Single Cell Immunotherapy Data Science Challenge 3

Team "It's MAGIC": Rachit Mukkamala, Ananth Shyamal, Victory Yinka-Banjo

February 2, 2023

Introduction to our Scoring Function

During analysis of Perturb-Seq datasets, it is necessary to design a scoring function to rank gene perturbations in order of desired outcome. For the purposes of this data science challenge, the desired outcome is for the perturbation to shift the distribution of T-cell states towards a desired, optimal distribution, which we define as Q_d. The scoring functions currently being used to score perturbations in Challenge 2 solely depend on Q_i, or the predicted 5-state distribution for a perturbation, and do not take into account P_0 or P_i, which represent the 15,077-dimensional average gene expression vector for the unperturbed and perturbed states, respectively.

For each cell state s, we iterate over all 15,077 genes and perform a T-test for differential expresssion (DE) between state s and the 4 other states. After performing FDR-correction for multiple hypothesis testing and ordering by q-value (corrected p-value), we obtain a rank-ordered list of DE "marker" genes that define each cell state. The core assumption of our scoring function is that the expression values of the top d differentially expressed genes reflect the classification boundary between cell states.

Thus, unlike in challenge 2, which uses Q_i to summarize P_i, **the statistic $S(\cdot)$ that we compute to summarize P_i and/or P_0 simply subsets the expression values of the top d differentially expressed (DE) genes per cell state from P_0 and/or P_i.** Since there are 5 cell states, **the dimension of $S(P)$ will be $5 \cdot d$ - dimensional.** When using our statistic and scoring function, a machine learning model intended to predict effects of unseen perturbation would be trained to predict $S(P_i)$, which is much easier to predict that the entire P_i vector since $S(P_i)$ has reduced dimensionality.

After obtaining $S(P_i)$ and $S(P_0)$, we then compute the log fold change between these two vectors, which we can define as the $5 \cdot d$ dimensional vector D_R. **Ultimately, our scoring function is a weighted sum of each component in D_R.**

How are the weights in this weighted sum determined? We decided that these weights should depend on:

- **(a)** the significance of each DE gene term
- **(b)** the direction of enrichment of each DE gene term
- **(c)** the difference between the desired cell state vector Q_d and the unperturbed cell state vector Q_0.

To fulfill criteria **(a)** and **(b)**, we multiply each component in D_R by the t-statistic (or Z-score) for the component's corresponding DE gene. These t-values were obtained earlier from the T-test for differential expression. We then normalize each column such that the absolute value of the sum of the column equals 1. This weight choice fulfills both criteria since a t-statistic reflects both the significance and direction of a gene's differential expression.

Let W_R (Weighted ratios) be the weighted version of D_R described in the previous paragraph. Let Q_R be the log fold change between Q_D and Q_0. To fulfil criterion **c**, the d components in W_R corresponding to each cell are added together, creating a 5-dimensional vector. We then take the dot product of this 5-dimensional sum vector with Q_R to yield our final perturbation score.

Mathematical Formulation

$$S(P_i) := \begin{pmatrix} | & | & | & | & | \\ d_i^{progenitor} & d_i^{effector} & d_i^{exhausted} & d_i^{cycling} & d_i^{other} \\ | & | & | & | & | \end{pmatrix} \ (d \times 5)$$

$$S(P_0) = \begin{pmatrix} | & | & | & | & | \\ d_0^{progenitor} & d_0^{effector} & d_0^{exhausted} & d_0^{cycling} & d_0^{other} \\ | & | & | & | & | \end{pmatrix} \ (d \times 5)$$

$$W = \begin{pmatrix} | & | & | & | & | \\ t_{progenitor} & t_{effector} & t_{exhausted} & t_{cycling} & t_{other} \\ | & | & | & | & | \end{pmatrix} \ (d \times 5)$$

$S(P_i)$ (DE gene matrix for perturbed state) is a $d \times 5$ matrix containing the predicted post-perturbation expression values for the top d differentially expressed markers genes per cell state. Each column in $s(P_i)$ represents the expression of the top d marker genes for that cell state. We chose to represent $s(P_i)$ as a reshaped $5 \times d$ matrix instead of a length $5 \cdot d$ vector to make analysis easier.

Similarly, $S(P_0)$ (DE gene matrix for unperturbed state) is a $d \times 5$ matrix that contains the average unperturbed state expression for the exact same genes.

