Amino Acid Encoding Schemes from Protein Structure Alignments: Multi-dimensional Vectors to Describe Residue Types

KUANG LIN, ALEX C. W. MAY AND WILLIAM R. TAYLOR*
†

†Division of Mathematical Biology, National Institute for Medical Research, The Ridgeway, Mill Hill NW7 1AA, UK

(Received on 28 August 2001, Accepted in revised form on 5 December 2001)

Bioinformatic software has used various numerical encoding schemes to describe amino acid sequences. Orthogonal encoding, employing 20 numbers to describe the amino acid type of one protein residue, is often used with artificial neural network (ANN) models. However, this can increase the model complexity, thus leading to difficulty in implementation and poor performance. Here, we use ANNs to derive encoding schemes for the amino acid types from protein three-dimensional structure alignments. Each of the 20 amino acid types is characterized with a few real numbers. Our schemes are tested on the simulation of amino acid substitution matrices. These simplified schemes outperform the orthogonal encoding on small data sets. Using one of these encoding schemes, we generate a colouring scheme for the amino acids in which comparable amino acids are in similar colours. We expect it to be useful for visual inspection and manual editing of protein multiple sequence alignments.

Introduction

The artificial neural network (ANN) is a sophisticated modelling technique capable of modelling extremely complex functions and automatically learning the structure of data (Bishop, 1995). ANNs have been widely applied to many different problems in bioinformatics (for reviews, see Baldi & Brunak, 1998; Wu & McLarty, 2000).

In neural network methodology, samples are often subdivided into “training” and “testing” sets. The training set is a set of examples used for “learning”: fitting the parameters (i.e. weights) of a neural network. The testing set is a distinct set of examples used to assess the performance of a trained neural network. It is important to maintain a strict separation of these data sets with the testing set being applied only after determination of network architecture and connection weights.

A basic assumption in neural network training (and model optimization approaches of other machine learning methods) is that the training data exhibit an underlying systematic aspect but are corrupted with random noise (Bishop, 1995). The central goal of model optimization is to produce a system able to make good predictions for cases not in the training set. It requires the model to represent the underlying mechanism correctly. Training an over-complex model may fit the noise, not just the signal, leading to “overfitting”. Such a model will have low training error but a much higher testing error. Generally, its performance on new cases will be poor. The best way to avoid overfitting is to use...
a large and diverse training set. However, given a training set of a limited size, model selection can be employed to improve generalization. With small training and testing sets, simpler models are often preferable for better performance (Bishop, 1995; Müller et al., 1996).

Orthogonal encoding of amino acid types has been used in many bioinformatic neural network models: 20 input units are assigned to describe one protein residue. In the 20-dimensional space, the vector \([1, 0, 0, 0 \ldots 0, 0, 0]\) represents alanine, and \([0, 0, 0 \ldots 0, 0, 0, 1]\) stands for valine. With this encoding, a typical input window of 13 residues requires 260 \((13 \times 20)\) input units. It can easily lead to large input layers, many connecting weights, and hence complex models. Without sufficient data to support training, over-complex models are prone to overfitting. Unfortunately, in many bioinformatic problems, huge data sets can be simply unavailable. Even when they are available, analysing them is often very computationally demanding. Simplified encoding schemes use less input units to describe a given amino acid sequence; thus, we can use smaller models to describe the same phenomena. By introducing these simplified models, we can reduce the reliance on huge data sets and improve performance. To increase the level of neural network generalization, Skolnick et al. (1997) defined a 10-unit input scheme for representation of amino acid type. Each amino acid was described using ten numbers. Their representation was based on the amino acid features described by Taylor (1986): each unit corresponds to one biochemical feature, amino acids sharing many features have similar codes. Weiss & Herzel (1998) suggested two differing properties, “sequence-derived hydrophobicity” and “sequence-derived polarity”, based on correlations in protein sequences. Jagla & Schuchhardt (2000) applied an adaptive encoding neural network to find automatically a classifier with a low-dimensional encoding matrix. Their encoding scheme was tested on the prediction of cleavage sites in human signal peptides of secretory proteins.

