# Boosted Categorical Restricted Boltzmann Machine for Computational Prediction of Splice Junctions

**Taehoon Lee** TAEHOONLEE@SNU.AC.KR
**Sungroh Yoon** SRYOON@SNU.AC.KR
Department of Electrical and Computer Engineering, Seoul National University, Seoul 151-744, Korea

## Abstract

Splicing refers to the elimination of non-coding regions in transcribed pre-messenger ribonucleic acid (RNA). Discovering splice sites is an important machine learning task that helps us not only to identify the basic units of genetic heredity but also to understand how different proteins are produced. Existing methods for splicing prediction have produced promising results, but often show limited robustness and accuracy. In this paper, we propose a deep belief network-based methodology for computational splice junction prediction. Our proposal includes a novel method for training restricted Boltzmann machines for class-imbalanced prediction. The proposed method addresses the limitations of conventional contrastive divergence and provides regularization for datasets that have categorical features. We tested our approach using public human genome datasets and obtained significantly improved accuracy and reduced runtime compared to state-of-the-art alternatives. The proposed approach was less sensitive to the length of input sequences and more robust for handling false splicing signals. Furthermore, we could discover non-canonical splicing patterns that were otherwise difficult to recognize using conventional methods. Given the efficiency and robustness of our methodology, we anticipate that it can be extended to the discovery of primary structural patterns of other subtle genomic elements.

## 1. Introduction

In living organisms, biological information flows from deoxyribonucleic acid (DNA) to ribonucleic acid (RNA) to protein. DNA is a sequence of four types of nucleotides: adenine (A), guanine (G), cytosine (C), and thymine (T). A gene is a segment of DNA that constitutes the basic unit of heredity. As shown in Fig. 1, genetic information is delivered from DNA to protein through a procedure called *gene expression* (Lockhart & Winzeler, 2000). There are three major steps in gene expression: transcription, splicing, and translation.

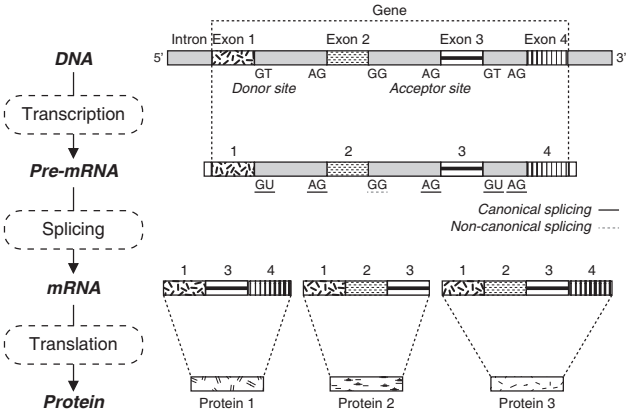


*Figure 1.* Gene expression process. Dimers GU(GT) and AG represent canonical donor and acceptor splice sites, respectively. Dimer GG shows an example of non-canonical (donor) sites.

Eukaryotic genes have an internal structure that includes two types of subunits: exons (protein-coding regions) and introns (non-coding regions). Introns intervene between exons, and the boundary between an exon and an intron is referred to as the *splice junction (site)*. The majority of splice sites contain consensus strings called *canonical splicing* patterns. The most frequent patterns are dimer GT (called *donor*) and dimer AG (called *acceptor*) at intron/exon boundaries (Burset et al., 2000).

During transcription, DNA is copied into precursor messenger RNA (pre-mRNA), and then the introns in the transcribed pre-mRNA are removed by splicing (Keren et al., 2010). For a single gene, various combinations of al-

*Proceedings of the $32^{nd}$ International Conference on Machine Learning*, Lille, France, 2015. JMLR: W&CP volume 37. 

ternative exons selectively remain in the resulting mature mRNA, allowing the construction of multiple proteins from the gene. Consequently, this alternative splicing gives rise to the enormous diversity of proteins (Nilsen & Graveley, 2010), and the identification of splice sites is a crucial step to fully understand gene expression.

Existing junction prediction methods belong two categories: sequence alignment-based and machine learning-based. Alignment-based strategies (Trapnell et al., 2010; Wang et al., 2010; Grant et al., 2011) reconstruct exons by mapping millions of short RNA sequences to the whole genome sequence and then estimate where splicing occurs using adjacent exons. Despite the need for a reference genome, alignment methods can identify novel splice sites in addition to the original sequence locations in the reference. Alternatively, machine-learning techniques construct a descriptive model of splicing by training with known junction signals. The learning models used include artificial neural network (ANN, Stormo et al., 1982; Noordewier et al., 1990; Brunak et al., 1991), support vector machine (SVM, Degroeve et al., 2005; Huang et al., 2006; Sonnenburg et al., 2007), and hidden Markov model (HMM, Reese et al., 1997; Pertea et al., 2001; Baten et al., 2006).

Learning-based approaches have produced promising results since the early 1980's, but they often suffer from practical limitations, such as excessive false positives and limited scalability. With the advent of next-generation sequencing technology, alignment-based methodologies using RNA-seq technology (Chu & Corey, 2012) have gained popularity over the last decade as an alternative to learning-based approaches. However, existing alignment-based methods, such as TopHat (Trapnell et al., 2009) and SpliceMap (Au et al., 2010), consider only *canonical* splicing signals (`GT` or `AG`) in their splitting and merging procedures and often miss important splicing signals. To improve the accuracy of prediction, not only canonical but also *non-canonical* patterns are important. Given that machine learning-based methodologies can learn and predict such non-canonical junction signals, we believe that learning-based and alignment-based approaches should be used in a complementary way to boost performance.

In this paper, we propose a new machine learning method for predicting splicing patterns using a deep belief network (DBN, Hinton & Salakhutdinov, 2006). A DBN learns high-level features from unlabeled data (Hinton, 2002) and then fine-tunes its weights to improve the discriminative performance. Leveraged by this two-step learning, a DBN has a strong generalization ability and it has resulted in breakthroughs in various applications (Erhan et al., 2010).

