Journal of
Applied & Computational Mathematics

Bin and Jing, J Appl Computat Math 2014, 3:4
DOI: 10.4172/2168-9679.1000166

Research Article Open Access

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

Wei Bin[1]* and Zhao Jing[2,3]

[1]Engineering University of CAPF, Xi'an 710086, China
[2]Xijing University, Xi'an 710123, China
[3]Xi'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.

$$N_{h1d} = \frac{N_{lu}}{3} + N_{ld} + 2 \quad (1)$$

$$N_{h1a} = \frac{N_{ld}}{3} + N_{lu} + 2 \quad (2)$$

***Corresponding author:** Wei Bin, Engineering University of CAPF, Xi'an 710086, China, Tel: 86-13991387365; E-mail: weibin82@126.com

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

Citation: Bin W, Jing Z (2014) A Novel Artificial Neural Network and an Improved Particle Swarm Optimization used in Splice Site Prediction. J Appl Computat Math 3: 166 doi:10.4172/2168-9679.1000166

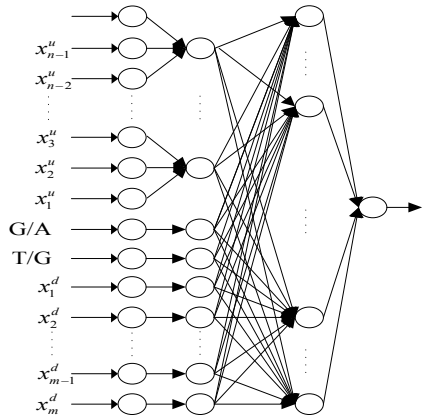

Figure 1: The schema of the NANN architecture.

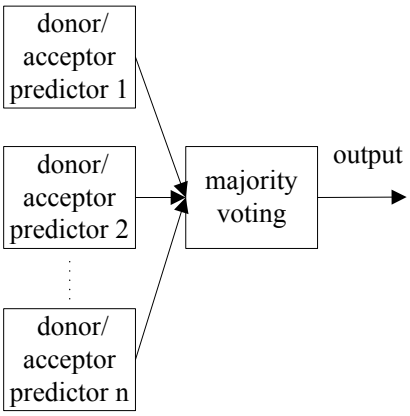

Figure 2: The schema of the ensemble NANN.

where N_{lu} is the number of inputs in upstream, N_{ld} is the number of inputs in downstream.

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 used as the component of the ensemble one (Figure 2). The main steps of the ensemble NANN are given as follows:

1) Several NANNs are trained with various window size (input) and second hidden layer's nodes.

N of the better model is selected.

Create an ensemble NANN consisting of N individual NANNs according to the step 2.

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

IPSO

Particle swarm optimization: PSO is an iterative optimization algorithm inspired by the observation of collective behaviors in animals (e.g., bird flocking) [29,30]. In PSO, each candidate solution to an optimization problem is represented by one particle. Each particle i is described by its position x_i and velocity v_i. The algorithm starts with random initialization of the particles. Then, the particles change their positions according to their velocities, which are updated in each iteration. Given that p_i is the best position found by particle i in all the preceding iterations and p_g is the best position found so far by the entire swarm, the velocity and position of particle i in bit j will be updated according to the following formulae:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_1 (p_{ij}(t) - x_{ij}(t)) + c_2 r_2 (p_{gj}(t) - x_{ij}(t)) \quad (3)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (4)$$

where r_1 and r_2 are random numbers between 0 and 1, and c_1 and c_2 define the degree of influence of p_i and p_g on the particle's velocity. The velocity v_{ij} is bounded within a range of $[-V_{\max}, V_{\max}]$ to prevent the particle from flying out of the solution space.

