Proceedings of the 2007 IEEE Symposium on Computational
Intelligence in Bioinformatics and Computational Biology (CIBCB 2007)

Biological Sequence Mining Using Plausible Neural Network and its Application to Exon/intron Boundaries Prediction

Kuochen Li, Dar-jen Chang, and Eric Rouchka
CECS, University of Louisville, Louisville, KY 40292, USA

Yuan Yan Chen
PNN Technologies Inc, PO Box 7051, Woodbridge, VA 22195, USA

***Abstract*— Biological sequence usually contains yet to find knowledge, and mining biological sequences usually involves a huge dataset and long computation time. Common tasks for biological sequence mining are pattern discovery, classification and clustering. The newly developed model, Plausible Neural Network (PNN), provides an intuitive and unified architecture for such a large dataset analysis. This paper introduces the basic concepts of the PNN, and explains how it is applied to biological sequence mining. The specific task of biological sequence mining, exon/intron prediction, is implemented by using PNN. The experimental results show the capability of solving biological sequence mining tasks using PNN.**

I. INTRODUCTION

Research in computational biology has grown rapidly in the past ten years due to advances in molecular biology techniques yielding high-throughput data. One of the most important research interests in computational biology is sequence mining. It is much harder to extract knowledge hidden in the sequences than to generate biological sequences. With biological mutation and evolution, sequence datasets usually are enormous and complex. Analysis models become a critical factor for biological sequence mining. Several tasks related to sequence mining such as pattern discovery, classification, prediction, and clustering, can be implemented by statistical, neural network, or data mining models [1][2]. Those models can be used to capture the knowledge or patterns in order to predict, classify, or analyze sequence data.

A newly developed neural network model, plausible neural network (PNN), which combines probabilistic and possibilistic inferences for binary random variables was introduced in [3] and subsequently patented by Chen in 2003 [4]. PNN is similar to Hopfield neural networks in that both are bidirectional and symmetrical. But in inference interpretation, PNN is similar to Bayesian neural networks. Specifically the synaptic weights in PNN and Bayesian neural networks have probabilistic meanings and thus are transparent. However, PNN with the weights based on mutual information content as described in section II provides two important features. First, mutual information content weights put the PNN architecture of winner-take-all activation of competing neurons on a solid statistical inference foundation, in which no prior distribution is required as in Bayesian neural networks. Second, mutual information content weights can be used to compute Shannon mutual information between attributes with little added cost.

Several advantages of PNN lead to applying it to biological sequence mining. The PNN model is very intuitive in that the user can easily train it with different initial states to discover patterns hidden in large-scale, complex data. In addition, the model is very general that it can be applied to clustering, prediction, classification, and pattern discovery with the same architecture. Moreover, PNN represents missing data as a null vector so that they do not participate in the PNN inference process. Most of other clustering and classification methods handle missing data by adding artificial data or removing the incomplete data. Since the learning process and activation method in PNN are simple, the computation for training or activation can be finished in a short period of time.

This paper applied PNN to one of the biological sequence mining tasks, the prediction of exon/intron boundaries. The accuracy rate comparing to other computational intelligent models shows the quality result of PNN.

II. A PNN MODEL FOR BIOLOGICAL SEQUENCE MINING

A. PNN Architecture

A basic PNN architecture for biological sequence mining consists of two layers (input layer and hidden layer) of cooperative and competitive neurons. The connection between input neurons and hidden neurons is complete, bidirectional, and symmetric.

The competitive neurons are grouped into a winner-take-all (WTA) ensemble and, in turn, WTA ensembles cooperate in decision making. Fig.1 illustrates the architecture for DNA sequence data. The hidden neuron WTA ensemble represents un-labeled (thus unknown or hidden) patterns of similar sequences. Each input WTA ensemble encodes the values of an input attribute, which is either a sequence base or the sequence

class. For example, in Fig. 1 the input WTA ensemble (*IE*, *EI*, *N*) encodes the values of attribute Class (i.e. DNA sequence class label) and each WTA ensemble (*A*, *C*, *G, T*) encodes a DNA base in the sequence.

