# Recognition of gene regulatory sequences by bagging of neural networks

Gert Thijs[1], Yves Moreau[1], Stéphane Rombauts[2], Bart De Moor[1], Pierre Rouzé[3]

[1]Department of Electrical Engineering, Katholieke Universiteit Leuven,
Kardinaal Mercierlaan 94, B-3001 Leuven, Belgium
[2] Department of Plant Genetics, Flemish Institute for Biotechnology,
[3]Associate Laboratory of the French Institute for Agronomic Research (INRA, France),
[2,3]University of Ghent, K.L. Ledeganckstraat 35, B-9000 Ghent, Belgium

## Abstract

We use an ensemble of multilayer perceptrons to build a model for a type of gene regulatory sequence called a G-box. A variant of the bagging method (bootstrap-and-aggregate) improves the performance of the ensemble over that of a single network. Through a decomposition of the generalization error of the ensemble into bias and variance components, we estimate this error from the hold-out samples of the individual networks. We test the model on putative G-boxes, on sequences upstream of light-regulated genes, and on a control group and demonstrate that the model separates these groups efficiently.

## 1 Regulatory sequences

Genome-sequencing projects have churned out gigabytes of data and their pace is increasing. Millions of sequences made up from a small alphabet (four letters for DNA or RNA data, twenty letters for protein data) are available together with their biological description. Confronted to the amount and complexity of the data, molecular biology has turned to statistical and machine learning techniques to assist in the interpretation of this data. Neural networks are an important technique for such tasks because they can model the complex nonlinear relationships present in biological sequences [1].

The small alphabets describing biological sequences provide a powerful and homogeneous quantification of biological data. Because of this quantitative aspect, many problems from molecular biology are amenable to computational techniques. The two main classes of problems when trying to discover relevant patterns in biological sequences are search-by-content and search-by-signal.

In search-by-content, we try to identify biological regions having a significant extension (for example, more than a few hundred bases [letters]) by modeling their sequential content. Examples of important extended regions in the case of DNA sequences are genes or coding regions in genes (called exons). Methods for solving these problems are hidden Markov models, hybrids of hidden Markov models and neural networks, and stochastic grammars [1].

In search-by-signal, we try to identify smaller regions (typically, a few bases to a few tens of bases) by modeling the statistical correlations between the bases. Examples of important DNA signals are splice sites, which determine the beginning and end of non-coding regions of genes, or the signals controlling the activity of a gene (called regulatory sequences). Neural networks such as multilayer perceptrons [7, 10] (MLPs) and time-delay neural networks [12] have been widely used for these tasks. Such neural networks work as a window of several tens of bases sliding along the DNA sequence. Each base (Adenine, Guanine, Thymine, or Cytosine) is represented by a group of four binary inputs for which one of the inputs is set to one while the others are set to zero (one-out-of-four coding). The neural network predicts whether or not the middle of the sequence corresponds to the middle of the signal region.

Our interest lies with the recognition of regulatory sequences (see [4, 5]) in DNA sequences from plants (with a focus on *Arabidopsis Thaliana*, which is a model organism whose genome will be completely sequenced by 2001). In an organism, genes code the information necessary for the production of one or several proteins by an organism. The main process that sets off, controls, and stops the production of a protein is the binding of some specific proteins to special short sites on the DNA sequence. Such sites are called regulatory sequences. The identification of regulatory sequences is still a difficult problem, especially in plants. A model that would recognize regulatory sequences or at least make reliable guesses about them would be truly useful to biologists.

Most regulatory sequences occur in the few hundred or few thousand bases preceding the beginning of a gene; in this case, they are called upstream regulatory sequences. By focusing on this upstream region, we avoid searching a much larger piece of the genome where regulatory sequences are less likely, thereby preventing many false detections.

While a lot of genomic data is available, the annotation of this data to describe relevant biological features is limited and its overall quality is quite poor. In the framework of other research projects, we compiled a database called PlantCARE [13] of 319 regulatory sequences in plants. For this database, each regulatory sequence was cross-checked against the available literature. This procedure is obviously time consuming but it guarantees that the database is reliable. It contains occurrences of regulatory sequences from different genes in different plants. The number of occurrences for a given type of regulatory sequence is low: between 2 to 20. For the regulatory sequences with the least occurrences, it is impossible to construct a predictive model. But a carefully constructed neural network can deal with the sequences with the most occurrences. As the PlantCARE database grows over time, we expect to build models for more and more regulatory sequences.

