skipping model are the 3' splice site for the major (skipped exon) exon, the 3' splice site for minor (cryptic) exon, the constitutive 3' splice site, and the non-splicing region respectively. See Figure 20.

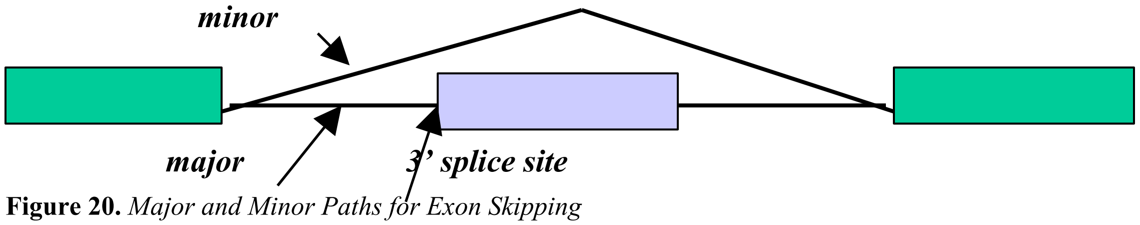


Figure 20. *Major and Minor Paths for Exon Skipping*

The sensitivity remained fairly constant for classes 1, 2, and 4 over the nucleotide distribution. Class 3, the constitutive class, showed a slight improvement as the

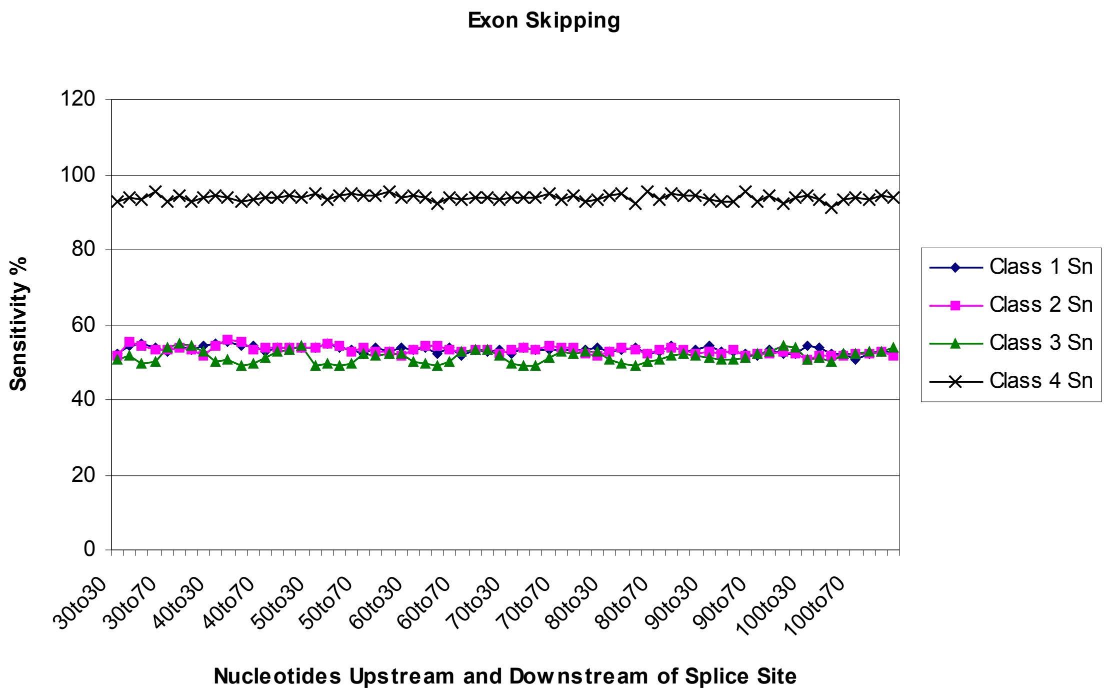


Figure 21. *Sensitivity Scores Across Nucleotide Distributions for Each Class in the Exon Skipping Model*

downstream length increased, witnessed by the slight waveform seen in the Figure 21. From the comparison, I chose to use the nucleotide range of 30 upstream to 80 downstream of the 3' splice site for the skipped exon. This range had the best model accuracy at 65.246% and the sensitivity scores for the three 3' splice site classes, classes 1, 2, and 3, were between the range of 53.5% and 55%. The sensitivity for the non-splicing class was 94.44%.

Performance Assessment

I used sensitivity (S_n), specificity (S_p), and total accuracy (TA) to evaluate the performance of the algorithms for each of the three models. I analyzed these three parameters on each class in each of the three models to see how well each model separated each class. I used True Positive (TP) and False Negative (FN) to denote the numbers of positive data for each class. Then S_n and S_p were defined as [53]:

$$S_n = \frac{TP}{TP+FN} \times 100\%$$

and

$$S_p = \frac{TN}{TN+FP} \times 100\%$$

S_n and S_p are the proportion of positive and negative data that is correctly predicted. TA was defined as [53]:

$$TA = \frac{TP+TN}{TP+FN+TN+FP} \times 100\%$$

As described in the Feature Selection main section earlier in this chapter, I described the use of the grid.py tool. The tool used five fold cross-validation on *C* values 2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, …, 2^{15}, and gamma values 2^{-15}, 2^{-13}, 2^{-11},…, 2^{3}. The grid search done by the grid.py tool tested pairs of (C, γ) until the one with the best five fold cross-validation was found [32]. This method was computationally expensive, but not much more than advanced methods since there are only two parameters [32]. Basically the grid-search exhaustively searches for the best perpendicular hyper-plane, which separates two classes. The *C* and gamma ranges described above were used as a course grid search for the alternative 3' and 5' models. The best *C* and γ pair for the alternative 3' model was $C = 2^1 = 2.0$ and $\gamma = 2^{-7} = 0.0078125$, and the best *C* and γ pair for the alternative 5' model was $C = 2^1 = 2.0$ and $\gamma = 2^{-5} = 0.03125$, for the course grid search. I then performed a finer grid search around these best pairs for each model. For the alternative 3' model the range of the *C* values were 2^0, $2^{0.25}$, $2^{0.5}$, $2^{0.75}$, …, 2^2, and gamma values 2^{-8}, $2^{-7.75}$, $2^{-7.5}$,…, 2^{-6}. The best C and γ pair for the finer grid search on the alternative 3' model was $C = 2^1 = 2.0$ and $\gamma = 2^{-7} = 0.0078125$. So, there was no improvement from the finer grid search. The best five-fold cross validation accuracy was 68.5025%.

