SEQUENCE ANALYSIS VIA FEED-FORWARD NEURAL NETWORK AND BAYESIAN THEORY

M. Ribeiro-Alves and F.F. Nobre

Biomedical Engineering Department, Alberto Luiz Coimbra Institute, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil

flavio@peb.ufrj.br

Abstract: With its capabilities, artificial neural networks are currently gaining a major focus to solve various problems in genome informatics. We present a search by signal in genome sequence using artificial neural networks and statistical methods. More specifically, we used Bayes inference theory, incorporating prior knowledge for regularization learning, and evidence procedure for training the ANN. The prediction of the consensus sequence is obtained using principles of automatic relevance determination to discard less predictive nucleotides bases. We exemplify the approach in the identification of promoter sequences of *Escherichia coli*. We achieved models with an accuracy of approximately 99% using a very simple network. We predicted both the promoter consensus sequence and a sequence of improbable nucleotides. The former contains 19 nucleotide bases, and the improbable sequence has 35 relevant bases from the 57 present in the original sequence window.

Introduction

Genome informatics, a new field of computational molecular biology, involves, among others, methods for gene recognition, sequence functional analysis and its structure and family determination. Earlier approaches to gene recognition predict individual functional elements, using "gene search by content" or "gene search by signal" methods [1]. The content approach compares a genome sequence, with unknown function, to different sequences with known context features, usually stored in a database. The similarity between them is achieved using a similarity measure, called coding potential. For high correlated sequences this measure indicates the potential of the unknown sequence to code the know context features.

In the "gene search by signal" approach, functional signals represented by local binding sites involved in gene expression, recognized by the ribosomal machinery, such as splice sites, are represented by rules describing the consensus sequence or weight matrices showing the positional agreement of a nucleotide base and its counterpart in the window sequence.

Gene prediction typically has two phases: coding region prediction and determining gene structure, known as gene parsing. Features of individual DNA functional elements and signals are obtained and then combined to form a gene model. The major features of interest are related with the promoters, exons, introns, intergenic regions, splice, and start sites of the genome sequence.

Different methods for coding region prediction have already been reported in the literature, including the use of neural networks [2,3]. One of the first applications of ANN, involved the use of an adapted perceptron learning algorithm to predict ribosomal binding sites in *Escherichia coli* [4].

Artificial neural networks are currently gaining a major focus to solve various problems in genome informatics and molecular sequence analysis [5]. A neural network is characterized by the connections between neurons and its method for determining the connections weights; known as training or learning algorithms. Neural networks learn from examples and exhibit some capabilities for generalization beyond the training data [6]. This latter feature makes this computational model an important tool in applications with small or incomplete understanding of the problem, but where training data is readily available.

This work presents a search by signal in genome sequence analysis using artificial neural networks and statistical methods. More specifically, Bayes inference theory, incorporating prior knowledge for regularization learning, and evidence procedure were used for training the ANN. The prediction of the consensus sequence is obtained using principles of automatic relevance determination to discard less predictive nucleotides bases from the original sequence window. We exemplify the approach in the identification of promoter sequences of the enteric bacteria *Escherichia coli*.

Background

Usually, network training is based on minimization of an error function. Within this framework, a more complex network, for instance with more units in the hidden layer, may produce a better fit to the training data, but when presented with new data its performance is very poor. This is known as the bias-variance trade off. To improve generalization, available data is usually split in training and validation sets, reducing the number of samples for learning. Additionally, the common

ISSN: 1727-1983

training methods results in a single set of connection weights.

The *Bayesian learning* approach uses all the available data for training. Furthermore, this approach considers a probability distribution function over the weight space, providing relative degrees of belief for different sets of the weight vector. This probability function is initially set to some prior distribution and, using observed data, the Bayes' theorem provides the posterior distribution, which is used to evaluate the predictions of the trained network for new values of the input variables.

The posterior density of the weights w of the network, given a data set D, is:

$$p(w \mid D) = \frac{p(D \mid w)p(w)}{p(D)} \tag{1}$$

where $p(w)$ is the prior distribution, $p(D|w)$ is the likelihood, and $p(D)$ is a normalization factor ensuring that the posterior integrates to one.

