Supplementary Methods for Reconstructing dynamic regulatory maps

1 Probabilistic Model

1.1 Model Overview

The model DREM uses to integrate time series expression data with static ChIP-chip or motif data is based on a specific instance of an Input-Output Hidden Markov Model (IOHMM) (Bengio & Frasconi, 1995) which we will denote as M (also see Figure 1). Formally M is a tuple $(H, E, \Psi, \Theta, n, \gamma)$ where

- n is a parameter for the number of discrete time points that M will be modeling.
- H is a set of hidden states. Each hidden state, h, is associated with a Gaussian output distribution, f_h. Each hidden state $h \in H$ is associated with one time point, denoted $h.t$, where $t \in \{0, 1, ..., n-1\}$.
- Θ is the set of parameters for the output distributions. For each hidden state $h \in H$ there is an element $(\mu_h, \sigma_h) \in \Theta$ corresponding to the mean and standard deviation of the Gaussian distribution f_h.
- E contains the set of directed edges connecting hidden states of H, corresponding to valid transitions among hidden states. If and only if $(h_a, h_b) \in E$ then there is an edge from hidden state h_a to hidden state h_b. The set of edges, E, connecting the hidden states, H, are constrained to enforce a tree structure. Each hidden state is constrained to have at most γ children. Formally we assume there is exactly one state $h \in H$ with $h.t = 0$ which forms the root of a tree. For any state $a \in H$ with $a.t < n-1$ there must be at least one hidden state $b \in H$ with $(a, b) \in E$ satisfying $a.t + 1 = b.t$ and no more than γ such b. For any hidden state $b \in H$ with $b.t > 0$ there must be exactly one state a such that $(a, b) \in E$, additionally the relation $a.t + 1 = b.t$ must be satisfied for that state. If paths are allowed to be merged as a post-processing step discussed below, then this last requirement is relaxed so that there may be more than one such state a.
- Ψ contains the parameters controlling transition probabilities between hidden states. If a state $h \in H$ has two or more children, that is there are

$a, b \in H$ such that $(h, a) \in E$ and $(h, b) \in E$ and $a \neq b$, then there is an element $\psi_h \in \Psi$. ψ_h is a vector of parameters for a logistic regression classifier.

A logistic regression classifier (Krishnapuram et al, 2005) associated with state h is used to map a static input vector for a gene g, I_g, to transition probabilities among its children states. For this paper the j^{th} element of I_g is 1 if transcription factor j is predicted to regulated gene g and 0 otherwise. In general this prediction can be based on ChIP-chip data, motif data, or a combination thereof. We consider here the case $\gamma = 2$, which is the setting of γ used in this paper. If the gene g at time $i-1$ is in a state h that has child states a and b, then the probability it is in state a that at time point i can be written as

$$\frac{1}{1 + e^{-\psi_{h.INT} - \sum_x \psi_h.x \times I_g.x}}$$

where the sum is over each element, $I_g.x$, of I_g and $\psi_h.x$ is the corresponding element of ψ. $\psi_{h.INT}$ is the intercept parameter of the logistic regression classifier.

As an example suppose our static input vector is based on four transcription factors GCN4, CBF1, FHL1, and SFP1. Suppose the static input vector for gene g is

$$(I_{g.GCN4}, I_{g.CBF1}, I_{g.FHL1}, I_{g.SFP1}) = (1, 1, 0, 0).$$

Further suppose that

$$\psi_h = (\psi_{h.GCN4}, \psi_{h.CBF1}, \psi_{h.FHL1}, \psi_{h.SFP1}, \psi_{h.INT}) = (0.925, 0.607, -1.806, -2.018, 0.676)$$

then for a gene g in state h the probability of transitioning to state a is

$$\frac{1}{1 + e^{-\psi_{h.INT} - \sum_x I_g.x \times \psi_h.x}} = \frac{1}{1 + e^{-0.676 - 0.925 \times 1 - 0.607 \times 1 + 1.806 \times 0 + 2.018 \times 0}} = 0.901$$

In contrast if $I_g = (0, 0, 1, 1)$ then the probability of the gene transitioning to state a is 0.041. If γ is greater than 2, then the logistic regression classifier naturally generalizes to the multi-class case (Krishnapuram et al, 2005).