The finer grid search for the alternative 5' model had the range of the C values 2^0, $2^{0.25}$, $2^{0.5}$, $2^{0.75}$, …, 2^2, and gamma values 2^{-6}, $2^{-5.75}$, $2^{-5.5}$,…, 2^{-4}. The best C and γ pair for the finer grid search on the alternative 5' model was $C = 2^{0.5} = 1.414213$ and $\gamma = 2^{-6} = 0.015625$. So, there was a slight improvement from the finer grid search. The best five-fold cross-validation accuracy improved from 62.0804 to 62.6454%.

I only performed the course grid search on the exon skipping model. The best C and γ pair for the exon skipping model was $C = 2^3 = 8.0$ and $\gamma = 2^{-7} = 0.0078125$.

In my case, where my SVM models have four classes, there are six perpendicular hyper-planes, one hyper-plane for each combination of classes. So, there is a perpendicular hyper-plane separating Class 1 and Class 2, another separating Class 1 and Class 3, etc. The figures below show how finding the best penalty parameter, C, and gamma values can find a better classifier, thus reducing the chances of having an overfitting classifier and improving the accuracy of the SVM model. See Figure 22.

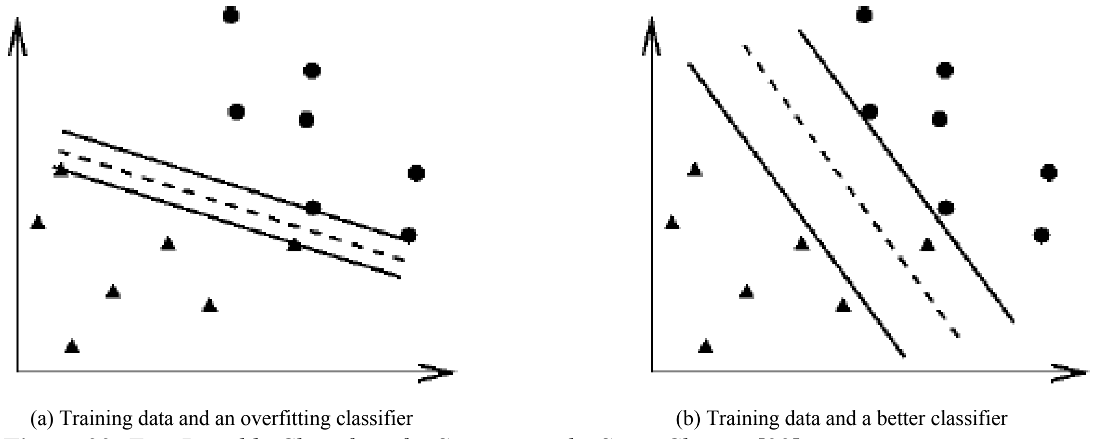

(a) Training data and an overfitting classifier
(b) Training data and a better classifier

Figure 22. *Two Possible Classifiers for Separating the Same Classes.* [32]

Table 3 shows the performance of the three SVM classifiers with the highest accuracy scores, using the best C and γ pairs from the grid search.

	S_n (%)	S_p (%)	TA (%)
Alternative 3' model			
Class 1 (upstream SS)	34.381	94.271	83.478
Class 2 (downstream SS)	29.837	95.263	83.211
Class 3 (constitutive 3')	82.162	67.789	72.352
Class 4 (non-splicing)	94.869	97.802	96.869
Alternative 5' model			
Class 1 (upstream SS)	24.777	93.228	79.067
Class 2 (downstream SS)	24.818	92.421	78.207
Class 3 (constitutive 5')	80.741	63.424	68.447
Class 4 (non-splicing)	95.807	98.785	97.913
Exon Skipping model			
Class 1 (major exon)	54.405	80.428	73.714
Class 2 (minor exon)	53.651	91.663	83.972
Class 3 (constitutive 3')	54.920	82.595	75.163
Class 4 (non-splicing)	94.444	98.830	97.641

Table 3. *Sensitivity, Specificity, and Total Accuracy Scores for Each Class in the Three Models.*

I plotted the results of sensitivity (true positive rate) vs. the False Positive rate (1 – specificity). This type of graph is called a Receiver Operating Characteristic curve (ROC curve), and is used to show the cutoff between sensitivity and specificity. [58]

