全ての２残基間の相関を考慮したSplice Siteのモデリング

津田 宏治　　有田 正規*
産総研 生命情報科学研究センター
東京都江東区青海 2-41-6
(* 科技団さきがけ)

Modeling Splice Sites with All Pairwise Correlations

Koji Tsuda and Masanori Arita*
AIST Computational Biology Research Center (CBRC)
2-41-6, Aomi, Koto-ku, 135-0064, Tokyo, Japan
(* also at PRESTO, JST)
{koji.tsuda,m-arita}@aist.go.jp

abstract　本稿では、確率分布の Bahadur 表現を用いて、Splice Site の予測を行う方法を提案する。この手法は、従来のマルコフモデルを用いる方法に比べて、全ての残基間の相関を考慮に入れる点に特長がある。学習にかかる計算コストは、サンプル数を n、各サンプル内の残基数を m とすると、$O(m^2n)$ であり、SVM などに比べ非常に高速である。また。予測精度を、ヒトの DNA で測定した結果、従来法に比べて優れた結果を得た。

A new method for splice site prediction is introduced. It approximates the multiple correlations among residuals in splice sites using all pairwise correlations, and its computational cost for learning is $O(m^2n)$ where n is the number of examples, each of length m. We show by computational experiments that its prediction accuracy is superior than previously reported Markov models for splice sites' prediction in human genomic DNA.

Introduction　In eukaryotic genes, splice sites exhibit specific signals to be recongnized by snRNA-proteins, which cleave intron segments from pre-mRNAs. Since the cleaved product forms actual protein coding regions, the computer recognition of these sites constitutes a crucial part in gene finding systems [1], which are the heart of comparative genomics.

Two types of splicing junctions are commonly referred as an acceptor (3' splice site) and a donor (5' splice site) site. Donor sites correspond to the beginning of introns and contain GU, while acceptor sites are the ending of introns, cotaining AG (GU-AG rule). The present knowledge on these sites in vertebrate is the rough consensus patterns: 5'–AG↓GUAAGU–3' for donor sites, and 5'–PyPyPyPyPyPyNCAG↓–3' for acceptor sites [2] (A down-arrow denotes the site of cleavage. Py denotes pyrimidine base U or C, and N denotes any of four bases).

The mainstream of splice sites' prediction has been first-order Markov models. Surprisingly, a rather good result was already obtained with a linear-chain Markov model [3]. Its prediction accuracy could be only slightly improved with a more general, tree-shaped Markov model [4], or with a higher order decision tree model [5]. These results suggest that Markov models are not sufficient for improving splice site recognition.

In order to model distant multiple correlations, we adopt a more general model called 'Boltzmann machine' in neural network community. In its original form, however, the model requires huge computational cost for training [6]. For this reason, Boltzmann machine has been practically inapplicable for learning from large data, typically obtained from genomic sequences.

This paper introduces the approximation scheme of Boltzmann machine to obtain a suboptimal but acceptable solution. We model the probability distribution with Bahadur expansion [7], which is an representation of the probability distribution into different levels of correlations. When it is truncated at the second order, it gives a good estimate of the original probability distribution with a reasonable amount of computation: The cost of our method is $O(m^2n)$, where n is the number of examples, each of length m.

We will show in computational experiments that its prediction accuracy is better than first-order Markov models, including the tree shaped variant, for splice sites' prediction. The tree-shaped Markov model is obtained by computing the correlations between states, finding the maximum spanning tree by linking states of high correlations, and computing the conditional probability for each linked states (thus Markovian). Throughout the paper, we compare our model with two Markov models: linear

chain Markov model, and its tree-shaped variant. Note that the latter model was shown to produce as good results as the higher order Markov model by Burge and Karlin [4].

