A guide to D-SPIN: constructing regulatory network models from single-cell RNA-seq perturbation data

Jialong Jiang[1,2] and Matt Thomson[1]

[1] Division of Biology and Biological Engineering, California Institute of Technology, Pasadena, CA, 91125, USA
[2] Center for Studies in Physics and Biology, The Rockefeller University, New York, NY, 10065, USA

Abstract. D-SPIN (Dimension-scalable Single-cell Perturbation Integration Network) is a computational framework for constructing regulatory network models from large-scale single-cell perturbation data. D-SPIN models the transcriptional state of cells as a set of individual genes or gene programs that interact through a condition-dependent spin network model or Markov random field. A single unified network is learned across multiple conditions, where the perturbations modulate the activity of individual network nodes. The architecture allows information integration between different perturbation conditions and efficient inference algorithms that scale to thousands of genes, thousands of conditions, and millions of single cells. For interpretability, D-SPIN can also operate at the level of co-expressing gene programs, which are extracted via unsupervised orthogonal non-negative matrix factorization (oNMF). Here we present the theory and detailed procedures for applying D-SPIN to build program-level or gene-level network models from single-cell data. With a public immune dictionary dataset, we illustrate multiple applications of D-SPIN, including identifying gene programs and network modules, classifying perturbation responses of cytokines, and nominating key regulators mediating cytokine responses.

Keywords: D-SPIN, Gene regulatory networks, Single-cell perturbation experiments, Spin networks, Markov random fields, Probabilistic graphical models, Perturb-seq

1 Introduction

Gene regulatory networks (GRNs) orchestrate gene expression in response to internal and external cues to achieve the physiological functions and survival of cells [1, 2]. However, the organizing principles of GRNs, especially their global control logic at the transcriptome scale, remain poorly understood. Recent single-cell RNA-seq approaches with large-scale multiplexed perturbation experiments, such as MULTI-seq and Perturb-seq, enable profiling of the transcriptomic cellular response to thousands of genetic or chemical perturbations [3–6]. While

perturbations are widely applied in genetics and known to be informative in revealing regulatory interactions [7, 8], how to convert these massive perturbation datasets into cell-scale models of regulatory networks has been a theoretical and computational challenge.

To address these limitations, we have developed D-SPIN (Dimension-scalable Single-cell Perturbation Integration Network), a novel theoretical framework that integrates multi-condition single-cell data into unified regulatory network models of perturbation responses [9]. D-SPIN models the cell as a collection of interacting genes or gene expression programs and constructs a probabilistic model, known as a spin network or Markov random field, to infer regulatory interactions between genes and applied perturbations. Spin network models originated in statistical physics and have been generalized and applied to a wide range of systems and applications through work on maximum entropy methods in physics and on associative memory of Hopfield networks in machine learning [10–12]. Spin network models have shown promise for the inference of gene regulatory network models from bulk gene expression data with the original Hopfield memory storage algorithms [13, 14]. However, the application of spin network models in the single-cell context remains limited due to the absence of an effective statistical inference framework that can scale to large networks, incorporate perturbations, and capture the information contained in the entire cell state distribution measured in single-cell experiments.

To enable information integration in large datasets with thousands of conditions and millions of single cells, D-SPIN exploits a natural factoring within the mathematical structure of the Markov random field model to separate the learning problem into two parts: 1) construction of a unified regulatory network model and 2) inference of how each perturbation interacts with the network model. Technically, the gradient function of the inference problem is estimated on each condition separately, similar to the minibatch training schemes commonly adopted in deep network architectures. Therefore, the training can be parallelized to hundreds of CPU cores, significantly speeding up the computation. Depending on the size of the network, D-SPIN employs three training algorithms with increasing scalability: exact maximum likelihood, Markov chain Monte Carlo (MCMC) maximum likelihood, and pseudolikelihood. Each of the methods is guaranteed to converge to the optimal solution with sufficient training data and computation time [15–17]. In benchmarking tests, we demonstrated that D-SPIN achieves state-of-the-art accuracy with perturbations compared with existing methods on both synthetic datasets and biological datasets with hundreds of elements [9, 18]. D-SPIN can also incorporate prior information on network topology during network inference, such as TF-motif binding or chromatin accessibility from ATAC-seq, to further improve inference accuracy.

Although D-SPIN scales well to large networks, network models with thousands of nodes are difficult to interpret. We designed D-SPIN to operate on both single genes and gene expression programs. Gene programs are a group of co-expressing genes across perturbation conditions [19, 20], and the program-level description enables D-SPIN to derive a coarse-grained description of a cellu-

lar regulatory network in terms of physiological functions and pathways, and their interaction with applied perturbations. Building upon existing algorithms, D-SPIN contains an automatic procedure for identifying gene programs from single-cell data with unsupervised orthogonal non-negative matrix factorization (oNMF) and phenotypic gene-program annotation, and can also accommodate pre-defined gene sets from prior biological knowledge [21–23]. Furthermore, to investigate the mechanisms of program-level response, we also developed a computational framework to identify key regulators from the gene-level networks that mediate the observed program responses.

D-SPIN provides an interpretable and generative framework for obtaining insights into the global regulatory mechanisms as well as identifying perturbation actions in the context of the regulatory networks. D-SPIN can accommodate a wide range of different perturbation types, including genetic perturbations, small molecules, signaling conditions, and even physiological states of health and disease. In this chapter, we present the detailed theoretical derivation and practical steps for the three major types of network construction in D-SPIN with our public Python package, including program-level and gene-level network construction and associating program-level response with gene regulators in the single-gene networks. With a public immune dictionary dataset of *in vivo* cytokine responses [24], D-SPIN reveals the cell-type-based network partitioning of immune response gene programs on CD4/CD8/regulatory T cells, dissects selected cytokines into three major categories (Th1-polarizing, IL-1 subfamily, and type I interferon), and proposes Stat2-Irf7/Stat1-Irf1 as the key diverging effectors mediating type I/type II interferon responses.

2 Materials

2.1 Software and code

D-SPIN is primarily implemented in Python and released through PyPI and GitHub (`github.com/JialongJiang/DSPIN`). D-SPIN can be installed through pip with `pip install dspin`. While the package dependence of D-SPIN is relatively simple, primarily `scanpy` and other basic scientific computing packages, we recommend setting up a new `conda` or `mamba` environment to avoid potential version conflicts between packages.

The Python implementation of D-SPIN is sufficient to handle datasets of 100k cells in a few hours on a local desktop. For large datasets with millions of cells, we also provide a MATLAB implementation that supports parallelization to multiple threads and significantly speeds up the computation. Through `parfor` parallelization, users can distribute computations over multiple local CPU cores or cluster nodes. The parallelization significantly improves running speed, often scaling linearly with the number of cores utilized.

2.2 Input data

D-SPIN is flexible regarding types of perturbations; the main assumption is that the core regulatory network is conserved across the conditions, even if the cell

state distributions vary. Examples of perturbations include Perturb-seq, drug treatments, signaling conditions, healthy/disease conditions, different patients, different time points, or even different local microenvironments from spatial transcriptomics. D-SPIN requires single-cell data, as D-SPIN explicitly models the complete distribution of transcriptional states. Performance on bulk sequencing data is more limited due to the loss of distribution information.

The single-cell data should be stored in `AnnData` format with the following attributes

- `adata.X`: Single-cell count matrix. Basic preprocessing and quality control steps are assumed for the single-cell data, such as filtering for low-quality cells and high mitochondrial gene content, and highly variable gene selection. The count matrix should be normalized by library size and log-transformed with the $\log(x + 1)$ function.
- `adata.obs['sample_id']`: Label for experimental conditions. Cells with the same `sample_id` are from the same perturbation condition. D-SPIN is tolerant to conditions with low cell numbers, as the network learning procedure would share information between different conditions. Nonetheless, each condition should contain a reasonable number of cells, for example, more than 25 cells per condition.
- `adata.obs['batch']`: Label for experimental batches. A batch refers to cells processed together in the same experimental run. Perturbation responses are primarily compared within the same batch for batch effect correction.
- `adata.obs['if_control']`: Label for whether the experimental condition is a control. Perturbation responses are compared with the control samples to obtain the relative response of perturbations. In the situation of no control samples, perturbations are compared with the average of all conditions.

