Supplementary Materials

Contents

1 Supplementary Methods

Notation. Let $\mathbf{X}$ denote an $N \times C$ matrix of cell type-specific expression values for N genes in C cell types, where each column $\mathbf{X}(:, c)$ is the measurement of all genes in cell type $c \in \{1, \cdots, C\}$, and $\mathbf{X}(g, c)$ is the expression level of gene g in cell type c. $\mathcal{Y} = \{\mathbf{Y}_1, \cdots, \mathbf{Y}_C\}$ denotes collection of feature matrices, one for each cell type, c. Each $\mathbf{Y}_c$ is an $N \times F$ matrix, with the g^{th} row, $\mathbf{Y}_c(g, :)$ specifying the values of F features for gene g. The values inside each feature matrix, $\mathbf{Y}_c$ can be either context-independent (*e.g.*, a sequence-based motif network) or context-specific (*e.g.*, a motif network informed by accessibility value, histone modifications). Let τ denote the lineage tree which describes how the C cell types are related and K denote the number of gene modules. For each cell type c, $\mathbf{R}_c =< G_c, \Theta_c >$, denotes the RMN for c. G_c is a bipartite graph specifying the edges between F features and K modules, and Θ_c, are the parameters of the regulator programs for each module that relate the selected regulatory features to the expression of genes in the module. Let $\mathbf{M}$ be a $N \times C$ matrix denoting the module assignments of the N genes across all cell types, with $\mathbf{M}(g, c)$ denoting the module assignment of gene g in cell type c. In DRMN, we additionally have transition probability distributions $\mathbf{\Pi} = \{\Pi_1, ..., \Pi_C\}$ (**Figure 1**), which capture the dynamics of the module assignments in one cell type c given its parent cell type, pa(c) in the lineage tree τ. Specifically, $\Pi_c(i, j)$ is the probability of any gene being in module j given that its parental assignment is to module i. For the root cell type, this is simply a prior probability over modules. We first define the RMN model for one cell type and extend it to multiple related cell types for DRMN.

RMN. In RMN each cell type's data is modeled independently and we optimize the posterior probability of each cell type c's model as: $P(\mathbf{R}_c|\mathbf{X}_c, \mathbf{Y}_c)$, where $\mathbf{X}(:, c) = \mathbf{X}_c$ which by Bayes rule is proportional to $P(\mathbf{X}_c|\mathbf{Y}_c, \mathbf{R}_c)P(\mathbf{R}_c)$. For the last term, we assume $P(\mathbf{R}_c|\mathbf{Y}_c) = P(\mathbf{R}_c)$ and denotes a prior on the structure of the model. The data likelihood for each cell type, c is $P(\mathbf{X}_c|\mathbf{R}_c, \mathbf{Y}_c)$, and is obtained by marginalizing out the hidden module variables, $\mathbf{M}(:, c)$: $\sum_{\mathbf{M}(:,c)} P(\mathbf{X}(:, c), \mathbf{M}(:, c)|\mathbf{R}_c, \mathbf{Y}_c)$. This will further decompose for each gene's expression $\mathbf{X}(c, g)$ as $\prod_g \sum_{\mathbf{M}(g,c)} P(\mathbf{X}(g, c)|\mathbf{M}(g, c), \mathbf{R}_c, \mathbf{Y}_c(g, :))P(\mathbf{M}(g, c)|\mathbf{R}_c)$, where $\mathbf{M}(g, c)$ is the module assignment for gene g in cell type c. Thus the RMN model specifies each gene's expression to be generated from a mixture model where $P(\mathbf{M}(c, g)|\mathbf{R}_c) = P(\mathbf{M}(c, g))$ is the prior probability of each module specified by $\mathbf{M}(c, g)$. Letting, $\mathbf{X}(g, c) = X_{cg}$ and $\mathbf{M}(g, c) = M_{cg}$, we rewrite

the probability of each gene as $P(X_{cg}|\mathbf{R}_c, M_{cg}, \mathbf{Y}_c(g,:))$. Let $\mathbf{Y}_c^{(i)}$ denote the subset of regulatory features associated module i and cell type c and F_{ci} denote the number of columns of $\mathbf{Y}_c^{(i)}$. For $M_{cg} = i$, the expression of a gene, X_{cg} and features $\mathbf{Y}_c^{(i)}$ are modeled jointly as a $F_{ci}+1$ dimensional Gaussian. The expression of a gene given the features is a conditional mean derived as a linear combination of the feature values with parameters θ_{ci} and conditional variance $\Sigma_{x|\mathbf{Y}_{cg}^{(i)}}$. Given, θ_{ci} and $\Sigma_{x|\mathbf{Y}_{cg}^{(i)}}$, the probability of $\mathbf{X}_c(g)$ from module i in cell type c given its features $\mathbf{Y}_{cg}^{(i)}$ is:

$$P(X_{cg}|M_{gc}, \mathbf{R}_c, \mathbf{Y}_c^{(i)}(g,:)) \sim \mathcal{N}\left(\sum_{f\in\{0\cdots F_{ci}\}} \theta_{ci}(f)\mathbf{Y}_c^{(i)}(g,f), \Sigma_{x|\mathbf{Y}_{cg}^{(i)}}\right) \tag{1}$$

Here, $f = 0$ corresponds to the bias term.

