Bayesian Network: A Bayesian Statistics Based Probabilistic Graphical Model for Prediction of Eukaryotic Genetic Splice Sites

1st Ziwen Zhao
College of Life Science and Technology
Huazhong University of Science and Technology
Wuhan, China
justn582@gmail.com

***Abstract*—Splice sites are a vitally important gene sequence pattern amongst the functional sites inside a eukaryotic gene. However, splice site prediction does not come easy thanks to the extreme complexity of human genome. In this paper, we use a probabilistic graphical model which considers long range interdependency among nucleotide sequences by learning a direct acyclic graph (DAG), finding all of the conditional probabilities for each of the variables which will be used to search for missing labels of the testing sequences. We discussed several structural learning and parametric learning methods and chose the fittest one with the best type I error found. We evaluated our model with some other approaches on the renowned Kulp & Reese human genome dataset, during which we found the limitations of its applications and made an analysis and elaboration for Bayesian networks. The source code is available on GitHub and can be obtained from https://github.com/Newiz430/SplicePredictor.**

***Index Terms*—splice site, Bayesian network, direct acyclic graph, PC algorithm, maximum likelihood algorithm, long range interdependency**

I. Introduction

GENE finding by computational methodologies, the foundation for all further investigation in functional genomics, has attracted considerable research attention since the 20th century [4]. With the thriving of functional genomics after the completion of Human Genome Project (HGP), functions of the elements of eukaryotic gene sequences were beginning to surface. Researchers came to realize that DNA sequences, other than genes, contain a huge amount of information, most of which is correlated with the structural features of nucleic acids and in general determines the DNA - protein, or DNA - RNA interactions. Such information is mostly provided by a variety of functional sites (i.e. sequence motif). The splice sites are a vitally important eukaryotic gene sequence pattern amongst all these sites. As terminal points of RNA splicing, splice sites label the junction of transcript exons and introns, assisting biologists in identifying and positioning the coding sequence within a gene. Splicing, itself, also influences the structure and function of genes, which makes genes more "modular", allowing new combinations of exons to be created during evolution. Furthermore, new exons can be inserted into old introns, creating new proteins without disrupting the function of the old gene [5]. Hence the discovery and

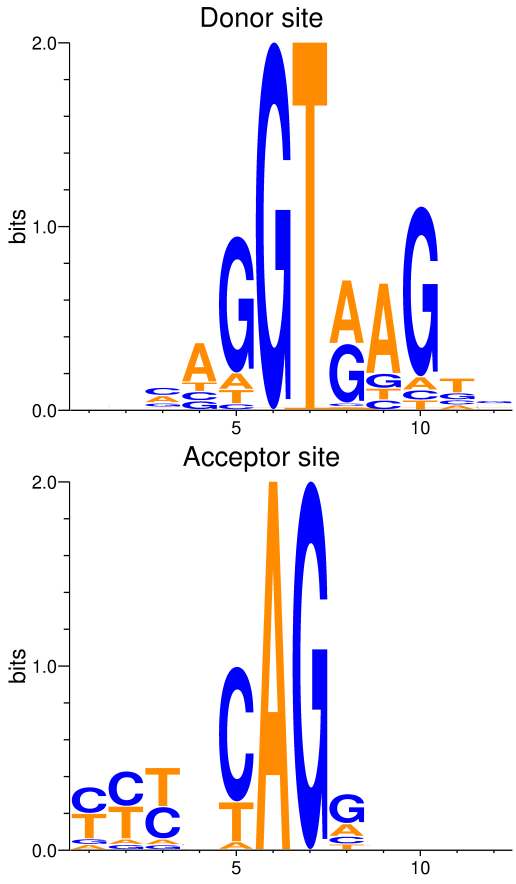

Fig. 1. Base distribution represented as sequence logos [1] for eukaryotic gene splice sites (5 upstream sites, 2 conservative sites & 5 downstream sites): **top**, donor site; **bottom**, acceptor site. The most conserved sites is revealed within logo pictures: for donor sites, GT (6, 7); acceptor sites, AG (6, 7). Despite these two, distribution of adjacent sites (5, 8, 9, 10 for the donor, 5 for the acceptor) appear to have some consistency, too. The polypyrimidine tract can also be observed upstream of the acceptor site (1, 2, 3). The information content at a certain point R_i is given by $R_i = \log_2 4 - (H_i + e_n)$ where $H_i = -\sum_{i=1}^{4} P_i(\log_2 P_i)$ is the Shannon entropy [2], using bit as the basic unit. Higher the bits, higher the relative frequency of that base [3].

prediction of splice sites are of great significance for genetic selective expression research.

A splice site locates in the edge of an intron, including a donor site (5' end of the intron) and an acceptor site (3' end of the intron). As a typical sequence motif, the donor site includes an almost invariant sequence GU at the 5' end of the intron, within a larger, less highly conserved region. The splice acceptor site at the 3' end terminates the intron with an almost invariant AG sequence [6]. Some sections of the intron foretell the positions of these two sites. For example, a fragment of sequence upstream of the acceptor consisting of cytosines and thymines, which is called a polypyrimidine tract [7]. Fig. 1 shows the base distributions adjacent of the splice donor sites and acceptor sites.

As a matter of fact, accurate prediction does not come easy thanks to the extreme complexity of human genome. On one hand, the number and length of exons and introns in a eukaryotic gene exhibit great uncertainty. One eukaryotic gene contains 5.48 exons with 30 - 36 bps long on average. While the longest exon in the human genome is 11555 bp long, several exons have been found to be only 2 bp long [8]. On the other, the existence of alternate splicing make it harder to predict [6]. In this paper, we apply a Bayesian method for gene functional site finding to predict eukaryotic gene splice sites, and prove its feasibility.

