Learning robust statistics for scoring perturbations

February 3, 2023

1 Summary

We propose that the desired statistic s should take the form:

$$s(P) = H(\mathbb{E}_{x\sim P}(f(x))), \text{ for some } f : \mathbb{R}^{15000} \to \mathbb{R}^{m_1} \text{ and } G : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$$

where P is the gene expression distribution for a given perturbation, and $m_1 \approx 10$ should be interpreted as a collection of potentially nonlinear, gene expression programs and $m_2 \approx 10$ should be interpreted as a collection of multicellular programs. Three considerations inform our proposal for how to learn f:

1. $s(P)$ should be stable when estimated across variable (in particular, small) numbers of cells.
2. $s(P)$ should provide the information required to predict $d(Q, q(P))$ where Q is a 'test' cell type proportion vector and $q(P)$ is the cell type proportion vector associated to P
3. $s(P)$ should be predictable from sequence features of the perturbed gene.

Since we do not know a priori what the statistics that satisfy these requirements are, we propose learning f using gradient based optimization to explicitly model each of these requirements. We first describe the procedure, and next describe how it addresses these requirements. Implementation details for our proof-of-concept experiments are provided in Section 2, and our preliminary results are provided in Section 3.

- Parameterize f as a neural network.
- Define a prediction neural network

  $$H : \mathbb{R}^m \times \Delta^5 \to [0, 3],$$

  where Δ^5 denote the space of possible target proportion vectors, which will use the estimated statistic to predict the distance away from a desired cell type proportion vector
- Define a recognition network $T : \mathbb{R}^k \to \mathbb{R}^m$ which will predict $s(P)$ from k features of each perturbation (such as sequence features of each gene).
- At each training step:

1. Sample a small number (around twenty) of cells from each perturbation $P_1, \ldots, P_n$
2. Estimate of $\hat{s}(P_i)$ using the sampled cells (i.e., evaluate f on each sampled cell, take the mean, apply G).
3. Sample, for each perturbation P_i, a 'target' proportion vector Q_i
4. Evaluate $H(\hat{s}(P_i), Q_i)$, and define a prediction loss $\mathcal{L}_1(f, G, H) = H(\hat{s}(P_i), Q_i) - d(q(P_i), Q_i))$, where d denotes the total variation distance as in the previous challenges.
5. Compute a recognition loss as $L_2(f, G, T) = ||T(g_i) - \hat{s}(P_i)||^2$ where g_i denotes the features of the perturbation (e.g. sequence features).
6. Perform a gradient update that minimizes $L(f, G, H) + L(f, T)$.

This procedure addresses the requirements above as follows:

1. By sampling small and variable numbers of cells, the network can only learn statistics that are stable to the sampling procedure.
2. By including the prediction network trained on sampled target proportion vectors, the network can only learn statistics that provide information to predict the distance from the target.
3. By including the recognition network, the network can only learn statistics that can be predicted from features available without experimentation.

2 Implementation details

2.1 Architecture of f and G

We use a linear projection followed by tanh to parametrize f (the gene expression programs should be simple and interpretable). We use a self attention to parameterize G (each multicellular program attends to the other multicellular programs).

2.2 Architecture of H

We have utilized a cross attention approach. A target vector is projected by a feed forward neural network to a nxm key matrix, and an n dimensional value vector (where n is the number of value dimensions). Multiplying this matrix by the m dimensional vector $\hat{s}(P_i)$, followed by softmax, produces an n-dimensional attention mask for the target vector, input perturbation pair. Finally, this is multiplied elementwise by the n-dimensional value vector to produce the output prediction, which is then applied to a feedforward neural network to predict the final output.

2.3 Architecture of T

As proof of concept, we have computed language model embeddings for the 200 512 mers at a stride of 32 surrounding the TSS of each gene (this is provided as an extra data file), and used a feedforward neural network to regress on the statistic output.

2.4 Sampling strategies

We sample a uniformly distributed number (between 2 and 10) cells for each perturbation, reducing the average number of sampled cells as training occurs. We sample target proportion vectors by randomly sampling those observed in the original data, and adding noise (uniform noise followed by normalization so the proportions sum to 1).

3 Results

We selected 50 genes for training and 10 for validation. Figure 1a shows the baseline model, which had no recognition network and involved taking an average of 60 cells per sample. The validation prediction loss with the baseline is 0.23. Figure 1b shows that we do not lose too much in validation prediction accuracy by (a) including the recognition network and (b) taking an average number of 10 cells per sample - we reach validation accuracy of 0.28, but also a very low loss for the validation recognition network, indicating we may have learned a statistic that can be predicted across datasets.

Further analysis is ongoing to determine significantly enriched gene programs learned by the models

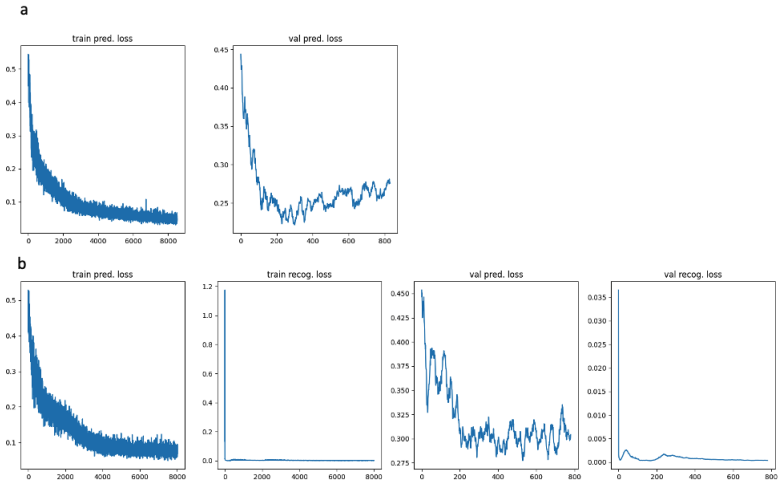

Figure 1: a. Baseline model training and validation prediction loss. b. Training and validation prediction and recognition loss for model with recognition network and 25 cells sampled per perturbation.