BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–10

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, Massachusetts, United States.
[2] Brigham Women's Hospital and Harvard Medical School, Massachusetts, United States
[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, United States

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

We present a method, Lineage Program, 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 based methods controls the parameters in three distinct ways: the number of genes which 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 nonnegative matrix factorization show that our method provides higher predictive power in held-out tests while inducing sparse and biologically relevant gene sets.

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 which result in the same cellular states. 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 [1]. 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 [4, 5], factor or topic decompositions [2, 10], mixture models [20], and self organizing maps [24], 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 [26, 25] as well as de novo tree estimation [6, 7].

The remainder of this paper presents the LineageProgram model (Section 2), a comparative analysis of LineageProgram with other methods on lineage-associated expression data from

*to whom correspondence should be addressed

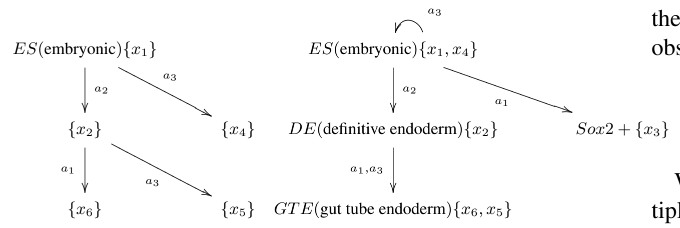


Fig. 1: Example experimental tree (left) and lineage tree (right). LineageProgram attempts to identify salient cellular states and covarying gene sets. Different treatments may result in the same cell state (a_1 and a_3 after definitive endoderm), while some treatments may have no effect (a_3 at stem cell). Our goal is to merge and prune these types of treatments.

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 θ_i, whose kth component is the probability that gene k is transcribed. The expression measurement is modeled as proportional to a multinomial draw from θ. Methods such as GeneProgram [10, 16] have successfully used this discrete-count model for expression. Our objective function is the continuous extension of the multinomial 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 θ, which we represent by a log-odds change η. The change of a gene k from a parent state i with vector θ_i to child state j is written:

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

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 ϕ 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 ϕ. Let $\mathcal{P}_j$ 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 by:

$$\theta_{(j,k)} = \frac{\exp(\sum_{i \in \mathcal{P}_j} \eta_{(i,k)} + \phi_k)}{\sum_l \exp(\sum_{i \in \mathcal{P}_j} \eta_{(i,l)} + \phi_l)}$$

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

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

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

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

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

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

$$f(\eta) = -llh(\eta) + \lambda_1 |\eta|_1 + \lambda_2 \sum_i ||\eta_i||_2 + \lambda_3 ||\eta||_{TR}$$

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

$$\begin{aligned} \lim_{\beta \to 0} \sum_{t,k} (\lfloor X_{t,k}/\beta \rfloor) \Bigg(& \frac{\lfloor X_{t,k}/\beta \rfloor}{\sum_{k'} \lfloor X_{t,k'}/\beta \rfloor} \\ & - \frac{\exp\left((DT\eta)_{(t,k)} + \phi_k\right)}{\sum_{k'} \exp\left((DT\eta)_{(t,k')} + \phi_{k'}\right)} \Bigg) \\ \to \frac{1}{\beta} \sum_{t,k} (X_{t,k}) \Bigg(& \frac{X_{t,k}}{\sum_{k'} X_{t,k'}} \\ & - \frac{\exp\left((DT\eta)_{(t,k)} + \phi_k\right)}{\sum_{k'} \exp\left((DT\eta)_{(t,k')} + \phi_{k'}\right)} \Bigg) \end{aligned}$$

which is the continuous extension of llh up to a constant $\frac{1}{\beta}$. The regularization terms $|\eta|_1$, $||\eta||_2$ and $||\eta||_{TR}$ 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 1e-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 η. The trace norm regularization implicitly penalizes the rank of matrix η, and for large λ_3, η 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 by Nesterov [21].