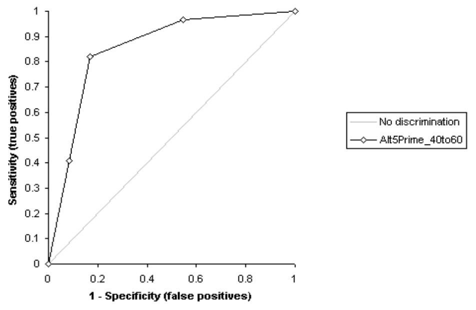


Figure 23. *ROC Curve for Alternative 5' Model*

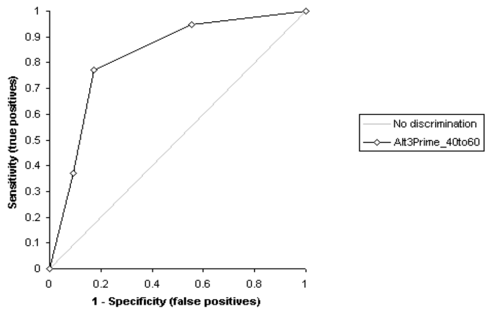


Figure 24. *ROC Curve for Alternative 3' Model*

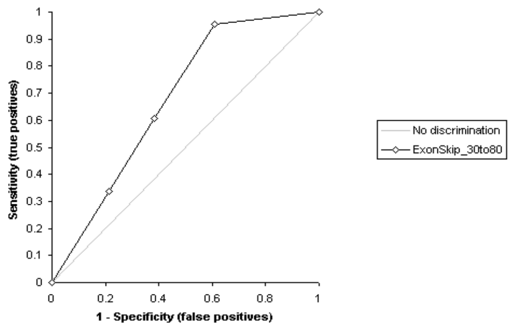


Figure 25. *ROC Curve for Exon Skipping Model*

Requirements

Project Scope

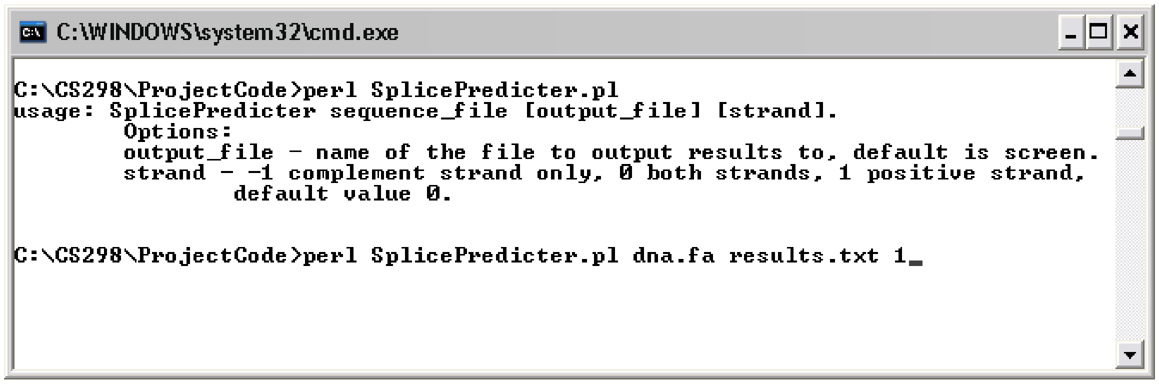
The scope of this project was to create software program that is able to predict the following types of splice sties in human genes: constitutive 5', constitutive 3', alternative 3' upstream, alternative 3' downstream, alternative 5' upstream, alternative 5' downstream, 3' splice site of a skipped exon, and 3' splice site of a cryptic exon. An important part of the project was the analysis of the training and testing data, in order to create the most accurate predictions possible.

The project did not include all types of alternative splice types. I excluded mutually exclusive exons and intron retention. I also excluded rare occurrences, such as multiple skipped exons. I only included instances in the training and testing data where a single exon was skipped, even though there were a few cases in which two or more exons were skipped. I did not include multiple skipping because I felt that the computational expense of comparing all neighboring exons could not be justified.

The aim of this program was to predict the above mentioned splice sites, so that it would give a biologist a way to examine a previously predicted gene or known gene, for possible alternative forms, that are not known and may warrant further investigation.

User Interface

The user interface is through the command line. There are several parameters that the program will accept from the command line. The first parameter that the program accepts is a required parameter, the input file name containing the DNA sequence to be run through the program. The second and third parameters are optional. The second parameter is the name of the output file and the third parameter is a flag indicating if you want the program to search the reverse strand of the DNA sequence for results.



```
C:\CS298\ProjectCode>perl SplicePredicter.pl
usage: SplicePredicter sequence_file [output_file] [strand].
        Options:
        output_file - name of the file to output results to, default is screen.
        strand - -1 complement strand only, 0 both strands, 1 positive strand,
                default value 0.


C:\CS298\ProjectCode>perl SplicePredicter.pl dna.fa results.txt 1_
```

Figure 26. *Command Line User Interface.*

Design

SVM Package and Cross-Validation

I used the LIBSVM, Library for Support Vector Machines, SVM package, version 2.81, written by Chih-Chung Chang and Chih-Jen Lin [59]. The latest version of the software is publicly available and can be downloaded at "http://www.csie.ntu.tw/~cjlin/libsvm". I decided to write my project in Perl, and translated the Java implementation into the Perl language. I tested several sample files for training and testing using different parameters to make sure that my Perl implementation gave the same results as the Java version that came with the package.

I also used the grid search tool, Grid.py, which came with the package to perform a course and fine grid search on my input data. The grid search tool is a command line Python script to exhaustively search for the best C and γ parameters for an SVM model using five-fold cross-validation.

Architecture

I used Perl as my programming language for this project. I used a combination of object-oriented Perl modules, BioPerl modules [60], and the Perl data types of scalars, arrays, and hashes to store and process the data.

I wrote several Perl scripts for parsing the data that was extracted from the ASAP II database, into different training and testing files that were formatted to the SVM input format. Some other scripts were written to parse the SVM output data to into statistical results. Most of these parser scripts followed the functional programming methodologies, however the script that was used to format the training and testing input files used a multiple paradigm of object-oriented and functional methodologies. I created a Perl module, or object, that inherited the BioPerl sequence object. This object was designed to hold a DNA sequence object, the function of the BioPerl sequence object, along with the additional functions for processing the DNA sequence into a binary representation and also into the SVM input format. I was able to use this object in my main program also.

The main program used multiple paradigm design. It incorporated object-oriented design and functional programming with sub-procedures. Objects were used to represent the DNA input sequence, and many objects were used in the SVM implantation.

Implementation

I decided to use Perl as the programming language for this program since it is a commonly used language in Bioinformatics. The first step was to convert the LIBSVMs

Java implementation code into Perl. This presented some challenges, since the some of the functions in Java are not available in Perl. Java is a pure object-oriented language while Perl is more of a functional language with the ability to build objects. Another challenge was that Perl's only data types are scalars, arrays, and hashes. A scalar can be a string, integer, double, or a reference to an object, array, or hash. The good thing about a scalar is that it very flexible for the data type that can be stored in it. The bad thing about a scalar is that, because it is very flexible for the data type that can be stored in it, every time it has to do a mathematical operation it has to make sure the data type is a number, which is very costly for running a program with a lot of mathematical operations. Java and other high level languages declare the data type, so they do not have to perform the operation of checking the data type. I believe this caused the performance of the Perl implementation to be well below that of the Java implementation.

