Summary

Our solution utilizes an ensemble of five identical multi-layer perceptrons (MLPs) that we trained end-to-end. The model has three main components: an encoder, a decoder, and a classifier. The encoder inputs perturbation genes and random noise and outputs a latent vector. When this vector is given to the decoder, it outputs cell expression data. We train the classifier to accurately classify the cell state distribution (such as 'progenitor', 'effector', etc.) for each latent vector produced by the encoder. Moreover, we use pre-trained Gene2vec embeddings as vector representations for the perturbation genes.

Network Architecture

The model comprises three multi-layer perceptrons (MLPs): an encoder network f_E, a decoder network f_D, and a classifier f_C. The classifier only receives the output of the generator as input. The generator has three layers and the classifier has two layers.

Training

To train the joint network, we sample both a perturbation gene embedding g and a cell's gene expression with that perturbation c_g. We feed the gene embedding concatenated to a vector of random noise $\mathbf{z} \sim N(\mathbf{0}, 0.5\mathbf{I}), \mathbf{z} \in \mathbb{R}^{64}$ to the encoder whose output is h_g. We feed h_g to a classifier f_C that is trained to predict (one-hot vector representation of) the state y (i.e. 'progenitor', 'effector', etc.) of the perturbed cell c_g. Additionally, the decoder takes the latent vector and is tasked to reproduce c_g, the gene expression of the sampled cell. In summary, the joint network is trained to minimize the loss:

$$L := ||c_g - f_D(h_g)||_1 + D_{KL}(y||f_C(h_g)), h_g := f_E(g \oplus \mathbf{z})$$

where $\oplus$ is the concatenation operator, $D_{KL}(\cdot||\cdot)$ is the Kullback–Leibler divergence.

Inference

To generate the cell-state proportions we only use the encoder and classifier networks. That is, given held-out gene g_{test} we use the encoder to generate latent vectors $S = \{f_E(g_{\text{test}} \oplus \mathbf{z_i})_i | \mathbf{z}_i \sim N(\mathbf{0}, \mathbf{I}), i = 1, 2, .., 1000\}$. These are then fed to the classifier to output the cell state probabilities $f_C(S) \in \mathbb{R}^{1000 \times 5}$. To calculate the final cell state proportions, we add together the probabilities from $f_C(S)$ along the sample dimension, resulting in a five-dimensional vector. We then normalize this vector so that it sums to one.

Data Processing

We reduce the dimensionality of the gene expression matrix data using the PCA algorithm down to 256 dimensions. We embed all perturbation genes using Gene2vec (Du et al., 2018), which encodes how often genes are expressed together in human cells.

References

[1] Du, J., Jia, P., Dai, Y., Tao, C., Zhao, Z., Zhi, D. (2018). Gene2vec: distributed representation of genes based on co-expression. BMC Genomics, 20.

[2] Fang, Z., Liu, X., Peltz, G. (2022). GSEApy: a comprehensive package for performing gene set enrichment analysis in Python. Bioinformatics, 39.