Identifying Splice Sites Of Messenger RNA Using Support Vector Machines

Paige Diamond, Zachary Elkins, Kayla Huff, Lauren Naylor, Sarah Schoeberle, Shannon White, Timothy Urness, Matthew Zwier
Drake University
Des Moines, Iowa 50311
Timothy.urness@drake.edu

Abstract

In eukaryotic cells, gene expression encompasses converting DNA to functional proteins. In order for proteins to be synthesized, the sequence must go through splicing, which excises pre-mRNA to incorporate only the regions of the sequence necessary for the protein. A computational algorithm that can predict canonical and non-canonical splicing patterns would provide insight into such diseases.

One approach utilizes a method based on Term Frequency – Inverse Document Frequency scoring to identify keywords in the base pair sequences. Each sequence is encoded as a vector and then fed into a support vector machine (SVM) that determines the best separation between the positive and negative sequences.

Another approach uses a method based on the Naive Bayes algorithm. This SVM algorithm takes in strings of base pairs from a genome sequence and encodes them as a unique 4-digit identifier of 1s and 0s. The SVM linearly separates the positive and negative examples.

1 Background

1.1 Splicing

In biology, gene expression is the process of synthesizing the molecules that perform the cellular functions required for life. Gene expression requires several steps including converting the genetic instructions found in the DNA into the mRNA (messenger RNA) intermediate, modifying the mRNA, splicing, and converting the mRNA into the final functional protein product. The gene, which is the region of the DNA that encodes one functional mRNA or protein, is a sequence of 1,000 to 1,000,000 nucleotides. These nucleotides can be adenine (A), thymine (T), guanine (G), and cytosine (C). Chemical bonds connect these nucleotides in unique sequence to form individual genes. RNA is similar in nature to DNA; however, it utilizes the nucleotide uracil (U) instead of thymine.

Splicing is a type of modification to mRNA necessary to express functional protein products. Splicing excises non-coding sequences of the original mRNA, which are referred to as introns, and joins together the essential sequences of nucleotides called exons to form the final mRNA template. After splicing, mRNA is available to be made into protein.

The general splicing mechanism involves breaking bonds between specific nucleotides to remove non-essential intron sequences and recreate bonds between exon sequences. During this process, the designated nucleotides of the intron sequence interact with each other to form a lariat structure. This lariat structure is then released from the mRNA and the nucleotides at the end of the exons bond together. The splice site is dictated by patterns of nucleotides found in the intron sequences: beginning of the intron is characterized by GU sequence and end of the intron is characterized by AG sequence. The regions surrounding these patterns dictate whether GU and AG sequences will result in a splicing event (1,2). Our goal is to utilize machine learning to recognize these patterns and predict the location of splice sites within a nucleotide sequence of a gene.

The ability to locate splice sites is important in understanding the final protein product created by a gene and the physiological consequences if a gene is not spliced correctly. Complexity is added to this process because the cell is able to utilize varied combinations of splice sites within certain genes. This process, known as alternative splicing, results in final mRNA sequences that integrate exons in a variety of patterns. The distinct but related proteins created from the same gene template by alternative splicing are called isoforms. These naturally occurring isoforms benefit the cell by enhancing protein diversity and allowing specialization of proteins in different tissues (1,2). However genetic mutations can cause abnormal splicing patterns that produce defective proteins and lead to diseases such as atypical cystic fibrosis and certain types of cancer (2,3). A machine learning program, such as the ones developed, would be able to accurately determine splice sites and provide insight to these abnormal splicing patterns.

1.2 Machine Learning

Within the study of computer science, there is a subfield called artificial intelligence that studies and develops machines and computers capable of intelligent behavior. Within artificial intelligence is another subgroup called machine learning that uses pattern recognition and algorithms to make predictions on data sets. A training set of data is used to build a model algorithm that can then be used to make predictions on a testing data set. Some examples of machine learning include when Netflix offers recommendations for what you should watch next, email spam filtering, and even the self-driving Google cars. With recent advances in technology, the amount of data that we have access to greatly increases the possibilities of what we can do with machine learning.