DRMN. The DRMN model is defined by a set of RMNs, $\mathbf{R} = \{\mathbf{R}_1, \cdots, \mathbf{R}_C\}$ linked via the lineage tree τ, and the posterior likelihood of the model given the data, $P(\mathbf{R}_1, \cdots, \mathbf{R}_C|\mathbf{X}_1, \cdots, \mathbf{X}_C, \mathbf{Y_1}, \cdots, \mathbf{Y}_C, \tau)$. Based on Bayes rule, this can be rewritten as $P(\mathbf{X}_1, \cdots, \mathbf{X}_C, |\mathbf{R}_1, \cdots, \mathbf{R}_C, \mathbf{Y}_1, \cdots, \mathbf{Y}_C)P(\mathbf{R}_1, \cdots, \mathbf{R}_C)$. As in the case of RMN, we introduce the module variables and decompose this over a gene, but now across all the cell types together:

$$\prod_g P(X_{1g}, \cdots, X_{Cg}|M_{1g}, \cdots, M_{Cg}, \mathbf{R}, \mathcal{Y})P(M_{1g}, \cdots, M_{Cg}|\mathbf{R}, \tau)P(\mathbf{R}|\tau) \tag{2}$$

Given the module assignments, the gene expression across cell types are independent of each other. Hence the probability of each gene expression is:

$$\left(\prod_c P(X_{cg}|M_{cg}, \mathbf{R}_c, \mathbf{Y}_c(g,:))\right) P(M_{1g}, \cdots, M_{Cg}|\tau)P(\mathbf{R}|\tau)$$

Due to the tree structure, we assume the module assignment in a current cell type c, M_{cg} is independent of everything else, given its parent $\text{pa}(c) = c'$ in τ. Thus, $P(M_{1g}, \cdots, M_{Cg}|\tau)$ is rewritten as $P(M_{1g})\prod_{c'\to c\in\tau} P(M_{cg}|M_{c'g})$, where 1 denotes the root of the tree and $c' \to c \in \tau$ denotes a parent child

relationship in τ. Combining the two, the probability of a gene's expression across cell types is:

$$P(X_{1g}|M_{1g}, \mathbf{R}_1, \mathbf{Y}_1(g,:))P(M_{1g}) \left(\prod_{c \to c' \in \tau} P(X_{cg}|M_{cg}, \mathbf{R}_c, \mathbf{Y}_c(g,:))P(M_{cg}|M_{c'g}) \right) P(\mathbf{R}|\tau) \quad (3)$$

$P(X_{cg}|M_{cg}, \mathbf{R}_c, \mathbf{Y}_c(g,:))$ is computed in the same way as **Eqn 1**. $P(M_{cg}|M_{c'g})$ is obtained from the transition probabilities. $P(\mathbf{R}|\tau)$, specifies a prior over the structure/feature sets in each of the cell type-specific RMN models, $\mathbf{R}_c$. The prior controls for each module, what sets of features get associated with each module i, and how they change over time. We used two formulations for the prior to enable sharing information: DRMN-Structure Prior (DRMN-ST) defines a structure prior over the graph structures $P(G_1, ..., G_C)$ while DRMN-FUSED uses a regularized regression framework and implicitly defines priors on the $P(\Theta_1, \cdots, \Theta_C)$. In both frameworks, we share information between the cell types/conditions to learn the regulatory programs of each cell type/condition. The procedure is specific to DRMN-ST and DRMN-FUSED.

1.1 DRMN-ST: Structure prior approach.

In DRMN-ST, information is shared by specifying a structure prior, $P(G_1, ..., G_C)$, defined only over the graphs. The parameters are set to their maximum likelihood setting. $P(G_1, ..., G_C)$ determines how information is shared between different cell types at the level of the network structure and encourages similarity of features, indicative of regulators, between cell types. $P(G_1, ..., G_C)$ is computed using the transition matrices $\{\Pi_1, ..., \Pi_C\}$ and decomposes over individual regulator-module edges within each cell type as follows:

$$P(G_1, \cdots, G_C) = \prod_{f \to k} P(\mathbf{I}_{f \to k}) \quad (4)$$

where

$$P(\mathbf{I}_{f \to k}) = P(I^{root}_{f \to k}) \prod_{c' \to c \in \Psi} P(I^{c}_{f \to k}|I^{c'}_{f \to k}) \quad (5)$$

where $I^c_{f\to i}$ is an indicator function for the presence of the edge $f \to i$ for cell type c, between a regulator f and a module i. To define $P(I^c_{f\to i}|I^{c'}_{f\to i})$, we use the transition probability of the modules as:

$$P(I^c_{f\to i}|I^{c'}_{f\to i}) = \begin{cases} \Pi_c(i|i) & \text{if } I^c_{f\to i} = I^{c'}_{f\to i} \\ \frac{1-\Pi_c(i|i)}{K-1} & \text{otherwise,} \end{cases}$$

where K is the number of modules. Here the first option gives the probability of maintaining the same state from parent to child cell types (is present or absent in both c and c'), and the second option gives the probability of changing the edge state. The data likelihood is written as

$$P(\mathbf{X}^{(k)}_c|\mathbf{R}_{c,k}, \mathbf{Y}^{(k)}_c) \tag{6}$$

where $\mathbf{X}^{(k)}_c$ is gene expression vector of module k in cell line c, $\mathbf{Y}^{(k)}_c$ is the feature matrix of module k in cell line c, and $\mathbf{R}_{c,k} =< G_{c,k}, \Theta_{c,k} >$ is the regulatory program of module k consisting of $G_{c,k}$ the inferred interactions and $\Theta_{c,k}$ the corresponding regression coefficients. The data likelihood can be estimated as a conditional normal distribution

$$P(\mathbf{X}^{(k)}_c|\mathbf{R}_{c,k}, \mathbf{Y}^{(k)}_c) \sim \prod_{g\in \text{module}k} \mathcal{N}\left(\sum_{f\in G_{c,k}} \theta_{c,k}(f)\mathbf{Y}^{(k)}_c(g,f), \Sigma_{x|G_{c,k}}\right) \tag{7}$$

and $\Sigma_{x|G_{c,k}}$ denotes conditional variance. We incorporated the structure prior within a greedy hill climbing algorithm for estimating the DRMN model. **Algorithm 1** outlines the greedy hill climbing algorithm that is called in each iteration of EM algorithm (see **Algorithm 6**) for each module k. The parameter $maxIter$ is set to 5, meaning that at most 5 new regulators can be added to the regulatory program of a module in each iteration of the EM algorithm.

