Genomics 112 (2020) 1847–1852

Contents lists available at ScienceDirect

Genomics

journal homepage: www.elsevier.com/locate/ygeno

Splice sites detection using chaos game representation and neural network

Tung Hoang[a], Changchuan Yin[a], Stephen S.-T. Yau[b,*]

[a] Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA
[b] Department of Mathematical Sciences, Tsinghua University, Beijing 100084, P.R. China

ARTICLE INFO

Keywords:
Exon
Intron
Splice sites
Acceptor
Donor
Artificial neural network
Chaos game representation

ABSTRACT

A novel method is proposed to detect the acceptor and donor splice sites using chaos game representation and artificial neural network. In order to achieve high accuracy, inputs to the neural network, or feature vector, shall reflect the true nature of the DNA segments. Therefore it is important to have one-to-one numerical representation, i.e. a feature vector should be able to represent the original data. Chaos game representation (CGR) is an iterative mapping technique that assigns each nucleotide in a DNA sequence to a respective position on the plane in a one-to-one manner. Using CGR, a DNA sequence can be mapped to a numerical sequence that reflects the true nature of the original sequence. In this research, we propose to use CGR as feature input to a neural network to detect splice sites on the NN269 dataset. Computational experiments indicate that this approach gives good accuracy while being simpler than other methods in the literature, with only one neural network component. The code and data for our method can be accessed from this link: https://github.com/thoang3/portfolio/tree/SpliceSites_ANN_CGR.

1. Introduction

Determining DNA sequence is only a first step to learn how genetic information works. Understanding the functionality, detecting the locations of genes, protein-coding regions, and regulatory sites are the ultimate goals. Among these tasks, splice sites detection plays an important role, since identification of protein coding regions requires the locations of all the exons and introns for each gene. In *eukaryotes*, a gene comprises several *exons* and *introns*, where introns are the non-coding segments which would be spliced out in the mRNA processing [1]. The remaining segments of a gene are called exons, which encode for proteins. The intron/exon boundaries are called *splice sites*, where the intron to exon boundary is called *acceptor* splice site, and the exon to intron boundary is called *donor* splice site.

Direct experimental analysis of biological sequences in the laboratories is expensive and time-consuming, given the complexity of the data. Therefore, computational techniques like machine learning are ideal solutions for the tasks like splice sites detection and protein-coding regions detection. In the literature, Farber et al. [14] reported a neural network for determining eukaryotic protein coding and non-coding regions in a DNA sequence. Uberbacher and Mural [35] used a multiple sensor-neural network approach to locate protein-coding regions in human DNA sequences, where exons are identified by combining information from numerous content statistics and these scores are weighted using a neural network. Naito [28] used a hybrid neural network consists of convolutional layers and bidirectional long short-term memory layers for splice sites prediction in human. In many cases, sequence's signals such as splice sites, start and stop codons are evaluated and weighted, and these features are used as input to feed in neural networks.

An artificial neural network (ANN) is an interconnected group of artificial neurons, which by imitating the way the brain works, can represent complex input-output relationships [5]. ANNs can learn from examples, and are especially helpful when an algorithmic solution cannot be formulated, but plenty of training data is available. A multilayer neural network has the ability to capture and discover high-order correlations of input data. As a result, neural networks have been applied to various bioinformatics problems, such as sequence classification, gene identification, and have gained more and more attention in the last few decades.

Beside ANNs, several alternative methods have also been proposed. Pashaei et al. [29] combined support vector machine (SVM) and random forest (RF) in splice site detection of Human genome. Baten et al. [4] proposed to combine a first order hidden Markov model (HMM) and support vector machine for splice sites identification. As generative models, HMMs make the Markovian assumption, in which the current state depends only on a predefined number of previous states, and transitions between these states are represented by a

* Corresponding author.
E-mail address: yau@uic.edu (S.S.-T. Yau).

https://doi.org/10.1016/j.ygeno.2019.10.018
Received 18 April 2018; Received in revised form 18 March 2019; Accepted 29 October 2019
Available online 05 November 2019
0888-7543/

transition matrix. In the literature, high order HMMs are better in capturing interactions among the nucleotides around splice sites [2,8]. Nevertheless, the transition matrix grows exponentially with respect to the order of the HMM model. For a k-th order HMM, the total number of parameters is 4^{k+1}, where 4 is the size of the bases {*A, C, G, T*} [4].