3 Methods

3.1 Overview of D-SPIN

Theory of the D-SPIN framework Here, we develop the mathematical framework and computational strategies to integrate data across many different single-cell perturbation experiments into a gene regulatory network model. D-SPIN solves the gene regulatory network inference problem by constructing a probabilistic model that encodes the distribution of transcriptional states within a cell population across a set of perturbed conditions. Mathematically, D-SPIN builds a spin network model or Markov random field, which can be interpreted as an interaction network. Spin-network models were initially introduced in physics to study magnetic materials known as spin glasses, whose properties emerge through positive and negative interactions between localized magnetic moments or spins [15]. In physics and machine learning, spin network models have been generalized and applied to many different systems of interacting elements, including neural networks, bird flocks, and proteins [12, 14, 25], and also to study

the storage of memories in spin networks known as Hopfield networks [10]. However, the application of spin network models to single-cell data has been limited due to mathematical and computational challenges associated with solving the spin network inference problem on large datasets.

In general, spin network models consider a network of M interacting elements s_i. The goal of the framework is to construct a model that encodes the probability $P(s_1, \ldots, s_M)$ of each potential configuration of the network of elements with an energy function. Mathematically, the energy function $E(\boldsymbol{s})$ computes the effective energy of a given transcriptional state $\boldsymbol{s}$ by balancing the influence of the interactions between elements (encoded in $\boldsymbol{J}$) and the activity of individual elements (encoded in $\boldsymbol{h}$). The resulting probability of a given transcriptional state $\boldsymbol{s}$ is

$$P(\boldsymbol{s}) = \frac{1}{Z} \exp[-E(\boldsymbol{s}; \boldsymbol{J}, \boldsymbol{h})] \tag{1}$$

$$E(\boldsymbol{s}; \boldsymbol{J}, \boldsymbol{h}) = -\sum_{ij} J_{ij} s_i s_j - \sum_i h_i s_i \tag{2}$$

$$Z = \sum_{\boldsymbol{s}} \exp[-E(\boldsymbol{s}; \boldsymbol{J}, \boldsymbol{h})]\,, \tag{3}$$

where Z is a normalizing constant ensuring all probabilities sum to 1. The regulatory network $\boldsymbol{J}$ accounts for activating ($J_{ij} > 0$) and inhibitory ($J_{ij} < 0$) interactions between network nodes. The activity $\boldsymbol{h}$ exerts a bias on each element to increase the likelihood of being on ($h_i > 0$) or off ($h_i < 0$) (Fig. 1A).

The key insight in D-SPIN is that the regulatory network remains largely conserved across conditions, while the activities of individual genes are modulated by the perturbations to activate different internal modes of the network, creating shifted transcriptional state distributions. Specifically, all conditions share a unified $\boldsymbol{J}$ while each perturbation n has a distinct $\boldsymbol{h}^{(n)}$ (*see* **Note 1**) (Fig. 1B). Conceptually, the interaction matrix $\boldsymbol{J}$ defines an energy landscape of possible transcriptional states, while the perturbations $\boldsymbol{h}^{(n)}$ exert linear tiltings to the landscape to bias the observed cell state distributions. Specifically, D-SPIN optimizes the following joint log-likelihood function

$$\log \mathcal{L}(\boldsymbol{J}, \boldsymbol{h}^{(n)}) = \sum_{n}^{N} \sum_{\boldsymbol{s}^{(n)}_{\text{Data}}} \log P(\boldsymbol{s}^{(n)}_{\text{Data}} | \boldsymbol{J}, \boldsymbol{h}^{(n)})\,. \tag{4}$$

In D-SPIN, s_i is chosen to be three-state discrete variables in $[-1, 0, 1]$ instead of two-state variables in $[-1, 1]$ as is common in statistical physics. The three-state representation is chosen to model intermediate levels of gene expression, which can be functionally relevant for gene regulation and for improved quantitative agreement between model and experimental cell state distributions (*see* **Note 2**). Similar choices of discretization have been deployed to reduce noise and accommodate nonlinear dependencies from the pioneering works in modeling gene expression networks using microarray measurements [26]. The continuous expression level of genes or programs is discretized with K-means clustering (*see* **Note 3**).

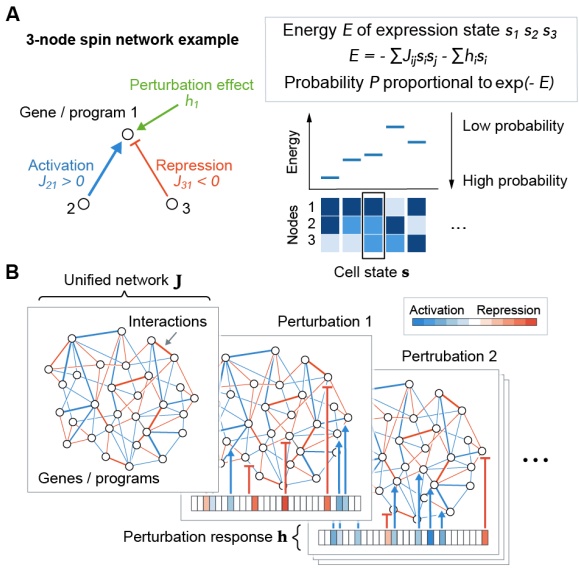

Fig. 1. Schematics of the D-SPIN framework. (A) The interactions between genes or programs and the perturbation effects define an energy function E. Cell states with lower energy are more likely states. (B) Across perturbation conditions, the network $\boldsymbol{J}$ remains constant while the activities of individual genes or programs are modulated by the perturbation $\boldsymbol{h}$.

Inference algorithms for D-SPIN The mathematical form of D-SPIN not only provides a natural and interpretable representation of perturbation action, but also enables us to develop efficient and scalable learning algorithms by separating the training of each condition. Training of general graphical models typically requires estimating the gradient function for every data point separately, which is computationally expensive. In contrast, the gradient of the D-SPIN objective function is pooled for each experimental condition. Consequently, the training of the network can be deployed in parallel, with each computational thread computing the gradient for a single experimental condition. The approach eliminates the need to estimate gradients on each single cell separately, and minimizes data communication cost by only passing the computed gradient of the parsimonious number of parameters in D-SPIN. With the parallelization toolbox in MATLAB, the pipeline can easily utilize hundreds of CPU threads, enabling efficient inference in million-level single-cell datasets.

The multi-condition spin network optimization solved by D-SPIN is a convex problem, ensuring that the only local optimum is also the global optimum [9]. Therefore, gradient-based optimizers can be deployed without the risk of getting trapped in local optima. However, challenges remain due to indistinguishable model and floppy model directions [27, 28]. Perturbations that generate diverse transcriptional states can help address these issues, leading to more accurate

inference results, which is also an important theoretical motivation for D-SPIN (*see* **Note 4**).

Specifically, we adapted three inference methods for D-SPIN, which are the most accurate methods among the various computational methods proposed in the spin network inference problem [15]. Each method can be advantageous under specific niches depending on the problem setting, such as network size, number of cells, and number of perturbation conditions.