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 $n = \frac{|T_i|}{k}, T_i \in \{A, B, C\}$, $|\bullet|$ is the number of particles in $\bullet$. 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_2 is linearly decreased during the iterations; inversely, the cognitive coefficient c_1 is increased linearly.

$$\begin{cases} c_1 = (c_{\max} - c_{\min}) * \dfrac{iter_{cru}}{iter_{\max}} + c_{\min} \\ c_2 = (c_{\max} - c_{\min}) * \dfrac{iter_{\max} - iter_{cru}}{iter_{\max}} + c_{\min} \end{cases} \quad (5)$$

where $iter_{cru}$ 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 the $rand() < p_c$, the following operation is executed.

$$x_{i_1 j}^{new}(t+1)=\begin{cases}x_{i_1 j}(t), & if\ j \le j_{rand}\\ x_{i_2 j}(t), & otherwise\end{cases} \quad (6)$$

$$x_{i_2 j}^{new}(t+1)=\begin{cases}x_{i_2 j}(t), & if\ j \le j_{rand}\\ x_{i_1 j}(t), & otherwise\end{cases} \quad (7)$$

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.

$$x_{ij}(t+1)=\begin{cases}rand(x_{\min}, x_{\max}), & if\ \ rand() < p_m\\ x_{ij}(t+1), & otherwise\end{cases} \quad (8)$$

where p_m is the mutation probability.

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

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

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

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

Step 4: Selecting. n particles are randomly selected from *A* and *C* to constitute the set D.

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

Step 4.2: Mutating. If $rand() < p_m$, (8) is executed.

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).

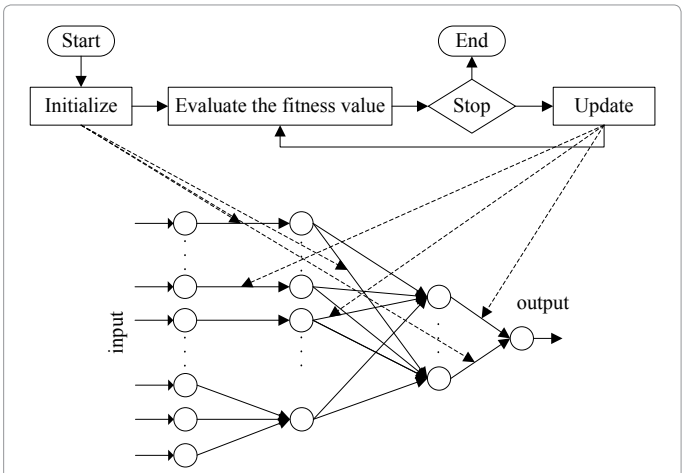

Figure 3: The hybrid of IPSO and NANN.

SN with FD encoding

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

$$M_{ij}=\frac{1}{n}\sum_{t=1}^{n}O_i(N_{tj}), i=-2,-1,1,2;\ j=1,2,\cdots,l \quad (9)$$

$$O_i(x)=\begin{cases}1, i=x\\ 0, else\end{cases}$$

where n is the number of samples in the true set, l is the number of nucleotide in each sample and $N_{tj} \in \{-2,-1,1,2\}$. 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).

$$M_{ij}^{*}=\log(p_j(N_{tj}\mid c_t))\times\frac{1}{n}\sum_{t=1}^{n}O_i(N_{tj}) \quad (10)$$

where $p_j(N_{tj}\mid c_t)$ is the frequency of N_{tj} in the class c_t, $c_t \in \{true, false\}$.

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 PSO, EGPSO and MPSO_SCSS [31,35,36]. All the functions used in this paper are defined in Table 1.

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_m =0.01, p_c =0.8 and k =5. Table 2 shows the results of the four algorithms. In the table, "F" 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 packing even for the difficult instance where m=1000. The results on the three datasets confirm the superiority of the IPSO in comparison with others.

Experiment for traveling salesman problem testing: Traveling salesman problem (TSP) is one of most widely studied combinational 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.

	Test functions	Search space	Global optimum
F1	x_i^2	(-5.12,5.12)	0
F2	$100(x_1^2 - x_2^2)^2 + (1 - x_1)^2$	(-2.048,2.048)	0
F3	$\sum_{i=1}^{n} i \cdot x_i^4 + random[0,1)$	(-1.28,1.28)	0
F4	$0.5 + \frac{\sin^2\sqrt{x^2+y^2} - 0.5}{(1.0 + 0.001(x^2+y^2))^2}$	(-100,100)	0

Table 1: Details of benchmark functions.

