

Splice Site Prediction Using Artificial Neural Networks

Øystein Johansen, Tom Ryen, Trygve Eftesøl, Thomas Kjosmoen,
and Peter Ruoff

University of Stavanger, Norway

Abstract. A system for utilizing an artificial neural network to predict splice sites in genes has been studied. The neural network uses a sliding window of nucleotides over a gene and predicts possible splice sites. Based on the neural network output, the exact location of the splice site is found using a curve fitting of a parabolic function. The splice site location is predicted without prior knowledge of any sensor signals, like 'GT' or 'GC' for the donor splice sites, or 'AG' for the acceptor splice sites. The neural network has been trained using backpropagation on a set of 16965 genes of the model plant *Arabidopsis thaliana*. The performance is then measured using a completely distinct gene set of 5000 genes, and verified at a set of 20 genes. The best measured performance on the verification data set of 20 genes, gives a sensitivity of 0.891, a specificity of 0.816 and a correlation coefficient of 0.552.

1 Introduction

Gene prediction has become more and more important as the DNA of more organisms are sequenced. DNA sequences submitted to databases are often already characterized and mapped when they are submitted. This means that a molecular biologist has already used genetics and biochemical methods to find genes, promoters, exons and other meaningful subsequences in the submitted material. However, the number of sequencing projects are increasing, and a lot of DNA sequences have not yet been mapped or characterized. Having a computational tool to predict genes and other meaningful subsequences is therefore of great value, and can save a lot of expensive and time consuming experiments for biologists.

This study tries to utilize an *artificial neural network* to predict where the splice sites of a gene can be located. The splice sites are the transitions from exon to intron or from intron to exon. A transition from exon to intron is called a *donor splice site* and a transition from intron to exon is called *acceptor splice site*.

2 Neural Network

The main premise in this study is to use a window of nucleotides that moves stepwise over the sequence to be analysed. The inputs to the neural network

are calculated from the *input calculator*. For each step of the sliding window the neural network will give an output score if it recognizes there is a splice site in the window. A diagram of the entire prediction system is shown in Fig. 2. The window size is chosen to be 60 nucleotides. This is hopefully wide enough to find significant patterns on both sides of the splice site. A bigger window will make the neural network bigger and thereby harder to train. Smaller window would maybe exclude important information around the splice site.

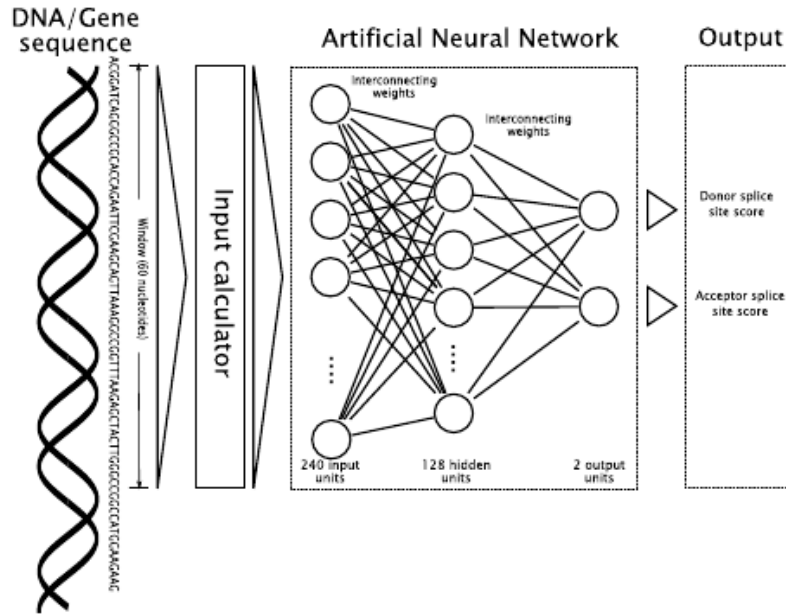


Fig. 1. The connection between the DNA sequence and the neural network system. A sliding window covers 60 nucleotides, which is calculated to 240 input units to the neural network. The neural network feedforward calculates a score for donor and acceptor splice site.

