Lineage-based identification of cellular states and expression programs

Tatsunori Hashimoto\(^1\), Tommi Jaakkola\(^1\), Richard Sherwood\(^2\), Esteban O. Mazzoni\(^3\), Hynek Wichterle\(^3\) and David Gifford\(^1\)

\(^1\)Department of Computer Science and Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139

\(^2\)Brigham Women’s Hospital and Harvard Medical School, Boston, MA 02115 and

\(^3\)Departments of Pathology, Neurology and Neuroscience, Center for Motor Neuron Biology and Disease and Columbia Stem Cell Initiative, Columbia University Medical Center, New York, NY 10032, USA

ABSTRACT

Summary: We present a method, LineageProgram, that uses the developmental lineage relationship of observed gene expression measurements to improve the learning of developmentally relevant cellular states and expression programs. We find that incorporating lineage information allows us to significantly improve both the predictive power and interpretability of expression programs that are derived from expression measurements from in vitro differentiation experiments. The lineage tree of a differentiation experiment is a tree graph whose nodes describe all of the unique expression states in the input expression measurements, and edges describe the experimental perturbations applied to cells. Our method, LineageProgram, is based on a log-linear model with parameters that reflect changes along the lineage tree. Regularization with \(L_1\) that based methods controls the parameters in three distinct ways: the number of genes change between two cellular states, the number of unique cellular states, and the number of underlying factors responsible for changes in cell state. The model is estimated with proximal operators to quickly discover a small number of key cell states and gene sets. Comparisons with existing factorization, techniques, such as singular value decomposition and non-negative matrix factorization show that our method provides higher predictive power in held, out tests while inducing sparse and biologically relevant gene sets.

Contact: gifford@mit.edu

1 INTRODUCTION

The directed differentiation of embryonic stem (ES) cells into therapeutically important cell types holds great potential for regenerative medicine. Identifying stage-specific transcription factor candidates for the directed differentiation of ES cells has been difficult, and the computational identification of lineage-associated transcription factor programs would significantly benefit this process.

LineageProgram is a new method that identifies the experimental protocols that result in the same cellular states (Figure 1). It further decomposes these states into interpretable expression programs, which we define to be sets of co-varying genes. In contrast to analyzing the correlation between genes, we define a program to be sets of genes that co-vary during a differentiation event. Analysis of developmental expression data has revealed the existence of expression programs regulating pluripotency across many cell types \(^9\) as well as lineage-specific programs. We provide a principled method that discovers both types of programs. LineageProgram is a log-linear model that uses latent factors and \(L_1\) regularization to obtain sparse parameters, structures and spectra.

Our primary goal is to estimate expression programs that are informed and improved by lineage information. To our knowledge, LineageProgram is the first method to approach this problem. Prior work in the estimation of developmental expression programs has used biclustering, factor or topic decompositions, mixture models and self organizing maps, all of which simultaneously identify expression programs and sample clusters, but without incorporating information from the experimental lineage. As shown in studies of expression time series, treating dependent expression data as independent observations can lead to significant loss of information \(^3\). Separate prior work has focused on estimating lineage trees and lineage states in the absence of expression programs. Differentiation has been modeled both as a time series without branching and de novo tree estimation.

The remainder of this article presents the LineageProgram model (Section 2), a comparative analysis of LineageProgram with other methods on lineage-associated expression data from motor neuron and pancreatic development (Section 3) and a conclusion about what we have learned about using lineage information (Section 4).
2 THE LINEAGEPROGRAM METHOD

Model

LineageProgram operates on $N$ expression measurements of $P$ genes made on an experimental tree with $M$ nodes, where the root corresponds to the ES cell state and edges correspond to experimental interventions. We represent the cell state at a node $i$ with the probability vector $\theta_i$, whose $k$th component is the probability that gene $k$ is transcribed. The expression measurement is modeled as proportional to a multinomial draw from $\theta_i$. Methods such as GeneProgram [10, 16] have successfully used this discrete-count model for expression. Our objective function is the continuous extension of the multimodal likelihood function. We will show later that this natural continuous extension exists as a discretization limit and allows us to handle continuous data such as microarray measurements directly.

