

Supplementary Material for DISCOVER: A feature-based discriminative method for motif search in complex genomes

A1. Formal definitions of some features

Sequence Conservation

The PWM offers a simple and straightforward way to formally model a TFBS specific to a single TF, and to score the DNA segments according to its likelihood of being a true motif or not based only on the sequence content within the segment in question in Naughton *et al* [*Nuc Acids Res*, 2006] takes a slightly different direction and models a motif type as a bag of instances, using a graph cut method to identify motifs in a set of n-grams. We use the standard PWM definition. The feature captures the degree of conservation of a potential motif binding site i given the position weight matrix of the motif, $\theta^{(m)}$.

The feature function is defined as:

$$\begin{aligned} f_{SC}^{(m)}(y_i, \mathbf{x}) &= f_{SC}^{(m)}(y_i, x_{i:i+l^{(m)}-1}) \\ &= \delta(y_i, M^{(m)}) \sum_{j=1}^{l^{(m)}} \beta(\theta_j^{(m)}, x_{i+j-1}) \end{aligned} \quad (5)$$

$$\beta(\theta_j^{(m)}, k) = \log \theta_{jk}^{(m)} - \log \theta_{0k}; \quad (6)$$

where $\theta^{(m)} = \{\theta_{jk}^{(m)} : j = 1, \dots, l^{(m)}, k \in \{A, C, G, T\}\}$ is the PWM of motif type m , $l^{(m)}$ is the length of the motif, and $\theta_0 = \{\theta_{0k} : k \in \{A, C, G, T\}\}$ is the nucleotide frequency in background. The δ function equals 1 when y_i is assigned to state $M^{(m)}$ and 0 otherwise.

State Transition

The State transition feature captures the relationship between neighboring states. The feature function is defined as:

$$f_T^{s_1 \rightarrow s_2}(y_i, y_{i+1}, \mathbf{x}) = \delta(y_i, s_1) \delta(y_{i+1}, s_2) \quad (7)$$

where $s_1, s_2 \in \mathbf{S}$. $\delta(y_i, s_1)$ ¹ equals 1 when y_i is s_1 , and 0 otherwise. $\delta(y_{i+1}, s_2)$ likewise. Thus, the feature function equals 1 only when y_i is state s_1 and y_{i+1} is state s_2 , and 0 otherwise. There are $(2 + N_M)^2$ features of this category in total. State transition features are an effort to model the architecture of the regulatory region.

GC-Content

A high percentage of nucleotide *guanine* (G) and *cytosine* (C) may indicate a region containing regulatory elements. The feature function is defined as:

$$f_{GC}(y_i, \mathbf{x}) = \delta(y_i, M) \left(p(x_{i-w/2:i+w/2}) - p_0 \right) \quad (8)$$

$$p(x_{left:right}) = \frac{1}{right - left + 1} \sum_{i=left}^{right} \left(\delta(x_i, G) + \delta(x_i, C) \right) \quad (9)$$

¹ δ function can be viewed as an Identity function.

where w is the window size, p is the GC-percentage inside the window whose value lies in $[0,1]$, and p_0 is the average GC-percentage over the dataset. The $\delta(y_i, M)$ function equals 1 when y_i is assigned to any motif state and 0 otherwise.

As an example, the sum of conservation symmetry features can be computed as:

$$F_{CS}(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^L f_{CS}(y_i, \mathbf{x}) \quad (10)$$

where f_{CS} is defined in Eq 13 and L is the length of the sequence. $F_{CS}(\mathbf{y}, \mathbf{x})$ is one of the elements in function vector $\mathbf{F}(\mathbf{y}, \mathbf{x})$ used in a CRF model in Eq 1.