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_t, we generate the next iterate x_{t+1} with the following update:

$$\begin{aligned} x_{t+1} &= \text{Prox}_{g,\epsilon}\left(x_t - \epsilon \nabla f^*(x_t)\right) \\ \text{Prox}_{g,\epsilon}(x_t) &= \text{argmin}_Y ||x_t - Y||/2 + \epsilon g(Y) \end{aligned}$$

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

$$\begin{aligned} f(\eta) = -\overbrace{llh(\eta)}^{f*} + \overbrace{\lambda_1|\eta|_1}^{g_1} + \overbrace{\lambda_2 \sum_i ||\eta_i||}^{g_2} \\ + \overbrace{\lambda_3||\eta||_{TR}}^{g_3} \end{aligned}$$

The gradient for f^* is given by the difference between predicted and observed counts:

$$\begin{aligned} \nabla f^*(\eta_j)_k = T^T \sum_t \Bigg(X_{(t,k)} \\ - \left(\sum_{t',l} X_{(t',l)}\right) \frac{\exp\left((DT\eta)_{(t,k)} + \phi_k\right)}{\sum_l \exp\left((DT\eta)_{(t,l)} + \phi_l\right)} \Bigg) \end{aligned}$$

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

$$\text{Prox}_{g_1,\epsilon}(\eta_{(i,j)}) = \begin{cases} \eta_{(i,j)} - \epsilon\lambda_1 & : \eta_{(i,j)} > \epsilon\lambda_1 \\ \eta_{(i,j)} + \epsilon\lambda_1 & : \eta_{(i,j)} < -\epsilon\lambda_1 \\ 0 & : |\eta_{(i,j)}| < \epsilon\lambda_1 \end{cases}$$

$$\text{Prox}_{g_2,\epsilon}(\eta_{(i,k)}) = \begin{cases} \eta_{(i,k)}(1 - \epsilon\lambda_2 \frac{1}{||\eta_i||_2}) & : ||\eta_i||_2 > \epsilon\lambda_2 \\ 0 & : ||\eta_i||_2 < \epsilon\lambda_2 \end{cases}$$

For g_3, the proximal operator can be written in closed form in terms of its SVD [14]. Let $\eta = UDV^T$ and $\max(D - \lambda_3\epsilon, 0)$ be the SVD and entrywise subtraction followed by thresholding, then the proximal operator takes the form:

$$\text{Prox}_{g_3,\epsilon}(\eta) = U \max(D - \lambda_3\epsilon, 0)V^T$$

At each step of the optimizer, we take some x_t and step size ϵ and produce the next iterate with:

$$x_{t+1} = \text{Prox}_{g_3,\epsilon}(\text{Prox}_{g_2,\epsilon}(\text{Prox}_{g_1,\epsilon}(x_t - \epsilon\nabla f^*(x_t))))$$

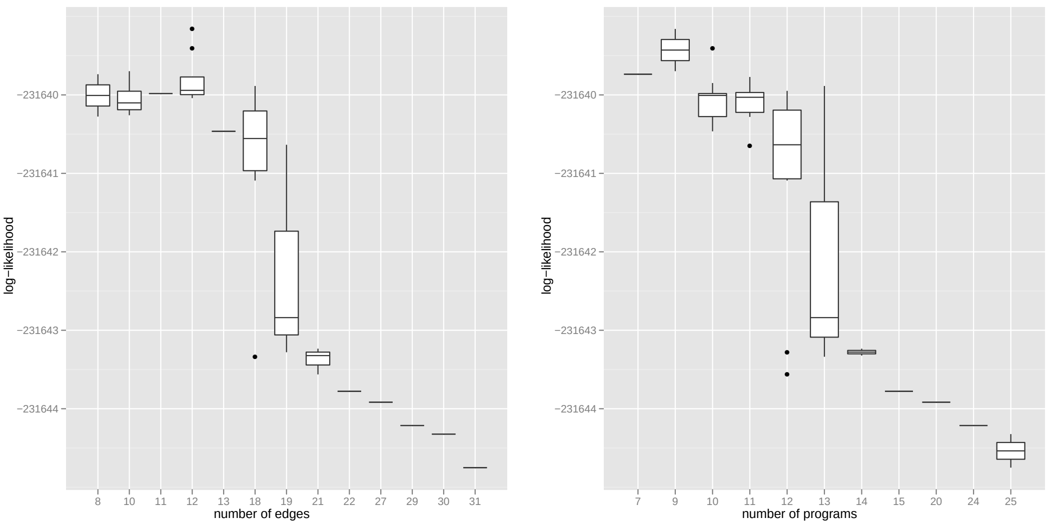
The sequential proximal gradient converges for our objective due to separability [18]. We also make use of the accelerated gradient method by Nesterov [21], which finds a sequence of x_t which converges toward the optima, using an internal variable y_t and a magnification of the gradient, a_t to increase convergence rates near the mode.