We note that requiring each hidden state to be associated with exactly one time point and the constraints on transitions among the hidden states is specific to our application and not a general property of an IOHMM. Also in a general IOHMM the output distribution can be dependant on the input vector and the input vector can be different at each time point both of which are not the case in this application.

1.2 Likelihood Function

Let $o_g = (o_g(1), ..., o_g(n-1))$ be the log ratio expression values for gene g at time points 1 to $n-1$ relative to a time point 0 control. Define H_i as the hidden state variable at time t. Denote the transition probability for a gene g to transition to state h_b at time point t given that it is in hidden state h_a at time point t as $P(H_t = h_b | H_{t-1} = h_a, I_g)$. This probability is defined as 0 if h_b is not a child of h_a and 1 if h_b is the only child of h_a. If h_a has two or more children then the transitions are probabilistic and depend on the static input vector I_g. The mapping from static input vector I_g to transition probabilities for a hidden state h_a are determined by a trained logistic regression classifier (Krishnapuram et al, 2005). The likelihood density, r, for a set of genes G for the model M is

$$r(G|M) = \sum_{g \in G} \log \sum_{q \in Q} \prod_{t=1}^{n-1} f_{q(t)}(o_g(t)) \prod_{t=1}^{n-1} P(H_t = q(t) | H_{t-1} = q(t-1), I_g)$$

Q is the set of all paths of hidden states of length n starting from the root with non-zero probability. For a path $q \in Q$, $q(i)$ is the hidden state of the path at time point i. The first product is the product of the output densities for the expression values and a given sequence of hidden states. If an expression value is missing then its corresponding output density term is omitted from the product. The second product is the product of transition probabilities for a given sequence of hidden states. The inner sum is over all paths with non-zero probability. The outer sum is over all genes in G.

2 Model Learning

2.1 Parameter Learning

For a given tree structure determined by H and E finding the setting of the parameters that globally optimizes the equation above is a non-linear optimization problem and in general cannot be guaranteed to be found. However a local maximum can be found using a modification of the Baum-Welch training algorithm for a regular Hidden Markov Model (Durbin et al, 1998). For our model above during each maximization step of the Baum-Welch algorithm each logistic regression classifier is retrained. The retraining algorithm was based on the method of (Krishnapuram et al, 2005), which uses an L_1 penalty on the coefficients to promote sparsity on the number of features used by the classifier. The regularization parameter associated with the L_1 penalty was set to one. When training a classifier, for every gene in the training set the classifier is given a weighted example of the gene transition to each child state. The weight of the example is the probability of that gene going through that hidden state based on the current values of all the parameters in the model.

2.2 Structure Learning

To find a structure for the model, as determined by (H,E), a search starts from a single chain of hidden states. The algorithm then performs a search over various structures. To train and test these structures, the algorithm splits the set of genes into two sets, G_{train} and G_{test}. G_{train} contains 75% of the genes and these genes are used for training the parameters of structures under consideration. The remaining 25% of genes, G_{test}, are used for scoring the structures. The parameter training tries to find settings of Ψ and Θ that maximize $r(G_{train}|M)$ while the score for a model, M, is $r(G_{test}|M)$. The search considers adding and deleting paths to the structure while $r(G_{test}|M)$ increases. The search algorithm is summarized in the pseudocode below. After no more paths can be added or deleted from the model without improving the test set score, the algorithm removes weakly supported paths that might be overfitting G_{test}. To do this the algorithm randomly resplits the set of genes used for training and testing, generating a new test set $test'$. The algorithm then deletes any path if the score for the retrained model with the path deleted, M_{new}, compared to old model, M_{old}, satisfies

$$(1+\kappa)r(G_{test'}|M_{new}) - r(G_{test'}|M_{old}) \geq \zeta$$

where $\kappa \geq 0$ and generally small and $\zeta \leq 0$. Here we set κ to 0.0015 and $\zeta = 0$, increasing κ or decreasing ζ would have the effect of removing more of the least supported splits. If multiple paths satisfy the above property, then the path resulting in the largest score is selected first. Once a path has been selected M_{new} becomes M_{old} and more paths can be deleted if they satisfy the above property. A similar procedure is used to delay splits for which the split could have been placed one time point later with no significant effect on the score.