A differentiation event is a change in $\theta$, which we represent by a log-odds change $g$. The change of a gene $k$ from a parent state $i$ with vector $\theta_i$ to child state $j$ is written as

$$\theta_{i,j}(k) = \frac{\theta_{i,j}(k) \exp(g_{i,j}(k))}{\sum_l \theta_{i,j}(l) \exp(g_{i,j}(l))}$$

This formulation of a log-odds count model has been shown to outperform analogous Latent Dirichlet Allocation type models [8].

We represent the root stem cell state in the experimental tree as a log probability vector $\phi$ of size $P$, and the remaining states are represented as log odds changes from their parent. For each node on the cell state tree, we write the expression probability as the sum of log changes along its path from the ES state and the ES cell expression $\phi$. Let $P_j$ be the set of nodes along the path from node $j$ to the ES state. Then the probability of observing gene $k$ at node $j$ is given as follows:

$$\theta_{i,j}(k) = \frac{\sum_{l \in P_j} \phi_{l,j}(k)}{\sum_{l} \exp(\sum_{i \in P_j} \phi_{l,i}(k))}$$

We represent the experimental structure as two matrix multiplications: an $M \times M$ path sum matrix $T$ with $T_{j,i} = 1$ if $i \in P_j$ and zero otherwise and an $N \times M$ observation matrix $D$ with $D_{i,j} = 1$, if the $i$th expression measurement was made at node $j$ and zero otherwise.

The parameters are represented as two matrices, $M \times P$ parameter matrix $\eta$ (which we will regularize to be sparse and low-rank) and a $1 \times P$ ES expression vector $\phi$.

Given the $N \times P$ data matrix $X_{i,j,k}$, which we interpret to be proportional to counts of an expression event, our log likelihood takes the form

$$llh(\eta) = \sum_{i,k} X_{i,j,k} \log \left( \frac{\exp((DT\eta)_{i,j,k} + \phi_k)}{\sum_l \exp((DT\eta)_{i,j,l} + \phi_l)} \right)$$

We design our regularization with three objectives: there should be few genes changing per differentiation event ($L_1$ penalty on $\eta$), few unique differentiation events ($L_2$ penalty on the rows of $\eta$) and few programs needed to explain the lineage (trace norm penalty on $\eta$). The $L_1$ penalty is the sum of absolute values of $\eta$, which induces $\eta$ to have entries with zeroes. The $L_2$ penalty is the sum of the row-norms of $\eta$, which induces $\eta$ to have rows that are all zeroes. Finally, the trace norm penalty is the sum of the singular values of $\eta$, which induces $\eta$ to have low rank.

With the regularization and defining the shorthand notation $|\eta|_1 = \sum_{i,k} |\eta_{i,k}|$, and $|\eta|_2 = \sum_{i,k} |\eta_{i,k}|^{1/2}$, the overall objective $f$ takes the form

$$f(\eta) = -\text{llh}(\eta) + \lambda_1 |\eta|_1 + \lambda_2 |\eta|_2 + \lambda_3 |\eta|_1^2 |\eta|_2^2$$

We show that discretizing the data $X$, to some precision $\beta$, and taking the limit as $\beta \to \infty$ has the equivalent minima (up to scaling and the zero set) by using $X$ without discretization. Note that the gradient of the log-likelihood function takes the form

$$\lim_{\beta \to 0} \frac{\partial \text{llh}(\eta)}{\partial \eta_i} = -\sum_{k} \frac{X_{i,k}}{\sum_l \exp((DT\eta)_{i,l,k} + \phi_k)}$$

which is the continuous extension of $\text{llh}$ up to a constant $1/\beta$. The regularization terms $|\eta|_1$, $|\eta|_2$ and $|\eta|_1^2 |\eta|_2^2$ are convex, but not strictly convex, so the optima of the continuous extension and the limit can differ up to elements of the zero set. Testing both small discretization and the continuous extension, we find no difference in results, but for completeness we use a threshold of $1 \times 10^{-5}$ to set a small neighborhood near zero to be part of the zero set.

Finally, we define the concept of an expression program as a set of basis vectors spanning $\eta$. The trace norm regularization implicitly penalizes the rank of matrix $\eta$, and for large $\lambda_3$, $\eta$ will have small rank and can be represented as the linear combination of a few “basis” programs. We choose the singular value decomposition of $T\eta$ as our program decomposition. The first $k$ programs have a natural interpretation as the best rank-$k$ approximation of the unnormalized log expression parameters.