A. Related Work

Several typical computational methods that attempt to predict eukaryotic splice sites from unknown gene sequences (i.e. *ab initio* prediction) have been proposed previously.

Frequency-based methods count the nucleotide frequencies of each site via multiple sequence alignment, etc. and work out the log-odds ratio to compare and find conservative sections in the alignment results. Rodger, et al. (1983) [9] proposed a computational model using a weight matrix to represent each type of recognition sequence. A weight matrix is a two dimensional array of values that represent the score for finding each of the possible sequence characters at each position in the target signal. The Weight Matrix Model (WMM) now becomes deprecated owing to its poor accuracy and its independence assumption, that is, WMM only takes point-wise base distribution into consideration, regardless of the potential dependence between adjacent points which is more conformable to the realistic situations. Zhang et al. (1993) [10] optimized the weight matrix to weight arrays which take into account the correspondence between current position and an adjoining position, which is certified conducive to promote accuracy of splice site prediction, but this is still relying on the hypothesis that the intrinsic interdependency only exists between adjacent sites. To deal with the predicament, Burge, et al. (1997) [4] explored a maximal dependence decomposition method and developed the software GENSCAN which identifies complete exon / intron structures of genes, which performs better but still generates many false positive results.

Supervised learning methods learn a model from existing training set which is able to identify the most effective pattern automatically. Duan, et al. (2008) [11] developed the support vector machine (SVM) for position-specific residue preference feature prediction which determines the second structure of double helices. Ryen et al. (2008) [12] introduced the artificial neural network (ANN) in this area and trained the model with backpropagation, which can make predictions without prior knowledge of any sensor signals. Accurate and efficient learning approaches they are, supervised learning methods are heavily dependent on the mass and quality of training sets. Models may not be improved and a computational resource waste may happen when an unbalanced dataset or one with too many noises is provided. For SVMs, kernel function selection is a tricky problem, and neural networks acquire a suitable framework and initial hyperparameters.

B. Contributions

In this work, we use a probabilistic graphical model which considers long range interdependency among nucleotide sequences by learning a direct acyclic graph (DAG), finding all of the conditional probabilities for each of the variables which will be used to search for missing labels of the testing sequences. Our contributions are listed as follows.

- We implemented the Bayesian network by Pgmpy, Python [13] using the given KR set and estimated its performance on the BG set, referring to the existing experiment by Chen, et al. (2005) [14]
- We assessed the influence of different type I errors on the construction of DAG and performance on the given dataset.
- We did a comparison study among WMM, WAM, 3 kinds of SVM and our network to show the performance (also limitations) of Bayesian networks.

II. Methods

Our method is illustrated in Fig. 2, which mainly contains three parts. A DAG structure is learned from the training data during the "structural learning" step, the conditional probability frames of variables are calculated during the "parametric learning" step and sequences of the testing set are scored by the network at the "predicting" step. See below for more detailed presentation.

A. Hypothesis

We assume conservation around the functional splice sites through the entire experiment. This is the fundamental premise for fundamental site finding methods. As for data, we assume that everything about the splice site pattern remained unknown (including the obligatory sites GT / AG) until we dug them out, in order to guarantee the generality of our model, since our aim is to make it possible to branch out to other unidentified functional sites.

B. Data Extraction

Dash et al. (2001) [17] found in predicting splice sites by a Bayesian network that it achieves better performance when both of the upstream and downstream feature lengths are greater than 15. With a view to simplifying model and