F	D	Average/(Standard Deviation)			
		PSO	EGPSO	MPSO_SCSS	IPSO
F1	5	2.6177*10^{-115} (2.8075*10^{-115})	2.1337*10^{-68} (3.7709*10^{-68})	2.4057*10^{-30} (4.4985*10^{-30})	1.8713*10^{-169} (0)
	10	5.6843*10^{-15} (1.1874*10^{-14})	7.1827*10^{-11} (1.5699*10^{-10})	1.4352*10^{-18} (1.3143*10^{-18})	6.9166*10^{-35} (1.6381*10^{-34})
F2	2	0 (0)	0 (0)	0 (0)	0 (0)
F3	5	2.7501*10^{-4} (1.7581*10^{-4})	5.3997*10^{-4} (3.0793*10^{-4})	5.0818*10^{-4} (2.5073*10^{-4})	2.9644*10^{-323} (0)
	10	1.3445*10^{-2} (8.0537*10^{-3})	2.1525*10^{-2} (1.1245*10^{-2})	4.0352*10^{-3} (1.8352*10^{-3})	1.1449*10^{-61} (3.5054*10^{-61})
F4	2	0 (0)	6.5451*10^{-3} (4.7254*10^{-3})	0 (0)	0 (0)

Table 2: Average and the standard deviation of the optimal values obtained by PSO, EGPSO, MPSO_SCSS and IPSO.

Dataset	Average/(Standard Deviation)			
	PSO	IBPSO	MPSO	IPSO
VR100	36 (0)	36.6 (0.5040)	36 (0)	35.1 (0.1568)
VR500	178.8 (0.6644)	180.0 (1.1592)	178.8 (0.6477)	176.2 (0.3488)
VR1000	406 (126.2)	416 (74.3)	409 (62.1)	397 (36.8)

Table 3: The results of the bin packing problem for 30 trials.

Dataset (best known)	Average/(Standard Deviation)		
	PSO	PSO_c3dyn	IPSO
Eil51 (426)	441.5 (3.62)	457.9 (4.17)	434.4 (2.68)
Berlin52 (7542)	7657.4 (144.84)	8062.2 (68.37)	7548.4 (13.58)
St70 (675)	685.8 (6.94)	700.6 (5.09)	674.8 (6.05)
Pr76 (108159)	117390.6 (1251.30)	119550.3 (933.02)	108852.0 (232.21)

Table 4: The results of the Traveling Salesman Problem for 30 trials.

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^9.

$$Sn = \frac{TP}{TP + FN} \tag{11}$$

$$Sp = \frac{TN}{TN + FP} \tag{12}$$

$$Acc = \frac{TP + TN}{TP + FP + FN + TN} \tag{13}$$

$$q^9 = \begin{cases} \frac{TN - FP}{TN + FP} & if \quad TP + FN = 0 \\ \frac{TP - FN}{TP + FN} & if \quad TN + FP = 0 \\ 1 - \sqrt{2[(\frac{FN}{TP+FN})^2 + (\frac{FP}{TN+FP})^2]} & if \quad TP + FN \neq 0 \\ & and \quad TN + FP \neq 0 \end{cases} \tag{14}$$

$$Q^9 = \frac{1 + q^9}{2} \tag{15}$$

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 which used as the inputs of NANN_IPSO. Y-axis indicates 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^9 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 variousk, 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-