Reverse Complementarity

This feature assesses how likely a potential binding site i is reverse complementary with itself. In other words, that is how similar the site is to its counterpart on the other genomic strand. The higher similarity may suggest a true motif. The feature function is defined as:

$$f_{RC}(y_i, \mathbf{x}) = \sum_m \delta(y_i, M^{(m)}) \left(s(x_{i:i+l(m)-1}) - s_0 \right) \quad (11)$$

$$s(x_{i:i+l-1}) = \frac{1}{\lfloor l/2 \rfloor} \sum_{j=1}^{\lfloor l/2 \rfloor} \delta_{pair}(x_{i+j-1}, x_{i+l-j}) \quad (12)$$

where s is the reverse complementary score of a potential binding site whose value lies in $[0,1]$, s_0 is an offset value that is set at the mean, and l is the length of the motif. The $\delta(y_i, M^{(m)})$ function equals 1 when y_i is the state of motif type m and 0 otherwise. The $\delta_{pair}(a, b)$ function equals 1 if and only if a and b are a Watson-Crick pair.

Conservation Symmetry

This feature captures the symmetry of the degree of sequence conservation given motif PWM within a motif binding site with respect to the center. The feature is defined as:

$$f_{CS}(y_i, \mathbf{x}) = \sum_m \delta(y_i, M^{(m)}) \left(cs(\theta^{(m)}, x_{i:i+l(m)-1}) - cs_0 \right) \quad (13)$$

$$cs(\theta^{(m)}, x_{i:i+l(m)-1}) = \frac{1}{\lfloor l^{(m)}/2 \rfloor} \sum_{j=1}^{\lfloor l^{(m)}/2 \rfloor} \left| \beta(\theta_j^{(m)}, x_{i+j-1}) - \beta(\theta_{l^{(m)}+1-j}^{(m)}, x_{i+l(m)-j}) \right| \quad (14)$$

where cs averages the conservation symmetry score over a potential binding site, cs_0 is an offset value of choice, $l^{(m)}$ is the length of the motif, and β function is the conservation score of a single base defined in Eq 6.

Melting Temperature

This feature provides an estimated melting temperature of sequences within a certain size of window by a formula:

$$f_{MT}(y_{i:i+w-1}, \mathbf{x}) = 64.9 + \frac{41 * (G + C - 16.4)}{A + T + G + C} \quad (15)$$

where w is the window size, and A, T, G and C are the counts of the four types of nucleotides within the window. We set the window size to 15, which is about the length of a long TFBS.

Distance to Transcription Start Site

Sites closer to a transcription start site are more likely to be TFBSs, so we adopt this feature to assess how close each site is to a nearest transcription start site. It is easy to understand that a distance change from 0-bp to 1k-bp makes more difference than a distance change from 10k-bp to 11k-bp though both of them are shifted by 1k-bp, so the feature score should not be linear on distance. We apply a logarithm function and a small constant to avoid logarithm going to negative infinity. The feature scores are calculated as:

$$f(z) = \log(z + 5) \quad (16)$$

where z is the distance in base-pair.

A2. Model Parameters

Feature weights constitute the set of model parameters. Some of them can be fixed and the others are free. More free parameters make the CRF model more complex, which might be harder to learn. As a guide line, we want to avoid redundant free parameters, since they will not make any contribution. On the other hand, parameters that are not likely to be properly learned from training data should never be included, because including them will only increase the chance of over-fitting. In this part, our main focus is on the weight of state transition features, because they account for a large portion in the whole parameter set.

In the CRF model, we assign a parameter as a weight to each of the features defined in the previous subsection. Those are the vector λ in Eq 1. However, some of them are not free parameters because of the context. In state transition, it is not allowed to reach an M state directly from a G state, since it is enforced that state M's representing TFBSs are surrounded by state C representing *cis*-Regulatory Module region. Thus, the corresponding state transition features have a weight being *-inf*, which means that the transitions will never happen in the CRF model. In practice, we set the weights to a small enough number.

For the sake of a good performance, we want to have a reasonable number of free model parameters. More free parameters will promote the expressing ability of the model, but at the same time the hardness of model learning will increase, the running time of learning algorithm will rise, and some parameters may be overfitting due to the lack of data describing the related features. In our case, the state transitions from a motif state to a motif state are rare, if they ever happened, which will make those transition features an inevitable overfit if we set them free. Our solution is banning the transition between motif states and setting the matching weights to *-inf*. As a result, the number of all possible state transitions reduces dramatically.