There are two common types of machine learning, supervised learning and unsupervised learning. The most common type, supervised learning, takes in example inputs along with the desired outcome. For our research, we input strings of base pairs as well as a Boolean value of whether or not the location was identified as a splice site. The machine learning algorithm uses this input to create, or train, a classifier. We can then give the classifier a sequence, and it will predict whether or not the sequence contains a splice site. In order to measure the quality of the classifier, we set aside a portion of our data for testing - this data is not used to train the classifier. In our results, we measure the accuracy, precision and recall of the classifier on the test data. Accuracy measures the percentage of test sequences that were correctly predicted. Precision is the percentage of test sequences that were predicted positive that actually were positive. Recall is the percentage of test sequences that were positive that were predicted to be positive.

Unsupervised learning is where no labels are given and the algorithm has to determine what the data is saying. Inputs can be in the form of classification, when the data can be divided into two or more classes, and regression, where the data is continuous. There are many machine learning algorithms to choose from as well as creating your own.

1.3 Support Vector Machines

A Support Vector Machine (SVM) is an example of a machine learning algorithm and allows for the separation of data. Data is read into the SVM in the form of a vector to find an optimal hyperplane for linearly separable patterns. If data is linearly separable, the positive and negative data points can be separated by a line or a hyperplane in cases for working in multiple dimensions. The data in **Figure 1** is linearly separable because the black and white circles can be linearly separated by both the H2 and H3 lines. To find the optimal separator, the SVM chooses the separator with the largest margin surrounding the line or hyperplane. In **Figure 1**, even though both H2 and H3 linearly separate the data, H3 maximizes the margins whereas H2 does not. When data isn't linearly separable in its

original form, a kernel can be used to transform that data and map it into a new space (**Figure 2**). One common kernel is the Gaussian kernel. Given two training examples X and Y, this kernel computes K(X, Y) = e^(-γ*|X-Y|^2), where γ is a parameter set by the programmer. The value of this function acts as a similarity measure of examples X and Y, and it is used by the SVM algorithm to map the training examples into the new space.

Figure 1: Three different examples (H1, H2, H3) of how a support vector machine could separate the data into two regions (5)