The prior distribution reflects our initial lack of knowledge of the weight values. Giving the observed data, we update this prior distribution to a posterior distribution using equation (1). This distribution shows what we have learned from the observed data, improving the estimated values for the connection weights.

To obtain the posterior distribution, we need to define expressions for the prior distribution, $p(w)$, and for the likelihood function, $p(D|w)$. We expect the underlying generator of our data sets to be smooth, and the network mapping should reflect this belief. A neural network with large weights may lead to a mapping with large curvature, and so we favor small values for the network weights. Common practice indicates the use of a Gaussian prior distribution, and the requirement for small weights suggests a distribution with zero mean

$$p(w) = \frac{1}{Z_w(\alpha)} \exp(-\frac{\alpha}{2} \| w \|^2) \tag{2}$$

where α represents the inverse variance of the distribution, and $Z_w(\alpha)$ is the normalization constant.

Taking the negative log of this prior, ignoring the normalization constant, we obtain an error term identical to the weight decay described in regularization theory [7].

$$\alpha E_w = \frac{\alpha}{2} \| w \|^2 = \frac{\alpha}{2} \sum_{i=1}^{W} w_i^2 \tag{3}$$

This error term regularizes the weights by penalizing large magnitudes. Using Bayes' theorem (1) and ignoring the normalization term $p(D)$, which does not depend on the weights, we obtain the Bayesian error term for a model given by

$$E = -\log p(w \mid D) - \log p(w) = E_D - \alpha E_w \tag{4}$$

For classification problems, when dealing with two classes, the noise model, E_D, using a cross-entropy error function, is given by [8]

$$p(D \mid w, x) = \prod_{n=1}^{N} y(x^n; w)^{t^n} (1 - y(x^n; w))^{1-t^n} \tag{5}$$

$$E_D(D \mid w, x) = -\sum_{n=1}^{N} \{t^n \ln y(x^n; w) + (1 - t^n) \ln(1 - y(x^n; w))\} \tag{6}$$

It follows that the posterior distribution of the weights has the form

$$p(w \mid D) = \frac{1}{Z_S} \exp(-E_D - \alpha E_W) = \frac{1}{Z_S} \exp(-S(w)) \tag{7}$$

This posterior distribution (7) can be approximated by a Gaussian distribution with mean vector w_{MP}, the most probable weight vector found by optimization of the total error function $E = S(w)$, and inverse covariance A, the Hessian matrix of $S(w)$ evaluated at w_{MP}. The set of parameters $\alpha_1, \ldots, \alpha_g$ can be learned from the training set using the evidence procedure described later in the paper.

For predictions, we compute the integral over the posterior weight distribution. The probability that an input vector x belongs to class C_1, is given by

$$P(C_1 \mid x) = \int P(C_1 \mid x, w) p(w \mid D) dw = \int y(x, w) p(w \mid D) dw \tag{8}$$

If we assume that the posterior is sharply peaked around w_{MP} there is still a problem because the output function $y(a)$, usually a sigmoid function with a being the activation of the output neural unit, is not linear. This means that the prediction is no longer equals the most probable output $y(x, w_{MP})$.

Since a is a linear function of the weights, MacKay [9] assumes that it has a Gaussian distribution with mean w_{MP} and variance $s^2 = g^T A^{-1} g$, where g is the gradient of a with respect to the weights w_{MP}. The integral in (8) does not have an analytic solution, therefore Mackay [9] suggests the approximation

$$P(C_1 \mid x, D) \approx f(k(s) a_{MP}) \tag{9}$$

where

$$k(s) = (1 + \frac{\pi s^2}{8})^{-1/2} \tag{10}$$

and a_{MP} denotes the logistic output a evaluated at w_{MP} .

The *evidence procedure* [9,10] assumes that the posterior density of the hyperparameters $p(\alpha|D)$ is sharply peaked around α_{MP}, the most probable values of the hyperparameters. Initially the hyperparameters values that optimize the weight posterior probability are found, and then all the other calculations involving the weight posterior distribution $p(w|D)$ are performed, which requires another approximation. Mackay [10]

ISSN: 1727-1983

proposed a spherical Gaussian distribution around a particular mode w_{MP}, based on a second-order Taylor series expansion of $S(w)$

$$S(w) \approx S(w_{MP}) + \frac{1}{2}(w - w_{MP})^T A (w - w_{MP}) \qquad (11)$$

where A is the Hessian matrix of the total error function.