We choose to study a regulatory sequence called G-box as it is one of the most frequent motifs in PlantCARE with 18 occurrences. The G-box is involved in the chain of chemical mechanisms directing the response of a plant to light. The most basic model of a regulatory sequence is the consensus sequence. The consensus indicates which base we are most likely to encounter a each position of a window that is highly conserved among the different occurrences. For the G-box, the consensus is `CACGTG`. Out of the 18 G-boxes in PlantCARE, 15 match the consensus exactly and the 3 others

| Accession | Regulatory element |
|---|---|
| K02067 | `CA`*CACATG*`GAA` |
| M12196 | `GC`*CACGTG*`GA` |
| S74719 | `GC`*CACGTG*`TCACTC` |
| X13611 | `TC`*CACGTG*`GCATT` |
| D10661 | *ACACGTG*`TCACC` |

Table 1: Some examples of G-boxes. The first column gives the accession number of a DNA sequence containing the element. The sequence can be retrieved through Internet databases (for example, *http://srs.ebi.ac.uk*). The second column gives the sequence of the regulatory element. Except for the first, all the sequences presented match the consensus `CACGTG`.

match the consensus with a single mismatch (see Table 1).

# 2 Modeling

The main problem we face is the low number of positive examples (18 occurrences). Indeed, for a window size of 50 (which is typical for this type of applications [7]), the neural network will have 200 binary inputs. We can generate negative examples almost certainly by taking windows at random positions in arbitrary sequences because of the low prevalence of G-boxes in the genome. The low number of positive examples makes it difficult to estimate a neural network with good generalization properties. To improve the quality of our model, we decided to use a combination of neural networks trained under different conditions.

There is a great diversity of methods for combining neural networks, a few examples of which are the ensemble method [11] (combining networks trained with different initial conditions), stacking [14] (using the output of different neural networks as inputs to another network), bagging [3] (training networks on bootstrap sample and averaging the network outputs). For this application, we tested a variant of the bagging method (using a hold-out set) because of its simplicity.

The procedure goes as follows in our case. We first construct a training set by taking a window of length 50 around each of the 18 examples of G-boxes and by taking another 360 negative windows at random. To compensate for the fact that there are 20 times more negative examples than positive ones, we use an asymmetric cost function that assigns 20 times more weight to the positive examples than to the negative ones. We train 75 single-layer perceptrons with 4 hidden units using a conjugate gradient algorithm (the faster Levenberg-Marquardt algorithm was too memory hungry).

For each network, we create a bootstrap set by drawing *with replacement* 720 samples from the training set. The samples that do not appear in the bootstrap set are set aside in a hold-out set. We train the neural network on the bootstrap set and stop training when the error reaches a minimum on the hold-out set. Once we have trained the 75 networks, we use the average of the predictions of the 75 networks on a window as the prediction of the ensemble (an alternative is to classify according to a majority vote among the networks, but we have not yet been able to explore this possibility).

Processing the alphanumerical data from the sequences together with the numerical values of the neural network is a tricky problem. We performed the training of the networks in Matlab with the neural network toolbox. To process the sequences, we use Perl because of its powerful string processing features. We have built a software bridge between Matlab and Perl that allows them to directly exchange of variables. We therefore have a flexible scripting environment that gives us a lot of control over our numerical experiments.

# 3 Assessment

We have used all the examples of G-boxes available from PlantCARE for the development of the ensemble. The resulting lack of positive examples makes it difficult to estimate the generalization performance of the ensemble.

## 3.1 Generalization performance

The standard procedure to estimate the generalization error of a model when no independent test set can be kept aside is leave-one-out cross-validation. The procedure goes as follows. We leave out a sample from the training set and train a model on the rest of the data; we then test the model on the sample we left out and keep the resulting error. We repeat this operation for every point in the training set. The average of the errors of all the models is then an estimate of the average error of the model on a previously unseen point generated from the same distribution as the training set. In our case, leave-out-one cross-validation is computationally prohibitive as we would have to train 75 networks for each of the over 378 data points of our training set.