W (weight matrix) is a $d \times 5$ matrix that contains the Z-scores (aka t-statistic) for each DE gene in D_i and/or D_0. W is normalized so that the absolute value of the sum of each column is equal to 1.

Additionally, let Q_d be the 5-dimensional column vector corresponding to the desired cell state fractions, and let Q_0 be the 5-dimensional column vector corresponding to the unperturbed cell state fractions.

$$D_R = \log\left(S(P_i) \oslash S(P_0)\right)$$
$$Q_R = \log\left(Q_d \oslash Q_0\right)$$
$$W_R = D_R \odot Q_R$$
$$O = [111...1] \text{ } (1 \times d \text{ 1's vector for column sum})$$
$$\text{Score} = O \times W_R \times Q_R = \text{diag}(D_R^T \times W) \times Q_R$$

Note: $\oslash$ represents element-wise (Hadamard) division and $\odot$ represents element-wise (Hadamard) product.

Intuition about the Behavior of our Scoring Function

Thinking more about the behavior of our scoring function and why it would be effective, consider an arbitrary differentially expressed gene d which is significantly overexpressed in progenitor cells, for instance. The more significant the difference between d's expression in progenitor cells versus the rest, the higher d's perturbation fold change would be weighted, which makes sense, since we want to give more importance to genes we are more significantly confident about.

Additionally, suppose that we want to increase the fraction of progenitor cells. If d was increased by a perturbation, since d is overexpressed in progenitor cells, our function would give d a positive score. Since the log fold change for Q's progenitor fraction would be positive, our score function would reward this change with a higher score. Alternatively, if suppose d's expression was decreased by a perturbation, our scoring function would penalize this suboptimal change since d's weighted score would be negative.

Lastly, suppose that we want to decrease the fraction of progenitor cells. If d was increased by a perturbation, since d is overexpressed in progenitor cells, our function would give d a positive score. BUT, since the log fold change for Q's progenitor fraction would be negative, our score function would penalize this opposite to optimal change. Alternatively, if d's expression was decreased by a perturbation, our scoring function would reward this change since d's weighted score would be negative, but the negative sign would be cancelled by the negative log fold change in Q_R's progenitor component.

Thus, through these examples we can clearly see that our scoring function rewards better perturbations with a more positive score, gives perturbations that do not deviate from the unperturbed state a score of zero, and lastly gives perturbations that perform worse than the default unperturbed state a negative score. This provides far more directional information than, say, an L1- or L2-loss score, which only provides the magnitude of the prediction error.

Results and Discussion

We benchmarked our scoring function on the genes perturbed in the training dataset using three example Q_d desired cell-state vectors. In these three examples, we set $d = 5$. We compared the scores our function produced with the scores produced by the L1-loss scoring function (which compared Q_d to Q_i, the observed post-perturbation cell state vector). Figures 1, 2, and 3 clearly indicate that our scoring function tracks well with the L1-loss, showing that our scoring function is an informative tool to evaluate perturbations.

In these tests, we imputed the "X" data matrix using $MAGIC$ imputation (van Dijk et al.) to ensure that differential expression results were due to truly significant changes and not just an arbitrary abundance of zeroes in certain cell clusters.

It is to be noted, however, that there are discrepancies between the L1-loss score and our scoring function. We believe that these discrepancies, within a reasonable bound, are acceptable, since they indicate that our scoring function is prioritizing different criteria as opposed to the L1-loss. Still, since the baseline L1-loss score is reasonably correlated to our score function, we can be confident about its predictions.

One of the biggest advantages of our scoring function is that, as mentioned above, it provides much more information regarding the direction of a perturbation's performance as opposed to a scoring function that is blind to other information contained in P_0, Q_0, and P_i.

Another major advantage of our scoring function is related to the nature of our summary statistic, $S(\cdot)$. Since our summary statistic is the predicted expression of a subset of genes, when training models to predict unseen perturbations, it is possible to incorporate

powerful gene function relationship information such as Gene Ontology networks into these models to boost the prediction accuracy of the summary statistic, $S(P_i)$. For instance, GEARS (Roohani et al.) is able to leverage functional annotations from gene ontology to predict not just our subset of genes, $S(P_i)$, but rather the entire 15,077-dimensional post-perturbation vector P_i! With our scoring function, frameworks such as GEARS which leverage functional network information can be readily adapted to predict and score unseen perturbations.