A close look at the remaining set of state transitions will reveal redundancy. Assuming that no CRM region is on the edge, the sequence of hidden states will start with a global background state and end with a global background state. In that case, the number of transition from state G to state C will be exactly the same as the number of transition from state C to state G along the sequence of states. The models are identical to each other as long as the sum of the weight of transition feature G-C and the weight of transition feature C-G is a constant, given all the other parameters unchanged. Only one of the two weights need be a free parameter, leaving the other one to be fixed at any finite value. For simplicity, we set the weight of C-G

to zero. Similar situations happen to the pair of state transition C-M^(m) and M^(m)-C, so we fix the weight of M^(m)-C at zero.

The free parameters of state transition features left so far are G-G, C-C, G-C and C-M's. The number of state transitions along the sequence is unchanging given the sequence, so there is one more degree of redundancy, a common offset within the weights of state transition features. We get rid of the common offset by fixing the weight of G-G at zero. The final free parameters of state transition features are those of C-C, G-C and C-M's.

For those free parameters, it is not a good idea to let them be totally free. A prior can be imposed on each of them, as a way to encode prior knowledge on them. This may help in the attempt to avoid over-fitting issues. For example, we can make a prior be a normal distribution of mean 0 and variance σ^2 .

A3. Model training

In this section, we briefly describe the model training procedure in which feature weights of the CRF model are learned from training data. A more thorough exposition is presented in Supplementary Materials. Firstly, a learning criterion is set up, which can be either to maximize likelihood or to maximize posterior probability. Then, it is turned into a convex optimization problem, and finally a Quasi-Newton method is applied.

Our goal in the model learning task is to learn the best setting for λ , the weights of features in the CRF model. What we have are a group of sequences as training data with their nucleotide types \mathbf{x} and state labels \mathbf{y} , so the value of feature functions \mathbf{f} can be computed given necessary hyper-parameters.

A criterion is needed to learn the feature weights λ from nucleotide types \mathbf{x} and state labels \mathbf{y} , or more precisely from feature values \mathbf{f} . In the CRF model, a reasonable criterion is to maximize the likelihood of λ with respect to \mathbf{y} conditioned on \mathbf{x} , which equals the probability of state labels \mathbf{y} given feature weights λ conditioned on nucleotide types \mathbf{x} , because the probability model itself is defined in this conditional scheme. The max likelihood estimator of λ can be expressed as:

$$\hat{\lambda} = \arg \max_{\lambda} L(\lambda | \mathbf{y}, \mathbf{x})$$

$$\text{where } L(\lambda | \mathbf{y}, \mathbf{x}) = P(\mathbf{y} | \mathbf{x}, \lambda)$$

For the simplicity of notation, we just showed likelihood function in a one-training-sequence circumstance. When multiple (for example, m) training sequences are used, as we do in our experiment, the likelihood function will be:

$$L(\lambda | \mathbf{y}, \mathbf{x}) = P(\mathbf{y} | \mathbf{x}, \lambda) = \prod_{k=1}^m P(\mathbf{y}^{(k)} | \mathbf{x}^{(k)}, \lambda)$$

where $\mathbf{x}^{(k)}$ and $\mathbf{y}^{(k)}$ represent the vector of nucleotide types and a vector of state labels of the k -th sequence, respectively.

Getting the maximum point of a likelihood function is equivalent to getting the maximum point of a log-likelihood function $L_{\lambda} = \log L(\lambda | \mathbf{y}, \mathbf{x})$, since logarithm function is monotonically increase.

$$L_{\lambda} = \sum_{k=1}^m \left[\lambda \cdot \mathbf{F}(\mathbf{y}^{(k)}, \mathbf{x}^{(k)}) - \log Z(\mathbf{x}^{(k)}, \lambda) \right]$$

