Generative Models for scRNA

Jackson Loper

April 2, 2018

Status Succesfully trained the new "iterative model," evaluated it to some extent

Todo Compare clusters against Allen, consider linear model, do more layers

1 Overview

The Allen Institute has developed a method for clustering, one that has apparently allowed them to distinguish 116 unique cell types using single cell RNA data. The method follows the following recursive procedure:

Algorithm 1: itclust

Data: RNA expression for a collection of cells

```
Run PCA on the data (e.g. take first 16 components);
Run T-SNE on the resulting low-d representation;
Circle things that look like clusters;
if there is more than one cluster then
    for each cluster c do
        run itclust on the cells inside c;
    end
end
```

This method is nice because the important low-dimensional representations *within* a cluster may be quite different from the ones that help you distinguish *between* two subclusters within that cluster. This is what allows them to achieve the level of granularity that they have.

Unfortunately, it is a bit difficult to grasp the uncertainty involved in the process. There is certainly noise in the data, and different practitioners of this basic procedure may have different understandings of what this noise is. The result is that when Bosilijka Tasic (one of the lead experimentalists at Allen) gives two data scientists the same data, they each come back with different answers. And they have no way to compare or evaluate these two different answers. One essential question is "what level of granularity does the signal-to-noise ratio support?" The algorithm above, as stated, runs iteratively until subclusters cannot be distinguished via T-SNE. But this is obviously a very subjective matter. What to do?

We propose to help clarify some of these issues by building a fully generative model of gene expression, one that rigorously measures some kinds of uncertainty in every step of the process. To be most useful to the Allen Institute, however, we also want this generative model to fit into the scientific paradigm under which they are operating. In particular, the treatment of clusters should be hierarchical in nature, and it is desirable that the model

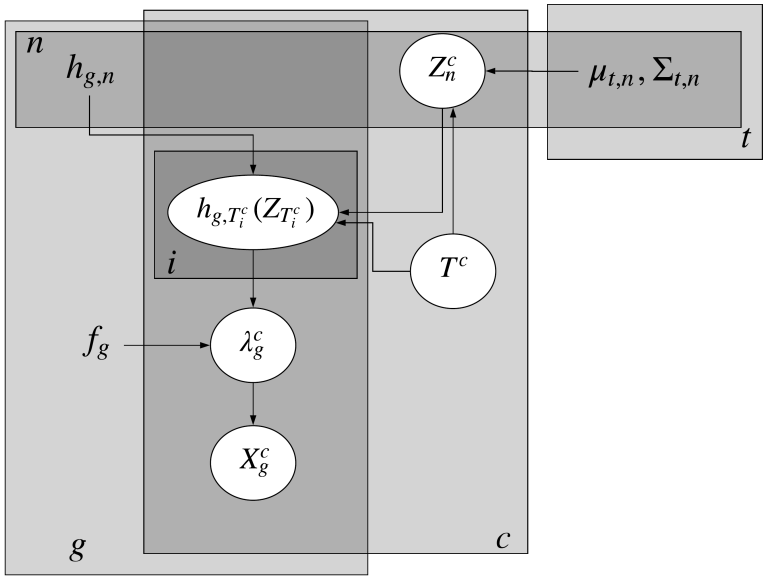


Figure 1: Plate diagram for the model. i indexes layers of the hierarchy of cell types, n indexes nodes in the hierarchy of cell types, c indexes cells, and g indexes genes.

have latent representations that roughly correspond to the low-dimensional representations found in the Allen algorithm.

2 Model

Towards this end, we propose the following generative model for gene expression of a single cell. We will denote the expression of gene g in cell c by X_g^c. Let H denote a tree of cell types.

