A Functional Interpretation of Coordinated Cell State Changes in Cancer Immunotherapy

@jackson6
xleratorxlerator9@gmail.com
University of Toronto
Feb 3, 2023

Abstract

Gene Set Enrichment Analysis (GSEA) (Subramanian et al., 2005; Mootha et al., 2003) is a computational method used to analyze gene expression data and identify pathways (or functionality-related groups of genes) that are up- or down-regulated across two biological states. Examples of pathways that can be analyzed using GSEA include gene sets related to the regulation of the cell cycle process, DNA metabolic process, and mitochondrial translational termination. Here we propose a statistic that is an estimate of the state proportion vector solely based on the identified pathways found using GSEA across the five cell states.

Our proposed scoring function is the distance between the desired state proportion vector and the estimate (e.g. our statistic), the best choice, we will argue, is the cosine loss. We empirically demonstrate that the cosine loss better accounts for the classification boundaries for each cell state. Specifically, we show that cosine loss best correlates to the accuracy of the statistic when using the statistic as an estimate of the cell state. Moreover, we avoid using accuracy as the proposed scoring function and compare the cosine loss to more informative metrics. Namely, the L1 loss, Kullback–Leibler divergence, and the Jensen-Shannon Divergence. (We provide the GSEA output files in a Supplementary Data folder alongside code in the Appendix to compute the statistic and scoring and scoring functions)

Preliminaries

Let j represent the different cell states (i.e. 'terminally exhausted', 'cycling', etc.). Define $\Omega(j) := \{G_0, G_1, ...\}$ as a set of genes sets that likely affect j. Each subset (or pathway) $G \in \Omega(j)$ contains a group of genes with a meaningful up- or down-regulation with respect to the cell state j. Where here we use the results of GSEA to compute $\Omega(j)$ for every state j.

The Statistic

For a perturbation gene i the proposed statistic $s(P_i)$ is given by a five-dimensional vector indexed by j. Where j represents the different cell states (i.e. 'terminally exhausted', 'cycling', etc.). Namely,

$$s(P_i)_j = s(i)_j := \frac{1}{|\Omega(j)|} \sum_{G \in \Omega(j)} \mathbb{1}_G(i) * \phi(G)$$

where $\mathbb{1}_G(\cdot)$ is the indicator function for the set G and the weighting function ϕ is taken to be the sign of the *enrichment score* for the gene set (given by the GSEA). In simple terms, given this definition for $\phi(G)$, the proposed statistic is an estimate of how much each cell state is expected to change given knock-out gene i based on a direct relationship to the meaningful pathways G with respect to the cell state j. Note, to let the statistic be predictable for unobserved knock-outs we only use the perturbation gene as input instead of directly using P_i or an estimand thereof.

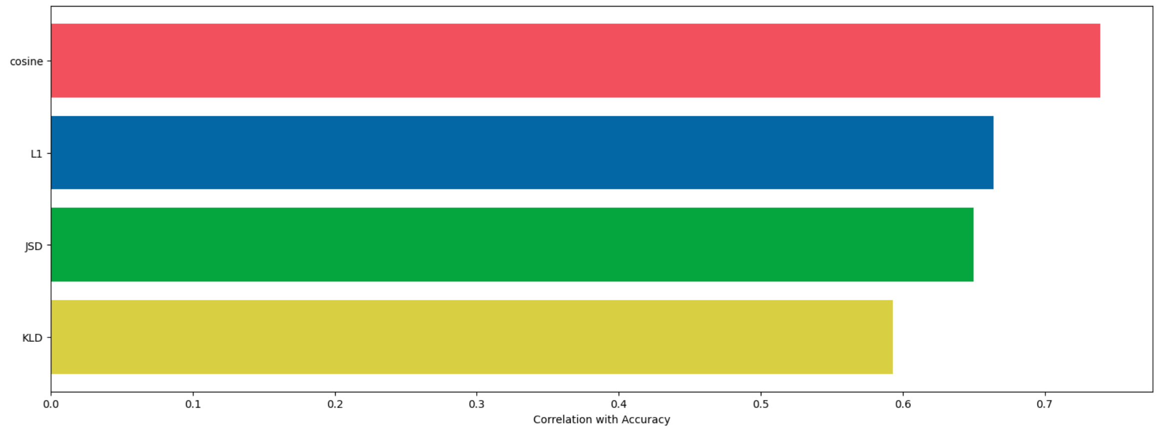

Figure 1: Comparison of how well each of the distance measures correlates with the classification accuracy produced using the proposed statistic.