Exact maximum likelihood inference The gradient of the objective function is

$$\frac{\partial \log \mathcal{L}}{\partial J_{ij}} = \sum_{n}^{N} \sum_{\boldsymbol{s}_{\text{Data}}^{(n)}} \frac{\partial}{\partial J_{ij}} \left[-E(\boldsymbol{s}_{\text{Data}}^{(n)}) - \log Z \right] \tag{5}$$

$$= \sum_{n}^{N} \sum_{\boldsymbol{s}_{\text{Data}}^{(n)}} \left\{ s_{i\text{Data}}^{(n)} s_{j\text{Data}}^{(n)} - \frac{1}{Z} \sum_{\boldsymbol{s}} s_i s_j \exp[-E(\boldsymbol{s}; \boldsymbol{J}, \boldsymbol{h}^{(n)}))] \right\} \tag{6}$$

The first term is the sum of $s_i s_j$ in all experimental samples, and the second term is a constant, which is the expectation of $s_i s_j$ of the current model, defined by the parameters $\boldsymbol{J}, \boldsymbol{h}^{(n)}$. We normalize the gradient by cell number and sample number to improve numerical stability under a given step size. With a similar derivation for $\partial \log \mathcal{L} / \partial h_i$, the gradients of the objective function can be written as follows.

$$\frac{\partial \log \mathcal{L}}{\partial J_{ij}} = \frac{1}{N} \sum_{n}^{N} \left(\langle s_i s_j \rangle_{\text{Data}}^{(n)} - \langle s_i s_j \rangle_{\text{Model}}^{(n)} \right) \tag{7}$$

$$\frac{\partial \log \mathcal{L}}{\partial h_i} = \langle s_i \rangle_{\text{Data}}^{(n)} - \langle s_i \rangle_{\text{Model}}^{(n)}, \tag{8}$$

where $\langle \cdot \rangle_{\text{Data}}^{(n)}$ is the average over the data of n-th conditions, and $\langle \cdot \rangle_{\text{Model}}^{(n)}$ is the expectation on the distribution defined by current model parameters $\boldsymbol{J}$, $\boldsymbol{h}^{(n)}$. Therefore, the modeling learning is essentially matching the cross-correlation $\langle s_i s_j \rangle$ and mean $\langle s_i \rangle$ between the model and data.

For small networks, the probability distribution $P(\boldsymbol{s}|\boldsymbol{J}, \boldsymbol{h})$ can be explicitly computed, thus the exact gradient of $\boldsymbol{J}, \boldsymbol{h}$ can be computed using Eq. 7, Eq. 8. Once the gradients are computed, any standard optimiser such as gradient ascent, Momentum, or Adam can be deployed to solve the optimization problem. As the gradient estimation requires enumerating all possible states, the computational complexity scales exponentially with the number of nodes. In practice, the method is only applicable to small networks of less than 15 nodes ($3^{15} \approx 1.4 \times 10^7$ states).

Markov Chain Monte Carlo (MCMC) maximum likelihood inference As the gradient only requires computing the mean and cross-correlation of the samples, the complete distribution $P(\boldsymbol{s}|\boldsymbol{J}, \boldsymbol{h})$ can be approximated by sampling an empirical distribution. Without evaluating the exact distribution, we can construct a

Markov Chain between states whose stationary distribution is $P(\boldsymbol{s}|\boldsymbol{J},\boldsymbol{h})$. Specifically, we utilize the Gibbs sampling scheme. Starting from a random initial state $\boldsymbol{s} = [s_1, s_2, \ldots, s_M]^\intercal$, at each step we randomly take an index k in the M nodes, and update the value of s_k by its conditional distribution given all other nodes $P(s_k|s_1, \ldots, s_{k-1}, s_{k+1}, \ldots, s_M, \boldsymbol{J}, \boldsymbol{h})$. After a burn-in period of steps to allow the Markov Chain to equilibrate, the sequence of samples is an empirical distribution of $P(\boldsymbol{s}|\boldsymbol{J},\boldsymbol{h})$ and can be used to estimate the gradient Eq. 7, Eq. 8.

In each sampling step, computing the marginal distribution is of complexity $\mathcal{O}(M)$. For accurate cross-correlation estimation, the required number of samples scales as $\mathcal{O}(M^2)$, without considering the burn-in period of the MCMC process. Therefore, the overall computational complexity is at least $\mathcal{O}(M^3)$. The MCMC method applies to medium-sized networks up to $30 \sim 50$ nodes.

Pseudolikelihood method The major challenge in scaling the inference to large networks is the partition function Z in the distribution $P(\boldsymbol{s}|\boldsymbol{J},\boldsymbol{h})$, which involves exponentially many terms with the network size M. An alternative approximation method called pseudolikelihood was developed originally for spatial statistics and has been adapted to spin network problems [15, 16, 29]. Pseudolikelihood methods approximate the distribution $P(\boldsymbol{s}|\boldsymbol{J},\boldsymbol{h})$ as the product of the conditional distribution of each individual variable, and provide exact inference in the limit of an infinite number of samples [15].

Specifically, we denote $\boldsymbol{s}_{\setminus k} = [s_1, \ldots, s_{k-1}, s_{k+1}, \ldots, s_M]^\intercal$ as the state $\boldsymbol{s}$ except s_k. The pseudolikelihood function for the inference problem of a single condition is

$$\text{Pseudo}P(\boldsymbol{s}|\boldsymbol{J},\boldsymbol{h}) = \prod_k^M P(s_k|\boldsymbol{s}_{\setminus k}, \boldsymbol{J}, \boldsymbol{h}) \tag{9}$$

$$= \prod_k^M \frac{\exp E(s_k, \boldsymbol{s}_{\setminus k}, \boldsymbol{J}, \boldsymbol{h})}{\sum_{r\in\{-1,0,1\}} \exp E(s_k = r, \boldsymbol{s}_{\setminus k}, \boldsymbol{J}, \boldsymbol{h})} \tag{10}$$

$$= \prod_k^M \frac{\exp(s_k\Theta_k + s_k^2 J_{kk})}{\exp(\Theta_k + J_{kk}) + 1 + \exp(-\Theta_k + J_{kk})}\,, \tag{11}$$

where $\Theta_k = h_k + \sum_{j\neq k} J_{jk}s_j$ is the effective field conditioned on all other nodes $\boldsymbol{s}_{\setminus k}$. The pseudolikelihood function decouples the mutual dependence between nodes, thus avoiding the computation of the exponentially complex partition function Z. As a cost, the pseudolikelihood function in general does not sum to 1, and thus is not a distribution; this is where the "pseudo" name comes from.

The gradient for the log pseudolikelihood objective function is

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{Pseudo}}}{\partial J_{ij,i\neq j}} &= \frac{\partial}{\partial J_{ij}} \sum_{k}^{M} \big(s_k \Theta_k + s_k^2 J_{kk} \\ &\qquad - \log\left(\exp(\Theta_k + J_{kk}) + 1 + \exp(-\Theta_k + J_{kk})\right)\big) \\ &= 2 s_i s_j - \sum_{k\in\{i,j\}} \frac{\exp(\Theta_k + J_{kk}) - \exp(-\Theta_k + J_{kk})}{\exp(\Theta_k + J_{kk}) + 1 + \exp(-\Theta_k + J_{kk})} \qquad (12)\end{aligned}$$

$$\frac{\partial \mathcal{L}_{\text{Pseudo}}}{\partial J_{ii}} = s_i^2 - \sum_{k=i} \frac{\exp(\Theta_k + J_{kk}) + \exp(-\Theta_k + J_{kk})}{\exp(\Theta_k + J_{kk}) + 1 + \exp(-\Theta_k + J_{kk})} \qquad (13)$$

$$\frac{\partial \mathcal{L}_{\text{Pseudo}}}{\partial h_i} = s_i - \sum_{k=i} \frac{\exp(\Theta_k + J_{kk}) - \exp(-\Theta_k + J_{kk})}{\exp(\Theta_k + J_{kk}) + 1 + \exp(-\Theta_k + J_{kk})}\,. \qquad (14)$$

The computational complexity of the gradient computation scales with $\mathcal{O}(M^2)$. Also, note that the effective field Θ_k depends on each cell state $\boldsymbol{s}$, so the gradient computational scales with the total number of cells. Practically, the pseudolikelihood method is highly scalable and even applies to networks of thousands of nodes. The key approximation of pseudolikelihood is to use the empirical distribution to approximate $\boldsymbol{s}_{\setminus k}$. The approximation is more accurate when the number of samples is high. Typically, the number of observed samples is far lower than the total number of possible states (3^M). Therefore, exact maximum likelihood and MCMC maximum likelihood are preferable when they are computationally feasible.