Algorithm 1: DRMN-ST algorithm

Input:
- $\mathbf{X}_c^{(k)}$ Gene expression vector for module k in cell line c
- $\mathbf{Y}_c^{(k)}$ Feature matrix for module k in cell line c
- Ψ Lineage tree

Output:
- $\mathbf{R}_{c,k} =< G_{c,k}, \Theta_{c,k} >$ The updated regulatory program

for $maxIter$ *number of iterations* **do**
  for *each feature* f **do**
    for *each cell line* c **do**
      $scoreImprovement \leftarrow$ improvement of data likelihood when adding f to regulatory program of k
      if $scoreImprovement > 0$ **then**
        $I^c_{f \to k} \leftarrow 1$
      end
      else
        $I^c_{f \to k} \leftarrow 0$
      end
    end
    Calculate $P(G_1, \cdots, G_C)$ ▷ Prior term, see **equations 4** and **5**
    Calculate $P(\mathbf{X}_c^{(k)} | \mathbf{R}_{c,k}, \mathbf{Y}_c^{(k)})$ ▷ Data likelihood term, see **equations 6** and **7**
  end
  Add f^*, feature with highest overall score improvement to the model.
end

1.2 DRMN-FUSED: Parameter prior approach.

In DRMN-FUSED, we use a fused group LASSO formulation to share information between the cell types for each module k. We assume that genes are already assigned to a module and optimize the following objective:

$$\min_{\Theta} \sum_{c} ||\mathbf{X}_c^{(k)} - \mathbf{Y}_c^{(k)}\theta_{ck}^T||_2^2 + \rho_1||\Theta_k||_1 + \rho_2||\Psi\Theta_k||_1 + \rho_3||\Theta_k||_{2,1}, \quad (8)$$

where $\mathbf{X}_c^{(k)}$ is the expression vector of genes in module k in cell line c, $\mathbf{Y}_c^{(k)}$ is the feature matrix for genes in module k, and θ_{ck} is a $1 \times F$ vector of regression coefficients for the same module and cell line (non-zero values correspond to selected features). θ_{ck} is analogous to the parameters for the conditional mean for each gene in **Eqn 1**. Θ_k is the C by F matrix resulting from stacking up the θ_{ck} vectors as rows. Ψ is a $C-1$ by C matrix, encoding the lineage tree. Each row of Ψ corresponds to a branch of the tree and each column corresponds to a cell type. If row i of Ψ corresponds to a branch $c' \rightarrow c$ in the lineage tree, $\Psi(i,c') = 1$, $\Psi(i,c) = -1$, and all other values in that row are 0. $\Psi\Theta_k$ is a $C-1$ by F matrix, where row i correspond to $\Theta_{c',k} - \Theta_{c,k}$, the difference between regression coefficients of cell lines c' and c. $||.||_1$ denotes l_1-norm (sum of absolute values), $||.||_2$ denotes l_2-norm (square root of sum of square of value), and $||.||_{2,1}$ denotes $l_{2,1}$-norm (sum of l_2-norm of columns of the given matrix). ρ_1, ρ_2 and ρ_3 correspond to hyper parameters, with ρ_1 for sparsity penalty, ρ_2 to encourage similarity between selected features of consecutive cell lines in the lineage tree, and ρ_3 to encourage selecting the same features for all cell types. Thus, ρ_2 controls the extent to which more closely related cell types are closer in their regression weights, while ρ_3 controls the extent to which all the cell types share similarity in their regulatory programs. We set these hyper parameters based on cross-validation by performing a grid search for ρ_1, ρ_2 and ρ_3 as described in the dataset-specific application sections. We implemented the optimization algorithm by extending the algorithm described in the MALSAR MATLAB package (Zhou et al. 2012) to handle branching topologies as described below. The learning algorithm uses an accelerated gradient method (Nesterov 2005; Nesterov 2007) to minimize the objective function above.

The FUSED LASSO objective is not smooth and therefore requires special handling, that is they have a smooth (l_2 loss) and a non-smooth part (Fused LASSO part). We follow the implementation in the MALSAR

package (Zhou et al. 2012), which makes use of Nesterov's accelerated gradient method (AGM) for composite functions (Nesterov 2005; Nesterov 2007). The MALSAR package uses the efficient fused LASSO algorithm which internally makes use of the Fused LASSO Signal Approximator (FLSA) algorithm (Liu et al. 2010). The FLSA algorithm is an iterative algorithm that makes use of a Sub-Gradient Finding Algorithm (SFA). However, the original implementation of this algorithm in the MALSAR MATLAB package is suitable only for linear time-series data. We extend this algorithm to handle general branching structure by re-implementing the Subgradient Finding Algorithm with gradient descent (**Algorithm 5**, SFA_G).