2.1 Network Topology

The neural network structure is a standard three layer feedforward neural network.¹ There are two output units corresponding to the donor and acceptor splice sites, 128 hidden layer units and 240 input units. The 240 input units were used since the orthogonal input scheme uses four inputs each nucleotide in the window. The neural network program code was reused from a previous study, and in this code the number of hidden units was hard coded and optimized for 128 hidden units. There is also a bias signal added to the hidden layer and the output layer. The total number of adjustable weights is therefore $(240 \times 128) + (128 \times 2) + 128 + 2 = 31106$.

¹ This kind of neural network has several names, such as multilayer perceptrons (MLP), feed forward neural network, and backpropagation neural network.

2.2 Activation Function

The activation function is a standard sigmoid function, shown in Eq. 1. The β values for the sigmoid functions are 0.1 for both the hidden layer activation and the output layer activation. Preliminary experiments were performed to test the effect of these values. These tests indicated that 0.1 was a suitable value.

$$f(x) = \frac{1}{1 + e^{-\beta x}} \quad (1)$$

When doing forward calculations and backpropagation, the sigmoid function is called repetitively. It is therefore important that this function has a high computational performance. A fast and effective evaluation of the sigmoid function can improve the overall performance considerably. To improve the performance of the sigmoid function, a precalculated table for the exponential function is used. This table lookup method is known to be very fast, and with an acceptable accuracy.

2.3 Backpropagation

The neural network was trained using a normal backpropagation method as described in Duda, Hart, Stork [3], Haykin [4] or Kartalopoulos [6]. There is no momentum used in the training. We have not implemented any second order methods to help the convergence of the weights.

3 Training Data and Benchmarking Data

Based on data from the The Arabidopsis Information Resource (TAIR) release 8 website [8], we compiled a certain set of genes. TAIR is an on-line database resource of genetic and molecular biology data of the model plant *Arabidopsis thaliana*.

3.1 Excluded Genes

All genes that contain unknown nucleotides were excluded from the data set. In addition, all single exon genes were excluded. Further, all genes with very short exons or introns were excluded. By "short" we mean 30 nucleotides or less. These genes were excluded to avoid very short exons or very short introns. Excluding these genes also simplifies the calculation of desired outputs, since it then can not be more than two splice sites in a window. For genes with alternative splicing, only one splicing variation was kept.

3.2 Training Data Set and Benchmark Data Set

The remaining data set, after exclusion of some genes, consists of 21985 genes. This set is divided into a training data set and a benchmarking data set. The training set and the benchmark set have 16965 and 5000 genes, respectively. The remaining 20 genes, four from each chromosome, was kept for a final verification. The number of genes in each set is chosen such that the benchmark set is large enough to achieve a reliable performance measure of the neural network. This splitting was done at random. Both data sets contains genes from all five chromosomes.

4 Training Method

The neural network training is done using standard backpropagation. This section describes how the neural network inputs were calculated and how the desired output was obtained.

4.1 Sliding Window

For each gene in the training set, we let a window slide over the nucleotides. The window moves one nucleotide each step, covering a total of $L_G - L_W + 1$ steps, where L_G is the length of the gene and L_W is the length of the window. As mentioned earlier, the length of the window is 60 nucleotides in this study.

4.2 Input to the Neural Network

For each nucleotide in the sliding window, we have four inputs to the neural net. The four inputs are represented as an orthogonal binary vector. (A=1000, T=0100, G=0010, C=0001). This input description has been used in several other studies [5], and is described in Baldi and Brunak [1]. This input system is called *orthogonal* input, due to the orthogonal binary vectors. According to Baldi and Brunak [1] this is the most used input representation for neural networks in the application of gene prediction. This input scheme also has the advantage that each nucleotide input is separated from each other, such that no arithmetic correlation between the monomers need to be constructed.

4.3 Desired Output and Scoring Function

The task is to predict splice sites, thus the desired output is 1.0 when a splice site is in the middle of the sliding window. There are two outputs from the neural network: One for indicating acceptor splice site and one for indicating donor splice site.

However, if it is only a 1.0 output when a splice site is in the middle of the window, and 0.0 when a splice site is not in the middle of the window, there will probably be too many 0.0 training samples that the neural network would learn to predict everything as 'no splice site'. This is why we introduce a score function which calculates a target output not only when the splice site is in the middle of the window, but whenever there is a splice site somewhere in the window. We use a weighting function where the weight of a splice site depends on the distance from the respective nucleotide to the nucleotide at the window mid-point. The further from the mid point of the window this splice site is, the lower value we get in the target values. The target values decrease linearly from the mid point of the window. This gives the score function as shown in Eq. 2

$$f(n) = 1 - \left| 1 - \frac{2n}{L_W} \right| \quad (2)$$

If a splice site is exactly at the mid point, the target output is 1.0. An example window is shown in Fig. 2.