The error function $S(w)$ is the negative log probability of the weight posterior density, therefore the later distribution is Gaussian:

$$p(w \mid \alpha, D) = \frac{1}{Z'_S} \exp(-S(w_{MP}) - \frac{1}{2} \alpha w^T A \Delta w) \qquad (12)$$

where $\Delta w = w - w_{MP}$ and Z'_S is the normalization constant for the approximating Gaussian.

From $Z_W(\alpha)$ and $Z'_S(\alpha)$ we can compute the log evidence as follows:

$$\ln p(D \mid \alpha) = -\alpha E_W^{MP} - \frac{1}{2} \ln \| A \| + \frac{W}{2} \ln \alpha - \frac{N}{2} \ln(2\pi) \qquad (13)$$

To optimize this expression for the log evidence with respect to α, Mackay [10] proposed a series of approximations that gives an implicit equation for α

$$2\alpha E_w^{MP} = W - \sum_{i=1}^{W} \frac{\alpha}{\lambda_i + \alpha} \qquad (14)$$

where λ_i is the eigenvalue of the matrix determinant ||A||. Components of the sum for which $\lambda_i >> \alpha$ make a contribution near to 1, while components for which $0 << \lambda_i << \alpha$ make a contribution near to 0. We can view the above summation, we will call it γ, as a measure of the number of well-determined parameters [10].

To apply the evidence procedure in classification problems, we need to convert equation (14) from conditions satisfied by the optimal α into practical estimation methods. The following algorithm describes each step of applying the evidence procedure:

1. Chose initial values for the hyperparameters α. Initialize the weights in the network. These weights need not to be drawn from the prior distribution defined by α;
2. Train the network with suitable optimization algorithm to minimize the misfit function $S(w)$;
3. When the network training has reached a local minimum, the Gaussian approximation can be used to compute the evidence for the hyperparameters. These can be re-estimated with the following formula, derived from equation (14)

$$\alpha^{new} = \frac{\gamma}{2E_W} \qquad (15)$$

These re-estimation formula can be iterated if desired.

4. Repeat steps 2 and 3 until convergence.

Selecting relevant input variables from a large set of possibilities is a common problem in applications of pattern recognition. *Automatic relevance determination* is a method that uses the hyperparameters to define the importance of the inputs. Bayesian framework associates a separate hyperparameter with each input variable. During Bayesian learning it is possible to modify the hyperparameters, finding their optimal value (evidence procedure). Hyperparameter represents the inverse variance of the prior distribution of the weights fanning out from inputs. A small hyperparameter value indicates that large weights are allowed, and we conclude that the corresponding input is important. Large hyperparameter signals means that the weights are closed to zero, and hence the corresponding input is less important [9].

Network outputs represent the probability of a given output to belong to class C_1 .

Materials and Methods

The dataset used here is a public available dataset, obtained from http://www.cs.wisc.edu/~shavlik/mlrg/publications.html, used in works related to the *E. coli* RNA binding sites discovery [11]. It comprises a total of 234 sequences with 57 nucleotide window length sequence (-50–7), that were used as positive examples. The negative examples were generated from the promoter-free λ-phage sequence, consisting of a 4977 base-sequence, which allowed 4921 sliding windows comprising 57-bases without replicates.

Each nucleotide base was represented by a four bit pattern where 1000 encodes adenine (A), 0100 encodes thymine (T), 0010 encodes guanine (G), and 0001 encodes cytosine (C), and the four bits 0000 (X) indicates the absence of that base, resulting in an input dimension of 228.

In previous studies of *E. coli* ribosomal binding sites (promoter) prediction using neural networks, several feed-forward topologies were tested, varying the number of neurons in the hidden layer [11,12]. We have implemented ten neural networks, varying from 1 to 10 neurons in the hidden layer, using a two layer feed-forward network architecture. The original dataset has been divided in two subsets, training and test sets. To determine the optimal ratio of promoter:non-promoter sequences (p:np ratio), we varied the composition of the training set with ratios equals to 1:1, 1:2, 1:5, and 1:10 . For the promoters, we used 70% of the available positive examples. The non-promoter sequences were randomly selected from the generated 4921 λ-phage sequence. The test set was composed by the remaining positive examples and the non-promoter sequences not selected previously.

