AN EFFECTIVE PROMOTER DETECTION METHOD USING THE ADABOOST ALGORITHM [*]

XUDONG XIE

Department of Electronic Engineering, City University of Hong Kong, Hong Kong
Department of Electronic Engineering, Tsinghua University, Beijing, China

SHUANHU WU

Department of Electronic Engineering, City University of Hong Kong, Hong Kong
School of Computer Science and Technology, Yantai University, China

KIN-MAN LAM

Department of Electronic and Information Engineering,
The Hong Kong Polytechnic University, Hong Kong

HONG YAN

Department of Electronic Engineering, City University of Hong Kong, Hong Kong
School of Electronic and Information Engineering, University of Sydney,
NSW 2006, Australia

In this paper, an effective promoter detection algorithm, which is called PromoterExplorer, is proposed. In our approach, various features, i.e. local distribution of pentamers, positional CpG island features and digitized DNA sequence, are combined to build a high-dimensional input vector. A cascade AdaBoost based learning procedure is adopted to select the most “informative” or “discriminating” features to build a sequence of weak classifiers. A number of weak classifiers construct a strong classifier, which can achieve a better performance. In order to reduce the false positive, a cascade structure is used for detection. PromoterExplorer is tested based on large-scale DNA sequences from different databases, including EPD, Genbank and human chromosome 22. The proposed method consistently outperforms PromoterInspector and Dragon Promoter Finder.

1 Introduction

In the past decade, many reliable methods have been developed for protein-coding regions prediction [1] in human genome annotation. However, for regulatory regions of genes, exact promoter detection still remains a challenge, and relatively few algorithms have been proposed to tackle the problem. A promoter is the region of a genomic sequence which is close to a gene’s *transcription start site* (TSS), and it largely controls the biological activation of the gene [2]. Therefore, promoter detection can be considered a fundamental and important step in gene annotation.

[*] This work is supported by research grants from City University of Hong Kong (Projects 9010003 and 9610034).

In order to discriminate a promoter region from non-promoters regions, many different features are considered, such as CpG islands [3, 4], TATA boxes [5, 6], CAAT boxes [5, 6], some specific *transcription factor binding sites* (TFBSs) [5, 6, 7], pentamer matrix [8] and oligonucleotides [9]. And also, various pattern recognition technologies are adopted for classifying, e.g. neural networks [3, 5, 6, 8], linear and quadratic discriminant analyses [4, 7], interpolated Markov model [6], *independent component analysis* (ICA) [10, 11] and *non-negative matrix factorization* (NMF) [11]. The experimental results and analyses in [12] show that selection of the right biological signals to be implemented in promoter prediction programs still remains an open issue. In fact, none of these signals can cover all promoter representations, and each feature abstracted from promoter sequences has its own limitation.

2 Feature Extraction from DNA Sequence

In our method, we consider three different kinds of features, i.e. local distribution of pentamers, positional CpG island features and digitized DNA sequence, which are described in the following sections respectively.

2.1. *Local Distribution of Pentamers*

In our method, we select pentamers as input features. For an input DNA sequence, a set of pentamers a_i, $i = 1, 2, \ldots, W$, can be obtained, where the maximal value of W is $4^5 = 1024$. In order to select the most informative pentamers for discriminating promoters and non-promoters, we consider the posterior probability of I given a_i, $P(I \mid a_i)$, where I is an indicator which equals 1 when the input sequence is a promoter, otherwise $I = 0$. If $P(I=1 \mid a_i) > P(I=0 \mid a_i)$, the input sequence should be a promoter with a higher probability, and *vice versa*. Define

$$\eta = \frac{P(I=1 \mid a_i)}{P(I=0 \mid a_i)}, \qquad i = 1,2,\cdots W, \tag{1}$$

and compute the value of η for each pentamer. According to the Bayes' Theorem, we have

$$P(I=1 \mid a_i) = \frac{P(a_i \mid I=1)P(I=1)}{P(a_i)}, \qquad i = 1,2,\cdots W, \tag{2}$$

and

$$P(I=0 \mid a_i) = \frac{P(a_i \mid I=0)P(I=0)}{P(a_i)}, \qquad i = 1,2,\cdots W\,. \tag{3}$$

From Eq. (1) ~ Eq. (3), we can obtain that

$$\eta = \frac{P(a_i \mid I=1)P(I=1)}{P(a_i \mid I=0)P(I=0)}, \qquad i = 1,2,\cdots W\,. \tag{4}$$

Considering $P(I=1)$ and $P(I=0)$ are constant, we define η as following equation, i.e.

$$\eta = \frac{P(a_i \mid I=1)}{P(a_i \mid I=0)}, \qquad i = 1,2,\cdots W, \quad (5)$$

and then the pentamers are ranked according to their η values. The pentamers which have the highest 250 η values are selected to combine a pentamer set *Pset*.