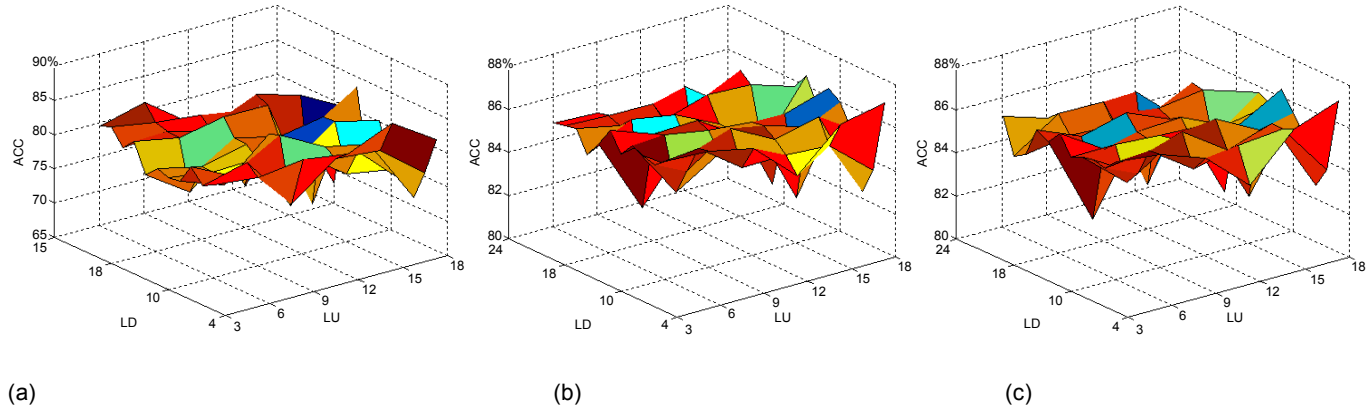

Figure 4: The performance of NANN with different encoding methods on donor site: (a) SN for donor, (b) SN_FD for donor, (c) SN_FD_DI for donor.

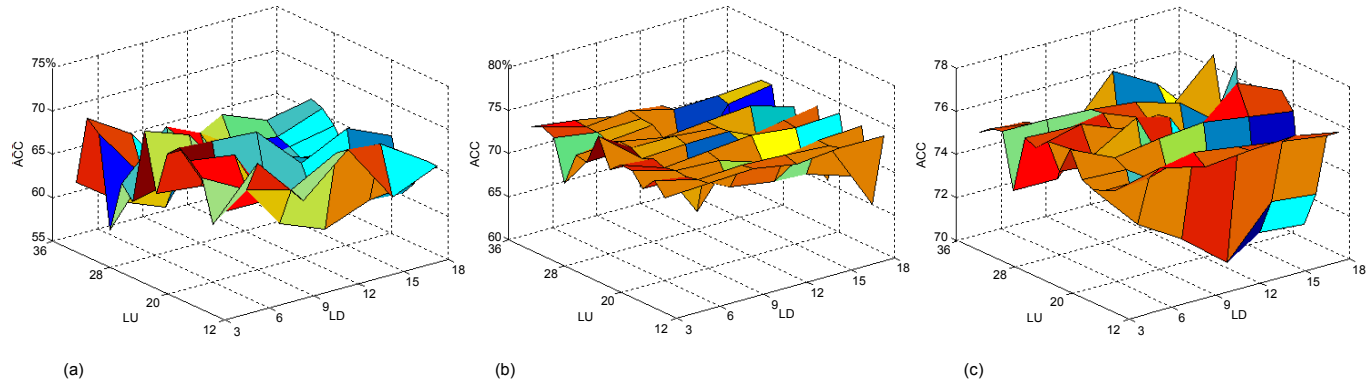

Figure 5: The performance of NANN with different encoding methods on acceptor site: (a) SN for acceptor, (b) SN_FD for acceptor, (c) SN_FD_DI for acceptor.