One last detail that we would like to cover is related to strategies for quantifying the uncertainty of a predicted perturbation score. Rather than just returning a predicted perturbation score, it would be incredibly valuable and informative to also compute a confidence interval for a scoring metric to take into account the predictive model's uncertainty and/or margin of error. One powerful, model-agnostic technique to construct confidence intervals for any scoring function, including ours, would be to train a randomized ensemble of 10-30 predictive models in order to obtain a distribution of predicted perturbation scores. Then, by empirically identifying the 5th and 95th percentiles, we can directly compute the 95% confidence interval for a predicted perturbation score.

If the model's architecture relies on random initialization/actions (i.e. neural network weights, random forest models), then an easy way to train an ensemble would be to train each model in the ensemble with a unique random seed. Alternatively, for purely deterministic models, we can generate bootstrap samples from the training dataset to ensure that each model (which is trained on a different bootstrap sample) behaves slightly differently from the rest.

Python Code

```
perturbation = '----' #insert gene name here
d = 5
sc.tl.rank_genes_groups(adata, 'state', layer='magic', method='t-test', rankby_abs=True, n_genes=d)

cell_states = ['progenitor', 'effector', 'terminal_exhausted', 'cycling', 'other']
W = pd.DataFrame({s: adata.uns['rank_genes_groups']['scores'][s] for s in cell_states})
W = W / np.abs(np.sum(W, axis=0))

S_i = pd.DataFrame({s: X_perturbed.loc[adata.uns['rank_genes_groups']['names'][s], perturbation].to
S_0 = pd.DataFrame({s: X_perturbed.loc[adata.uns['rank_genes_groups']['names'][s], 'Unperturbed'].t

Q_0 = Y['Unperturbed']
Q_d = np.array([a,b,c,d,e]) #insert desired cell-state vector here

D_R = np.log((S_i + tol) / (S_0 + tol))
W_R = W * D_R
Q_R = np.log((Q_d + tol) / (Q_0 + tol))
score = np.dot(np.sum(W_R, axis=0), Q_R)
```