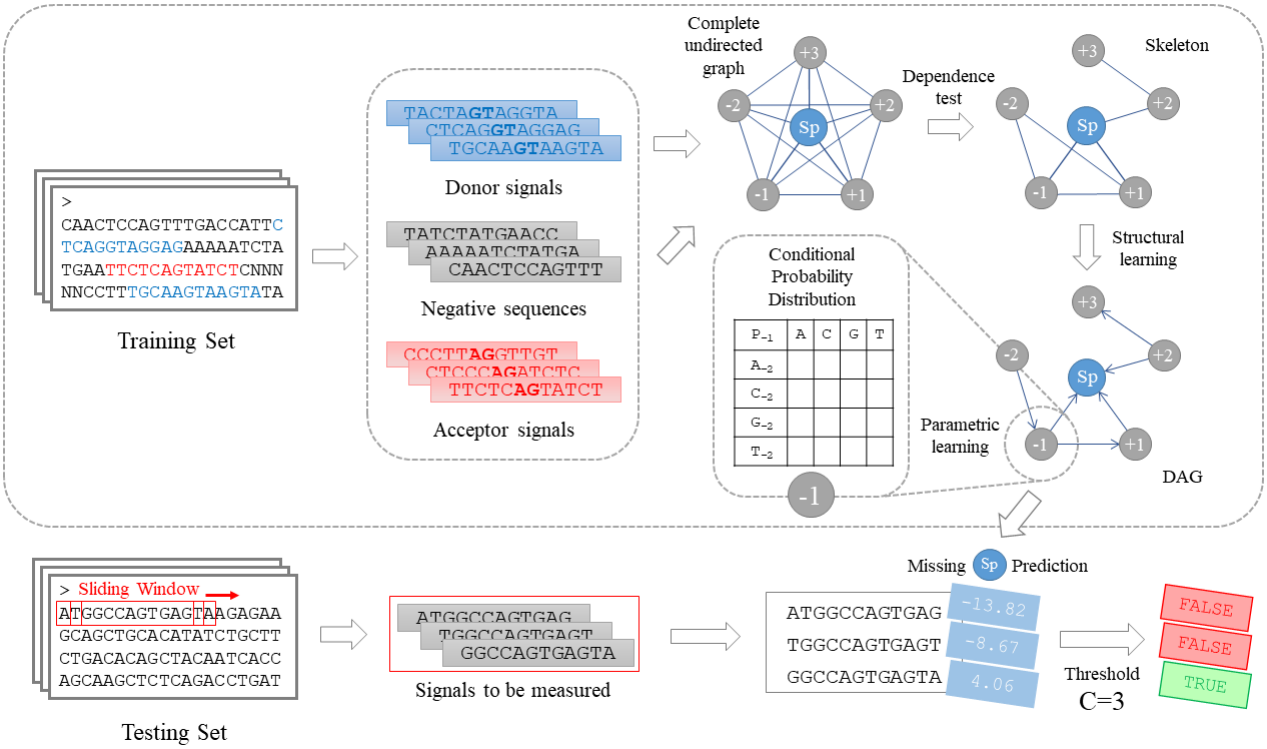

Fig. 2. Overall architecture of the Bayesian network splice predictor. Taking the PC Algorithm [15] as an example, we use the training set to learn a network by the following steps: extracting positive site signals & randomly choosing negative signals, creating a complete undirected graph first, removing edges by dependency tests, orienting edges to form a DAG, and create condition probability distribution (CPD) tables for each node [16]. This network is used to predict unknown signals by these steps: extracting testing sequences of same length by window-sliding, scoring the sequences position-wise by computing the conditional probability of the missing label node 'Sp', calculating the binary log-odds scores, and comparing them with a given threshold to make final judgment.

decreasing the computation, we choose 5 upstream sites and 7 downstream sites of intron / exon junctions to form 12 nt long signal sequences from the primary training set. We abandon sequences containing ambiguous bases, whose correspondence with the splice sites we consider inapparent. The training set provides 2,381 donor signals and 2,381 acceptor signals. As for negative samples, Chen, et al. [14] used pseudo splice sites as false data, extracted by searching for negative sample sequences with $P_{+1}P_{+2}$ = GU / AG whereas, according to the splice site hypothesis above, We randomly selected about 5,000 sites in sections which do not intersect with all donor and acceptor sites, and combined with positive ones to get actual training dataset, the positive-negative ratio of which is about 1:2. Additionally, we export sequences with the same length by window sliding from the primal testing set and build the actual testing set in a positive-negative ratio of 1:20.

For the data imbalance of negative sites versus positive ones, a feasible approach was proposed by Wen, et al. (1999) [18] who split the SVM classifier into 3 sub-classifiers which use a shared set of positive samples but different sets of negative samples, and picked the lowest prediction score on the testing set among all 3 scores generated by sub-classifiers as the final result. Instead we simply used the oversampling strategy, as we made a copy of the positive sites to expand the positive sample size up to 4,762 sequences, which converts the positive-negative ratio into about 1:1.

C. Structural Learning

For Bayesian network structure building, experts are usually needed for confirming the dependency of variables. For certain of tasks, however, this could be extremely difficult considering the massive time cost and complicated correlation between long range sites. Experts as they are, such interdependency is unlikely to be given explicitly in splice site prediction tasks without the help of existing research findings. Thus it is of great importance to learn a dedicated structure for the Bayesian network via the given data, because what we expect is the dependency between every nodes can be directly mined from statistical training data, instead of the guidance of experts. As a matter of fact, the learning methods indeed exist. We considered two structure learning algorithms with different principles.

Scoring & Searching. Two parts constitute the scoring & searching based structure learning approaches. To start with, an appropriate scoring criterion should be selected which assigns a value to each DAG (or DAG pattern) under consideration based on the data [16]. The most common scoring functions include Bayesian Dirichlet Scores (BDeu, K2) and Bayesian Information Criterion (BIC) [19].

The following model searching consists of determining and selecting the DAG patterns with maximum probability conditional on the data (There could be more than one maximizing DAG pattern in general) [16]. For complex DAGs, exhaustive

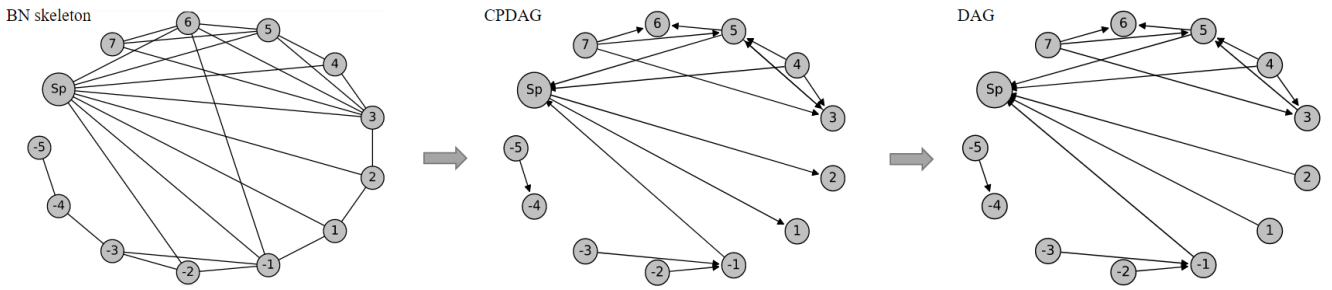

Fig. 3. How a Bayesian network structure is learned by constraint-based statistical analysis. After we gain the skeleton - also called a dependency graph - by chi-square tests (network on the left), we orient part of some edges by d-separation judgment to form a complete partial DAG. Now we have a network with mostly oriented edges and several undirected edges (labeled with bi-directional arrows in the graph, e.g. $3 \leftrightarrow 5$ in the middle) which we have to cope with. We transform our CPDAG into a DAG (network on the right) by orienting these edges and doing some corrections. The networks are generated during our experiment, with type I error $\alpha = 10^{-5}$.

searching is not recommended since it costs a large amount of time to find the global optimum. Hill-climb searching [20] heuristically searches for a local optimum by adding and removing a single edge, which performs at a way higher speed but is too greedy to find the global optimum point. Our experiment shows that DAG structure learned by hill climbing is way too complicated (only a few edges are deleted from the complete graph) which consumes unacceptable time for parametric learning and prediction, so this method is also excluded.