1. $T^c \sim \text{Categorical}(\pi)$, one of the leaves of H. Let $t_0, t_1(T^c), t_2(T^c), \cdots, t_\ell^c(T^c)$, indicate the nodes in the unique path in H which leads from the root to the leaf T. Here t_0 is the root of the tree, ℓ^c is the depth of the leaf T^c, and $t_\ell^c(T^c) = T^c$. To consolidate notation, we take $T_i^c \triangleq t_i(T^c)$.
2. For each $i < \ell$ let $Z_{T_i^c}^c \sim \mathcal{N}(\mu_{T_{i+1}^c}, \Sigma_{T_{i+1}^c})$, and take $Z_{T_\ell^c}^c \sim \mathcal{N}(0, I)$
3. For each gene g, let $\lambda_g^c \in \mathbb{R}^p$ denote a set of parameters. These will be a deterministic function of the Zs:
$$\lambda_g^c = f_g\left(\sum_{i=0}^{\ell} h_{g,T_i^c}(Z_{T_i^c}^c)\right)$$
4. Conditioned on Z^c we will have $X_g^c \sim p(x_g; \lambda_g^c)$.

This can be a bit confusing to take in. The plate diagram in Figure 1 may help. We can also try to unpack some of this in words:

- The latent representation Z_i^c corresponds to features of that cell which are specific to all the cells which are descendents of T_i^c.

- The the cell's subcluster T^c_{i+1} effects the distribution of its latent features at the ith level, Z^c_i. That is, the latent distribution on the ith latent representation is governed by what subcluster it is part of within T^c_i.
- The distribution of the final latent representation, Z^c_T, is simply a standard normal.
- The functions h_{g,T^c_i} indicates how these cell-type-specific features alter the distribution on the gene expressions for cells in that branch of the tree.

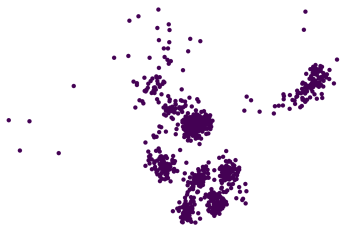
For inference, we use amortized variational inference. In particular, we take the variational family that T and all the Z_{T_i} are independent, T is some categorical, and Zs are isotropic gaussians. We build a neural network to estimate the variational parameters from gene expressions. Note that this network must estimate the parameters for a gaussian at *each node* in the tree H.

3 Empirical results

At the current moment, we have only actually implemented this model in the case the H is a tree with depth 2 (i.e. one root node and some children). We take the data model to be a zero-inflated negative binomial. It looks fairly promising. For a simple qualitative assessment, consider the following.

Scatterplot of the aggregate posterior distribution of Z_{T_0}, colored by posterior mean of the cluster assigment. That is, each dot is a cell, its position is given by a posterior sample of Z_{T_0}, and its color has something to do with what cluster we expect it to be in.

Notice that there are only so many clusters that are really clearly distinct in this latent space. However, let's focus on that cluster on the right-hand side. What happens if we look at the Z_{T_1} variables for those cells? We get substantial subclustering apparent in the Z_{T_1} space. In the following picture, each dot is a cell from the cluster on the right-hand-side of the previous figure, and the position is given by a posterior sample of Z_{T_1}.



3.1 Held out log likelihood

To quantitatively analyse our method, we used a held out log likelihood:

- Train the method on 75% of the cells
- For each of the remaining 25% of the cells, select 75% of the genes ("observed"):
  - Compute the posterior mode of Z given the expression of those genes
  - Compute the log probability of the observed genes given the posterior mode
  - Compute the log probability of the unobserved genes given the posterior mode
- Average values over those 25% of the cells

Here are the results for this "Posterior mode predictive (PMP)" approach:

Method	PMP, observed	PMP, unobserved
Factor analysis $k = 16$	-4.2	-4.2
Factor analysis $k = 64$	-4.1	-4.3
HNBPF $k = 2$	-5.7	-5.7
HNBPF $k = 16$	-3.8	-3.9
HNBPF $k = 64$	-3.8	-5.7
ZINB-Vae $k = 2$	-3.6	-3.6
Smooth ZINB-Vae $k = 2$	-3.6	-3.6
Smooth Hierarchical ZINB $k = 2 + 2$	-3.5	-3.5

Here HNBPF is a negative-binomial-poisson-factorization approach, ZINB-Vae is based on a Michael Jordan paper, k is the number of latent dimensions. The results are certainly not mind-blowing, but its nice to see that we're at least not doing worse.

