A Novel Artificial Neural Network and an Improved Particle Swarm Optimization used in Splice Site Prediction

Wei Bin and Zhao Jing

1Engineering University of CAPF, Xi’an 710086, China
2Xi’an Jiaotong University, Xi’an 710049, China

Abstract

The amount of DNA sequence data produced by several genomic projects had increased dramatically in recent years. One of the main goals of bioinformatics was to identify genes. A crucial part of the gene identification was to precisely detect the exon intron boundaries, i.e. the splice sites. This paper introduced a new type of artificial neural network (called NANN), which was designed specifically to solve the splice sites prediction problem. Moreover, the network connection weights of NANN were determined by an improved particle swarm optimization which was inspired by the wolves’ activities circle. In addition, three types of encoding approaches were applied to generate the input for the NANN. Intensive experiments were presented in this paper, and the results showed that our algorithm was better than some current methods, that is, the NANN_IPSO was applicable to splice site prediction problem.

Keywords: Particle swarm optimization; Neural network; Ensemble; Splice sites prediction

Introduction

With the completion of sequencing the genome from several eukaryotes [1], discovering the location and structure of genes (often referred to as gene prediction or gene finding) is an increasingly important task [2]. Eukaryotic genes are composed of two types of segments: exons and introns; while the exons are regions of the genetic material that are encoded proteins, on the other hand the introns are non-coding regions that are removed from the primary transcript [3]. The intron-exon boundary is referred to as acceptor splice site and the exon-intron boundary as donor splice site [4]. The donor and acceptor splice sites are the most critical signals for gene prediction [5].

In the past few decades, numerous methods have been proposed to detect the splice sites, such as hidden Markov model [5,6], Bayesian networks [7,8], artificial neural network (ANN) [9,10], support vector machine [11,12] and decision-trees [13]. However, due to complex dependencies existing among the bases around splice sites [7,14], the splice site prediction is still a difficult problem, i.e., splice site prediction is still a major bottleneck in gene finding [15]. Thus, development of new methods to accurately predict the splice sites is continuing to be expected.

Recently, the ANN has attracted wide spread attention [16-18], and it can be constructed without detailed domain knowledge [19]. However, defining the architecture of ANN is difficult. We propose a novel ANN (we call it NANN, hereafter) that incorporates into the model the codon, which is consecutive three bases that encode an amino acid.

Although ANN model has capability for extracting nonlinear relationships through training, the training process is difficult. Traditional training methods such as back propagation are very slow and possibly stuck at local minimum [20,21]. Recently, several researchers applied the PSO algorithm in training the neural network [22]. However, studies showed that similar to other population based methods, such as genetic algorithm, PSO may be trapped in local optimum and the convergence rate is slow in the later iterations [23-26]. Thus, mimicking the activities of wolves circle, we propose an IPSO, which aims to speed up the convergence and avoid getting into the local optimum, to train the NANN. Intensive experiments were presented in this paper, and the results showed that our algorithm was significantly better than some current methods.

Methods

A novel ensemble ANN

NANN: Studies showed that ANN with enough hidden layers (include number of nodes per layer) can achieve any function very well through an appropriate training. The NANN consists of four layers: an input layer, two hidden layers and an output layer. On the basis of fact that every three-base (codon) in exons stands for an amino acid (e.g. CAC→histidine), the connection between the input layer and the first hidden layer is defined as follows: every three-base in the region of exons is connected to one node in the first hidden layer, and other bases are one-to-one connected with the node in that layer. The other layers’ nodes are full-connected (Figure 1).

The input is a segment of nucleotide sequences with an uneven window size. The output of the networks comes from just one unit giving a value of 0 or 1, which is used to represent the category (splice site or non-splice site). The number of nodes in the first hidden layer for donor and acceptor splice sites prediction problem are defined in (1) and (2), respectively:

\[ N_{hid} = \frac{N_w}{3} + N_{in} + 2 \]  
\[ N_{hid} = \frac{N_w}{3} + N_{in} + 2 \]

*Corresponding author: Wei Bin, Engineering University of CAPF, Xi’an 710086, China, Tel: 86-13991387365; E-mail: weibin21@126.com

Received April 21, 2014; Accepted June 11, 2014; Published June 16, 2014

Copyright: © 2014 Bin W, et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.
Ensemble NANN: The theoretical and experimental results showed that combing multiple neural networks (training several individual ANNs and combining their outputs) can effectively improve the performance of ANN [27,28]. Therefore, in this paper, several individual NANNs are set to 2.2 and \( c \) is random numbers between 0 and 1, and \( x_i \) is the number of inputs in upstream, \( x_d \) is the number of inputs in downstream.