$$\begin{aligned} x_{t+1} &= \text{Prox}_{g_3,\epsilon}(\text{Prox}_{g_2,\epsilon}(\text{Prox}_{g_1,\epsilon}(y_t - \epsilon\nabla f^*(y_t)))) \\ a_{t+1} &= (1 + \sqrt{4a_t^2 + 1})/2 \\ y_{t+1} &= x_t + \frac{a_t - 1}{a_{t+1}}(x_t - x_{t-1}) \end{aligned}$$

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 is available from our website at psrg.csail.mit.edu/resources.html

For the remainder of the paper, we use a convergence tolerance of 10^{-5} and hot starts, which allows us to quickly find solutions over a list of candidate λ_1 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 λ_1 for the 105 array experiments below within 10 minutes on a computer with a Core 2 Duo e6300 CPU and 2 gigabytes of memory.

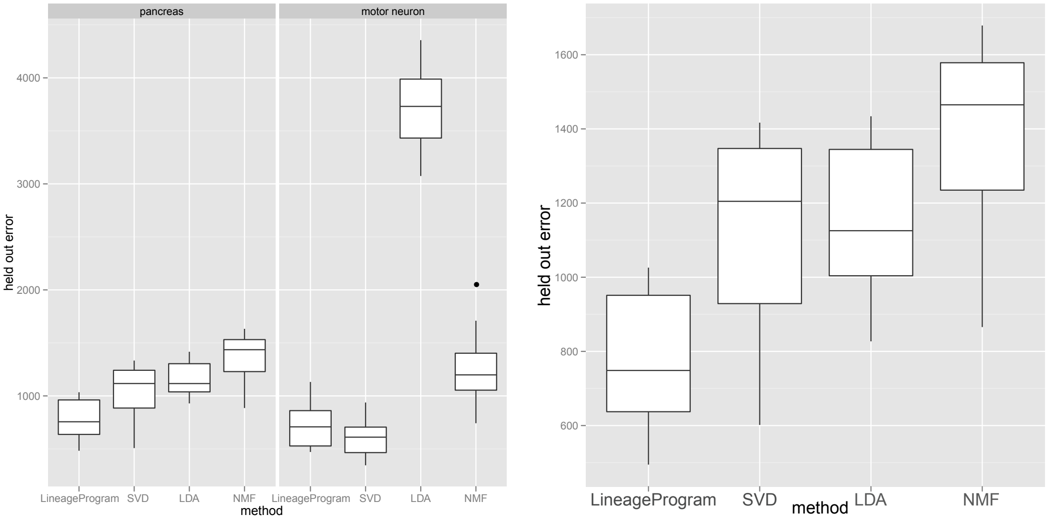
Inference with this method is fast enough that we are able to fit the model across a $50 \times 50 \times 50$ grid of all valid $\lambda_1, \lambda_2, \lambda_3$, which we use to set the regularization parameters as described in the next section.