Maximum Entropy Modeling Let $\boldsymbol{x} = (x_1, \cdots, x_n)$, $x_i \in \{0, 1, 2, \cdots, S-1\}$ denote a sequence of length n with S alphabets. For DNA sequence, the number of alphabet S is 4 and the residuals A,C,G,T are coded into integers 0,1,2,3 respectively. The purpose of probabilistic modeling is to obtain the underlying probability distribution of sequences $p^*(\boldsymbol{x})$ from finite number of examples $x_1, \ldots, x_N$ [8]. One simple way is to construct the empirical distribution $\hat{p}(\boldsymbol{x}) = \frac{1}{N}\sum_{k=1}^{N} I(\boldsymbol{x} = \boldsymbol{x}_k)$ where I denotes the indicator function whose value is 1 if the equation holds and 0 otherwise. However, except when the number of samples is close to infinity, $\hat{p}$ gives a very poor estimate of p^*. Typically, $\hat{p}$ gives a very sparse distribution whose value is 0 for most sequences. Thus, we have to smooth out $\hat{p}$ to obtain a better estimate. In terms of information theory, this "smoothing" amounts to maximizing the entropy of distribution.[1] In order to obtain a good estimate, it is important to determine how far the distribution is smoothed. Pushing it to its limit, for example, it ends up with a truely randomized distribution with the maximum entropy. For this purpose, one must add constraints on p to prevent the entropy from going too large. Typically, these constraints are derived from training samples. This probabilistic modeling technique is called "Maximum Entropy Modeling"(MEM) [9].

The moment constraints (i.e. mean and correlation) are often used in MEM. Let $\hat{m}_{si}$ and $\hat{C}_{si,tj}$ denote the mean and correlation of examples: $\hat{m}_{si} = \frac{1}{N}\sum_{k=1}^{N} I(x_{ki} = s)$, $\hat{C}_{si,tj} = \frac{1}{N}\sum_{k=1}^{N} I(x_{ki} = s)I(x_{kj} = t)$. When we constrain the moments to coincide with those of training samples, the maximum entropy modeling is formulated as follows: *Find p which maximizes $-\sum_{\boldsymbol{x}} p(\boldsymbol{x}) \log p(\boldsymbol{x})$ such that $\sum_{\boldsymbol{x}} I(x_i = s)p(\boldsymbol{x}) = \hat{m}_{si}$, $\sum_{\boldsymbol{x}} I(x_i = s)I(x_j = t)p(\boldsymbol{x}) = \hat{C}_{si,tj}$ where $\sum_{\boldsymbol{x}}$ denote the sum over every possible $\boldsymbol{x}$.* It is known [9] that the optimal p belongs to the parametric family

$$p(\boldsymbol{x}) = \exp\Big(-\varphi + \sum_{i=1}^{n}\sum_{s=0}^{S-1}\theta_{is}I(x_i = s) + \sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{s=0}^{S-1}\sum_{t=0}^{S-1} w_{si,tj}I(x_i = s)I(x_j = t)\Big)$$

where φ is a normalization constant determined such that $\sum_{\boldsymbol{x}} p(\boldsymbol{x}) = 1$. When parameters θ and w are determined such that the mean and correlation satisfy the constraints, $p(\boldsymbol{x})$ gives the maximum entropy distribution. This parametric model of distribution is known as Boltzmann machines [6] in neural network community. However, it is known that the exact determination of θ (in other words "training of the Boltzmann machine") takes exponential time with respect to the sequence length n [6]. So, we need to apply some approximation to obtain a suboptimal but acceptable solution.

Bahadur Expansion For approximated training of Boltzmann machines, we will adopt the Bahadur expansion of probability distribution [7, 10]. The aim of Bahadur expansion is to expand the logarithm of the empirical distribution $\hat{p}(\boldsymbol{x})$ into different levels of correlations, as in the Fourier transform. When this expansion is truncated up to the second order, we obtain a suboptimally trained Boltzmann machine, which gives good results in practice. In this section, we will only describe the method enough for the readers to reproduce our results. Detailed theoretical discussions are available in other literatures [7].