The weighting function $\phi(G)$

The weighting function measures how much the cell state j changes given that a pathway G is affected by the knock-out of the perturbation gene i. Note again that here ϕ is defined as the sign of the *enrichment score* (ES) for the gene set. Where ES < 0 if the gene set is down-regulated for the cell state j. Similarly, ES > 0 if the gene set is up-regulated for the cell state j. However, this choice of ϕ assumes that the knock-out of a gene affects all pathways containing the gene (and affects them equally) resulting in a change to the expected cell state. Nonetheless, these simplifying assumptions allow the proposed statistic to be parameterless, interpretable, and easily predictable for unseen perturbations.

GSEA

To explore the relevant signaling pathways, GSEA (Gene Set Enrichment Analysis) (Subramanian et al., 2005) was conducted in a *one-vs-rest* manner for each of the five cell states. Hence, for a given cell state we have a list of gene sets (pathways) with significant upregulation and downregulation with respect to the state using GSEA. We only keep gene sets with a *p-value* < 0.001 and a *false-discovery rate* < 0.25 as we regard them as statistically significant.

Due to computational limitations, we can only perform GSEA on a subset of about 8100 genes per run. Hence in practice, we perform GSEA over two halves of the genes in the dataset. The first half contains 8100 genes considered highly variable by "Seurat v3" (Stuart et al., 2019) and the second half contains the remaining genes. GSEA is performed using the Python package *GSEApy* (Fang et al., 2022).

The Scoring Function

The proposed statistic $s(P_i)$ is designed such that $\widehat{Q} := softmax(-s(P_i)/10)$ can be used as an estimate of the true proportions vector. Hence, the proposed scoring function (denoted as D) is defined in terms of the negative of a distance function d:

$$D(Q, \widehat{Q}) := 1 - d(Q, \widehat{Q}) = \frac{Q \cdot \widehat{Q}}{\|Q\|_2 \|\widehat{Q}\|_2}$$

where above we take $d(\cdot, \cdot)$ to be the cosine distance. Therefore, D is the cosine loss. For the remainder of this write-up, I will argue that the cosine distance is the best choice for d and will provide some empirical evidence comparing it to reasonable alternatives.

The cosine loss, the L1 loss, Kullback–Leibler divergence (KLD), and the Jensen-Shannon Divergence (JSD) loss all are used for classification but they each promote a different balance between robustness and accuracy. In the classification setting Barz & Denzler (2019) suggest using the cosine loss directly with the class probabilities works well with small datasets and can prevent overfitting. Classifiers trained with cosine loss still remain competitive with other regularisation techniques that use data augmentation or specialized classifier architectures (Brigato et al, 2022). The L1 loss is provably robust to label noise but struggles to optimize accuracy (Ghosh et al, 2017). Moreover, classifiers trained using the L1 loss often underfit and get a low accuracy (Ghosh et al, 2017; Zhang & Sabuncu, 2018). On the other hand, KLD is widely known to strongly promote strong accuracy results but suffers from overfitting to label noise. JSD is a more principled version of the KLD as it is symmetric in its arguments and is a similarity score between distributions. Englesson & Azizpour (2021) prove that JSD is an interpolation between the L1 loss and KLD and can take on the strengths of both.

We select $d(\cdot, \cdot)$ (and therefore the proposed statistic $D := 1 - d$) that best preserves the classification boundaries of each cell state. Namely, we will compare the cosine loss, L1 loss, KLD, and JLD by their correlation with classification accuracy. Where the classification accuracy is computed between the desired proportions Q and the proportions predicted by the statistics $\widehat{Q}$. We plot the results in Figure 1, which provides the basis for selecting the cosine loss.

References

[1] Barz, B., Denzler, J. (2019). Deep Learning on Small Datasets without Pre-Training using Cosine Loss. 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), 1360-1369.

[2] Brigato, L., Barz, B., Iocchi, L., & Denzler, J. (2022). Image Classification with Small Datasets: Overview and Benchmark. IEEE Access, PP, 1-1.

[3] Englesson, E., Azizpour, H. (2021). Generalized Jensen-Shannon Divergence Loss for Learning with Noisy Labels. Neural Information Processing Systems.

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

[5] Ghosh, A., Kumar, H., & Sastry, P.S. (2017). Robust Loss Functions under Label Noise for Deep Neural Networks. AAAI Conference on Artificial Intelligence.

[6] Mootha, V.K., Lindgren, C.M., Eriksson, K.-., Subramanian, A., Sihag, S., Lehár, J., Puigserver, P., Carlsson, E., Ridderstråle, M., Laurila, E., Houstis, N.E., Daly, M.J., Patterson, N.J., Mesirov, J.P., Golub, T.R., Tamayo, P., Spiegelman, B.M., Lander, E.S., Hirschhorn, J.N., Altshuler, D.M., & Groop, L.C. (2003). PGC-1-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. Nature Genetics, 34, 267-273.

