**Boundary Classification Learning from Splice-junction Gene Sequences Using Machine Learning Algorithms**

*Kamalanathan S. Govindan, Wenting Guo*

**Abstract**

Exons and introns have important roles in molecular biology, as splicing is the editing of the nascent precursor messenger RNA transcript, and after splicing, introns are removed and exons are joined together. The problem for us is, when given a sequence of DNA, we can identify the boundaries between the introns and exons, and classify the two types of boundaries: exon-intron site known as donors and intron-exon site, known as acceptors.

The dataset is from UCI Machine learning repository, which all examples were taken from Genbank 64.1 (ftp site: genbank.bio.net). We will use cross-validation method to randomly select 1000 samples each time from the full 3190 data. We will use decision tree ID3 and C4.5 algorithm, and compare the accuracy rate of three different ensemble methods for constructing the tree: Bagging, Boosting, and Randomization. The algorithms will be implemented using python. It is expected that the result of C4.5 should perform better than ID3.

**Keywords**

Exon–intron splice sites, DNA sequence, Machine learning

**Introduction**

Genetic information of an organism is stored in the genes, the functional subunits of the genome, arranged in the strands of the DNA double helix in the nucleus. This information is transcribed from DNA into a messenger RNA (mRNA) template by a process called transcription. However, before the mRNA can be translated into proteins, non-coding portions of the sequence, called introns, must be removed and protein-coding parts, called exons, joined by RNA splicing to produce a mature mRNA.

Scientists have discovered alternative patterns of pre-mRNA splicing that produced different mature mRNAs containing various combinations of exons from a single precursor mRNA. Alternative splicing therefore is a process by which exons or portions of exons or noncoding regions within a pre-mRNA transcript are differentially joined or skipped, resulting in multiple protein isoforms being encoded by a single gene. This mechanism increases the informational diversity and functional capacity of a gene during post-transcriptional processing and provides an opportunity for gene regulation.

Alternative splicing generates a tremendous amount of proteomic diversity in humans and significantly affects various functions in cellular processes, tissue specificity, developmental states, and disease conditions. Thus, finding the intron-exon and exon-intron boundaries are important for further studying of RNA splicing process, and gene expression.

**The Data**

The dataset of 3190 rows contains three main attributes.:

The first is the column that indicates the class variables - {ei, ie, n}: indicating whether the sequence in the third column contains an exon-intron boundary, intron-exon boundary or neither. An intron-exon boundary sequence is otherwise called acceptor and an exon-intron boundary sequence is called a donor. The second column contains the name of the instance relevant to the sequence. The third column contains the 60 base pairs starting from a position or index of -30 to +29 where the indices -1,0 denote the boundary of the intron-exon or exon-intron in the case of ie or ei sequence. This establishes the notion pertaining to the dataset that the first half or the second half or neither in the sequences provided in the dataset are intron sequences which is spliced out in the transcription process.

**Neural Network**

The implementation of Neural Network was done in two methods. The first was to construct the neural network from scratch and the second was to implement the MLPClassifier module from sklearn Python package. The following methodology and results are given for the latter. The code for the first method is also provided in the team repository.

Neural Networks are implemented to this dataset by converting the data into numeric quantities carrying meaning. From the genome sequence dataset with 60 basepairs length, the first order Markov transition matrix is prepared for the transitions from first to second, second to third and so on. The probabilities are now held accountable as quantifiable data. The back propagation reduces the error and tunes the **w** such that the rights weights are obtained for each node in the subsequent hidden layer as a linear combination of the weights dotted with the activations in the previous layer.

An additional modification in the data includes to split the dataset into half such that there are two datasets with one half of a sequence in each. This gives us two datasets with 3190 rows and sequences with 30 basepairs length. Owing to that fact that in our data, either the first half of the sequence is definitely an intron sequence or the second half of the sequence is definitely an intron sequence or neither, and also to the fact that there is a known pattern in the terminal ends of the intron sequences, we can hope to get reasonable weights at the ends when trained on the markov probabilities of the nucleotides in an intron sequence. After the data is split into two different datasets, each dataset is trimmed (by rows) according to whether the sequence is an intron and the training label provided in the dateset for the sequence. It is taken care that in each dataset formed, the sequences either come exclusively from donor sequences and neither sequences or exclusively from acceptor sequences and neither sequences. This is to maintain consistency with the training method and to keep consistency in the order of calculating the probability with respect to the positions in a sequence.

The data obtained using this method is provided in output.csv file.