Algorithm 2 is the main algorithm for the AGM method. The parameters α and γ, control the step size of AGM. The algorithm has two nested loops. The outer loop updates the step size parameters executing the main AGM framework. The inner loop uses the fused LASSO penalty to estimate new regression weights (**Algorithm 3**), which internally calls **Algorithm 4** (FLSA) and **Algorithm 5** (SFA_G) and updates the model parameters due to the non-smooth part.

Algorithm 2: DRMN-FUSED algorithm

Input:

- $\mathbf{X}_c^{(k)}$ Gene expression vector for module k in cell line c
- $\mathbf{Y}_c^{(k)}$ Feature matrix for module k in cell line c
- ρ_1, ρ_2, ρ_3
- Ψ Lineage tree

Output:

- W the resulting regression weights ($C \times F$, number of tasks by number of features)

$$W = \begin{pmatrix} W_1 \\ W_2 \\ \vdots \\ W_C \end{pmatrix}$$

Initialze:

Initialize regression weights for each task, $W_c = 0$, $W_c^{old} = 0$

$t = 1, t_{\text{old}} = 0$, $\gamma = 1, \gamma_{\text{inc}} = 2$, ▷ Rate parameters for AGM

$F_{old} = 0$

while *not converged* **do**

- $\alpha = \frac{t_{\text{old}} - 1}{t}$
- $W_c^{new} = (1 + \alpha) W_c - \alpha W_c^{old}$
- $\nabla W_c^{new} = (\mathbf{Y}_c^{(k)} W_c^{\intercal} - \mathbf{X}_c^{(k)})^{\intercal} \mathbf{Y}_c^{(k)}$ ▷ Compute gradient for the squared loss for all c
- $F_{new} = \sum_c \frac{1}{2} ||\mathbf{X}_c^{(k)} - \mathbf{Y}_c^{(k)} W_c^{new\intercal}||_2^2$
- **while** *True* **do**
  - $V_c = W_c^{new} - \nabla W_c^{new} / \gamma$ ▷ Get initial estimate of regression weight for all c
  - $W^{proj} = \text{getFGLASSO}(V, \frac{\rho_1}{\gamma}, \frac{\rho_2}{\gamma}, \frac{\rho_3}{\gamma}, \Psi)$ ▷ Fused Group LASSO projection (**Algorithm 3**)
  - $F_{proj} = \sum_c \frac{1}{2} ||\mathbf{X}_c^{(k)} - \mathbf{Y}_c^{(k)} W_c^{proj\intercal}||_2^2$
  - $\Delta W^{proj} = W^{proj} - W^{new}$
  - $F_\gamma = F_{new} + \sum_{i,j} (\Delta W^{proj} \odot \nabla W^{new})_{i,j} + \frac{\gamma}{2} ||\Delta W^{proj}||_2^2$
  - **if** $F_{proj} \leq F_\gamma$ **then**
    - *break*
  - **end**
  - $\gamma = \gamma \times \gamma_{\text{inc}}$
- **end**
- $W_c^{old} = W_c$
- $W_c = W_c^{proj}$
- $F_{new} = \sum_c ||\mathbf{X}_c^{(k)} - \mathbf{Y}_c^{(k)} W_c^{\intercal}||_2^2 + \rho_1 ||W||_1 + \rho_2 ||\Psi W||_1 + \rho_3 ||W||_{2,1}$
- **if** $|F_{new} - F_{old}| <$ *tolerance* **then**
  - converged=true
- **end**
- $F_{old} = F_{new}$
- $t_{old} = t$
- $t = \frac{1}{2}(1 + \sqrt{1 + 4t^2})$

end

Algorithm 3: getFGLASSO algorithm

Input:

- V: $C \times F$, matrix of initial regression weights, C is the number of tasks, F is the number of features)
- $\lambda_1, \lambda_2, \lambda_3$
- Ψ Lineage tree

Output:

- W^{proj} Fused Group LASSO projection

for $i = 1...F$ **do**

  $\mathbf{v} = V_{.,i}$ ▷ Column i of matrix V

  $\mathbf{w} = \text{flsa}(\mathbf{v}, \lambda_1, \lambda_2, \Psi)$

  $\mathbf{w}_i = \frac{\max(||\mathbf{w}||_2 - \lambda_3, 0)}{||\mathbf{w}||_2} \times \mathbf{w}$

end

$W^{proj} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_F)$

Algorithm 4: flsa algorithm

Input:

- $\mathbf{v}$, $C \times 1$, column of estimate of regression weights, C is the number of tasks
- λ_1, λ_2, regularization parameters of Fused LASSO
- Ψ Lineage tree

Output:

- $\mathbf{w}$ updated estimate of regression weights based on Fused LASSO penalty

Solve $\Psi\Psi^\intercal \mathbf{z} = \Psi\mathbf{v}$ for $\mathbf{z}$

$z_{max} = ||\mathbf{z}||_\infty$ ▷ Get the max absolute value of $\mathbf{z}$

if $\lambda_2 \geq z_{max}$ **then**

  $t = \text{mean}(\mathbf{v})$

  $t = \begin{cases} t - \lambda_1 & \text{if } t > \lambda_1 \\ t + \lambda_1 & \text{if } t < -\lambda_1 \\ 0 & otherwise \end{cases}$ ▷ Soft-thresholding