Unlike HMMs, ANNs do not need to make the Markovian assumption. Their rich internal state allows them to keep track of long-term dependencies when modeling a sequence of symbols. In ANNs, hidden states are distributed efficiently that enables them to store a lot of information about the past. Moreover, ANNs apply non-linearity activation functions to the outputs at every timestep, which allow them to update the hidden states in more abstract ways [5].

In order to achieve high accuracy, inputs to a neural network, or feature vector, should be able to represent the original data. Chaos game representation (CGR) is able to give both numerical and graphical representations for genomic sequences in a one-to-one manner [21]. By the unique properties of CGR, subsequences of a gene or genome exhibit the main characteristics of the whole sequence, thus it is useful in the detection of special genome features, especially when parts of the genome have not been available yet [12]. In many cases, certain patterns can be observed in the representation of a DNA sequence. Therefore, it would be beneficial to discover the essential properties of the sequence, which in turn may lead to a meaningful conclusion that cannot be detected from the symbolic sequences.

In this research, we devise a method for detecting splice sites using chaos game representation and artificial neural network. We employ chaos game representations of DNA sequences, and use CGR as feature vector input to an ANN to classify splice sites of the NN269 data set [30].

2. Materials and methods

2.1. Search by signal

There are two main classes of computational approaches to detecting genes and the corresponding protein-coding regions in DNA sequences: *search by content* and *search by signal*. Search by content locates genes and coding regions directly by exploiting the statistical difference of general properties between coding and non-coding regions. Some common content measures are codon usage, nucleotide composition, GC content, base occurrence periodicity, and hexamer frequency [10]. Meanwhile, search by signal recognizes genes and coding regions indirectly by finding some special signals that are associated with gene expression. A signal is a localized region of DNA sequence that performs a particular function, such as binding an enzyme. Some common signals are translation initiation sites (start codons), translation termination sites (stop codons), transcription initiation sites (promoters), transcription termination sites (terminators), enhancers, splice-junction sites (acceptors/donors) [10].

Search by signal approaches can be considered as a classification problem in machine learning: given a fixed-length window on a DNA sequence, determine if the window contains the signal of interest based on some associated features of the signal. This is a supervised learning problem, where the classifier is trained on a set of given positive and negative examples.

In this research, we classify splice sites (acceptors/donors) and non splice sites using neural networks. Approximately 99% of all introns have a GT base pair as donor site, and an AG base pair as acceptor site [9]. We focus on classifying these canonical splice sites (Fig. 1). Therefore, the input data to the neural network would be windows containing GT or AG at a fixed location of the sequence, and the output would be 1 for true splice sites, and 0 for false splice sites.

2.2. Chaos game representation

Chaos game representation (CGR) is able to give both numerical and graphical representations for genomic sequences [21]. For a DNA sequence $s_1s_2\ldots s_n\ldots$, the corresponding CGR sequence $(X_n) = (x_n, y_n)$ is given by:

$$X_0 = \left(\frac{1}{2}, \frac{1}{2}\right),\ X_n = \frac{1}{2}(X_{n-1} + W)$$

where W is coordinates of the corners of the unit square A = (0,0); C = (0,1); G = (1,1); T = (1,0) if s_n is *a, c, g, t* respectively.

Geometrically speaking, given the unit square in the Euclidean plane, the binary CGR vertices are assigned to the four nucleotides as A = (0,0); C = (0,1); G = (1,1); T = (1,0). Starting with the center of the unit square, $(\frac{1}{2}, \frac{1}{2})$, the CGR position of each nucleotide of the DNA sequence is calculated by moving a pointer to half the distance between the previous point and the corner square of the current nucleotide. Fig. 2 illustrates the CGR of a short sequence "*gaattc*" [19].

By definition, there is a one-to-one correspondence between the subsequences counted from the start of a DNA sequence and points of the CGR [19]. With this unique property of CGR, the subsequences of a gene or genome exhibit the main characteristics of the whole sequence, thus it is useful in the detection of special genome features, especially when parts of the genome have not been available yet [12]. Fig. 3 shows the CGR of Human beta globin region on Chromosome 11 (HUMHBB) as an example of 2-D fractal pattern of CGR of a DNA sequence.

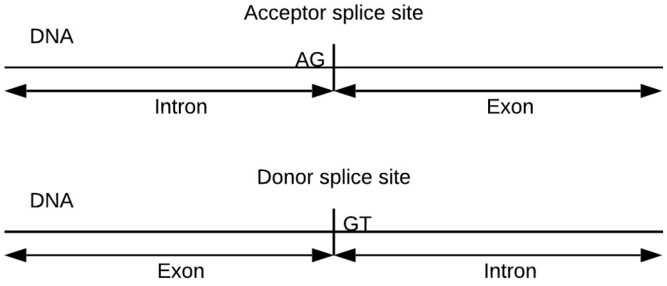

Fig. 1. Acceptor and donor splice sites.