Assigning directions to the inferred interactions Identifying directions of interactions from stationary single-cell data has been a difficult problem [30]. The form of pseudolikelihood is closely related to regression models, enabling the assignment of directionality to the inferred network. The distribution of a single node conditioned on all other nodes

$$P(s_k | \boldsymbol{s}_{\setminus k}, \boldsymbol{J}, \boldsymbol{h}) = \frac{\exp(s_k \Theta_k + s_k^2 J_{kk})}{\exp(\Theta_k + J_{kk}) + 1 + \exp(-\Theta_k + J_{kk})} \qquad (15)$$

can be interpreted as a regression problem, where we predict the state of the dependent variable s_k with other variables are predictors. The interactions J_{ij} are the coefficients of the regression problem. If gene A predicts gene B better than B predicts A, this suggests a regulation direction of A to B. To compute the directional network, the gradient estimation Eq. 12 can be simply replaced with an asymmetric version of

$$\frac{\partial \mathcal{L}_{\text{Pseudo}}}{\partial J_{ij,i\neq j}} = s_i s_j - \sum_{k=i} \frac{\exp(\Theta_k + J_{kk}) - \exp(-\Theta_k + J_{kk})}{\exp(\Theta_k + J_{kk}) + 1 + \exp(-\Theta_k + J_{kk})} \qquad (16)$$

While the algorithm can infer a directed network and achieve high accuracy on synthetic benchmarking datasets [9], it is essential to interpret any directed

model from stationary single-cell RNA-seq data with caution. There are several constraints.

First, directed networks cannot define a stationary distribution on transcriptional states when time information is not supplied (*see* **Note 5**). In the context of probabilistic graphical models, directed models are always constrained to be acyclic, i.e., with no cycle. Such a constraint is reasonable in the field of causal inference, where the circularity of causal relations is rare. However, in cellular regulatory networks, feedback loops are prevalent to maintain homeostasis or to achieve signal amplification.

Second, for steady-state snapshot data, an undirected model can adequately explain the observed distribution of transcriptional states (*see* **Note 6**). Identifying directions, in principle, requires the dataset to contain dynamics, even if the time information is not explicitly measured in experiments.

Third, the direction assignment is hinged on the key assumption that the asymmetry of predictive power indicates the direction of information flow. The empirical assumption has been adapted in multiple methods such as GENIE3 and GRNBoost2 [30–32], but other factors may also create such asymmetry, for example, the noise level of gene expression. High-expression genes with low observation noise may appear to predict the noisy low-expression genes better compared with the reverse way of prediction.

Identifying gene regulators of programs The directed network inference algorithm enables associating gene-level networks with program expressions to reveal their mechanisms of regulation. In principle, each gene program can be treated as a "pseudo-gene" and included in the directed network construction. However, the intuitive approach has two major issues. First, as we have discussed, the inferred direction of interaction also depends on the noise level of genes or programs. As a weighted average of multiple genes, the program has a significantly lower noise level than single genes and may appear to be a regulator in all interactions, even though the causal direction should be the reverse. Second, allowing each gene program to have a condition-specific bias term $h^{(n)}$ would cause the bias term to explain a great proportion of variance without any interpretable power.

To address these issues, we developed a framework where the conditional dependence is computed only in the direction of gene-to-program rather than program-to-gene. Each gene program is also limited to a single global bias h that remains fixed across conditions. With these constraints, the model takes on a regression form similar to the pseudolikelihood approach. Specifically, for program expression w and gene expression state s, the likelihood function of the

program state is

$$\log \mathcal{L}(\boldsymbol{j}, h) = \sum_{n}^{N} \sum_{w^{(n)}, \boldsymbol{s}_{\text{Data}}^{(n)}} \log P(w^{(n)} | \boldsymbol{s}_{\text{Data}}^{(n)}, \boldsymbol{j}, h) \tag{17}$$

$$= \sum_{n}^{N} \sum_{w^{(n)}, \boldsymbol{s}_{\text{Data}}^{(n)}} \frac{\exp \Theta w^{(n)}}{\exp \Theta + 1 + \exp(-\Theta)}, \tag{18}$$

where $\Theta = h + \sum_i \boldsymbol{s}_{i\text{Data}}^{(n)} \boldsymbol{j}_i$ is the effective field on the program created by single genes and the global bias h. The form is similar to Eq. 15, thus the same optimization procedures can be applied.

3.2 Identification of gene programs

Non-negative matrix factorization We apply orthogonal non-negative matrix factorization (oNMF) to identify gene programs because its mathematical constraints yield highly accurate, biologically interpretable approximations of single-cell transcriptional states while preventing overlap between programs. Compared to typical matrix factorization methods like principal components analysis (PCA), oNMF imposes two constraints on the gene programs: non-negative weights and orthogonality, both of which are essential for building interpretable interaction models.

1. Linearity and non-negativity: Each program is expressed as a linear combination of individual genes with strictly positive weights, ensuring that it captures a coherent set of co-expressed genes. Positive weights eliminate the ambiguity that would arise from interpreting negatively weighted genes across different programs.
2. Orthogonality: Enforcing orthogonality makes programs non-overlapping, so each can be interpreted as a distinct biological function, and each cell is represented as a collection of biological function activities. In the absence of orthogonality, a cell would be dominated by a single broad program encompassing all genes expressed by its type. Within D-SPIN, gene sharing between programs would introduce confounding interactions, so the independence conferred by orthogonality is essential.

Formally, given a non-negative gene matrix of C cells and M genes, $X \in \mathbb{R}^{C \times M}$ and a chosen program number K, oNMF solves the following optimization problem.

$$\begin{aligned} \underset{W,H}{\text{minimize}} \quad & \|X - WH\|_F^2 \\ \text{subject to} \quad & HH^\intercal = I, \quad W_{ij} \geq 0, \quad H_{ij} \geq 0 \end{aligned} \tag{19}$$

where $H \in \mathbb{R}^{K \times M}$ is the gene program representation, $W \in \mathbb{R}^{C \times K}$ is the cell state represented on the gene programs and $\| \cdot \|_F^2$ is the matrix Frobenius norm.

We implemented a published iterative algorithm proposed [21], with W and H randomly initialized by uniform distribution on $[0, 1]$. The random initialization of H is orthogonalized using singular value decomposition (SVD) and takes the absolute value before the start of the iteration.

Because the oNMF objective weights genes and cells according to their contribution to the reconstruction error, highly abundant cell states and highly expressed genes can dominate the fit. To balance the contribution from each gene and cell state and improve the quality of gene programs, we subsample the single cells to reduce the proportion of highly abundant cell states (*see* **Note 7**), and normalize the gene expression by the standard deviation of each gene (*see* **Note 8**).

Exact non-negative matrix factorization is an NP-hard problem [33]. The practical algorithms of oNMF rely on heuristic approximations of iterative matrix update, and have solutions that vary with random initialization. To enhance robustness, we repeat oNMF with multiple random seeds and derive a consensus decomposition. For example, we take each row of H (feature vector) across different random seed repeats, and perform K-means clustering to compute the consensus composition of each gene program. The consensus feature vector would be the average of all feature vectors in each cluster, and each gene is assigned to the consensus program with the largest weights.

Evaluating gene program expression Even though for oNMF evaluation, the cell state representation W is computed along with program composition H, we need to re-evaluate W after computing the consensus program because averaging between H of different random seeds would disrupt W. At this stage, we can also append programs derived from prior biological knowledge. We again solve the optimization problem Eq. 20, but with the extra information of the gene composition of every program. Mathematically, we have an extra constraint of knowing the nonzero entries of H.

