Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

Sequence2Vec: A novel embedding approach for modeling transcription factor binding affinity landscape

Hanjun Dai[1,*], Ramzan Umarov[2,*], Hiroyuki Kuwahara[2], Yu Li[2], Le Song[1,†], and Xin Gao[2,†]

[1] College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA.
[2] King Abdullah University of Science and Technology (KAUST), Computational Bioscience Research Center (CBRC), Computer, Electrical and Mathematical Sciences and Engineering (CEMSE) Division, Thuwal, 23955-6900, Saudi Arabia.

[†] All correspondence should be addressed to Xin Gao (xin.gao@kaust.edu.sa) and Le Song (lsong@cc.gatech.edu).

Associate Editor: Dr. Oliver Stegle

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: An accurate characterization of transcription factor (TF)-DNA affinity landscape is crucial to a quantitative understanding of the molecular mechanisms underpinning endogenous gene regulation. While recent advances in biotechnology have brought the opportunity for building binding affinity prediction methods, the accurate characterization of TF-DNA binding affinity landscape still remains a challenging problem.

Results: Here we propose a novel sequence embedding approach for modeling the transcription factor binding affinity landscape. Our method represents DNA binding sequences as a hidden Markov model (HMM) which captures both position specific information and long-range dependency in the sequence. A cornerstone of our method is a novel message passing-like embedding algorithm, called Sequence2Vec, which maps these HMMs into a common nonlinear feature space and uses these embedded features to build a predictive model. Our method is a novel combination of the strength of probabilistic graphical models, feature space embedding and deep learning. We conducted comprehensive experiments on over 90 large-scale TF-DNA data sets which were measured by different high-throughput experimental technologies. Sequence2Vec outperforms alternative machine learning methods as well as the state-of-the-art binding affinity prediction methods.

Availability: Our program is freely available at https://github.com/ramzan1990/sequence2vec.

1 Introduction

In 1967, seminal works in the phage λ switch and the lactose metabolism in *Escherichia coli* discovered that repressors interact with short DNA segments in a sequence-specific fashion to prevent transcription of downstream genes (Ptashne, 1967; Gilbert and Müller-Hill, 1967). Interactions of transcription factors (TFs) with DNA binding sites have since then been appreciated as general molecular mechanisms to control the recruitment of RNA polymerase and to regulate transcription (Alberts *et al.*, 2002). Importantly, TFs discriminate DNA binding sites and selectively interact with specific sequence targets, allowing cells to orchestrate a spatiotemporal regulation of their endogenous genes (Von Hippel and Berg, 1986; Ptashne, 2013).

A key determinant of the sequence specificity is affinity differences of all possible binding sites in the genome; in the *in vivo* concentration range, each TF molecule diffuses around and recognizes a DNA site with higher binding affinity much more frequently than that with lower binding affinity. Thus, an accurate characterization of TF-DNA affinity landscape is crucial to a quantitative understanding of the molecular mechanisms underpinning endogenous gene regulation. This knowledge is also expected to facilitate better insights into how phenotypes are established and maintained (Davidson, 2006), how organisms might have evolved their gene regulatory systems (Wray, 2007), and how artificial gene circuits can be designed and optimized (Gertz *et al.*, 2009; Kuwahara *et al.*, 2013; Fan

[*] The first two authors contributed equally to this work.

et al., 2015). The most commonly used approach to describe the specificity of each TF is to use a consensus DNA sequence based on the assumption that the nucleotides in a sequence-specific binding site contribute independently to the binding affinity. However, several studies gave compelling evidence that the TF-DNA binding affinities could depend strongly on their context (Hochschild and Ptashne, 1986; Bulyk *et al.*, 2002). In particular, by showing strong interdependence among nucleotides in the binding affinity landscape of zinc finger TFs, Bulyk *et al.* (2002) suggested that accurate prediction of binding affinity would require an affinity data set with large numbers of sequence variants. With advances in biotechnology, there are now several *in vitro* high-throughput methods available to measure TF-DNA binding affinity of large numbers of DNA variants. These methods include a DNA microarray-based method called *protein-binding microarray* (PBM) (Berger and Bulyk, 2006), a microfluidic-based method called *mechanically induced trapping of molecular interactions 2.0* (MITOMI 2.0) (Fordyce *et al.*, 2010), and a second-generation DNA sequencing-based method called *high-throughput sequencing – fluorescent ligand interaction profiling* (HiTS-FLIP) (Nutiu *et al.*, 2011). Although the increase in the availability of high-throughput data sets from such methods has propelled the development of data-driven prediction methods (e.g., Berger and Bulyk, 2009; Agius *et al.*, 2010; Lee *et al.*, 2011; Annala *et al.*, 2011; Wang *et al.*, 2014; Barrera *et al.*, 2016; Wong *et al.*, 2013; Hassanzadeh and Wang, 2016; Wong *et al.*, 2015; Zhou *et al.*, 2016), the accurate characterization of TF-DNA binding affinity landscape still remains a challenging problem that has not been entirely solved (Levo *et al.*, 2015; Deplancke *et al.*, 2016).

In this paper, we propose a novel embedding approach to the modeling of the TF binding affinity landscape, which combines the strength of probabilistic graphical models, feature space embedding, and deep learning. Our method represents DNA binding sequences as a hidden Markov model (HMM). However, instead of performing standard maximum likelihood estimation for these models, we devised a new message passing-like embedding approach, called Sequence2Vec, which maps these HMMs into a common nonlinear feature space and uses these embedded features to build a nonlinear predictive model. Importantly, unlike many existing methods that consider the feature extraction and regression as two separate steps (e.g., Stormo, 2000; Bulyk *et al.*, 2002; Liu *et al.*, 2002; Foat *et al.*, 2006; Chen *et al.*, 2007; Berger and Bulyk, 2009; Annala *et al.*, 2011; Agius *et al.*, 2010; Lee *et al.*, 2011; Wang *et al.*, 2014; Siebert and Söding, 2016), Sequence2Vec enables end-to-end learning of nonlinear features together with the predictive model directly from the data. In addition, since embedded features that are derived from an HMM can capture potential long-range dependencies in a sequence, a TF-DNA binding affinity landscape model learned from our new method can be used to illuminate how the sequence contexts, such as flanking regions of a core sequence motif (Nutiu *et al.*, 2011; Levo *et al.*, 2015), can quantitatively affect the TF-DNA interaction. Comprehensive experiments demonstrated that the proposed method significantly outperforms previous state-of-the-art methods and learns meaningful sequence motifs.

2 Related Work

2.1 Position weight matrix and linear models

A common technique to characterize binding affinity is the position weight matrix (PWM). PWMs characterize the DNA sequence preference of a TF as a $D \times L$ matrix, where D is the number of possible bases (4 for DNA), and L is the length of the binding sequences. There are different variants of PWM, but their common characteristic is that they assume independence of base positions. Methods have also been developed to encode short-range information by building larger matrices for subsequences (k-mers) (Bulyk *et al.*, 2002; Berger and Bulyk, 2009). Recently, Siebert and Söding proposed a Bayesian method for motif discovery by encoding sequence order dependency through Markov models (Siebert and Söding, 2016). Their method significantly outperformed PWM on discovering motifs from ChIP-seq data.