  $\mathbf{w} = (t, t, ..., t)^\intercal$ ▷ $C \times 1$ vector

end

else

  $\mathbf{w} = \textbf{SFA}_G(\mathbf{v}, \lambda_2, \Psi, \mathbf{z})$

  $\mathbf{w}(i) = \begin{cases} \mathbf{w}(i) - \lambda_1 & \text{if } \mathbf{w}(i) > \lambda_1 \\ \mathbf{w}(i) + \lambda_1 & \text{if } \mathbf{w}(i) < -\lambda_1 \\ 0 & otherwise \end{cases}$ ▷ Soft-thresholding on all elements of $\mathbf{w}$

end

Algorithm 5: Subgradient Finding Algorithm with Gradient descent (SFA$_G$)

Input:

-$\mathbf{v}$, λ_2,$\mathbf{z}$

-Ψ: Lineage

Output:

- $\mathbf{w}$: updated estimate of regression weight

L=largest eigen value of Ψ

while *not converged* **do**

▷ Convergence determined by max iterations or the duality gap

$g = \Psi\Psi^{\mathsf{T}}\mathbf{z} - \Psi\mathbf{v}$ ▷ Compute gradient

$\mathbf{z} = \mathbf{z} - (g/L)$

$\mathbf{z}(i) = \begin{cases} \lambda_2 & \text{if } \mathbf{z}(i) > \lambda_2 \\ -1 * \lambda_2 & \text{if } \mathbf{z}(i) < -\lambda_2 \end{cases}$ ▷ Project $\mathbf{z}$ in the limit of $[-\lambda_2, \lambda_2]$

$\mathbf{s} = \Psi\Psi^{\mathsf{T}}\mathbf{z} - \Psi\mathbf{v}$ ▷ get the gradient again

$gap = \lambda_2||\mathbf{s}||_1 + < \mathbf{s}, \mathbf{z} >$ ▷ Get the duality gap

if *gap* $<$ *tolerance* **then**

convergence=true

end

end

$\mathbf{w} = \mathbf{v} - \Psi^{\mathsf{T}}\mathbf{z}$

1.3 EM algorithm for DRMN learning

DRMNs are learned by optimizing the overall DRMN likelihood using an Expectation Maximization (EM) style algorithm that searches over the space of possible graphs for a local optimum (**Algorithm 6**). In the Maximization (M) step, we estimate transition parameters (M1 step) and the regulatory program structure (M2 step). In the Expectation (E) step, we compute the expected probability of a gene's expression profile to be generated by one of the regulatory programs. The M2 step uses multi-task learning to jointly learn the regulatory programs for all cell types using either the framework of DRMN-ST or DRMN-FUSED.

Algorithm 6: DRMN Algorithm

Input:
- Expression data $\mathbf{X} = \{\mathbf{X}_1, \cdots, \mathbf{X}_C\}$,
- Regulatory features $\mathbf{Y} = \{\mathbf{Y}_1, \cdots, \mathbf{Y}_C\}$,
- Initial module assignments $\mathbf{M} = \{M_1, ..., M_C\}$

Output:
- Regulatory programs $\mathbf{R} = \{\mathbf{R}_1 = (G_1, \Theta_1), ..., \mathbf{R}_C = (G_C, \Theta_C)\}$,
- Transition probabilities $\mathbf{\Pi} = \{\Pi_1, ..., \Pi_C\}$

while *not converged* **do**
- M1: Estimate transition parameters $(\Pi_1, \cdots, \Pi_C)$
- M2: Update regulatory programs $(G_1, \cdots, G_C, \Theta_1, \cdots, \Theta_C)$
- E: Update module assignment probabilities and module assignments $(\mathbf{\Gamma}, M_1, ..., M_C)$

end

Update soft module assignments (E step): Let $\gamma_{i|j}^{g,c}$ be the probability of gene g in cell type c to belong to module i, given that in its parent cell type c', it belonged to module j. Let $\mathbf{\Gamma}$ denote these probabilities for all genes across cell types. We also introduce α_g^c, a vector of size $K \times 1$ where each element $\alpha_g^c(c')$ specifies the probability of observations given the parent state is c'. We estimate the probabilities using a dynamic programming procedure, where values at internal nodes in the lineage tree are computed using the values for all descendent nodes, down to the leaves.

If c is a leaf node, we calculate

$$\gamma_{i|j}^{g,c} = P(X_{gc}|\mathbf{R}_c^{(i)}, \mathbf{Y}_c^{(i)}(g,:))\Pi_c(i|j)$$

where the first term is the probability of observing expression of gene g in cell type c in module i and $\mathbf{R}_c^{(i)}$

is the regulatory program and features of module i. The second term is the probability of transitioning from module j in the parent cell type c' to module i in cell type c.

For a non-leaf cell type c:

$$\gamma_{i|j}^{g,c} = P(X_{gc}|\mathbf{R}_c^{(i)}, \mathbf{Y}_c^{(i)}(g,:))\Pi_c(i|j) \prod_{c \to l \in \tau} \alpha_g^l(i)$$

For both internal and leaf cell types, we write the joint probability of gene g in cell type c to belong to module i, and, in its parent cell type c', to module j as $\gamma_{i,j}^{g,c} = \frac{\gamma_{i|j}^{g,c}}{\alpha_g^c(j)}$, where $\alpha_g^c(j) = \sum_i \gamma_{i|j}^{g,c}$ is the probability of g's expression in any module given parent module j.

Estimate transition parameters (M1 step): Let $\gamma_{k,k'}^{g,c}$ be the joint probability of gene g to belong to module k in cell type c, and, module k' in its parent cell type c' (computed above). We calculate the probability of transitioning from k' in c' to k in c as $\Pi_c(k,k') = \frac{\sum_g \gamma_{k,k'}^{g,c}}{\sum_{g,k,k'} \gamma_{k,k'}^{g,c}}$.