First, let us define several notations to describe the Bahadur expansion. Define a factorized function $q_{[1]}(\boldsymbol{x})$ as

$$q_{[1]}(\boldsymbol{x}) = \prod_{s=0}^{S-1}\prod_{i=1}^{n}[\hat{m}_{si}]^{I(x_i=s)}[1-\hat{m}_{si}]^{I(x_i\neq s)} \quad (1)$$

Also, for each x_i, define a function

$$f_{si}(x_i) = (I(x_i = s) - \hat{m}_{si})/\sqrt{\hat{m}_{si}(1-\hat{m}_{si})}.$$

By the Bahadur expansion, $\log p(\boldsymbol{x})$ is expanded as

$$\log\hat{p}(\boldsymbol{x}) \approx \log q_{[1]}(\boldsymbol{x}) + c + \sum_{s,i} d_{si}f_{si}(x_i) + \sum_{\{s,i\}\neq\{t,j\}} e_{si,tj}f_{si}(x_i)f_{tj}(x_j), \quad (2)$$

where $0 \le s, t \le S-1$ and $1 \le i, j \le n$. Also, the coefficients are determined as follows:

$$c = \sum_{\boldsymbol{x}} q_{[1]}(\boldsymbol{x})\{\log\hat{p}(\boldsymbol{x}) - \log q_{[1]}(\boldsymbol{x})\}$$

$$d_{si} = \sum_{\boldsymbol{x}} f_{si}(x_i)q_{[1]}(\boldsymbol{x})\{\log\hat{p}(\boldsymbol{x}) - \log q_{[1]}(\boldsymbol{x})\}$$

$$e_{si,tj} = \sum_{\boldsymbol{x}} f_{si}(x_i)f_{tj}(x_j)q_{[1]}(\boldsymbol{x})\{\log\hat{p}(\boldsymbol{x}) - \log q_{[1]}(\boldsymbol{x})\}$$

These coefficients include the sum over every possible $\boldsymbol{x}$, but assuming that $\hat{p}(\boldsymbol{x}) > 0$, we can compute c efficiently as

$$c = \frac{1}{N}\sum_{k=1}^{N}\frac{q_{[1]}(\boldsymbol{x}_k)}{\hat{p}(\boldsymbol{x}_k)}\{\log\hat{p}(\boldsymbol{x}_k) - \log q_{[1]}(\boldsymbol{x}_k)\} \quad (3)$$

[1] The entropy of distribution p is defined as $-\sum_{\boldsymbol{x}} p(\boldsymbol{x})\log p(\boldsymbol{x})$.

Table 1: Learning results of 1174 donor sites, tested on 1140 positive, 222089 negative samples. Window size is 15 bases (including GT), beginning from -5 position of the exon-intron boundary.

	truncated Bahadur		tree Markov		chain Markov	
Training FN (%)	True test FN (%)	False test FP (%)	True test FN (%)	False test FP (%)	True test FN (%)	False test FP (%)
0	0.1754	77.99	**0.08772**	**76.12**	**0.08722**	82.51
1	2.018	**18.68**	2.456	26.18	**1.667**	25.39
2.5	3.772	**13**	3.947	19.57	**3.509**	17.6
5	**5.614**	**10.55**	5.877	14.54	6.842	12.46
10	**10.61**	**7.301**	12.63	8.967	11.14	8.807
20	**20.35**	**4.155**	23.16	4.845	21.32	4.803
25	**26.05**	**3.229**	29.56	3.674	27.54	3.543
30	**31.58**	**2.487**	33.16	3.034	33.25	2.775

Table 2: Learning results of 1166 acceptor sites, tested on 1211 positive and 321845 negative samples. Sequence length is 15 bases (including AG), beginning from -10 position of the intron-exon boundary.