Annala *et al.* (2011) proposed a linear model, HK→ME, that represents binding affinity as the sum of the binding affinity contributions of the constituent subsequences. The k-mers present in the training sequences are represented as a design matrix H, so that $h_{s,t} = 1$, if k-mer t is found in sequence s, and 0 otherwise. The k-mer affinity contributions are obtained by solving a linear system $p = H\alpha + \epsilon$, where p is a vector containing binding affinities of the training sequences, α is a vector of k-mer affinity contributions, and ϵ presents noise. This method was ranked top in the Dialogue for Reverse Engineering Assessment and Methods 5 (DREAM5) TF-DNA Motif Recognition Competition.

2.2 Kernel methods

Kernel methods are a successful family of methods for building predictive models (Schölkopf *et al.*, 2004). Such methods work by first defining a so-called kernel function between pairs of inputs, and then learning a predictive model based on these kernel function values. One can think of these functions as a similarity measure where a pair of inputs, χ and χ', are first transformed into a common feature space, $\phi(\chi)$ and $\phi(\chi')$ respectively, and then their inner product is used to define the kernel, i.e., $k(\chi, \chi') = \langle \phi(\chi), \phi(\chi') \rangle$ (Leslie *et al.*, 2002, 2004; Rätsch *et al.*, 2005).

Many string kernels are designed based on the idea of "bag of k-mers", where each sequence is represented as a vector of counts for short k-mers. For instance, the spectrum kernel and variants for strings fall into this category (Leslie *et al.*, 2002). Recently, Wang *et al.* (2014) proposed a two round support vector regression (SVR) model based on weighted degree kernels. Among different types of string kernels, Wang *et al.* (2014) showed that weighted degree (WD) kernels with shifts and mismatches (Leslie *et al.*, 2004; Rätsch *et al.*, 2005) work best for binding affinity prediction by accounting for alternations in DNA subsequences. Their method showed significant improvements over PWM and HK→ME (Wang *et al.*, 2014). However, the feature design in these kernels is fixed before learning, with each dimension corresponding to a particular k-mer which is independent of the supervised learning task at hand. Furthermore, typically only short k-mers are used to keep the kernel computation tractable.

Another class of kernels uses probabilistic graphical models (GMs) to describe the noisy and structured data, and then designs kernels based on the GMs. For instance, one can use HMMs for sequence data, which is the key idea of the Fisher kernel (Jaakkola and Haussler, 1999). Typically the parameterization of these GM kernels is chosen before hand. Although the process of fitting generative models allows the kernels to adapt to the geometry of the input data, the resulting feature representations are still independent of the discriminative task at hand.

2.3 Deep learning approach

In recent years, deep learning methods have achieved the state-of-the-art performance in many machine learning applications (Bengio, 2009; Schmidhuber, 2014). Deep learning models, unlike their predecessor artificial neural networks (ANNs), consist of multiple hidden layers which make them more expressive in modeling complex structures in input data than their shallow counterparts.

Multi-layer neural networks (DNNs), where the hidden layers are fully connected, have been successfully applied to computational biology problems, such as learning the tissue-regulated splicing code (Leung *et al.*, 2014). However, such fully connected layers do not take into account the sequence nature of DNA and may require a lot of examples to train.

Convolutional neural networks (CNNs), where the first layer contains convolutions, have also been applied to computational biology problems recently. For instance, Alipanahi *et al.* (2015) developed a CNN-based method, DeepBind, to predict the binding sites of DNA- and RNA-binding proteins. DeepBind takes input sequences and feeds them into a convolutional layer which detects motifs in those sequences. The convolutional layer essentially consists of filters (motif detectors) which are small matrices, for example $k \times D$ where k is called the filter length (motif length) and D is the dimension of the 1-hot representation of each sequence position ($D = 4$ for DNA sequences). These filters are convolved with the input, i.e. they are moved spatially across the input and the dot product is calculated at each position, which results in feature maps (motif scan). The next stage is a $ReLU$ layer which is referred to as a rectified motif scan. After convolution and rectification, there is a max pooling layer, which takes the feature maps from all the filters as inputs and operates on each feature map to reduce the spatial dimension. The output is then fed into a fully connected layer which learns how to combine the motif detectors and produces a real-valued score.

3 Methods

In this section, we present a novel method, Sequence2Vec, for embedding DNA sequences into nonlinear feature spaces, and learning a regression model from features to the binding affinity. Our method is designed to take two aspects into account: long-range interaction between distant positions, and joint learning of sequence features and the regression model.

3.1 Overview

Our method models DNA sequences as an HMM where each nucleotide in the sequence is associated with an observed variable and a latent variable, and these latent variables are linked together by a Markov chain (see Figure 1 for illustration). Note that for one transcription factor, there is one single HMM and each sequence is an independent realization of it. In such a model, the posterior distribution of each latent variable given the entire input DNA sequence are supposed to be good sequence features, since they are generally different in different nucleotide positions (position specific), and they are conditioned on the entire DNA sequence (captures potential long-range interaction) (Section 3.2).

To extract these features, traditional graphical model approaches need to first learn the model parameters and then perform inference, such as message passing, to compute these posteriors. However, these feature extraction steps can be time-consuming and furthermore the parameterization of the model is independent of the binding affinity prediction task. We will instead design a nonlinear embedding approach which combines the graphical model learning and inference (message passing) steps using an alternative parameterization (Section 3.3).

More specifically, suppose we are provided with a training data set $\mathcal{D} = \{\chi^{(n)}, y^{(n)}\}_{n=1}^{N}$, where each $\chi^{(n)}$ is a DNA sequence and $y^{(n)} \in \mathbb{R}$ is the corresponding binding affinity value. The DNA sequence $\chi^{(n)} = [x_1^{(n)}, x_2^{(n)}, x_3^{(n)}, \ldots, x_L^{(n)}]$ can be considered as a string of length L, where the character $x_i^{(n)}$ at position i is taking values from the alphabet $\Sigma = \{A, T, C, G\}$. Furthermore, x_i is of dimension $|\Sigma|$, which is represented by one hot vector that encodes the label from the alphabet. For simplicity of exposition, we assume L is a constant within a given data set. That is, the DNA sequences are of the same length, though our method can work with sequences of different lengths in general.

The nonlinear feature embedding $f : \Sigma^L \mapsto \mathbb{R}^m$ will transform each DNA sequence $\chi^{(n)}$ into a vector of dimension m. Furthermore, we will design f to be a composition of two nonlinear operations $h : \Sigma^L \mapsto \mathbb{R}^{d \times L}$ and $g : \mathbb{R}^{d \times L} \mapsto \mathbb{R}^m$, such that

$$f(\chi^{(n)}) = g(h(\chi^{(n)})), \;\text{ and }\; h(\chi^{(n)}) = [\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_L]. \tag{1}$$

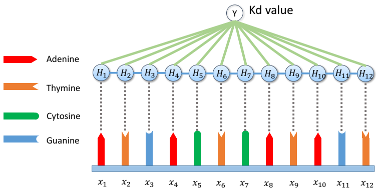


Fig. 1: The proposed graphical model for embedding a 12-mer DNA binding sequence. The x_i is the nucleotide at position i in the binding sequence, H_i is the hidden variable at position i, and K_d is the affinity value of this binding sequence.

Here $\vec{\mu}_i$ captures features at position i and potential long-range interactions, and g is an aggregation operator which generates a fixed dimensional vector representation for the entire sequence.

The nonlinear embedding operators will be parameterized as deep learning models (Section 3.4), and we will learn them jointly with the regression model (Sections 3.5, 3.6, and 3.7). That is, we will learn these models via mean square error minimization,

$$\min_{f, \vec{v}} l(f, \vec{v}) := \frac{1}{2} \sum_{n=1}^{N} \left(y^{(n)} - \vec{v}^{\top} f(\chi^{(n)}) \right)^2, \tag{2}$$

where $\vec{v} \in \mathbb{R}^m$ is the parameters for the linear regression. Since f can already extract nonlinear features, the linear regression model here is expressive enough to model complex relations. Furthermore, we will learn all model parameters end-to-end using stochastic gradient descent, which makes it very scalable for large data sets.