(a) Edge sparsity

(b) Program sparsity

Fig. 2: The goodness of fit as measured by held out likelihood shows that the optimal model has few of edges and programs



(a) Held-out errors on the full lineage tree

(b) Held-out error trained only on pancreatic lineage

Fig. 3: Cross-validated mean square errors on the full cell tree shows good predictive performance by LineageProgram

3 RESULTS

The algorithm was tested on directed differentiation experiments for murine pancreatic progenitors and motor neurons, as shown in Figure 4. Both lines were produced using known differentiation protocols [19]. The pancreatic line has a large number of states, but relatively few replicates, while the motor neuron line has multiple replicates per state. The 88 microarray measurements of the pancreatic line were performed with Illumina bead arrays, while the 17 in the motor neuron line were performed with Affymetrix 430a2 microarrays. We rank-match

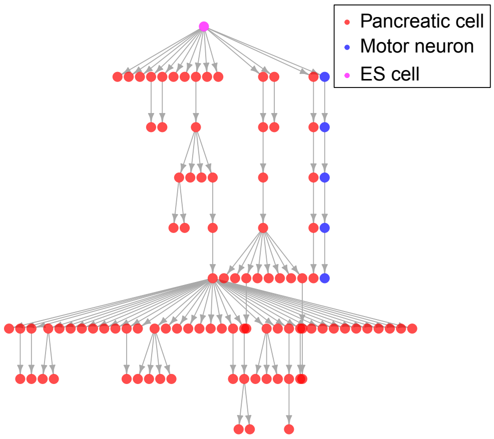


Fig. 4: Murine differentiation tree representing the derivation of 88 pancreatic and 17 motor neuron expression measurements. Each edge is an experimental treatment, each vertex a experimental state.

the Illumina data to the Affymetrix data in order to reduce the data to the same scale.

The quantile normalization technique is described in further detail by Irizarry et al [12].

Cross-validation procedure

We use cross-validation to estimate the regularization parameters λ_1, λ_2, λ_3 over the $50 \times 50 \times 50$ grid of all nondegenerate values. For every experimental state with more than one observation, we include one observation in the training set, and include the rest in the held-out set. We choose to leave out observations per-node rather than over all observations, since we must leave at least one observation at each node in order to make a prediction at the node. The performance of the model is measured in terms of held-out likelihood, which indicates goodness of fit, and squared error, which measures predictive performance.

Cross-validation results in Figure 2 shows the existence of a sharp drop in predictive power around a dozen edges and programs; this sharp transition suggests that there exists a necessary level of regularization for our algorithm to generalize well.

The optimal regularization parameters discovered by cross-validation significantly penalize both the number of edges and rank of the parameter η. 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 λ_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, λ_1 and λ_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 via 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, seven of the sixteen 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.