Our proposal includes a novel procedure for training restricted Boltzmann machines (RBMs) comprising a DBN. This procedure improves contrastive divergence (CD, Hinton, 2002) when training an RBM for DNA sequences, and more generally, for binary representations of categorical information. The idea of our approach resembles that of boosting in ensemble learning. We thus name the proposed training scheme *boosted CD with categorical gradient*.

In our experiments, our approach achieved F1-scores nearly 20% higher than the alternatives and was particularly effective for training in class imbalanced problems. We present our work in the context of genomics, but it is applicable to learning from other types of data that contain categorical features (*i.e.*, text mining).

## 2. Background

Deep networks have been adopted successfully for various tasks, such as image recognition (Krizhevsky et al., 2012), audio classification (Lee et al., 2009), and face recognition (Reed et al., 2014). Applications in bioinformatics include protein structure prediction (Lena et al., 2012), drug-target interaction prediction (Wang & Zeng, 2013), and tissue-regulated splicing prediction at RNA level using RNA-seq data (Leung et al., 2014). To the best of our knowledge, the present study is the first attempt to predict splicing signals using DBN at DNA level.

### 2.1. Restricted Boltzmann Machine (RBM)

A restricted Boltzmann machine contains stochastic binary-valued hidden units $\mathbf{h} = \{h_1, ..., h_{n_h}\}$ and visible units $\mathbf{v} = \{v_1, ..., v_{n_v}\}$, where $n_h$ and $n_v$ are the numbers of hidden and visible units, respectively. As shown in Fig. 2(a), the two layers $\mathbf{v}$ and $\mathbf{h}$ are tied with symmetrically weighted connections denoted by $W$, forming a bipartite graph. $W$ is represented by an $n_v \times n_h$ matrix, where $w_{ij}$ is the weight for the connection between $v_i$ and $h_j$.

Each hidden node has an activation probability given by $P(h_j = 1|\mathbf{v}) = \text{sigm}(c_j + \sum_{i=1}^{n_v} v_i w_{ij})$, where the activation function $\text{sigm}(x) = 1/(1 + \exp(-x))$ is shown in Fig. 2(b) and $c_j$ is the bias weight for the hidden unit $h_j$. Similarly, the activation probability of visible unit $v_i$ is given by $P(v_i = 1|\mathbf{h}) = \text{sigm}(b_i + \sum_{j=1}^{n_h} w_{ij} h_j)$, where $b_i$ is the bias unit for the visible unit $v_i$. The energy of the network can be defined as

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{n_v} b_i v_i - \sum_{j=1}^{n_h} c_j h_j - \sum_{i=1}^{n_v} \sum_{j=1}^{n_h} v_i w_{ij} h_j. \quad (1)$$

The joint probability for two vectors $\mathbf{v}$ and $\mathbf{h}$ is given by $P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})}$ where the normalization constant $Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$. The probability of observing a set of visible units $\mathbf{v}$ is given by $P(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$. We can formulate an optimization problem for training RBM to minimize the negative log-likelihood (NLL) of data.

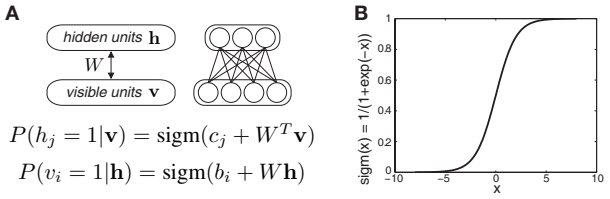


*Figure 2.* (a) RBM representation; (b) sigmoid activation.

### 2.2. Contrastive Divergence (CD) for Training RBM

When the gradient descent is utilized to minimize the negative log-likelihood, the CD procedure is normally used to approximate the gradient due to the intractable $Z$ inside $P(\mathbf{v})$. The derivative of the NLL with respect to $w_{ij}$ is

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}\mathbf{E}\left[-\log\left(\frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v}_n,\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}\right)\right] = \mathbf{E}_{data}[v_i h_j] - \mathbf{E}_{model}[v_i h_j].$$

While $\mathbf{E}_{data}[v_i h_j]$ is a simple expectation over observed vectors, $\mathbf{E}_{model}[v_i h_j]$ requires the probabilities of all possible visible states.

To approximate the expectation, CD exploits Gibbs sampling (Lawrence et al., 1993), which enables the estimation of the distribution using samples close to observations. The derivative from $k$-step Gibbs sampling (see Fig. 3) is

$$\frac{\partial L}{\partial W} \approx \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{v}_n^{(0)}\mathbf{h}_n^{(0)^T} - \mathbf{v}_n^{(k)}\mathbf{h}_n^{(k)^T}\right). \quad (2)$$

In the same manner, the derivatives of $\mathbf{b}$ and $\mathbf{c}$ can be approximated as follows (Bengio, 2009): $\frac{\partial L}{\partial \mathbf{b}} \approx \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{v}_n^{(0)} - \mathbf{v}_n^{(k)}\right)$, $\frac{\partial L}{\partial \mathbf{c}} \approx \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{h}_n^{(0)} - \mathbf{h}_n^{(k)}\right)$.

## 3. Proposed Methodology

Fig. 3 shows an overview of the proposed methodology.

As training data, we use a set of $N$ DNA sequences, each of which has $m$ nucleotides (nt). Some of these contain either an acceptor or a donor site, and the others contain no splice site. Each training sequence has a label: acceptor, donor, or non-site. Each sequence is converted into a binary vector by orthogonal encoding (see Section 3.1).

After preprocessing, our approach proceeds in two main steps: (1) unsupervised pre-training of component RBM using the proposed boosted contrastive divergence with categorical gradient; and (2) organizing DBN by RBM stacking and supervised fine-tuning of the DBN. The label of each training sequence is not used for the first pre-training step but only for the second fine-tuning step.