Finally, to understand the trained model and the sequence features, we also propose an approach to visualizing these features (Section 3.8). We will explain the details of these steps in the subsections below.

3.2 Graphical model of a sequence

In our work, we model DNA sequences as an HMM. However, instead of explicitly learning its parameters (e.g., transition probabilities and emission probabilities), we will learn it discriminatively via a novel embedding process. The theoretical background can be referred to (Dai *et al.*, 2016).

Formally, the nucleotide at each position corresponds to an observed variable x_i. To take into account potentially noisy observation and long-range interactions, an additional hidden variable H_i is associated to each observation x_i (see Figure 1 for illustration).

Given the graphical model parameters (or the potentials), the posterior marginals for the hidden variables, $p(H_i \mid \{x_j\}_{j=1}^{L})$ for $i = 1, \ldots, L$, can be viewed as the features of a sequence. Such features capture sequence uncertainties, since they model distributions; they also capture position specific information, since each position is associated with such a distribution; and furthermore, they capture long-range dependencies, since each distribution is conditioned on the entire sequence.

However, computing these posterior distributions (i.e., the inference problem in graphical models) is not a trivial task and involves multidimensional integration. Fortunately, the forward and backward message passing (Pearl, 2001; Bishop, 2006) has been designed to perform this operation efficiently.

More specifically, let $m_{i,j}$ be the message passed from position i to position j. During the forward pass, the algorithm sets $m_{0,1} = 1$, and computes messages from variable H_i to H_{i+1}, which are unnormalized distributions, for $i = 1, \ldots, L-1$, as

$$m_{i,i+1}(H_{i+1}) := \mathcal{T}_1 \circ (H_{i+1}, x_i, m_{i-1,i}(H_i)), \tag{3}$$

where $\mathcal{T}_1$ is an abstract marginalization operator defined on H_{i+1}, x_i and the incoming message $m_{i-1,i}$. We can define the backward message $m_{i,i-1}$ similarly with the same operator.

Denote $q_i(H_i) = p(H_i|\,\{x_j\}_{j=1}^{L})$. For $i = 2, \ldots, L-1$, we have

$$q_i(H_i) := \mathcal{T}_2 \circ (H_i, m_{i-1,i}(H_i), m_{i+1,i}(H_i))\,, \tag{4}$$

where $\mathcal{T}_2$ is another abstract product operator defined on H_i and two incoming messages, $m_{i-1,i}$ and $m_{i+1,i}$.

Typically, EM or other local search heuristics need to be used for learning the HMM parameters. During learning, one also needs to perform multiple rounds of forward and backward message passing. In summary, although q_i's are good features for representing sequences, they are not easy to compute. Furthermore, it is not clear how good q_i is for the predictive task since the parametric form of the distribution is chosen before seeing the binding affinity values. We will instead learn an alternative representation of the graphical model which simultaneously considers both the learning and inference steps.

3.3 Feature space embedding of distributions

We use the idea of distribution embedding to represent the posteriors. We will first provide a brief review. *Feature (or Hilbert) space embedding of distributions* maps distributions into nonlinear feature spaces (Borgwardt *et al.*, 2006; Smola *et al.*, 2007; Song *et al.*, 2007),

$$\vec{\mu}_H \;:=\; \mathbb{E}_H\left[\phi(H)\right] \;=\; \int_{\mathcal{H}} \phi(H)\, q(H)\, dH, \tag{5}$$

where $\phi(H)$ is a generic nonlinear feature transformation. When the embedding is injective (Sriperumbudur *et al.*, 2008), μ_H can be viewed as a sufficient statistics of the original density $q(H)$. Such nonparametric embedding approach has been successfully applied to other computational biology problems (Borgwardt *et al.*, 2006; Song *et al.*, 2007).

The injective property of feature space embedding allows us to express computation on distributions using their embeddings only. We will extensively exploit this property of injective embeddings to express the message passing algorithm in HMMs, by assuming that there exists a rich enough nonlinear feature space such that the embeddings are injective.

Specifically, we embed the marginal distributions $p(H_i|\,\{x_j\})$ by $\vec{\mu}_i = \int_{\mathcal{H}} \phi(H_i)\, p(H_i|\,\{x_j\})\, dH_i$, and the messages $m_{i,j}$ as $\vec{\nu}_{ij} = \int_{\mathcal{H}} \phi(H_j)\, m_{ij}(H_j)\, dH_j$.

With the assumption that there is an injective embedding for each forward and backward message, $m_{i,i+1}$ and $m_{i,i-1}$, respectively, and for each posterior marginal $p(H_i|\,\{x_i\})$, we can express the message passing operation (3) and the marginal computation in (4) as

$$\vec{\nu}_{i,i+1} = \widetilde{\mathcal{T}}_1 \circ (x_i, \vec{\nu}_{i-1,i})\,, \tag{6}$$

$$\vec{\nu}_{i,i-1} = \widetilde{\mathcal{T}}_1 \circ (x_i, \vec{\nu}_{i+1,i})\,, \tag{7}$$

$$\vec{\mu}_i = \widetilde{\mathcal{T}}_2 \circ (x_i, \vec{\nu}_{i+1,i}, \vec{\nu}_{i-1,i})\,, \tag{8}$$

where $\widetilde{\mathcal{T}}_1$ and $\widetilde{\mathcal{T}}_2$ are the operators corresponding to $\mathcal{T}_1$ and $\mathcal{T}_2$.

One can think of these embeddings as extracting some nonlinear features associated with the messages $m_{i,i+1}$ and $m_{i,i-1}$, and the posterior $p(H_i|\,\{x_j\})$ at position i. The nonlinear feature $\vec{\mu}_i$ represents position specific features which have already incorporated long-range dependencies using message passing.

3.4 Parameterizing the embedding operators

The operators $\widetilde{\mathcal{T}}_1$ and $\widetilde{\mathcal{T}}_2$ have a nonlinear dependency on the node and edge potentials of the HMM, and generally there is no restriction on the parametric form. Instead of first learning the graphical model potentials and then computing (or approximating) these nonlinear feature embeddings, we will directly parameterize the embedding operators using neural networks. We choose neural networks because they are universal approximators given a large enough number of hidden units (Barron, 1993), and they are also easy to learn using gradient based methods as we show later.

