IDENTIFY GENE SPLICED SITES BASED ON BP NN

Sun Xiao, Xie Xueying*, Lu Zuhong
(Dept. Of Biology & Biomedical Engineering, Southeast University, Najjing 210096, China)
*email address: biomi2@seu.edu.cn

Abstract: The neural network (NN) has a strong self-adaption ability of learning, memorizing, association and recognizing. It has been applied in many fields broadly. In this paper, we have studied the NN application for analyzing eukaryotic DNA coding sequences. The simulation result shows that BP NN is a good model for recognizing gene splice sites.

Key words: Neural Network; DNA coding sequence; BP; splice site

Introduction
The rapid expansion in the amount of DNA and inferred protein sequence data resulting from the progress of genome initiative and other projects has led to a compelling need for computational aids in analyzing and identifying important, functional segments within these sequences [1]. In this paper, we apply the neural network to mining these sequence data, because the NN has an excellent non-linear capacity to manage abundant data with incomplete information and noise infection.

Some important biology knowledge is used in different ways in this paper, so the following paragraph expounds it simply within the context of this study.

Recognition of genes in eukaryotic DNA is seriously complicated owing to their intrinsic structure [4]. Simply to speaking, the eukaryotic gene contains coding sequences spaced with non-coding sequences, the coding sequence is called exon, and non-coding sequence named intron. Therefore, eukaryotic gene is composed of a series of introns and exons with different length (Fig1). In addition, the gene begins with initial site ATG, ends with TGA generally. It has been found that introns usually begin with GT(named donor splice site) and end with AG dinucleotides (named acceptor splice site). These ATGs, TGAs, GTs and AGs, with a general designation, are named splice sites [4]. Certainly, not all these substrings in DNA are really splice sites because the splice sites are not independent; they are associated with their context closely. We call substrings, but which are not true splice sites, false splice sites.

Figg1

\[ e_1, i_1, e_2, i_2, \ldots, e_k, i_k, e_n, i_n \]

i--intron, e--exon

So, the gene recognition in eukaryote depends on identify of splice sites largely. In our research, we first train the BP network with train samples, then, test the trained network with test samples to compute the recognition ratio. The network with a recognition ratio above threshold has an excellent performance to discovery unknown splice sites, that are important to identify eukaryote genes. This detailed process will be explained in the following Method and Algorithm.

Environment and Materials
We carried out this research with software of Visual C++ 6.0 version in hard environment of PIII 450MHz, 128M memory.

Sequences analyzed in experiment are all from Genbank, in which the boundaries of exon or intron have been known. We can extract the true splice sites easily. Part examples are listed in the following table 1.

Table 1.

<table>
<thead>
<tr>
<th>ID</th>
<th>Sequence Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB006686</td>
<td>Homo sapiens gene for myocilin</td>
</tr>
<tr>
<td>AB007115</td>
<td>Homo sapiens k12 gene for keratin 12</td>
</tr>
<tr>
<td>AB009806</td>
<td>Homo sapiens gene for osteonidogen</td>
</tr>
<tr>
<td>AF001951</td>
<td>Homo sapiens gonadotropin releasing hormone receptor (GNRHR) gene</td>
</tr>
<tr>
<td>AF009249</td>
<td>Homo sapiens putative chloride channel gene (CLCN6)</td>
</tr>
<tr>
<td>AF010398</td>
<td>Homo sapiens transaldolase-related protein gene</td>
</tr>
<tr>
<td>AF012339</td>
<td>Homo sapiens glutaryl-CoA dehydrogenase (GCDH) gene</td>
</tr>
</tbody>
</table>
There are both positive samples and negative samples in sample set. Positive samples are those substrings around true splice sites, and vice versa, negative samples are substrings around false splice sites. We choose 1000 positive samples and 780 negative samples from sequences. Both kinds of samples are scattered random in the set. 75 percent of the sample set forms a train set, and the other forms a test set.

**Method and Algorithm**

We first provide the classic BP model structure graph as fig2. There are three layers in this model, the left one is input layer, middle hidden layer and right output layer. The number of neurons, which is denoted with circle, in input layer and output layer are computed in sequence coding. Because there is no an excellent principle to determine the number of neurons in hidden layer, we have done many experiments to specify it. Because the BP algorithm is a very classic method, we wouldn’t introduce it in detail. If you are interested in it, you can see [2]. In the following, We will explain some parameters and functions used by BP network which applied to identify the splice sites.

**Input preprocess:** We apply the Bin4 or Bin2 to coding the samples (table 2). So, the number of neurons in input layer is 4*Lenth (Bin4) or 3*Lenth(Bin2).