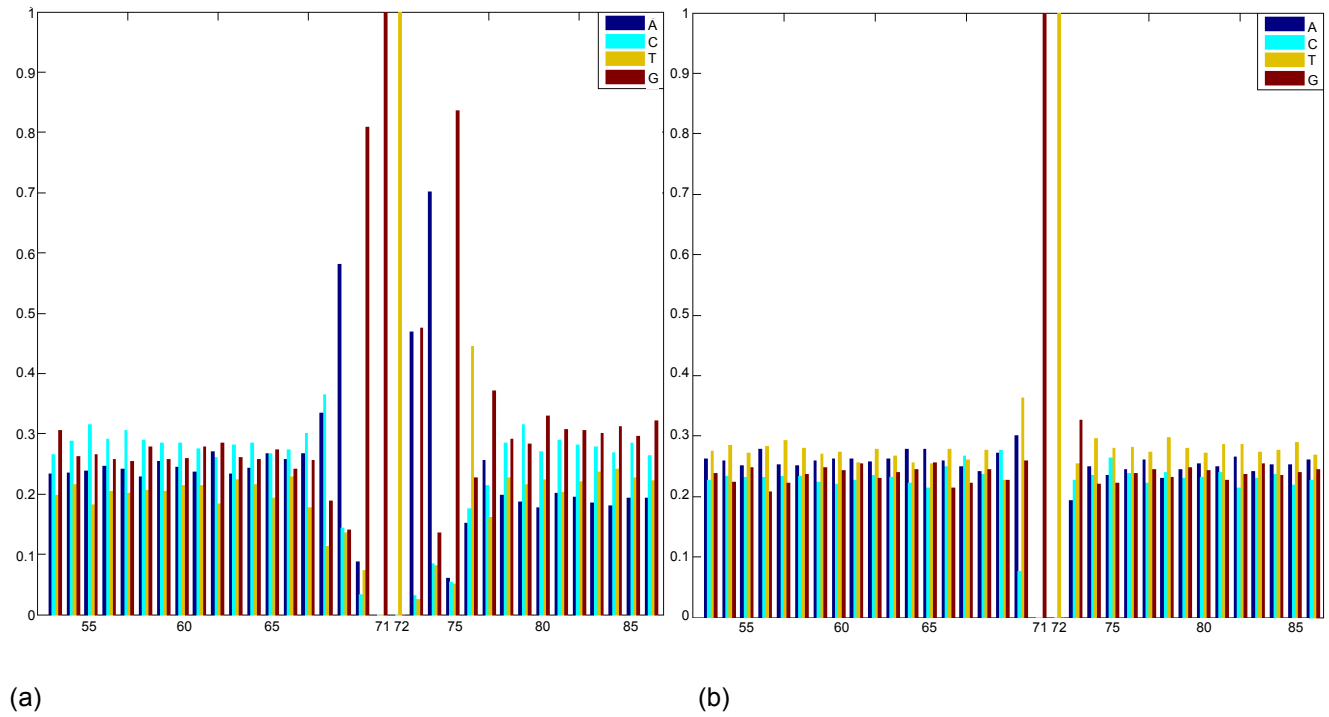

Figure 6: The frequency of nucleotides composition bias of donor site: (a) true donor, (b) false donor.

N	*Sn*	*Sp*	*Acc*	Q^9
1	76.18%	98.28%	87.23%	83.11%
3	86.91%	98.93%	92.92%	90.71%
5	90.77%	98.93%	94.85%	93.43%
7	85.62%	99.14%	92.38%	89.82%
9	84.12%	98.71%	91.42%	88.73%

Table 5: Donor sites prediction with different *N* NANN(s).

N	*Sn*	*Sp*	*Acc*	Q^9
1	57.50%	97.08%	77.29%	69.88%
3	77.92%	97.71%	87.81%	84.30%
5	85.63%	97.62%	91.77%	89.73%
7	75.42%	98.13%	86.77%	82.57%
9	74.17%	97.50%	58.83%	81.65%

Table 6: Acceptor sites prediction with different *N* NANN(s).

Method	donor			acceptor		
	Sn	*Sp*	Q^9	*Sn*	*Sp*	Q^9
MM1-SVM	93.06%	91.31%	92.10%	90.24%	87.57%	88.79%
SVM-B	94.31%	90.99%	92.38%	90.90%	88.16%	89.37%
HMM	94.24%	92.42%	93.23%	88.23%	90.11%	89.40%
Our method	90.77%	98.93%	93.43%	85.63%	97.62%	89.73%

Table 7: Prediction accuracies obtained by three algorithms.

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.

k	*Sn*	*Sp*	*Acc*	Q^9
20	85.41%	98.93%	92.17%	89.65%
10	87.34%	99.79%	93.56%	91.05%
5	90.77%	98.93%	94.85%	93.43%
2	87.55%	98.50%	93.03%	91.14%

Table 8: Variation of the results obtained by NANN_IPSO with different k (donor site).

k	*Sn*	*Sp*	*Acc*	Q^9
20	83.33%	97.56%	90.63%	88.12%
10	84.17%	97.92%	91.04%	88.71%
5	85.63%	97.62%	91.77%	89.73%
2	83.75%	97.92%	90.83%	88.42%

Table 9: Variation of the results obtained by NANN_IPSO with different k (acceptor site).