Algorithm 1 **Extracting position specific feature**

```
Input: parameter $\mathbf{W}$ in $\widetilde{\mathcal{T}}_1$ and $\widetilde{\mathcal{T}}_2$, and a sequence $\chi$.
Initialize $\vec{\nu}_{ij}^{(0)} = \vec{\mathbf{0}}$, for all $(i,j), |i-j| = 1, i, j = 1, \ldots, L$.
for $t = 1$ to $T$ do
    for $i = 1$ to $L$ do
        $\vec{\nu}_{i,i+1}^{(t)} = \sigma(W_1 x_i + W_2 \vec{\nu}_{i-1,i}^{(t-1)})$.
        $\vec{\nu}_{i,i-1}^{(t)} = \sigma(W_1 x_i + W_2 \vec{\nu}_{i+1,i}^{(t-1)})$.
    end for
end for
for $i = 1, \ldots, L$ do
    $\vec{\mu}_i = \sigma(W_3 x_i + W_4 \vec{\nu}_{i-1,i}^{(T)} + W_4 \vec{\nu}_{i+1,i}^{(T)})$.
end for
Return $h = [\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_L]$.
```

Specifically, we assume that d hidden units are used. In practice, d is chosen via cross-validation as we show in results. This will correspond to a nonlinear embedding of μ_i and v_{ij} into $\mathbb{R}^d$. Then

$$\vec{\nu}_{i,i+1} = \sigma\Big(W_1 x_i + W_2 \vec{\nu}_{i-1,i}\Big), \tag{9}$$

$$\vec{\nu}_{i,i-1} = \sigma\Big(W_1 x_i + W_2 \vec{\nu}_{i+1,i}\Big), \tag{10}$$

$$\vec{\mu}_i = \sigma\Big(W_3 x_i + W_4 \vec{\nu}_{i+1,i} + W_4 \vec{\nu}_{i-1,i}\Big), \tag{11}$$

where $\sigma(\cdot)$ is a nonlinear activation function. For example, if we use rectified linear unit, then $\sigma(\cdot) := \max\{0, \cdot\}$, and $\mathbf{W} = \{W_1, W_2, W_3, W_4\}$ is the collection of weights in the neural network, where $W_1, W_3 \in \mathbb{R}^{d \times |\Sigma|}$ and $W_2, W_4, \in \mathbb{R}^{d \times d}$. A bias term, which is typically used in a fully connected neural network, is also included.

3.5 Extracting position specific features

Once we have learned parameters $\mathbf{W} = \{W_1, W_2, W_3, W_4\}$ in the embedding operators $\widetilde{\mathcal{T}}_1$ and $\widetilde{\mathcal{T}}_2$ (we will learn them together with the regression model as explained in (2)), we can then extract position specific nonlinear features using the message passing algorithm. So far, for simplicity, we have explained the message passing algorithm with a full sequential forward and backward pass. However, to avoid the sequential dependency between the messages and for implementation efficiency, we use a parallel version of the message passing algorithm (Gonzalez *et al.*, 2009), where all messages are initialized to zeros at the first iteration; and then each node performs the forward and backward message updates simultaneously as in (9) and (10).

The algorithm to extract position specific features is summarized in Algorithm 1, which will also serve as the building block for our learning algorithm described later. Algorithm 1 performs T rounds of parallel message update iterations. The number of rounds, T, for the message update controls the range of dependencies captured by the algorithm. If we unroll these iterations into computational layers, then the algorithm can be viewed as a recurrent neural network system (Figure S1), where the parameters are shared across layers. The output of this algorithm is a collection of position specific features $h = [\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_L]$ as we mentioned in (1).

3.6 Extracting sequence level features

In this section we derive the sequence level feature $g(\cdot)$ in (1), which takes as input function h obtained from the last section. Earlier research discovered that weighted degree (WD) kernels with shifts and mismatches work best for binding affinity prediction by accounting for alternations in DNA subsequences (Wang *et al.*, 2014). We also take this into account when we design our sequence level features. A natural idea is to use a local sequence aggregation (pooling) first, then collect all these local pooled features to get the target embedding for the entire sequence. The max pooling operator will mimic the functionality of the shift operation in string kernels. By focusing on the maximum in each local context, the shifting of features in a small range will not affect the results.

Formally, suppose we have obtained the embedding $h = [\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_L]$, we perform pooling in each local context $\{\vec{\mu}_{i-1}, \vec{\mu}_i, \vec{\mu}_{i+1}\}$ (three adjacent positions). That is

$$g([\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_L]) = \sum_{i=1}^{L} \max\{\vec{\mu}_{i-1}, \vec{\mu}_i, \vec{\mu}_{i+1}\}, \tag{12}$$

where $\max\{\cdot\}$ finds maximum among its arguments. Note that $\max\{\cdot\}$ is performed dimension-wisely here on its arguments and returns a vector of the same dimension as $\vec{\mu}_i$. The boundary case is taken care of by setting $\vec{\mu}_0 = \vec{\mu}_{L+1} = \vec{\mathbf{0}}$.

Due to this sequence level feature aggregation operation, the final feature embedding f is in $\mathbb{R}^m$, which is independent of the input sequence length L. That is, our method can handle inputs of different lengths.

Algorithm 2 End-to-end parameter training

```
Input: Dataset $\mathcal{D} = \{(\chi^n, y^n)\}_{n=1}^N$.
Initialize parameters $\mathbf{U}^0 = \{v^0, \mathbf{W}^0\}$ randomly.
for $r = 1$ to $R$ do
    Sample $(\chi^r, y^r)\}$ uniform randomly from $\mathcal{D}$.
    Feed forward embedding computation:
    Compute feature embedding by Algorithm 1 with $\mathbf{W}^{r-1}$.
    $f(\chi^r) = g([\vec{\mu}_1^r, \ldots, \vec{\mu}_L^r]) = \sum_{i=1}^L \max\{\vec{\mu}_{i-1}, \vec{\mu}_i, \vec{\mu}_{i+1}\}$.
    Make prediction $\tilde{y}^r = \vec{v}^{r-1\top} f(\chi^r)$.
    Back propagation of gradients:
    Calculate loss $l = \frac{1}{2}(y^r - \tilde{y}^r)^2$.
    Calculate gradients $\frac{\partial l}{\partial v}$, $\frac{\partial l}{\partial \mathbf{W}}$ as in Section S1.
    Update $\vec{v}^r = \vec{v}^{r-1} - \eta_r \frac{\partial l}{\partial \vec{v}}$ where $\eta_r$ is the learning rate.
    Update $\mathbf{W}^r = \mathbf{W}^{r-1} - \eta_r \frac{\partial l}{\partial \mathbf{W}}$.
end for
Return $\mathbf{U}^R = \{v^R, \mathbf{W}^R\}$.
```

3.7 End-to-end discriminative training

In previous sections, we assumed that the parameters in the nonlinear feature embeddings are given. In this section, we will explain the learning algorithm for these parameters. We will learn them jointly with the regression model. That is, we will learn these models via mean square error minimization, $\min_{\mathbf{W},\vec{v}} l(f, \vec{v}) := \frac{1}{2}\sum_{n=1}^N \left(y^{(n)} - \vec{v}^\top f(\chi^{(n)})\right)^2$, where $\vec{v} \in \mathbb{R}^m$ are the parameters for the linear regression. We will use stochastic gradient descent to learn these parameters. The derivations of gradients with respect to parameters $\mathbf{W}$ and $\vec{v}$ are shown in Section S1. Though we described the case where the stochastic gradient is computed using a single pair of sequence and label, in practice we use a mini-batch of examples to reduce the variance of the stochastic gradients and to speed up training.

The overall training algorithm is summarized in Algorithm 2, which learns the parameter $\mathbf{W}$ in the feature embedding operators and the parameter $\vec{v}$ in the regression model. The algorithm shares similarities to the one for training recurrent neural networks (Jaeger, 2003). It is recurrent since the parameters involved in the message iterations are shared, which is also reflected in the gradient computation. See Section S2 for an illustration.

3.8 Interpreting the learned features

After we train the Sequence2Vec model, we can also obtain the important k-mers for interpretation. Since our sequence level features f can handle inputs of different lengths, we can enumerate all possible k-mers, and check the corresponding prediction values using the fully trained model. Specifically, for $\chi \in \Sigma^k$, the corresponding affinity prediction is given by $\vec{v}^\top f(\chi)$. For instance, if $k = 3$, we have 64 possible k-mers, and we obtain the pairs of the k-mer and its predicted score as $\{[\text{AAA}, \vec{v}^\top f(\text{AAA})], [\text{AAC}, \vec{v}^\top f(\text{AAC})], \ldots, [\text{TTT}, \vec{v}^\top f(\text{TTT})]\}$. For $k = 8$, the number of possible k-mers is 65,536, which is still easy to handle in our model. The k-mers that yield large binding affinity are the important ones predicted by the model. We use this criterion to rank all the k-mers.