Inference

The advantage of our method over topic model formulations is the convexity of our objective $f$, which guarantees fast convergence to the global maxima. Our overall inference strategy is to use gradient descent on the likelihood combined with proximal steps on each of the regularization terms. To speed convergence, we also use the accelerated proximal gradient method.

The proximal gradient method allows us to efficiently optimize convex functions of the form $f = f^* + g$, where $f^*$ is convex differentiable and $g$ is convex and continuous. $f$ is optimized with a gradient step on $f^*$ followed by a proximal operator, which uses a quadratic approximation of $f^*$ to optimize $f^* + g$. Given $x_k$, we generate the next iterate $x_{k+1}$ with the following update

$$x_{k+1} = \text{Prox}_{g,\epsilon}(x_k - \epsilon \nabla f^*(x_k))$$

$$\text{Prox}_{g,\epsilon}(x_k) = \text{argmin}_y \|y - x_k\|^2 + \epsilon g(y)$$
The goodness of fit as measured by held-out likelihood shows that the optimal model has few of edges and programs.

Cross-validated mean square errors on the full cell tree shows good predictive performance by LineageProgram.

In our case, our log likelihood is concave differentiable, and there are three convex continuous functions: $g_1$, $g_2$ and $g_3$, corresponding to each regularization term:

$$ f^*(\eta) = -\ln f(\eta) + \lambda_1 ||\eta||_1 + \lambda_2 \sum_i ||\eta_i|| + \lambda_3 ||\eta||_R. $$

The proximal operators for $g_1$ and $g_2$ are the soft-threshold operators:

$$ \text{Prox}_{\varepsilon}(\eta_{i,j}) = \begin{cases} \eta_{i,j} - \varepsilon \lambda_1, & : \eta_{i,j} > \varepsilon \lambda_1 \\ \eta_{i,j} + \varepsilon \lambda_1, & : \eta_{i,j} < -\varepsilon \lambda_1 \\ 0, & : ||\eta_{i,j}|| < \varepsilon \lambda_1 \end{cases} $$

The gradient for $f^*$ is given by the difference between predicted and observed counts.
For $\lambda_3$, the proximal operator can be written in closed form in terms of its singular value decomposition (SVD) [14]. Let $\eta = UD^T$ and $\max(D - \lambda_3, 0)$ be the SVD and entrywise subtraction followed by thresholding, then the proximal operator takes the form

$$\text{Prox}_{x, \epsilon}(\eta) = U \max(D - \lambda_3, 0) V^T.$$ 

At each step of the optimizer, we take some $\lambda_i$ and step size $\epsilon$ and produce the next iterate with

$$x_{t+1} = \text{Prox}_{x, \epsilon}(\text{Prox}_{x, \epsilon}(\text{Prox}_{x, \epsilon}(x_t - \epsilon x^\top x(x_t))).$$

The sequential proximal gradient converges for our objective due to separability. We also make use of the accelerated gradient method by which finds a sequence of $\lambda_i$ which converges toward the optima, using an internal variable $\eta_i$ and a magnification of the gradient, $\gamma_i$ to increase convergence rates near the mode.

$$x_{t+1} = \text{Prox}_{x, \epsilon}(\text{Prox}_{x, \epsilon}(\text{Prox}_{x, \epsilon}(y_t - \epsilon y^\top y y_t)))$$

$$\gamma_{t+1} = \frac{1 + \sqrt{4\eta_t^2 + 1}}{2}$$

$$y_{t+1} = x_t + \frac{\eta_t - 1}{\eta_{t+1}} (x_t - x_{t-1}).$$

In the context of a single proximal operator, this produces the optimal quadratic first-order convergence rate. In our case, the multiple proximal operators do not provide a guaranteed convergence rate, but in practice, we find the accelerated gradient makes convergence significantly faster. An implementation of this inference method as well as the results of our analysis are available from our website at http://rpreg.csail.mit.edu/resources.html.