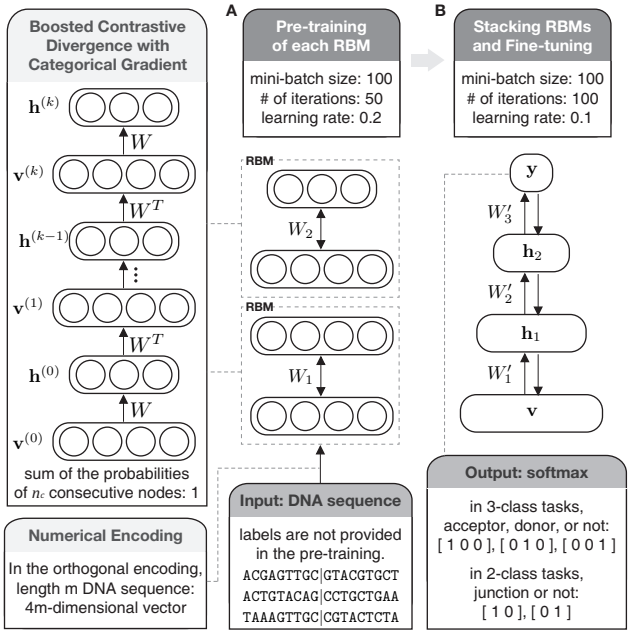


*Figure 3.* Proposed methodology: (a) pre-training; (b) fine tuning.

In the inference step (not shown in Fig. 3), an unlabeled DNA sequence is fed to the trained DBN, and it predicts whether the sequence contains a splice junction or not.

### 3.1. Input Representation: Encoding DNA Sequences

DNA can be considered as a sequence of categorical values. Machine learning based methods typically employ an encoding scheme to convert a DNA sequence into its numerical representation for downstream processing.

For biological sequences, the orthogonal encoding (*e.g.,* 1-hot encoding) is widely used (Baldi & Brunak, 2001). We employ $n_c$-bit 1-hot encoding. For $n_c = 4$, `A`, `C`, `G`, and `T` are encoded by $1000, 0100, 0010$, and $0001$, respectively (we use the notations $1000$ and $[1, 0, 0, 0]$ interchangeably).

Orthogonal encoding may cause the trained model to have limited generalization ability because of the sparsity of encoding. For example, sequence `AGTT` is encoded by 16-dimensional binary vector $1000001000010001$, of which 75% of the elements are zero. To alleviate this issue, we devise a new regularization technique that incorporates prior knowledge on the sparsity, as will be detailed shortly.

### 3.2. Training RBM using Boosted Contrastive Divergence with Categorical Gradients

For a DNA sequence of $m$ nucleotides, we convert it into a binary vector $\mathbf{v}$ of $n_v = (n_c \times m)$ elements by the $n_c$-bit 1-hot encoding as described previously. This $\mathbf{v}$ becomes the visible units of the RBM described in Section 2.1. In

**Algorithm 1** Boosted CD with Categorical Gradient

**Input:** $N$ encoded DNA sequences $\mathbf{v}_1, \ldots, \mathbf{v}_N$
**Output:** weights $W, \mathbf{b}, \mathbf{c}$
Initialize $W \sim \mathcal{N}(0, 0.1), \mathbf{b} = \mathbf{0}, \mathbf{c} = \mathbf{0}$
**for each** epoch **do**
  **for each** minibatch with size $N$ **do**
    Compute $E_{min} = -\sum_i b_i - \sum_j c_j - \sum_i \sum_j w_{ij}$
    **for** $n = 1$ to $N$ **do**
      Compute $\mathbf{h}_n^{(0)} = P(\mathbf{h} = 1|\mathbf{v}_n^{(0)})$
      Sample $\mathbf{v}_n^{(1)}$ from $P(\mathbf{v} = 1|\mathbf{h}_n^{(0)})$
      Compute $\mathbf{h}_n^{(1)} = P(\mathbf{h} = 1|\mathbf{v}_n^{(1)})$
      Compute $\alpha_n = E(\mathbf{v}_n^{(1)}, \mathbf{h}_n^{(1)}) - E_{min}$
    **end for**
    Normalize $\alpha_n = N \cdot \alpha_n / \sum_n \alpha_n$ for each $n$
    Update $W$, $\mathbf{b}$, $\mathbf{c}$ using (3), (4), (5) with $\alpha_n$'s
  **end for**
**end for**

Eq. 2, $\mathbf{v}^{(0)}$ and $\mathbf{v}^{(k)}$ can be considered as the original input and its reconstruction vectors, respectively. We require that the sum of the probabilities of $n_c$ consecutive nodes in the reconstructed $\mathbf{v}^{(k)}$ units be 1. We add a regularization term that penalizes the deviation of the sum of $n_c$ visible units from 1. We called this term the *categorical gradient*.

The NLL of input data now becomes

$$\min_{W,\mathbf{b},\mathbf{c}} \mathbf{E}\big[-\sum_{n=1}^{N} \log P(\mathbf{v}_n)\big] + \frac{\lambda_c}{2} \sum_{i=1}^{m} \Big(\sum_{j=1}^{n_c} v_{n_c(i-1)+j}^{(k)} - 1\Big)^2$$

where $N$ is the number of input sequences, and $\lambda_c$ represent sensitivity to the categorical gradient. The update rules for minimizing the aforementioned NLL are as follows:

$$\frac{\partial L}{\partial W} \approx \text{Eq. (2)} + \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{v}_n^{(k)}) \mathbf{h}_n^{(k-1)} \tag{3}$$

$$\frac{\partial L}{\partial \mathbf{b}} \approx \frac{1}{N} \sum_{n=1}^{N} \Big(\mathbf{v}_n^{(0)} - \mathbf{v}_n^{(k)} + f(\mathbf{v}_n^{(k)})\Big) \tag{4}$$

$$\frac{\partial L}{\partial \mathbf{c}} \approx \frac{1}{N} \sum_{n=1}^{N} \Big(\mathbf{h}_n^{(0)} - \mathbf{h}_n^{(k)}\Big) \tag{5}$$

$$f(\mathbf{v}) = \mathbf{v} \circ (1 - \mathbf{v}) \circ g(\mathbf{v}),\ g(\mathbf{v})_i = \sum_{j=1}^{n_c} v_{n_c[\frac{i-1}{n_c}]+j} - 1$$

where operator $\circ$ denotes the Hadamard product and $g(\mathbf{v})$ represents an element-wise operation that replicates the deviations of the sum of the $n_c$ nodes from 1. For example, $g(\mathbf{v}) = [g_1, g_1, g_1, g_1, g_2, g_2, g_2, g_2]$, where $g_1 = \sum_{i=1}^{4} v_i - 1$ and $g_2 = \sum_{i=5}^{8} v_i - 1$, for $n_v = 8$ and $n_c = 4$.