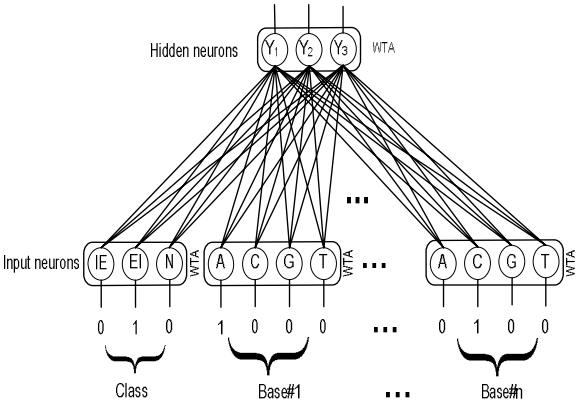

Fig.1. PNN architecture for predicting exon/intron boundaries

B. Attribute Value Coding

PNN encodes categorical or continuous attributes into WTA ensembles. For sequence mining, only the categorical attribute encoding is required. For a categorical attribute with k categories, its value is encoded as a WTA ensemble of k binary-valued (i.e. 0 or 1) neurons ($X_1, X_2, \ldots, X_k$). For example, splice-junction gene sequences datasets may have a class attribute with three possible values *IE, EI*, and *N* representing "intron to exon boundary", "exon to intron boundary", and "None", respectively. For such examples, the class attribute can be encoded in a three-neuron WTA ensemble (*IE*, *EI*, *N*). The input vector (*1, 0, 0*) for the class attribute WTA ensemble indicates the sequence is of the *IE* class, (*0, 1, 0*) indicates *EI*, and (*0, 0, 1*) indicates *N*.

Each base in a DNA sequence is encoded by a four-neuron WTA ensemble (*A*, *C*, *G, T*). This DNA encoding can allow for uncertain base value given in the IUPAC code as well. For example, if the base value is *R* (means *G* or *C*), the input vector for the base is (*0, 0.5, 0.5, 0*). Missing attribute values are represented by a null vector, i.e. all X_i's in the attribute ensemble are zero.

Given the coding scheme, an input WTA ensemble ($X_1, X_2, \ldots, X_k$) representing an attribute value satisfies the following conditions:

$$\sum_{i=1}^{k} X_i = 1 \quad \text{and} \quad 0 \le X_i \le 1 \text{ for all } X_i \tag{1}$$

Furthermore, missing attribute values are represented by a null vector, i.e. all X_i's in the attribute ensemble are zero. With this coding, since the input from every neuron in the attribute ensemble is zero, activation potential (i.e. input times the weight) contributed by the attribute is zero. As a consequence, the attribute with missing value will not participate in the WTA activation of the competing hidden neurons.

C. Connection Weights

Assume each neuron in the PNN is a binary random variable. The connection weight between an input neuron X_i and a hidden neuron Y_j is given as follows

$$\omega_{ij} = \ln\left(\frac{P(X_i = 1, Y_j = 1)}{P(X_i = 1)P(Y_j = 1)}\right) \tag{2}$$

In (2), *P* denotes the probability of the enclosed event. The weight (2) contains the firing history or *mutual information content* of two connected neurons. It is clear that X_i and Y_j are positively associated (excitatory), statistically independent, or negatively associated (inhibitory) if ω_{ij} is positive, 0, or negative, respectively. Note that binary random variable is assumed to define (2), but the estimation of (2) as described in section *E* and PNN applications also apply to neurons satisfying (1).

D. Forward and Reverse Firing

The PNN is bidirectional since the firing in PNN can be carried out in both directions between input neurons and hidden neurons. For convenience, forward firing is referred to the activation of hidden neurons given the states of the input neurons and reverse firing is referred to the activation of input neurons given the states of hidden neurons. Suppose an ensemble of competitive WTA hidden neurons $Y_1, Y_2, \ldots, Y_n$ receive input signals from the input neurons $X_1, X_2, \ldots, X_m$. The firing states of $Y_1, Y_2, \ldots, Y_n$ are computed in the following steps:

1. Compute the activation potential Z_j , $j = 1, 2, \ldots, n$, as follows

$$Z_j = \exp\left(\sum_i \omega_{ij} X_i\right) \tag{3}$$

2. Compute *Max*, the maximum value of Z_j, $j = 1,2,..,n$.
3. Perform α-cut ($0 < \alpha \le 1$) on Z_j, $j = 1,2,..,n$, i.e. set Z_j to zero if

$$\frac{Z_j}{Max} \le \alpha$$

4. Finally calculate the activation level of Y_j, $1 \le j \le n$, by

$$Y_j = \frac{Z_j}{\sum_k Z_k} \tag{4}$$

Step 3 is a soft WTA competition since there could be multiple winners with different activation level. Alternatively, one winner could be chosen with the largest activation level. The latter is called hard WTA. Step 4 is a normalization step to make sum of Y_j equal to one.