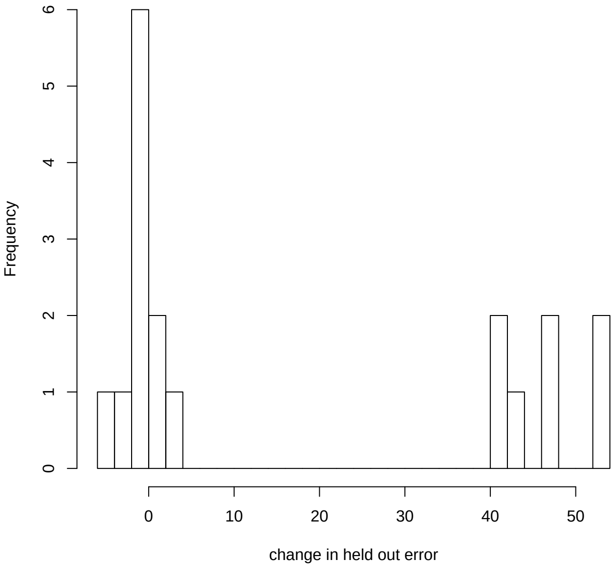
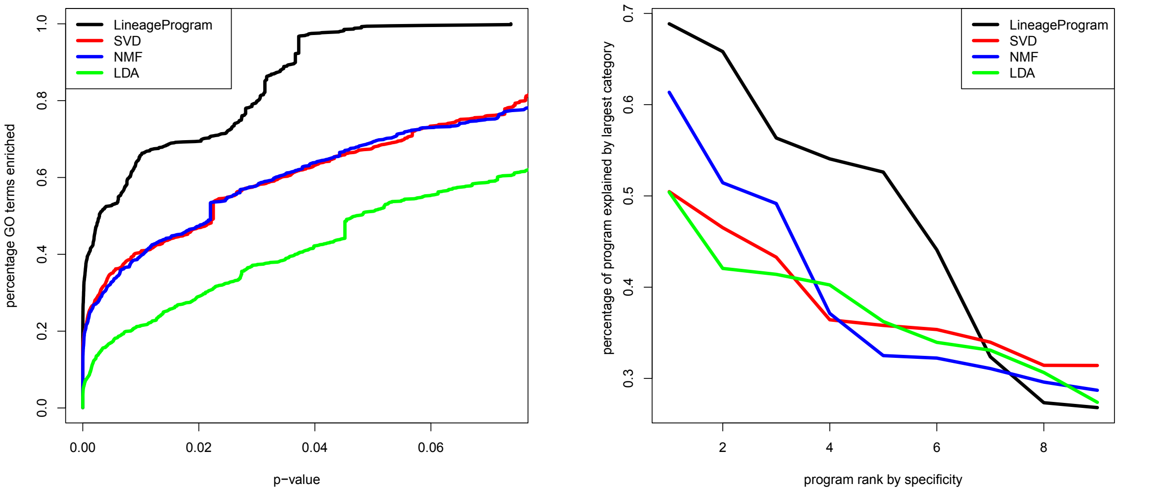


Fig. 5: Measureing held out error in models with and without trace norm penalty shows that removing trace norm penalty causes large increases in held out error for seven out of sixteen held out sets.



(a) Gene set enrichment analysis shows that LineageProgram recovers biologically relevant gene sets.

(b) The explanatory power of programs estimated by LineageProgram come from known developmental gene sets

Fig. 6: Gene set enrichment in GO and manually curated developmental gene sets show programs recovered by LineageProgram are biologically relevant.

Methods compared

Our algorithm was compared against three existing classes of approaches: singular value decomposition (SVD), nonnegative 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 PCA), and mean subtraction on pancreatic and motor neuron branches; simple mean subtraction outperformed the others and is shown here. For NMF, we tested KL divergence and square distance minimization objective proposed by Lee and Seung [17] as well as sparse NMF [11]; we use KL divergence minimization. For LDA, we used the collapsed Gibbs sampler as well as variational Bayes updates and found the Gibbs sampler with 20,000 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 3a), 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 as well as SVD. In the worst case of nonbranching data, our algorithm compares favorably to current factorization methods. Interestingly, LDA degenerates on the motor neuron held-out sets, with behavior consistent on both training methods and 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 types in the pancreatic and motor lineages, we re-ran the test, fitting and evaluating only on the pancreatic lineage (Figure 3b).

The tests on the pancreatic lineage show nearly the same results for all methods, with all methods, particularly LDA, performing slightly better. However, the general ordering of the methods remains the same, and the performance results are due to the inability of current methods to handle the large branching factors in the pancreatic lineage rather than the combination of multiple lineages.

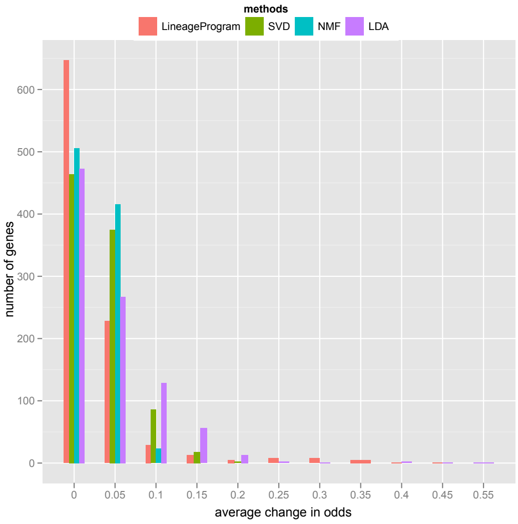


Fig. 7: The distribution of average odds change over all programs per gene shows that LineageProgram produces sparse programs with many genes at zero and few dominant genes.