Using these modified derivatives allows us to extract novel features that would work well for both reconstruction and classification. For training with the proposed categorical gradient, we propose a new approach that improves the CD-1 approach. Conventional CD often provides a reasonable approximation of the model distribution but still suffers from a critical drawback: the computation of the negative phase needs to be more precise than $\mathbf{v}^{(k)}\mathbf{h}^{(k)^T}$ to approximate the model expectation $\mathbf{E}_{model}[\mathbf{v}\mathbf{h}^T]$. There have been approaches to address the issue, such as persistent CD (Tieleman, 2008) and parallel tempering (Cho et al., 2010). They often draw samples from the model distribution more accurately than conventional CD, but their approximations can still be far from the model expectation.

If we assign the same weight to all the data, the performance of Gibbs sampling would degrade in the regions that are hardly observed. To approximate the model expectation precisely, we need to sample these regions. Borrowing the idea of boosting in ensemble learning, we emphasize unstable observations that have high energy during the training procedure, given that high-energy states typically have low likelihood and provide a high reconstruction error. When sampling, we therefore weight each observation by the energy of its reconstruction $E(\mathbf{v}_n^{(k)}, \mathbf{h}_n^{(k)})$. This re-weighting is also linked to importance sampling, in which the density function is scaled in order to move the probability mass to the desired event region (Neal, 2001).

The rationale behind our idea is as follows: At the beginning of training, the joint probability distribution $P(\mathbf{v}, \mathbf{h})$ is highly unstable and the update direction is affected by the first several mini-batches. If important observations are not included therein, the possibility of sampling further from these regions decreases because RBM assigns high energy to these regions. CD training is looped over all mini-batches and can alleviate this issue to some extent. However, when there is a significant class imbalance, as in the junction prediction, we are not likely to extract appropriate hidden representations of those observations.

Algorithm 1 presents the pseudocode of our training algorithm, which is named *boosted CD with categorical gradients* to emphasize the notion of re-weighting. We first compute the minimum energy $E_{min}$ under the current configuration $(W, \mathbf{b}, \mathbf{c})$ and assign energy-proportional weight $\alpha_n$ to individual data $\mathbf{v}_n^{(0)}$. We then normalize these $\alpha$'s so that the coefficients vary from 0 to $N$. Most of the coefficients will be around 1 because most of the energy values deviate from $E_{min}$. Combining $\alpha_n$'s with the update rules is straightforward. For example, Eq. 3 becomes $\frac{1}{N} \sum_{n=1}^{N} \alpha_n \Big(\mathbf{v}_n^{(0)}\mathbf{h}_n^{(0)^T} - \mathbf{v}_n^{(k)}\mathbf{h}_n^{(k)^T} + f(\mathbf{v}_n^{(k)})\mathbf{h}_n^{(k-1)}\Big)$.

### 3.3. Stacking and Fine-Tuning

After pre-training RBMs with the proposed approach, we stack them and place an output layer on them to construct a DBN (see Fig. 3) and then train it in a supervised manner.

For three-class problems, the output softmax layer consists of three nodes and the resulting output vector $\mathbf{y}$ is one of

*Table 1.* GWH genome-wide data (Sonnenburg et al., 2007)

| two-class, 398nt long, contains canonical signals only | | |
|---|---|---|
| Data ID | # of positives | # of negatives |
| GWH-donor | 160,601 (0.21%) | 76,335,126 |
| WH-acceptor | 158,217 (0.29%) | 54,469,623 |

*Table 2.* UCSC genome browser database (Kent et al., 2002)

| three-class, 60nt long, contains non-canonical signals as well | | | |
|---|---|---|---|
| Data ID | # of donors | # of acceptors | # of non-site |
| UCSC-hg19 | 62,819 | 62,819 | 62,819 |
| UCSC-hg38 | 63,454 | 63,454 | 63,454 |

the following: 100, 010, or 001 for acceptor, donor, and non-site, respectively. For two-class problems, the output layer consists of two nodes, and $\mathbf{y}$ is either 10 or 01 for splice-site (donor/acceptor) or non-site, respectively.

In the fine-tuning step, we utilize backpropagation to minimize the squared loss function $J(W, \mathbf{c}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}||f(\mathbf{v}_n) - y_n||_2^2$, where $f(\mathbf{v}_n)$ and $y_n$ are the final output vector of the network for input $\mathbf{v}_n$ and the desired output vector, respectively. Through this optimization, pre-trained $W_1$, $W_2$, and $W_3$ are fine-tuned to the final weights $W_1'$, $W_2'$, and $W_3'$, respectively.

### 3.4. Initialization and Parameter Setting

To speed up the pre-training and fine-tuning procedures, the input datasets were split into mini-batches of $M = 100$ sequences.

In the pre-training step, we initialized the values of $W$ in component RBMs using Gaussian $\mathcal{N}(0, 0.1)$ and set both $\mathbf{b}$ and $\mathbf{c}$ to the zero vector. We set the number of iterations and the learning rate to $T = 50$ and $\alpha = 0.2$, respectively. We used a different number of hidden nodes or layers for different datasets, as will be explained in the next section.

In the fine-tuning stage, the weights ($W_3$, $\mathbf{c}_3$) for the output layer were initialized using the Gaussian $\mathcal{N}(0, 0.1)$ and the zero vector, respectively. The number of iterations and the learning rate used were $T = 100$ and $\alpha = 0.1$.

## 4. Results and Discussion

### 4.1. Experiment Preparation

We tested our approach with the datasets listed in Tables 1 and 2. The genome-wide datasets for humans (GWH, Sonnenburg et al., 2007) consist of two types of datasets: GWH-donor and GWH-acceptor. Each of these two datasets includes sequences from the 24 human chromosomes (22 autosomes and 2 sex chromosomes). All the sequences in both GWH-donor and GWH-acceptor are of length 398nt. The splice signals from these sequences are all canonical, and all the sequences have dimer `GT` or `AG` in the middle. That is, each sequence from GWH-donor has dimer `GT` in nucleotide positions 200 and 201, and each sequence from GWH-acceptor has dimer `AG` in positions 198 and 199. These dimers indicate true splice sites for positive examples, whereas they do not represent splice sites for negative examples.

