Gene expression

**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 and $^2$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

$^*${To whom correspondence should be addressed.}

The authors wish it to be known that, in their opinion, the first two authors should be regarded as Joint First Authors.

Associate Editor: Oliver Stegle

Received on March 20, 2017; revised on July 7, 2017; editorial decision on July 23, 2017; accepted on July 26, 2017

**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 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 hidden Markov models 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 datasets 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 and implementation:** Our program is freely available at https://github.com/ramzan1990/sequence2vec.

**Contact:** xin.gao@kaust.edu.sa or lsong@cc.gatech.edu.

**Supplementary information:** Supplementary data are available at Bioinformatics online.

**1 Introduction**

In 1967, seminal works in the phage $\lambda$ 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 (Gilbert and Müller-Hill, 1967; Ptashne, 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 (Prashne, 2013; Von Hippel and Berg, 1986).

A key determinant of the sequence specificity is affinity differences of all possible binding sites in the genome, in the in vitro 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 (Davidson, 2006), how organisms might have evolved their gene regulatory systems (Wray, 2007), and how artificial gene circuits can be designed and optimized (Fan et al., 2015; Gertz et al., 2009; Kuwahara et al., 2013). 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 (Bulyk et al., 2002; Hochschild and Prashne, 1986). 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 dataset 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 interaction profiling (HTS-FLIP) (Nutiu et al., 2011). Although the increase in the availability of high-throughput datasets from such methods has propelled the development of data-driven prediction methods (e.g. Agius et al., 2010; Annala et al., 2011; Barrera et al., 2016; Berger and Bulyk, 2009; Hassanzadeh and Wang, 2016; Lee et al., 2011; Wang et al., 2014; Wong et al., 2013, 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 (Deplancke et al., 2016; Levo et al., 2015).

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. Annala et al., 2011; Agius et al., 2010; Berger and Bulyk, 2009; Bulyk et al., 2002; Chen et al., 2007; Foat et al., 2006; Lee et al., 2011; Liu et al., 2002; Siebert and Söding, 2016; Stormo, 2000; Wang et al., 2014), 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 (Levo et al., 2015; Nutiu et al., 2011), 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) (Berger and Bulyk, 2009; Bulyk et al., 2002). 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, $HK \rightarrow ME$, 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 $h_{c, t} = 1$, if $k$-mer $t$ is found in sequence $s$, and 0 otherwise. The $k$-mer affinity contributions are obtained by solving a linear system $p = Hx + e$, where $p$ is a vector containing binding affinities of the training sequences, $x$ is a vector of $k$-mer affinity contributions, and $e$ 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, $x$ and $x'$, are first transformed into a common feature space, $\phi(x)$ and $\phi(x')$ respectively, and then their inner product is used to define the kernel, i.e. $k(x, x') = \langle \phi(x), \phi(x') \rangle$ (Leslie et al., 2002, 2004; Rak chains, 2008).

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; Rak chains, 2005) work best for binding affinity prediction by accounting for alternations in DNA subsequences. Their method showed significant improvements over PWM and HK $\rightarrow$ 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 beforehand. 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 shallower 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 these 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 Materials and 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 Fig. 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 DNA sequence is 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 dataset \( D = \{ \chi^{(n)}, y^{(n)} \}_{n=1}^{N} \), where each \( \chi^{(n)} \) is a DNA sequence and \( y^{(n)} \in R \) is the corresponding binding affinity value. The DNA sequence \( \chi^{(n)} = [x_1^{(n)}, x_2^{(n)}, \ldots, x_L^{(n)}] \) 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 dataset. 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 \( f: \Sigma^L \rightarrow \mathbb{R}^m \) 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 \( h: \Sigma^L \rightarrow \mathbb{R}^{d_L} \) and \( g: \mathbb{R}^{d_L} \rightarrow \mathbb{R}^m \), such that

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

Here \( \vec{\mu}_j^{(n)} \) captures features at position \( j \) 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,

\[
\min_{\vec{\mu}} L(f, \vec{\mu}) := \frac{1}{2} \sum_{n=1}^{N} \left( y^{(n)} - \vec{\mu}^T f(\chi^{(n)}) \right)^2,
\]

where \( \vec{\mu} \in \mathbb{R}^m \) 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 datasets.

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 will be referred to (Dai et al., 2016).

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

Given the graphical model parameters (or the potentials), the posterior marginals for the hidden variables, $p(H_t|x_t)$ for $t = 1, \ldots, L$, 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 (Bishop, 2006; Pearl, 2001) has been designed to perform this operation efficiently.