2.3. Dataset

NN269 dataset has been extracted from 269 human genes as a benchmark splice site dataset [30]. It consists of 1324 true donor splice sites, 4922 false donor splice sites, 1324 true acceptor sites, and 5552 false acceptor sites. Each donor splice site has 15 nucleotides with GT at

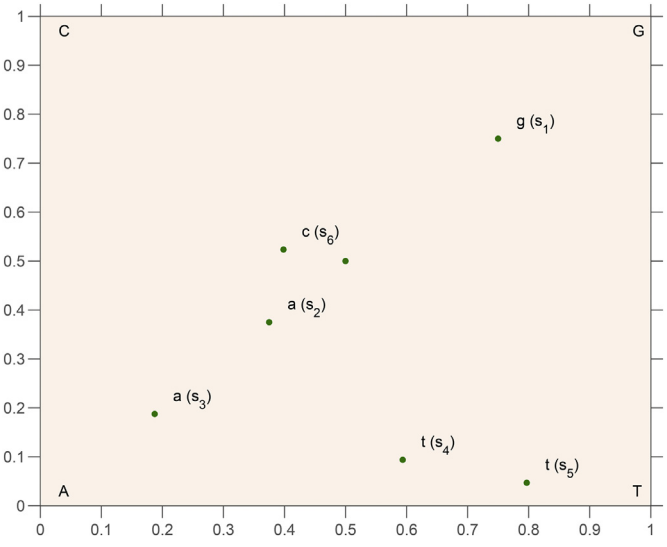

Fig. 2. CGR of $s_1s_2s_3s_4s_5s_6$ = "*gaattc*".

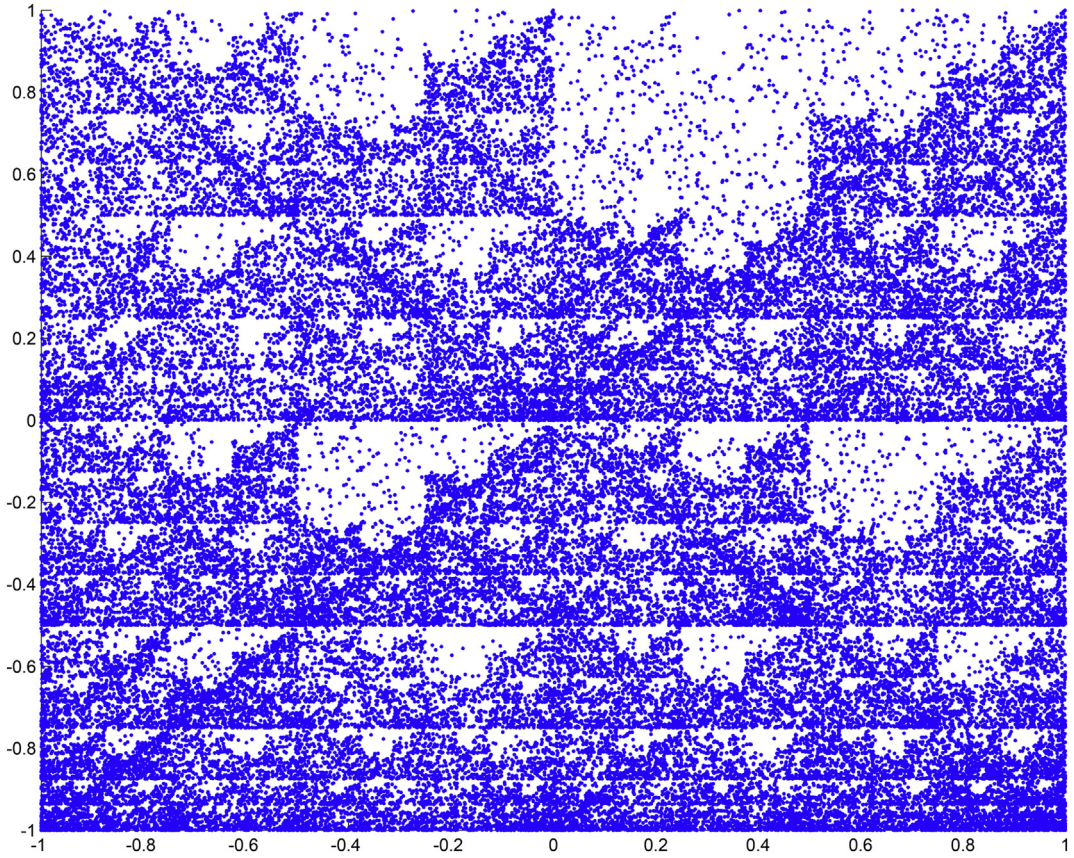

Fig. 3. CGR of Human beta globin region on Chromosome 11 (HUMHBB).