In order to solve the small sample problem, all pentamers in *Pset* are considered as one class, and the others as another class. In other words, 1024 pentamer patterns are converted into two kinds of patterns: pentamers in *Pset* and pentamers out of *Pset*. For each position of a DNA sequence, not only the pentamer at the position concerned, but also the pentamers within its neighborhoods are considered. A window of 51 bp moves across the sequence at 1 bp intervals and the number of pentamers in *Pset* within this window is taken as a feature at the center of the window. Therefore, for a DNA sequence with a length l, the number of features, which represent local distributions of pentamers in *Pset*, is $l - 4$.

2.2. *Positional CpG Island Features*

CpG islands are regions of DNA near and in the promoter of a mammalian gene where a large concentration of phosphodiester-linked cytosine (C) and guanine (G) pairs exist. The usual formal definition of a CpG island is a region with at least 200 bp and with a GC percentage greater than 50% and with an observed/expected CpG ratio greater than 0.6 [13]. CpG islands can be used to locate promoters across genomes [2, 3, 4]. The most widely used CpG island features are GC percentage (*GCp*) and observed/expected CpG ratio (*o*/*e*), which are defined as follows:

$$GCp = P(C) + P(G), \quad (6)$$

and

$$o/e = \frac{P(CG)}{P(C) \times P(G)}, \quad (7)$$

where $P(CG)$, $P(C)$ and $P(G)$ are percentages of CG, C and G in a DNA sequence, respectively.

GCp and *o*/*e* are two global features for G+C rich or G+C related promoters. However, for the promoters that are G+C poor, CpG island features cannot be used to predict the position of a promoter. It is a reasonable assumption that there are some short regions, which are G+C rich, in a G+C poor promoter sequence, and then these regions can be used for promoter detection. In other words, if we consider *GCp* and *o*/*e* a sequence of local features instead of global features, more promoters can be described based on CpG islands. Similar to pentamer feature extraction, a sliding window 51 bp in length is used, and *GCp* and *o*/*e* are calculated for each window. Then, for an l-length DNA sequence, the number of extracted positional CpG island features is $2l - 8$.

2.3. *Digitized DNA Sequence*

Beside the local distribution of pentamers and the positional CpG island features, we also adopt the digitized DNA sequence as input features. In our method, each nucleotide is represented using a single integer as given by: A = 0; T = 1; G = 2; and C = 3.

From the discussion above, we can see that for an l-length input DNA sequence, the number of extracted features, including local distribution of pentamers, positional CpG island features and digitized DNA sequence, is $l - 4 + 2\,l - 8 + l = 4l - 12$. These features are concatenated to form a high-dimensional vector, and then a cascade AdaBoost learning algorithm is used for feature selection and classifier training for promoter detection.

3 Feature Selection and Classifier Training with AdaBoost

AdaBoost (Adaptive Boosting) is a boosting algorithm [14], which runs a given weak learner several times on slightly altered training data, and combines the hypotheses to one final hypothesis, in order to achieve higher accuracy than the weak learner's hypothesis would have [15]. The main idea of AdaBoost is that each example of the training set should act a different role for discrimination at different training stage. The examples, which can be easily recognized, should be considered less in the following training; while the examples, which are incorrectly classified in previous rounds, should be paid more attention to. In this way, the weak learner is forced to focus on the informative or the “difficult” examples of the training set. The importance of each example is represented by a weight.

As discussion in Section 2, for an input DNA sequence with a length l, the number of features extracted is $N = 4l - 12$, e.g. if $l = 250$, then $N = 988$. We assume that of these features, only a small number are necessary to form an effective strong classifier. We therefore define our weak classifier as follows:

$$h_j(\mathbf{X}) = \begin{cases} 1 & \text{if } x_j > \theta_j \\ -1 & \text{otherwise} \end{cases} \qquad j = 1,2,\cdots,N, \tag{8}$$

where $\mathbf{X}$ is an input feature vector, x_j is the j^{th} feature of $\mathbf{X}$, and θ_j is a threshold. Suppose we have a set of training samples: $(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_m, y_m)$, where $\mathbf{X}_i \in \mathbf{X}$ and $y_i \in \mathbf{Y} = \{1, -1\}$ (‘1’denotes positive examples and ‘–1’ is used for negative examples). In order to create a strong classifier, the following procedure is used:

1. Initialize the weights for each training example:

$$w_{1,i} = \begin{cases} \dfrac{1}{2N^+} & y_i = \ \ 1 \\ \dfrac{1}{2N^-} & y_i = -1 \end{cases} \qquad i = 1,2,\cdots,m \ , \tag{9}$$

where N^+ and N^- are the number of positives and negatives respectively.