Constraint Statistical Analysis. We choose this approach to learn a Bayesian network. Fig. 3 shows how it is done by operating an initial *complete undirected graph* of base position variables. Other than that, independencies of samples are needed for this structure learning approach. Here we use a chi-squared hypothesis test [21] to find out whether a node u is independent from another node v, given a set of variables. Next we just remove the edges connecting independent nodes of the complete undirected graph to build a *skeleton.*

Because of the causality between two related variables, the edges require orientation and decyclization to change a skeleton into a *DAG.* Firstly we confirm part of some edges' directions by d-separation judgment, turning the skeleton into a *complete partial DAG* (CPDAG) [22]. This is called a PC algorithm for DAG pattern finding [15][16]. Then the CPDAG is expanded to a DAG if it is DAG-faithful, which means no more v-structures will be introduced during the expansion. See [16] for more details of PC algorithm.

Our splice site predicting Bayesian network is built by a slightly different way. (1) Aside from the positions of signal sequences, we also add a label node 'Sp' which indicates the state of each signal by Trues and Falses. (2) Since 'Sp' is dependent on the base distribution of other position nodes, it is reasonable to set the out-degree of 'Sp' to zero, thereby we reverse the arrows pointing to it manually (Fig. 3).

D. Parametric Learning

As a DAG structure is not able to assist directly in predicting hidden splice sites in the genomic DNA sequence, parametric learning of the Bayesian network is acquired to obtain the conditional probability distributions for each state of every node. Two approaches are proposed for parametric learning.

Heckerman, et al. (1998) [23] proposed a maximum likelihood algorithm for network learning: let θ be the CPD of V_A given the parents $V_{P_i}(i = 1, 2, \cdots)$, the learning process searches for a perfect θ s.t. the summation of $P(V_A|V_P, \theta)$ for all parents reaches the top. To this end, the algorithm computes the state counts $\text{freq}(V_A = N)$ and divides each cell by the (conditional) sample size $\text{freq}(V_{P_i} = M_i)$ [19]. θ is filled with conditional probabilities given by:

$$P(V_A = N \mid V_{P_1} = M_1, V_{P_2} = M_2, \cdots) = \frac{\text{freq}(V_A = N, V_{P_1} = M_1, V_{P_2} = M_2, \cdots)}{\text{freq}(V_{P_1} = M_1, V_{P_2} = M_2, \cdots)} \quad (1)$$

$$P_0(V_A = N) = \frac{\text{freq}(V_A = N)}{\sum_N \text{freq}(V_A = N)} \quad (2)$$

where $P_0(V_A = N)$ is the initial probability for nodes with in-degree = 0.

Since the maximum likelihood algorithm may cause overfitting with lack of data, Bayesian Parameter estimating algorithm is introduced. The premise of Bayesian estimation is the prior distribution of each site to be known. These distributions act as pseudo-counts in the Laplace smoothing method [24] and are added to the actual state counts [19].

E. Prediction

The last part of our work is that we use the completed network to perform statistical inference on the testing data. Above all, the extracted testing data (by a sliding window) is preprocessed by dropping the columns which are conditionally independent of all the other positions according to the network structure (sometimes separated vertices appear in the network which is considered weakly-related to the other variables), that is, useless for splice site prediction. We calculate posterior probabilities $P(V_{Sp} = \text{True})$ using the condition probabilities provided by θs that is:

$$\begin{aligned} P(V_{Sp} = \text{True}) &= \sum_{M_1, M_2, \cdots} P(V_{Sp} = \text{True}, V_1 = M_1, V_2 = M_2, \cdots) \\ &= \sum_{M_1, M_2, \cdots} \prod_{i,j} P(V_i = M_i \mid V_j = M_j), \\ &\quad \text{if } V_j \to V_i \text{ exists.} \end{aligned} \quad (3)$$