In the reverse firing, the states of the hidden neurons $Y_1, Y_2, \ldots, Y_n$ are given and the computation of the activation of

input neurons is carried out for each input WTA ensemble separately using steps 1-4 above with some modifications. Specifically, for each input WTA ensemble (X_1, X_2,…, X_k), the activation of neurons X_j's is computed using steps 1-4 with these modifications:

a. $Z_j = \exp(\sum_i \varpi_{ij} Y_i)$, $j = 1, 2,...,k$ where $\varpi_{ij} = \omega_{ji}$

($\varpi_{ij} = \omega_{ji}$ is why the PNN is said to be symmetric.)

b. In the normalization step 4, $X_j = \frac{Z_j}{\sum_k Z_k}$, $1 \le j \le k$.

c. Usually, the α value chosen for this activation is much smaller than that used for the forward firing activation.

E. The Principle of Inverse Inference

The PNN process of input data rows is via the WTA competition among the hidden neurons. Assuming the hard WTA competition is used, one hidden neuron will become the winner of an input data row. The outcome of the WTA competition is judged by the activation level of each hidden neuron from the forward firing of a data row as described in subsection D. The hidden neuron with the highest activation level will become the winner (i.e. the data row in question belongs to the cluster represented by the hidden neuron.) This competition process can be justified by the principle of inverse inference, which is stated as "Given the evidence, the more probable a hypothesis can produce such evidence the more likely it to be true" [3]. To paraphrase the principle in PNN terms, given an input data row (i.e. given input neurons' values), the more probable hidden neuron can response to such an input the more likely the data row belongs to the cluster represented by the hidden neuron. Suppose X_1, X_2,…, and X_m denote the states (0 or 1) of the input neurons in a PNN, the "evidence" (or the possibility) of a hidden neuron, Y_j, responding to the input can be measured by the following conditional probability:

$$P(X_1, X_2, \ldots, X_m \mid Y_j) \tag{5}$$

Therefore, to justify the principle of inverse inference as applied to PNN, it needs to show that the winning hidden neuron Y_j judged by the forward firing using the weight (2) has indeed the highest quantity (5). Using (2) to compute the activation potential in (3), yields

$$\sum_i \omega_{ij} X_i = \sum_i \ln(P(X_i \mid Y_j) - \sum_i \ln(P(X_i)) \tag{6}$$

If the input attributes are statistically independent, (6) can be reduced to:

$$\sum_i \omega_{ij} X_i = \ln(P(X_1, \ldots, X_m \mid Y_j) / \prod_i P(X_i)) \tag{7}$$

Plugging (7) into (3), the hidden neuron Y_j's pre-WTA activation value, Z_j, becomes:

$$Z_j = P(X_1 \ldots, X_m \mid Y_j) / \prod_i P(X_i) \tag{8}$$

Comparing (8) and (5), Z_j is equal to the conditional probability (5) divided by a factor common to all hidden neurons. As a result, using (8) to determine the winning hidden neuron yields the same result as using (5). This concluded the justification of the principle of inverse inference used in PNN. Furthermore since the sum of (5) over all Y_j is one, the normalization actually computes (5) as the activation level for the hidden neuron Y_j.

The statistical independence assumption on input attributes is a limitation of PNN as that in the naïve Bayesian neural networks. However, in data exploration without any prior knowledge of the data, the limitation would not be a big problem in PNN applications. One important note about PNN is no prior distribution is needed due to the uniform prior present in the denominator of (8). This is a huge advantage over Bayesian neural networks, which require a difficult task of estimating prior distribution.

F. PNN Training

To apply PNN, a training (learning) procedure is needed to estimate the weight (2). The PNN training algorithm estimates the weights from a given dataset. Given the past co-firing history of two neurons X and Y, (x_k, y_k), $k = 1,2,\ldots,n$, the weight (2) between X and Y in a PNN can be estimated by

$$\omega = \ln\left(\frac{n\sum_{k=1}^{n} x_k y_k}{\sum_{k=1}^{n} x_k \sum_{k=1}^{n} y_k}\right) \tag{3}$$

Suppose that a training set of DNA sequences is given. First, fire the hidden neurons randomly for each sequence to produce an initial fire matrix. Based on the initial fire matrix and training sequences, calculate the new weights between input neurons and hidden neurons using (3). In turn, use the new weights in the forward firing of training sequences to get a new fire matrix. Repeat this process until the PNN is stable, i.e. until the fire matrix is not changed. Fig.2 shows the PNN training algorithm.