Therefore, the objective function can be split by each program and optimized individually as

$$\|X - WH\|_F^2 = \sum_{k}^{K} \sum_{m|H_{km}>0} (X_{\cdot,m} - W_{\cdot,k} H_{km})^2 \,. \tag{20}$$

For every program k, we only need to consider genes m that are known to belong to the program ($H_{km} > 0$). For notation simplicity, we denote $X^{(k)}$ as the matrix of genes that are known to belong to program k, $W^{(k)}$ as the cell expression level on program k, and $H^{(k)}$ as the weights on genes in the program k. The optimization problem on program k is

$$\begin{aligned} \underset{W^{(k)}, H^{(k)}}{\text{minimize}} \quad & \|X^{(k)} - W^{(k)} H^{(k)}\|_F^2 \\ \text{subject to} \quad & \|H^{(k)}\| = 1, \quad W_{ij} \geq 0, \quad H_{ij} \geq 0 \end{aligned} \tag{21}$$

This is essentially a 1-component non-negative matrix factorization (NMF) problem and has mature implementations in the Python `scikit-learn` package.

3.3 Case study: dissecting the immune dictionary dataset with D-SPIN

Overview of the immune dictionary dataset To demonstrate the applications of D-SPIN, we apply the framework to a public immune dictionary dataset to dissect the action of cytokines on multiple immune cell types *in vivo* [24]. In the experiments, 86 different cytokines were injected into mice, and skin-draining lymph nodes were collected and profiled with single-cell RNA-seq 4 hours after injection. The full dataset is composed of 17 immune cell types with 386,703 single cells. For simplicity of the demonstration, we focus on the subsetted data of CD4, CD8, and regulatory T cells and 12 cytokines that were found to be capable of inducing specific polarization states of all three cell types [24]. The selected cytokines include type I/II interferons (IFN-α1, IFN-β, IFN-ϵ, IFN-κ, IFN-γ), IL-1 family (IL-1α, IL-1β, IL-18, IL-36α) and other interleukins (IL-2, IL-12, IL-15), each driving the cell population to distinct polarization states as rendered on the t-distributed stochastic neighbour (t-SNE) embedding (Fig. 2A).

Program-level network model partitions cytokine categories D-SPIN extracted 20 gene expression programs from 2,000 highly variable genes that capture the principal patterns of co-expression (Fig. 2B). These include broad cell-type signature programs composed of known gene markers such as P10-CD8 T signature (Nkg7, Cd8b1, Cd8a) and P15-Treg signature (Il2rb, Il2ra, Foxp3), as well as programs linked to specific cell polarization states and cytokine responses. For example, P18-Type I IFN response program is enriched for interferon-stimulated genes (ISGs) (Ifi203, Sp100, Isg15) that reflect direct response to the stimulation. Whereas P6-IL-1 response program contains genes directly induced by IL-1 (Stat3, Bcl3, Cebpb) together with multiple receptors of other cytokines (Ifngr1, Il2r, Il6st, Il4ra), indicating the complex *in vivo* crosstalks between signals and cell types.

The inferred program regulatory network exhibits a distinct modular structure with four network modules of tightly interacting gene programs (Fig. 3A). Interactions between programs in the same module are predominantly positive. Each module aligns with a major cell type in the population. For example, the module associated with regulatory T cells contains programs for Treg, effector Treg, T exhaustion, and NF-κB signaling. There are two modules of CD8 T cells; one is standard CD8 T functions, including cytotoxic and NK-like programs; the other is antivirus and interferon responses.

Together with the network, D-SPIN infers the interactions of the 12 cytokines with the regulatory network that drive the observed cell state transitions. These perturbation responses classify them into three groups: (i) Th1-polarizing cytokines (IL-2, IL-12, IL-15, IL-18, IFN-γ); (ii) IL-1 subfamily (IL-1α, IL-1β, IL-36α); and (iii) type I interferon (IFN-α1, IFN-β, IFN-ϵ, IFN-κ). The classification agrees with the cell polarization analysis of the original immune dictionary study [24]. Cytokines in the same group share dominant perturbation responses yet retain distinctive secondary signatures. For instance, all four type I interferons strongly activate P18-Type I IFN response program, while IFN-α1,

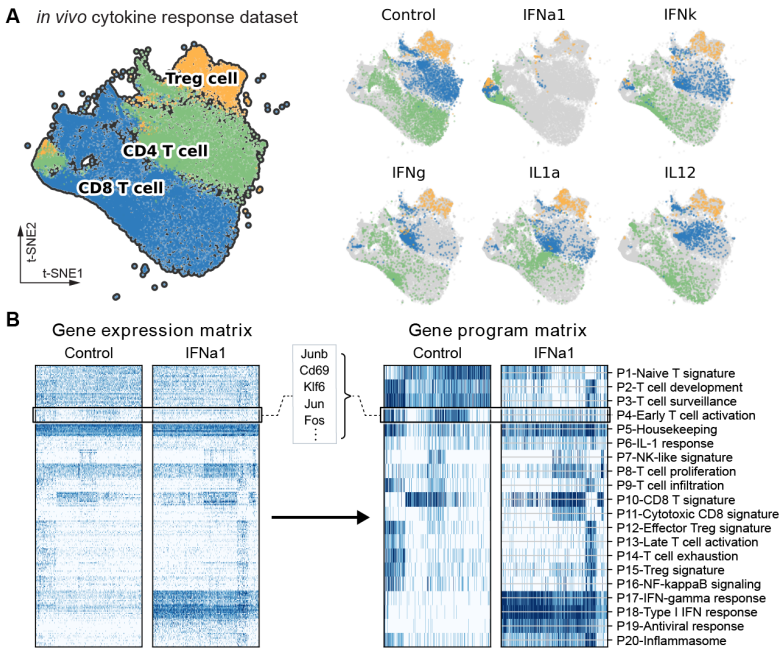

Fig. 2. Overview and program discovery for the immune dictionary dataset. (A) t-distributed stochastic neighbour (t-SNE) embedding for a subsetted cell population from the immune dictionary dataset [24]. The subset dataset includes CD4, CD8, and regulatory T cells treated by 12 different cytokines, as well as corresponding control samples treated by phosphate-buffered saline (PBS). (B) Heatmaps of gene expression and discretized gene program levels for control and IFN-α1-treated samples. The gene programs are weighted averages of single-gene expressions that characterize and denoise the major expression pattern of the gene matrix.

IFN-β additionally induce stronger P17-IFN-γ response and suppress P3-T cell surveillance. The different responses reflect the potency difference between the two subgroups. IFN-$\alpha 1/\beta$ are universally expressed in most cells in the body and can trigger global antivirus responses when stimulated by the pattern recognition receptors (PRRs) [34]. By contrast, the expressions of IFN-ϵ/κ are constitutively expressed at the mucosal and cutaneous barrier and signal with much lower efficacy due to reduced affinity for the receptor IFNAR2 [35].