8th and 9th positions. The donor dataset is partitioned into a training set containing 5256 splice sites (1116 true and 4140 false), and a test set containing 990 splice sites (208 true and 782 false). Similarly, each acceptor splice site has 90 nucleotides with AG at 69th and 70th positions. The acceptor dataset is partitioned into a training set containing 5788 splice sites (1116 true and 4672 false), and a test set containing 1089 splice sites (208 true and 881 false).

2.4. Machine learning techniques

Machine learning is a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty [27]. In other words, machine learning is concerned with the development and application of computer algorithms that can "learn" and improve with experience [6]. Its major focus is to automatically extract information from data by computational and statistical methods. Machine learning has many applications: natural language processing, stock market analysis, spam detection, image detection, recommender systems, robotics, game playing, and medical diagnosis.

There are two main categories for machine learning methods: *unsupervised learning* and *supervised learning*. *Unsupervised learning* methods discover structure in data without using labels, such as clustering algorithm. This approach is very useful when a labeled training set is not available.

Supervised learning methods like classification are trained on labeled data and used to make predictions about new unlabeled data. First, an algorithm is developed and provided with a large collection of data, with positive and negative labels. The data is split into training data and testing data. The algorithm is trained using training data to maximize the ability to classify labels. It then is tested on the testing data. If the learning process is successful, then ideally all or most of the predicted labels on testing data will be correct. Among supervised learning methods, the artificial neural network is one of the most common and effective methods.

2.4.1. Artificial neural network

An *artificial neural network* consists of simple processing units called neurons [6]. Neurons are arranged into layers, so that a neural network has one input layer, one output layer, with zero or more hidden layers in between. Neurons from one layer are connected to the ones in the adjacent layer with associated weights. Each neuron in the current layer computes the weighted sum of the input values from all or part of neurons in the previous layer. The weighted sum is then fed in a respective function of the current neuron, called *activation function* (or *transfer function*), to get an output value. This output value can then be used as input to neurons in the next layer and so on. Several activation functions are used in designing neural networks, such as: hyperbolic tangent sigmoid function ($tansig(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$), logistic sigmoid function ($logsig(x) = \frac{1}{1+e^{-x}}$), linear function ($purelin(x) = x$). It is remarkable that artificial neural networks try to mimic biological neural networks to some extent.

A neural network with no hidden layer is called a perceptron [32]. This is the simplest form of a neural network, which is suitable for simple tasks such as linearly separable classification [17]. The weights of the perceptron are learned using an error-correction rule named Perceptron Convergence Theorem [26].

Multilayer perceptron (MLP) is a neural network with one or more hidden layers (Fig. 4, [6]). MLPs are more generalized in the sense that it can approximate any function to any level of accuracy, given enough training data and hidden neurons [36]. In general, neural network is motivated by the neurons of the brain. It allows "features" to be constructed at hidden levels.

Most of the MLP models use supervised learning algorithms. In the

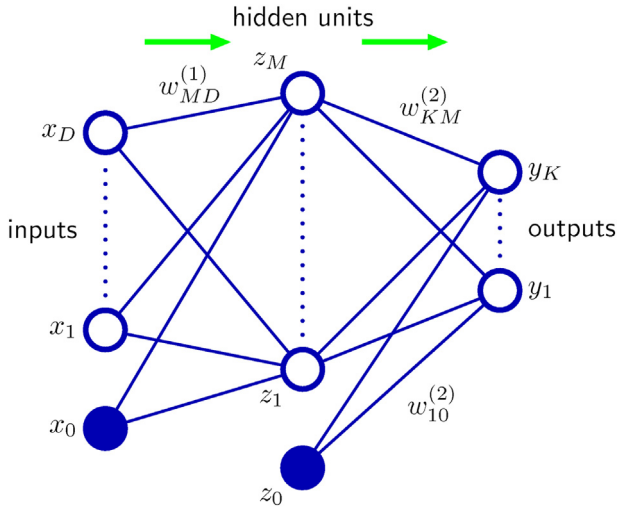

Fig. 4. Multilayer perceptron.