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-

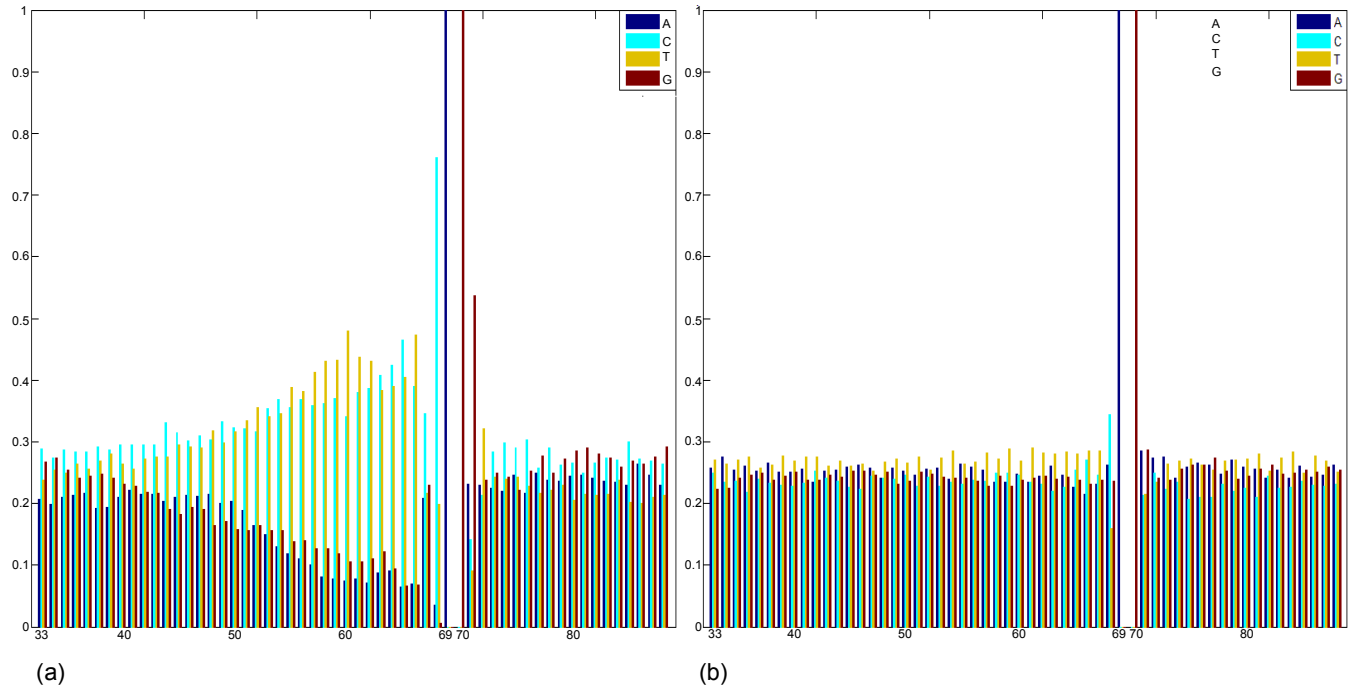

Figure 7: The frequency of nucleotides composition bias of acceptor site: (a) true acceptor, (b) false acceptor.

p_c	Sn	Sp	Acc	Q^9
0.2	86.70%	98.50%	92.60%	90.53%
0.4	88.41%	98.71%	93.56%	91.76%
0.6	88.20%	99.36%	93.78%	91.64%
0.8	90.77%	98.93%	94.85%	93.43%

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

p_c	Sn	Sp	Acc	Q^9
0.2	82.50%	97.92%	90.21%	87.54%
0.4	84.79%	98.33%	91.56%	89.18%
0.6	85.00%	98.13%	91.56%	89.31%
0.8	85.63%	97.62%	91.77%	89.73%

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

p_m	Sn	Sp	Acc	Q^9
0.01	90.77%	98.93%	94.85%	93.43%
0.05	87.12%	99.14%	93.13%	90.88%
0.1	84.76%	99.14%	91.95%	89.12%
0.2	83.91%	98.93%	91.42%	88.59%