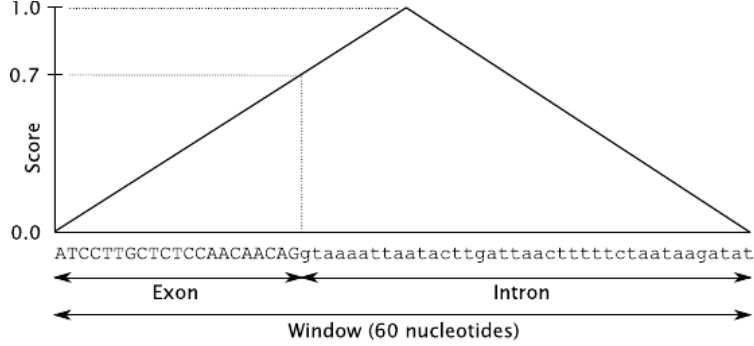


Fig. 2. Score function to calculate the desired output of a sliding window. The example window in the figure has a splice site after 21 nucleotides. This makes the desired output for the acceptor splice site 0.7.

In some cases there may be two splice sites in a window. It is then one acceptor splice site and one donor splice site.

4.4 Algorithm for Training on One Single Gene

The algorithm for training on a single gene is very simple. The program loops through all the possible window positions. For each position, the program computes the desired output for the current window, computes the neural net input and then calls `BACKPROPAGATION()` with these computed data. See Algorithm 1.

Algorithm 1. Training the neural net on one single gene

```

1: procedure TRAINGENE( $NN, G, \eta$ )                                ▷ Train the network on gene  $G$ 
2:    $n \leftarrow \text{length}[G] - L_W$ 
3:   for  $i \leftarrow 0$  to  $n$  do
4:      $W \leftarrow G[i..(i + L_W)]$                                 ▷ Slice the gene sequence
5:      $desired \leftarrow \text{CALCULATEDESIRED}(W)$ 
6:      $input \leftarrow \text{CALCULATEINPUT}(W)$ 
7:     BACKPROPAGATION( $NN, input, desired, \eta$ )
8:   end for
9: end procedure

```

In this algorithm, NN is a composite data structure that holds the neural network data to be trained. G is one specific gene that the neural network should be trained on. The first integer variable n is simply the total number of positions of the sliding window. L_W is the number of nucleotides in the sliding window, which in this study, is set to 60 nucleotides.

The desired output is calculated as described in Section 4.3, and is listed in Algorithm 2.

Algorithm 2. Calculating the desired output based on a given window

```

1: function CALCULATEDESIRED( $W$ )                                ▷ Calculate desired output
2:    $D \leftarrow [0.0, 0.0]$                                        ▷ Initialize the return array
3:    $prev \leftarrow \text{ISEXON}(W[0])$                                 ▷ Boolean value
4:   if  $prev$  then
5:      $j \leftarrow 0$ 
6:   else
7:      $j \leftarrow 1$ 
8:   end if
9:   for  $i \leftarrow 1$  to  $L_W$  do
10:     $this \leftarrow \text{ISEXON}(W[i])$ 
11:    if  $prev \neq this$  then
12:       $D[j] \leftarrow D[j] + 1 - |1 - \frac{i}{30}|$                     ▷ Score function
13:       $j \leftarrow 1 - j$                                          ▷ Flip the index 0 to 1, 1 to 0
14:    end if
15:     $prev \leftarrow this$ 
16:  end for
17:  return  $D$ 
18: end function

```

5 Evaluation Method

The evaluation of a gene is simply the forward calculation performed for all the window positions in that gene. The neural network outputs are accumulated as an indicator value for each nucleotide in the gene.

5.1 Sliding Window

Because the neural network is trained to recognize splice sites in a 60 nucleotides wide window, the forward calculation process is also performed on the same sized window. The window slides over the gene in the same way as in the training procedure.

5.2 Cumulative Output and Normalization

As the sliding window moves over the gene and forward calculates whenever there is a splice site in the window. A nucleotide gets a score contribution from from 60 outputs corresponding to the sliding window passing over it. All these outputs are accumulated.