training phase, first the weights of the MLP are given some initial, often random, values. At each round of training, called epoch, training examples are fed into the input nodes so that the algorithm can "learn". The weights are adjusted in each epoch according to some learning rules in order to minimize the difference between the actual output and the desired output. One of the most prominent learning rules is the backpropagation algorithm (BP) [5]. In the backpropagation method, error signals are computed recursively for each neuron starting at the output layer, and propagated backward layer by layer through the network to derive the errors of the hidden neurons. Weights are then adjusted accordingly to decrease the difference between the target output and actual output [37]. If the accuracy of the method on testing data satisfies the need of the classification task, the MLP model is saved with all of its parameters, activation functions, and weights; and is ready to be applied in a predictive model to future data. On the other hand, if the accuracy of the method does not satisfy the requirement, then factors like activation functions, feature extraction methods, number of hidden layers, and number of neurons in each layer should be adjusted so that the MLP could be retrained.

2.5. Neural network design

In the literature [7,14,18], it is common for DNA base symbol to be encoded into a 4-bit string as follows:

A → 1000 T → 0001 G → 0010 C → 0100

With this encoding, the number of input neurons would be 4 times larger than the number of nucleotides in the DNA segment input. Consequently, the number of weights that needed to be trained in the neural network will increase. For example, if the DNA segment input has length 50 nucleotides, then the number of input neurons would be 200. In this work, we employ the advantage of Chaos Game representation, where each nucleotide can be represented by a point in the Euclidean plane, or a pair of real numbers. As a result, the number of input neurons would be only 2 times larger than the number of nucleotides in the DNA segment input.

Chaos game representation is used as features inputs for an artificial neural network. Two separate ANNs are constructed, one is to distinguish donor sites from non-donor sites, and the other is to distinguish acceptor sites from non-acceptor sites. The method is trained using NN269 data. We design two 1-hidden layer neural networks M x L x 1 as follows, where M represents the input layer, L represents the hidden layer, and there is 1 neuron in the output layer:

Input: A set of n DNA segments of a fixed length (15 for donor, 90 for acceptor).

Output: Splice site classification, with 1 being True splice site, and 0 being False splice site.

Steps:

1. Convert DNA sequence into numeric sequence using CGR.
2. Use CGR coordinates as feature inputs.
3. Feed in the neural network, with M = 15 × 2 = 30 for donor neural network, and with M = 90 × 2 = 180 for acceptor neural network.

The size of the hidden layer L is tested during the training process. The networks are trained using backpropagation such that the output neuron attains a value near 1 if the input contains a splice site in the segment, and a value near 0 otherwise. The neural networks in this research are implemented using MATLAB Neural Network Toolbox on a PC with the configuration of Intel Core i7 CPU 2.40GHz and 8Gb RAM.

2.6. Evaluation methods

The prediction accuracy is measured in terms of area under the Receiver Operator Characteristic Curve [25], or auROC, and area under the Precision Recall Curve [11], or auPRC. The overall classification accuracy is not a good measure in this case due to imbalanced class size (the false, or negative, class is 4 times larger than the true, or positive, class). Both ROC and PRC measures are based on TP (true positives), FP (false positives), TN (true negatives), and FN (false negatives) (Fig. 5). The ROC depicts the relative trade-offs between true positives and false positives, which can compare the performance of the classifier across the entire range of class distributions [15]. Specifically, the ROC plots the true positive rate (TPR = $\frac{TP}{TP+FN}$) as a function of the false positive rate (FPR = $\frac{FP}{FP+TN}$). If auROC is close to 0.5 then the performance is close to random, else if auROC is close to 1 then the performance is close to being perfect. However, if the data is too imbalanced, then auPRC is a better measure, as auROC is independent of class size ratios [34]. The PRC measures the fraction of negatives misclassified as positives, and so plots the precision ($\frac{TP}{TP+FP}$) as a function of recall (TPR or sensitivity). See Fig. 6 for an illustration of auROC and auPRC.

3. Results

The proposed method is tested on hidden layer size from 5 to 60. Larger layer size does not give better results on performance accuracy. Different activation functions have been tried for both activation functions from the input layer to the hidden layer, and from the hidden layer to the output layer. The functions that provide the best results are logsig and tansig (Section 2.4.1).