Update regulatory programs (M2 step): Recall that the regulatory program for each cell type c is $\mathbf{R}_c =< G_c, \Theta_c >$, where G_c is a set of regulatory interactions $f \to i$ from a regulatory feature f to a module i, and Θ_c are the parameters of a regression function for each module that relates the selected regulatory features to the expression of the genes in a module. $G_c^{(i)}$ denotes the set of features for module i. In the DRMN-ST approach, the regulatory interactions are learned for one module at a time, across all cell types at a time using a greedy hill-climbing framework. At initialization, for each module i, $G_c^{(i)}$ is an empty graph, and the Gaussian parameters are computed as the empirical mean and variance of the genes initially assigned to module i in each cell type. In each iteration, we score each potential regulatory feature based on its improvement to the likelihood of the model, and choose the regulator with maximum improvement. This regulator is added to the module's regulatory program for all cell types for which it improves the cell type-specific likelihood. In DRMN-FUSED, the structure and parameters of $\mathbf{R}_c^{(i)}$ are learned by optimizing the objective in **Eqn 8** using an accelerated gradient method (Nesterov 2005; Nesterov 2007).

Termination: DRMN inference runs for a set number of iterations or until convergence. Final module assignments are computed as maximum likelihood assignments using a dynamic programming approach. While module assignments between consecutive iterations do not change significantly, the final module as-

signments are significantly different from the initial module assignments, and predictive power of model significantly improves as iterations progress (though improvements are small after 10 iterations, **Supplemental Figure S17**). In our experiments, we ran DRMN for up to ten iterations. When using greedy hill climbing approach, the M2 step was run until up to five regulators were added per module.

1.4 Examining different features in DRMN

We used DRMN's expression modeling framework to examine the contribution of different regulatory features, such as sequence motifs, histone modifications and accessibility to variation in expression at each time point or cellular stage of a dynamic process. We used a nested CV scheme described in **Section Evaluating the ability to model expression**. These feature set types were examined with the reprogramming array and sequencing datasets. Briefly, we perform 3 fold cross validation, where we split the genes into 3 sets, use two to train our models, and use the trained model to predict the expression for genes in the remaining set. Our metric for comparison was Pearson's correlation between true and predicted expression in each module in a test set. We considered the following features for each gene to predict its expression.

- Motif. We defined motif features based on the presence of a motif instance of a transcription factor (TF) within the gene's promoter region, defined as ± 2500 around the gene TSS. We downloaded a meta-compilation of position weight matrices (PWMs) from various resources (see dataset-specific sections for details) for human (e.g., Cis-BP) or mouse (Cis-BP for dedifferentiation and Sherwood et al (Sherwood et al. 2014) for the two reprogramming datasets). For the two reprogramming datasets, we applied FIMO (Grant et al. 2011) to scan the mouse genome for significant motif instances ($p < 1e-5$). For the dedifferentiation and human ESC differentiation the PIQ software (Sherwood et al. 2014) was used to identify the significant motif instances. For each gene, we generated a vector of motif presence, one dimension for each motif with the value equal to the $-\log_{10}(p\text{-value})$ of a motif instance. If a gene had multiple motif instances for the same motif, we used the most significant instance (smallest p-value).

- Histone. For datasets with histone modifications measured, we used each histone mark as a separate feature. This included 8 features for the reprogramming array dataset, 9 features for the sequencing

dataset and 8 features for the H1ESC differentiation dataset. The feature value was the aggregated count value around the gene TSS followed by log transformation.

- Histone + Motif. This was the concatenation of histone modification features (Histone) where available, with the motif features for each gene.

- Accessibility. The Accessibility feature was a single feature repressing the aggregated ATAC-seq or DNase-seq reads around the gene promoter. After aggregating to the gene promoter, we quantile normalized and log transformed the values.

- Accessibility + Motif. This feature set represents the concatenation of the ATAC feature with the Motif feature set for each gene.

- Q-Motif. This feature set represents sequence-specific motif features scored by the ATAC-seq/DNase-seq signal producing a total of as many features as there are motifs with significant instances. We used BEDTools (`bedtools genomecov -ibam input.bam -bg -pc > output.counts`) to obtain the aggregated signal on each base pair. We defined the feature value as the log-transformed mean read count under each motif instance. If multiple instances of the same motif were mapped to the same transcript, the signal was summed. If a TF was mapped to multiple transcripts of the same gene, or multiple motifs of the same TF were mapped to the same gene, the max value was used.

- Histone + Accessibility + Motif. This feature set represents the concatenation of the Histone feature set, ATAC feature and the Motif feature set.

- Histone + Q-Motif. Similar to the Histone + Motif feature set, Histone + Q-Motif represents the concatenation of the Histone and Q-Motif feature set for each gene.

- Histone + Accessibility + Q-Motif. This feature set is similar to Histone+ATAC+Motif and represents the concatenation of Histone and Q-Motif feature sets with the ATAC feature.

We first compared DRMN-ST (**Figure S3B**,**C**) and DRMN-FUSED (**Figure S3D**,**E**) using sequence-specific motifs alone (Motif), histone marks (Histone) and a combination of the two (Histone+Motif), as these features were available for both array and sequencing datasets. In both models, motifs alone (blue

markers) have low predictive power across different k for both array (**Figure S3B**, **D**) and sequencing (**FigureS3C**, **E**) data. As expected, histone marks alone (red marker) have higher predictive power, however adding both histone marks and motif features (magenta) has the best performance for $k = 3$ and 5, with the improved performance being more striking for DRMN-ST. For DRMN-Fused, Histone only and Histone+Motif seemed to perform similarly, although at higher k using histone marks alone is better. Between different cell types the performance was consistent in array data, while for sequencing data, the MEF and MEF48 cell types were harder to predict than ESC and preIPSC (**Supplemental Figure S1**).