The next step was to create an interface with the SVM. The process for doing this was to read the DNA input sequence from the user, and loading it into two sequence objects, one for the positive strand and one for the complement strand. One of the functions of the sequence object was a method to return the SVM input format for a subsequence of the DNA sequence. I used this method to create a temporary file of input vectors. Each input vector is a subsequence of the DNA sequence whose length is the same as the nucleotide length that the SVM model was built on. For example, I chose the upstream and downstream region of the alternative 3' model as 40 and 60 nucleotides respectively. This means that the subsequence length will be 100 nucleotides. The number of input vectors is then, DNA sequence length minus subsequence length. There is one temporary file for each model.

The temporary file of input vectors is then passed to the predict method of the SVM for its corresponding SVM model. The predicted class for each input vector was then stored in an array, one corresponding to each model. I used the hash data type to store the predicted splice sites. The key for the hash was the predicted position and its value was the predicted splice type, such as "Alternative 3' upstream". I did not load any of the non-splicing prediction sites into the hash table. I also created Position Specific Scoring Matrices (PSSM), based on the for each prediction type. These PSSMs were used to filter any SVM predicted splice sites that did not conform to any known splice site sequences fro the predicted splice site type. The PSSMs were based on frequencies for 9-mer sequences around the 5' splice site, and an 8-mer sequence around the 3' splice site. I compiled these frequencies from all the extracted data that the training and testing files were created from.

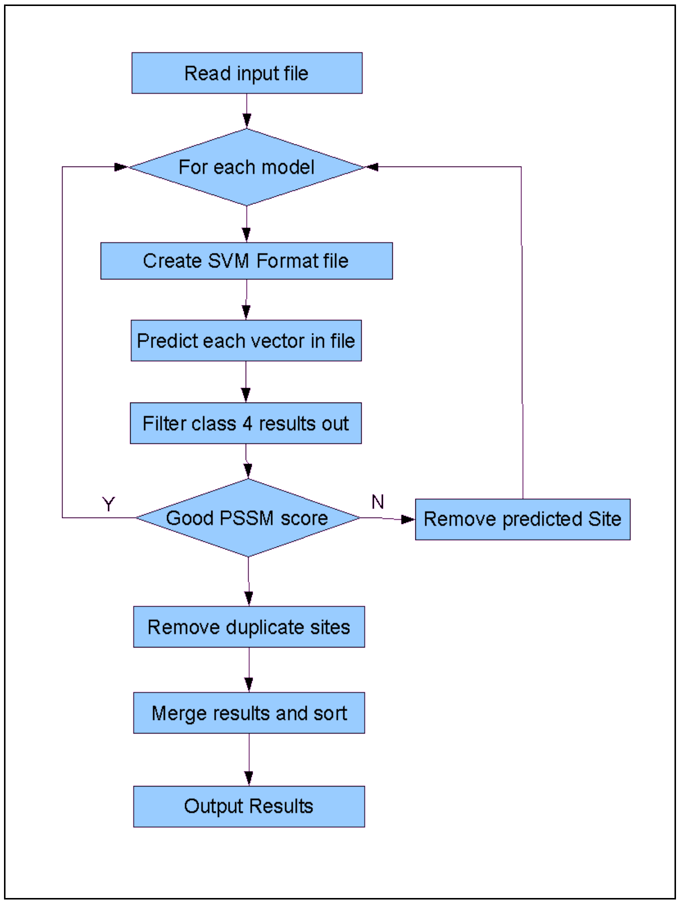


Figure 27. *Flow Chart for Splice Site Prediction Program*

Next I had to do was check to see if any of the three models predicted the same position as splice site. If there was a conflict at a position, I then kept the one whose PSSM score was higher then the others, and deleted the lower scoring prediction. The PSSM score was based on the sum of the weights for each position, corresponding to the weight of the nucleotide at a specific position, divided by the highest score that could be achieved for the PSSM. The score is a percentage of how close the sequence is to the consensus sequence for that particular PSSM.

A total of eight PSSMs were constructed for the different types of splice sites the SVMs were predicting. Figures 28 through 35 show the nucleotide distributions of each nucleotide flanking the splice sites for the PSSMs. A Perl script was used to create the distributions of each splice site from the data extracted from the ASAP II database. The script also kept track of the minimum score of a good splice site sequence for each splice site type. This minimum score was used as the minimum cute off score for the PSSM.

		Exon			Intron					
		-3	-2	-1	+1	+2	+3	+4	+5	+6
Constitutive 5'	**A**	2171	4149	707	515	0	3650	4324	675	1186
Splice Site	**C**	2332	723	422	0	0	264	574	517	1140
	G	1243	790	5013	6103	0	2390	846	4865	1353
	T	872	956	476	0	6618	314	874	561	2939

Figure 28. *Constitutive 5' Splice Site Nucleotide Distribution*

		Exon			Intron					
		-3	-2	-1	+1	+2	+3	+4	+5	+6
Alternative 5'	**A**	391	717	143	0	0	534	679	154	208
Upstream Alternative Site	**C**	455	152	44	0	0	94	204	127	260
	G	256	174	942	1238	0	532	180	806	317
	T	136	195	109	0	1238	78	175	151	453

Figure 29. *Alternative 5' Upstream Splice Site Nucleotide Distribution*

		Exon			Intron					
		-3	-2	-1	+1	+2	+3	+4	+5	+6
Alternative 5'	**A**	331	732	99	0	0	612	655	146	208
Downstream Alternative	**C**	463	144	45	0	0	77	147	107	215
Site	**G**	267	194	989	1238	0	476	241	847	305
	T	177	168	103	0	1238	73	195	138	510

Figure 30. *Alternative 5' Downstream Splice Site Nucleotide Distribution*

		Intron							Exon
		-7	-6	-5	-4	-3	-2	-1	+1
Constitutive 3'	**A**	807	672	639	1528	438	6639	0	1850
Splice Site	**C**	2341	2473	2138	1967	4269	0	515	1008
	G	684	531	505	1342	126	0	6124	2994
	T	2807	2963	3357	802	1806	0	0	787