On donor splice site neural network, the best performance is 97.82 for auROC, and 91.57 for auPRC. This result is achieved for activation functions tansig and tansig, with 30 being the size of the hidden layer. On acceptor splice site neural network, the best performance is 97.12 for auROC, and 89.47 for auPRC. This result is achieved for activation functions logsig and tansig, with 20 being the size of the hidden layer.

		Actual class: Negative	Actual class: Positive	Total
Predicted class	Negative	True Negatives (TN)	False Negatives (FN)	$TN + FN$
	Positive	False Positives (FP)	True Positives (TP)	$TP + FP$
	Total	$FP + TN$	$TP + FN$	

Fig. 5. Confusion matrix. True Positive Rate (or Recall) TPR = $\frac{TP}{TP+FN}$. False Positive Rate (or Fall-out) FPR = $\frac{FP}{FP+TN}$. Positive Predictive Value (or Precision) PPV = $\frac{TP}{TP+FP}$.

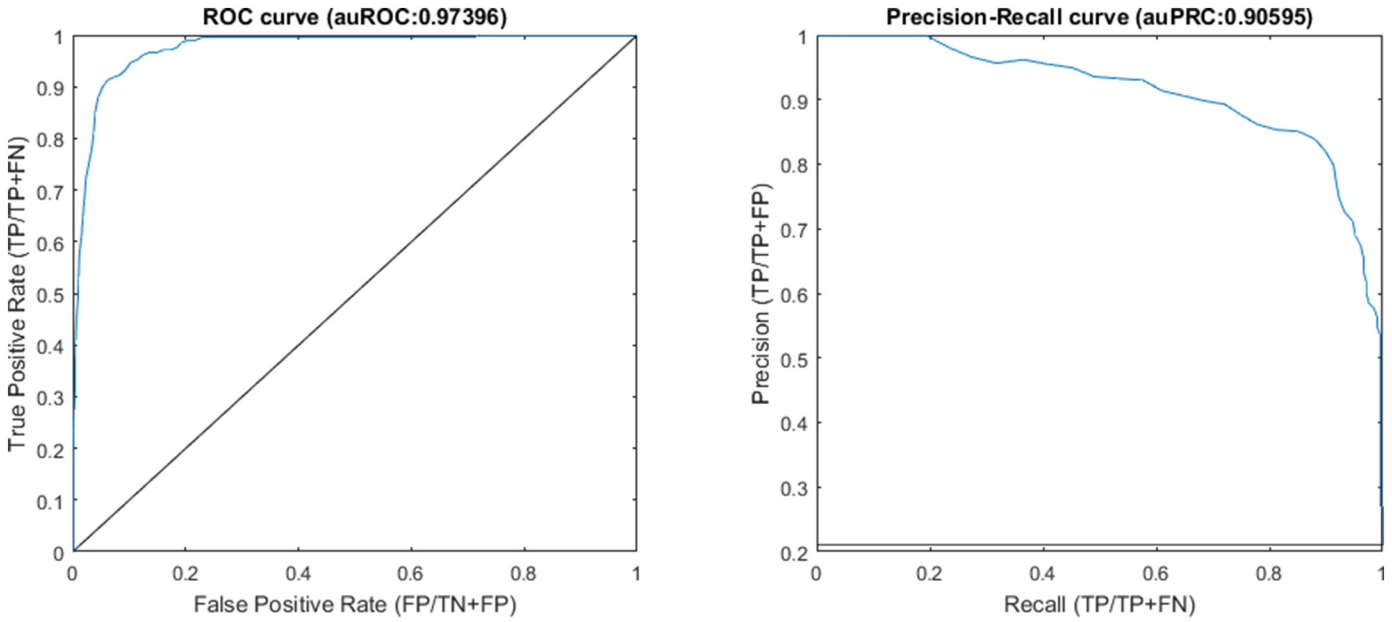

Fig. 6. Area under Receiver Operating Characteristic curve (auROC, measures the relative trade-offs between true positives and false positives) and area under Precision-Recall curve (auPRC, measures the fraction of negatives misclassified as positives).

The result is comparable to other research on NN269 benchmark dataset [4,22,23,24,34].