	truncated Bahadur		tree Markov		chain Markov	
Training FN (%)	True test FN (%)	False test FP (%)	True test FN (%)	False test FP (%)	True test FN (%)	False test FP (%)
0	0.2477	**60.3**	**0**	74.89	0.2477	60.46
1	**0.7432**	36.08	1.321	38.86	0.9909	**33.87**
2.5	**2.477**	**25.83**	2.642	30.05	2.23	26.85
5	4.707	**19.6**	5.945	21.62	**4.955**	19.82
10	9.001	**13.07**	11.23	15.44	**10.07**	13.26
20	**20.07**	**6.46**	22.71	8.645	22.3	6.544
25	**25.76**	5.201	27.09	7.119	27.99	**5.149**
30	32.37	**3.959**	**31.38**	5.69	33.61	4.081

Other coefficients d and e can be computed in the same way.

Let us define $\log p_{[2]}(\boldsymbol{x})$ as the rightside of (2) without higher orders. Typically, $p_{[2]}$ is not a probability distribution, that is, $\sum_{\boldsymbol{x}} p_{[2]}(\boldsymbol{x}) \neq 1$. However, when normalized, $p_{[2]}$ belongs to the parametric family of Boltzmann machines. Although $p_{[2]}$ does not give the exact maximum entropy solution, the truncated Bahadur expansion $p_{[2]}$ gives a useful approximation as will be shown in the following experiments.

Computational Costs In learning with the Bahadur expansion, three coefficients c, d, e in (2) need to be computed. It takes $O(m^2 n)$ computational cost, where m and n are the sequence length and the number of examples, respectively. Note that computing a log-likelihood $\log p(\boldsymbol{x})$ for each test example takes $O(m^2)$ cost. The costs for other methods are as follows: The linear chain Markov model takes $O(mn)$ for learning and $O(m)$ for testing (i.e. computing a log-likelihood). Also, the tree-shaped Markov model takes $O(m^2 n)$ for learning and $O(m)$ for testing. Thus, the learning cost of our method is the same as that for tree-shaped Markov model, except for the quadratic cost for testing. This quadratic cost is indispensable as long as all pairwise correlations are considered, meaning our computational cost is optimal. In an ordinary situation, this quadratic term can be tolerated, since m is relatively small (e.g. $m = 15$ in our experiment). When the number of examples n is very large, $O(n)$ term dominates the computation. Note that other complicated learning methods may take much larger cost: For example, in SVM [11], a quadratic programming problem of $n \times n$ coefficient matrix must be solved for learning.

Experiments A set of 2314 donor and 2377 acceptor sites were extracted as positive samples from 462 human genes. The set was randomly partitioned into 50% training and 50% testing data: 1174 training and 1140 testing for donors, and 1166 training and 1211 testing for acceptors. Each dataset was fed to the truncated Bahadur expansion model, a tree-shaped Markov model, and a chain Markov model, respectively. The occurrence of GT/AG sites which are not splicing junctions were used as negative samples.

Table 1 shows the comparison of prediction accuracies for donor sites, consisting of 15 bases begin-