For the GWH data, there is a substantial imbalance between the numbers of positive and negative examples: only 0.21% (0.29%) of the examples are positive for GWH-donor(acceptor). To see the effect of having this imbalance in training, we randomly sampled negative examples and used them, thus varying the so-called *decoy rate*[1] (Sonnenburg et al., 2007) from 5 to 15.

While the GWH datasets are for two-class classification (splice sites or not), the UCSC datasets (Kent et al., 2002) are for three-class classification (donor, acceptor, or neither). The UCSC-hg38 dataset contains 24,279 genes with 1–173 (on average 9.44) exons per gene. The UCSC datasets also consist of the sequences from 24 chromosomes. Most of these exons have duplicates in the annotation database due to alternative splicing (Keren et al., 2010). We randomly chose 63,454 unique exons out of 229,255. According to Noordewier et al. (1990), we then generated three examples by taking the sequences centered at the left, middle, and right boundaries of each exon. They correspond to acceptor, non-site, and donor examples, respectively. The UCSC-hg19 dataset was also utilized to generate additional examples in the same manner. Both UCSC datasets include non-canonical splice signals (*i.e.*, other than `GT` and `AG`) in addition to canonical signals.

We carried out all the experiments using MATLAB. For comparison with SVM, we used the LIBSVM package (Chang & Lin, 2011). Deepmat code (Cho et al., 2010) was used for the implementation of persistent CD and parallel tempering.

In the following, the architecture of a DBN is denoted by the number of nodes in each layer. For instance, a 1592-160-16-2 DBN has four layers with 1592 input units, 160 units in the first hidden layer, and so on.

### 4.2. Improved Prediction Performance and Runtime

To evaluate the prediction performance of our approach, we measured the F1-score and accuracy values[2], and the runtime, as shown in Fig. 4. For comparison,

[1] Denoted by $r =$ # negative samples/# positive samples

[2] F1-score $= 2TP/(2TP + FP + FN)$, accuracy $= (TP + TN)/(P + N)$

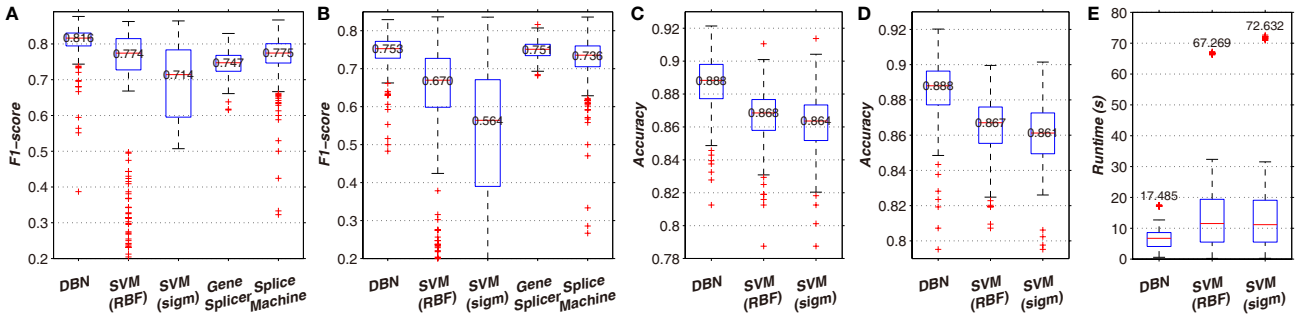


*Figure 4.* Comparison of classification performance: (a) F1-score for GWH-donor; (b) F1-score for GWH-acceptor; (c) accuracy for UCSC-hg19; (d) accuracy for UCSC-hg38; (e) runtime for UCSC-hg38 (chromosome 1).

we also included SVM (with RBF and sigmoid kernels) and two existing tools for splice junction prediction: GeneSplicer (Pertea et al., 2001) and SpliceMachine (Degroeve et al., 2005), which are based on decision trees and linear SVM, respectively. Note that these two existing tools were designed only for two-class problems and we were able to test them for the GWH data only. We used $(398 \times 4)$-160-16-2 DBN for the GWH data and $(60 \times 4)$-120-30-3 DBN for the UCSC data.

As shown in Fig. 4(a) and (d), the proposed method outperformed the existing state-of-the-art prediction tools in terms of the F1-score and accuracy. Note that we performed 10-fold cross validation for each of the 24 individual chromosomes to test a single dataset and each of the boxplots shows the distribution of 240 values. Quantitatively, our method showed 4.1–10.2% and 0.2–18.9% higher performance in terms of the median F1-score for the GWH-donor and GWH-acceptor datasets, respectively. For the UCSC datasets, our method produced 2.0–2.4% and 2.1–2.7% higher median accuracy than the SVM method.

Fig. 4(e) shows the runtime of the three different methods measured on the chromosome 1 dataset (the largest one) in UCSC-hg38. The proposed approach ran 3.86 times faster and 4.15 times faster than the SVM with the RBF and sigmoid kernel functions, respectively, in the worst case. Although the computational complexity of DBN depends on the numbers of layers and nodes, we still expect that DBNs will run faster than SVM for large datasets. This is because SVMs require quadratic programming, which takes $O(N^2)$ time with standard interior-point methods (Alizadeh, 1995). On the other hand, DBN takes only $O(N)$ because the number of computations for updating gradients is proportional to $N$.

### 4.3. More Robust Prediction by Proposed Approach

We further tested our approach in terms of robustness to the input sequence and imbalance in training examples, as shown in Fig. 5. The dataset used came from the chromosome 20 part in the GWH-acceptor data.

Fig. 5 (a1–a3) shows how the performance measures[3] (F1-score, precision, and recall) change as we vary the sequence length $m$ from 38 to 398 with decoy-rate $r$ fixed at 5. The DBN architecture was accordingly changed from $(38 \times 4)$-160-16-2 to $(398 \times 4)$-160-16-2. With increasing $m$, the SVM method produced slowly increasing precision, while rapidly decreasing recall and the F1-score. This indicates that longer sequences incur a rapid increase in the number of false negatives for SVM. Thus, SVM is apt to miss true splice sites if too many bases are considered around a `GT` or `AG` dimer. Longer sequences also resulted in reduced F1-scores for DBN, but the degree of reduction was noticeably smaller for our DBN method, which indicates a higher level of robustness to sequence lengths. DBN achieved the best F1-score at 77.13% with length $m = 138$.