4. Discussion

Several computational methods have been proposed for the prediction of splice sites and proteins-coding regions, most of them are either probabilistic approach or machine learning based approach, or a combination of both [24]. In Huang et al. [20], the difference in the dinucleotide frequencies between true and false splice sites are used as features for the prediction of splice sites using SVM with RBF kernel. Baten et al. [4] generated features based on first order Markov model. These features are then used as input in SVM for splice site prediction using a polynomial kernel. Positional, compositional and dependency features were extracted for the true and false splice sites and were used as input in SVM classifiers [23]. EFFECT uses a two-stage process, where a set of candidate sequence-based features are constructed in the first stage and then the most effective subset is selected for the classification in the second stage [22]. Both stages make heavy use of evolutionary algorithms to efficiently guide the search towards informative features capable of discriminating true and false splice site sequences [22]. Snyder and Stormo [33] uses dynamic programming and neural networks for protein coding regions identification. Their program scores all subintervals in a sequence for content statistics indicative of introns and exons, and for sites that identify their boundaries. This information is weighted by a neural network combined with dynamic programming to get the optimal solution. In general, signal sensors are not sufficient to determine gene structure, thus it is necessary to combine them with content sensors to obtain satisfactory gene and protein-coding region prediction [31].

5. Conclusion

With the advancement of DNA sequencing, determining the exact sequence of bases in a DNA molecule is no longer a big challenge. As a result, databases of DNA and other biological sequences have been expanding exponentially, leading to the booming of the interdisciplinary field of bioinformatics [3,13,16].

Identification of protein coding regions and other regulatory regions in eukaryotic genomic sequences plays an important role in gene prediction and genome annotation. Even though being reliable, predicting protein coding regions via experimental methods is quite difficult to do on a large scale. Given the enormous amount of available training data, machine learning techniques have become essential in gene prediction and protein coding regions.

To understand the genome of an organism, it is also necessary to understand the mechanisms that regulate the expression of its genes. A major problem in gene prediction and genome annotation is to identify protein coding regions and splice sites in genes. The complex nature of genes makes it impractical to manually design algorithms to detect coding regions, as well as splice sites. Machine learning methods like neural networks provide a promising approach to this issue.

Feature generation and selection play an essential role in machine learning method's classification performance. The proposed method uses chaos game representation of DNA sequence as feature vector input to neural network. The novelty of the method is that, with the use of the one-to-one chaos game representation, feature inputs to the neural network indeed reflect the true nature of the original DNA segments. As a result, our method gives comparable predictive power to other methods in literature while using only one neural network component. It therefore can be used as a complementary method to the existing ones for the prediction of splice sites. Another search by content component for locating protein coding regions will be integrated into the neural network component in order to improve the method's accuracy. The method described may also be used for predicting other genomic elements in future research.

Acknowledgment

This research is supported by the National Natural Sciences Foundation of China (91746119), Tsinghua University Education Foundation fund (042202008), and Tsinghua University start-up fund (to Stephen S.-T. Yau). Our work is completed on the “Explorer 100” cluster system of Tsinghua HPC Platform.

References

[1] B. Alberts, Molecular Biology of the Cell, Garland science, 2017.
[2] V.B. Bajic, S.H. Seah, A. Chong, S. Krishnan, J.L. Koh, V. Brusic, Computer model for recognition of functional transcription start sites in RNA polymerase ii promoters of vertebrates, J. Mol. Graph. Model. 21 (5) (2003) 323–332.