The accumulated output is then normalized. Most of the nucleotides will get a contribution from 60 different window positions, and these nucleotides are normalized by dividing the cumulative output by the area under the score function (30.0). These normalized cumulative scores are called *acceptor splice site indicator* and *donor splice site indicator*.

5.3 Algorithm for Evaluating One Single Gene

The pseudo code of the evaluation of a gene is given in Algorithm 3. The algorithm contains two loops. The first loop a slides a window over all positions in the gene and adds up all the predictions from the neural network. The second loop normalizes the splice site indicators.

Algorithm 3. Evaluation of gene

```

function EVALUATEGENE( $NN, G$ )                                ▷ Calculate splice site indicators
in: Neural network ( $NN$ ), Gene ( $G$ )
out: Two arrays,  $D$  and  $A$ , which contains the donor and acceptor splice site indi-
      cator.
   $n \leftarrow \text{length}[G] - L_W$ 
  for  $i \leftarrow 0$  to  $n$  do
     $W \leftarrow G[i..(i + L_W)]$                                 ▷ Slice the gene sequence
     $input \leftarrow \text{CALCULATEINPUT}(W)$ 
     $pred \leftarrow \text{EVALUATE}(NN, input)$                         ▷ Gets predicted output
    for  $j \leftarrow i$  to  $L_W + i$  do
       $D[j] \leftarrow D[j] + pred[0]$ 
       $A[j] \leftarrow A[j] + pred[1]$ 
    end for
  end for
  for  $i \leftarrow 0$  to  $\text{length}[G] - 1$  do                        ▷ Normalizing loop
     $D[i] \leftarrow 2D[i]/L_W$ 
     $A[i] \leftarrow 2A[i]/L_W$ 
  end for
  return  $D, A$ 
end function

```

The normalizing loop in the implemented code also takes into account that the nucleotides close to the ends of the gene gradually gets evaluated by less window positions.

6 Measurement of Performance (Benchmark)

For monitoring the learning process and knowing when it is reasonable to stop the training, it is important to have a measurement of how well the neural network performs. This measurement process is also called *benchmarking*.

6.1 Predicting Splice Sites

As mentioned earlier the transition from exon to intron is called a *donor splice site*. The algorithm for predicting exons and introns in the gene is more or less done as a finite state machine with two states – *exon* state and *intron* state. The gene sequence starts in *exon* state. The algorithm then searches for the first high

value in the donor splice site indicator. When the algorithm finds a significant top in the donor splice site indicator, the state switches to *intron*. The algorithm continues to look for a significant top in the acceptor splice site indicator, and the state is switched back to *exon*. This process continues until the end of the gene. The gene must end in the *exon* state.

In the above paragraph, it is unclear what is meant by a *significant top*. To indicate a top in a splice site indicator, the algorithm first finds a indicator value above some threshold value. It then finds all successive indicator data points that are higher than this threshold value. Through all these values, a second order polynomial regression line is fitted, and the maximum of this parabola is used to indicate the splice site. This method is explained with some example data in Fig. 3. In this example the indicator value at 0 and 1 is below the threshold. The value at 2 is just above the threshold and the successive values at 3,4,5 and 6 is also above the threshold and these five values are used in the curve fitting. The rest of the data points are below the threshold and not used in the curve fitting.

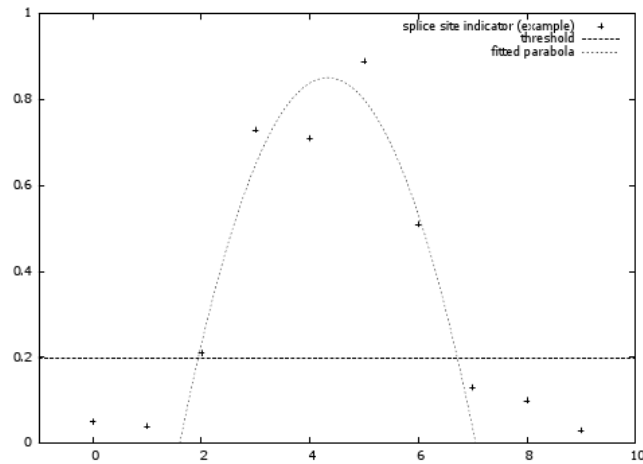


Fig. 3. Predicting a splice site based on the splice site indicator. When the indicator reaches above the threshold value, 0.2 in the figure, all successive data points above this threshold are used in a curve fitting of a parabola. The nucleotide closest to the parabola maxima is used the splice site.