Collectively, the program-regulatory network constructed by D-SPIN organizes the core cellular functions and cytokine response pathways into an interpretable diagram of interactions. By computing the interactions of each cytokine perturbation on this network, the D-SPIN model collapses the cytokine effect on distinct cell types, CD4, CD8, and regulatory T cells, into a single vector of coefficients on gene programs. The response vector provides a fingerprint for each cytokine that registers how the perturbation shifts the entire landscape of

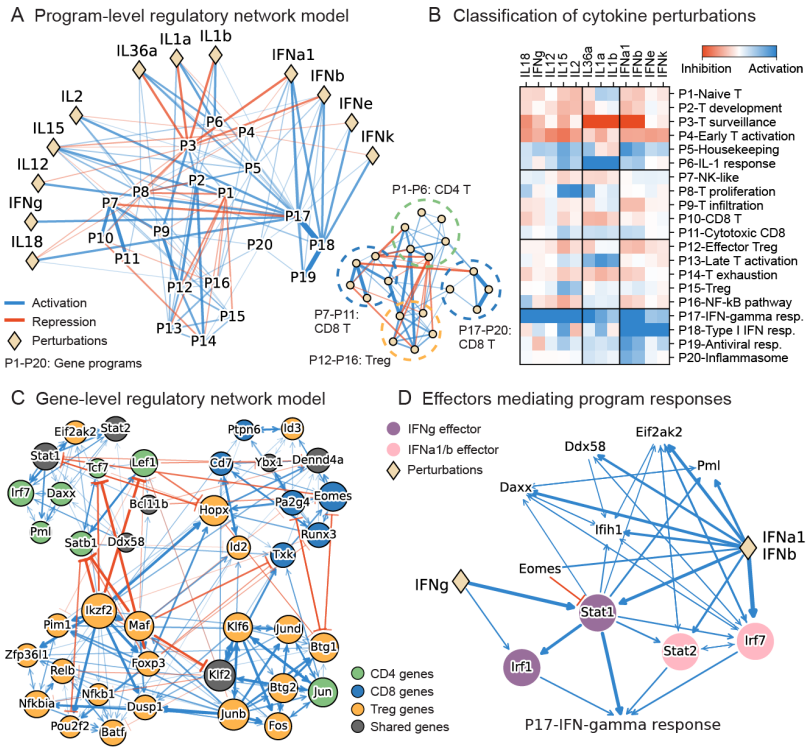

Fig. 3. Program/gene-level regulatory network inferred by D-SPIN. (A) Diagram of D-SPIN-inferred network model on gene programs. The network is partitioned into 4 modules, each associated with a T cell type in the population. (B) Heatmap of the program response of each cytokine. Clustering of the response partitioned the cytokines into 3 major categories. (C) Diagram of the core subnetwork of the D-SPIN-inferred gene network model. The node sizes scale with the number of identified interactions. The network is partitioned into 4 modules, each primarily composed of genes that have elevated expression in one specific T cell type. (D) Interaction diagram for the subnetwork of IFN-α1/IFN-β and IFN-γ acting on the program P18-IFN-γ response. D-SPIN model shows that Type I and Type II interferons have different effectors to activate the program.

T cell states. These unified fingerprints of responses provide systematic classifications of cytokine actions and reveal the different signatures of each cytokine associated with their physiological roles.

Gene-level network model identifies key effectors of interferon responses To identify gene-level regulators of cytokine actions and gain a fine-grained view of the regulatory network controlling cytokine signaling, we also construct a gene-level network model with D-SPIN. Upon cytokine stimulation, regulation is carried out through both DNA-binding interactions of transcription

factors (TFs) and signal transduction cascades with kinases and phosphatases. Therefore, we selected 115 highly expressed regulatory genes of 91 TFs, 14 kinases, and 10 phosphatases for the network construction.

Among all these regulatory genes, D-SPIN identifies a core interaction network of 41 genes, highlighting these signaling hubs during the cytokine response (Fig. 2C). The core subnetwork is partitioned into four modules, where three modules contain genes highly expressed in CD4, CD8, and regulatory T cells each, and one module of genes involved in the immune response of multiple cell types. The top signaling hub genes with the highest number of interactions participate in multiple core immune functions, including early signal response by AP-1 (Junb, Jun), cell lineage maintenance (Ikzf2, Hopx), differentiation (Klf2, Klf6), and T cell quiescence (Btg1/2).

Furthermore, D-SPIN identifies regulators for each gene program to couple them to the gene-level regulatory network, enabling the model to nominate the key effectors for the observed program-level responses. As an example, both type I interferon IFN-$\alpha 1/\beta$ and type II interferon IFN-γ induce the activation of P17-IFN-γ response in the *in vivo* experiment. The circuit model inferred by D-SPIN reveals the different pathways of action for the two types of cytokines (Fig. 2D). The circuit of IFN-γ action is straightforward, with only Stat1 and Irf1. Stat1 is the most important TF mediating transcription response of IFN-γ [36]. Irf1 is also preferentially induced by IFN-γ and participates in its signal transduction [36, 37]. On the contrary, apart from the core effector Stat1, the effect of IFN-$\alpha 1/\beta$ is mediated by Stat2 and Irf7. Stat2 is a core component of type I interferon signaling by forming the ISGF3 complex [36], and Irf7 is the master regulator in immune response of type I IFN signaling [38]. Moreover, the circuit of IFN-$\alpha 1/\beta$ action contains many secondary regulators, indicating that the response is a consequence of cell-cell communications through the secretion of other cytokines.

Overall, the gene-level D-SPIN network extracts the core signaling network, exposes cell-type-specific hubs, and dissects convergent program responses into cytokine-specific circuit models. By associating program response to the gene regulatory network, D-SPIN helps to clarify how type I and type II interferons produce overlapping transcriptional signatures while relying on distinct mechanisms of action, therefore nominating key regulatory nodes for modulation of the T cell cytokine responses.

3.4 Building network models with the D-SPIN package

Gene-level regulatory network models Here we present the detailed steps of constructing gene-level regulatory network models with the D-SPIN package. For the input `AnnData` object `adata` prepared as described, the basic steps include

```
from dspin.dspin import DSPIN

model = DSPIN(adata, save_path, num_spin=adata.shape[1])
model.network_inference(sample_id_key='sample_id',
    method='pseudo_likelihood', directed=True,
    sample_list=None, perturb_matrix=None,
    prior_network=None, run_with_matlab=False,
    params={'stepsz': 0.05, 'lam_l1_j': 0.01})
model.response_relative_to_control(sample_id_key='sample_id',
    if_control_key='if_control', batch_key='batch')
```

The computed regulatory network model is stored in the `model` object:

- `model.network`: regulatory network $\boldsymbol{J}$
- `model.responses`: gene-level responses $\boldsymbol{h}^{(n)}$
- `model.relative_responses`: response relative to control. The relative responses are preferably compared with the control sample of each experimental batch.

A few remarks regarding the input parameters in the gene-level network construction

- `num_spin` set the number of elements in the network. Therefore, setting `num_spin=adata.shape[1]` will construct gene-level networks.
- `directed` set whether to infer a directed network. Note that only the pseudolikelihood method supports directed network inference.
- `prior_network` (optional) is a binary matrix indicating more likely network edges obtained from prior knowledge, such as TF-motif interactions or chromatin accessibility.
- `sample_list` (optional): List of ordered experimental conditions. The list is used for only including a subset of conditions in the model, or specifying the order of samples for input of the perturbation matrix.
- `perturb_matrix` (optional): Putative effect of each perturbation if known. The matrix has the shape of (sample number) * (gene number). Due to the different types and complicated actions of perturbations, prior knowledge of the perturbation effect may not be available. In cases where the perturbations have a well-defined effect, such as Perturb-seq, we use the matrix as a prior to bias the inferred perturbation effect.
- `run_with_matlab` is for large datasets where parallelization is needed to speed up. When set to `True`, the package will not perform inference, but write organized variables under `save_path`, so that the MATLAB implementation deposited in GitHub can take over.
- `params` is a dictionary of hyperparameters for the optimization. For example, the step size of the gradient ascent, the number of epochs for training, and ℓ_1 or ℓ_2 regularization of the model parameters.

Program-level regulatory network models The steps for program-level network models are mostly similar but with an extra step of gene program discovery.

```
from dspin.dspin import DSPIN

model = DSPIN(adata, save_path, num_spin=15)
model.gene_program_discovery(num_repeat=10, seed=0,
    cluster_key='cell_type')
model.network_inference(sample_id_key='sample_id',
    method='mcmc_maximum_likelihood')
model.response_relative_to_control()
```

Apart from the computed network $\boldsymbol{J}$ and responses $\boldsymbol{h}$, program-level D-SPIN `model` object also contains information about the identified gene programs. Results of individual oNMF runs are saved under `save_path/onmf`. The gene compositions of the consensus programs are saved under `save_path`.