2. For $t = 1, \ldots, T$:
   1. For each feature x_j, train a classifier $h_j(\mathbf{X})$, which implies selecting the optimal θ_j to produce the lowest error. The error for the classifier h_j considers all input samples with the condition of θ_j, which is defined as $\varepsilon_j = \sum_{i=1}^{m} w_t \left[y_i \neq h_j(\mathbf{X}_i) \right]$.
   2. Find the classifier $h_t : \mathbf{X} \to \{1,-1\}$ that minimizes the error with respect to the distribution w_t: $h_t = \arg\min_{h_j \in \mathrm{H}} \varepsilon_j = \sum_{i=1}^{m} w_t \left[y_i \neq h_j(\mathbf{X}_i) \right]$. Here $\varepsilon_t = \min_{h_j \in \mathrm{H}} \varepsilon_j$ should be larger than 0.5.
   3. Update the weights of the examples for the next round $w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$, where e_i = 0 if example $\mathbf{X}_i$ is correctly classified, otherwise e_i = 1, and $\beta_t = \dfrac{\varepsilon_t}{1-\varepsilon_t}$.
   4. Normalize the weights to make w_{t+1} a probability distribution: $w_{t+1,i} = \dfrac{w_{t+1}}{\sum_{j=1}^{m} w_{t+1,j}}$.

After T iterations, the resulting strong classifier is:

$$h(\mathbf{X}) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(\mathbf{X}) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ -1 & \text{otherwise} \end{cases} , \tag{10}$$

where $\alpha_t = \log \dfrac{1}{\beta_t}$. The procedure described above not only selects features, which produce the lowest error ε_t when computing weak classifiers, but also trains the weak classifiers and the combined strong classifiers, i.e. the optimal values of θ_j, w_t and α_t are determined based on the training set.

For an input DNA sequence, the number of non-promoter segments is much larger than the number of promoters. Therefore, it is best to remove as many non-promoter segments from consideration as possible early on. Then we can cascade our classifiers to filter out most of the non-promoters, where a number of strong classifiers are used. In the early stage, few features, or weak classifiers, are considered, which can rapidly filter out most of the non-promoters while maintain most of the promoters. In the later stages, increasingly more complex features are adopted. For each stage, total positive samples (promoters) and only the negative samples (non-promoters), which are incorrectly classified in the previous stage, are used for training. In our method, a five-layer cascade is used, and for each strong classifier, the number of weak classifiers is 10, 20, 50, 100 and 200, respectively.

4 Experimental Results

In this section, we will evaluate the performance of the proposed algorithm, namely PromoterExplorer, for promoter detection based on different databases. The training set is from the Eukaryotic Promoter Database (EPD), Release 86 [16], and the testing databases include EPD, six Genbank genomic sequences and human chromosome 22 (http://www.sanger.ac.uk/HGP/Chr22/).

For training, the positive samples are 2,426 promoter sequences in EPD, which are from 200 bp upstream to 50 bp downstream of the TSS. The negative samples are randomly extracted 11,515 sequences of 250 bp, which are out of the range [–1000, 1000] relative to the TSS locations. In our experiments, all training sequences are constructed only by A, T, G and C; in other words, the sequence which includes the letter ‘N’ is excluded.

When perform testing, an input DNA sequence is divided into a set of segments of 250 bp, which are overlapped each other with a 10 bp shift. As described in Section 2, features including the local distribution of pentamers, the positional CpG island features and the digitized DNA sequence are obtained, following a cascade AdaBoost for classification. From Eq. (10), if the final output is larger than zero, a TSS candidate is marked. Those TSS candidates, which have no more than 1,000 nucleotides apart from their closest neighboring prediction, should be merged into a cluster. Then a new TSS prediction is used to represent this cluster, which is obtained by averaging all TSS candidates within the cluster. In order to fairly compare the performance of PromoterExplorer with other methods, a similar merging mechanism is also adopted for PromoterInspector and DPF. As the criteria proposed in [12], when one or more predictions fall in the region [-2000, +2000] relative to the reference TSS location, a true positive is counted; otherwise the predictions are denoted as false positives. When the known gene is missed by this count, it represents a false negative. Sensitivity (S_n) and specificity (S_p) are two criteria widely used to evaluate the performance of promoter prediction program, which are defined as following:

$$S_n = \frac{TP}{TP + FN}\,, \tag{11}$$

$$S_p = \frac{TP}{TP + FP}\,, \tag{12}$$

where *TP*, *FP* and *FN* denote number of true positives, false negative and false negative, respectively. For DPF, the values of S_n can be preset which is used to control the predictions. In our algorithm, the sensitivity can be modulated by the number of TSS candidates within a cluster. For each cluster to be merged, if the number of TSS candidates within this cluster is larger than a threshold, the merged TSS prediction is considered a true prediction; otherwise, the cluster is removed from the output. Various thresholds will result in different outputs.

4.1. *Experimental Results Based on EPD*

In this section, we will test the PromoterExplorer based on the whole 2,541 vertebrate promoters in EPD, which include a total of 40,656,000 base pairs in the genome sequences. The sensitivity- specificity curve is shown in Figure 1. We can see that when the sensitivity is 68.5%, the specificity is about 68.6%. In this case, the average of the distance between the predicted TSS and the real TSS location is about 320. This result is better than the performances evaluated in [12], where no program simultaneously achieves sensitivity and specificity >65%.

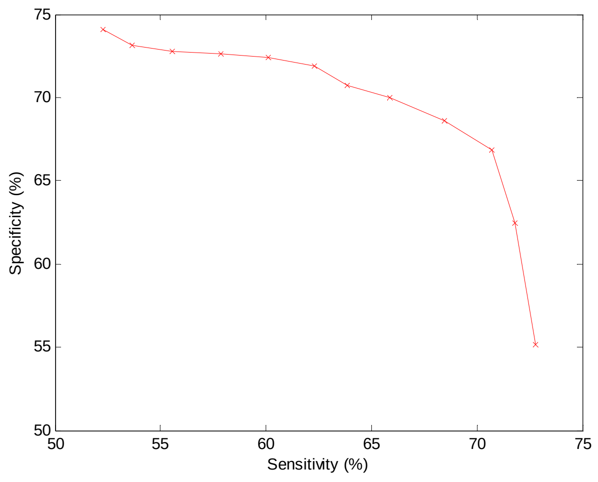

Figure 1. The sensitivity-specificity curve based on EPD using PromoterExplorer.

4.2. *Experimental Results Based on Genbank*

We also evaluate PromoterExplorer on another test set, which contains six Genbank genomic sequences with a total length of 1.38 Mb and 35 known TSSs in these sequences. In Figure 2, the sensitivity-specificity curves of PromoterExplorer, PromoterInspector and DPF are shown.

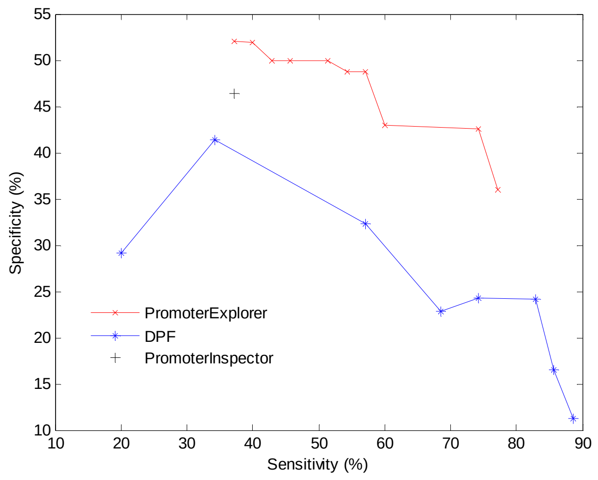

Figure 2. The sensitivity-specificity curves based on Genbank.

From Figure 2, we can see that PromoterExplorer outperforms others. When the sensitivity of the prediction is about 35%, the specificity of PromoterExplorer, PromoterInspector and DPF is about 52.0%, 46.4% and 41.4%, respectively; also, the corresponding average distance between the predicted TSS and real TSS is 467, 472, and 486, respectively.

4.3. *Experimental Results Based on Human Chromosome 22*

Finally, we evaluate PromoterExplorer on Release 3 of the human chromosome 22, which includes 34,748,585 base pairs and 393 known genes. The comparative experimental results are shown in Figure 3. Similar to the observation in Section 4.2, PromoterExplorer performs better than PromoterInspector and DPF. The average distance between the predicted TSS and real TSS for PromoterExplorer, PromoterInspector and DPF is 306 (S_n=63.9), 351 (S_n=63.6), and 315 (S_n=67.7), respectively.