To estimate the generalization error, we chose to use a decomposition of the generalization error of an ensemble of networks into bias and variance components as proposed by Krogh and Vedelsby [9]. Let us assume that we approximate a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ (the setup for this decomposition is more general than classification and bagging only). Suppose the data is generated by a probability distribution $p(x)$. Suppose we have $K$ networks $N_i(x)$. The prediction of the ensemble is

$$\bar{N}(x) = \sum_{i=1}^{K} N_i(x).$$

The quadratic error of the ensemble on the input $x$ is $e(x) = (f(x) - \bar{N}(x))^2$, the error of an individual network is $\epsilon_i(x) = (f(x) - N_i(x))^2$. We now define the *ambiguity* of a network as $a_i(x) = (\bar{N}(x) - N_i(x))^2$. Is is possible to show [9] that

$$e(x) = \frac{1}{K} \sum_i \left(\epsilon_i(x) - a_i(x)\right).$$

Over the probability distribution $p(x)$, we have the generalization error $E$ of the ensemble $\bar{N}$, the generalization error $E_i$ of network $N_i$, and the ambiguity $A_i$ of network $N_i$ with respect to the ensemble $\bar{N}$:

$$\begin{cases} E &= \int_{\mathbb{R}^n} e(x)p(x)dx, \\ E_i &= \int_{\mathbb{R}^n} \epsilon_i(x)p(x)dx, \\ A_i &= \int_{\mathbb{R}^n} a_i(x)p(x)dx. \end{cases}$$

By the linearity of the integral, we then have that

$$E = \frac{1}{K} \sum_i (E_i - A_i).$$

The advantage of this form is that we can estimate the generalization of the ensemble from the generalization of the individual networks. Furthermore, we can estimate the generalization of the individual networks from their hold-out sample. We have adapted this decomposition further to the case of classes with different weights. For network $N_i$, $C_i$ is the number of samples in the hold-out set, $x_{ij}^h$ is the $j$-th sample of the hold-out set, $t_{ij}^h$ is the desired target (zero or one), and $W(t)$ is the weight associated to the target (1 or 20 in our case). Our estimate of the generalization error is then

$$\begin{aligned} E \approx\ & \tfrac{1}{K} \textstyle\sum_{i=1}^{K} \tfrac{1}{C_i} \sum_{j=1}^{C_i} W(t_{ij}^h) \\ & \times \left(t_{ij}^h - N_i(x_{ij}^h)\right)^2 - \left(\bar{N}(x_{ij}^h) - N_i(x_{ij}^h)\right)^2. \end{aligned}$$

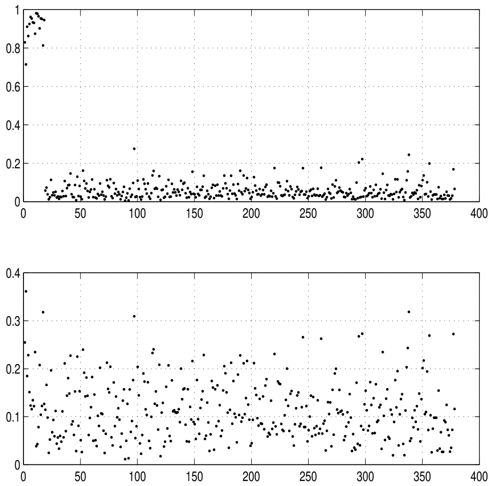


Figure 1: Training results. At the top, the scores of the ensemble on the training set. At the bottom, standard deviation of the outputs of the individual networks. The $x$-axis gives the index of the input pattern. The first 18 patterns are G-boxes while the others are not.

On our data, this estimate for the mean squared error of the ensemble is $E = 0.0096$. Note that the average generalization error of an individual network is $(1/K)\sum_i E_i = 0.0338$, which is more than three times larger than the error of the ensemble. Bagging did clearly improve our modeling.

Another by-product of our ensemble is that we can derive confidence intervals on our prediction. Heskes [8] showed that we get an estimate of a confidence interval for our prediction by multiplying the variance of the ensemble (which we call the ambiguity here) on a given point by a chosen percentage point of a Student's $t$-distribution. We can see the results on the training sets in Figure 1.