Figures

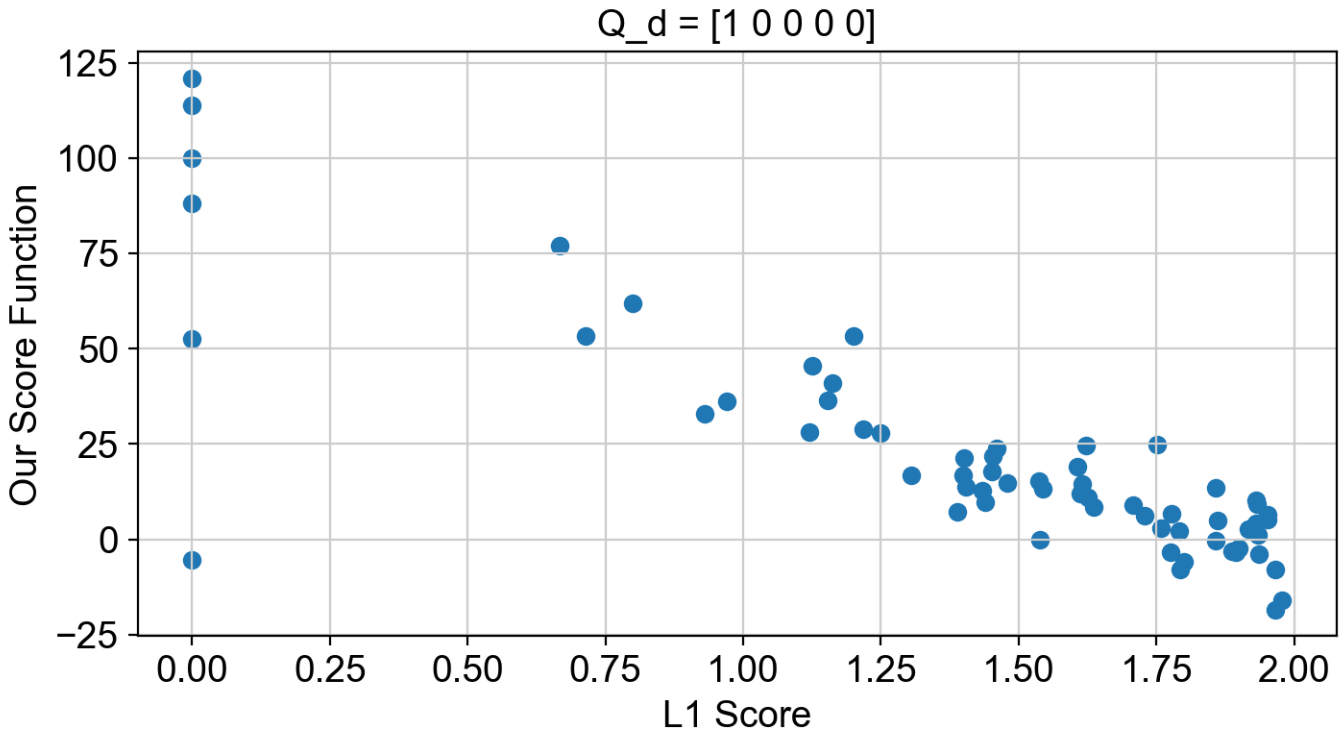

Figure 1: Our score function's score versus the L1 loss between Q_d and the observed Q_i across all 65 perturbations in the train set. Here, we used Q_d = [1,0,0,0,0].

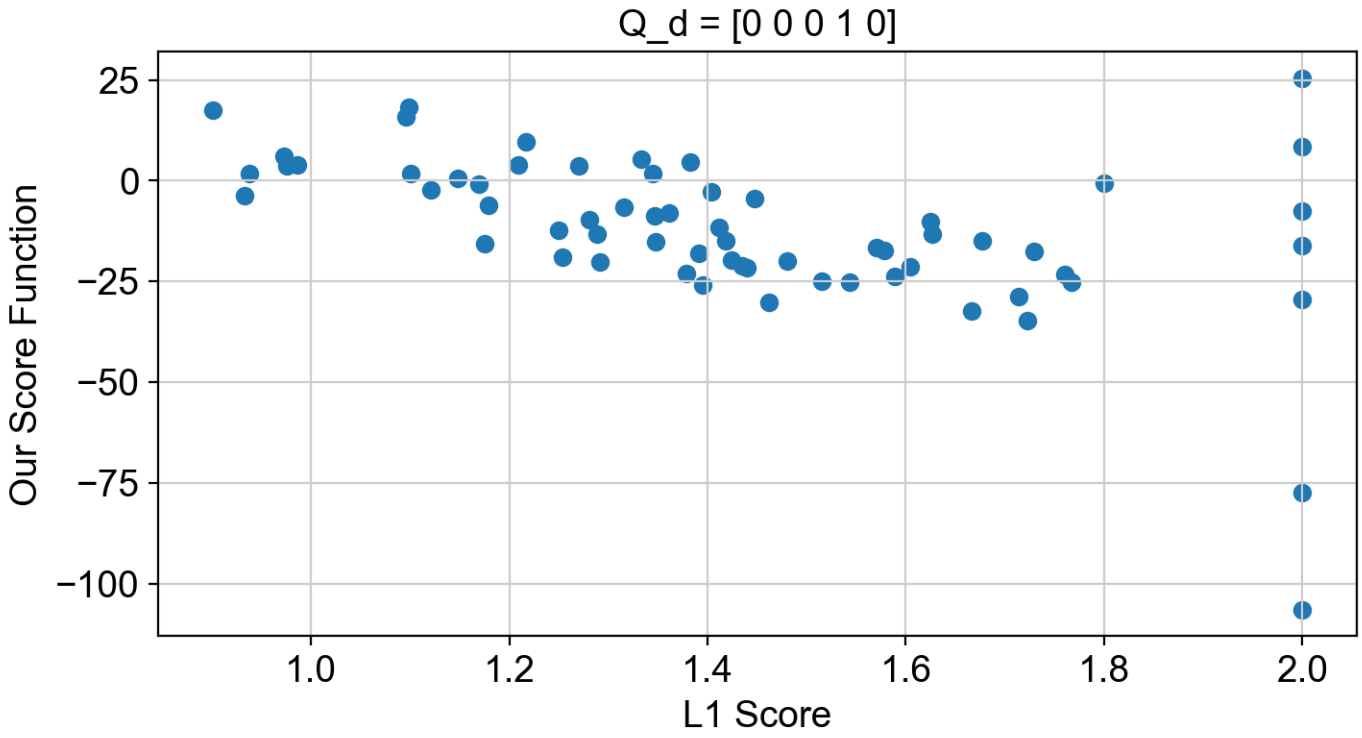

Figure 2: Our score function's score versus the L1 loss between Q_d and the observed Q_i across all 65 perturbations in the train set. Here, we used Q_d = [0,0,0,1,0].