4 Results and Discussion

4.1 Data sets

We first applied the proposed method to model the binding affinity landscape of Gcn4p in *Saccharomyces cerevisiae* based on the data set from HiTS-FLIP (Nutiu *et al.*, 2011). Gcn4p is a transcriptional activator of more than 30 amino acid biosynthetic genes (Natarajan *et al.*, 2001; Hinnebusch and Natarajan, 2002). Therefore, accurate modeling of the binding affinity landscape between Gcn4p and its promoter sites is crucial not only for elucidating the regulatory mechanisms involved in such stress-response pathways, but also for designing a synthetic Gcn4-induced response pathway in yeast (Hinnebusch, 2005).

The HiTS-FLIP data set contains the binding affinity values (K_d) of 83,252 12bp DNA sequences, where the binding affinity is defined as the concentration of the TF at which the DNA region is occupied 50% of the time at equilibrium. The K_d values in the HiTS-FLIP data set range from 8 nM to 1000 nM, where a small K_d represents a high binding affinity. In contrast to (Wang *et al.*, 2014) which tested their string kernel-based SVR model only on the subset with K_d less than 100 nM (1,391 12-mers), here we tested the performance of all methods on the entire HiTS-FLIP data set. The entire data set was randomly partitioned into 10 subsets to perform 10-fold cross-validation (CV). The reported results for all methods in this paper are the average over the same 10-fold CV.

We further tested the proposed method on 28 *Saccharomyces cerevisiae* TF data sets (Fordyce *et al.*, 2010). For each of the 28 TFs, the relative binding affinities to oligonucleotides covering all possible 8bp DNA sequences were measured by MITOMI 2.0, which is capable of constructing binding affinity landscape through the measurement of binding interactions at equilibrium by a microfluidic device. Each data set contains the relative binding affinities for nucleotide sequences with 52bp in length. After removing the sequences with "nan" (not a number) and taking average relative affinities for the same sequences, the number of 52bp sequences in each data set ranges from 1,084 to 1,456, with their corresponding relative binding affinities ranging from -0.27 to 0.99. It is worth noting that these 28 data sets are much more difficult for the computational methods to model than the aforementioned HiTS-FLIP data set, because the sequences in these data sets are much longer than those in the HiTS-FLIP data set (52bp v.s. 12bp), while the number of available samples is much smaller (1,084–1,456 v.s. 83,252). Again, each data set was randomly split into 10 folds, and the results reported for all methods are the average over the same 10-fold CV.

Finally, we evaluated the performance of the proposed method on protein-binding microarray (PBM) data from the revised DREAM5 TF-DNA Motif Recognition Challenge (Weirauch *et al.*, 2013). The PBM data represent 86 different mouse TFs, each measured using two independent array designs. Twenty of these TFs have their array intensity values given for both array types. Thus they were given to the participants for validating their models and were not used for the testing phase of the challenge. For the rest 66 TFs, participating teams were given the probe intensities for only one array design and had to make predictions on intensity values of the other array.

We chose these 66 DREAM5 TF data sets because they were used in the recently proposed DeepBind method (Alipanahi *et al.*, 2015) to evaluate their method. Since DeepBind was optimized and available on these data sets, we chose them to have a fair comparison between our method and DeepBind. We obtained the 66 TF data sets directly from the supplementary materials of Alipanahi *et al.* (2015). Following the same preprocessing step, we removed per-probe multiplicative bias from the training sets. Following the same training and testing procedure of DeepBind, for each TF Sequence2Vec was trained by cross-validation on one array and tested on the other array.

4.2 Compared methods

We compared the proposed method with seven other methods, including four state-of-the-art binding affinity prediction methods and three powerful machine learning methods. These include the PWM model, the recently proposed Bayesian Markov model (BaMM) (Siebert and Söding, 2016), the DREAM-winning HK→ME model (Annala *et al.*, 2011), the weighted degree kernel-based SVR model (Wang *et al.*, 2014), the multi-layer neural network (DNN), the convolutional neural network (CNN) (DeepBind when evaluated on the DREAM5 data sets), and the SVR model with the Fisher kernel (Jaakkola and Haussler, 1999).

For PWM, following Wang *et al.* (2014), we used the PWM model by solving the function $A \cdot x = K$ where A is an $n \times u$ matrix, where n is the number of training sequences, $u = 4L$ where L is the length of each training sequence, and K is the vector containing the binding affinity values for all the training sequences. $A[i, j]$ is set to 1 if the i-th sequence contains the specific nucleotide at the specific position indicated by the index j, otherwise 0. The x is a $4L$-dimensional column vector, which is the column concatenation of the PWM to be trained. Once the PWM is learned, for a query sequence, its binary vector representation is multiplied by the PWM to predict the binding affinity of this sequence.

For BaMM, since it was not designed for regression, we trained the generative model by varying the binding affinity threshold of defining positive/negative sequences (i.e., to convert our data sets with real-valued affinities to binary classification data sets). And for the same reason, the RMSE metric is not applicable to BaMM. We carefully tuned the parameters including the split threshold in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$, the degree of HMM in $\{2, 3, 4\}$ and the usage of inverse-complete training via cross validation.

For the HK→ME model, and the SVR models with the WD kernel and the Fisher kernel, in each of the 10-fold CV, we trained the model by fine tuning their parameters and used the best parameters on the training fold to make predictions on the test fold. We used HMM as the generative model for the Fisher kernel, where the number of hidden states $M \in \{4, 5, 6, 7, 8, 9\}$ is also fine tuned.

In addition, we implemented both DNN and CNN as a baseline for comparison. Both of the deep learning models were implemented using Keras (https://github.com/fchollet/keras). RMSprop optimizer (Tieleman and Hinton, 2012) was used for training with mean squared error as a loss function, and its parameters were left at their default values as recommended. The DNN architecture consists of 3 stacked ReLU layers with 512, 256 and 128 neurons, respectively. The CNN model follows the same architecture of DeepBind (Alipanahi *et al.*, 2015), which consists of a convolutional layer with 16 filters as shown in Figure 2. Note that for the data sets on which DeepBind provided their optimized models (*i.e.*, the DREAM5 data sets), we directly used the available DeepBind for comparison. For other data sets, we re-trained the CNN model with the same architecture as DeepBind and used it to evaluate the performance of DeepBind on these data sets. The filter length was searched within $\{14, 16, 24, 32\}$ by cross-validation and finally fixed to 24. Alternative values for the filter length were tested as well, but it did not bring much improvement. The convolutional layer is followed by a max pooling layer, and then a fully connected ReLU layer. Dropout was used as a regularizer to prevent overfitting in both models. We tuned the number of neurons for the fully connected layer $N \in \{32, 64, 128, 256, 512, 1024\}$ and dropout probability $P \in \{0.25, 0.50, 0.75\}$ by cross-validation. Batch size of 16 was chosen via cross-validation.

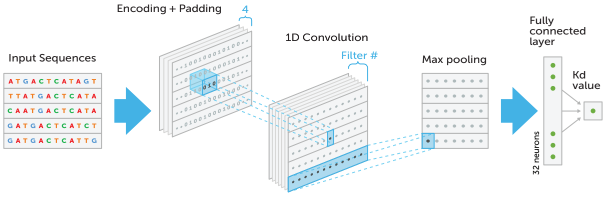


Fig. 2: The architecture of the baseline CNN model.