Comparison of program quality

While 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 this objective by the coefficient distribution of the programs. Second, we want each program to encode for biologically meaningful sets of genes. As with prior work [10], we measure GO term enrichment as a proxy for biological relevance. Lastly, 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. The L_2 norm is measure of average change in odds across 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 8.

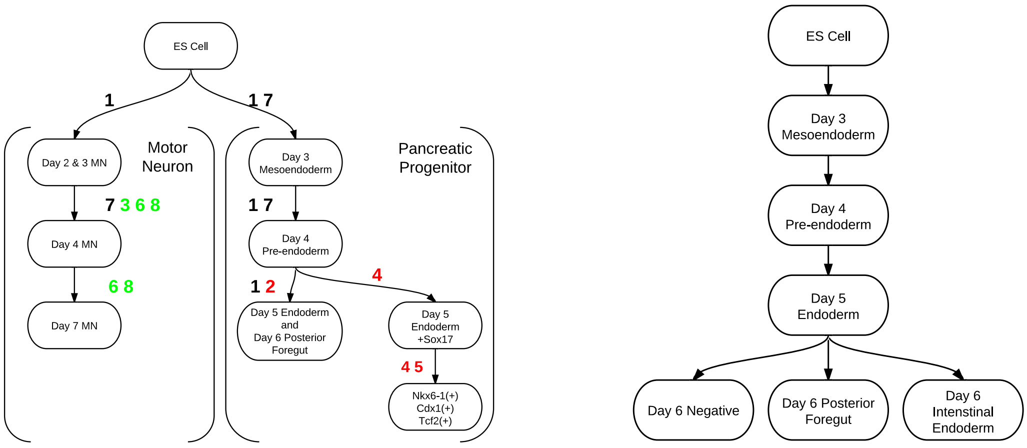
Higher biological relevance of extracted programs

To measure the biological relevance of the discovered programs, we performed GO enrichment analysis using the weighted Kolmogorov statistic from gene set enrichment analysis [23]. Plotting GO enrichment across all programs as a function of Benjamini-Hochberg corrected p-values (Figure 6a), 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.

To ensure that GO enrichment is in developmentally relevant categories, we also test the uniqueness of each program. In an ideal decomposition, each program would encode a different aspect of the developmental process. We use six developmental gene sets found in the literature, including four GO categories (“stem cell maintenance”,“endocrine pancreas differentiation”,“embryonic skeletal development”, and “anterior posterior development”), as well as marker genes for motor neurons (*Pax6, Mnx1, Isl1, Lhx1*,and *Lhx3*) [13] and pancreas (*Prox1, Pdx1, Hb9, Nkx6-1*) [15]. For each program, we measure the proportion of the program’s L_2 norm that is accounted by genes in the most activated category. The results in Figure 6b show that LineageProgram as well as NMF have programs closely matching known differentiation programs. These correspond to the loss of pluripotency; LDA and SVD were unable to discover programs that could be mapped solely to pluripotency. Importantly, LineageProgram maintains a relatively high correspondence to known gene sets across most of its programs, indicating its ability to encode small, highly specific gene modules corresponding to both the pancreatic and motor neuron lineages.

Analysis of full lineage data

Finally, we train LineageProgram on the full dataset to estimate lineage programs and use cross-validation on replicates to estimate regularization parameters. The resulting tree is sparse, with only two branching points (Figure 8). The first branching point corresponds to the split between motor neuron and pancreatic lineages. The other branching point differentiates a successful day 6 pre-endoderm differentiation from a late-stage Sox17 overexpression experiment.