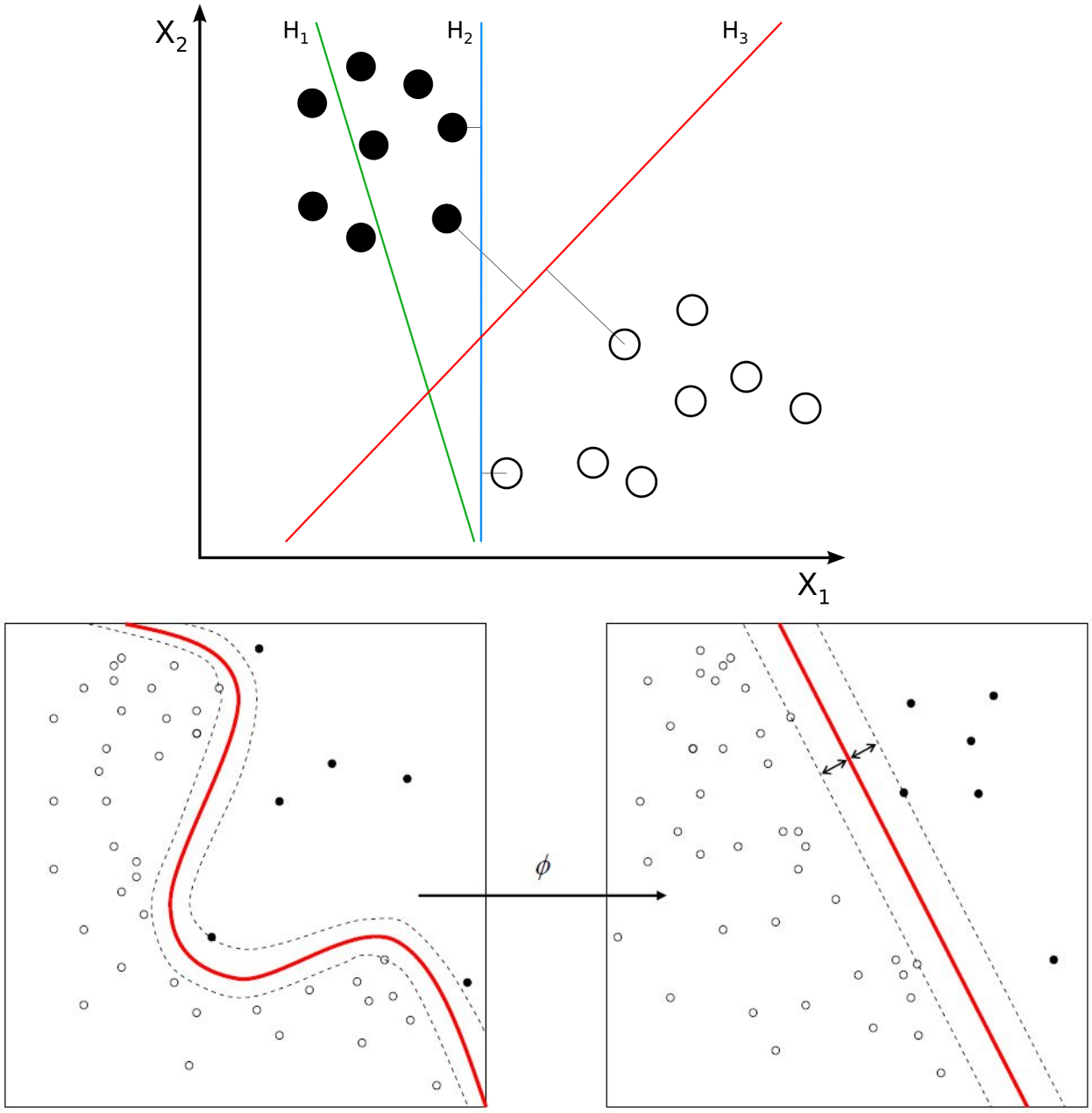

Figure 2: A kernel is used to transform a nonlinear function into a linearly separable function (5)

2 Algorithms

2.1 TF-IDF Scoring

The first algorithm developed in this study uses the positions of "keywords" to predict splice site locations. Our "keywords" are combinations of base pairs (i.e. 'AGT', 'CA', etc.) that are found using a method based on term-frequency inverse-document-frequency (TF-IDF) scoring.

TF-IDF scoring is used to identify keywords in a single text document that distinguish it from a collection of documents (4). For a particular document, a word's term-frequency (TF) is defined as

TF = (The number of times the word appears in the document)/(Total number of words in the document)

A word's inverse-document frequency (IDF) is

IDF = ln((total number of documents)/(number of documents containing the word))

The word's score for the particular document is its TF*IDF. The higher a word's TF-IDF score, the more likely it is to be a keyword. If the word appears many times in the document, its TF will increase, but if it also appears in many other documents, its IDF will lower its total score.

To identify keywords in the mRNA sequences, a maximum word length L was specified. All L-length combinations of 'A', 'C', 'G', and 'T' are generated.

Each word is given a positive and negative TF and IDF. The positive TF is calculated as the average number of times the word appears in positive sequences multiplied by the fraction of positive sequences it appears in. The positive IDF is calculated as the natural log of the total number of negative sequences divided by the number of negative sequences in which the word appears more than 75% of the positive average. The negative TF and IDF are calculated in a similar way (with all instances of 'positive' and 'negative' reversed). A word's overall TF-IDF score is taken to be the difference between its positive TF*IDF and negative TF*IDF. The bigger this difference is, the better the word is at identifying either positive or negative sequences.

We use the 30 words with the highest TF-IDF score to encode our sequences as vectors.

To encode each sequence as a vector, a similar method as Zhang et al. was used (6). First, we make a positive and negative probability vector for each keyword. Each probability vector is the same length as our sequences. Each index of the vector contains the probability that the keyword is at that position in a positive (negative) sequence. These probabilities are found by counting the number of positive (negative) training sequences in which the word is at that index and dividing by the total number of positive (negative) training sequences (6).