Table 1. Comparison of different methods on the HiTS-FLIP data set with 83,252 12bp DNA sequences (Nutiu et al., 2011). PWM: position weight matrix; BaMM: Bayesian Markov Model motif discovery (Siebert and Söding, 2016); LM: the DREAM-winning HK→ME linear model (Annala et al., 2011); SVR: the two round WD kernel-based SVR model (Wang et al., 2014); DNN: the multi-layer neural network model; CNN: the convolutional neural network model (Alipanahi et al., 2015); and S2V: the proposed Sequence2Vec method. The best performance under each measure is in bold.

Measure	PWM	BaMM	LM	SVR	DNN	CNN	S2V
RMSE	181.87	N/A	128.61	115.16	116.70	113.70	**108.70**
PCC	0.27	0.39	0.73	0.79	0.79	0.80	**0.82**
SCC	0.01	0.33	0.63	0.71	0.70	0.71	**0.75**

For our method, we tuned the number of message passing iterations to be $T \in \{2, 3, 4, 5\}$, the embedding dimension $d \in \{32, 64, 128\}$, and also learning parameters such as the batch size and the learning rate via cross-validation. For each held-out fold, all the parameters were tuned on the training set only, and the performance reported was the average over the 10 folds.

4.3 Performance measures

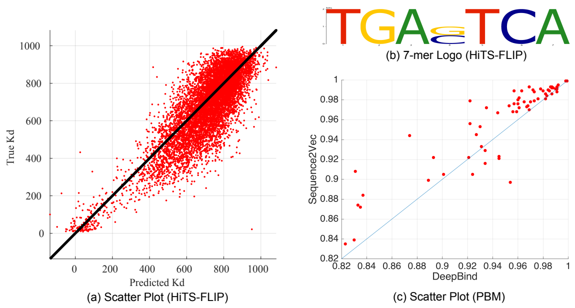
We measured performance using the root mean square error (RMSE), Pearson product-moment correlation coefficient (PCC), and Spearman's rank correlation coefficient (SCC), which are defined as in Section S3.

For the 66 TF data sets from DREAM5, we used the same evaluation criteria as DeepBind where they used area under the curve (AUC), computed by setting high-intensity probes as positives and the remaining probes as negatives, instead of RMSE. We used the code provided by DeepBind to calculate the AUC.

4.4 Comparison on predictive performance

The performance of the seven compared methods on the HiTS-FLIP data set is shown in Table 1. The Fisher kernel-based SVR model could not finish in one day of running on this data set and thus was not reported. The proposed Sequence2Vec method models this data set well (see Figure 3(a)) and outperforms the other methods under all the performance measures. Specifically, the RMSE, the PCC, and the SCC of Sequence2Vec improves over the second best method under each measure by 4.4%, 2.5%, and 5.6% respectively. The CNN model is the second best method, followed by the WD kernel-based SVR model and the DNN model. The PWM model, on the other hand, has surprisingly poor performance, which suggests that the assumption of each mononucleotide contributing independently to the binding affinity is not accurate.

We next experimented with the more comprehensive and difficult MITOMI 2.0 data sets for 28 TFs in *Saccharomyces cerevisiae* (Fordyce *et al.*, 2010), as shown in Tables 2 & S1. Compared to the HiTS-FLIP data set, these data sets contain much longer DNA binding sequences and



(a) Scatter Plot (HiTS-FLIP)

(b) 7-mer Logo (HiTS-FLIP)

(c) Scatter Plot (PBM)

Fig. 3: (a) The scatter plot between the true and predicted K_d values by Sequence2Vec on one fold of the HiTS-FLIP data set; (b) the 7-mer sequence logo made by the top-ranked 7-mers, predicted by Sequence2Vec, of the HiTS-FLIP data set; and (c) the scatter plot between the AUC of DeepBind and that of Sequence2Vec over the 66 PBM data sets.

Table 2. Comparison of the average performance of different methods over the MITOMI 2.0 data sets for 28 TFs in Saccharomyces cerevisiae (Fordyce et al., 2010). PWM: position weight matrix; BaMM: Bayesian Markov Model motif discovery (Siebert and Söding, 2016); LM: the DREAM-winning HK→ME linear model (Annala et al., 2011); WD: the two round weighted degree kernel-based SVR model (Wang et al., 2014); DNN: the multi-layer neural network model; CNN: the convolutional neural network model (Alipanahi et al., 2015); FS: the Fisher kernel-based SVR model (Jaakkola and Haussler, 1999); and S2V: the proposed Sequence2Vec model. The best performance under each measure is in bold.

Measure	PWM	BaMM	LM	SVR	DNN	CNN	FS	S2V
RMSE	0.049	N/A	0.080	0.042	0.044	0.039	0.043	**0.035**
PCC	0.06	0.24	0.26	0.41	0.16	0.45	0.34	**0.62**
SCC	0.07	0.23	0.11	0.23	0.13	0.20	0.26	**0.29**

much fewer samples. Note that RMSE in the two tables are not directly comparable because in the HiTS-FLIP data set, the target value is binding affinity K_d (in nM), whereas in the MITOMI data sets, the target values are relative binding affinities.

Overall, the Sequence2Vec model performed best among all the compared methods in terms of all the performance measures (Tables 2 & S1). Specifically, Sequence2Vec has the lowest RMSE on 19 of the 28 data sets. The average RMSE of Sequence2Vec reduces that of the second best method, the CNN model, by 10.3%. The advantage of Sequence2Vec becomes even more significant in terms of the two correlation coefficients. Sequence2Vec achieves the highest PCC on 26 out of the 28 data sets and the highest SCC on 20 out of the 28 data sets. The average PCC of Sequence2Vec is 37.8% higher than that of the second best method, the CNN model, whereas the average SCC of Sequence2Vec is 11.5% higher than that of the second best method, the Fisher kernel-based SVR model. Our results demonstrate that the proposed Sequence2Vec method performs well for both large data sets and long sequences.

Finally, we compared Sequence2Vec with DeepBind on the 66 TF data sets from the DREAM5 challenge. The performance of DeepBind was directly taken from (Alipanahi *et al.*, 2015) as their model was already optimized on these data sets. Overall, Sequence2Vec outperforms DeepBind in terms of PCC, SCC, and AUC (Table S2). Both methods can achieve a quite high average AUC which suggests that both perform well on these TF data sets. Detailed analysis reveals that among the 66 TFs, Sequence2Vec outperforms DeepBind on 46 TFs in terms of PCC, on 36 TFs in terms of SCC, and on 55 TFs in terms of AUC (Table S2 and Figure 3(c)).

4.5 Interpreting the learned features

It is known that 7-mer motifs are important for binding affinity of the DNA binding sequences for Gcn4p (Hill *et al.*, 1986; Sellers *et al.*, 1990). We thus measured the importance of all the 7-mers in our trained Sequence2Vec model. For this purpose, we fed each consecutive 7-mer motif ($4^7 = 16,384$ possibilities) as the input to our trained model because our model can accept different lengths of inputs. We then used the corresponding prediction values as the importance scores for these 7-mers.