(a) Model estimate of lineage tree

(b) Manually annotated pancreatic subtree

#	type	description	GO terms	genes
1	global	loss of pluripotency	(-)Stem cell maintenance*	(-)Nanog (-)Pou5f1
2	pancreatic	general endoderm specification	(+)Endocrine pancreas specification*	(-)Gbx2 (-)Sox2
3	neural	neural positioning program	(+)Embryonic skeletal system morphogenesis** (+)anterior/posterior formation*	(+)Hoxb5,Hoxc4,Hoxa3
4	pancreatic	suppression of mesodermal differentiation	(-)axial mesoderm development*	(-)Lhx1
5	pancreatic	endodermal specification	(+)endodermal cell fate specification*	(+)Nkx6-1 (+)Hnf1b
6	neural	postmitotic motor neuron specification	(+)Embryonic skeletal system morphogenesis*	(+)Isl1 (+)Mnx1 (-)Pax6
7	global	suppression of alternative paths	None below 0.05	(-)Hand1 (-)Hoxb8
8	neural	postmitotic motor neuron specification	None below 0.05	(+)Myl1 (+)Isl1

Fig. 8: (Top) Estimated lineage tree for motor neuron and pancreatic progenitor; nodes are grouped by cell type and edges are labeled by program involved; color indicates program type (see table below)
(Bottom) Major GO terms and Genes corresponding to developmental programs stars denote Benjamini-Hochberg corrected p-value of enrichment t-test $0.05 > * > 0.01 > **$

.

We compared the lineage tree from the model with a manual annotation produced separately from this project. Taking only the vertices corresponding to known cellular states, we match the manual curation almost exactly, successfully reducing all the branches before the day 4 preendoderm and only mismerging the day 5 endoderm with the day 6 posterior foregut. The regularization and cross-validation has successfully pruned nearly all spurious branches of the linage tree.

The Sox17 branch in Figure 8 represents a late stage experiment aimed at recovering competency in day 8 endoderm using Sox17 overexpression [22]. While the experiment shows significant expression level differences from the day 4 preendoderm, we have not positively identified this state as a biologically significant cell state.

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 provides 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 recapitulates 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.

While 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.

Acknowledgments

National Institutes of Health (P01-NS055923 and 1-UL1-RR024920 to D.K.G)

REFERENCES

[1]Koichi Akashi, Xi He, Jie Chen, Hiromi Iwasaki, Chao Niu, Brooke Steenhard, Jiwang Zhang, Jeff Haug, and Linheng Li. Transcriptional accessibility for genes of multiple tissues and hematopoietic lineages is hierarchically controlled during early hematopoiesis. *Blood*, 101(2):383–389, January 2003. PMID: 12393558.

[2]O Alter, P O Brown, and D Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences of the United States of America*, 97(18):10101–10106, August 2000. PMID: 10963673.

[3]Ziv Bar-Joseph. Analyzing time series gene expression data. *Bioinformatics*, 20(16):2493 –2503, November 2004.

[4]Pedro Carmona-Saez, Roberto D Pascual-Marqui, F Tirado, Jose M Carazo, and Alberto Pascual-Montano. Biclustering of gene expression data by non-smooth non-negative matrix factorization. *BMC Bioinformatics*, 7:78, 2006. PMID: 16503973.

[5]Y Cheng and G M Church. Biclustering of expression data. *Proceedings / ... International Conference on Intelligent Systems for Molecular Biology ; ISMB. International Conference on Intelligent Systems for Molecular Biology*, 8:93–103, 2000. PMID: 10977070.

[6]Ivan Costa, Stefan Roepcke, and Alexander Schliep. Gene expression trees in lymphoid development. *BMC Immunology*, 8(1):25, 2007.

[7]Ivan G. Costa, Stefan Roepcke, Christoph Hafemeister, and Alexander Schliep. Inferring differentiation pathways from gene expression. *Bioinformatics*, 24(13):i156 –i164, July 2008.

[8]J. Eisenstein, A. Ahmed, and E.P. Xing. Sparse additive generative models of text. 2011.