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

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.

p_m	Sn	Sp	Acc	Q^9
0.01	85.63%	97.62%	91.77%	89.73%
0.05	81.88%	98.75%	90.31%	87.15%
0.1	80.00%	98.33%	89.17%	85.81%
0.2	78.75%	98.96%	88.85%	84.96%

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

Acknowledgements

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

References

1. Saeys Y, Degroeve S, Aeyels D, Van De Peer Y, Rouzé P (2003) Fast feature selection using a simple estimation of distribution algorithm: a case study on splice site prediction. Bioinformatics 19: ii179-ii188.
2. Dogan RI, Getoor L, Wilbur WJ, Mount SM (2007) Features generated for computational splice-site prediction correspond to functional elements. BMC Bioinformatics 8: 410.
3. Tavares LG, Lopes HS, Lima CRE (2009) Evaluation of weight matrix models in the splice junction recognition problem. IEEE International Conference on Bioinformatics and Biomedicine Workshop, BIBMW 14-19.
4. Guigo R, Flicek P, Abril JF, Reymond A, JulienLagarde, et al. (2006) EGASP: the human ENCODE Genome Annotation Assessment Project. Genome Biol 1: S2.1-31.

5. Quanwei Z,Qinke P, Zhang Q, Yanhua Y, Kankan L, et al. (2010) Splice sites prediction of human genome using length-variable Markov model and feature selection. Expert Systems with Applications, 2010: p. 2771-82.
6. Yin MM, Wang JTL (2001) Effective hidden Markov models for detecting splicing junction sites in DNA sequences. Information Sciences 139: 139-163.
7. Chen TM, Lu CC, Li WH (2005) Prediction of splice sites with dependency graphs and their expanded bayesian networks. Bioinformatics 21: 471-482.
8. Cai DY, Delcher A, Kao B, Kasif S (2000) Modeling splice sites with Bayes networks. Bioinformatics 16: 152-158.
9. Rajapakse JC, Ho LSH (2005) Markov encoding for detecting signals in genomic sequences. IEEE-Acm Transactions on Computational Biology and Bioinformatiocs 2: 131-142.
10. Marashi SA, Goodarzi H, Sadeghi M, Eslahchi C, Pezeshk H (2006) Importance of RNA secondary structure information for yeast donor and acceptor splice site predictions by neural networks. ComputBiolChem 30: 50-57.
11. Baten AK, Chang BC, Halgamuge SK, Li J (2006) Splice site identification using probabilistic parameters and SVM classification. BMC Bioinformatics 7 Suppl 5: S15.
12. Sonnenburg S, Schweikert G, Philips P, Behr J, Rätsch G (2007) Accurate splice site prediction using support vector machines. BMC Bioinformatics 8 Suppl 10: S7.
13. Lopes HS, Lima CRE, Murata NJ (2007) A configware approach for high-speed parallel analysis of genomic data. Journal of Circuits Systems and Computers 16: 527-540.
14. Churbanov A, Rogozin IB, Deogun JS, Ali H (2006) Method of predicting splice sites based on signal interactions. Biol Direct 1: 10.
15. Tsai KN, Lin SH, Shih SR, Lai JS, Chen CM (2009) Genomic splice site prediction algorithm based on nucleotide sequence pattern for RNA viruses. Comput Biol Chem 33: 171-175.
16. Anastassiou GA (2011) Multivariate sigmoidal neural network approximation. Neural Network 24: 378-386.
17. Hinoshita W, Arie H, Tani J, Okuno HG, Ogata T (2011) Emergence of hierarchical structure mirroring linguistic composition in a recurrent neural network. Neural Network 24: 311-320.
18. Liu JJ, Wang CG, Sun JX (2011) The Detection and Recognition of Electrocardiogram' s Waveform Based on Sparse Decomposition and Neural Network. Signal Processing 843-850.
19. Desai KM (2005) Use of an artificial neural network in modeling yeast biomass and yield of beta-glucan. Process Biochemistry 40: 1617-1626.
20. Zhang C, Lin M, Tang M (2008) BP neural network optimized with PSO algorithm for daily load forecasting. International Conference on Information Management, Innovation Management and Industrial Engineering 82-85.