We can prove that the function of L_{λ} is concave with respect to λ (see supplementary material), so it turns into a typical convex optimization problem to find the maximum point [Boyd and Vandenberghe, 2004]. Gradient method or Newton's method can be adopted, and convergence is assured in theory. Both of them

are iterative methods which first get a search direction and then find a proper step length in each iteration. The update scheme is:

$$\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}^{(n)} + t\Delta\boldsymbol{\lambda}$$

where n is the iteration round, $\Delta\boldsymbol{\lambda}$ is the search direction, and t is the step length. The search direction is set to the negative of the first derivative of log-likelihood function $-\nabla L_\lambda$ in Gradient method, and $-\nabla L_\lambda / \nabla^2 L_\lambda$ in Newton's method. The step length is determined by a Back-track Search method (see supplementary material). The initial point $\boldsymbol{\lambda}^{(0)}$ can be picked by experience.

It can be shown that the first derivative of log-likelihood function with respect to $\boldsymbol{\lambda}$ is:

$$\nabla\boldsymbol{\lambda} = \sum_{k=1}^m \left\{ \mathbf{F}(\mathbf{y}^{(k)}, \mathbf{x}^{(k)}) - \mathbb{E}[\mathbf{F}(\mathbf{y}, \mathbf{x}^{(k)}) | \mathbf{x}^{(k)}, \boldsymbol{\lambda}] \right\}$$

The derivative is tractable, because the conditional expectation of feature sums $\mathbf{F}(\mathbf{y}, \mathbf{x}^{(k)})$ given genomic sequence $\mathbf{x}^{(k)}$ and feature weights $\boldsymbol{\lambda}$ is computational feasible (see supplementary material).

In practice, however, gradient method is likely to converge slowly, and the second derivative term in Newton's method is hard to compute efficiently. A Quasi-Newton method [Avriel, 2003] is more practical, in which an approximation is applied to the inverse of the second derivative of log-likelihood with respect to feature weights $\boldsymbol{\lambda}$ and the rest parts are the same as Newton's method. More specifically we use BFGS approximation method (see supplementary material).

Besides choosing the likelihood of $\boldsymbol{\lambda}$ as the target function to maximize, we can instead use the posterior probability:

$$P(\boldsymbol{\lambda} | \mathbf{y}, \mathbf{x}) = \frac{P(\boldsymbol{\lambda}, \mathbf{y}, \mathbf{x})}{P(\mathbf{y}, \mathbf{x})} = \frac{P(\mathbf{y} | \mathbf{x}, \boldsymbol{\lambda}) P(\mathbf{x} | \boldsymbol{\lambda}) P(\boldsymbol{\lambda})}{P(\mathbf{y}, \mathbf{x})}$$

As long as feature weights are independent of genomic sequences, $P(\mathbf{x} | \boldsymbol{\lambda}) = P(\mathbf{x})$, which is constant. So,

$$P(\boldsymbol{\lambda} | \mathbf{y}, \mathbf{x}) \propto P(\mathbf{y} | \mathbf{x}, \boldsymbol{\lambda}) P(\boldsymbol{\lambda})$$

The full version of posterior probability for multiple (m) training sequences is:

$$P(\boldsymbol{\lambda} | \mathbf{y}, \mathbf{x}) \propto P(\boldsymbol{\lambda}) \prod_{k=1}^m P(\mathbf{y}^{(k)} | \mathbf{x}^{(k)}, \boldsymbol{\lambda})$$

assuming state labels of different sequences $\mathbf{y}^{(k)}$ are independent of each other given $\mathbf{x}^{(k)}$ and $\mathbf{y}^{(k)}$ is independent of $\mathbf{x}^{(j)}$ given $\mathbf{x}^{(k)}$ when $j \neq k$.

The new target function is concave, as long as the prior distribution function of $\boldsymbol{\lambda}$ is log-concave. We keep using L_λ to represent the logarithm of posterior probability. As an example, the full version for multiple (m) training sequences is:

$$L_\lambda = \sum_{k=1}^m \left[\boldsymbol{\lambda} \cdot \mathbf{F}(\mathbf{y}^{(k)}, \mathbf{x}^{(k)}) - \log Z(\mathbf{x}^{(k)}, \boldsymbol{\lambda}) \right] - \frac{\boldsymbol{\lambda} \cdot \boldsymbol{\lambda}}{2\sigma^2} + C$$

if each λ follows a $\mathcal{N}(0, \sigma^2)$ as a prior. C is a constant in the equation. The equation has a similar form to a regularized log-likelihood.