More specifically, let $m_{i,t}$ 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 $H_t$, to $H_{i+1}$, which are unnormalized distributions, for $i = 1, \ldots, L - 1$, as

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

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

Denote $q_i(H_i) = p(H_i|x_i)$. For $i = 2, \ldots, L - 1$, we have

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

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

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 (Borgwardt et al., 2006; Smola et al., 2007; Song et al., 2007),

$$\bar{\phi}_H := \mathbb{E}_H[\phi(H)] = \int_H \phi(H) q(H) dH, \quad (5)$$

where $\phi(H)$ is a generic nonlinear feature transformation. When the embedding is injective (Sriperumbudur et al., 2008), $\bar{\phi}_H$ 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 $p(H_t|x_t)$ by

$$\bar{\mu}_t = \int \phi(H) p(H_t|x_t) dH_t$$

and the messages $m_{ij}$ as

$$\bar{\nu}_{ij} = \int \phi(H) m_{ij}(H_i) dH_i.$$

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

$$\bar{\nu}_{i-1} = \bar{\mathcal{T}}_1 \circ (\bar{x}_i, \bar{\nu}_{i-1}), \quad (6)$$

$$\bar{\nu}_{i+1} = \bar{\mathcal{T}}_1 \circ (\bar{x}_i, \bar{\nu}_{i+1}), \quad (7)$$

$$\bar{\mu}_i = \bar{\mathcal{T}}_2 \circ (\bar{x}_i, \bar{\nu}_{i+1}, \bar{\nu}_{i+1}), \quad (8)$$

where $\bar{\mathcal{T}}_1$ and $\bar{\mathcal{T}}_2$ 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 $m_{n, i+1}$ and $m_{n, i-1}$, and the posterior $p(H_t|x_t)$ at position $i$. The nonlinear feature $\bar{\mu}_i$ represents position specific features which have already incorporated long-range dependencies using message passing.

3.4 Parameterizing the embedding operators

The operators $\bar{\mathcal{T}}_1$ and $\bar{\mathcal{T}}_2$ 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

1: Input: parameter $W$ in $\bar{T}_1$ and $\bar{T}_2$, and a sequence $x$.
2: Initialize $\bar{\mathcal{V}}^{(0)} = \bar{\mathcal{V}}$, for all $(i, j), |i - j| = 1, i, j = 1, \ldots, L$.
3: for $t = 1$ to $T$ do
4: for $i = 1$ to $L$ do
5: $\bar{\mathcal{V}}^{(t)}_{ij} = \mathcal{A}(W_{ij} x_i + W_{ij} W_{ij}^{(t-1)}(i, j), W_{ij}^{(t-1)}(i, j))$.
6: $\bar{\mathcal{V}}^{(t)}_{i-1} = \mathcal{A}(W_{i-1} x_i + W_{i-1}^{(t)} W_{i-1}^{(t-1)}(i, j), W_{i-1}^{(t-1)}(i, j))$.
7: end for
8: end for
9: for $i = 1, \ldots, L$ do
10: $\bar{\mu}_i = \mathcal{A}(W_{i} x_i + W_{i}^{(t)} W_{ij}^{(t)}(i, j) + W_{i}^{(t)} W_{ij}^{(t)}(i, j))$.
11: end for
12: Return $b = [\bar{\mu}_1, \bar{\mu}_2, \ldots, \bar{\mu}_L]$. 

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/33/22/3575/4042128 by guest on 05 March 2018
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 $\mu_k$ and $\nu_k$ into $R^d$. Then

$$
\overrightarrow{v}_{i+1} = \sigma(W_1 x_i + W_2 \overrightarrow{v}_{i-1}),
$$

(9)

$$
\overleftarrow{v}_{i-1} = \sigma(W_1 x_i + W_2 \overleftarrow{v}_{i+1}),
$$

(10)

$$
\overrightarrow{\mu}_i = \sigma(W_3 x_i + W_4 \overrightarrow{v}_{i+1} + W_5 \overleftarrow{v}_{i-1}),
$$