To do posterior inference on held out likelihood, our approach is to first set all of the "unobserved" entries to the average value for the corresponding genes, and apply our inference network unchanged to get estimates for the variational parameters. We then refine these estimates with gradient descent. We note that if we use completely random guesses to initialize the gradient descent, we often converge to inferior local minima; this shows the benefit of the amortized inference.

3.2 Posterior predictive samples

One limitation of the method as we have implemented it is that the posterior predictive distribution seems to be substantially underfitting. To see what we mean, consider Figure 2:

Here we four rows and three columns of plots. Each row corresponds to a different gene. Each dot in each plot corresponds to a cell. The position of the dot indicates the latent representation of the cell, as inferred by posterior inference. The color indicates the amount of expression for the particular gene for that cell, measured three ways:

1. The first column shows the true expression of the appropriate gene for each cell (i.e the color of each dot is the gene expression for that cell)
2. The second column shows a posterior predictive sample of the gene expression for each cell
3. The third column shows the mean of such samples for each cell

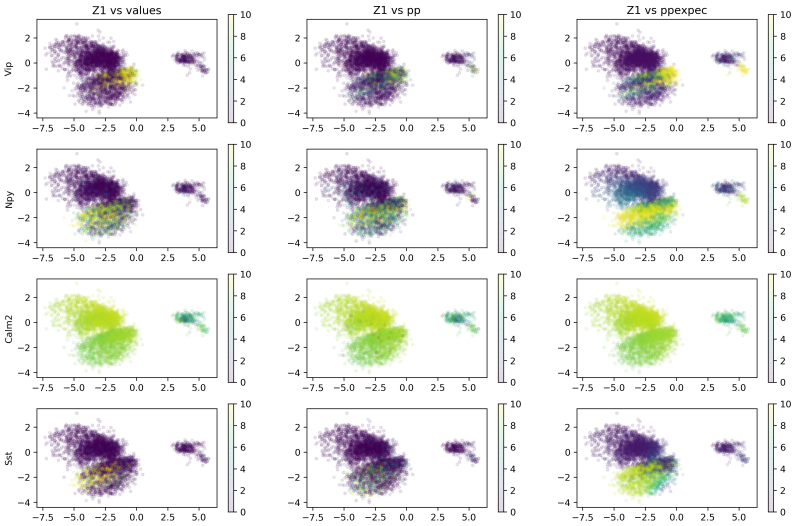


Figure 2: Colors indicate gene expression level.

We see that the true expression doesn't match the posterior predictive expression for the Vip and Sst genes as well as we might like. Something better is needed. Possibly more layers. Possibly more training. Worth thinking about.

Another way to see that the model isn't quite right is to look at posterior predictive p values. That is, for each entry of the matrix, we can compute the cumulative distribution function of that entry in light of the parameters predicted by the posterior mode. In Figure 3 we have done that for the first 1000 entries.

These results are actually fairly encouraging. Mostly it looks basically like white noise. However, we do see that for some genes the p values are almost always near 1: these are genes which express essentially never, and so have a high chance of turning off, and so their p value is nearly 1 whenever they are off, which is virtually always. If we restrict ourselves to those genes whose average probability of going off is less than 20%, we obtain Figure 4. We do this in Figure 5.

Another way to dig deeper into the underfitting question is by looking at individual genes, and comparing the expected value for that gene under the model with the actual value. There are clearly some systematic differences, as we see in Figure 6.

One potential reason for this systematic error might be that the negative binomial data distribution is inappropriate. To diagnose this, we fitted a zero-inflated negative binomial distribution to each collection of cells Allen Institute dubbed a "cluster." We then measured the goodness-of-fit, in terms of whether various moments matched. In particular, we compared the empirical mean, standard deviation, and proportion of zeros to the corresponding statistics which would be predicted by the learned models. We show this in Figure 7 and 8.

In terms of mean and zero proportion, these seem nearly okay, but in terms of variance we can see that we are not doing the greatest job. For a great many cell types we are predicting standard deviations which are much higher than the reality. This is perhaps not surprising, since the negative binomial model is inherently overdispersed, and simply cannot accomodate underdispersed data.