The final decision is made by combining ensemble of NANN with unweighted majority voting.

IPSO: Biologists studied on wolves and found that each pack of wolves has a 15 km radius of activities circle. Miniaturizing three wolves’ activities circles to the drawings, they found that the circles are crossing, neither isolated nor completely blending. The intersection provides the possibility of hybrid, and the other sections make them retain some of personality. When the activities circle overlap, wolves are killing; conversely, complete isolation will bring degradation [31]. Inspired by the above phenomenon, we propose an improved PSO named IPSO.

In IPSO, the swarm consists of three sub-swarms \( A, B \) and \( C \). At each iteration, \( n \) particles are randomly selected from each sub-swarm to constitute the set \( D \), where \( n = \frac{|D|}{k} \), \( T_i \{ A, B, C \} \), \( D \) is the number of particles in \( D \). \( D \) represents the intersection of \( A, B \) and \( C \) which ensures the information sharing. In addition, three enhancing strategies are used in this paper including time-varying acceleration coefficients strategy, crossover and mutation.

**a) Time-varying acceleration coefficients strategy**

The coefficients change with iteration so as to speed up the convergence during the earlier iterations and keep the diversity during the later iterations. The social coefficient \( c_i \) is linearly decreased during the iterations; inversely, the cognitive coefficient \( c_i \) is increased linearly.

\[
\begin{align*}
    c_1 & = (c_{max} - c_{min}) \times \frac{iter_{cu}}{iter_{max}} + c_{min} \\
    c_2 & = (c_{max} - c_{min}) \times \frac{iter_{max} - iter_{cu}}{iter_{max}} + c_{min}
\end{align*}
\]

where \( iter_{cu} \) and \( iter_{max} \) are the current and maximal iteration, respectively, and \( c_{max} \) is the upper bound, and \( c_{min} \) is the lower bound. In this paper, the values of \( c_{max} \) and \( c_{min} \) are set to 2.2 and 1.8, respectively.

**b) Crossover**

In PSO, the particle swarm has “memory” of past successes, and tends to converge upon the regions of search space that have afforded success previously. However, there is no mechanism for leaps from one region to another, and crossover allows that leaps [32]. Therefore, the following mechanism is used in this paper to prevent the algorithm from trapping in local optimization. If \( rand() < p_c \), the following operation is executed.

![Figure 1: The schema of the NANN architecture.](image1.png)

![Figure 2: The schema of the ensemble NANN.](image2.png)
where \( p_c \) is the crossover probability, \( j_{rand} \) is randomly chosen between 1 to \( M \) with equal probabilities.

c) Mutation

Lack of the diversity, particularly during the later stages of the optimization, is the dominant factor of the convergence to local optimum [33]. Hence, the concept of “mutation” is adopted to enhance the global search capability.

\[
x_{ij}^{new}(t+1) = \begin{cases} 
\text{rand}(x_{\text{min}}, x_{\text{max}}), & \text{if } \text{rand()} < p_m \\
x_j(t), & \text{otherwise}
\end{cases}
\]

(8)

where \( p_m \) is the mutation probability.

Based on the above analysis, the main steps of the IPSO are given as follows:

1. **Step 1: Initializing.** Generate three independent swarms \( A, B \) and \( C \) (each one include \( M \) particles) randomly.

2. **Step 2: Calculating.** Calculate the fitness value of each particle.

3. **Step 3: Updating.** \( A, B \) and \( C \) are updated independently according to the equations (3), (4) and (5)

4. **Step 4: Selecting.** \( N \) particles are randomly selected from \( A \) and \( C \) to constitute the set \( D \).

5. **Step 4.1: Crossover.** The particles in \( D \) are crossed according to (6) and (7).

6. **Step 4.2: Mutating.** If \( \text{rand()} < p_m \), (8) is executed.

7. **Step 5: Checking.** If the termination criteria are satisfied, stop. Otherwise, go to Step 2.

### Hybrid NANN and IPSO

Although the NANN model has been reported in the previous section, the problem of training the network connection weight is not a trivial problem. It is a nonlinear and dynamic process in that any change of one weight requires adjustment of many others. In this paper, the IPSO is applied to optimize the weights (Figure 3). The fitness function is defined as the rate of misclassification.

### Encoding Method

In most of the previous studies, the nucleotide encoding was often ignored. In this paper, three encoding methods are used: single-nucleotide (SN) encoding, SN with frequency difference between the true and false sets (SN_FD) [34], and SN_FD with distribution information (SN_FD_DI).