To see the effect of decoy-rate $r$, we varied $r$ from 5 to 15 for $m = 138$ with $(138 \times 4)$-160-16-2 DBN and measured the F1-score, precision, and recall, as shown in Fig. 5 (b1–b3). Similarly to the experiments on sequence lengths, the proposed DBN approach outperformed the alternative in terms of $r$, suggesting that our approach can cope with the imbalance in training samples better. For instance, as $r$ increased, so did the precision of SVM, simply because it predicted the label of most examples to be negative. The other measures (recall and F1-score) of SVM decreased more significantly than those of DBN as we increased $r$.

### 4.4. Effects of Regularization on Performance

There exist conventional RBM-based approaches to modeling categorical data by normalizing the probabilities of binary units in a softmax way (Salakhutdinov et al., 2007). By contrast, our approach utilizes a regularization term for applying RBM to model discrete data, allowing the sum of probabilities to slightly deviate from 1 if that is helpful for

[3] Precision $= TP/(TP + FP)$, recall $= TP/(TP + FN)$

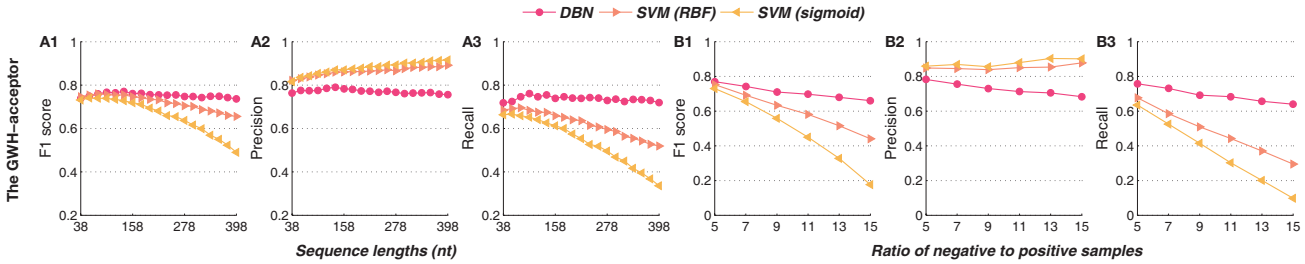


*Figure 5.* Effects of sequence length $m$ and decoy rate $r$ on performance: (a) varying $m$ from 38 to 398 with fixed $r = 5$; (b) varying $r$ from 5 to 15 with fixed $m = 138$. [data: chromosome 20 in GWH-acceptor]

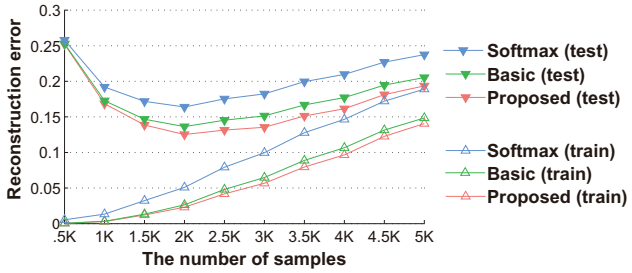


*Figure 6.* Comparing three types of RBMs (basic, softmax, and proposed) in terms of the reconstruction error. [data: chromosome 19 in GWH-donor, sequence length: 200nt, # iterations: 500, learning rate: 0.1, L2-decay: $10^{-3}$, $\lambda_c$: 0.02]

minimizing energy.

For a performance comparison, we measured the training and test error of three types of RBMs (basic, softmax, and regularized) using samples of the chromosome 19 sequences in the GWH-donor data. In the results shown in Fig. 6, the proposed regularization-based RBM consistently outperformed the softmax version in training and test error. When the number of samples is less than 2,000, we observe overfitting (*i.e.*, decreasing training error with increasing test error) for all the approaches compared.

A part of the motivation for our approach comes from the tradeoff between minimizing energy and maintaining the probability sum at 1. Given that having low energy is likely to produce low reconstruction error, the proposed regularized RBM succeeded in achieving lower error by slightly sacrificing the probability sum constraint. Leveraged by the regularization term, our approach could find more appropriate hidden representations than the alternatives.

### 4.5. Efficient RBM Training by Boosted CD

To further validate the proposed boosted CD training, we tested it with a modified version of the MNIST dataset (LeCun et al., 1998). To simulate a class-imbalance situation, we randomly dropped observations with different drop rates for different classes and created two training sets (with 12,000 and 25,000 samples each) and a test set (with 10,000 samples). We repeated this procedure 10 times.

*Table 3.* Test accuracy by different training methods

| Method ↓ # of samples → | 12,000 | 25,000 | 60,000 |
|---|---|---|---|
| Boosted CD (proposed) | 96.09% | 95.69% | 98.23% |
| CD | 94.49% | 94.36% | 98.17% |
| Persistent CD | 45.58% | 46.46% | 98.36% |
| Parallel tempering† | 95.84% | 95.74% | 98.52% |

†approximately 10 times slower than boosted CD

Table 3 lists the classification accuracy (averaged over the 10 runs) obtained by a 784-200-100-10 DBN trained with four different methods using the test data. As reference, the table also shows the accuracy values from the unmodified MNIST dataset (with 60,000 training examples).

For the two class-imbalance cases, proposed boosted CD and parallel tempering showed the best performance. However, due to the need for extra sampling, the time demand of parallel tempering was approximately 10 times higher to achieve the level of accuracy comparable to boosted CD. The performance of persistent CD was notably deteriorated for the class-imbalance cases. In such cases, the Gibbs sampler in training would hardly draw samples from low-density (but important) regions. Because training by persistent CD continues sampling from the Gibbs chain of the previous iterations, errors may have accumulated, giving unsatisfactory performance.