A few remarks regarding the input parameters of the program-level network construction

- `num_spin` set the number of programs. Although a variety of methods have been suggested for choosing the optimal number of clusters in dimensional reductions, there is no universally accepted standard for achieving the best results. In practice, we recommend setting `num_spin` to roughly five times the number of cell types or clusters in the dataset, but not exceeding 40 programs (*see* **Note 9**).
- `cluster_key` set the cluster label for balancing proportions of different cell types. The package downsamples highly abundant cell types to avoid overfitting to their states (*see* **Note 7**).

Identifying gene regulators of programs For a program-network D-SPIN object `model_program` and a gene-level D-SPIN object `model_gene`, the following code computes the gene regulators in the gene-level network for each program.

```
from dspin.dspin import DSPIN

model_gene.program_regulator_discovery(
    model_program.program_representation,
    sample_id_key='sample_id',
    params={'stepsz': 0.02, 'lam_l1_interaction': 0.01})
```

Note that the `adata` objects corresponding to `model_program` and `model_gene` should have the same cell metadata `adata.obs`, so that the gene expression state and program-expression state of each cell are aligned. The computed interactions between genes and programs are stored in the `model_gene` object:

- `model_gene.program_interactions`: regression coefficients $\boldsymbol{j}$ between genes and programs.
- `model_gene.program_activities`: the global activity h for each program.

Module discovery and visualization of network models With the inferred network, D-SPIN also includes functions for network module discovery and visualizations.

```
model.compute_modules(resolution=1)

from dspin.plot import plot_module_heatmap, plot_network_diagram

plot_module_heatmap(model.network, model.modules,
    spin_name_list=model.name_list)
plot_network_diagram(model.network, model.modules, pos=None,
    spin_name_list_short=model.name_list_short,
    spin_name_list=model.name_list)
```

Modules are identified through Leiden clustering, and `resolution` controls the granularity of detected modules. For program-level models, `spin_name_list` are the names of gene programs manually specified or model-identified representative gene names in the program. `spin_name_list_short` are index-like short names in graph rendering. For gene-level models, both name lists will be simply gene names.

4 Notes

1. Rigorously speaking, the model contains an $\boldsymbol{h}^{(\text{control})}$ as the activity of individual nodes without the perturbation. The perturbation modulates the node activity with an extra term $\boldsymbol{h}^{(\text{perturbation})}$. As the energy $E = -\sum_{ij} J_{ij} s_i s_j - \sum_i h_i s_i$ is a linear function of $\boldsymbol{h}$, $\boldsymbol{h}^{(\text{control})}$ and $\boldsymbol{h}^{(\text{perturbation})}$ are combined into notation $\boldsymbol{h}^{(n)}$ together for simplicity. Therefore for each perturbation, $\boldsymbol{h}^{(n)}$ is to be compared with the control condition to obtain the relative response or net response under the perturbation condition.
2. We discretize the expression into 3 levels with the form $[-1, 0, 1]$ due to the following considerations
   (a) A Higher number of levels raises computational complexity and makes the model behave more like a Gaussian distribution with a single probability peak, which cannot capture cell populations with multiple distinct cell types.
   (b) A simple on-off discretization makes it difficult to capture different levels of responses or dosage effects.
   (c) The $[-1, 0, 1]$ scheme create interpretable self-interaction J_{ii}. With $J_{ii} > 0$, the node prefers -1 or 1 state, similar to a bistable switch. With $J_{ii} < 0$, the node prefers the 0 state, similar to negative feedback.
3. Various methods have been empirically used to partition continuous expression into discrete levels, such as using percentiles or standard deviations. Discretization with K-means was found to perform well [39] and have a clear interpretation of minimizing the variance inside each expression level category.

In practice, sometimes the gene expression has long tails, which means a small proportion of cells have very high expressions. Clipping each gene expression by its percentiles, such as the 95th percentile will help K-means to better characterize the distribution of expression.

4. The phenomenon of indistinguishable models refers to suboptimal solutions with nearly identical objective values but differ in parameters. The existence of suboptimal solutions can be quantified by Fisher information, which characterizes the curvature in the space of a parametric family of probabilities. Suboptimal solutions correspond to the linear subspaces with small Fisher information eigenvalues. We found in separate research that different perturbations correspond to different subspaces with suboptimal solutions, and combining information between perturbation conditions enables eliminating these suboptimal solutions and achieving accurate network inference [28].
5. The acyclic constraint of directed probabilistic graphical models is fundamental, as cycles in the conditional dependence between variables will produce inconsistent distributions. As mentioned in the seminal work of Judea Pearl on causal inference [40], even three cyclic dependent variables "will normally lead to inconsistencies", and the acyclic constraint can ensure consistency of the distributions. Constructing distributions on even two mutually dependent variables requires nontrivial constraints [41].
6. For any distribution $P(\boldsymbol{x})$, we can construct a stochastic differential equation (SDE) whose stationary distribution is $P(\boldsymbol{x})$ and that only contains symmetric interaction between variables. Therefore, if the single-cell data is sampled from a stationary distribution, in principle, an undirected model will be sufficient to explain the data.
   The sketch of such SDE construction is as follows. The distribution can be written in the form of an exponential family $P(\boldsymbol{x}) = \frac{1}{Z} \exp -2V(\boldsymbol{x})/\sigma^2$. The distribution is the steady-state solution of the Fokker-Planck equation

$$\frac{\partial P(\mathbf{x}, t)}{\partial t} = -\sum_{i=1}^{N} \frac{\partial}{\partial x_i} \left[-\frac{\partial V(\boldsymbol{x})}{\partial x_i} P(\mathbf{x}, t) \right] + \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\partial^2}{\partial x_i \, \partial x_j} \left[\frac{\sigma^2}{2} P(\mathbf{x}, t) \right] ,$$

   which corresponds to the SDE of

$$d\mathbf{X}_t = -\nabla V(X) dt + \sigma d\mathbf{W}_t \, .$$

   The SDE always has symmetric interactions between variables, as the driving force is the gradient of a potential function.
7. The gene program discovery may overfit to cell states that are over-represented in the cell population. We use two class-balancing strategies from machine learning by sub-sampling cells according to their clusters obtained by clustering or cell-typing.
   (a) Equal-sample balancing. We sample an equal number of cells from each cluster to facilitate identifying programs that only express in a small subpopulation.

(b) Square-root balancing. We sample each cluster in proportion to the square root of its size, an empirically effective technique for reducing the proportion of large clusters.

8. Similar to standardizing variables in PCA, we divide the expression of each gene by its standard deviation. Low-expression genes with only a few nonzero entries may become disproportionately large after the normalization. To mitigate this, the standard deviation can be lower clipped by percentiles of the standard deviation of all genes, for example, the 20th percentile.
9. The selection of the number of gene programs is a trade-off between model expressive power, computational complexity, and model interpretability. The optimal choice of program number typically hinges on the specific requirements of the application scenario. Apart from estimating the program number by the number of cell types, the elbow method also provides a heuristic approach. This method involves plotting a relevant cost function or objective function against the number of gene programs and looking for a point in the plot where the rate of change drastically alters, resembling an "elbow". For oNMF, the cost function could be the reconstruction error by the gene programs. A special note is that as the single-cell data is typically noisy, the gene matrix should be denoised with methods such as data diffusion [42] before computing the reconstruction error.