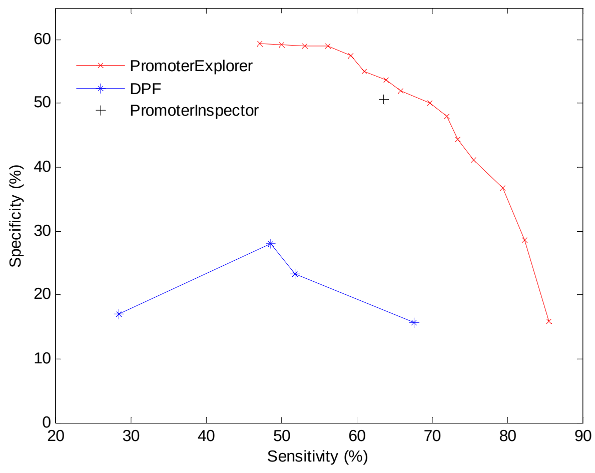

Figure 3. The sensitivity-specificity curves based on Human Chromosome 22.

5 Conclusions

In this paper, we have proposed an effective promoter detection algorithm, which is called PromoterExplorer. In our approach, different kinds of features, i.e. local distribution of pentamers, positional CpG island features and digitized DNA sequence, are extracted and combined. Then a cascade AdaBoost algorithm is adopted to perform feature selection and classifier training. An advantage of our algorithm is the most "informative" features in different classifying stages can be selected for classification. PromoterExplorer is tested based on large-scale DNA sequences, which are from different databases. It has superior performance to existing techniques. Our method can achieve a balance between the sensitivity and specificity of the predictions; therefore it can be used to detect unknown prompter locations in a new DNA sequence.

References

1. Claverie J.M., Computational methods for the identification of genes in vertebrate genomic sequences. *Human Molecular Genetics*., 6:1735-1744, 1997.
2. Pedersen A.G., Baldi P., Chauvin Y., Brunak S., The biology of Eukaryotic promoter prediction: a review. *Computers & Chemistry*, 23:191-207, 1999.
3. Bajic V.B. and Seah S.H., Dragon gene start finder: an advanced system for finding approximate locations of the start of gene transcriptional units. *Genome Research*, 13:1923-1929, 2003.
4. Davuluri R.V., Grosse I., Zhang M.Q., Computational identification of promoters and first exons in the human genome. *Nature Genetics*, 29:412-417, 2001.

5. Knudsen S., Promoter2.0: for the recognition of PoIII promoter sequences. *Bioinformatics*, 15:356-361, 1999.
6. Ohler U., Liao G.C., Niemann H., Rubin G.M., Computational analysis of core promoters in the *Drosophila* genome. *Genome Biology*, 3(12):RESEARCH0087, 2002.
7. Solovyev V.V. and Shahmuradov I.A., PromH: promoters identification using orthologous genomic sequences. *Nucleic Acids Research*, 31:3540-3545, 2003.
8. Bajic V.B., Chong A., Seah S.H., Brusic V., An intelligent system for vertebrate promoter recognition. *IEEE Intelligent Systems Magazine*, 17(4):64-70, 2002.
9. Scherf M., Klingenhoff A., Werner T., Highly specific localization of promoter regions in large genomic sequences by PromoterInspector: a novel context analysis approach. *Journal of Molecular Biology*, 297:599-606, 2000.
10. Matsuyama Y. and Kawamura R., Promoter recognition for E. coli DNA segments by independent component analysis. *Proceedings of the Computational Systems Bioinformatics Conference*, 2004:686-691, 2004.
11. Hiisila H. and Bingham E., Dependencies between transcription factor binding sites: comparison between ICA, NMF, PLSA and frequent sets. Proceedings. *IEEE International Conference on Data Mining*, 4:114-121, 2004.
12. Bajic V.B., Tan S.L., Suzuki Y., Sugano S., Promoter prediction analysis on the whole human genome. *Nature Biotechnology*, 22:1467-1473, 2004.
13. Gardiner-Garden M. and Frommer M., CpG islands in vertebrate genomes. *Journal of molecular biology*, 196(2):261-282, 1987.
14. Duda, R.O., Hart, P.E., and Stork, D.G., Pattern Classification, Second Edition, John Wiley & Sons Inc., 2001.
15. Freund, Y. and Schapire, R.E., A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119-139, 1997.
16. Schmid, C.D., Périer R., Praz, V., Bucher, P., EPD in its twentieth year: towards complete promoter coverage of selected model organisms. *Nucleic Acids Research*, 34:82-85, 2006.