### 4.6. Identification of Non-Canonical Splice Sites

To recognize non-canonical splice sites, such as `GC` pairs at donor sites, we trained the models using a combination of the two UCSC datasets. The merged dataset contained 378,819 sequences and the numbers of donors with `GT` pairs and acceptors with `AG` pairs were 109,000 (86.32% of donors) and 106,868 (84.63% of acceptors), respectively. First, we trained the 240-120-3 DBN with both the UCSC-hg19 and UCSC-hg38 datasets, including canonical splicing. The other parameters were the same as in the previ-

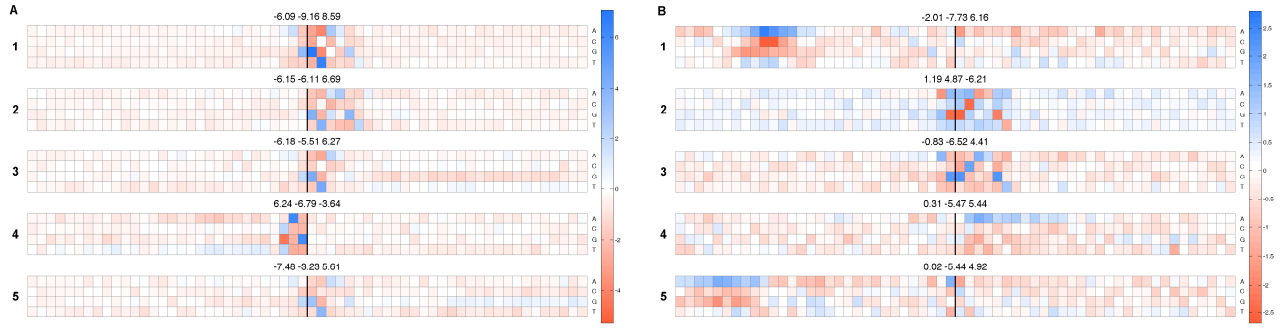


*Figure 7.* Top discriminative features for UCSC datasets: (a) including canonical splicing sites; (b) excluding canonical splicing sites.

```
1 NNNNNNNNAAAAANNNNNNNNNNNNNNNNN|NNNNNNNNNNNNNNNNNNN
2 NNNNNNNNNNNNNNNNNNNNNNNNNNNNAG|GCAAGTNNNNNNNNNNNNN
3 NNNNNNNNNNNNNNNNNNNNNNNNNNNNAG|GCANGTNNNNNNNNNNNNN
4 NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN|NAAAAAANANNNNNNNNNN
5 NNAAAAANANNNNNNNNNNNNNNNNNNNNA|NNNNNNNNNNNNNNNNNNN
```

*Figure 8.* The most likely sequences representing non-canonical splice sites inferred from analyzing Fig. 7(b).

ous experiments. The training yielded two weight matrices, $W_1 \in \mathbf{R}^{240\times120}$ and $W_2 \in \mathbf{R}^{120\times3}$. Here, we can regard the two matrices as a set of 120 feature vectors and the discriminative scores for the three classes, respectively. In other words, each column in $W_1$ is a feature vector and its corresponding row in $W_2$ is composed of three weight values for the three classes.

The discriminability of a feature vector can be defined by the variance of the corresponding row in $W_2$. That is, a feature vector is not capable of discriminating any classes when the differences among the three discriminative scores of the feature vector are close to zero. The feature vectors were ranked in order of discriminability and the five most discriminative patterns are shown in Fig. 7(a). The vectors were reshaped into matrix forms (each row denotes `A`, `C`, `G`, and `T`) and colored according to weight values. A darker blue represents a higher positive value, whereas a darker red represents a lower negative value. For example, we can infer the most likely sequence from Fig. 7(a1) as '..`GT`..' because the third and fourth rows at the boundary are bold blue. The three numbers presented above each template represent the three discriminative scores for acceptor, non-boundary, and donor, from left to right.

As expected, we observed that the most discriminative features of Fig. 7(a) are '..`GT`..' (a1, a2, a3, and a5) and '..`AG`..' (a4). Because only a few examples of non-canonical splicing were included, it was hard for the feature vector to detect the subtle signals. Therefore, we trained the same DBN again, using the 162,951 examples that remained after excluding the canonical splice sites. Fig. 7(b) shows the five best patterns for the non-canonical sites. We can derive the most likely sequences from the best weight vectors, as seen in Fig. 8. We found that non-canonical splicing arose when introns contained `GCA` or `NAA` sequences at their boundaries or contiguous `A`'s occurred around the boundaries in exon regions. Using these methods, we may be able to reveal more novel patterns related to exons and alternative splicing that otherwise cannot be identified using existing machine learning techniques.

## 5. Conclusion

We have presented a novel DBN-based approach to splice site prediction at DNA level. Our contributions include the following:

- A new RBM training method called *boosted CD with categorical gradients* that improves conventional CD;
- Significant boosts in splice site prediction in terms of accuracy and runtime, along with reduced susceptibility to sequence lengths;
- Increased robustness when handling high-dimensional class-imbalanced data;
- The ability to detect subtle non-canonical splicing signals that often could not be identified by traditional methods.

Given the accuracy, efficiency, and robustness of our DBN-based methodology, we anticipate that it can be extended to the discovery of primary structural patterns of other genomic elements that are often too subtle to detect using existing computational techniques.

## Acknowledgments

This work was supported in part by the National Research Foundation (NRF) of Korea grants funded by the Korean Government (Ministry of Science, ICT and Future Planning) [No. 2011-0009963 and No. 2014M3C9A3063541], in part by the ICT R&D program of MSIP/ITP [14-824-09-014, Basic Software Research in Human-level Lifelong Machine Learning (Machine Learning Center)], in part by SNU ECE Brain Korea 21+ project in 2015, and in part by Samsung Electronics Co., Ltd.

## References

Alizadeh, Farid. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

Au, Kin Fai, Jiang, Hui, Lin, Lan, Xing, Yi, and Wong, Wing Hung. Detection of splice junctions from paired-end RNA-seq data by SpliceMap. *Nucleic Acids Research*, 38(14):4570–4578, 2010.

Baldi, Pierre and Brunak, Søren. Chapter 6. neural networks: applications. In *Bioinformatics: The Machine Learning Approach*. MIT press, 2001.