[9]Francesco Ferrari, Stefania Bortoluzzi, Alessandro Coppe, Dario Basso, Silvio Bicciato, Roberta Zini, Claudia Gemelli, Gian Antonio Danieli, and Sergio Ferrari. Genomic expression during human myelopoiesis. *BMC Genomics*, 8:264–264. PMID: 17683550 PMCID: 2045681.

[10]Georg K. Gerber, Robin D. Dowell, Tommi S. Jaakkola, and David K. Gifford. Automated discovery of functional generality of human gene expression programs. *PLoS Computational Biology*, 3:e148, 2007.

[11]P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004.

[12]R.A. Irizarry, B. Hobbs, F. Collin, Y.D. Beazer-Barclay, K.J. Antonellis, U. Scherf, and T.P. Speed. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, 4(2):249, 2003.

[13]T.M. Jessell. Neuronal specification in the spinal cord: inductive signals and transcriptional codes. *Nature Reviews Genetics*, 1(1):20–29, 2000.

[14]S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 457–464. ACM, 2009.

[15]M.C. Jørgensen, J. Ahnfelt-Rønne, J. Hald, O.D. Madsen, P. Serup, and J. Hecksher-Sørensen. An illustrated review of early pancreas development in the mouse. *Endocrine reviews*, 28(6):685, 2007.

[16]Je-Gun Joung, Dongho Shin, Rho Hyun Seong, and Byoung-Tak Zhang. Identification of regulatory modules by co-clustering latent variable models: stem cell differentiation. *Bioinformatics*, 22(16):2005 –2011, 2006.

[17]D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13, 2001.

[18]A.F.T. Martins, N.A. Smith, E.P. Xing, P. Aguiar, and M. Figueiredo. Online learning of structured predictors with multiple kernels. In *Proc. of the 14th Intl Conf. on Artificial Intelligence and Statistics*, 2011.

[19]E.O. Mazzoni, S. Mahony, M. Iacovino, C.A. Morrison, G. Mountoufaris, M. Closser, W.A. Whyte, R.A. Young, M. Kyba, D.K. Gifford, et al. Embryonic stem cell-based mapping of developmental transcriptional programs. *Nature methods*, 2011.

[20]M Medvedovic, K Y Yeung, and R E Bumgarner. Bayesian mixture model based clustering of replicated microarray data. *Bioinformatics (Oxford, England)*, 20(8):1222–1232, May 2004. PMID: 14871871.

[21]Y. Nesterov and I.U.E. Nesterov. *Introductory lectures on convex optimization: a basic course*. Applied optimization. Kluwer Academic Publishers, 2004.

[22]K.K. Niakan, H. Ji, R. Maehr, S.A. Vokes, K.T. Rodolfa, R.I. Sherwood, M. Yamaki, J.T. Dimos, A.E. Chen, D.A. Melton, et al. Sox17 promotes differentiation in mouse embryonic stem cells by directly regulating extra-embryonic gene expression and indirectly antagonizing self-renewal. *Genes & development*, 24(3):312, 2010.

[23]A. Subramanian, P. Tamayo, V.K. Mootha, S. Mukherjee, B.L. Ebert, M.A. Gillette, A. Paulovich, S.L. Pomeroy, T.R. Golub, E.S. Lander, et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 102(43):15545, 2005.

[24]Henry Wirth, Markus Loffler, Martin von Bergen, and Hans Binder. Expression cartography of human tissues using self organizing maps. *BMC Bioinformatics*, 12(1):306, 2011.

[25]Lan Zagar, Francesca Mulas, Silvia Garagna, Maurizio Zuccotti, Riccardo Bellazzi, and Blaz Zupan. Stage prediction of embryonic stem cell differentiation from genome-wide expression data. *Bioinformatics*, 27(18):2546 –2553, 2011.

[26]Bo Zhang, Beibei Chen, Tao Wu, Zhenyu Xuan, Xiaopeng Zhu, and Runsheng Chen. Estimating developmental states of tumors and normal tissues using a linear time-ordered model. *BMC Bioinformatics*, 12(1):53, 2011.