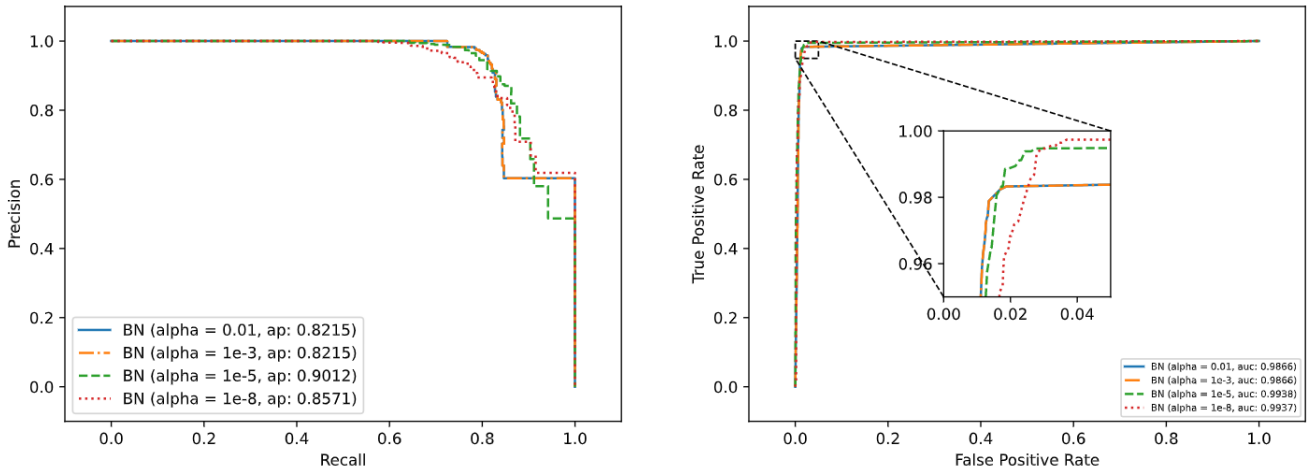

Fig. 4. Performance of BN models with different type I error settings on the KR set. **Left**, P-R curves for comparison. Research shows that BN with type I error $\alpha = 10^{-5}$ outperforms the other models, with the average precision surpassing the runner-up 0.044. This implies that $\alpha = 10^{-5}$ does a great tradeoff between the correspondence and independency of position vertices. However, BNs with $\alpha = 0.01$ and $\alpha = 10^{-3}$ learn noisy dependencies and misclassify some positive samples with very low scores, so false positive samples rise sharply only when Recall is about 0.8. **Right**, ROC curves for comparison. BN with $\alpha = 10^{-8}$ shows a very close performance to the best model compared by AUC values, which means that it has also learned the essential correlations of long range positions.

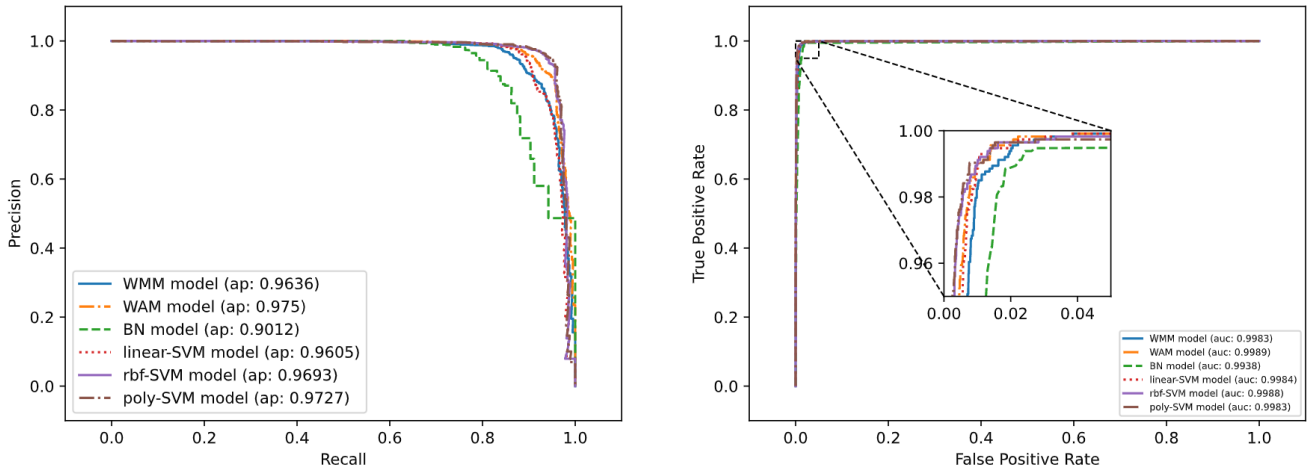

Fig. 5. Overall comparison of splice predictors on a balanced KR set. **Left**, P-R curves which show an abnormal comparison result of Bayesian model performance. Even the BN with the best $\alpha = 10^{-5}$ performs the worst among all models, which is unacceptable considering its huge amount of predicting time cost. **Right**, ROC curves, where the AUC of BN show a -0.004 less than the simplest predictor. This may be explained by the simplicity of splice site patterns: models need to pay more attention on the short range correlations instead of long ones in predicting splice sites. Models considering short correlations only perform better (with WAM reaching the top, unexpectedly), which also proves our explanation.

In this way, we score the extracted data with the log-odds function $S(X)$:

$$S(X) = \ln \frac{P(V_{Sp} = \text{True})}{P(V_{Sp} = \text{False})} \tag{4}$$

Since the posterior probabilities may contain zeros created by the CPD product, we set $P = 10^{-6}$ to avoid division-by-zero error which, in the meantime, guarantees a higher penalty for a sequence including a $P(V_{Sp} = \text{True}) = 0$ site, and vice versa.

Transformation from scores to predicting results needs the comparison. We can filter the true positive sites we need from batches of scores by taking different thresholds. It is necessary to exercise caution in selecting the threshold C. A large C will exclude potential positive sites, while a small C misclassifies negative sites as positive. Hence an appropriate threshold is a tradeoff based on the specificity and sensitivity of a model. For the threshold optima selection, see IV for details.

III. Experiments

A. Data

We conduct our experiment on the eukaryotic gene sequence dataset Kulp & Reese [25] and Burset & Guigo [26]. Human genome sequence dataset Kulp & Reese (KR set) is used as training set which contains 462 sequence text files, each

records the name, length, CDS terminal points and the segment. 2,381 donor sites and 2,381 acceptor sites are extracted from the KR set, which are doubled to 4,762 sequences for balancing. Vertebrate genome dataset Burset & Guigo (BG set) is used as testing set which contains 570 sequence text files with a similar format, except for a lack of the sequence length.

The KR and BG set is open access and you can get the entire dataset at https://www.fruitfly.org/sequence/human-datasets.html and http://www1.imim.es/databases/genomics96/.