The most probable weight values of the 40 neural nets tested was obtained using the conjugated scaled gradient with Polak-Ribiere line search direction with error backpropagation [7]. The training was extended until the network error, or the gradient difference between epochs, on the training set reached 10^{-15}. We implemented evidence procedure for determining optimal weight and hyperparameter values with two

ISSN: 1727-1983

iterations cycles. All the initial weights and α hyperparameters, one for each input, were randomly assigned.

Network outputs, varying between 0 and 1, correspond to the probabilities of given nucleotides being a promoter sequence base. Evaluation of the trained networks to discriminate promoters and non promoters was done using two methods. The first one, considered the classical threshold of 0.5, measuring the discriminatory efficacy, using the ratio among all sequences in the test set of correct-predicted sequences in the form

$$P_{CC} = \frac{C_P + C_{NP}}{C_{ALL}} \qquad (16)$$

where C_P is the number of promoter sequences correctly predicted in the test set, C_{NP} is the number of non-promoter sequences correctly predicted, and C_{ALL} is the total number of sequences in the test set.

For the second method, the threshold varied between 0.01 and 0.99, with a step size of 0.01. A series of probabilities of corrected and false alarm classification, respectively P_{CC} and P_{FA}, were used to construct Receiver Operating Characteristic (ROC) curves (Fig.1). The area below each curve is measures the discriminatory efficacy.

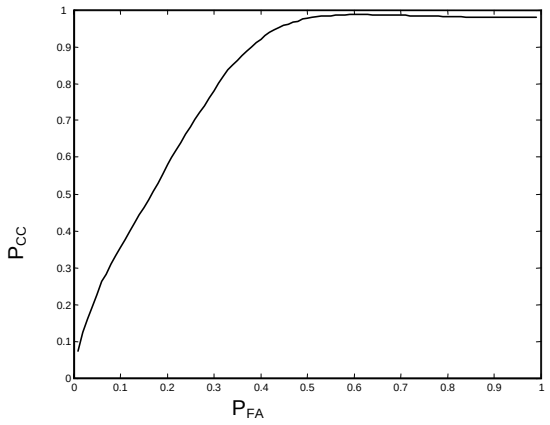

Figure 1: ROC curve for an ANN architecture with 10 hidden nodes, and for an input p:np ratio equals to 1:10.

For consensus sequence prediction, we used the ANN with only one hidden unit. This approach was used because in this case it is possible to provide relationships between output and the weights fanning out from the input layer. Based on the best network with one neuron in the hidden layer, selected using ROC approach, we identify the inputs having related hyperparameter values greater than 10. Since the weights in these cases are not consistent, according to Mackay [9] , we code the associated inputs as 0.

To code the remaining inputs, to recover the four bit pattern of the nucleotides, we assigned 1 for inputs associated with largest positive weight values in each pattern. For patterns having all weights less than zero, we assign 0 to all four bits sequence representing the X character. The consensus sequence was determined by relating these four bit patterns to the corresponding nucleotides coding characters A, T, C, and G.

Largest negative values for each four bit pattern were used to determine improbable nucleotides in a particular base position, generating the most improbable sequence for an *E. coli* promoter.

All analysis in this work was implemented in Matlab software. The routines for neural network design, training, and prediction, used the Netlab Pattern recognition toolbox.

Results

Varying the number of hidden nodes from 1 to 10 and training set composition (p:np equals 1:1, 1:2, 1:5, and 1:10), we ended with 40 neural nets to evaluate. Using the first neural net performance evaluation method (0.5 threshold for the output values), we obtained that neural nets with more informative training set (nets with more negative examples) had better predictive corrected classification values P_{CC}, while, when the network architecture were augmented by inserting hidden neurons, an almost constant P_{CC} was observed, indicating no better classification accuracy (fig.2).