We next examined the contribution of accessibility (ATAC-seq) data in predicting expression using the sequencing dataset for which accessibility was available (**Figure S3C**, **E**). We incorporated the ATAC-seq data in five ways: a single feature defined by the aggregated accessibility of a particular promoter and individual motifs (Accessibility+motif, orange markers), using ATAC-seq to quantify the strength of a motif instance (Q-Motif, light blue markers), combining the Accessibility feature with histone and motifs (dark purple marker), combining Q-Motif with histone (Histone+Q-motif, light green), and the Accessibility feature with histone and Q-motifs (Histone+Accessibility+Q-motif, dark green, **Figure S3C**, **E**). We also considered ATAC-seq alone but this was not very helpful (**Supplemental Figure S2**).

Combining the Accessibility feature together with Motif improves performance over Motif alone (**Figure S3C**, **E** orange vs. dark blue markers). The Q-Motif feature (light blue markers) was better than the Motif only (dark blue) at lower k (k=3), however, it did not outperform Motif at higher k. One possible explanation is that the Q-Motif feature is sparser than Motif because a motif instance that is not accessible will have a zero value and does not add predictive power at higher k. Finally, Accessibility feature combined with Histone+Motif features is comparable to Histone+Motifs (**Figure S3C**, magenta markers). We observe similar trends with Histone+Q-Motif and Histone+Accessibility+Q-motif features (**Figure S3C**, light and dark green **Figure S3C**). It is possible that the overall cell type specific information captured by the accessibility profile is redundant with the large number of chromatin marks in this dataset and we might observe a greater benefit of ATAC-seq if there were fewer or no marks.

When we directly compared DRMN-ST to DRMN-FUSED, DRMN-FUSED was able to outperform DRMN-ST on both Motif and Histone and comparable on Histone+motif on array data (**Figure S3F**). On the sequencing data, DRMN-FUSED had a higher performance that DRMN-ST on Motif, Q-Motif, Acce-

sibility+motif and Histone alone features (**Figure S3G**). It is likely that DRMN-ST learns a sparser model at the cost of predictive power (**Supplemental Figure S4**). For the application of DRMNs to real data, we focus on DRMN-FUSED due to its improved performance.

1.5 Effect of hyper-parameters on DRMN-FUSED results

We used the reprogramming array and sequencing datasets to study the effect of hyper-parameters on the performance of DRMN-FUSED with different feature sets. The hyper-parameters are ρ_1 (sparsity in each task), ρ_2 (selection of more similar features for closely related cell types) and ρ_3 (selection of similar features for all cell types). We performed a grid search on a range of parameters values: $\rho_1 \in \{0.5, 1, 2, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130\}$, $\rho_2 \in \{0, 10, 20, 30, 40, 50\}$ and $\rho_3 = \{0, 10, 20, 30, 40, 50\}$ (**Supplemental Figure S19**). To compare different feature sets we used three-fold cross validation as described in **Section Evaluating the ability to model expression** to assess the predicted expression for each module and cell stage. We used the average over modules and cell stages to assess DRMN performance for a particular feature set and hyper-parameter setting. We observe that increase in ρ_3 was generally not beneficial for the Motif feature for all k (number of modules), and for $k \geq 7$ when using Histone and Histone+Motif (blue ρ_3=0 *vs.* cyan, ρ_3=50, **Supplemental Figure S19**). For a fixed value of ρ_3, increasing sparsity ρ_1 is beneficial for the Histone and Histone + Motif features, upto $\rho_1 = 30 - 60$, beyond which the performance decreases or does not improve. The ρ_2 feature was also most useful when using the Histone feature.

For the sequencing dataset we considered all the feature types described in the section, **Feature sets tested in DRMN**, **Supplemental Methods**. The Motif feature was generated in a similar manner as the reprogramming array dataset. In addition, we included, Q-Motif, Accessibility and their combinations with the Histone feature. Similar to the array dataset, we used this dataset to study the effect of hyper-parameters, ρ_1, ρ_2 and ρ_3 on the performance of DRMN-FUSED using a similar three-fold cross-validation framework (**Supplemental Figures S21**, **S22**, **S23**). As in the array dataset, increasing values of ρ_3 was not beneficial for Motif or Q-Motif alone. Higher value of ρ_3 was useful for some of settings of histones features combined with Accessibility, Motif or Q-motif ($k = 3, 5$ for generally lower values of ρ_1, **Supplemental Figures S21, S22, S23**). We next investigate the impact of ρ_1 and ρ_2, for different values of ρ_3. We observe

that when using histone features (Histone+Motif, Histone+Accessibility+Motif, Histone+Accessibility+Q-Motif), increase in ρ_2 (increasing the similarity of inferred networks) improves the predictive power of the method. Increase in ρ_1 is beneficial for these features up to a limit (typically, ρ_1=60 or 70). Conversely, for the feature sets that do not use histone features (Motif, Q-Motif, and Accessibility+Motif), increase in ρ_1 (sparser models) decrease the predictive power of the model, which is consistent with the decrease in performance of Motif features in the array dataset.

