Cancer Immunotherapy Data Science Grand Challenge
Challenge 3

Name: John Gardner
Handle: gardn999

Here is my scoring function proposal:

$$score_i = \left(1 - \frac{\sum \left(w_{state} \times \left|Q_{state} - s_{i,state}\right|\right)}{\sum \left(w_{state} \times \left|Q_{state} - s_{0,state}\right|\right)}\right) \times \left(1 - \frac{1}{\sqrt{n_i}}\right)$$

w = the vector of weights for each of the cell states
Q = the desired cell state proportion vector
s_i = the cell state proportion vector for gene i sample data
s_0 = the cell state proportion vector for the unperturbed sample
n_i = the number of samples in the gene i sample data

The numerator and denominator each sum over the T-Cell states with the numerator using the gene i sample and the denominator using the unperturbed sample. This results in the unperturbed sample score always being zero. A negative score indicates the perturbation tends to move cells to a less desirable state and a positive score to a more desirable state. The term on the right is an attempt to take into account uncertainty by setting a higher penalty to the score for lower sample sizes. A maximum score of 1.0 could only be achieved if the cell state proportion exactly matched Q with an infinite sample size. I did not attempt any proposal for the selection of Q and the weight term, w, is also adjustable.

On the following page are scores using my function for all knocked-out genes in the training data. In this example, Q is arbitrarily set to [0.5, 0.3, 0.0, 0.2, 0.0] and w is set to match the weights of the formula used for Challenge 2, part B) Car T-Cell Therapy: [1/0.0675, 1/0.2097, 1/0.3134, 1/0.3921, 0]. On the last page, is an implementation in Python.