21. Luitel B, Venayagamoorthy GK (2010) Quantum inspired PSO for the optimization of simultaneous recurrent neural networks as MIMO learning systems. Neural Netw 23: 583-586.
22. del Valle Y, Venayagamoorthy GK, Mohagheghi S, Hernandez JC, Harley RG (2008) Particle swarm optimization: Basic concepts, variants and applications in power systems. IEEE Transactions on Evolutionary Computation 12: 171-195.
23. Pantoja FM, Bretones AR, Garcia Ruiz F, Garcia SG, Martin RG (2007) Particle-swarm optimization in antenna design: optimization of log-periodic dipole arrays. IEEE Antennas and Propagation Magazine 34-47.
24. Jiang Y (2007)An improved particle swarm optimization algorithm. Applied Mathematics and Computation 193: 231-239.
25. Samanta B, Nataraj C (2009) se of particle swarm optimization for machinery fault detection. Engineering Applications of Artificial Intelligence 22: 308-316.
26. Chen HY, Leou JJ (2010) Saliency-directed image interpolation using particle swarm optimization. Signal Processing 90: 1676-1692.
27. Hansen LK, Salamon P (1990) Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence 12: 993-1001.
28. Hashem S, Schmeiser B (1995) Improving model accuracy using optimal linear combinations of trained neural networks. IEEE Trans Neural Netw 6: 792-794.
29. Kennedy J, Eberhart R (1995) Particle swarm optimization. Ieee International Conference on Neural Networks Proceedings 4: 1942-1948.
30. Wang B (2010) A Sensor Pre-assignment Method Based on Particle Swarm Optimization. Signal Processing 26.
31. Yong-ling Z, Long-hua M, Zhang LY, Ji-xin Q (2003) Empirical study of particle swarm optimizer with an increasing inertia weight. Congress on Evolutionary Computation (IEEE Cat.No.03TH8674).
32. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. IEEE International Conference on Systems, Man, and Cybernetics.
33. Ratnaweera A, Halgamuge SK, Watson HC (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. Ieee Transactions on Evolutionary Computation 8: 240-255.
34. Huang J, Li T, Chen K, Wu J (2006) An approach of encoding for prediction of splice sites using SVM. Biochimie 88: 923-929.
35. Wei-Bing, Xian JW (2008) An evolutionary game based particle swarm optimization algorithm. Journal of Computational and Applied Mathematics 30-35.
36. Xingjuan C, Zhihua C,Jianchao Z, Ying T (2008) Particle swarm optimization with self-adjusting cognitive selection strategy. International Journal of Innovative Computing, Information & Control 4: 943-52.
37. Byung-In K, Juyoung W (2010) Last two fit augmentation to the well-known construction heuristics for one-dimensional bin-packing problem: An empirical study. International Journal of Advanced Manufacturing Technology 50: 1145-1152.
38. Yuan XH, Nie H, So AJ, Wang L, Yuan YB (2009) An improved binary particle swarm optimization for unit commitment problem Expert Systems with Applications 36: 8049-8055.
39. Lin JT, Yin-Yann C (2009) A modified particle swarm optimization for production planning problems in the TFT array process. Expert Systems with Applications 12264-12271.
40. Albayrak M, Allahverdi N (2011) Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms. Expert Systems with Applications 38: 1313-1320.
41. Aras N, Oommen BJ, Altinel IK (1999) The Kohonen network incorporating explicit statistics and its application to the travelling salesman problem. Neural Network 12: 1273-1284.
42. Garcia-Villoria A, Pastor R (2009) Introducing dynamic diversity into a discrete particles warm optimization. Computers & Operations Research 36: 951-966.
43. RamponeS, Pollastro P (2003) HS^{3}D: Homo sapiens splice site data set. Nucleic Acids Research.
44. Chao-Hsien C, Zhang Y, Chen Y, XiangY, Hongyuan Y (2006) Splice site prediction using support vector machines with a Bayes kernel. Expert Systems with Applications 30: 73-81.