### 3.2 Independent test

Because of the positive results on the G-box data from PlantCARE, we test our model further in a more difficult environment. From databases on the Internet, we gather a number of sequences with a reference to a G-box and also a number of genes regulated by light (such genes can potentially be regulated by a G-box, although not necessarily). Furthermore, we create a control set with sequences in which there is normally no G-box. All the sequences have been restricted to the region between 1000 bases upstream from the beginning of the gene and 200 bases within the gene (each test sequence is thus at most 1200 bases long). Note that we have not had the opportunity to analyze the sequences with a reference to a G-box. We expect that a large number of these sequences will actually contain a G-box but we have no guarantee about the individual sequences.

| Sequences | Genes | Consensus |
|---|---|---|
| G-box | 18 | 18 + 195 |
| Light regulated | 20 | 9 + 188 |
| Control | 17 | 8 + 80 |
| 400kbp | 105 | 148 + 827 |

Table 2: Overview of the properties of the different test sets. The second column gives the number of genes found in each set and the third column gives the number of occurrences of the consensus `CACGTG` (without any mismatch plus with a single mismatch).

We also have at our disposal a large sequence of 400.000 bases (400kbp; `Z99707` and `Z99708`) that we have annotated manually. The manual annotation makes this sequence reliable and useful for testing. We can check the significance of a high score for a certain window, if we check the position of the window in the annotation.

Table 2 gives an overview of the properties of the selected sequences. The 400kbp sequence contains 105 genes and some of these genes are regulated by light. There is however no reference to a G-box in the annotation. Searching for the consensus `CACGTG` on the given strand reveals 148 occurrences of the consensus and 827 sites with a single mismatch.

We score first the three test sets (G-box, light regulated, and control) with our ensemble. Our classification of a window will depend on whether its score will be greater than a given threshold. We see that the sequences containing G-boxes reach high scores much more frequently than random sequences (see Figure 2). The same thing holds for the light-regulated sequences. Our ensemble is thus able to separate G-boxes and light regulated sequences from random sequences. Further work will be necessary to assess the predictions of the ensemble over individual windows instead of over a whole sequence.

For this application, we are willing to use high classification thresholds and be conservative. Indeed, the number of windows that we potentially have to score is enormous (a plant genome is at

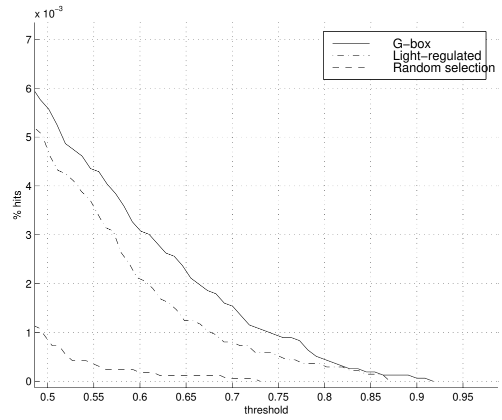


Figure 2: Percentage of hits in the test sequences with respect to a given threshold. The sequences containing G-boxes (solid line) reach high thresholds more frequently than the random sequences (dashed line). The sequences related to light-regulated genes (dotted line) reach high thresholds more frequently than the random sequences but less than G-boxes.

least 100 million bases long) while the number of G-boxes is low. False alarms will mask the true patterns except if we keep the rate of false detections extremely low. This control will come at the cost of missing positive patterns but it is the only option.

A preliminary analysis of the predictions of the ensemble on the 400kbp is encouraging. This analysis has shown that the ensemble frequently predicted G-boxes upstream of known light-regulated genes. Also, some false positive can be easily detected. For example, a high scoring window in the coding region of a gene should be a false positive. In contrast, a cluster of high scoring windows upstream of a gene is a strong indication of regulation by G-boxes. We can use these rules to improve the performance of the model but also to harvest new positive or negative examples for a further development of the model.

# 4 Research directions

The quality of the proposed methodology can now justify deeper biological studies. By assessing the predictions of the current model, it will be possible to generate more examples of G-boxes and more informative examples of sequences that are *not* G-boxes. The negative examples in our current modeling are random windows; these examples do not describe well the decision boundary between positive and negative examples as they are probably scattered fairly far away from this boundary. Difficult negative examples (wrong predictions) are a lot more informative. With this additional data, we will be in a position to devise a powerful model for G-boxes.

Our next step will be to optimize the numerous parameter settings to increase the performance of the ensemble. The most important parameters in the proposed procedure are

1. The length of the input window
2. The total number of negative examples in the data set
3. The number of MLPs in the ensemble
4. The number of hidden units in each MLP
5. The number of samples in the bootstrap set.