(11)

where $\sigma(\cdot)$ is a nonlinear activation function. For example, if we use rectified linear unit, then $\sigma(\cdot) := \max(0, \cdot)$, and $W = \{W_1, W_2, W_3, W_4\}$ is the collection of weights in the neural network, where $W_1, W_2 \in R^{d \times d}$ and $W_3, W_4 \in R^{d \times d}$. 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 $W = \{W_1, W_2, W_3, W_4\}$ in the embedding operators $\overrightarrow{T}$ and $\overleftarrow{T}$ (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 (Supplementary Fig. S1), where the parameters are shared across layers. The output of this algorithm is a collection of position specific features $b = [\overrightarrow{\mu}_1, \overrightarrow{\mu}_2, \ldots, \overrightarrow{\mu}_l]$ 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 $b$ 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 $b = [\overrightarrow{\mu}_1, \overrightarrow{\mu}_2, \ldots, \overrightarrow{\mu}_l]$, we perform pooling in each local context $[\overrightarrow{\mu}_{i-1}, \overrightarrow{\mu}_i, \overrightarrow{\mu}_{i+1}]$ (three adjacent positions), That is

$$
g(\{\overrightarrow{\mu}_1, \overrightarrow{\mu}_2, \ldots, \overrightarrow{\mu}_l\}) = \max \{\overrightarrow{\mu}_{i-1}, \overrightarrow{\mu}_i, \overrightarrow{\mu}_{i+1}\},
$$

(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 $\overrightarrow{\mu}_i$. The boundary case is taken care of by setting $\overrightarrow{\mu}_0 = \overrightarrow{\mu}_{l+1} = 0$.

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

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, $\min_W \mathbb{E}_f \mathbb{E}_v \mathbb{E}_y (f, \overrightarrow{v}) := \frac{1}{N} \sum_{n=1}^{N} \left(y^{(n)} - f(\overrightarrow{v}^{(n)})\right)^2$, where $\overrightarrow{v} \in R^m$ 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 $W$ and $\overrightarrow{v}$ are shown in Supplementary 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 $W$ in the feature embedding operators and the parameter $\overrightarrow{v}$ 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 Supplementary 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 $x \in \Sigma^L$, the corresponding affinity prediction is given by $f(\overrightarrow{v}(x))$. 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 $\{(\text{AAA}, f(\overrightarrow{v}(\text{AAA}))], [\text{AAC}, f(\overrightarrow{v}(\text{AAC})), \ldots, [\text{TTT}, f(\overrightarrow{v}(\text{TTT}))\}$. 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 Datasets

We first applied the proposed method to model the binding affinity landscape of Gcn4p in Saccharomyces cerevisiae based on the dataset from HiTS-FLIP (Nutiu et al., 2011). Gcn4p is a transcriptional activator of more than 30 amino acid biosynthetic genes (Hinnebusch and Natarajan, 2002; Natarajan et al., 2001). 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 dataset contains the binding affinity values ($K_d$) of 83252 12 bp 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 dataset range from 8 to 1000 nM, where a small $K_d$
represents a high binding affinity. In contrast to (Wang et al., 2014), we chose to evaluate their method. Since DeepBind was optimized and available on these datasets, we chose them to have a fair comparison between our method and DeepBind. We obtained the 66 TF datasets directly from the supplementary materials of Alipanahi et al. (2015). Following the same pre-processing 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 Soding, 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 datasets) 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 BAAM, 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 datasets with real-valued affinities to binary classification datasets). 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/keras-team/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 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 datasets on which DeepBind provided their optimized models (i.e. the DREAM5 datasets), we directly used the available DeepBind for comparison. For other datasets, we re-trained the CNN model with the same architecture as DeepBind and used it to evaluate the performance of DeepBind on these datasets. The filter length was searched within

---

**Algorithm 2** End-to-end parameter training

Input: Dataset $\mathcal{D} = \{(x_i^j, y_i^j)\}_{i=1}^N$.

Initialize parameters $U_0 = \{u^L, W_0\}$ randomly.

for $r = 1$ to $R$ do

Sample $(x', y')$ uniformly randomly from $\mathcal{D}$.

Feed forward embedding computation:

Compute feature embedding by Algorithm 1 with $W^{-1}$.

$f(x') = g(\{\bar{u}_1, \ldots, \bar{u}_L\}) = \max_{i=1}^L \{\bar{u}_{i-1}, \bar{u}_i, \bar{u}_{i+1}\}$.

Make prediction $y' = \bar{W}^{-1} f(x')$.

Back propagation of gradients:

Calculate loss $l = \frac{1}{2} (y' - y)^2$.

Calculate gradients $\frac{\partial l}{\partial \mathbf{W}}$ as in Supplementary Section S1.