For example, assume the following positive training sequences: 'ACATG', 'CATCC', and 'TGCAT', and that CAT is a keyword. CAT occupies index 0 in one out of our three training sequences, so the first entry of CAT's positive probability table is 1/3. CAT occupies index 1 in two of our training sequences, so its second entry would be 2/3. The entire positive probability table would be [1/3, 2/3, 1, 2/3, 1/3].

Each sequence is then converted to a vector twice its length. For the first half of the vector, we find all positions in the sequence that contain a keyword. For each of these positions, we look up the keyword's positive probability for that position, and enter it into that position of the sequence's vector. The same is done with the second half of the vector, using the negative probability tables. Positions that do not contain keywords are set to zero.

Continuing the previous example, suppose we want to encode the sequence 'GCATT', and CAT is our only keyword. We have its positive probability table: [1/3, 2/3, 1, 2/3, 1/3] and say we found its negative probability table to be: [1/4, 2/3, 1/2, 1/3, 0].
We would then convert 'GCATT' to the vector [0, 2/3, 1, 2/3, 0, 0, 2/3, 1/2, 1/3, 0].

2.1 Multinomial Naive Bayes

Our second algorithm encodes each A, C, G, and T as a numeric code of 4 binary digits. The algorithm doesn't vary the length of each sequence of converted bases inputted into the algorithm. Each sequence input is 60 base pairs long before conversion to numerical values. This form of encoding is known as sparse encoding. Sparse encoding is implemented in order to convert strings of characters into a sequence of numbers that can be processed computationally by the algorithm of choice. We found a research paper written by Zhang et al. that detailed the sparse encoding process, and we implemented the same encoding values as the ones in their paper: A → 1000, C → 0100, G → 0010, T → 0001.

The program first converts every sequence of base pairs into a numerical sequence via the sparse encoding process detailed above. Then, the program splits the data into a training set and a test set, 80% and 20% of the data, respectively. After encoding the data, we passed the training set through an SVM running a multinomial Naive Bayes kernel. The

multinomial Naive Bayes kernel is a supervised learning algorithm that implements the Bayes theorem while assuming independence between each example. Bayes theorem is a mathematical theory that describes the probability of an event happening. By applying this mathematical theorem to an SVM, we can use information gained from the inputs to predict the probability of whether or not each input contains a splice site. The multinomial aspect merely refers to the fact that we have more than two examples, and our data has a multinomial distribution. The kernel mapped our data onto a virtual graph. Then, the SVM formed a line on the graph that best separated the mapped data based on whether or not each sequence contained a splice site or not.

After mapping the training data, the SVM then maps the test set onto the a virtual graph containing the same line that the SVM formed based on the training set. It then predicts whether or not each sequence contains a splice site based on which side of the line it falls. The program then spits out an accuracy rating for how many of the test set sequences it correctly predicted.

3 Results

3.1 TF-IDF Scoring Results

Our data consisted of mRNA sequences 60 basepairs in length from human genes. We had 786 positive examples, that had a splice site located in the center of the sequence, and 1938 negative examples that did not contain splice sites. For each test, we randomly selected 80% of our data for training and 20% for testing. The encoded sequences were fed into an SVM using a Gaussian kernel, with γ=.1. We found the average accuracy, precision, and recall over 50 runs.
We varied the maximum length of words, but this did not have a large impact on the results. We do see a large difference when using words with the highest TF-IDF scores versus words with the lowest TF-IDF scores. Figure 3 shows the accuracy, precision, and recall from using the top 30 words as well as using the bottom 30 ranked words, with a maximum word length of five.

