Challenge 3: Score Metrics Proposal

The online version of this proposal can be found here:
https://docs.google.com/document/d/1ZdVqbCYGD7GjkSI8AKqgScYvCGbYRORSzvjpRybq7vA/edit?usp=sharing

Author: OnePenguinCoder (the penguin)

Keywords: kullback–Leibler divergence(KLD); graphical neural network(GNN); dynamical model

Abstract

In this proposal, three general metrics are proposed by the penguin. [TODO]

Method

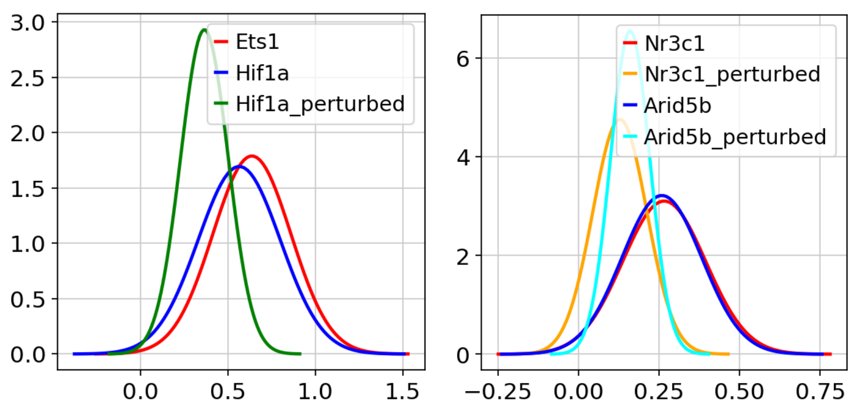
A KL-divergence based metric

One of the most straightforward methods based on the challenge's setting is to use metrics based on KL divergence to measure how each knock-out gene changes the class states. KL divergence tells how one distribution differs from the other. (P and Q below)

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right).$$

(wikipedia)

Because the ground-truth probabilistic distributions of our perturbation genes are unknown but the empirical experimental data, we need to take factors such as technical errors and sample size (aka. the number cells under different knockout conditions) into account. Let's ignore these factors and start with a simple metric.



From the penguin's notebooks included in their challenge submissions: on the right is a plot of Ets1, a gene in the test set, and Hif1a; on the left are gene Nr3c1 and Arid5b. Each gene's most similar gene based on KLD under the perturbation condition are included in the submission notebooks. The KLD scores are calculated based on the fitted gaussian distribution assumption of each gene and our normalization process includes log1p.

For a knockout gene, the score metrics based on KLD can be defined as:

$$score_j = \sum_i KLD(gene^i_{unperturbed}, gene^i_{perturbed})$$

(score_j =\sum_{i} {KLD(gene^{i}_{unperturbed}, gene^{i}_{perturbed})})

However, this metric has disadvantages including but not limited to 1) not possible for predicting unseen perturbations 2) not taking sample sizes into account.

Graphical neural network based metric

In challenge 1, the penguin took part of their on-going research work and implemented a GNN with graph convolution layer to learn a graphical model based on the data. Unfortunately, this graphical model's performance is bad in terms of predicting each cell sample's state. (accuracy=0.4, with random forest without fine-tuning achieved 0.42)

"The gist is to construct a graph consisting of gene nodes to describe the joint probability distribution of genes. This can be done with message passing among gene nodes. We optimize this graph network by CrossEntropyLoss from pytorch. The inputs are the graphs (each cell sample as a graph) and outputs are the states. Our trained models only attained 0.40 accuracy regarding cell state classification."

Assume we have good graphical representation of gene graphs trained in each condition (knockout gene subsets), we can define metrics based on how much the graphs change in each condition.

Dynamical system based metric

Since the organizer did not provide the competitors including the penguin with either the raw fastq file or (unspliced and spliced) matrices, the penguin did not apply any RNA velocity and dynamical system based methods to this iPSc dataset.

Reference

[TODO] due to the penguin’s tight schedule