The two neural networks are trained separately and two different trainers are obtained as a result. This is important because we will use both the trainers in predicting the splice boundary. The two trained trainers are passed through an XOR trainer and checked for validation. Inference is made based on the results obtained from both the trainers and combining them using the third collective neural network. When the output is 1 (for the cases of {0,1} and {1,0}) we know that there it is a splice boundary and the type can be determined by which of the two trainers is 1. In case of 0 (for cases of {0,0} and {1,1}) we can infer that there is an anomaly and the trainer will classify as absence of boundary site in the middle of the sequence.

The MLPClassifier module from the sklearn package was used to implement the neural networks algorithm. The results were compared for difference solvers and optimization technique combinations. The following bar graphs represent the accuracy of training the neural networks with the mentioned parameters:



We can see that the first trainer is performing well in almost all the combinations with an accuracy of about 94% which is not bad at all. The second trainer however, is not doing as well as the first. This is because of the error in the Markov Probability of the second half of the sequence. The numbers are inconsistent because for the first nucleotide in the second half of the sequence, is counted for with respect to the previous nucleotide in the original sequence. This inconsistency with the new dataset causes the learning algorithm to perform poorly on the second trainer.

**Decision Trees and Ensemble Methods**

The algorithm C4.5 developed by [Ross Quinlan](https://en.wikipedia.org/wiki/Ross_Quinlan) can generate a [decision tree](https://en.wikipedia.org/wiki/Decision_tree_learning) for classification.

Based on a set of training data, C4.5 builds decision trees using [information entropy](https://en.wikipedia.org/wiki/Entropy_(information_theory)). The training data has features as well as the labeled class. At each node of the tree, C4.5 chooses one feature that has the highest normalized information gain, which means it can most effectively split its set of samples into subsets enriched in one class or the other. We will implement C4.5 using the J48 in Weka, and 10 fold cross validation method to limit overfitting issues.

Ensemble methods can construct a collection of weaker methods to increase accuracy. We will mainly compare the three popular ensemble methods for decision tree classifier: Bagging, AdaBoost, and Random Forest.

Both bagging and AdaBoost is to use weak learners and operate on different training sets for many times. Bagging uses bootstrap sample, which means to generate new training sets from the original training set by sampling uniformly and with replacement, to average the voting for classification. It can help reduce variance and avoid overfitting. Adaboost is to combine the weak classifiers but give the samples in training set updated weights based on the accuracy from the weak learners. The weights for the examples that are misclassified increase and the weights for the correctly classified examples decrease. However, AdaBoost may be sensitive to noisy data and [outliers](https://en.wikipedia.org/wiki/Outlier).

Random forest is a variant of bagged trees. Instead of using all the features, random forest randomly chooses a certain number of features each time when constructing the decision tree. Random forest can decrease the problem of overfitting. This special ensemble method for decision tree helps to reduce variance as the trees are more independent because of the combination of bootstrap samples and random draws of predictors. At the same time, it can reduce bias as a very large number of predictors can be considered, and local feature predictors can play a role in the tree construction.

**Results**

The results obtained by the neural networks show that the accuracy of the model built for the case of the first learner is 94% and for the second learner is about 75%. This means that the probability of a sequence to be classified correctly is 0.94 for acceptor sequences and those which are neither acceptors nor donors, while it is around 0.75 for donor sequences.

The results of the decision tree and the three ensemble methods can be found in figure 1. The performance of these traditional methods are better than our neural network algorithms in the previous experiment.

As expected, the ensemble methods work better than its base learner. Among them, random forest works the best reaching the highest averaged accuracy of 95.7%, with bagging following it at an average of 94.4%.