Here, we use a supervised back-propagation neural network model to develop a series of schemes using several \((1-10)\) input units to describe an amino acid. In these low-dimensional representations, amino acids with similar biophysical properties are clustered together. These schemes are tested on the simulation of amino acid substitution matrices. With small training sets, simpler schemes can achieve better results. By using those simplified encoding schemes, we can greatly speed up the propagation and training of neural network models.

There is a clear need for a well-grounded amino acid colouring scheme to ease the interpretation of sequence alignments. Colouring comparable amino acids in similar colours facilitates manual examination and modification of sequence alignments. Different approaches have been taken to colour amino acid types according to their hydrophobicity, size and other biochemical properties (for example, Taylor, 1997). Here, we generate a Red–Green–Blue (RGB) colour scheme by linearly transforming the values in our encoding scheme with three hidden units. This automatically constructed scheme can be easily adapted for other bioinformatic software. We write a simple Java program to browse protein alignments with this colouring scheme.

**Method**

**DATA SETS**

We use the CATH protein structural domain database (Orengo et al., 1997) (v1.6) to select training and testing sets. Domains with breaks in their alpha carbon backbones are excluded. Firstly, we select 681 pairs of protein domains, in which the two domains of each pair are in the same sequence family but not near identical structures. Then, another 339 pairs are chosen, with each domain sharing the same homologue family with the other one in the pair, but being in different sequence families. We align these domain pairs using Structure Alignment Program (SAP) (Taylor & Orengo, 1989; Taylor, 1999). In SAP, the pair-wise relationships between residues from different domains are scored on the spatial position of residues relative to the local coordinate frame. The score ranges from 0 to several hundreds and most significantly similar residue pairs score more than 1. Thus, to avoid noise from amino acids aligned without significant similarity, we set a threshold of SAP score to 1. Aligned residue pairs with
lower scores are discarded (8% of all aligned residue pairs). We use four-fifths of these structure alignments (800 randomly selected alignments, 133,609 aligned residue pairs) for training of the neural network, the remaining fifth (220 alignments, 33,825 aligned residue pairs) for testing.

CONSTRUCTION OF THE ENCODING SCHEMES

We use a feed-forward neural network with the logistic transformation function

\[ f(x) = \frac{1}{1 + \exp(x)} \]  

We employ the back-propagation algorithm with the root-mean-squared (RMS) error function. A six-fold cross-validation approach (Bishop, 1995) is used in the training. Each model is randomly initialized and trained 10 times. Only the model with the lowest cross-validation error will be used for further analysis. Details of the training procedure were described in our previous paper (Lin et al., 2001).

After training this neural network, we present each amino acid type to the input layer, propagate the network, and take the values of the hidden units as the encoding of the according amino acid. Here, the size of the hidden layer determines the size of encoding schemes.

The encoding scheme based on the recognition of human signal peptide cleavage sites is obtained from Jagla & Schuchhardt (2000).

We test different encoding schemes on the simulation of substitution matrices using the same cross-validation approach (Lin et al., 2001). For each encoding scheme, we adjust the size of the neural network input layer, translate the input amino acid types to corresponding codes, and perform the same training procedure. Three training sets of different sizes are employed. However, all models are tested on the same testing set, even when the test set is much larger than the training set (Fig. 1).

Results and Discussion

There are different approaches to measure the complexity of a neural network model (e.g. Maass, 1995). In our testing, the size of the hidden layer and output layer of models are specified. A larger encoding scheme directly leads to a larger input layer, more weighted connections and a more complex model.

An important feature of an encoding scheme is its size. Small encoding schemes use only a few numbers to describe types while large schemes employ many units. The simplest scheme tested here has a size of zero: the model completely ignores input amino acid types. The largest one, the orthogonal encoding scheme, utilizes one input unit for each amino acid type. We have tested all intermediate-sized schemes from our neural network models. Smaller schemes bring simpler models, which often perform better on small training sets. Nevertheless, models with two schemes of the same size can have different testing errors because of the different composition of the schemes. We want to find an approach to optimize automatically schemes so that small schemes can most efficiently describe the amino acid types.