B. Metrics

Our model accuracy measures are given by (5) – (11):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{7}$$

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

where TP, FP, TN, FN are metrics of the confusion matrix [27]. Precision-Recall curves and ROC curves [28][29] are plotted to make the performance of our model more intuitive. We also calculate areas under the curves by:

$$\text{AP} = \int_0^1 P(R)\mathrm{d}R = \sum_{i=1}^{n} P_i \Delta R_i \tag{10}$$

$$\text{AUC} = \int_0^1 T(F)\mathrm{d}F = \sum_{i=1}^{n} T_i \Delta F_i \tag{11}$$

where AP summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight [30]. AUC is equal to the probability that the model will rank a randomly chosen positive sample higher than a randomly chosen negative one (assuming that "positive" ranks higher than "negative"), where F denotes false positive rate and T denotes true positive rate [29].

C. Environmental Settings

Throughout our experiments, the PC algorithm and maximum likelihood method are used respectively for structural and parametric learning. Chi-squared test is chosen for independence hypothesis testing. We use the parallel version of stable PC with multiple threads in order to promote the efficiency of Bayesian network learning. The maximum conditional variables is set to 5 to prevent a runaway growth of CPD size.

D. Implementation

We encapsulate the Bayesian network model in the class `BayesNet` which is derived from the base class `Base` in `./Model/basemodel.py`. Sequences are extracted by `./Utils/extract.py` and saved temporarily in an `Sequence` object. All of the statistical graphs involved in this paper are drawn by the scripts in `./Utils` using Matplotlib [31], NetworkX [32] & Weblogo, and saved in `./Pics`. We evaluate the model using Scikit-Learn for confusion matrices, precision-recall pairs and FPR-TPR pairs [33]. These tools saved considerable time for model training and prediction. Models can be easily saved or loaded by methods `save_model()` and `load_model()`. All the components have their corresponding interface methods provided in the aforementioned classes. For the code implementation details of Bayesian network, see `./Model/bn.py`.

Training & Predicting process is operated by Ubuntu 18.04.5 LTS on 16 CPU cores with 2 threads each. The source code is available on GitHub and can be obtained from https://github.com/Newiz430/SplicePredictor.

IV. Results

A. Type I Errors

Type I error is commonly used in chi-squared hypothesis tests which denotes the rejection of a true null hypothesis [34], i.e. two variables are dependent of each other. The judgment of conditional independency between two sites is influenced by the value of type I error, so does the structure of DAG. Distinct structures of DAGs affect the quality and efficiency of prediction.

We choose 4 values of type I error $\alpha = 0.01$, 10^{-3}, 10^{-5}, 10^{-8} for chi-squared test from the PC algorithm and evaluated the model behaviors. Results are shown in Fig. 4 and Table I. Note that BNs with $\alpha = 0.01$ by default and $\alpha = 10^{-3}$ are nothing like as good as ones with lower αs, including the tedious prediction and poor precision. It seems that a higher acceptance of the true null hypothesis would lead to futile learning of numerous useless or noisy edges and massive computation of posterior probabilities, which doubles the decline of model performance. Instead, BNs with $\alpha = 10^{-5}$ by default and $\alpha = 10^{-8}$ are ideal predictors which achieve higher accuracy with a relatively small time cost.

TABLE I
TIME COST OF BNS WITH DIFFERENT TYPE I ERROR SETTINGS

Type I Error	Training Time ($\times 10^3 s$)	Predicting Time (h)
0.01	7.37	5.22
1e-3	5.02	5.27
1e-5	3.87	1.23
1e-8	**0.893**	**1.13**

B. Donor Site Prediction

We compared the performance of various splice site predicting models such as WMM, WAM, SVM and the Bayesian network. The results shown in Fig. 5 acknowledge that Bayesian network do not perform as good as expected, giving the proof that, under the same circumstances, it is more difficult for Bayesian models to learn the pattern of splice sites because of its sophisticated training steps, without any human

TABLE II
PERFORMANCE OF BN AGAINST OTHER DONOR SITE PREDICTORS[1]

Method	Precision	Recall (TPR)	FPR	F1-score	Run time(s)
WMM (threshold = 2.26)	0.8503	**0.9697**	0.0085	0.9061	**1.6614**
WAM (threshold = 3.09)	0.9006	0.9543	0.0053	0.9267	1.7940
Linear - SVM[2](threshold = 1.63)	0.8801	0.9678	0.0066	0.9219	2.3974
RBF - SVM[2](threshold = 2.94)	0.9265	0.9278	0.0037	0.9272	3.4894
4D poly - SVM[2](threshold = 3.00)	**0.9290**	0.9384	**0.0036**	**0.9337**	3.0368
BN (threshold = 2.32)	0.8067	0.9410	0.0113	0.8688	4.23e+3

[1] The argmax thresholds are assigned to these models to get the best metrics. Run time represents the seconds cost in the predicting step. Same for tables below.
[2] "Linear - SVM", "RBF - SVM", "4D poly - SVM" are support vector machine approaches with different kernels.

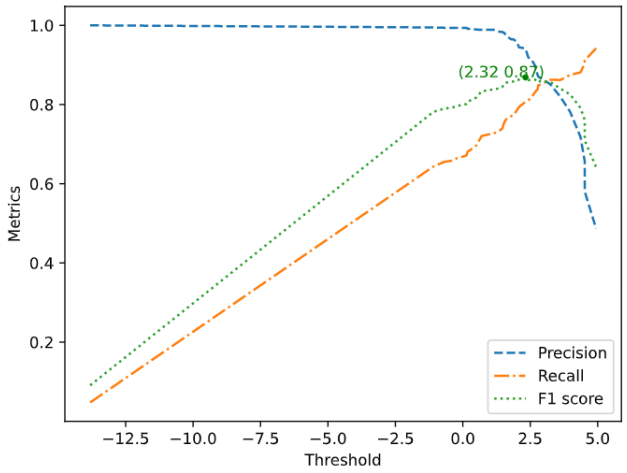

Fig. 6. Searching for the best threshold for donor prediction. F1-score (9) is a balanced metric between precision and recall which expresses best performance of models. We have found the maximum F1 point for BN: (2.32, 0.87).

intervention on the prior knowledge. It verifies that donor site prediction extremely relies on nothing but two conservative points shown in Fig. 1, whose strong impact on positive splice sites, however, is not paid attention by the network model. Chances are that the hyperparameters we set for the BN are not perfect, additionally, since we abandoned parameter grid searching thanks to its training time consumption.