A4. Evaluation metrics

We first compare the predictions from the inference step with ground truth labels to obtain counts of true positive (TP), false positive (FP) and false negative (FN) prediction instances. Predictions within a fixed 3bp tolerance window of an actual ground truth instance are taken to be TP. We sum up the TP, FP and FN counts over all sequences and calculate the overall precision and recall values from the overall TP, FP and FN counts using the definitions $\text{Precision} = \text{TP}/(\text{TP} + \text{FP})$ and $\text{Recall} = \text{TP}/(\text{TP} + \text{FN})$. Finally, $\text{F1-score} = 2/(1/\text{Precision} + 1/\text{Recall})$. An alternative method is calculating the precision and recall values on individual sequences first and averaging them second, but this gives equivalent weights to each sequence and unequal importance to each TFBS, and hence we do not pursue this latter method.

A5. Scope of the Model

Our approach takes advantage of the CRF models, which can overcome label bias problems that often happen in HMM models. The CRF model is a discriminative method that is based on a set of feature designs. The flexible forms of feature designs make it possible and easier to encode current knowledge in the field as well as to incorporate new information on TFBS when they are available. For example, we have made use of the knowledge about *cis*-Regulatory Module architecture as well as the abundance level of *guanine* and *cytosine* in nearby region in our predictor for TFBS. A feature weight, a parameter in the CRF model, determines the degree to which the feature influences the probability model. Priors can be put on the parameters, as long as they do not break the concavity of the target function. The concavity (or convexity) is such a good characteristic that we no longer need to worry about the annoying local maximum (or minimum) issues in iterative methods, and convergence is guaranteed theoretically. As expected, our method outperforms window-based methods and HMM-based methods in the experiment.

The CRF model also allow us to put together more than necessary features, because the feature weights that we got from the learning step will decide whether they are in use or not in the final model. However, as for now, the limited data size we got may prevent us from learning out the actual value of some under-represented features, and may result in severe over-fitting if we introduce too many features at a time. On the other hand, the iterative methods in the learning step may have a higher difficulty in convergence as more and more free feature parameters are added into the model, because an approximation is being used. Sometimes, singularity may occur in the approximation to the Hessian matrix². In such case, we used the identity matrix to replace it, which is the same as its initial setting. The analysis and improving of convergence speed regarding various free parameter set could be a future work.

As for now, our feature functions are limited to containing only neighboring hidden states. More variety of features, such as long distance features between two hidden states that are away from each other and features involving more than two hidden states, are desired when trying to encode some knowledge. For example, we will need long distance features to encode motif co-occurrence, some other kind to directly describe motif spacing and CRM length, etc. However, complex feature functions could make the algorithms currently used in the learning step invalid, therefore alternative algorithms need be studied. There is a (hidden) trade-off between the express power of feature functions and the efficiency of learning. This will be one of the future directions to work on.

It is noticeable that an offset is presented in Eq (8) (11) (13), which tries to move the mean value of a feature to 0. The motivation is trying to minimize the impact of adding/removing the feature to other weights. It is helpful in practice.

A special prediction scheme, rank decoding, is used in the paper. We control the number of positive predictions made rather than a common threshold for probability values. This can strike a good balance

²The second derivative matrix of target function, log-likelihood or log-posterior-probability, with respect to the variable vector, λ .

between sequences, because longer sequences tend to fit into a model worse when it is different from the (unknown) real model. On the other hand, this scheme is reasonable in the sense of working load when we want to verify the predictions in biology experiments. Sequence decoding, another prediction scheme, does not work at most time, which barely output positive predictions, because of the modeling error accumulated along the long sequence. MAP decoding may sometimes work well.