1.6 Predicting regulators for transitioning gene sets

Simple linear regression approach. To identify regulators associated with transitioning gene sets in datasets with ≤ 5 samples (the two reprogramming datasets and H1ESC differentiation dataset), we used a simple regression-based approach to find regulators that can explain the overall variation in expression of genes in the set. Briefly, for each transitioning gene set with n genes across C cell types, we created a $n * C \times 1$ expression vector by stacking of the C-dimensional vector for each of the genes across the C cell types. We repeated this procedure for all the features associated with the gene set to produce a $n * C \times F$ matrix, where F is the total number of features in our dataset. Next, we used regularized regression to select which features are most predictive of the expression levels. Any regularized regression framework can be used; we used the sparsity imposing regression framework of the MERLIN algorithm (Roy et al. 2013), which uses a probabilistic framework with a prior term (tuned using a hyper-parameter) to infer sparser models. We ran MERLIN (with default settings) on each transitioning gene set to identify regulatory features associated with that set. We finally filtered the predicted regulators per gene set by assessing the correlation of the regulator/feature with the gene expression of a gene across the cell lines and included a regulator if was correlated to at least 5 genes with a Pearson's correlation of 0.6 or higher.

Multi-Task Group LASSO. To identify regulators associated with transitioning gene sets in datasets with ≥ 6 samples, e.g., the dedifferentiation dataset, we used a multi-task regression framework called Multi-Task Group LASSO (MTG-LASSO). In MTG-LASSO, we perform multiple regression tasks, one for each gene in the set to select regulatory features as predictors for each gene's expression levels. The "group" penalty of MTG-LASSO enables us to select the same regulator for all genes in the gene set but with

different parameters. The regulatory feature defines the "group", which is the set of regression weights for the regulator and each gene in the gene set. MTG-LASSO selects or unselects entire groups, and therefore, regulatory feature, of regression weights. The MTG-LASSO objective for each gene set s is:

$$\min_{\Psi_s} \sum_{g} ||X_{sg} - \mathbf{Y}_{sg}\Psi_{sg}||_2^2 + \lambda \sum_{f} ||\psi_{f.}||_2,$$

Here X_{sg} is the $C \times 1$ vector of expression values over C samples for gene g, and $\mathbf{Y}_{sg}$ is the $C \times F$ matrix of regulatory features for g over samples. $\Psi_{sg} = [\psi_{1g}, \cdots, \psi_{Fg}]^{\mathsf{T}}$ is the $F \times 1$ vector of regression weights for predicting g's expression from the F regulatory features. The first term denotes the regression task for each gene g, while the second term denotes the L1/L2 regularization required for the MTG-LASSO framework. The sum over f imposes the L1 penalty selecting a small number of groups (one $\theta_{f.}$ for each feature f), and the $||\theta_{f.}||_2$ imposes the L2 norm for smoothness of the regression coefficients across genes. λ is the hyper-parameter controlling for the strength of the regularization.

We applied MTG-LASSO to each transitioning gene set using the MATLAB SLEP v4.1 package (Jura et al. 2008) to infer the most predictive regulatory features for the gene set. We performed leave-one-out cross-validation, where one sample is left out from training, a model is fit on the remaining samples and used to predict the left out sample. We computed a confidence for each feature based on the percentage of models in which the feature is selected. Additionally, we computed a p-value for the selection of each regulator by comparing the number of times it was selected to a null distribution of feature selection obtained from randomizing the data 40 times and training MTG-LASSO models. A feature was selected as a regulator if it was in at least 60% of the trained values and had a p-value< 0.05. We tried different hyper-parameter values ($\lambda \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$) and selected the value that resulted in reasonable number of regulators across all transitioning gene sets ($\lambda = 0.7$). Once regulators were selected for each gene set, we further filtered the features based on the Pearson's correlation of the gene expression and feature values as in the simple regression case.

References

Grant CE, Bailey TL, and Noble WS. 2011. FIMO: scanning for occurrences of a given motif. *Bioinformatics*. **27**: 1017–1018.

Jura J, Wegrzyn P, Korostynski M, Guzik K, Oczko-Wojciechowska M, Jarzab M, Kowalska M, Piechota M, Przewlocki R, and Koj A. 2008. Identification of interleukin-1 and interleukin-6-responsive genes in human monocyte-derived macrophages using microarrays. *Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms*. **1779**: 383–389.

Liu J, Yuan L, and Ye J 2010. An Efficient Algorithm for a Class of Fused Lasso Problems. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '10. Washington, DC, USA: Association for Computing Machinery, pp. 323–332.

Nesterov Y. 2005. Smooth minimization of non-smooth functions. *Mathematical Programming*. **103**: 127–152.

Nesterov Y. 2007. Gradient methods for minimizing composite objective function. *Center for Operations Research and Econometrics (CORE)*. **5**: 4.

Roy S, Lagree S, Hou Z, Thomson JA, Stewart R, and Gasch AP. 2013. Integrated Module and Gene-Specific Regulatory Inference Implicates Upstream Signaling Networks. *PLoS Comput. Biol.* **9**: e1003252+.

Sherwood RI, Hashimoto T, O'Donnell CW, Lewis S, Barkal AA, van Hoff JP, Karun V, Jaakkola T, and Gifford DK. 2014. Discovery of directional and nondirectional pioneer transcription factors by modeling DNase profile magnitude and shape. *Nat Biotechnol*. **32**: 171–178.

Zhou J, Chen J, and Ye J 2012. MALSAR: Multi-tAsk Learning via StructurAl Regularization – User's Manual Version 1.1.