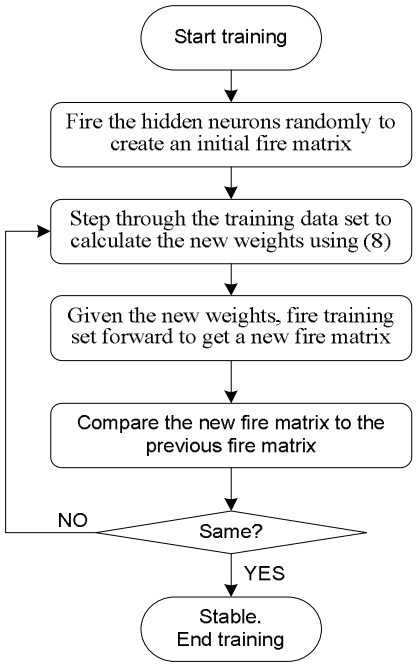

Fig.2. The PNN training algorithm

III. EXPERIENTS

A. Datasets

Two datasets are used to test the performance of PNN. The first dataset is downloaded from [7]. This dataset contains 3190 sequences which are taken from the GenBank 64.1. Each data sample contains a class attribute (I/E, E/I, or N), instance name, and 60-base sequence with 30 bases on either side of the boundary. 767 sequences are classified as E/I. 768 sequences are classified as I/E. 1655 sequences are classified as N.

In order to determine PNNs could be used to determine splice site prediction with large dataset, a set of sequences used to benchmark gene prediction programs was downloaded from [5]. The second dataset contains 9204 sequences extracted from 570 different genes. From this set of sequences, 2629 Exon/Intron (E/I) boundaries and 2634 Intron/Exon (I/E) were extracted, with 30 bases of sequence on either side of the boundary reported. In addition, 3941 60-base sequences not occurring in an E/I or I/E (reported as an N) were extracted. This dataset was then separated randomly into two parts. One was used to train a PNN with 8 hidden neurons to distinguish between these three classes. The other one was used to test the performance of the PNN for I/E and E/I recognition.

B. Comparing methods

Eight different classification algorithms are tested using the first sequence dataset in [8]. The algorithms are listed as follows:

1. ID3: a decission tree algorithm
2. BP: the backpropagation algorithm
3. LVQ: a combination of procedures of LVQ
4. FS+LVQ: a LVQ learning with feature selection
5. 1-nn: a 1-nearest neighbor neural network
6. FSlNN: a 1-nearest neighbor neural network with a previous feature selection
7. k-nn: a k-nearest neighbor neural network
8. FSKNN: **a** k-nearest neighbor neural network with a previous feature selection

In order to compare PNN to the above algorithms, same dataset is used to test the PNN and the accuracy rate is evaluated. The tested PNN contains 8 hidden neurons and the soft WTA competition is used for the network. Table 1 shows the accuracy rates of PNN and the other algorithms obtained from [8]. The result shows the PNN achieved higher accuracy rate than any algorithms listed above.

Table 1. Accuracy rate of E/I I/E prediction using different algorithms

PNN	ID3	BP	LVQ	FS+LVQ
96.667	89.50	91.20	77.66	85.08

1-nn	FS1NN	k-nn	FSKNN
66.38	71.86	72.23	77.77

C. Evaluating the performance of PNN

The second dataset is employed to test the PNN performance of handling the large dataset. From the total of 9204 sequences, 6204 randomly selected sequences were used to train the PNN, and the other 3000 sequences were used to test the PNN. The PNN contains 8 hidden neurons and 243 input neurons (3 for the class attribute and 4 for each base). Alpha cut was set to 0.8 in order to have more precise prediction. The PNN was trained for 200 iterations and the training was completed in minutes but yet to meet the stable condition. The 3000 testing sequences then input to the trained PNN to test the performance. The result showed that 2830 out 3000 sequences were correctly classified. In other words, 94.3% accuracy rate was achieved. The performance parameters: sensitivity (Sn), specificity (Sp), and correlation co-efficiency (CC) were calculated to assess the performance of the PNN. The definitions of Sn, Sp and CC are as follows:

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

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

$$CC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$$

where TP stands for True Positive, FN: for False Positive, TN for True Negative, and FN for False Negative

Table 2 shows the performance parameters of this experiment. Notice Sn, Sp, and CC have rather high percentages for both the I/E and E/I predictions using PNN even the PNN had not yet reached an optimal state. However, the training was extended for couple more hundreds of iterations, but the performance was not improved significantly. To improve training performance, the next experiment used a larger training dataset.