With the largest training set, almost all models with size \( \geq 3 \) can achieve good generalization: testing errors are low and differences between training and testing errors are small. Both the training and testing errors decrease with scheme size. With this set, overfitting is far from problematic: the most complex model gives the best performance (Fig. 2(a)). When we change to the smallest training set, all models are overfitted in training: testing errors increase and training
errors decrease. Small models like the model with amino acid encoding scheme from neural network, size 3 (AESNN3) (Table 1) and AESNN4 have the lowest testing errors (Fig. 2(b)). The most complex model, which uses the orthogonal encoding scheme, has the largest difference between its training and testing errors. Although we used the cross-validation approach in the training, this model still suffers overfitting. On the other hand, schemes that are too small (like AESNN1 and AESNN2) are less than adequate for the description of amino acid types. Models with these small schemes perform badly on all training sets. AESNN3 and AESNN4 are recommended: on the largest training set, their testing errors are not much higher than that of the orthogonal encoding; with the smallest set, they are better (Table 2).

Another reason to use small encoding schemes is speed. In training, the model with AESNN3 is about 9 times faster than the model with the orthogonal encoding (data not shown). To propagate a simpler model is slightly faster as well. It can be a considerable factor when we are dealing with huge sequence databases.

By analysing 18 amino acid substitution matrices derived from different procedures, May (1999) gives a list of the reliable residue clusters after hierachical classification of the 20 amino acids. Amino acids are grouped according to relationships confirmed by different matrices. All groupings of amino acids ranked more than four in this list (occurring more than 4 times in different classifications) are clustered to adjacent regions in our AESNN3. This observation strengthens the soundness of our projections.

Here, we present a series of encoding schemes of the amino acid types. They can perform better than the traditional orthogonal encoding on small data sets in the simulation of an amino acid substitution matrix. It can be assumed that for different tasks of sequence analysis, different properties of residues are needed in descriptions (in this problem, AESNN2 performs slightly better than JNS2, the encoding scheme of Jagla & Schuchhardt, 2000). Using this approach, we can develop different encoding schemes.

---

**Table 1**

<table>
<thead>
<tr>
<th></th>
<th>AESNN3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-0.99</td>
</tr>
<tr>
<td>B</td>
<td>-0.61</td>
</tr>
<tr>
<td>C</td>
<td>0.00</td>
</tr>
<tr>
<td>D</td>
<td>0.28</td>
</tr>
<tr>
<td>E</td>
<td>-0.99</td>
</tr>
<tr>
<td>F</td>
<td>-0.24</td>
</tr>
<tr>
<td>G</td>
<td>0.59</td>
</tr>
<tr>
<td>H</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Note: Each amino acid type is described using a three-dimensional vector. Values are taken from the three hidden units from the neural network trained on structure alignments. We linearly transform the values to the range (-1, 1).

---

**Fig. 2.** Training and testing errors of encoding schemes: (a) networks with the largest training sets (size 133, 609 aligned residue pairs) and (b) networks with the smallest sets (size 1350 aligned residue pairs). All testing errors are calculated with the same testing set (size 33, 826 aligned residue pairs). Unit of error is nats [natural digits, unit of Shanon’s entropy (Bishop, 1995)]. (---)Training error; (—)testing error.
Fig. 3. The colouring scheme. Amino acid types are coloured according to AESNN3. For example, in AESNN3 alanine is described as the vector $[-0.99, -0.61, 0.00]$. We linearly transfer values in the scheme from the range $(-1, 1)$ to $[0, 255]$. So, the letter A is coloured in RGB colour $[1, 49, 126]$. Two-dimensional coordinates of letters are from AESNN2.
Table 2