#### SN encoding

Each nucleotide is given an integer number: A (-1), T (1), G (-2) and C (2).

#### SN_FD encoding

A position weight matrix is derived from the true set by counting the frequency which each nucleotide occurs at each position.

\[
M_y = \frac{1}{n} \sum_{i=1}^{n} O_i(N_y), i = -2, -1, 1, 2, \ y = 1, 2, \ldots, l
\]

(9)

\[
O_i(x) = \begin{cases} 
1, & i = x \\
0, & \text{else}
\end{cases}
\]

where \( n \) is the number of samples in the true set, \( l \) is the number of nucleotide in each sample and \( N_y \in \{-2, -1, 1, 2\} \). In the same way, a position weight matrix can be obtained for the false set. The SN with FD encoding is obtained by subtracting the true coding matrix from the false one [34].

#### SN_FD with distribution information encoding

However, SN_FD method has its limits. For example, A’s frequency is 0.8 in true set and 0.7 in false set. C’s frequency is 0.2 in true set and 0.1 in false set. When the SN_FD is used, the two features have the same value. However, the contribution of the two bases may be different. Therefore, we propose a novel encoding method that the distribution information (DI) is added to (9).

\[
M_y' = \log(p_j(N_y \mid c_i)) \times \frac{1}{n} \sum_{i=1}^{n} O_i(N_y)
\]

(10)

where \( p_j(N_y \mid c_i) \) is the frequency of \( N_y \) in the class \( c_i \), \( c_i \in \{true, false\} \).

### Experimental Results

In this section, we tested IPSO on four benchmark functions, traveling salesman problem and bin packing problem. Then, NANN_IPSO was tested on the Homo Sapiens Splice Sites Dataset (HS3D), and the results were compared with those of several current known algorithms.

#### Tested the effect of IPSO

**Experiment for benchmark testing:** Four well-known benchmark functions were used to evaluate the effectiveness of IPSO, and the results were compared with those of several current known algorithms.

**Experimental Results:** In this section, we tested IPSO on four benchmark functions, traveling salesman problem and bin packing problem. Then, NANN_IPSO was tested on the Homo Sapiens Splice Sites Dataset (HS3D), and the results were compared with those of several current known algorithms.
Each experiment was conducted 30 times, and the average results and the standard deviation were presented. Based on the parameter sensitive analysis (see 4.2.4), we set the parameters of IPSO as follows: \( p_a = 0.01, p_r = 0.8 \) and \( k = 5 \). Table 2 shows the results of the four algorithms. In the table, “I” indicates the function, and “D” indicates the dimension of function.

It can be seen from the table that all the methods performed well on functions 2 and 4. However, for the 1 and 3, IPSO improved the average solutions significantly compared with those of others. Therefore, we can reach the conclusion that IPSO outperforms the other three algorithms.

**Experiment for bin packing problem testing:** The bin packing problem (BPP) is a NP-hard combinatorial optimization problem where the aim is to pack a finite number of items using the least bins possible [37]. The problem can be stated as follows: given a finite set of numbers (the items) and a constant C (the bin’s capacity), find the packing pattern that requires the minimum number of bins. The m items were randomly drawn from the range (0,1) to be packed into bins of capacity 1. In this paper, three sets of parameters were used: (1) \( m = 100 \); (2) \( m = 500 \); (3) \( m = 1000 \). Table 3 shows the results of the four algorithms (PSO, IBPSO [38], MPSO [39] and IPSO). According to Table 3, the IBPSO was a very poor method for the BPP. On the other hand, the IPSO considerably outperformed others, finding the better results obtained by our algorithm were better than the mean results tested on four instances. According to Table 4, it can be seen that IPSO obtained the best results in all instances. From the table, even the worst results obtained by our algorithm were better than the mean results obtained by others in some of cases. According to standard deviations, the IPSO is more stable.

**Experiment for traveling salesman problem testing:** Traveling salesman problem (TSP) is one of the most widely studied combinatorial optimization problems in which a salesman has to visit every city in the shortest way [40]. The TSP instances were derived from the TSPLIB [41]. The three versions of PSO (PSO, PSO_c3dyn [42] and IPSO) were tested on four instances. According to Table 4, it can be seen that IPSO obtained the best results in all instances. From the table, even the worst results obtained by our algorithm were better than the mean results obtained by others in some of cases. According to standard deviations, the IPSO is more stable.

**Experiment for splice site prediction**

**Dataset:** The dataset was extracted from GenBank Rel.123. HS3D contains 2796 donor sites and 2880 acceptor sites. In addition, there are 271937 false donor sites and 329374 false acceptor splice sites, which were selected by searching GT and AG dinucleotides in non-splicing positions [43].