References

[1] Bray D (1995) Protein molecules as computational elements in living cells. Nature 376(6538):307–312

[2] Regev A, Shapiro E (2002) Cellular abstractions: Cells as computation. Nature 419(6905):343–343

[3] McGinnis CS, Patterson DM, Winkler J, Conrad DN, Hein MY, Srivastava V, Hu JL, Murrow LM, Weissman JS, Werb Z, et al (2019) Multi-seq: sample multiplexing for single-cell rna sequencing using lipid-tagged indices. Nature methods 16(7):619–626

[4] Dixit A, Parnas O, Li B, Chen J, Fulco CP, Jerby-Arnon L, Marjanovic ND, Dionne D, Burks T, Raychowdhury R, et al (2016) Perturb-seq: dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. Cell 167(7):1853–1866

[5] Replogle JM, Saunders RA, Pogson AN, Hussmann JA, Lenail A, Guna A, Mascibroda L, Wagner EJ, Adelman K, Lithwick-Yanai G, et al (2022) Mapping information-rich genotype-phenotype landscapes with genome-scale perturb-seq. Cell

[6] Gehring J, Hwee Park J, Chen S, Thomson M, Pachter L (2020) Highly multiplexed single-cell rna-seq by dna oligonucleotide tagging of cellular proteins. Nature Biotechnology 38(1):35–38

[7] Tegnér J, Björkegren J (2007) Perturbations to uncover gene networks. TRENDS in Genetics 23(1):34–41

[8] Seçilmiş D, Hillerton T, Tjärnberg A, Nelander S, Nordling TE, Sonnhammer EL (2022) Knowledge of the perturbation design is essential for accurate gene regulatory network inference. Scientific reports 12(1):16531
[9] Jiang J, Chen S, Tsou T, McGinnis CS, Khazaei T, Zhu Q, Park JH, Strazhnik IM, Vielmetter J, Gong Y, et al (2024) D-spin constructs gene regulatory network models from multiplexed scrna-seq data revealing organizing principles of cellular perturbation response. BioRxiv pp 2023–04
[10] Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences 79(8):2554–2558
[11] Jaynes ET (1957) Information theory and statistical mechanics. Physical review 106(4):620
[12] Schneidman E, Berry MJ, Segev R, Bialek W (2006) Weak pairwise correlations imply strongly correlated network states in a neural population. Nature 440(7087):1007–1012
[13] Lang AH, Li H, Collins JJ, Mehta P (2014) Epigenetic landscapes explain partially reprogrammed cells and identify key reprogramming genes. PLoS computational biology 10(8):e1003734
[14] Teschendorff AE, Feinberg AP (2021) Statistical mechanics meets single-cell biology. Nature Reviews Genetics 22(7):459–476
[15] Nguyen HC, Zecchina R, Berg J (2017) Inverse statistical problems: from the inverse ising problem to data science. Advances in Physics 66(3):197–261
[16] Aurell E, Ekeberg M (2012) Inverse ising inference using all the data. Physical review letters 108(9):090201
[17] Ravikumar P, Wainwright MJ, Lafferty JD (2010) High-dimensional ising model selection using ℓ_1-regularized logistic regression. The Annals of Statistics pp 1287–1319
[18] Pratapa A, Jalihal AP, Law JN, Bharadwaj A, Murali T (2020) Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. Nature methods 17(2):147–154
[19] Roguev A, Bandyopadhyay S, Zofall M, Zhang K, Fischer T, Collins SR, Qu H, Shales M, Park HO, Hayles J, et al (2008) Conservation and rewiring of functional modules revealed by an epistasis map in fission yeast. Science 322(5900):405–410
[20] Segal E, Shapira M, Regev A, Pe'er D, Botstein D, Koller D, Friedman N (2003) Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. Nature genetics 34(2):166–176
[21] Choi S (2008) Algorithms for orthogonal nonnegative matrix factorization. In: 2008 ieee international joint conference on neural networks (ieee world congress on computational intelligence), IEEE, pp 1828–1832
[22] Heimberg G, Bhatnagar R, El-Samad H, Thomson M (2016) Low dimensionality in gene expression data enables the accurate extraction of transcriptional programs from shallow sequencing. Cell systems 2(4):239–250
[23] Chen S, Rivaud P, Park JH, Tsou T, Charles E, Haliburton JR, Pichiorri F, Thomson M (2020) Dissecting heterogeneous cell populations across drug

and disease conditions with popalign. Proceedings of the National Academy of Sciences 117(46):28784–28794

[24] Cui A, Huang T, Li S, Ma A, Pérez JL, Sander C, Keskin DB, Wu CJ, Fraenkel E, Hacohen N (2024) Dictionary of immune responses to cytokines at single-cell resolution. Nature 625(7994):377–384

[25] Bialek W, Cavagna A, Giardina I, Mora T, Silvestri E, Viale M, Walczak AM (2012) Statistical mechanics for natural flocks of birds. Proceedings of the National Academy of Sciences 109(13):4786–4791

[26] Friedman N, Linial M, Nachman I, Pe'er D (2000) Using bayesian networks to analyze expression data. In: Proceedings of the fourth annual international conference on Computational molecular biology, pp 127–135

[27] Machta BB, Chachra R, Transtrum MK, Sethna JP (2013) Parameter space compression underlies emergent theories and predictive models. Science 342(6158):604–607

[28] Jiang J, Sivak DA, Thomson M (2019) Active learning of spin network models. arXiv preprint arXiv:190310474

[29] Besag J (1974) Spatial interaction and the statistical analysis of lattice systems. Journal of the Royal Statistical Society: Series B (Methodological) 36(2):192–225

[30] Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P (2010) Inferring regulatory networks from expression data using tree-based methods. PloS one 5(9):e12776

[31] Moerman T, Aibar Santos S, Bravo González-Blas C, Simm J, Moreau Y, Aerts J, Aerts S (2019) Grnboost2 and arboreto: efficient and scalable inference of gene regulatory networks. Bioinformatics 35(12):2159–2161

[32] Glymour C, Zhang K, Spirtes P (2019) Review of causal discovery methods based on graphical models. Frontiers in genetics 10:524

[33] Vavasis SA (2010) On the complexity of nonnegative matrix factorization. SIAM Journal on Optimization 20(3):1364–1377

[34] McNab F, Mayer-Barber K, Sher A, Wack A, O'garra A (2015) Type i interferons in infectious disease. Nature Reviews Immunology 15(2):87–103

[35] Harris BD, Schreiter J, Chevrier M, Jordan JL, Walter MR (2018) Human interferon-ϵ and interferon-κ exhibit low potency and low affinity for cell-surface ifnar and the poxvirus antagonist b18r. Journal of Biological Chemistry 293(41):16057–16068

[36] Platanias LC (2005) Mechanisms of type-i-and type-ii-interferon-mediated signalling. Nature Reviews Immunology 5(5):375–386

[37] Negishi H, Fujita Y, Yanai H, Sakaguchi S, Ouyang X, Shinohara M, Takayanagi H, Ohba Y, Taniguchi T, Honda K (2006) Evidence for licensing of ifn-γ-induced ifn regulatory factor 1 transcription factor by myd88 in toll-like receptor-dependent gene induction program. Proceedings of the National Academy of Sciences 103(41):15136–15141

[38] Honda K, Yanai H, Negishi H, Asagiri M, Sato M, Mizutani T, Shimada N, Ohba Y, Takaoka A, Yoshida N, et al (2005) Irf-7 is the master regulator of type-i interferon-dependent immune responses. Nature 434(7034):772–777

[39] Pe'er D, Regev A, Elidan G, Friedman N (2001) Inferring subnetworks from perturbed expression profiles. BIOINFORMATICS-OXFORD- 17:S215–S224

[40] Pearl J (2022) Fusion, propagation, and structuring in belief networks. In: Probabilistic and Causal Inference: The Works of Judea Pearl, ACM, pp 139–188

[41] Arnold BC, Press SJ (1989) Compatible conditional distributions. Journal of the American Statistical Association 84(405):152–156

[42] Van Dijk D, Sharma R, Nainys J, Yim K, Kathail P, Carr AJ, Burdziak C, Moon KR, Chaffer CL, Pattabiraman D, et al (2018) Recovering gene interactions from single-cell data using data diffusion. Cell 174(3):716–729