gene:	score	n	s
Unperturbed:	0.00000	4978	[0.06750, 0.20972, 0.31338, 0.39213, 0.01728]
Tox2:	-0.23644	4333	[0.01731, 0.05700, 0.36787, 0.53081, 0.02700]
Tpt1:	0.59410	25	[0.44000, 0.16000, 0.12000, 0.28000, 0.00000]
Tcf7:	0.14024	297	[0.10438, 0.29293, 0.31313, 0.27609, 0.01347]
Il12rb1:	-0.09092	861	[0.03252, 0.18699, 0.31591, 0.45064, 0.01394]
Ikzf3:	-0.10437	566	[0.05300, 0.12721, 0.38869, 0.41519, 0.01590]
Nr4a3:	0.39446	592	[0.28041, 0.16723, 0.23649, 0.30236, 0.01351]
Litaf:	0.07381	363	[0.07163, 0.35262, 0.23691, 0.32782, 0.01102]
Elf1:	0.42322	73	[0.26027, 0.34247, 0.17808, 0.20548, 0.01370]
Irf2:	-0.10804	1155	[0.03463, 0.16277, 0.33333, 0.45195, 0.01732]
Arid5b:	-0.26456	2402	[0.01124, 0.04455, 0.41299, 0.51374, 0.01749]
Zeb2:	-0.14050	858	[0.01748, 0.11538, 0.28205, 0.43939, 0.14569]
Satb1:	0.30031	504	[0.19246, 0.25992, 0.20635, 0.31944, 0.02183]
Dvl2:	-0.17154	2173	[0.02439, 0.11551, 0.30419, 0.54947, 0.00644]
Nr4a1:	0.34899	184	[0.28261, 0.13043, 0.29348, 0.28804, 0.00543]
Hif1a:	-0.15421	685	[0.03212, 0.11679, 0.33723, 0.50657, 0.00730]
Crem:	-0.01551	761	[0.03417, 0.42576, 0.21419, 0.30880, 0.01708]
Runx2:	-0.00429	349	[0.03438, 0.28080, 0.27507, 0.39542, 0.01433]
Ctnnb1:	0.68385	43	[0.53488, 0.20930, 0.13953, 0.11628, 0.00000]
Tcf3:	0.12240	89	[0.14607, 0.16854, 0.21348, 0.44944, 0.02247]
Foxo1:	0.03662	90	[0.10000, 0.18889, 0.35556, 0.35556, 0.00000]
Dvl1:	0.02696	595	[0.04202, 0.30756, 0.26723, 0.36471, 0.01849]
Gsk3b:	0.37570	235	[0.29787, 0.11064, 0.25106, 0.31064, 0.02979]
Dkk3:	0.21408	48	[0.18750, 0.20833, 0.22917, 0.37500, 0.00000]
Hmgb1:	0.02436	377	[0.07162, 0.23077, 0.33687, 0.34218, 0.01857]
Dvl3:	0.49319	92	[0.34783, 0.16304, 0.23913, 0.22826, 0.02174]
Sox4:	-0.07570	330	[0.06970, 0.11212, 0.29091, 0.51212, 0.01515]
Fzd1:	-0.04345	265	[0.09811, 0.07925, 0.35094, 0.43019, 0.04151]
Stat4:	0.45561	30	[0.30000, 0.30000, 0.20000, 0.16667, 0.03333]
Nr4a2:	0.14947	680	[0.19706, 0.07941, 0.30000, 0.41029, 0.01324]
Sp100:	-0.01826	572	[0.02448, 0.28846, 0.28147, 0.39161, 0.01399]
Rela:	0.38823	92	[0.22826, 0.29348, 0.14130, 0.28261, 0.05435]
Ldhb:	0.00141	194	[0.10309, 0.13402, 0.34536, 0.41237, 0.00515]
Eomes:	0.64041	16	[0.43750, 0.31250, 0.06250, 0.18750, 0.00000]
Zfp292:	0.13621	313	[0.12460, 0.55591, 0.09904, 0.21086, 0.00958]
Prdm1:	0.36080	384	[0.29948, 0.08854, 0.24740, 0.35417, 0.01042]
Atf2:	0.13604	141	[0.12057, 0.24823, 0.30496, 0.29787, 0.02837]
Il12rb2:	0.21590	187	[0.18182, 0.20321, 0.30481, 0.29412, 0.01604]
Egr1:	0.46150	233	[0.27468, 0.27897, 0.23176, 0.21459, 0.00000]
Id2:	0.39309	117	[0.27350, 0.20513, 0.21368, 0.29060, 0.01709]
Lef1:	0.44284	100	[0.27000, 0.28000, 0.18000, 0.26000, 0.01000]
Arid4b:	-0.10667	636	[0.05660, 0.11792, 0.38836, 0.42610, 0.01101]
Fzd6:	0.17731	199	[0.13568, 0.26131, 0.26131, 0.32663, 0.01508]
Foxp1:	0.08284	153	[0.11111, 0.20915, 0.32680, 0.33333, 0.01961]
Id3:	0.48544	46	[0.39130, 0.08696, 0.21739, 0.30435, 0.00000]
Fzd3:	0.24278	139	[0.19424, 0.20144, 0.22302, 0.35971, 0.02158]
Foxm1:	0.47022	56	[0.64286, 0.07143, 0.14286, 0.14286, 0.00000]
Nr3c1:	-0.13356	375	[0.05067, 0.09067, 0.31200, 0.53333, 0.01333]
Irf9:	0.55206	96	[0.37500, 0.17708, 0.23958, 0.19792, 0.01042]
Tox:	0.22440	37	[0.18919, 0.43243, 0.21622, 0.13514, 0.02703]
Hmgb2:	0.29051	95	[0.23158, 0.18947, 0.25263, 0.32632, 0.00000]
Oxnad1:	0.68020	59	[0.42373, 0.25424, 0.13559, 0.18644, 0.00000]
Sp140:	0.67002	33	[0.51515, 0.12121, 0.12121, 0.24242, 0.00000]
Sub1:	0.07193	134	[0.11194, 0.19403, 0.30597, 0.37313, 0.01493]
Yy1:	0.56775	25	[0.60000, 0.20000, 0.08000, 0.12000, 0.00000]
Lrp1:	0.36640	72	[0.30556, 0.16667, 0.37500, 0.13889, 0.01389]
Ep300:	-0.09750	24	[1.00000, 0.00000, 0.00000, 0.00000, 0.00000]
P2rx7:	0.66479	43	[0.46512, 0.13953, 0.11628, 0.25581, 0.02326]
Runx3:	-0.08377	10	[1.00000, 0.00000, 0.00000, 0.00000, 0.00000]
Rad21:	0.47584	10	[0.40000, 0.40000, 0.10000, 0.10000, 0.00000]
Klf2:	-0.06772	5	[1.00000, 0.00000, 0.00000, 0.00000, 0.00000]
Ezh2:	0.33319	26	[0.23077, 0.34615, 0.15385, 0.26923, 0.00000]
Myb:	0.59728	31	[0.41935, 0.22581, 0.19355, 0.16129, 0.00000]
Eef2:	-0.00000	1	[1.00000, 0.00000, 0.00000, 0.00000, 0.00000]
Batf:	0.24062	6	[0.66667, 0.00000, 0.16667, 0.00000, 0.16667]
Tbx21:	-0.03588	2	[1.00000, 0.00000, 0.00000, 0.00000, 0.00000]
Rps6:	-0.03588	2	[1.00000, 0.00000, 0.00000, 0.00000, 0.00000]

```
import sys
import numpy as np
import scanpy as sc

# usage: python3 calc_score.py <training file path>

states = ["progenitor", "effector", "terminal exhausted", "cycling", "other"]

# these parameters should be changed to desired values
Q = [0.5, 0.3, 0.0, 0.2, 0.0] # desired cell state proportion vector
w = [1/0.0675, 1/0.2097, 1/0.3134, 1/0.3921, 0] # use same weights as challenge 2 part B

################################################################################################
# input:
#   s0 - proportion vector for the unperturbed samples
#   si - proportion vector the gene i perturbed samples
#   ni - the number of gene i perturbed samples
# output: the score

def calc_score(s0, si, ni):
  if ni <= 0: print("No samples!"); return -1
  num = sum(w[j]*abs(Q[j] - si[j]) for j in range(5))
  den = sum(w[j]*abs(Q[j] - s0[j]) for j in range(5))
  if abs(den) < 0.0001: print("Unperturbed statistic matches desired state!"); return -1
  return (1 - num/den) * (1 - 1/ni**0.5)

################################################################################################

# read the training file
print('Reading "sc_training.h5ad"')
adata = sc.read_h5ad(sys.argv[1])
conditions = adata.obs["condition"].unique().to_numpy()
counts = adata.obs['condition'].value_counts()

# n = the number of gene i perturbed samples
n = [counts[condition] for condition in conditions]

# s = the state proportion vectors for each gene in conditions
s = []
df = adata.obs[['condition', 'state']]
props = df.groupby('condition', as_index=False).value_counts(normalize=True)
for condition in conditions:
  condProps = props[props['condition'] == condition]
  s.append([condProps[condProps['state'] == state].values[0][2] for state in states])
s0 = s[np.where(conditions == "Unperturbed")[0][0]]

# output scores for each gene in conditions
print("       gene:    score    n                                                    s")
for i in range(len(conditions)):
  score = calc_score(s0, s[i], n[i])
  print("%11s:%9.5f%5d [%0.5f, %0.5f, %0.5f, %0.5f, %0.5f]" \
        %(conditions[i], score, n[i], s[i][0], s[i][1], s[i][2], s[i][3], s[i][4]))
```