Finding a good threshold value is difficult. Several values have been tried. We have performed some simple experiments with dynamically computed threshold values based on average and standard deviation. However, the most practical threshold value was found to be a constant at 0.2.

6.2 Performance Measurements

The above method is used to predict the locations of the exons and introns. These locations can be compared with the actual exon and intron locations.

There are four different outcomes of this comparison, true positive (TP), false negative (FN), false positive (FP) and true negative (TN). The comparison of actual and predicted location is done at nucleotide level.

The count of each comparison outcome are used to compute standard measurement indicators to benchmark the performance of the predictor. The *sensitivity*, *specificity* and *correlation coefficient* has been the de facto standard way of measuring the performance of prediction tools. These prediction measurement values are defined by Burset and Guigó [2] and by Snyder and Stormo [7].

The sensitivity (Sn) is defined as the ratio of correctly predicted exon nucleotides to all actual exon nucleotides as given in Eq. 3.

$$Sn = \frac{TP}{TP + FN} \quad (3)$$

The higher the ratio, the better prediction. As we can see, this ratio is between 0.0 and 1.0, where 1.0 is the best possible.

The specificity (Sp) is defined as the ratio of correctly predicted exon nucleotides to all predicted exon nucleotides as given in Eq. 4.

$$Sp = \frac{TP}{TP + FP} \quad (4)$$

The higher the ratio, the better prediction. As we can see, this ratio is between 0.0 and 1.0, where 1.0 is the best possible.

The correlation coefficient (CC) combines all the four possible outcomes into one value. The correlation coefficient is defined as given in Eq. 5.

$$CC = \frac{(TP \times TN) - (FN \times FP)}{\sqrt{(TP + FN)(TN + FP)(TP + FP)(TN + FN)}} \quad (5)$$

6.3 The Overall Training Algorithm

The main loop of the training is very simple and is an infinite loop with two significant activities. First, the infinite loop trains the neural network on all genes in the training data set. Second, it benchmarks the same neural network on the genes in the benchmark data set. There are also some other minor activities in the main loop like reshuffling the order of the training data set, saving the neural network, and logging the results. The main training loop is shown in Algorithm 4.

In this algorithm, NN is the neural net to be trained, T is the data set of genes to be used for training and B is the data set of genes for benchmarking. In this algorithm the learning rate, η , is kept constant. The bm variable is a composite data structure to hold the benchmarking data.

The subroutine `SAVE()` saves the neural network weights, and the `SHUFFLE()` subroutine reorders the genes in the data set. `LOGRESULT()` logs the result to the terminal window and to a log file.

Algorithm 4. Main training loop

```

1: procedure TRAIN( $NN, T, B, \eta$ )                                ▷ Train the neural network
2:   repeat
3:     for all  $g \in T$  do                                         ▷ Train neural net on each gene in dataset
4:       TRAINGENE( $NN, g, \eta$ )
5:     end for
6:     SAVE( $NN$ )
7:     SHUFFLE( $T$ )                                                 ▷ A new random order of the training set
8:     for all  $g \in B$  do
9:       BENCHMARK( $NN, g, bm$ )
10:    end for
11:    LOGRESULT( $bm$ )
12:  until break                                                    ▷ Manually break when no improvement observed
13: end procedure

```

7 Experiments and Results

The training data set of 16965 genes were then used to train a neural network. The training was done in three sessions, and for each session we chose separate, but constant, learning rates. The learning rate, η , was chosen to be 0.2, 0.1, and 0.02, respectively. For each epoch² through the training data set, the neural networks performance was measured with the benchmark data set.

7.1 Finding Splice Sites in a Particular Gene

The splice site indicators can be plotted for a single gene. To illustrate our results, we present an arbitrarily chosen gene, AT4G18370.1. The curves in Fig. 4 represent the donor and acceptor splice site indicators for an entire gene. The donor splice sites are marked using a red line, the acceptor splice sites using a green line, and the predicted and actual exons are marked with the upper and lower dashed lines, respectively. The shown indicators are computed using a neural network which has been trained for about 80 epochs, with a learning rate of 0.2. As noted in the header of Fig. 4, the prediction on this gene achieves a better than average CC of 0.835. The results are promising. Most splice sites match the actual data, and some of the errors are most likely due to the low-pass filtering effect of using a sliding window, causing ambiguous splice sites.