Among all the 7-mer motifs, the top 5 motifs predicted by Sequence2Vec to have the best binding affinity are TGAGTCA, TGACTCA, TTAGTCA, TGACTAA, and GAGTCAT. Among them, TGAGTCA and TGACTCA are exactly the most well known 7-mer motifs for Gcn4p (Hill *et al.*, 1986; Sellers *et al.*, 1990), whereas TTAGTCA and TGACTAA are single nucleotide variants of this motif, which are also expected to be important. Interestingly, although our fifth ranked motif, GAGTCAT, is quite different from the known motif, it was actually reported as one of the most significant motifs for Gcn4p found from the deletion mutant microarray data (Chen *et al.*, 2004). The sequence logo made by the top-ranked 7-mers is shown in Figure 3(b). Sequence logos for 28 Saccharomyces cerevisiae TF data sets are shown in Section S6 and Table S3, which are in good agreement with the known motifs or their reverse complements (Fordyce *et al.*, 2010).

4.6 Convergence and computational efficiency

We plotted the convergence curves with respect to the training RMSE for Sequence2Vec on all the data sets (Section S7 and Tables S4-S10). It is clear that over 94 out of the 95 data sets evaluated, Sequence2Vec managed to converge before reaching the maximum number of iterations. In terms of the runtime for training and testing, Sequence2Vec is quite efficient (Section S8 and Table S11). For example, it took Sequence2Vec roughly 40 minutes to finish training on the HiTS-FLIP data set, whereas it only took it 4.66 seconds for testing, on a workstation with Intel Xeon CPU E5-1620 v2 @ 3.70GHz and 32G Memory.

4.7 Parameter sensitivity analysis

We conducted comprehensive experiments to analyze the sensitivity of the hyper-parameters on the performance of Sequence2Vec, including the range of dependencies our method encodes (controlled by the number of message passing rounds, T), the nonlinearity (controlled by the activation function, $\sigma(\cdot)$), the embedding size (m), and the batch size used during training. The overall conclusion is that Sequence2Vec is quite robust with respect to the hyper-parameters (Sections S9, and Tables S12 and S13). Generally speaking, when larger amount of training data are provided, the nonlinearity plays a more important role; and more rounds of message passing can often result in a higher accuracy. We further tested Sequence2Vec on synthetic data sets with implanted motifs, Sequence2Vec performed almost perfectly with respect to different levels of noise (Section S10 and Table S14).

5 Conclusion

In this paper, we proposed Sequence2Vec, a novel embedding approach for modeling the transcription factor binding affinity landscape. Different from the traditional kernel embedding methods which fix the embedding space beforehand, our method learns such embedding space discriminatively with the supervised information together. We demonstrate that by incorporating the sequence structure explicitly and utilizing the sequence location information, our method significantly outperforms alternative deep learning methods, as well as the state-of-the-art binding affinity prediction methods. Our method is expected to work well with a wide range of *in vivo* and *in vitro* data sets by providing a generic recipe to deal with many other structured data sets in computational biology, such as protein sequences, drug molecules, or even molecular dynamic trajectories. Essentially, we can model structured data as latent variable models and embed these models into nonlinear feature spaces. In our framework, both the feature embedding space and the discriminative model are learned jointly. Such a strategy leads to significant improvements in prediction which we believe will have an impact on a wide range of computational biology applications.

Acknowledgements

The research reported in this publication was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. URF/1/1976-04 and URF/1/3007-01. It was also supported in part by NSF IIS-1218749,NIH BIGDATA 1R01GM108341, NSF CAREER IIS-1350983, NSF IIS-1639792 EAGER, ONR N00014-15-1-2340, NVIDIA, Intel and Amazon AWS. This research made use of the resources of the computer clusters at KAUST.

References

Agius, P., Arvey, A., Chang, W., Noble, W. S., and Leslie, C. (2010). High resolution models of transcription factor-DNA affinities improve in vitro and in vivo binding predictions. *PLoS Comput Biol*, **6**(9).

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., and Walter, P. (2002). *Molecular biology of the cell*. Garland, 4 edition.

Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. (2015). Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nat Biotech*, **33**(8), 831–838.

Annala, M., Laurila, K., Lähdesmäki, H., and Nykter, M. (2011). A linear model for transcription factor binding affinity prediction in protein binding microarrays. *PLoS One*, **6**(5), e20059.

Barrera, L. A., Vedenko, A., Kurland, J. V., Rogers, J. M., Gisselbrecht, S. S., Rossin, E. J., Woodard, J., Mariani, L., Kock, K. H., Inukai, S., Siggers, T., Shokri, L., Gordân, R., Sahni, N., Cotsapas, C., Hao, T., Yi, S., Kellis, M., Daly, M. J., Vidal, M., Hill, D. E., and Bulyk, M. L. (2016). Survey of variation in human transcription factors reveals prevalent dna binding changes. *Science*, **351**(6280), 1450–1454.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inform. Theory*, **39**(3), 930–945.

Bengio, Y. (2009). Learning deep architectures for AI. *Found. Trends Mach. Learn.*, **2**(1), 1–127.

Berger, M. F. and Bulyk, M. L. (2006). Protein binding microarrays (PBMs) for rapid, high-throughput characterization of the sequence specificities of DNA binding proteins. *Methods Mol Biol*, **338**, 245–260.

Berger, M. F. and Bulyk, M. L. (2009). Universal protein-binding microarrays for the comprehensive characterization of the DNA-binding specificities of transcription factors. *Nat Protoc*, **4**(3), 393–411.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., and Smola, A. J. (2006). Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics (ISMB)*, **22**(14), e49–e57.

Bulyk, M. L., Johnson, P. L. F., and Church, G. M. (2002). Nucleotides of transcription factor binding sites exert interdependent effects on the binding affinities of transcription factors. *Nucleic Acids Res*, **30**(5), 1255–1261.

Chen, G., Hata, N., and Zhang, M. Q. (2004). Transcription factor binding element detection using functional clustering of mutant expression data. *Nucleic Acids Res*, **32**(8), 2362–2371.

Chen, X., Hughes, T. R., and Morris, Q. (2007). RankMotif++: a motif-search algorithm that accounts for relative ranks of *K*-mers in binding transcription factors. *Bioinformatics*, **23**(13), i72–i79.

Dai, H., Dai, B., and Song, L. (2016). Discriminative embeddings of latent variable models for structured data. *arXiv preprint arXiv:1603.05629*.

Davidson, E. H. (2006). *The regulatory genome: gene regulatory networks in development and evolution*. Elsevier Academic Press, Amsterdam.

Deplancke, B., Alpern, D., and Gardeux, V. (2016). The genetics of transcription factor DNA binding variation. *Cell*, **166**, 538–554.

Fan, M., Kuwahara, H., Wang, X., Wang, S., and Gao, X. (2015). Parameter estimation methods for gene circuit modeling from time-series mrna data: a comparative study. *Briefings in Bioinformatics*, **16**(6), 987–999.

Foat, B. C., Morozov, A. V., and Bussemaker, H. J. (2006). Statistical mechanical modeling of genome-wide transcription factor occupancy data by MatrixREDUCE. *Bioinformatics*, **22**(14), e141–e149.

Fordyce, P. M., Gerber, D., Tran, D., Zheng, J., Li, H., DeRisi, J. L., and Quake, S. R. (2010). *De novo* identification and biophysical characterization of transcription-factor binding sites with microfluidic affinity analysis. *Nat Biotechnol*, **28**(9), 970–975.

Gertz, J., Siggia, E. D., and Cohen, B. A. (2009). Analysis of combinatorial *cis*-regulation in synthetic and genomic promoters. *Nature*, **457**(7226), 215–218.

Gilbert, W. and Müller-Hill, B. (1967). The lac operator is DNA. *Proc Natl Acad Sci U S A*, **58**(6), 2415–2421.