Update $\mathbf{W} = \mathbf{W}^{-1} - \eta \frac{\partial l}{\partial \mathbf{W}}$ where $\eta$ is the learning rate.

Update $U = U^{-1} - \eta \frac{\partial l}{\partial \mathbf{U}}$.

end for

Return $U^*_R = \{i^*, W^*_R\}$.
each measure is in bold.

Table 1. Comparison of different methods on the HiTS-FLIP dataset with 83,252 12 bp DNA sequences (Nuti et al., 2011)

<table>
<thead>
<tr>
<th>Measure</th>
<th>PWM</th>
<th>BaMM</th>
<th>LM</th>
<th>SVR</th>
<th>DNN</th>
<th>CNN</th>
<th>S2V</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE</td>
<td>181.87</td>
<td>N/A</td>
<td>128.61</td>
<td>115.16</td>
<td>116.70</td>
<td>113.70</td>
<td>108.70</td>
</tr>
<tr>
<td>PCC</td>
<td>0.27</td>
<td>0.39</td>
<td>0.73</td>
<td>0.79</td>
<td>0.79</td>
<td>0.80</td>
<td>0.82</td>
</tr>
<tr>
<td>SCC</td>
<td>0.01</td>
<td>0.33</td>
<td>0.63</td>
<td>0.71</td>
<td>0.70</td>
<td>0.71</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Note: PWM, position weight matrix; BaMM, Bayesian Markov Model motif discovery (Siebert and Soding, 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); S2V, the proposed Sequence2Vec method. The best performance under each measure is in bold.

\{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 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 \(P \in \{0.25, 0.50, 0.75\}\) via cross-validation. Batch size of 16 was chosen via cross-validation.

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 Supplementary Section S3.

For the 66 TF datasets 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 dataset is shown in Table 1. The Fisher kernel-based SVR model could not finish in one day of running on this dataset and thus was not reported. The proposed Sequence2Vec method models this dataset well (see Fig. 3a) and outperforms the other methods under all the performance measures. Specifically, the RMSE, the PCC and the SCC of Sequence2Vec improve 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 datasets for 28 TFs in Saccharomyces cerevisiae (Fordyce et al., 2010), as shown in Table 2 and Supplementary Table S1. Compared to the HiTS-FLIP dataset, these datasets contain much longer DNA binding sequences and much fewer samples. Note that RMSE in the two tables are not directly comparable because in the HiTS-FLIP dataset, the target value is binding affinity \(K_d\) (in nM), whereas in the MITOMI datasets, 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 (Table 2 and Supplementary Table S1). Specifically, Sequence2Vec has the lowest RMSE on 19 of the 28 datasets. 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 datasets and the highest SCC on 20 out of the 28 datasets. 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 datasets and long sequences.

Finally, we compared Sequence2Vec with DeepBind on the 66 TF datasets from the DREAM5 challenge. The performance of DeepBind was directly taken from (Alipanahi et al., 2015) as their model was already optimized on these datasets. Overall, Sequence2Vec outperforms DeepBind in terms of PCC, SCC and AUC (and Supplementary Table S2). Both methods can achieve a quite high average AUC which suggests that both perform well on these TF datasets. 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 (and Supplementary Table S2 and Fig. 3c).

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 ($4^7 = 16384$ 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 five motifs predicted by Sequence2Vec to have the best binding affinity are TGAGTCA, TGACTCA, TTAGTCA, TGACTAA and GAGTCAT. Among them, TGAGTCA and TGACTAA 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 datasets are shown in Supplementary Section S6 and Supplementary 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 datasets (Supplementary Section S7 and Supplementary Tables S4–S10). It is clear that over 94 out of the 95 datasets 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 (Supplementary Section S8 and Supplementary Table S11). For example, it took Sequence2Vec roughly 40 min to finish training on the HiTS-FLIP dataset, whereas it only took it 4.66 s for testing, on a workstation with Intel Xeon CPU E5-1620 v2 @ 3.70 GHz and 32 G 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 (Supplementary Section S9 and Supplementary 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 datasets with implanted motifs, Sequence2Vec performed almost perfectly with respect to different levels of noise (Supplementary Section S10 and Supplementary 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 datasets by providing a generic recipe to deal with many other structured datasets 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.

Funding

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.

Conflict of Interest: none declared.

References


Sequence2Vec: an embedding approach for binding affinity prediction

3583


Wang, X. et al. (2014) Modeling DNA affinity landscape through two-round support vector regression with weighted degree kernels. BMC Syst. Biol., 8, SS.