Baten, Abdul KMA, Chang, Bill CH, Halgamuge, Saman K, and Li, Jason. Splice site identification using probabilistic parameters and SVM classification. *BMC Bioinformatics*, 7(Suppl 5):S15, 2006.

Bengio, Yoshua. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Brunak, Søren, Engelbrecht, Jacob, and Knudsen, Steen. Prediction of human mRNA donor and acceptor sites from the DNA sequence. *Journal of Molecular Biology*, 220(1):49–65, 1991.

Burset, M, Seledtsov, IA, and Solovyev, VV. Analysis of canonical and non-canonical splice sites in mammalian genomes. *Nucleic Acids Research*, 28(21):4364–4375, 2000.

Chang, Chih-Chung and Lin, Chih-Jen. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

Cho, KyungHyun, Raiko, Tapani, and Ilin, Alexander. Parallel tempering is efficient for learning restricted Boltzmann machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2010.

Chu, Yongjun and Corey, David R. RNA sequencing: platform selection, experimental design, and data interpretation. *Nucleic Acid Therapeutics*, 22(4):271–274, 2012.

Degroeve, Sven, Saeys, Yvan, De Baets, Bernard, Rouzé, Pierre, and Van de Peer, Yves. SpliceMachine: predicting splice sites from high-dimensional local context representations. *Bioinformatics*, 21(8):1332–1338, 2005.

Erhan, Dumitru, Bengio, Yoshua, Courville, Aaron, Manzagol, Pierre-Antoine, Vincent, Pascal, and Bengio, Samy. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

Grant, Gregory R., Farkas, Michael H., Pizarro, Angel D., Lahens, Nicholas F., Schug, Jonathan, Brunk, Brian P., Stoeckert, Christian J., Hogenesch, John B., and Pierce, Eric A. Comparative analysis of RNA-Seq alignment algorithms and the RNA-Seq unified mapper (RUM). *Bioinformatics*, 27(18):2518–2528, 2011.

Hinton, Geoffrey E. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14 (8):1771–1800, 2002.

Hinton, Geoffrey E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Huang, J., Li, T., Chen, K., and Wu, J. An approach of encoding for prediction of splice sites using SVM. *Biochimie*, 88(7):923–929, 2006.

Kent, W. James, Sugnet, Charles W., Furey, Terrence S., Roskin, Krishna M., Pringle, Tom H., Zahler, Alan M., Haussler, and David. The human genome browser at UCSC. *Genome Research*, 12(6):996–1006, 2002.

Keren, Hadas, Lev-Maor, Galit, and Ast, Gil. Alternative splicing and evolution: diversification, exon definition and function. *Nature Reviews Genetics*, 11(5):345–355, 2010.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 1097–1105, 2012.

Lawrence, Charles E, Altschul, Stephen F, Boguski, Mark S, Liu, Jun S, Neuwald, Andrew F, and Wootton, John C. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262 (5131):208–214, 1993.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, Honglak, Pham, Peter T., Largman, Yan, and Ng, Andrew Y. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 1096–1104, 2009.

Lena, Pietro Di, Baldi, Pierre, and Nagata, Ken. Deep spatio-temporal architectures and learning for protein structure prediction. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 521–529, 2012.

Leung, Michael KK, Xiong, Hui Yuan, Lee, Leo J, and Frey, Brendan J. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129, 2014.

Lockhart, David J and Winzeler, Elizabeth A. Genomics, gene expression and DNA arrays. *Nature*, 405(6788): 827–836, 2000.

Neal, Radford M. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

Nilsen, Timothy W. and Graveley, Brenton R. Expansion of the eukaryotic proteome by alternative splicing. *Nature*, 463(7280):457–463, 2010.

Noordewier, Michiel O, Towell, Geoffrey G, and Shavlik, Jude W. Training knowledge-based neural networks to recognize genes in DNA sequences. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 530–536, 1990.

Pertea, Mihaela, Lin, Xiaoying, and Salzberg, Steven L. GeneSplicer: a new computational method for splice site prediction. *Nucleic Acids Research*, 29(5):1185–1190, 2001.

Reed, Scott, Sohn, Kihyuk, Zhang, Yuting, and Lee, Honglak. Learning to disentangle factors of variation with manifold interaction. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 1431–1439, 2014.

Reese, Martin G, Eeckman, Frank H, Kulp, David, and Haussler, David. Improved splice site detection in Genie. *Journal of Computational Biology*, 4(3):311–323, 1997.

Salakhutdinov, Ruslan, Mnih, Andriy, and Hinton, Geoffrey. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 791–798, 2007.

Sonnenburg, Soren, Schweikert, Gabriele, Philips, Petra, Behr, Jonas, and Ratsch, Gunnar. Accurate splice site prediction using support vector machines. *BMC Bioinformatics*, 8(Suppl 10):S7, 2007.

Stormo, Gary D, Schneider, Thomas D, Gold, Larry, and Ehrenfeucht, Andrzej. Use of the 'Perceptron' algorithm to distinguish translational initiation sites in E. coli. *Nucleic Acids Research*, 10(9):2997–3011, 1982.

Tieleman, Tijmen. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 1064–1071, 2008. ISBN 1605582050.

Trapnell, Cole, Pachter, Lior, and Salzberg, Steven L. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, 25(9):1105–1011, 2009.

Trapnell, Cole, Williams, Brian A., Pertea, Geo, Mortazavi, Ali, Kwan, Gordon, van Baren, Marijke J., Salzberg, Steven L., Wold, Barbara J., and Pachter, Lior. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, 2010.

Wang, Kai, Singh, Darshan, Zeng, Zhen, Coleman, Stephen J., Huang, Yan, Savich, Gleb L., He, Xiaping, Mieczkowski, Piotr, Grimm, Sara A., Perou, Charles M., MacLeod, James N., Chiang, Derek Y., Prins, Jan F., and Liu, Jinze. MapSplice: accurate mapping of RNA-seq reads for splice junction discovery. *Nucleic Acids Research*, 38(18):e178, 2010.

Wang, Yuhao and Zeng, Jianyang. Predicting drug-target interactions using restricted Boltzmann machines. *Bioinformatics*, 29(13):126–134, 2013.