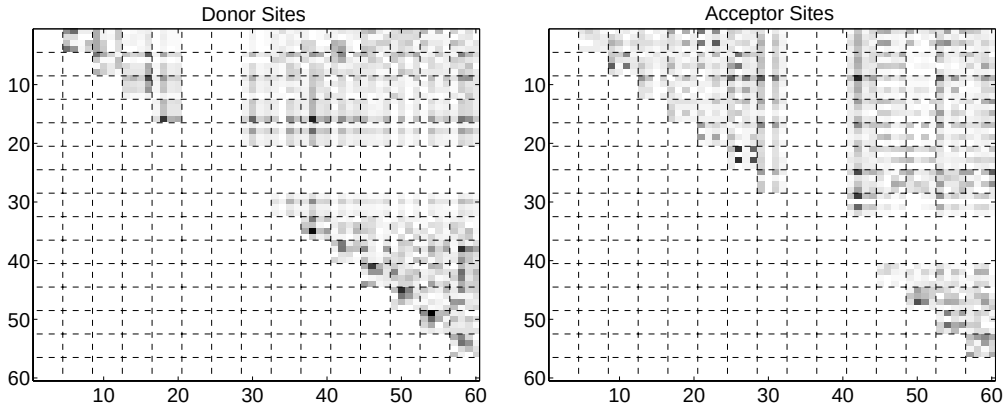

Figure 1: The absolute parameter value $|w_{si,tj}|$ which illustrates the contribution of $I(x_i = s)I(x_j = t)$. The X, Y-axis correspond to the indices t, j and s, i as $x = 4(j-1)+t+1$ and $y = 4(i-1)+s+1$, respectively. The dotted line shows the boundary between residuals, and the blank bands correspond to the fixed residuals GT or AG. A dark point shows high contribution to the probability distribution.

ning from -5 position of the exon-intron boundary. Table 2 shows the accuracies for acceptor sites, consisting of 15 bases beginning from -10 position of the intron-exon boundary. Each table shows the percentage of false positives and false negatives in the testing sequences. Each testing sequence was classified by the log likelihood computed in each model, according to the threshold determined by the ratio of false negatives in training samples. The tables are formatted as in the article by Cai *et al.* [4] The best scores among three models are shown in **bold** letters. In both tables, the truncated Bahadur model shows relatively better results over the two Markov counterparts. The prediction accuracy of Markov models for false negatives are consistent with previously reported results.

In order to see that our method actually utilizes distant correlations, we plot the absolute parameter value $|w_{si,tj}|$ in Fig.1. Note that $w_{si,tj}$ is derived from $e_{si,tj}$ as

$$w_{si,tj} = e_{si,tj}\sqrt{\hat{m}_{si}(1-\hat{m}_{si})}\sqrt{\hat{m}_{tj}(1-\hat{m}_{tj})}.$$

The value $|w_{si,tj}|$ illustrates the contribution of $I(x_i = s)I(x_j = t)$ in the obtained probability distribution. From this figure, we can read that many distant pairs have high contribution both in donor and accepter sites. Thus, distant correlations play an important role in our method, whereas the contributions are limited to neighboring residuals in linear-chain Markov model.

Acknowledgments We would like to thank K. Asai, O. Gotoh, T. Tanaka and S. Akaho for valudable discussions.

References

[1] D. Kulp D. Haussler M.G. Reese and F.H. Eeckman. A generalized hidden markov model for the recognition of human genes in dna. In *Proceedings of the 4th International Conference on Intelligent Systems for Molecular Biology (ISMB-96)*, pages 134–141. AAAI Press, 1996.

[2] T.A. Brown. *Genomes.* BIOS Scientific Publishers, 1999.

[3] S. Salzberg. A method for identifying splice sites and translational start sites in eukaryotic mRNA. *Comput. Appl. Biol. Sci.*, 13:365–376, 1997.

[4] D. Cai, A. Delcher, B. Kao, and S. Kasif. Modeling splice sites with Bayes networks. *Bioinformatics*, 16(2):152–158, 2000.

[5] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic dna. *J. Mol. Biol.*, 268:78–94, 1997.

[6] J. Hertz, A. Krogh, and R.G. Parmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[7] R.R Bahadur. A representation of the joint distribution of responses to n dichotomous items. In H. Solomon, editor, *Studies in Item Analysis and Prediction*, pages 158–168. Stanford University Press, 1961.

[8] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[9] T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley & Sons, 1991.

[10] R.M. Losee. Term dependence: Truncating the bahadur lazarsfeld expansion. *Information Processing & Management*, 30(2):293–303, 1994.

[11] A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering Support Vector Machine Kernels That Recognize Translation Initiation Sites. *BioInformatics*, 16(9):799–807, September 2000.