Figure 31. *Constitutive 3' Splice Site Nucleotide Distribution*

		Intron							Exon
		-7	-6	-5	-4	-3	-2	-1	+1
Alternative 3'	**A**	130	131	123	230	78	984	0	111
Downstream Alternative	**C**	358	398	335	353	738	0	2	122
Site	**G**	126	104	114	220	14	0	982	672
	T	370	351	412	181	154	0	0	79

Figure 32. *Alternative 3' Downstream Splice Site Nucleotide Distribution*

		Intron							Exon
		-7	-6	-5	-4	-3	-2	-1	+1
Alternative 3'	**A**	111	94	100	221	77	984	0	276
Upstream Alternative	**C**	358	398	335	309	620	0	2	263
Site	**G**	124	104	103	205	26	0	982	197
	T	390	411	422	249	261	0	0	248

Figure 33. *Alternative 3' Upstream Splice Site Nucleotide Distribution*

		Intron							Exon
		-7	-6	-5	-4	-3	-2	-1	+1
Exon Skipping	**A**	500	622	530	1105	393	4075	0	1293
Minor Exon	**C**	922	954	964	1005	2194	0	0	597
	G	515	372	456	740	147	0	4075	1588
	T	2138	2127	2125	1225	1341	0	0	597

Figure 34. *Minor Exon Skipping 3' Splice Site Nucleotide Distribution*

		Intron							Exon
		-7	-6	-5	-4	-3	-2	-1	+1
Exon Skipping	**A**	592	455	453	1237	287	5257	0	1120
Major Exon	**C**	358	398	335	309	620	0	2	768
	G	124	104	103	205	26	0	5257	2655
	T	2447	2598	2851	1555	1573	0	0	714

Figure 35. Major Exon Skipping 3' Splice Site Nucleotide Distribution

The last thing that needed to be done was that the three hash tables were merged into a single hash table. This final hash table was then sorted by position in the input file, and sent to output. Figure 36 shows a sample output from the program.

The output displays the predicted splice site, the coding strand that it was found on, the relative position on the input sequence, and the nucleotide sequence around the splice site. The lower case letters in the nucleotide sequence represents the intronic region and the upper case letters represent the exonic region.

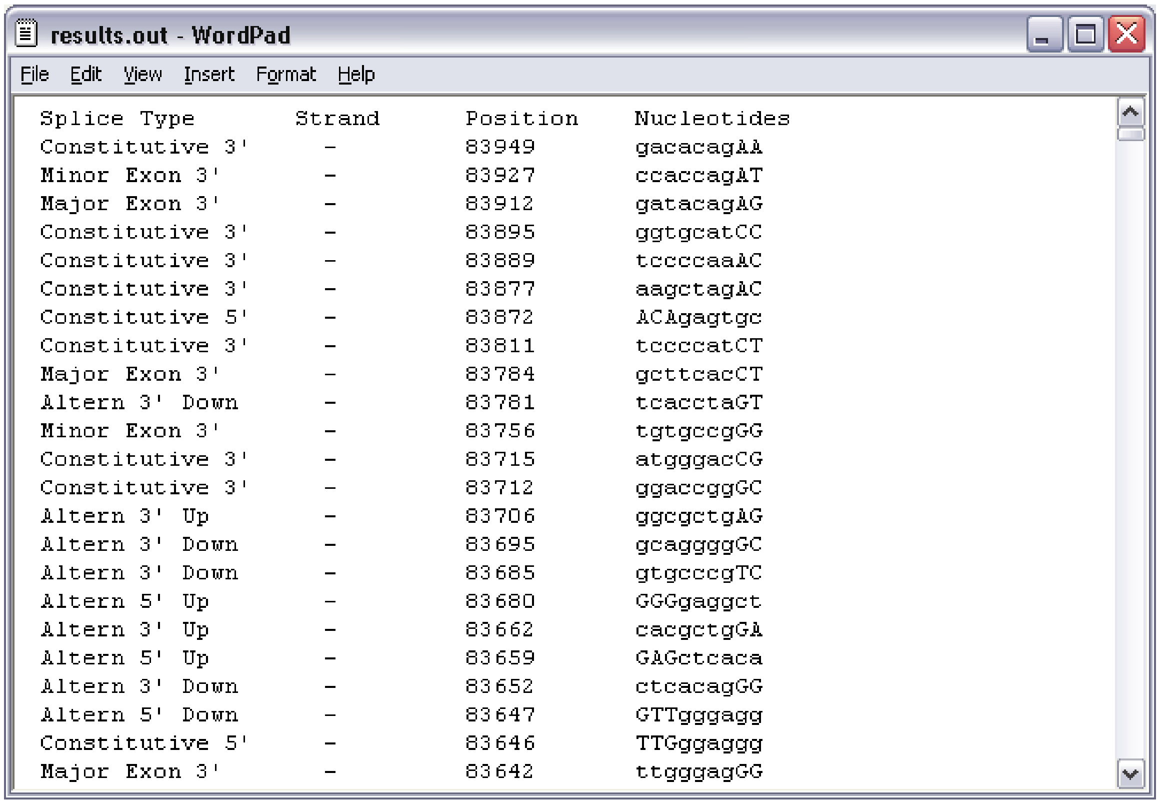
In the Results section I will discuss the results I obtained from the test sets and from a genomic sequence with known alternative splice sites.

Results

Prediction of Alternative Splice Sites

The prediction of alternative splice sites was performed on the testing set and also of the genomic sequence that codes the BRCA1 gene. For the genomic sequence of the BRCA1 gene I downloaded the complete genomic sequence containing the gene from the NCBI website. The size of the genomic sequence was 84,000 nucleotides in length. Figure 33 shows the partial output from the prediction program.