Another possible source of trouble is that in the particular model we use we have made certain assumption about how α, β, h can vary. This could be a big problem. To see why, we looked at each the fitted parameters of the zero-inflated negative binomial distribution for

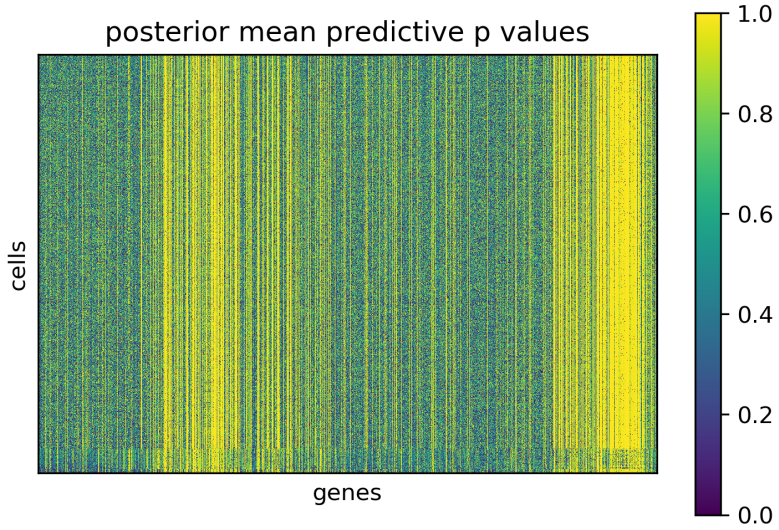


Figure 3: A p-value residual plot.

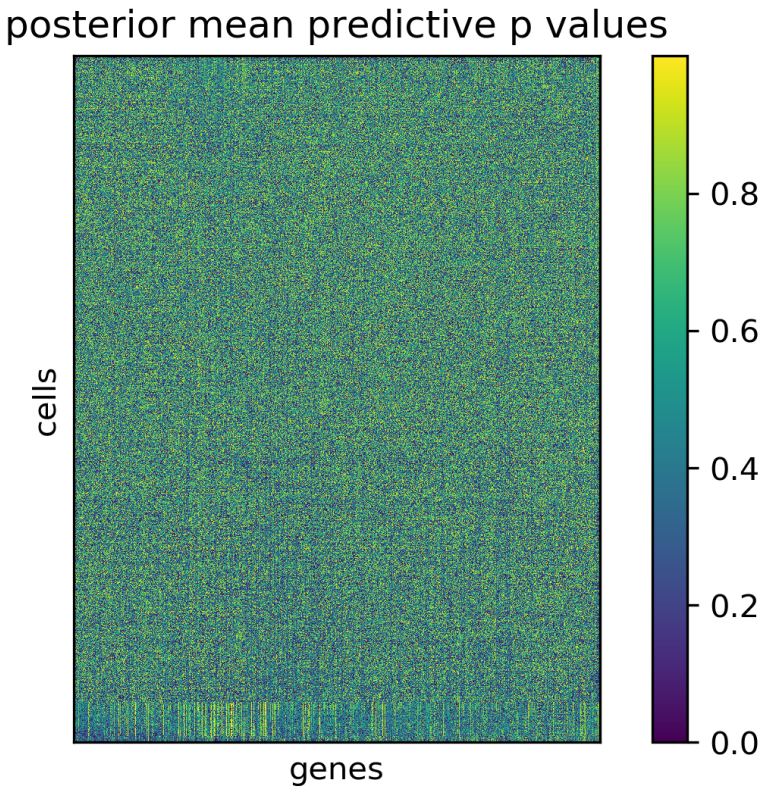


Figure 4: If we include only genes that express at least 80% of the time, the p-value residual plot looks less structured.