For the remainder of the article we use a convergence tolerance of $10^{-5}$ and hot starts, which allows us to quickly find solutions over a list of candidate $\lambda_3$ values by using the optima of one problem to initialize a new problem with similar regularization penalties. This allows us to obtain the regularization path over $\lambda_1$ for the 105 array experiments below within 10 min on a computer with a Core 2 Duo e6300 CPU and 2 GB of memory.
Fig. 6. Gene set enrichment in Gene Ontology (GO) and manually curated developmental gene sets show programs recovered by LineageProgram are biologically relevant.

The optimal regularization parameters discovered by cross-validation significantly penalize both the number of edges and rank of the parameter \( \eta \). At this optimal value, we find that the estimated cell state tree in Figure 8 is significantly sparser than the experimental tree. We also found that a strong \( L_1 \) penalty dominated the group \( L_2 \) penalties, resulting in \( \lambda_1 \) controlling both structural and parameter sparsity.

**Necessity of trace norm regularization**

The trace-norm penalty is a critical part of modeling large branching factors. The \( L_1 \) and \( L_2 \) penalties shrink the changes between nodes to zero and bring the probabilities at each child-node toward their common parent. Therefore, the two edge penalties, \( \lambda_1 \) and \( \lambda_2 \), control whether leaves differ at all from their parent, rather than how they differ. In contrast, the trace norm penalty restricts directions in which the leaves can differ by forcing the leaves to lie within a small subspace near their parent.

To test the necessity of trace norm regularization, we compared our model with and without trace-norm regularization. For both models, we fit the regularization parameters through cross-validation and compared the mean squared error on a common held-out set of 16 arrays selected by removing one array from each node with replicates. Results in Figure 5 indicate that without the trace norm penalty, 7 of the 16 held-out observations show significant increases in squared error. All seven of these observations are children of the highest degree node in Figure 8, suggesting that the trace norm penalty plays a key role preventing overfitting on highly branching data.

**Methods compared**

Our algorithm was compared against three existing classes of approaches: SVD, non-negative matrix factorization (NMF) and latent Dirichlet allocation (LDA) on both held-out prediction error and quantitative program metrics. We were restricted to considering non-lineage informed methods since we did not find any lineage informed methods in the literature.

For each competing method, we tested major variants of the algorithms and chose to compare only to the best variant. For SVD, we tried direct decomposition of the data, mean subtraction (resulting in Principal Component Analysis (PCA)) and mean subtraction on pancreatic and motor neuron branches; simple mean subtraction outperformed the others and is shown here. For NMF, we tested Kullback-Leibler (KL) divergence and square distance minimization objective proposed by as well as sparse NMF; we use KL divergence minimization. For LDA, we used the collapsed Gibbs sampler with 20000 samples to perform best.

**Low cross-validation error on branching data**

The held-out mean squared prediction error measures the generalization performance of each of the algorithms. We split the expression measurements into cross-validation and held-out sets selecting 16 replicate experiments one from each node with more than one measurement. The cross-validation set is split into training and test sets using the procedure described in the cross-validation section. The models are fit on the training data and we calculate the squared error between the predicted values from the model and the test data.

Training on the full set of motor neuron and pancreatic data (Figure 3), we find that overall, LineageProgram has the lowest mean squared prediction error. On lineages with no branching, we would expect to see SVD perform best due to its direct minimization of squared error. We find that on the motor neuron lineage with no branching, our algorithm performs like SVD. In the worst case of non-branching data, our algorithm compares favorably to current factorization methods. Interestingly, LDA degenerates on the motor neuron held-out sets with behavior consistent across multiple replications. We find that LDA overfits on the later stage pancreatic states at the expense of the motor neuron states, which is consistent with behavior observed in prior work [8]. To rule out the possibility that the low performance of the competing algorithms was due to the inclusion of two differing array
As with prior work \cite{10}, we measure Gene Ontology (GO) term enrichment as a proxy for biological relevance. Finally, we want each program to encode for biologically meaningful sets of genes. Hence, we quantify the quality of the decomposition in terms of sparsity and biological relevance. First, we would hope for each program to have several dominant genes in order to produce candidate genes for further analysis. We measure sparsity by the coefficient distribution of the programs. Second, we want each program to encode for biologically meaningful sets of genes. As with prior work \cite{10}, we measure Gene Ontology (GO) term enrichment as a proxy for biological relevance. Finally, we want each program to map to a unique developmental gene set. We measure this objective with the proportion of the program explained by the dominant developmental gene set.