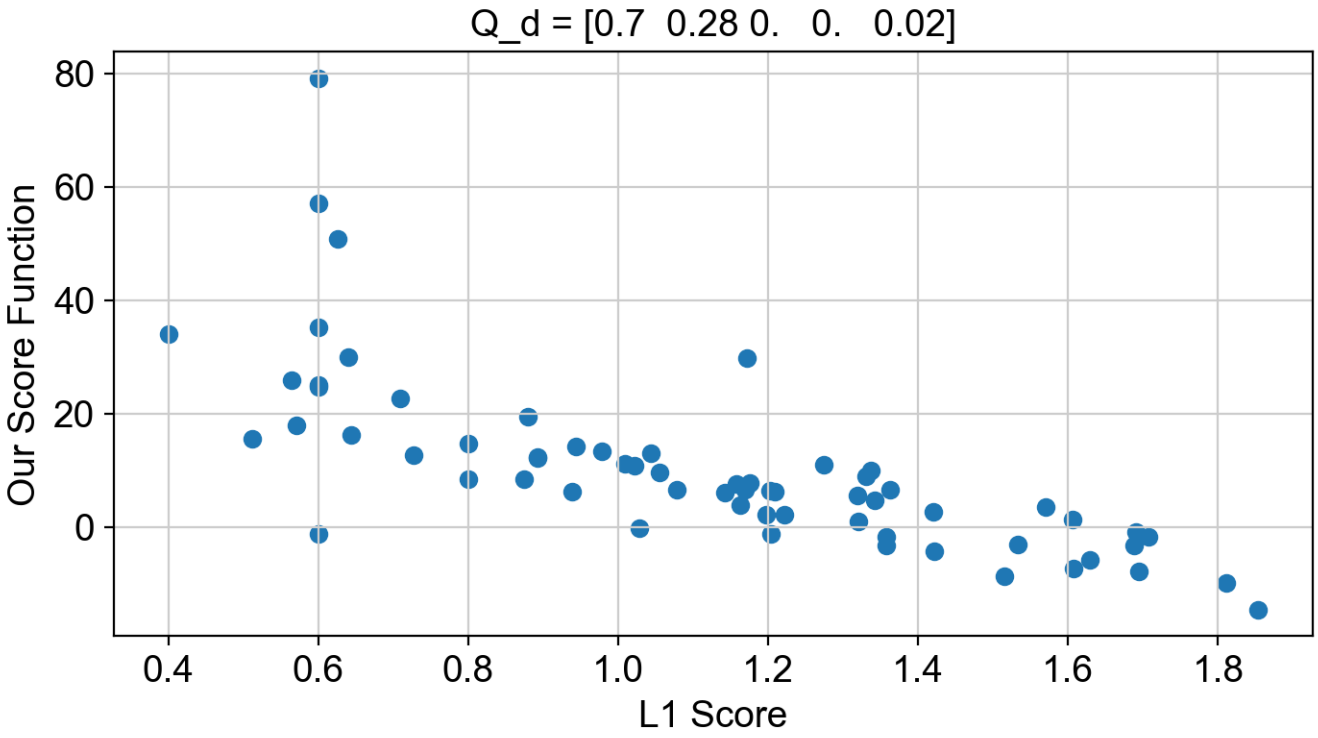

Figure 3: Our score function's score versus the L1 loss between Q_d and the observed Q_i across all 65 perturbations in the train set. Here, we used Q_d = [0.7,0.28,0,0,0.02].

References

- Roohani, Y., Huang, K., Leskovec, J. (2022). GEARS: Predicting transcriptional outcomes of novel multi-gene perturbations. In bioRxiv. https://doi.org/10.1101/2022.07.12.499735
- van Dijk, D., Sharma, R., Nainys, J., Yim, K., Kathail, P., Carr, A. J., Burdziak, C., Moon, K. R., Chaffer, C. L., Pattabiraman, D., Bierie, B., Mazutis, L., Wolf, G., Krishnaswamy, S., Pe'er, D. (2018). Recovering gene interactions from single-cell data using data diffusion. Cell, 174(3), 716-729.e27. https://doi.org/10.1016/j.cell.2018.05.061