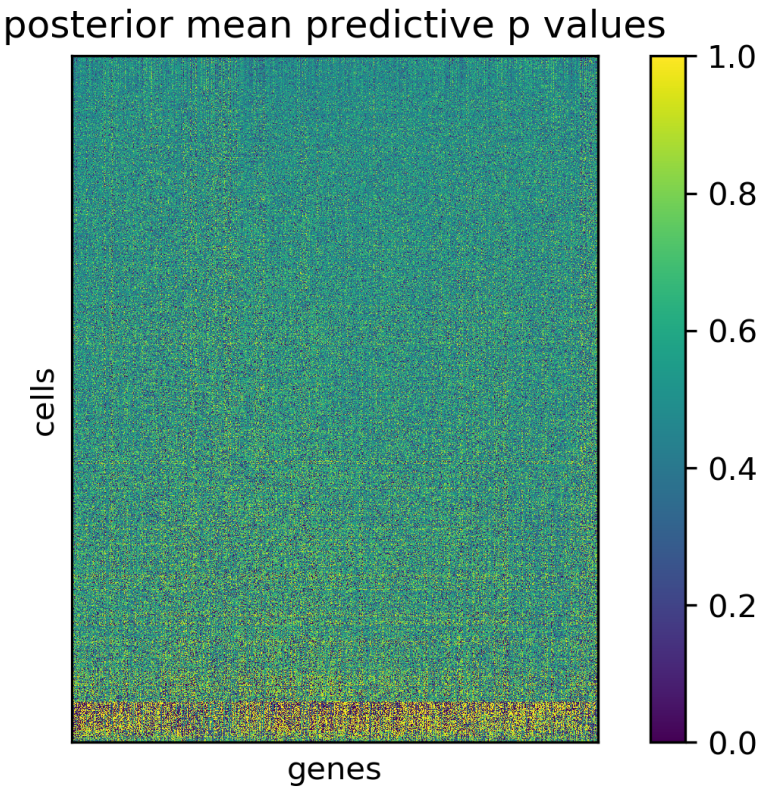


Figure 5: Using Factor Analysis for our model (with 16 factors), we see more significant underfitting. (Here we include only genes that express at least 80% of the time)