We want to find an optimal set of parameters based on the overall performance of the ensemble. But we have to bear in mind the computational limitations when we choose a set of parameters. For instance, the performance will increase with the number of MLPs in the ensemble, but more MLPs make the scoring of a sequence time consuming. The length of the input window and the number of hidden units determine the complexity of each MLP.

Bagging does improve modeling. However, there exists more advanced methodologies where we use a linear combination of networks instead of simply taking their average. Examples of such methods are the generalized ensemble method [11] or ambiguity maximization [9] (both of which try to minimize the correlations among the networks). Another important method is arcing (adaptive resampling and combining) [2] where the networks are generated by sampling preferentially the points where the classification is often wrong (this method is a variation of boosting [6] where we do not assume that the individual networks perform poorly). We plan to investigate these methods to improve our models further.

So far, we have only focused on one particular regulatory element to develop a good procedure. The next phase is to extend this procedure to model other regulatory elements in PlantCARE. To this end, we will have to collect more high quality examples in PlantCARE to train the models.

## 5 Conclusions

We have developed an ensemble of neural networks for the prediction of G-boxes in genomic sequences. By training the ensemble with a variant of bagging that uses a hold-out sample, we decrease the generalization error when compared to an individual network. We estimated the generalization error through a decomposition of the error of the ensemble into bias and variance components because we had no clean test set. When tested on new sequences containing G-boxes or related to light-regulated genes, the ensemble scored much higher than when tested on random sequences. The quality of the model warrants further biological studies and the collection of more data to develop an even better model in the near future.

### Acknowledgments

Bart De Moor is a senior Research Associate with the F.W.O. (Fund for Scientific Research-Flanders). Gert Thijs is a research assistant of the K.U.Leuven. Yves Moreau is a post-doctoral researcher of the K.U.Leuven. This work is supported in part by (1) Concerted Research Action GOA-MIPS (Model-based Information Processing Systems) and (2) Interuniversity Poles of Attraction Programme (IUAP P4-02 (1997-2001): Modeling, Identification, Simulation and Control of Complex Systems. The scientific responsibility is assumed by its authors.

## References

[1] Pierre Baldi and Søren Brunak. *Bioinformatics: The Machine Learning Approach.* A Bradford Book. MIT Press, Cambridge, MA, 1998.

[2] Leo Breiman. Arcing classifiers. Technical Report 460, University of California, Berkeley, 1996.

[3] Leo Breiman. Bagging predictors. *Machine Learning 26*, (2):123–140, 1996.

[4] Philip Bucher. Weight matrix description of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences. *Journal of Molecular Biology*, (212):563–578, 1990.

[5] James W. Fickett and Artemis G. Hatzigeorgiou. Eukaryotic promoter recognition. *Genome Research*, 7:861–878, 1997.

[6] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[7] Stefan M. Hebsgaard, Peter G. Kornig, Niels Tolstrup, Jacob Engelbrecht, Pierre Rouzé, and Soëren Brunak. Splice site prediction in Arabidopsis Thaliana pre-mRNA by combining local and global sequence information. *Nucleic Acid Research*, 24(17):3439–3452, 1996.

[8] Tom Heskes. Practical confidence and prediction intervals. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 176–182. The MIT Press, 1997.

[9] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. The MIT Press, 1995.

[10] Michiel Noorderwier, Geoffrey G. Towell, and Jude W. Shavlik. Training knowledge-based neural networks to recognize genes in DNA sequences. In R. Lippmann, J. Moody, and D. Touretsky, editors, *Advances in Neural Information Processing Systems*, volume 3. Morgan Kaufmann, 1991.

[11] Michael P. Perrone and Leon N. Cooper. When networks disagree: Ensemble method for neural networks. In *Neural Networks for Speech and Image Processing*. Chapman-Hall, 1993.

[12] M. G. Reese and F. H. Eeckman. Time-delay neural networks for eukaryotic promoter prediction. *submitted*, 1998.

[13] S. Rombauts, P. Déhais, M. Van Montagu, and P. Rouzé. PlantCARE, a plant cis-acting regulatory element database. *Nucleic Acids Research*, 27:295–296, 1999.

[14] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.