We present the stats of accuracy for all evaluated models in Table II, in which the best threshold for BN is searched by F1-scoring (Fig. 6).

C. Acceptor Site Prediction

We did the same experiment on acceptor sites. Consequences are displayed in Fig. 7 – 8 and Table III.

V. DISCUSSION

Overall, we formulate and re-implement an application of Bayesian network model by training it on the Kulp & Reese dataset. We find the best type I error to learn an appropriate DAG structure and compare its performance against other feasible approaches. Although no ideal results are given in this paper, we do not regard this as a failure. The availability for Bayesian networks on splice site prediction tasks is still proven in our work.

As a matter of fact, there are still some blemishes in our methods which need to be taken serious consideration. For the preprocessing of the training set, we only sampled a fraction of data for model learning thus the network may not attain its best performance with the given dataset. We ignored the odds of base indels in signal sequences. We omitted unambiguous bases at the beginning of our work which is likely to be part of the splice site patterns. What's more, we only tried one single feature selection tactic limited by the deadline (our model is actually designed for feature sequences of different lengths as input).

From the results above, a Bayesian network performs prediction at a pretty low speed (a reduction of one magnitude) compared to other models, for the massive computing of conditional probabilities. With the rapid development of high-throughput sequencing, promotion of predicting speed to match the sequencing is a matter of great concern nowadays, hence this is a serious problem for Bayesian methods.

The network learning algorithms need an improvement. Even the most efficient constraint-based structural learning algorithms have a tremendous time complexity and are only suitable for sparse networks, that is, they are still unacceptable for predicting large functional sites with complicated interdependency, let alone the searching methods.

Low robustness on unbalanced data. WAM and SVM shows eligible adaptability against training data with a partiality for negative samples but a Bayesian network does not. We found in our experiment that different ratios of positive and negative samples have a great impact on the performance of network because of the volatility of conditional probabilities of those variables. A characteristic of actual sequence data is the imbalance between positive and negative sites, the stability of pattern predictors against such data imbalance are accordingly in great demand. While the oversampling strategy works, as shown in our experiment, it lowers the generality of Bayesian networks and amplifies the noises mingled in the positive data.

Despite the downside, Bayesian networks still have potential for predicting functional sites efficiently and accurately. In the future, we will dedicate ourselves to improving Bayesian networks in general. Several methods for structural and parametric learning will be applied and further evaluated for

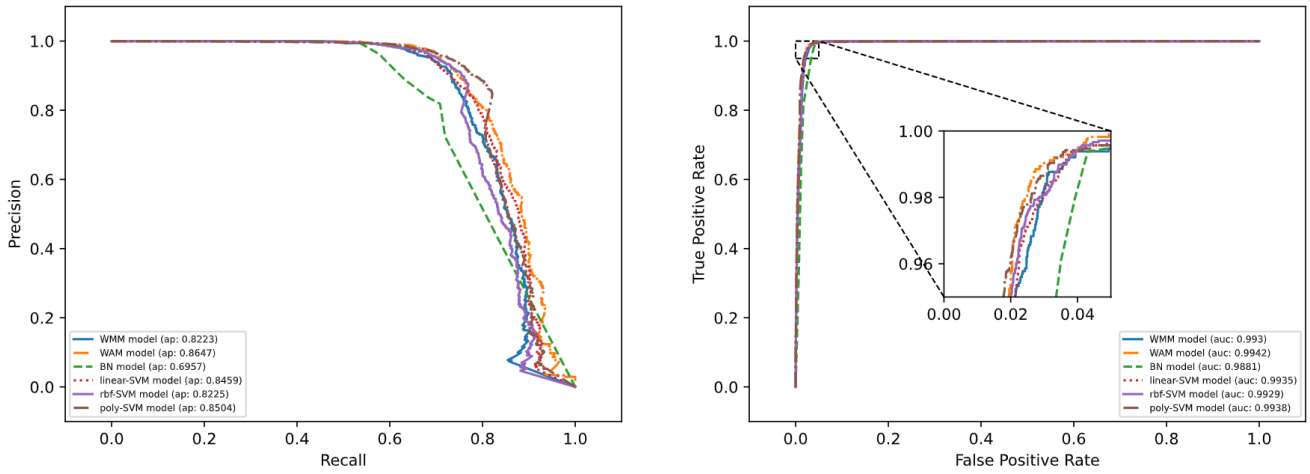

Fig. 7. Precision-Recall curves (**left**) and ROC curves (**right**) for acceptor signal. Bayesian network used for acceptor seems to have a lower predicting ability relative to the one for donor.