[7] Stuart, T., Butler, A., Hoffman, P.J., Hafemeister, C., Papalexi, E., Mauck, W.M., Hao, Y., Stoeckius, M., Smibert, P., & Satija, R. (2019). Comprehensive Integration of Single-Cell Data. Cell, 177, 1888-1902.e21.

[8] Subramanian, A., Tamayo, P., Mootha, V.K., Mukherjee, S., Ebert, B.L., Gillette, M.A., Paulovich, A.G., Pomeroy, S.L., Golub, T.R., Lander, E.S., & Mesirov, J.P. (2005). Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. Proceedings of the National Academy of Sciences of the United States of America, 102, 15545 - 15550.

[9] Zhang, Z., & Sabuncu, M.R. (2018). Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. ArXiv, abs/1805.07836.

Appendix

FAQ

Why use GSEA?

The purpose of using GSEA in the proposed statistic is to use a functional interpretation of large-scale gene expression data to identify measurable coordinated changes to the cell states that are indicative of biological processes.

Does the scoring function take uncertainty into account given by the different sample sizes for the perturbations in the training dataset?

No.

Does the scoring function take into account the predicted number of cells resulting from a guide and favor perturbations with a large growth rate?

No.

Does the scoring function take into account the classification boundaries of each cell state?

Yes. We chose cosine loss as the scoring function based on the higher correlation with classification accuracy when compared to other loss functions. Where the classification accuracy is computed between the desired proportions Q and the proportions predicted by the statistics $\widehat{Q}$.

Does the scoring function $s(P_i)$ depend on the distribution P_i?

$s(\cdot)$ is not dependent on the expression distribution directly. Instead, it only depends on the knock-out gene i.

Why not just include the accuracy in the scoring function to enable the function to take into account the classification boundaries?

The scoring function should not be a combination of several other scoring functions. Take for example the linear combination:

$$D(Q, \widehat{Q}) := D_1(Q, \widehat{Q}) + \lambda D_2(Q, \widehat{Q})$$

Where D_2 is the accuracy metric. Here, λ would need to be estimated, which is not advisable given a few (<100) ground truth proportion vectors Q.

Code

Here we provide the Python code needed for running the proposed statistic and scoring function we a simple example. We used Python 3.8.10 with `scipy==1.7.3`, `numpy==1.21.6` and `pandas==1.3.5`

```python
from collections import namedtuple
import pandas as pd
import numpy as np


GeneGroup = namedtuple('GeneGroup', 'name nes genes state size')
STATES = ['progenitor', 'effector', 'terminal exhausted', 'cycling', '
    other']


def create_gene_groups(dir_, max_p_val = 0.001, max_fdr = 0.25):
    # dir_: the directory of the GSEA csv files

    gene_groups = []
    for study_state in STATES:

      df = pd.read_csv(dir_ + f'res5-{study_state}.csv')
      df = df[df['NOM p-val'] < max_p_val]
      df = df[df['FDR q-val'] < max_fdr]
      for (t, n, g) in df[['Term', 'NES', 'Lead_genes']].values:
        genes = g.split(';')
        gene_groups.append(GeneGroup(t, n, genes, study_state, len(
    genes)))

      df = pd.read_csv(dir_ + f'res5r-{study_state}.csv')
      df = df[df['NOM p-val'] < max_p_val]
      df = df[df['FDR q-val'] < max_fdr]
      for (t, n, g) in df[['Term', 'NES', 'Lead_genes']].values:
        if isinstance(g, str):
          genes = g.split(';')
          gene_groups.append(GeneGroup(t, n, genes, study_state, len(
    genes)))
    return gene_groups

# set p the directory of the GSEA files
p = "/my/dir/gsea/"
gene_groups = create_gene_groups(p)
```

Listing 1: Loads and processes the GSEA output files

```
import numpy as np


def statistic(ko_gene, gene_groups):

  sub_groups = [g for g in gene_groups if ko_gene in g.genes]
  counts = {s:0 for s in STATES}
  for g in gene_groups:
    counts[g.state] += 1

  if len(sub_groups) == 0:
    return None

  res = np.zeros(5, 'float32')
  for group in sub_groups:
    i = STATES.index(group.state)
    res[i] += np.sign(group.nes) / counts[group.state]

  return res
```

Listing 2: The proposed statistic

```
from scipy.spatial.distance import cosine as cosine_distance
from scipy.special import softmax


def scoring(q, s):
  # q, s: 1d numpy arrays

  q = np.array(q, s.dtype)
  q_hat = softmax(-s/10)
  loss_ = 1. - cosine_distance(q, q_hat)
  return loss_
```

Listing 3: The proposed scoring function

```
s = statistic('AQR', gene_groups)
# s: [ 0., -0.00563698,  0.01188707,  0.00603136,  0.]

# the desired proportion
q = [0, 0, 1, 0, 0]
scoring(q, s) # 0.4467918276786804
```

Listing 4: A simple usage example