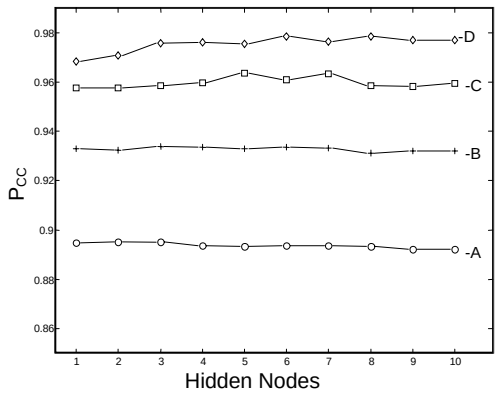

Figure 2: Network performance evaluation with PCC measure of accuracy. Promoter:non-promoter ratios equals: circle, 1:1; cross, 1:2; square, 1:5; and diamond, 1:10.

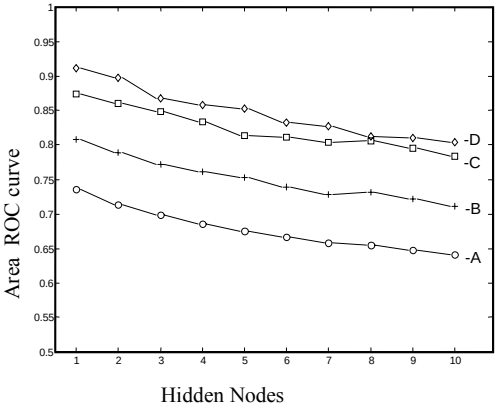

Figure 3: Network performance evaluation with ROC curve area measure of accuracy. Promoter:non-promoter ratios equals: circle, 1:1; cross, 1:2; square, 1:5; and diamond, 1:10.

ISSN: 1727-1983

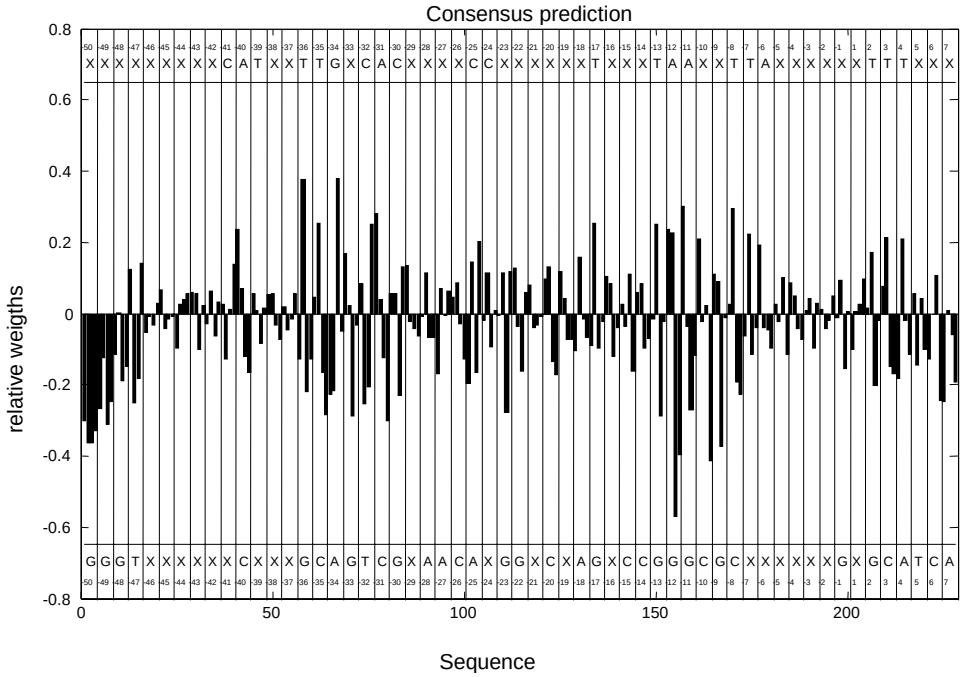

Figure 4: *E. coli* promoter 57-window sequence prediction. Most probable sequence predicted (positive weights), and most improbable sequence (negative weights)

.

The second method of classification was evaluated with ROC curves (fig.3). There is an loss of accuracy with increasing number of hidden neurons. Similarly as the classical method described previously, there is an increased accuracy with more informative training sets.