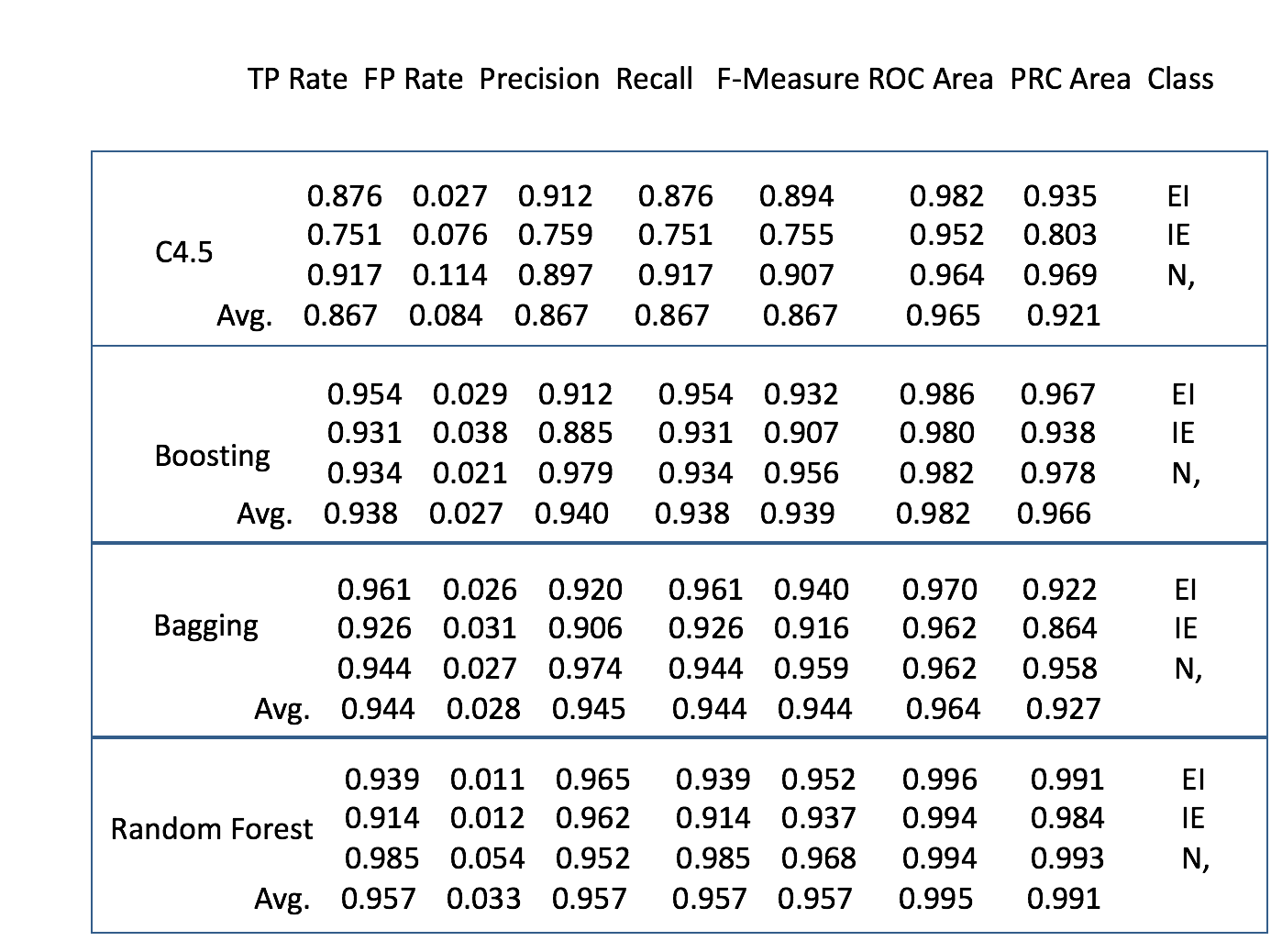


Figure 1

**Conclusion**

Decision trees are guaranteed to work in all the cases of classification, but Neural Networks work as good as the ensemble methods only for a select cases of the problem. The Neural Network can be improved by adjusting for the Markov inconsistency in the diving the sequences into two. The second training dataset can be modeled to be more similar to the first in order to get similar accuracy results.

**References**

*An Improved Method for Splice Site Prediction in DNA Sequences Using Support Vector Machines*

Neelam Goel, Shailendra Singh, Trilok Chand Aseri (2015)

*Classification of splice-junction sequences via weighted position specific scoring approach*

Efendi Nasibov[a](http://www.sciencedirect.com/science/article/pii/S1476927110000770#aff0005),  Sezin Tunaboylu[a](http://www.sciencedirect.com/science/article/pii/S1476927110000770#aff0005) (2010)

*Identifying splice-junction sequences by hierarchical multiclassifier*

Alessandra Lumini , Loris Nanni (2006)

*Prediction of human mRNA donor and acceptor sites from the DNA sequence*

Jacob Engelbrecht (1992)

*Recognition of splice junctions on DNA sequences by BRAIN learning algorithm*

Salvatore Rampone (June 10, 1998)

*Evaluation of Techniques for Classifying Biological Sequences*

Mukund Deshpande and George Karypis (2002)

*Splice site detection with a higher-order Markov model implemented on a neural network*

Ho and Rajapakse (2003)

*Splice Site Prediction Using Artificial Neural Networks*

Øystein Johansen, Tom Ryen, Trygve Eftesøl, Thomas Kjosmoen, and Peter Ruof (2008)