7.2 Measurements of the Best Neural Networks

The best performing neural network, achieved a correlation coefficient of 0.552. The correlation coefficients, as well as the sensitivity, specificity, and standard simple matching coefficient (SMC), are shown in Tab. 1. When calculating these performance measurements, the benchmark algorithm averages the sensitivities and specificities for all genes in the data set. In addition the specificity, sensitivity, and correlation coefficient for the entire dataset is reported.

² An *epoch* is one run through the data set of training data.

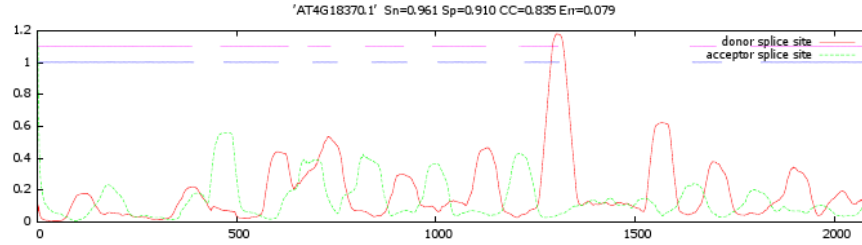


Fig. 4. The splice site indicators plotted along an arbitrary gene (AT4G18370.1) from the verification set. Above the splice site indicators, there are two line indicators where the upper line indicates predicted exons, and the other line indicates actual exons. The sensitivity, specificity and correlation coefficient of this gene is given in the figure heading. (*Err* is an error rate defined as the ratio of false predicted nucleotides to all nucleotides. $Err = 1 - SMC$.)

Table 1. Measurements of the neural network performances for each of the three training sessions. Numbers are based on a set of 20 genes which are not found in the training set nor the benchmarking set.

Session	Average		All nucleotides in set			
	<i>Sn</i>	<i>Sp</i>	<i>Sn</i>	<i>Sp</i>	<i>CC</i>	<i>SMC</i>
$\eta = 0.20$	0.864	0.801	0.844	0.802	0.5205	0.7761
$\eta = 0.10$	0.891	0.816	0.872	0.806	0.5517	0.7916
$\eta = 0.02$	0.888	0.778	0.873	0.777	0.4978	0.7680

8 Conclusion

This study shows an artificial neural networks used in splice site prediction. The best neural network trained in this study, achieve a correlation coefficient at 0.552. This result is achieved without any prior knowledge of any sensor signals, like 'GT' or 'GC' for the donor splice sites, or 'AG' for the acceptor splice sites. Also note that some of the genes in the data sets did not store the base case for splicing, but an alternative splicing, which may have disturbed some of the training. It is fair to conclude that artificial neural networks are usable in gene prediction, and the method used, with a sliding window over the gene, is worth further study. This method combined with other statistical methods, like General Hidden Markov Models, would probably improve the results further.

References

1. Baldi, P., Brunak, S.: Bioinformatics, The Machine Learning Approach, 2nd edn. MIT Press, Cambridge (2001)
2. Burset, M., Guigó, R.: Evaluation of gene structure prediction programs. *Genomics* 34(3), 353–367 (1996)
3. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, New York (2001)

4. Haykin, S.: Neural Networks, A Comprehensive Foundation, 2nd edn. Prentice-Hall, Englewood Cliffs (1998)
5. Hebsgaard, S., Korning, P.G., Tolstrup, N., Engelbrecht, J., Rouzé, P., Brunak, S.: Splice site prediction in *Arabidopsis thaliana* pre-mRNA by combining local and global sequence information. *Nucleic Acids Research* 24(17) (1996)
6. Kartalopoulos, S.V.: Understanding Neural Networks and Fuzzy Logic. IEEE Press, Los Alamitos (1996)
7. Snyder, E.E., Stormo, G.D.: Identifying genes in genomic DNA sequences. In: DNA and Protein Sequence Analysis. Oxford University Press, Oxford (1997)
8. The Arabidopsis Information Resource, <http://www.arabidopsis.org>