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>G</th>
<th>C</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bin4</td>
<td>0001</td>
<td>0010</td>
<td>0100</td>
<td>1000</td>
</tr>
<tr>
<td>Bin2</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

**Output postprocess:**

\[
\begin{align*}
 y_0' &= 1, \quad y_1' = 0 \quad \text{if } (y_0 - y_1) > \text{const} \\
 y_0' &= 0, \quad y_1' = 1 \quad \text{if } (y_0 - y_1) < -\text{const} \\
 y_0' &= 0, \quad y_1' = 0 \quad \text{other}
\end{align*}
\]

Where if \( y_0', y_1' = 01 \), then the input is an acceptor splice site; if \( y_1', y_0' = 10 \), the input is a donor splice site; and if \( y_1', y_0' = 00 \), the input is not a splice site.

**Activation function:**

\[
\begin{align*}
\text{Input layer} & \quad f(x_i) = x_i \\
\text{hidden and output layer} & \quad f(x_i) = \frac{1}{1+e^{-x_i}}
\end{align*}
\]

Where \( x_i \) is the input for \( i \) th neuron, \( f(x_i) \) is the corresponding output.

**Error function:**

\[
E_p = \frac{1}{2} \sum_{l=0}^{s} (t_i^p - y_i^p)^2
\]

for sample \( p \), \( E = \frac{1}{p} \) for system error, where \( y_i^p \) is the actual output of \( l \) th neuron in output layer, \( t_i^p \) is the corresponding ideal output.

**Weight modifying:**

\[
w_{kl}(n+1) = w_{kl}(n) + \eta d_{kl}(n) + \alpha \Delta w_{kl}(n)
\]

here \( w_{kl}(n) \) represents the weight between neuron \( k \) and \( l \) in the \( n \)th iteration; \( d_{kl}(n) \) is the adjustment value for \( w_{kl}(n) \) in \( n \)th iteration, \( d_{kl}(n) = \delta_{kl}^p x_k^p \) (5), where \( \delta \) is the error of every layer, it can be donated individually as following:

**Input layer:**

\[
\delta_{kl}^p = (t_i^p - y_i^p) y_i^p (1 - y_i^p) \quad l \in s
\]

**Hidden layer:**

\[
\delta_{jk}^p = \sum_{l=0}^{s-1} \delta_{kl}^p w_{kl} x_k^p (1 - x_k^p) \quad k \in m
\]

**Output layer:**

\[
\delta_{ij}^p = \sum_{k=0}^{m-1} \delta_{jk}^p w_{jk} x_j^p (1 - x_j^p) \quad j \in n
\]

Where \( s, m, n \) is the number of neurons in input layer, hidden layer and output layer respectively.

\[
\Delta w_{kl}(n) = \eta \ d_{kl}(n - 1) \quad (9)
\]

\( \alpha \) is a momentum factor, \( \eta \) is a learning factor, we
import them in order to accelerate the convergent rate and prevent wobble. They can be specified by user.

**Results and Discussion**

1. Astringency of Model

The principal character of BP model, which has the ability to identify patterns, is its astringency. We will analyze the model’s astringency by model system error or the learning speed. The bigger of system error curve’s gradient is, the faster of the learning speed is. In general, a model is astringent if the movement trend of its system error is descendant and to a constant. So, we focus on learning factor (LF) and momentum factor (MF) to analyze the system error. Fig2 shows different learning factor’s impact to system error. During the first iterations, the bigger of learning factor is, the faster of the system error descend. But after 1000 iterations, the model with LF=0.1 is convergent faster than which with LF=0.3. Because though the learning speed can be drastically increased with large learning factor, the learning may not be exact, with tendencies to overshoot, or it can be convergent to a worse state (fig3, where LF=0.8).

The purpose of the momentum factor is to accelerate the convergence of the BP model. Because if the error has a smooth change along a certain direction, the learning rate along it should be increased, and if the error changes its sign in two consecutive iterations, the learning rate should be decreased.
Simulation Results

Our purpose of constructing the BP model is to identify the splice sites in DNA sequences. In order to test the effectiveness of our model, it’s applied to several data sets. The simulation results are shown in Table 2. The performance of model is described by the recognition, which is defined as the ratio of correctly recognized splice sites versus the real splice sites.

Table 3

<table>
<thead>
<tr>
<th></th>
<th>Test set</th>
<th>Train set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>400</td>
</tr>
<tr>
<td>(\eta = 0.1)</td>
<td>92%</td>
<td>98.5%</td>
</tr>
<tr>
<td>(\eta = 0.3)</td>
<td>90.1%</td>
<td>97.25%</td>
</tr>
</tbody>
</table>

Reference

[4] [Italy] Luciano Milanesi *Gene Structure Prediction*