Gonzalez, J., Low, Y., and Guestrin, C. (2009). Residual splash for optimally parallelizing belief propagation. In *In Artificial Intelligence and Statistics (AISTATS)*, Clearwater Beach, Florida.

Hassanzadeh, H. R. and Wang, M. D. (2016). Deeperbind: Enhancing prediction of sequence specificities of dna binding proteins. *arXiv preprint arXiv:1611.05777*.

Hill, D. E., Hope, I. A., Macke, J. P., and Struhl, K. (1986). Saturation mutagenesis of the yeast his3 regulatory site: requirements for transcriptional induction and for binding by GCN4 activator protein. *Science*, **234**(4775), 451–457.

Hinnebusch, A. G. (2005). Translational regulation of GCN4 and the general amino acid control of yeast. *Annu Rev Microbiol*, **59**, 407–450.

Hinnebusch, A. G. and Natarajan, K. (2002). Gcn4p, a master regulator of gene expression, is controlled at multiple levels by diverse signals of starvation and stress. *Eukaryot Cell*, **1**(1), 22–32.

Hochschild, A. and Ptashne, M. (1986). Cooperative binding of λ repressors to sites separated by integral turns of the DNA helix. *Cell*, **44**(5), 681–687.

Jaakkola, T. S. and Haussler, D. (1999). Exploiting generative models in discriminative classifiers. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 487–493. MIT Press.

Jaeger, H. (2003). *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach*.

Kuwahara, H., Fan, M., Wang, S., and Gao, X. (2013). A framework for scalable parameter estimation of gene circuit models using structural information. *Bioinformatics*, **29**(13), i98–i107.

Lee, D., Karchin, R., and Beer, M. A. (2011). Discriminative prediction of mammalian enhancers from dna sequence. *Genome Res*, **21**(12), 2167–2180.

Leslie, C., Eskin, E., and Noble, W. S. (2002). The spectrum kernel: A string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575, Singapore. World Scientific Publishing.

Leslie, C. S., Eskin, E., Cohen, A., Weston, J., and Noble, W. S. (2004). Mismatch string kernels for discriminative protein classification. *Bioinformatics*, **20**(4), 467–476.

Leung, M. K. K., Xiong, H. Y., Lee, L. J., and Frey, B. J. (2014). Deep learning of the tissue-regulated splicing code. *Bioinformatics*, **30**(12), i121–i129.

Levo, M., Zalckvar, E., Sharon, E., Dantas Machado, A. C., Kalma, Y., Lotam-Pompan, M., Weinberger, A., Yakhini, Z., Rohs, R., and Segal, E. (2015). Unraveling determinants of transcription factor binding outside the core binding site. *Genome research*, **25**, 1018–1029.

Liu, X. S., Brutlag, D. L., and Liu, J. S. (2002). An algorithm for finding protein-DNA binding sites with applications to chromatin-immunoprecipitation microarray experiments. *Nat Biotechnol*, **20**(8), 835–839.

Natarajan, K., Meyer, M. R., Jackson, B. M., Slade, D., Roberts, C., Hinnebusch, A. G., and Marton, M. J. (2001). Transcriptional profiling shows that Gcn4p is a master regulator of gene expression during amino acid starvation in yeast. *Mol Cell Biol*, **21**(13), 4347–4368.

Nutiu, R., Friedman, R. C., Luo, S., Khrebtukova, I., Silva, D., Li, R., Zhang, L., Schroth, G. P., and Burge, C. B. (2011). Direct measurement of DNA affinity landscapes on a high-throughput sequencing instrument. *Nat Biotechnol*, **29**(7), 659–664.

Pearl, J. (2001). *Causality: Models, Reasoning and Inference*. Cambridge University Press.

Ptashne, M. (1967). Specific binding of the λ phage repressor to λ DNA. *Nature*, **214**(5085), 232–234.

Ptashne, M. (2013). Epigenetics: core misconcept. *Proc Natl Acad Sci U S A*, **110**(18), 7101–7103.

Rätsch, G., Sonnenburg, S., and Schölkopf, B. (2005). RASE: recognition of alternatively spliced exons in *C.elegans*. *Bioinformatics*, **21 Suppl 1**, i369–i377.

Schmidhuber, J. (2014). Deep learning in neural networks: An overview. *Neural Networks, Vol*, **61**,, pp85–117,Jan2015.

Schölkopf, B., Tsuda, K., and Vert, J.-P. (2004). *Kernel Methods in Computational Biology*. MIT Press, Cambridge, MA.

Sellers, J. W., Vincent, A. C., and Struhl, K. (1990). Mutations that define the optimal half-site for binding yeast GCN4 activator protein and identify an ATF/CREB-like repressor that recognizes similar DNA sites. *Mol Cell Biol*, **10**(10), 5077–5086.

Siebert, M. and Söding, J. (2016). Bayesian Markov models consistently outperform PWMs at predicting motifs in nucleotide sequences. *Nucleic acids research*, **44**(13), 6055–6069.

Smola, A., Gretton, A., Song, L., and Schölkopf, B. (2007). A Hilbert space embedding for distributions. In *In Algorithmic Learning Theory: 18th International Conference*, pages 13–31. Springer-Verlag.

Song, L., Bedo, J., Borgwardt, K., Gretton, A., and Smola, A. (2007). Gene selection via the BAHSIC family of algorithms. *Bioinformatics (ISMB)*, **23**(13), i490–i498.

Sriperumbudur, B., Gretton, A., Fukumizu, K., Lanckriet, G., and Schölkopf, B. (2008). Injective Hilbert space embeddings of probability measures. In *Proc. Annual Conf. Computational Learning Theory*, pages 111–122.

Stormo, G. D. (2000). DNA binding sites: representation and discovery. *Bioinformatics*, **16**(1), 16–23.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5 RMSprop:divide the gradient by the running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Von Hippel, P. H. and Berg, O. G. (1986). On the specificity of DNA-protein interactions. *Proc Natl Acad Sci U S A*, **83**(6), 1608–1612.

Wang, X., Kuwahara, H., and Gao, X. (2014). Modeling DNA affinity landscape through two-round support vector regression with weighted degree kernels. *BMC Syst Biol*, **8 Suppl 5**, S5.

Weirauch, M. T., Cote, A., Norel, R., Annala, M., Zhao, Y., Riley, T. R., Saez-Rodriguez, J., Cokelaer, T., Vedenko, A., Talukder, S., , D. R. E. A. M. C., Bussemaker, H. J., Morris, Q. D., Bulyk, M. L., Stolovitzky, G., and Hughes, T. R. (2013). Evaluation of methods for modeling transcription factor sequence specificity. *Nat Biotechnol*, **31**(2), 126–134.

Wong, K.-C., Chan, T.-M., Peng, C., Li, Y., and Zhang, Z. (2013). Dna motif elucidation using belief propagation. *Nucleic acids research*, **41**(16), e153–e153.

Wong, K.-C., Li, Y., Peng, C., Moses, A. M., and Zhang, Z. (2015). Computational learning on specificity-determining residue-nucleotide interactions. *Nucleic acids research*, page gkv1134.

Wray, G. A. (2007). The evolutionary significance of *cis*-regulatory mutations. *Nat Rev Genet*, **8**(3), 206–216.

Zhou, J., Lu, Q., Xu, R., Gui, L., and Wang, H. (2016). Cnnsite: Prediction of dna-binding residues in proteins using convolutional neural network with sequence features. In *Bioinformatics and Biomedicine (BIBM), 2016 IEEE International Conference on*, pages 78–85. IEEE.