The prediction program predicted about 6900 false non-splicing sites out of about 83,850 actual non-splicing sites, so the non-splice accuracy was around 92%, which is a about 3% lower than the test set results.



```
Splice Type        Strand       Position     Nucleotides
Constitutive 3'      -          83949        gacacagAA
Minor Exon 3'        -          83927        ccaccagAT
Major Exon 3'        -          83912        gatacagAG
Constitutive 3'      -          83895        ggtgcatCC
Constitutive 3'      -          83889        tccccaaAC
Constitutive 3'      -          83877        aagctagAC
Constitutive 5'      -          83872        ACAgagtgc
Constitutive 3'      -          83811        tccccatCT
Major Exon 3'        -          83784        gcttcacCT
Altern 3' Down       -          83781        tcacctaGT
Minor Exon 3'        -          83756        tgtgccgGG
Constitutive 3'      -          83715        atgggacCG
Constitutive 3'      -          83712        ggaccggGC
Altern 3' Up         -          83706        ggcgctgAG
Altern 3' Down       -          83695        gcaggggGC
Altern 3' Down       -          83685        gtgcccgTC
Altern 5' Up         -          83680        GGGgaggct
Altern 3' Up         -          83662        cacgctgGA
Altern 5' Up         -          83659        GAGctcaca
Altern 3' Down       -          83652        ctcacagGG
Altern 5' Down       -          83647        GTTgggagg
Constitutive 5'      -          83646        TTGggaggg
Major Exon 3'        -          83642        ttgggagGG
```

Figure 36. *Sample Prediction Output*

The results from the testing sets are discussed in the next section, Comparison with Existing Methods.

Comparison with Existing Methods

There are several methods that have recently been published for predicting alternative splice site using SVMs. The method proposed by H. Xia [53] used the mechanism of splice site competition. This method differed in my method, in that they started with a known splice site, and then predicted whether a competitive splice site was within 200 nucleotides upstream or downstream of the known splice site. My method classifies all of the positions in the input sequence, so it is difficult to compare my results to their results. They published that their method could predict about 70% of alternative splice sites accurately with a false positive rate of 30% [53]. Another difference in my method was that it predicts the type of splice site. For example, it will predict whether a splice site is an alternative 3' splice, and if it is upstream or downstream in relation to another alternative 3' splice site. It will also predict if the splice site is constitutive.

My results can only be compared by calculating the sensitivity and false positive rate when the models predict either alternative splice version. For either alternative 3' upstream or downstream splice sites, the predicted accuracy is ~40% with a false positive rate of 9%. For either alternative 5' upstream or downstream splice sites, the predicted accuracy is ~38% with a false positive rate of 10%. For either major or minor exon skipping splice sites, the predicted accuracy is ~73% with a false positive rate of 23%. These results show that the constitutive site is more likely to be predicted over the alternative site. This could be caused by the similarity in the splice site sequences.

The results from my prediction models are quite a bit lower than that of the prediction program that uses the mechanism of competitive splicing, however their method needs to know one splice site in order to predict if there is an alternative splice site near by.

Conclusion/Discussion

I was able to create a more inclusive alternative splice prediction program at the expense of having lower prediction accuracy than other alternative prediction splice sites. My prediction method predicts alternative 3'and 5' splice sites, and 3' splice sites on exons that are involved in exon skipping.

Future Work

There are two other alternative splicing events that I did not include in this project, mutually exclusive exclusion and intron retention. Adding these two events would make this program an all-inclusive alternative splice site prediction tool. The accuracy of identifying the correct type of splice site needs to be improved. I would probably need to extend the nucleotide lengths around the splice sites to see it would create a better classification for the splice site vectors.

The decision to use Perl as the language caused the computational time to be about 30 times slower than a Java implementation for the SVM training and prediction functions. The Perl scripts work well for parsing data and analyzing data in output files. A language that is faster at performing mathematical operations needs to be implemented to make this program run faster. One way may be to integrate C code with Perl, or use an object-oriented language like Java or C#. The program could be optimized to take advantage of the multi-core processors in many of today's computers. This would be possible, since the prediction algorithm on the different SVM models can be independently from each other.

References:

[1] Alexandersson, Marina; Cawley, Simon; Patcher, Lior. "Applications of Generalized Pair Hidden Markov Models to Alignment and Gene Finding Problems". RECOMB 2001, Montreal, Canada. Copyright ACM 2001 1-58113-353-7.

[2] Baldi, P., Y. Chauvin, T. Hunkapillar, and M.A. McClure. "Hidden Markov Models of Biological Primary Sequence Information"**.** Proceedings National Academy of Science*,* Vol. 91, pp. 1059-1063, 1994.

[3] Bao, Jingyue; Du, Wei; Clark, Terry; Johnson, Deborah; Kim, Yeong C.; Lee, Sanggyu; Lin, Wei; Lu, Xuemei; Shapiro, Joshua; Sun, Min; Tseng, Charles; Wang, Jian; Wang, Jun; Wang, San Ming; Wing, Claudia; Wu, Chung-I; Xu, Jinhua; Yang, Huanming; Zhang, Xiuqing; Zhang Shi, Run; Zhou, Guolin. "Detecting Novel Low-Abundant Transcripts in Drosophila". RNA (2005), 11:939–946. Cold Spring Harbor Laboratory Press. Copyright 2005 RNA Society.

[4] Blanchette, Mathieu. "A Comparative Analysis Method for Detecting Binding Sites in Coding Regions". RECOMB '03, April 10-13, 2003, Berlin, Germany. Copyright 2003 ACM 1-58113-635-8.

[5] Bockhorst, Joseph; Craven Mark; Page David; Shavlik Jude; and Glasner Jeremy. "A Bayesian Network Approach to Operon Prediction". Bioinformatics, 19(10) Oxford University Press 2003; Vol. 19 no. 10 2003, pages 1227–1235

[6] Brent, Michael R.; Duan, Daniel; Flicek, Paul; and Korf, Ian. "Integrating Genomic Homology into Gene Structure Prediction". Bioinformatics, Vol. 1 Supplement 1, pp. S1 - S9, 2001.

[7] Brown, Terence A. "Genomes". Second Edition. John Wiley and Sons, 2002.

[8] Bulyk, Martha L.; Church, George M.; Johnson, Philip L. F. "Nucleotides of Transcription Factor Binding Sites Exert Interdependent Effects on the Binding Affinities of Transcription Factors". Nucleic Acids Research, 2002, Vol. 30, No. 5 pp. 1255-1261.