Optionally the algorithm can also merge paths which share a common split. An example of two paths being merged is shown in Section 10 of the Supplementary results. The criteria to accept a model with two paths merged is the same as above except possibly with different choices of the parameters. To demonstrate a merging in the DTT example we set for the merging criteria κ to 0.0025 while ζ was still 0. If the algorithm decides to merge two paths, then the merged path can then be merged with another path in which it now shares a split. Once a path is merged it is not allowed to split again.

The algorithm then combines the train and test sets and retrains the model on the combined set without changing the structure. Genes are then assigned to their most likely path using the Viterbi algorithm (Durbin et al, 1998). Paths which have less than five genes assigned are removed. The splits and paths are then scored for association with transcription factors (next subsection). It is important to note that the map built will depend on the random training and test splits. Ideally thus the algorithm should be run several times with different random seeds generating different train and test splits of the data.

2.3 Pseudocode for Model Learning

```
Model Learning Algorithm
1. Partition gene set into a train set and test set.
2. Initialize the tree structure (H,E) to be a single chain, then train chain,
   and compute test score.
```

```
3. While test score improves do
    a.  (H',E') <- (H,E)
    b.  For each hidden state, h, that can have another child do
           i. Temporarily add a single chain of hidden states from h to (H',E')
           ii. Train the temporary model from step 3.b.i
           iii. If test set score is best found so far, then
                let (H,E) be the model structure from step 3.b.i
    c.  (H',E') <- (H,E)
    d.  For each hidden state, h, in H', which has a sibling in H'
           i.  Temporarily remove h and all descendants from (H',E')
           ii. Train the temporary model from step 3.d.i
           iii.  If the test set score is at least as good as the best so far,
                 then let (H,E) be the model structure from step 3.d.i
        If (H,E) was updated during 3.d.iii, then repeat step 3d
4.  Randomly resplit train and test data.
5.  Delete weakly supported paths, delay appropriate splits,
    and optionally merge appropriate paths.
6.  Train parameters of model using all genes.
7.  Assign genes to paths using the viterbi algorithm.
8.  Remove any path with fewer than 5 genes.
```

3 Transcription Factor Scoring

Consider a transcription factor f, a split S, and path A out of a split. Let n_S be the total number of genes assigned to the path into the split. Let n_A be the total number of genes on path A out of the split. Let c_S be the total number of genes into the spilt predicted to be regulated by transcription factor f, and let c_A be the number of these genes on path A. The score for transcription factor f for split S on path A is

$$\sum_{i=c_A}^{\min(c_S,n_A)} \frac{\binom{c_S}{i}\binom{n_S-c_S}{n_A-i}}{\binom{n_S}{n_A}}$$

The lower the score the stronger the association of a transcription factor with a path out of a split. This score is computed using the hypergeometric distribution however it does not represent true p-values since the transcription factor information was used to learn the model and assign genes to paths.

Overall enrichment for a transcription factor f along path A can be computed similarly, which takes into account enrichment based on filtering and prior splits. Let M be the total number of genes before filtering and c_M be the total number of genes predicted to be regulated by transcription factor f. The overall score for transcription factor f on path A is

$$\sum_{i=c_A}^{\min(c_M,n_A)} \frac{\binom{c_M}{i}\binom{M-c_M}{n_A-i}}{\binom{M}{n_A}}$$

4 References

1. Bengio Y, Frasconi, P. (1995) An Input Output HMM Architecture. *Advances in Neural Information Processing* **7**, 427-434.

2. Krishnapuram B, Figueiredo M, Carin L, and Hartemink A. (2005) Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Trans Pattern Anal Mach Intell.* **27**, 957-968 .

3. Durbin R, Eddy S, Krogh A, Mitchison G. (1998) Biological Sequence Analysis. Cambridge University Press, Cambridge.