**Fewer genes are used to model the lineage**

To measure sparsity, we normalize the programs from all methods to have unit norm and measure the $L_2$ norm of each gene over all programs. If there are a few genes dominating each program, we would expect to see only a few genes with high $L_2$ norm. Our goal is to recover interpretable programs, where each program contains a few key genes with large and unique activations. Our results in Figure 7 show that LDA and LineageProgram both produce a few large coefficients and a large number of coefficients within machine precision of zero. The dominance of few genes in each program allows us to label each program with a set of representative genes in Table 6.

**Higher biological relevance of extracted programs**

To measure the biological relevance of the discovered programs, we performed GO enrichment analysis using the weighted Kolmogorov statistic from gene set enrichment analysis. Plotting GO enrichment across all programs as a function of Benjamini Hochberg corrected $P$-values (Figure 6), we find that LineageProgram achieves significantly higher GO enrichments when compared to the other methods. The higher GO enrichment across all $P$-values suggests that the programs recovered by LineageProgram more closely match subtrees of the GO annotation than those recovered by competing methods. If this were purely a result of chance, we would expect LineageProgram to outperform the others on a small subset of $P$-value cutoffs, rather than across all cutoffs. In combination with our sparsity results, this suggests that LineageProgram discovers small sets of biologically relevant genes in each program.

Figure 7 shows that LDA and LineageProgram both produce a few key genes with large and unique activations. Our results in Figure 7 show that LDA and LineageProgram both produce a few large coefficients and a large number of coefficients within machine precision of zero. The dominance of few genes in each program allows us to label each program with a set of representative genes in Table 6.

**Comparison of program quality**

Although methods such as SVD can reconstruct expression values well, these program decompositions are often difficult to interpret biologically. Therefore, we quantify the quality of the decomposition in terms of sparsity and biological relevance. First, we would hope for each program to have several dominant genes in order to produce candidate genes for further analysis. We measure sparsity by the coefficient distribution of the programs. Second, we want each program to encode for biologically meaningful sets of genes. As with prior work \cite{10}, we measure Gene Ontology (GO) term enrichment as a proxy for biological relevance. Finally, we want each program to map to a unique developmental gene set. We measure this objective with the proportion of the program explained by the dominant developmental gene set.
Identification of enriched GO terms and genes also show close correspondence with known biological facts. We correctly identify all major stages of the motor neuron differentiation process, from loss of pluripotency and development of positional identity to spinal motor neuron specification. On the pancreatic branch, we identify loss of pluripotency, suppression of mesodermal identity and finally specification of endodermal and pancreatic identity. We include full summary output of the enriched genes and GOs for each program in the appendix.

**CONCLUSION**

We have presented LineageProgram, a log-linear sparse regularized model and inference algorithm for lineage-associated expression data that provide strong interpretability with no loss in predictive power. Existing flat modeling methods were unable to cope with the large number of leaves that occur at the ends of the differentiation experiment, and significant statistical power is lost re-estimating the background cellular expression levels. Our biological metrics also suggest that flat factorization methods do not extract biologically meaningful expression programs, and modeling the differences between each differentiation state is necessary to estimating fine behaviors occuring during differentiation.

Our analysis of the combined pancreas and motor neuron data provides a computational analysis of the pancreatic and motor neuron differentiation pathways that recapitulate known biological markers and states. The ability of the sparse regularization to remove spurious branches in the lineage tree suggests the use of this model to estimate novel cellular states in later-stage differentiation experiments, where explicit cellular states are less characterized.

Although we have restricted our model to expression-based data, we hope to extend this framework of $L_1$-regularized branching models to sequencing data with similar structure. The penalty model presented here would allow a wide variety of log-convex generative models to incorporate lineage-associated structure.

**Funding:** National Institutes of Health [P01-NS055923 and 1-UL1-RR024920 to D.K.G.].

**Conflict of Interest:** none declared.

**REFERENCES**

Cellular states and expression programs


Eisenstein, J. et al. (2011) Sparse additive generative models of text.