**Evaluation criteria:** The evaluation criteria used in this paper are sensitivity (Sn), specificity (Sp), accuracy (Acc) and \( Q^2 \).

\[
Sn = \frac{TP}{TP + FN}
\]

(11)

\[
Sp = \frac{TN}{TN + FP}
\]

(12)

\[
Acc = \frac{TP + TN}{TP + TN + FP + FN}
\]

(13)

\[
q^a = \sqrt{\frac{FN}{TP + FN} + \frac{FP}{TN + FP}}
\]

(14)

\[
Q^2 = \frac{1 + q^a}{2}
\]

(15)

True negative (TN): number of false splice sites that were correctly classified as false.
True positive (TP): number of true splice sites that were correctly classified as true.

False negative (FN): number of true splice sites that were wrongly classified as false.

False positive (FP): number of false splice sites that were wrongly classified as true.

In addition, the k-fold method was employed in the experiments, with the value of k set to five. For five-fold cross-validation, the whole dataset was divided into five subsets with approximately equal size. Then, the classifier was trained five times—each time, one subset was used as the testing data to validate the classifier.

### Results

Firstly, an individual NANN with different encoding methods and different inputs were used to analyze the effects of three encoding methods, and the results are showed in Figure 4 and Figure 5, where $LU$ is the length of upstream sequences of splice sites and $LD$ is the length of downstream sequences of splice sites. From the figures we can see that the SN obtained the worst Acc for both the donor and acceptor datasets. On the other hand, the SN_FD_DI obtained the best results compared with other two encoding methods. Therefore, it can be concluded that the SN_FD_DI encoding method can more accurately reflect the difference between true and false sites.

Secondly, the effects of the number of individual ($N$) NANN(s) used in the ensemble one were analyzed (Tables 5 and 6). It is clear that the results were greatly influenced by the value of $N$, and the best results were obtained when $N = 5$. Figures 6 and 7 shows the frequency of nucleotides composition bias. From figure we can see that: a) the adjacent sequences of splice sites have remarkable conservativeness; b) introns show more remarkable conservativeness than exons.

Finally, our method was compared with MM1-SVM [11], SVM-B [44] and HMM [5]. The results are showed in Table 7. We can see that all the algorithms generated better results for the donor dataset than the acceptor one, and the reason is that the donor set is more conservative than the acceptor one. Moreover, our method outperformed the other algorithms in the evaluation criteria $Q^*$ on all the datasets, and it can be explained by the effectiveness (the information used to construct NANN were usefulness) of the algorithm proposed in this paper.

### Sensitivity in relation to parameters

To study the effects of parameters, we solved the splice site prediction problem by NANN_IPSO with various $k$, $p_c$, and $p_m$.

- **Sensitivity in relation to the $k$**

  The $k$ is an important parameter for our algorithm. Tables 8 and 9 show the effects of $k$ on the solutions where the mutation probability and crossover probability were 0.01 and 0.8, respectively. From the table we can see that our algorithm achieved the best results when the $k$ equaled 5. This can be explained as follows: the small value leads to overlaps among the three groups, and then the particles become over-

![Figure 4: The performance of NANN with different encoding methods on donor site: (a) SN for donor, (b) SN_FD for donor, (c) SN_FD_DI for donor.](Image)

![Figure 5: The performance of NANN with different encoding methods on acceptor site: (a) SN for acceptor, (b) SN_FD for acceptor, (c) SN_FD_DI for acceptor.](Image)
competitive, and the swarms degrade in the end. In contrast, the large value makes it cannot communicate with others.

- **Sensitivity in relation to crossover probability**

Then, the effects of crossover probability on the results were investigated. Crossover probability plays a key role in the method. For small values, the algorithm will converge slowly. However, large values will lead the particles to a random walk in the search space. Therefore, a proper value is vital for the method. The effects of the crossover probability on the results are shown in Tables 10 and 11 where $k$ and mutation probability were set to 5 and 0.01, respectively. From the tables we can see that the algorithm achieved the best results when the crossover probability was 0.8.

**Sensitivity in relation to mutation probability**

Tables 12 and 13 show the effects of the mutation probability on the solutions where the $k$ and the crossover probability were 5 and 0.8, respectively. Variations of results were observed with different mutation probabilities. From the tables we can see that our method achieved the best results when the mutation probability equaled 0.01.