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Training set 1</th>
<th></th>
<th></th>
<th></th>
<th>Training set 2</th>
<th></th>
<th></th>
<th></th>
<th>Training set 3</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Training</td>
<td>Testing</td>
<td></td>
<td></td>
<td>Training</td>
<td>Testing</td>
<td></td>
<td></td>
<td>Training</td>
<td>Testing</td>
<td></td>
</tr>
<tr>
<td>AESNN1</td>
<td>1</td>
<td>2.112</td>
<td>2.118</td>
<td></td>
<td></td>
<td>2.108</td>
<td>2.128</td>
<td></td>
<td></td>
<td>2.046</td>
<td>2.209</td>
<td></td>
</tr>
<tr>
<td>AESNN2</td>
<td>2</td>
<td>1.942</td>
<td>1.940</td>
<td></td>
<td></td>
<td>1.896</td>
<td>1.918</td>
<td></td>
<td></td>
<td>1.849</td>
<td>2.021</td>
<td></td>
</tr>
<tr>
<td>AESNN3</td>
<td>3</td>
<td>1.877</td>
<td>1.878</td>
<td></td>
<td></td>
<td>1.852</td>
<td>1.861</td>
<td></td>
<td></td>
<td>1.729</td>
<td>1.977</td>
<td></td>
</tr>
<tr>
<td>AESNN4</td>
<td>4</td>
<td>1.865</td>
<td>1.863</td>
<td></td>
<td></td>
<td>1.888</td>
<td>1.863</td>
<td></td>
<td></td>
<td>1.739</td>
<td>1.982</td>
<td></td>
</tr>
<tr>
<td>AESNN5</td>
<td>5</td>
<td>1.863</td>
<td>1.855</td>
<td></td>
<td></td>
<td>1.866</td>
<td>1.855</td>
<td></td>
<td></td>
<td>1.729</td>
<td>1.989</td>
<td></td>
</tr>
<tr>
<td>AESNN8</td>
<td>8</td>
<td>1.843</td>
<td>1.836</td>
<td></td>
<td></td>
<td>1.832</td>
<td>1.848</td>
<td></td>
<td></td>
<td>1.688</td>
<td>2.007</td>
<td></td>
</tr>
<tr>
<td>JNS</td>
<td>2</td>
<td>1.992</td>
<td>1.989</td>
<td></td>
<td></td>
<td>1.967</td>
<td>1.974</td>
<td></td>
<td></td>
<td>1.901</td>
<td>2.080</td>
<td></td>
</tr>
<tr>
<td>ORT</td>
<td>2</td>
<td>1.942</td>
<td>1.940</td>
<td></td>
<td></td>
<td>1.896</td>
<td>1.918</td>
<td></td>
<td></td>
<td>1.849</td>
<td>2.021</td>
<td></td>
</tr>
<tr>
<td>Size of training sets</td>
<td>133609 13043 1350</td>
<td>13043</td>
<td>1350</td>
<td>1350</td>
<td>1350</td>
<td>1350</td>
<td>1350</td>
<td>1350</td>
<td>1350</td>
<td>1350</td>
<td>1350</td>
<td>1350</td>
</tr>
</tbody>
</table>

Note: Size—size of encoding schemes; training—cross-validation training error (relative entropy in nats); testing—testing error (relative entropy in nats); AESNN—encoding schemes from our neural network model; JNS—the encoding scheme from Jagla & Schuchhardt (2000); ORT—the orthogonal encoding scheme; size of training sets: number of aligned residue pairs in training sets.

optimized for prediction of protein secondary structure, prediction of contact matrices, etc. But we suggest these encoding schemes based on the simulation of substitution matrices can be used for general purposes.

Figure 3 shows our amino acid colouring scheme. In this colouring scheme, hydrophobic amino acids like methionine, leucine, isoleucine, valine, tryptophan, and phenylalanine are coloured in yellow–green colours. Polar amino acids are coloured in red, blue and purple. Proline is in black. Our colour scheme is automatically constructed according to the evolutionary relationships between amino acids encoded in protein structure alignments. However, it confirms some features identified in previous work of Taylor (1997) and May (1999). Without any arbitrary considerations, it should reflect more precisely properties of amino acids and their evolutionary relationships. We hope that our colouring scheme will be useful for manual analysis of protein alignments. A simple Java program has been written to demonstrate colouring schemes described here and by Taylor (1997). Our encoding schemes, colouring scheme and this program are available on the web at http://mathbio.nimr.mrc.ac.uk/kxlin/aesnn/

REFERENCES