Table 2. Performance parameters of E/I and I/E predictions for 3000 testing sequences

	Sn	Sp	CC
E/I	97.2	96.95	95.06
I/E	91.3	94.20	88.07

In this experiment, the training dataset was increased to 7204 randomly selected sequences. The rest of the 2000 sequences were used to test the performance of the PNN. The same PNN from the previous experiment was configured for this experiment. The training of the PNN was finished surprisingly in 58 iterations and 1883 out of 2000 testing sequences were correctly classified. Therefore, the PNN achieved about 94% accuracy rate in a very short period of training time. Table 3 also shows the performance parameters of this experiment.

Table 3. Performance parameters of E/I and I/E predictions for 2000 testing sequences

	Sn	Sp	CC
E/I	97.22	96.39	94.52
I/E	91.19	94.79	88.18

The measurement of accuracy for this experiment shows the similar result from the previous experiment. Additionally, for misclassification, the PNN provided some useful information (e.g. 0.45: N 0.55:E/I) which could be used for further analysis.

Since the training of PNN is based on mutual information instead of error correction, to improve the performance, a simple tuning algorithm can be applied. It simply repeats the training process and records the configuration of the PNN, which has the best accuracy rate for the classification.

Table 4. Measurement of E/I and I/E prediction for 2000 testing sequences

	Sn	Sp	CC
E/I	97.47	96.24	94.83
I/E	95.47	95.13	92.00

Table 4 shows the measurement of the PNN with the tuning algorithm. Comparing to 117 misclassified testing sequences from experiment 2, only 89 misclassified testing sequences were found out of 2000 sequences. Furthermore, the I/E prediction had significantly improved in Sn and CC as shown in Table 4.

IV. CONCLUSION

PNN not only shows good performance for the prediction of exon/intron boundaries but also can discover the patterns of exon/intron boundaries. For a trained PNN, each hidden neuron represents a found pattern. The pattern can be easily extracted by reverse-firing the particular hidden neuron. The extracted knowledge then provides better understanding of the dataset. Thus, the number of hidden neurons is crucial to the PNN performance. In this paper, different numbers of hidden neurons are tested in the experiments. The PNN with 8 hidden neurons shows the best performance regarding to the accuracy rate and the training time. However, to decide the minimum number of hidden neurons to achieve desired performance criteria is still an open issue.

V. FUTURE WORK

PNN has shown the abilities to solve problems such as classification, clustering, pattern recognition, function approximation, etc [9] [10]. In this paper, PNN has shown promise for gene splice-junction recognition. The fast convergence of PNN makes it feasible for large biological sequence analysis. For future work, the study of new junction pattern discovery and improvement of PNN for sequence mining in general are interesting to be implemented. In addition, the possibility of applying PNN to promoter prediction and protein profile pattern recognition will also be considered.

REFERENCES

[1] B. S. Everitt and G. Dunn, *Applied Multivariate Data Analysis*. London: Arnold, 2001, ch. 6.

[2] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2001, ch. 7 and ch. 8.

[3] Y. Y. Chen, "Plausible neural networks," *Advance in Neural Networks World,* A. Grmela and N. E. Mastorakis, Ed. WSEAS Press 2002, pp. 180-185.

[4] Y. Y. Chen, "Plausible neural network with supervised and unsupervised cluster analysis," U.S. Patent 20030140020, July 24, 2003.

[5] M. Burset and R. Guigo, "Evaluation of gene structure prediction programs," Genomics 34:353-357, 1996.

[6] E.Y. Chen,, M. Zollo, R.A. Mazzarella, A. Ciccodicola, C.N. Chen, L. Zuo, C. Heiner, F.W. Burough, M. Ripetto, D. Schlessinger and M. D'Urso, "Long-range sequence analysis in Xq28: thirteen known and six candidate genes in 219.4 kb of high GC DNA between the RCP/GCP and G6PD loci," Hum. Mol. Genet. 5 (5), 1966, pp. 659-668. vol. 7, no. 3, pp.368-369, June 1999.

[7] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz, "UCI Repository of machine learning databases", University of California, Department of Information and Computer Science, Irvine, CA, 1998 < http://www.ics.uci.edu/~mlearn/MLRepository.html>

[8] M. Mar Abad Grau, and L.D.H Molinero, " Feature selection in codebook based methods provides high accuracy", IJCNN'99, 1856-1860 vol.3, 1999.

[9] K. Li, D. Chang, and Y. Y. Chen, "High-speed Bi-directional Function Approximation Using Plausible Neural Networks," Proceedings of IJCNN'06, 2006.

[10] K. Li and D. Chang,"Fuzzy Membership Function Elicitation using Plausible Neural Network," Proceedings of ICAI'06 Volume I, 2006, pp. 141-147.