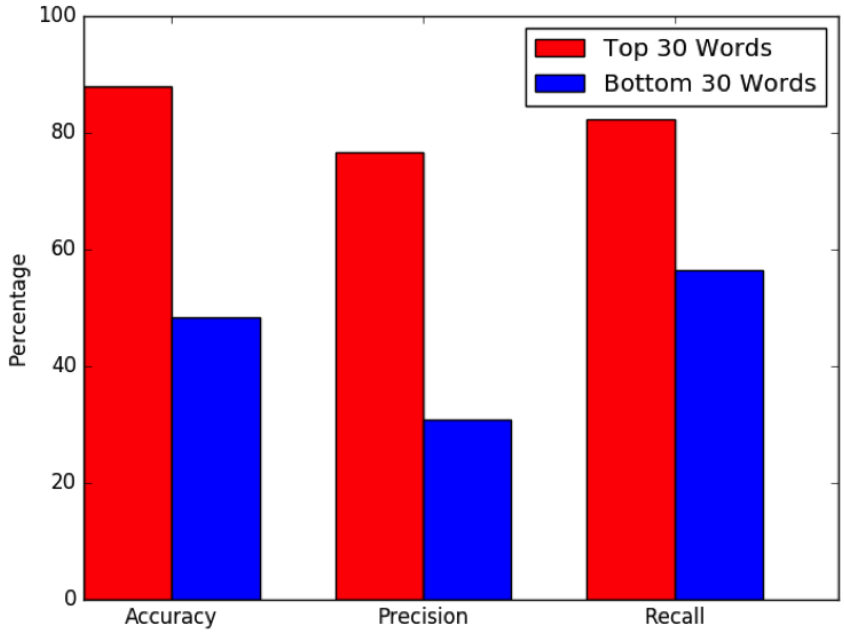

Figure 3: Comparison of the accuracy, precision, and recall percentages of using the top 30 words and bottom 30 words as scored by our TF-IDF measure to encode the sequences.

3.2 Multinomial Naive Bayes Results

Our data consisted of mRNA sequences 60 basepairs in length from human genes. We had 786 positive examples, that had a splice site located in the center of the sequence, and 1938 negative examples that did not contain splice sites. For each test, we randomly selected 80% of our data for training and 20% for testing. The sparse encoded sequences were ran through an SVM using a Multinomial Naive Bayes kernel. We found the average accuracy over 100 runs to be 71.06%.

4 Conclusion

We were able to obtain reasonable accuracy combining our TF-IDF scoring with SVMs. Further research could be done to determine if setting a minimum word length affects accuracy, as most of our top 30 words were two or three letters long, even when the maximum word length was increased. Additionally, our top 30 words frequently included word pairs such as ‘AA’ and ‘AAA’. Our method could be developed to treat ‘AA’ as a sub-word and only identify ‘AAA’ as a keyword.

As for the Multinomial Naive Bayes algorithm, we achieved a decent accuracy by converting sequences of 60 base pairs to a numerical sequence via sparse encoding. While the sparse encoding method performed well, the TF-IDF Scoring results showed better accuracy. Going forward, we would like to vary the length of the sparse encoded sequences entered into the SVM in order to see its impact on accuracy scores. In addition, the fact that our algorithm scored above average, while not scoring in the ninety percentile range, means that the algorithm might show generalization across different genomes. To test this, we must input different genomic sequences from different species as testing sets into the SVM; if the SVM gives us a similar accuracy, then it is more generalizable.

References

[1] Black, Douglas L. (June 2003). "Mechanisms of Alternative Pre-Messenger RNA splicing". *Annual Review of Biochemistry* **72**(1): 291-336.

[2] Busch, Anke and Klemens J. Hertel (2015). "Splicing predictions reliably classify different types of alternative splicing." *RNA* **21**: 1-11.

[3] Faustino, Nuno A and Thomas A. Cooper. (2003). "Pre-mRNA splicing and human disease." *Genes & Development* **17**: 419-437.

[4] Salton, Gerard, and Christopher Buckley (1988). "Term-weighting approaches in automatic text retrieval." *Information processing & management* **24.5**: 513-523.

[5] Support vector machine. (2016, March 3). In Wikipedia, The Free Encyclopedia. Retrieved 02:34, March 21, 2016, from https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=708088680

[6] Zhang, Ya, et al.(2006). "Splice site prediction using support vector machines with a Bayes kernel." *Expert Systems with Applications* **30.1**: 73-81.