[3] P. Baldi, S. Brunak, Bioinformatics: The Machine Learning Approach, MIT press, 2001.
[4] A.K. Baten, B.C. Chang, S.K. Halgamuge, J. Li, Splice site identification using probabilistic parameters and SVM classification, BMC Bioinformatics 7 (5) (2006) S15.
[5] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University press, 1995.
[6] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[7] S. Brunak, J. Engelbrecht, S. Knudsen, Prediction of human mrna donor and acceptor sites from the DNA sequence, J. Mol. Biol. 220 (1) (1991) 49–65.
[8] C. Burge, S. Karlin, Prediction of complete gene structures in human genomic DNA, J. Mol. Biol. 268 (1) (1997) 78–94.
[9] M. Burset, I. Seledtsov, V. Solovyev, Analysis of canonical and non-canonical splice sites in mammalian genomes, Nucleic Acids Res. 28 (21) (2000) 4364–4375.
[10] M.W. Craven, J.W. Shavlik, Machine learning approaches to gene recognition, IEEE Expert 9 (2) (1994) 2–10.
[11] J. Davis, M. Goadrich, The relationship between precision-recall and ROC curves, Proceedings of the 23rd International Conference on Machine Learning, ACM, 2006, pp. 233–240.
[12] P.J. Deschavanne, A. Giron, J. Vilain, G. Fagot, B. Fertil, Genomic signature: characterization and classification of species assessed by chaos game representation of sequences, Mol. Biol. Evol. 16 (10) (1999) 1391–1399.
[13] R. Durbin, S.R. Eddy, A. Krogh, G. Mitchison, Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University press, 1998.
[14] R. Farber, A. Lapedes, K. Sirotkin, Determination of eukaryotic protein coding regions using neural networks and information theory, J. Mol. Biol. 226 (2) (1992) 471–479.
[15] T. Fawcett, An introduction to ROC analysis, Pattern Recogn. Lett. 27 (8) (2006) 861–874.
[16] N. Goel, S. Singh, T.C. Aseri, A comparative analysis of soft computing techniques for gene prediction, Anal. Biochem. 438 (1) (2013) 14–21.
[17] S.S. Haykin, S.S. Haykin, S.S. Haykin, S.S. Haykin, Neural Networks and Learning Machines, 3 Pearson Upper Saddle River, NJ, USA, 2009.
[18] S.M. Hebsgaard, P.G. Korning, N. Tolstrup, J. Engelbrecht, P. Rouzé, S. Brunak, Splice site prediction in arabidopsis thaliana pre-mrna by combining local and global sequence information, Nucleic Acids Res. 24 (17) (1996) 3439–3452.
[19] T. Hoang, C. Yin, S.S.-T. Yau, Numerical encoding of DNA sequences by chaos game representation with application in similarity comparison, Genomics 108 (3) (2016) 134–142.
[20] J. Huang, T. Li, K. Chen, J. Wu, An approach of encoding for prediction of splice sites using svm, Biochimie 88 (7) (2006) 923–929.
[21] H.J. Jeffrey, Chaos game representation of gene structure, Nucleic Acids Res. 18 (8) (1990) 2163–2170.
[22] U. Kamath, K. De Jong, A. Shehu, Effective automated feature construction and selection for classification of biological sequences, PLoS One 9 (7) (2014) e99982.
[23] P.K. Meher, T.K. Sahu, A. Rao, S. Wahi, Identification of donor splice sites using support vector machine: a computational approach based on positional, compositional and dependency features, Algorithms for Molecular Biology 11 (1) (2016) 16.
[24] P.K. Meher, T.K. Sahu, A.R. Rao, S.D. Wahi, A statistical approach for 5 splice site prediction using short sequence motifs and without encoding sequence data, BMC Bioinformatics 15 (1) (2014) 362.
[25] C.E. Metz, Basic principles of roc analysis, Seminars in Nuclear Medicine, 8 Elsevier, 1978, pp. 283–298.
[26] M. Minsky, S. Papert, Perceptron: An Introduction to Computational Geometry, 19(88) MIT press, Cambridge, 1969, p. 2 expanded edition.
[27] K.P. Murphy, Machine Learning: A Probabilistic Perspective, MIT press, 2012.
[28] T. Naito, Human splice-site prediction with deep neural networks, J. Comput. Biol. 25 (8) (2018) 954–961.
[29] E. Pashaei, M. Ozen, N. Aydin, Random forest in splice site prediction of human genome, XIV Mediterranean Conference on Medical and Biological Engineering and Computing 2016, Springer, 2016, pp. 518–523.
[30] M.G. Reese, F.H. Eeckman, D. Kulp, D. Haussler, Improved splice site detection in genie, J. Comput. Biol. 4 (3) (1997) 311–323.
[31] S. Rogic, A.K. Mackworth, F.B. Ouellette, Evaluation of gene-finding programs on mammalian sequences, Genome Res. 11 (5) (2001) 817–832.
[32] F. Rosenblatt, Principles of Neurodynamics, Spartan, New York, 1962.
[33] E.E. Snyder, G.D. Stormo, Identification of protein coding regions in genomic DNA, J. Mol. Biol. 248 (1) (1995) 1–18.
[34] S. Sonnenburg, G. Schweikert, P. Philips, J. Behr, G. Rätsch, Accurate splice site prediction using support vector machines, BMC Bioinformatics 8 (10) (2007) S7.
[35] E.C. Uberbacher, R.J. Mural, Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach, Proc. Natl. Acad. Sci. 88 (24) (1991) 11261–11265.
[36] H. White, Artificial Neural Networks: Approximation and Learning Theory, Blackwell Publishers, Inc, 1992.
[37] C.H. Wu, Artificial neural networks for molecular sequence analysis, Comput. Chem. 21 (4) (1997) 237–256.