[9] Burge, Christopher. "Identification of Genes in Human Genomic DNA", Stanford University, March 1997. http://www.nslij-genetics.org/gene/burge.pdf.

[10] Burge, Christopher B.; Han, Kyoungha; Yeo, Gene; An, Ping; Grabowski, Paula. "A Combinatorial Code for Splicing Silencing: UAGG and GGGG Motifs". PLoS Biology, May 2005, Vol. 3, Issue 5 pp 843-860.

[11] Burge, Christopher; Yeo, Gene. "Maximum Entropy Modeling of Short Sequence Motifs with Applications to RNA Splicing Signals". RECOMB '03 April 10-13, 2003, Berlin, Germany. Copyright 2003 ACM 1-58113-635-8.

[12] Burges, Christopher. "A Tutorial on Support Vector Machines for Pattern Recognition". Data Mining Knowledge Discovery, pp. 121-167. Kluwer Acedemic Publishers, Boston, 1998.

[13] Cabello-Villegas, Javier; Giles, Keith E.; Soto, Ana Maria; Yu, Ping; Mougin, Annie; Beemon, Karen L.; Wang, Yun-Xing. "Solution Structure of the Pseudo-5' Splice Site of a Retroviral Splicing Suppressor". RNA (2004), 10:1388-1398. Copyright 2004 RNA Society.

[14] Castelo, Robert; Guigo, Roderic; "Splice Site Identification by idl*BN*s", Bioinformatics, Vol. 20 Suppl. 1, 2004, pg i69-i76.

[15] Chang, Hsun-Chang; Yu, Po-Shun; Huang, Tze-Wei; Lin, Yaw-Ling. "The Application of Alternative Splicing Graphs in Quantitive Analysis of Alternative Splicing Form from EST Database". Proceedings of the Forth IEEE Symposium on Bioinformatics and Boiengineering (BIBE '04) 0-7695-2173-8. Copyright IEEE 2004.

[16] Chang, Hwan-You; Hsu, F.R.; Lin, Yaw-Lin; Tsai, Yin-Te, Peng, Hui-Ling; Chen, Ying Tsong; Chen, Che Feng; Cheng, Chia Yang; Liu, Chia-Hung; Shih, Min Yao. "Genome-wide Alternative Splicing Events Detection Through Analysis of Large Scale ESTs". Proceedings of the Forth IEEE Symposium on Bioinformatics and Boiengineering (BIBE '04) 0-7695-2173-8. Copyright IEEE 2004.

[17] Chen, Pai-Hsuen; Fan, Rong-En; Lin, Chih-Jen. "Working Set Selection Using the Second Order Information for Training SVM". Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan.

[18] Chiara, Maria; Bennett, Maria; Champion-Arnuad, Patrick; Palandjian, Leon; Reed, Robin; "Identification of Proteins That Interact with Exon Sequences, Splice Sites, and Branchpoint Sequence During Each Stage of Spliceosome Assembly", Molecular and Cellular Biology, July 1996, Vol. 16, No. 7, pg. 3317-3326.

[19] Churbanov, Alexander; Rogozin, Igor; Deogun, Jitender; Ali Hesham; "Method of Predicting Splice Sites Based on Signal Interactions", Biology Direct, April 2006, http://www.biology-direct.com/content/1/1/10.

[20] Cohen, Jacques. "Bioinformatics – An Introduction for Computer Scientists". ACM Computing Surveys, Vol. 96, No. 2, June 2004, pp. 122-158.

[21] Delcher, Arthur L.; Gardner, Malcolm J.; Pertea, Mihaela; Salzberg, Steven L.; Tettelin, Herve. "Interpolated Markov Models for Eukaryotic Gene Finding". Genomics Vol. 59, pp 24-31. 1999.

[22] Dietrich, Rosemary C.; Peris, Seyboldt, Andrew S.; Padgett, Richard A. "Role of the 3' Splice Site in U12-Dependent Intron Splicing". Molecular and Cellular Biology, Mar. 2001, p. 1942-1952, 0270-73006.

[23] Dror, Gideon; Sorek, Rotem; Shamir, Ron; "Accurate Identification of Alternatively Splice Exons using Support Vector Machines". Bioinformatics, 2005, Vol. 21, No. 7, pg. 897-901.

[24] Frilander, Mikko J.; Stitz, Joan A. "Initial Recognition of U12-Dependent Introns Requires both U11/5'Splice-Site and U12/Branchpoint Interactions". Genes and Development 13:851-863, Copyright Cold Spring Harbor Laboratory Press.

[25] Frilander, Mikko J.; Meng, Xiaojuan. "Proximity of the U12 snRNA with both the 5' Splice Site and the Branch Point during Early Stages of Spliceosome Assembly". Molecular and Cellular Biology, June 2005, p. 4813-4825, 0270-7306.

[26] Green, Michael R.; Kan, Julie L.C. "Pre-mRNA Splicing of IgM Exons M1 and M2 is Directed by a Juxtaposed Splicing Enhancer and Inhibitor". Genes and Development 13:462-471, Copyright Cold Spring harbor Laboratory Press ISBN 0890-9369.

[27] Halloway, Dustin and DeLisi, Charles. "Predicting Transcription Factor Binding Sites Using a Bayesian Allocation Method". Department of Bioinformatics, Boston University, Boston, MA.

[28] Halloway, Dustin; Kon, Mark; and DeLisi, Charles. "Integrating Genomic Data to Predict Transcription Factor Binding". Molecular Biology Cell Biology and Biochemistry, Boston University, Boston, MA 02215, U.S.A

[29] Haussler, David. "A Brief Look at Some Machine Learning Problems in Genomics". COLT '97 Nashville, Tennessee. Copyright 1997 ACM 0-89791-891-6.

[30] Haussler, David; Wu, Jing. "Coding Exon Detection Using Comparative Sequences". Center for Biomolecular Science and Engineering, University of California, Santa Cruz. December 9, 2004.

[31] Ho, Loi Sy; Rajapakse, Jagath. "Markov Encoding for Detecting Signals in Genaomic Sequences". IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol. 2, No. 2, April-June 2005.