Using the area below the ROC curve as a criterion for selecting the best neural net with a single neuron in the hidden layer, we extracted consensus sequences for promoters and non-promoters. Rules were extracted from the vectors of trained consistent weights. Figure 4 displays a 57-window sequence. Positive bars are the weights associated with nucleotides of the most probable promoter or consensus sequence. Negative weights are related with improbable promoter sequence.

The consensus sequence contains 19 nucleotide bases and the other coded as X are non-relevant for promoter discrimination. For the improbable sequence we have 35 relevant bases.

Conclusion

In the search by signal approach, functional signals are often represented by consensus sequences or position weight matrices. As exemplified with *E. coli* promoter prediction, the association of artificial neural networks and Bayesian theory has a series of advantages when compared with techniques using error function minimization.

First, we could use all the training dataset without the necessity of splitting in training and validation samples. The Bayesian weight regularization allows input/output mapping generalization, preventing specialization of the network to the training data. This can be of utility for classification models in genomes where the knowledge of positive examples is limited.

Second, since outputs can be interpreted as probability, we have the possibility to use ROC to evaluate the accuracy of different models and select the more robust model to variations in the test dataset. This leads to higher specificity and sensitivity, which is of prime importance when we have a small numbers of known patterns. For samples where the sequenced organism is well known, network architectures with more hidden neurons can provide more accurate results (predictive power greater than 0.99, figs.2 and 3). Also, it became evident in the *E. coli* example used here, that increasing the number of negative samples in the training set, increases the network accuracy.

Third, using automatic relevance determination, yielded in a single analysis, both the promoter consensus sequence and a sequence of improbable nucleotides (fig.4). Excluding non-relevant bases minimizes the effort for constructing rule based classifiers.

Comparison with other works in the literature was not feasible. First, we do not have benchmark datasets, which is a current problem in genome informatics. Second, given the great diversity in the pre-analysis data manipulation, network topologies, and usage of other methodologies, such as multiple sequence alignment, made almost impossible to reproduce a few of them in a single work. Even with this restriction, the results were very enthusiastic. We had an accuracy of approximately 99% using a very simple network; which is easily implemented and avoids the alignment and class differentiation steps.

ISSN: 1727-1983

References

[1] STADEN R. (1984): 'Computer methods to locate signals in nucleic acid sequences'. *Nucleic Acids Res.* **12**, pp. 505–519.

[2] XU Y., MURAL R.J., UBERBACHER E.C. (1994): 'Constructing gene models from accurately predicted exons: an application of dynamic programming'. *Comput Appl Biosci.* **6**, pp. 613–623.

[3] REESE M.G., EECKMAN F.H., KULP D., HAUSSLER D. (1997): 'Improved splice site detection in genie'. *J Comput Biol.* **4**(3), pp. 311–323.

[4] STORMO G.D., SHNEIDER T.D., GOLD L., EHRENFEUCHT A. (1982): 'Use of 'perceptron' algorithm to distinguish translational initiation sites in *E. coli*'. *Nucleic Acids Res.* **10**, pp. 2997–3010

[5] WU C.H., MCLARTY J.W. (2000): 'Neural Networks and Genome Informatics'. (Elsevier).

[6] HAIKIN S. (1998): 'Neural Networks: A Comprehensive Foundation'. (Pearson Education, New York).

[7] BISHOP C. M. (1995): 'Neural Networks for Pattern Recognition'. (Oxford University Press).

[8] HOPFIELD J. J. (1987): 'Learning algorithms and probability distributions in feed-forward and feed-back networks'. *Proc Natl Acad Sci U S A.* **84**(23), pp. 8429–8433.

[9] MACKAY D. J. C. (1992): 'Bayesian interpolation'. *Neural Comp.* **4**, pp. 415–447.

[10] MACKAY D. J. C. (1992): 'A practical Bayesian framework for back-propagation framework'. *Neural Computation* **4**, pp. 448–472.

[11] DEMELER B., ZHOU G. (1991): 'Neural network optimization for *E. coli* promoter prediction'. *Nucleic Acids Res.* **19**, pp.1593–1599..

[12] MAHADEVAN I., GHOSH I. (1994): 'Analysis of *E. coli* promoter structures using neural networks'. *Nucleic Acids Res.* **22**, pp. 2158–2165.

ISSN: 1727-1983