In summary, the computational results confirmed that the method used in this paper works well. Its performance was fairly robust, and this is a very appealing feature, as most real world applications involve analyzing huge volumes of genome sequence data. Moreover, the five-
fold cross-validation experiment was reliable and appropriate for splice site prediction.

Conclusions

Predicting splice sites is an important part of gene prediction. Due to complex dependencies existing among bases around splice sites, splice site prediction remains a major bottleneck in gene prediction. A novel neural network model (NANN) was introduced, which was based on the fact that three-base code words (codons) in mRNA stand for amino acids in proteins, to predict the splice sites. In addition, the IPSO mimicking the activities of wolves circle was used as a training phase of NANN to determine the weights of network. Moreover, three encoding methods were applied and showed how the predicting performance was benefited from the use of the SN_FD_DI encoding approach. The experimental results demonstrated that the proposed algorithm was able to discriminate between the true and false splice sites and had better prediction accuracy than others.

Acknowledgements

This study was supported by the National Natural Science Foundation of China (Grant No. 61309022).

References


Table 10: Variation of the results obtained by NANN_IPSO with different crossover probability (donor site).

<table>
<thead>
<tr>
<th>p_c</th>
<th>Sn</th>
<th>Sp</th>
<th>Acc</th>
<th>Q^9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>86.70%</td>
<td>98.50%</td>
<td>92.60%</td>
<td>90.53%</td>
</tr>
<tr>
<td>0.4</td>
<td>88.41%</td>
<td>97.81%</td>
<td>93.56%</td>
<td>91.76%</td>
</tr>
<tr>
<td>0.6</td>
<td>88.20%</td>
<td>99.36%</td>
<td>93.78%</td>
<td>91.64%</td>
</tr>
<tr>
<td>0.8</td>
<td>90.77%</td>
<td>98.93%</td>
<td>94.85%</td>
<td>93.43%</td>
</tr>
</tbody>
</table>

Table 11: Variation of the results obtained by NANN_IPSO with different crossover probability (acceptor site).

<table>
<thead>
<tr>
<th>p_c</th>
<th>Sn</th>
<th>Sp</th>
<th>Acc</th>
<th>Q^9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>82.50%</td>
<td>97.92%</td>
<td>90.21%</td>
<td>87.54%</td>
</tr>
<tr>
<td>0.4</td>
<td>84.79%</td>
<td>98.33%</td>
<td>91.56%</td>
<td>89.18%</td>
</tr>
<tr>
<td>0.6</td>
<td>85.00%</td>
<td>98.13%</td>
<td>91.56%</td>
<td>89.31%</td>
</tr>
<tr>
<td>0.8</td>
<td>85.63%</td>
<td>97.62%</td>
<td>91.77%</td>
<td>89.73%</td>
</tr>
</tbody>
</table>

Table 12: Variation of the results obtained by NANN_IPSO with different mutation probability (donor site).

<table>
<thead>
<tr>
<th>p_m</th>
<th>Sn</th>
<th>Sp</th>
<th>Acc</th>
<th>Q^9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>90.77%</td>
<td>98.93%</td>
<td>94.85%</td>
<td>93.43%</td>
</tr>
<tr>
<td>0.05</td>
<td>87.12%</td>
<td>99.14%</td>
<td>93.13%</td>
<td>90.88%</td>
</tr>
<tr>
<td>0.1</td>
<td>84.76%</td>
<td>99.14%</td>
<td>91.95%</td>
<td>89.12%</td>
</tr>
<tr>
<td>0.2</td>
<td>83.91%</td>
<td>98.93%</td>
<td>91.42%</td>
<td>88.59%</td>
</tr>
</tbody>
</table>

Table 13: Variation of the results obtained by NANN_IPSO with different mutation probability (acceptor site).

<table>
<thead>
<tr>
<th>p_m</th>
<th>Sn</th>
<th>Sp</th>
<th>Acc</th>
<th>Q^9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>85.63%</td>
<td>97.62%</td>
<td>91.77%</td>
<td>89.73%</td>
</tr>
<tr>
<td>0.05</td>
<td>81.88%</td>
<td>98.75%</td>
<td>90.31%</td>
<td>87.15%</td>
</tr>
<tr>
<td>0.1</td>
<td>80.00%</td>
<td>98.33%</td>
<td>89.17%</td>
<td>85.81%</td>
</tr>
<tr>
<td>0.2</td>
<td>78.75%</td>
<td>98.96%</td>
<td>88.85%</td>
<td>84.96%</td>
</tr>
</tbody>
</table>

Figure 7: The frequency of nucleotides composition bias of acceptor site: (a) true acceptor, (b) false acceptor.