[32] Hsu, Chih-Wei; Chang, Chih-Chung; Lin, Chih-Jen; "A Practical Guide to Support Vector Classification", 2003, http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

[33] Huang, Haiyan; Speed, Terence; Zhao. "Finding Short DNA Motifs Using Permuted Markov Models". RECOMB '04, March 27-31, 2004, San Diego , California.

[34] Isarankura-Na-Ayudhya, Chartchalerm; Nantasenamat, Chanin; Naenna, Thanakorn; Prachayasittikul Virapong. "Recognition of DNA Splice Junction via Machine Learning Approaches"**.** EXCLI Journal 2005;4:114-129 – ISSN 1611-2156, October 2005.

[35] Karp, Gerald. "Cell and Molecular Biology, Concepts and Experiments". Third Edition. John Wiley and Sons, 2002.

[36] Keerthi, S. Sathiya; Lin, Chih-Jen; "Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel", Neural Computation, Vol 15, Issue 7, July 2003, pg 1667-1689.

[37] Krogh, A., M. Brown, I.S. Mian, K. Sjolander, and D. Haussler. "Hidden Markov Models in Computational Biology: Applications to Protein Modeling". Journal of Molecular Biology, Vol. 235, pp.1501-1531, 1994.

[38] Matter, Nathalie; Konig, Harald. "Targeted 'Knockdown' of Splicosome Function in Mammalian Cells". Nucleic Acids Research, 2005, Vol. 33, No. 4.

[39] Mount, David. "Bioinfromatics, Sequence and Genome Analysis". Cold Spring Harbor Press, 2001.

[40] Murphy, Kevin P. "Hidden Semi-Markov Models (HSMMs)". 20 November 2002.

[41] Norvig, Peter; Russell, Stuart. "Artificial Intelligence, A Modern Approach". Prentice Hall, Copyright 2003.

[42] Ohler, Uwe; Shomron, Noam; Burge, Christopher B. "Recognition of Unknown Conserved Alternatively Spliced Exons". PLoS Computational Biology, July 2005, Vol. 1, Issue 2, e15.

[43] Sakai, H.; Maruama, O. "Extensive Search for Discriminative Features of Alternative Splicing". *Pacific Symposium on Biocomputing* 9:54-65(2004).

[44] Schölkopf, Bernhard; Tsuda, Koji; Vert, Jean-Philippe; "Kernel Methods in Computational Biology", A Bradford Book, Copyright 2004 Massachusetts Institute of Technology.

[45] Schones, Dustin E.; Sumazin, Pavel; and Zhang, Michael Q. "Similarity of Position Frequency Matrices for Transcription Factor Binding Sites". *Bioinformatics,* Oxford University Press 2004; August 19, 2004.

[46] Sorek, Rotem; Shemesh, Ronen; Cohen, Yuval; Basechess, Ortal; Ast, Gil; Shamir, Ron; "A Non-EST-Based Method for Exon-Skipping Prediction", Genome Research, 2004, Vol. 14, pg 1617-1623.

[47] Thanaraj, T. A.; Clark, Francis; "Human GC-AG Alternative Intron Isoforms with Weak Donor Sites Show Enhanced Consensus at Acceptor Exon Positions", Nucleic Acids Research, 2001, Vol. 29, No. 12, pg 2581-2593.

[48] Theodoridis, Sergios; Koutroumbas, Konstantinos. "Pattern Recognition". Academic Press 1999.

[49] Vapnik, V. N. "The Nature of Statistical Learning Theory", Springer, New York. 1995.

[50] Yeo, Gene W.; Van Nostrand, Eric; Holste, Dirk; Poggio, Tomaso; Burge, Christopher B. "Identification and Analysis of Alternative Splicing Events Conserved in Human and Mouse". PNAS February 22, 2005, vol. 102, no. 8, pp 2850-2855.

[51] Yeo, Gene; Holste, Dirk; Kreiman, Gabriel; Burge, Christopher B. "Variation in Alternative Splicing Across Human Tissues". Genome Biology 2004, Vol. 5, Issue 10, Article R74.

[52] Yu, Yi-Tao; Steitz, Joan A. "Site-specific Crosslinking of Mammalian U11and U6 atac to the 5' Splice Site of an AT-AC Intron". Biochemistry, June 1997, vol. 94, pp 6030-6035, The National Academy of Science, 1997, 0027-8424.

[53] Xia, Huiyu; Bi, Jianning; Li, Yanda; "Identification of Alternative 5'/3' Splice Sites Based on the Mechanism of Splice Site Competition". Nucleic Acids Research, 2006, Vol. 34, No. 21, pp. 6305-6313.

[54] Zhang, Xiang; Heller, Katherine; Hefter, Ilana; Leslie, Christina; Chasin, Lawrence; "Sequence Information for the Splicing of Human Pre-mRNA Idnetified by Support Vector Machine Classification", Genome Research, 2003, Vol. 13, pg. 2637-2650.

Web References

[55] Mallory, Charles; University of Miami, "Alternative Splicing of Eukaryotic Genes", http://fig.cox.miami.edu/~cmallery/150/special/mice.men.htm

[56] McClean, Phillip; "Introns, Exons and Splicing hn RNA", http://www.ndsu.nodak.edu/instruct/mcclean/plsc731/transcript/transcript4.htm

[57] Alternative Splicing Annotation Project Database II, Lee Lab, University of California Los Angeles, http://www.bioinformatics.ucla.edu/ASAP2

[58] Tape, Thomas Dr., "Interpreting Diagnostic Tests", University of Nebraska Medical Center, http://gim.unmc.edu/dxtests/roc1.htm

Software References

[59] Chang, Chih-Chung; Lin, Chih-Jen. LIBSVM, a Library for Support Vector Machines, Version 2.81. http://www.csie.ntu.tw/~cjlin/libsvm.

[60] BioPerl – toolkit of Perl modules for building bioinformatics solutions. Version 1.5.1. http://www.bioperl.org/.

[61] Analyse-it – Statistical Software, http://analyse-it.com.