TABLE III
PERFORMANCE OF BN AGAINST OTHER ACCEPTOR SITE PREDICTORS

Method	Precision	Recall (TPR)	FPR	F1-score	Run time(s)
WMM (threshold = 2.69)	0.7385	**0.9038**	0.0160	0.8128	**1.3336**
WAM (threshold = 2.54)	0.7554	**0.9038**	0.0146	0.8229	1.4286
Linear - SVM (threshold = 2.46)	0.7603	0.8653	0.0137	0.8094	2.5636
RBF - SVM (threshold = 2.40)	0.7677	0.8999	0.0136	0.8285	4.6039
4D poly - SVM (threshold = 2.55)	**0.7774**	0.8975	**0.0129**	**0.8331**	3.3924
BN (threshold = 3.38)	0.7083	0.8187	0.0169	0.7595	1.02e+3

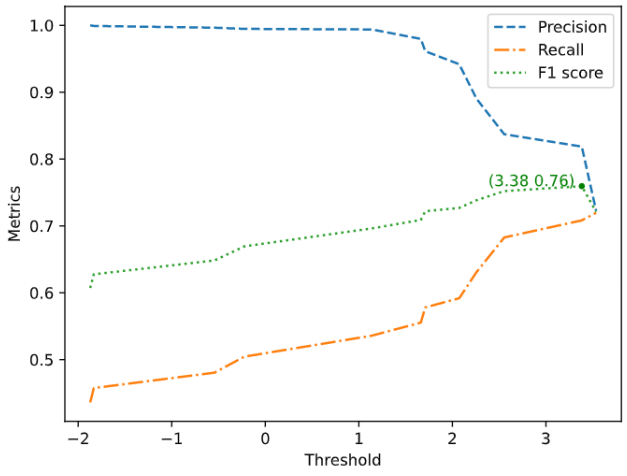

Fig. 8. Searching for the best threshold for acceptor prediction. We have found the maximum F1 point for BN: (3.38, 0.76).

this task. We will also try to add more interference in the training process to assist the network in finding the right interdependency between variables.

ACKNOWLEDGMENT

This work was supported by Prof. Zhou from College of Life Science & Technology, Huazhong University of Science and Technology, and Wuhan National Laboratory for Optoelectronics for providing computing resources. Also acknowledge our classmates for helpful suggestions & corrections.

REFERENCES

[1] Thomas D Schneider and R Michael Stephens. Sequence logos: a new way to display consensus sequences. *Nucleic acids research*, 18(20):6097–6100, 1990.

[2] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[3] Thomas D Schneider, Gary D Stormo, Larry Gold, and Andrzej Ehrenfeucht. Information content of binding sites on nucleotide sequences. *Journal of molecular biology*, 188(3):415–431, 1986.

[4] Chris Burge and Samuel Karlin. Prediction of complete gene structures in human genomic dna. *Journal of molecular biology*, 268(1):78–94, 1997.

[5] Suzanne Clancy et al. Rna splicing: introns, exons and spliceosome. *Nature Education*, 1(1):31, 2008.

[6] Douglas L Black. Mechanisms of alternative pre-messenger rna splicing. *Annual review of biochemistry*, 72(1):291–336, 2003.

[7] Harvey Lodish, Arnold Berk, Chris A Kaiser, Chris Kaiser, Monty Krieger, Matthew P Scott, Anthony Bretscher, Hidde Ploegh, Paul Matsudaira, et al. *Molecular cell biology*. Macmillan, 2008.

[8] Meena Kishore Sakharkar, Vincent TK Chow, and Pandjassarame Kangueane. Distributions of exons and introns in the human genome. *In silico biology*, 4(4):387–393, 2004.

[9] Rodger Staden. Computer methods to locate signals in nucleic acid sequences. 1984.

[10] MO Zhang and TG Marr. A weight array method for splicing signal analysis. *Bioinformatics*, 9(5):499–509, 1993.

[11] Mojie Duan, Min Huang, Chuang Ma, Lun Li, and Yanhong Zhou. Position-specific residue preference features around the ends of helices and strands and a novel strategy for the prediction of secondary structures. *Protein science*, 17(9):1505–1512, 2008.

[12] Tom Ryen, Trygve Eftes, Thomas Kjosmoen, Peter Ruoff, et al. Splice site prediction using artificial neural networks. In *International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*, pages 102–113. Springer, 2008.

[13] Ankur Ankan and Abinash Panda. pgmpy: Probabilistic graphical models using python. In *Proceedings of the 14th Python in Science Conference (SCIPY 2015)*. Citeseer, 2015.

[14] Te-Ming Chen, Chung-Chin Lu, and Wen-Hsiung Li. Prediction of splice sites with dependency graphs and their expanded bayesian networks. *Bioinformatics*, 21(4):471–482, 2005.

[15] Peter Spirtes and Clark Glymour. An algorithm for fast recovery of sparse causal graphs. *Social science computer review*, 9(1):62–72, 1991.

[16] Richard E Neapolitan et al. *Learning bayesian networks*, volume 38. Pearson Prentice Hall Upper Saddle River, NJ, 2004.

[17] Denver Dash and Vanathi Gopalakrishnan. Modeling dna splice regions by learning bayesian networks. 2001.

[18] Wen Fang, Lu Xin, Sun Zhirong, and Li Yanda. Splice sites prediction using support vector machine. *Shengwu Wuli Xuebao*, 15(4):733–739, 1999.

[19] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[20] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

[21] Karl Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900.

[22] Markus Kalisch and Peter Bühlman. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(3), 2007.

[23] David Heckerman. A tutorial on learning with bayesian networks. *Innovations in Bayesian networks*, pages 33–82, 2008.

[24] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[25] Martin G Reese, Frank H Eeckman, David Kulp, and David Haussler. Improved splice site detection in genie. *Journal of computational biology*, 4(3):311–323, 1997.

[26] Moises Burset and Roderic Guigo. Evaluation of gene structure prediction programs. *genomics*, 34(3):353–367, 1996.

[27] Stephen V Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89, 1997.

[28] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

[29] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[30] Mu Zhu. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2(30):6, 2004.

[31] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[32] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[33] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[34] Frederik Michel Dekking, Cornelis Kraaikamp, Hendrik Paul Lopuhaä, and Ludolf Erwin Meester. *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Science & Business Media, 2005.