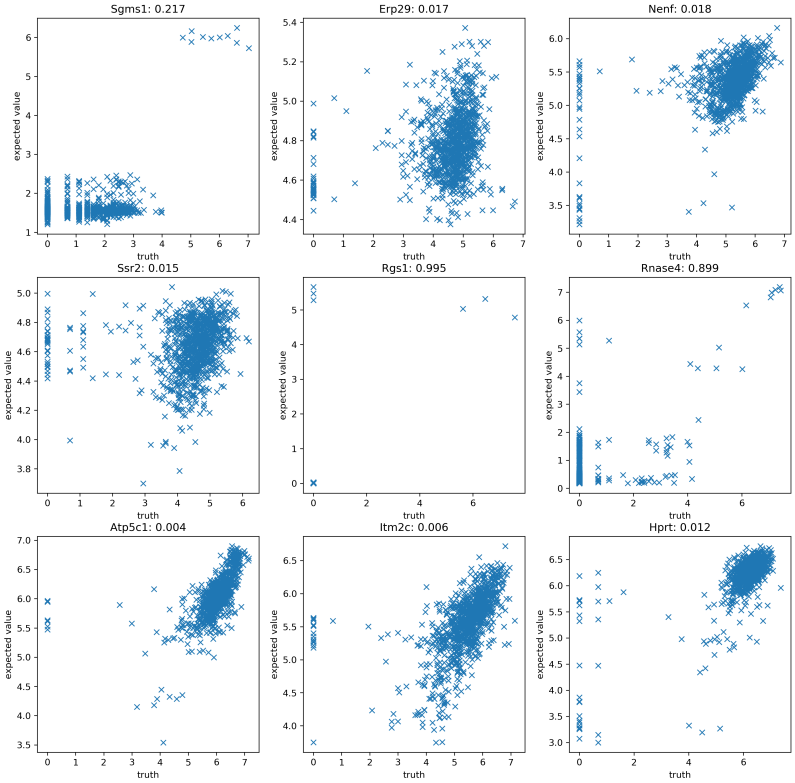


Figure 6

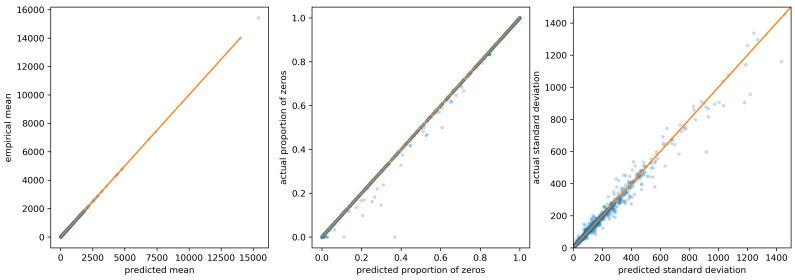


Figure 7

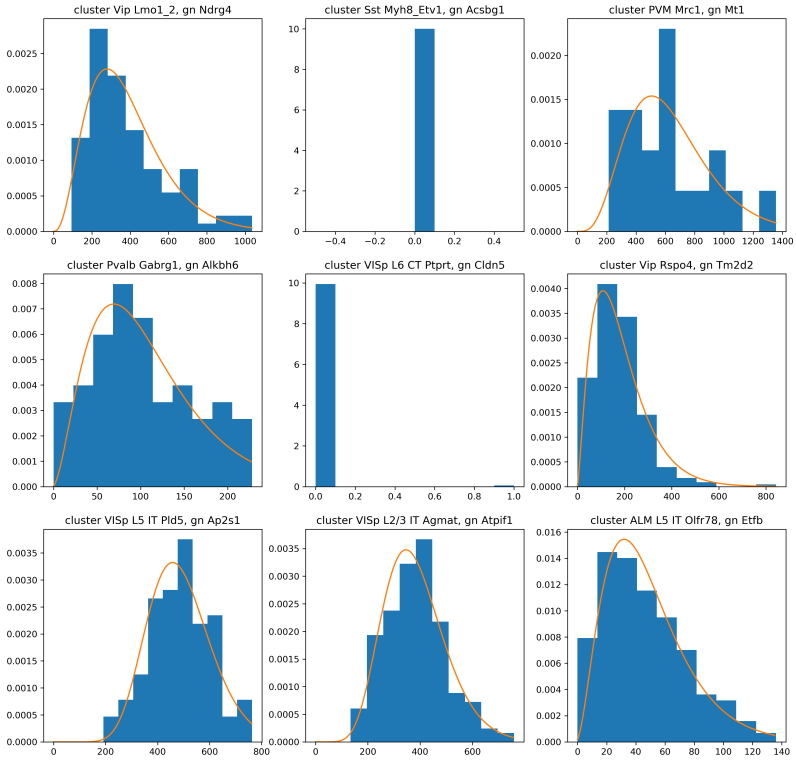


Figure 8: Example zinb fits for particular celltype-gene pairings

each cell-type. This yields a kind of three-dimensional scatterplot, in which each dot is a cell type, and the location indicates the value of each of its three fitted parameters. Here we show three projections of that scatterplot in Figure 9.

Clearly there is quite a lot of variety in terms of how these parameters may vary. They do not appear to lie on any kind of low-dimensional manifold. So we must be very careful in terms of what assumptions our model makes about how these parameters may be tied together.

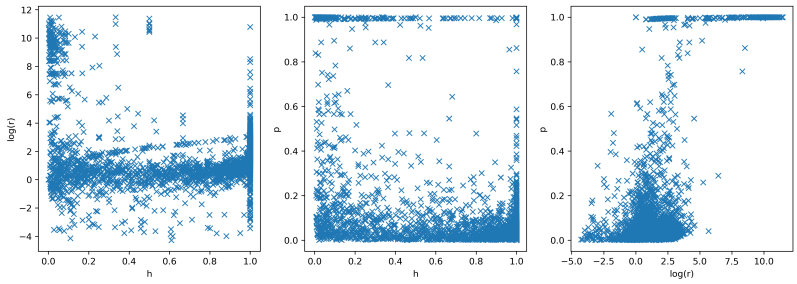


Figure 9