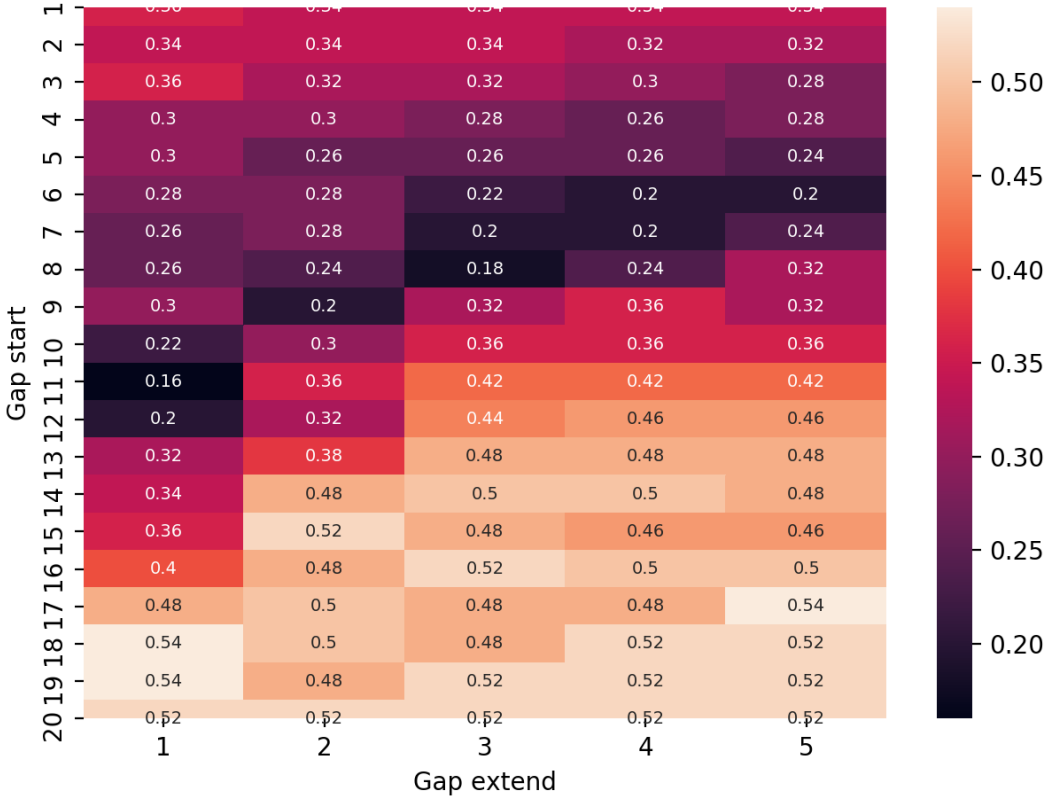
BMI203: HW3

Laura Shub

February 2020

1 Smith-Waterman

Q1. *Consider the false positive rate (proportion of negative pairs with scores that exceed a score threshold) when the true positive rate (proportion of positive pairs with scores above the threshold) is 0.7. What's the best false positive rate that you can achieve with varying both gap opening (from 1 to 20) and extension penalties (from 1 to 5) with the BLOSUM50 matrix? What is the best gap penalty combination?*

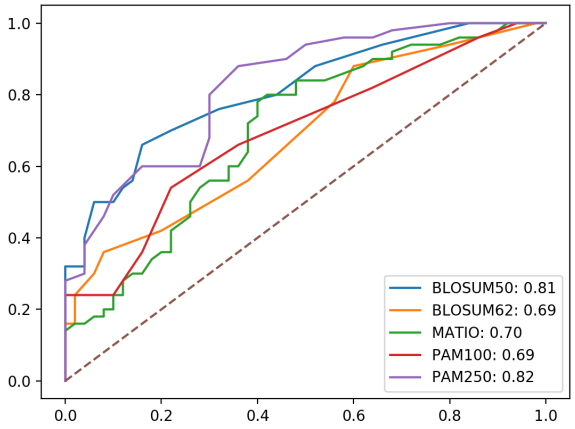


The best false positive rate that I achieved using a true positive rate of 0.7 was 0.16. This was observed with a gap opening penalty of 11 and a gap extension penalty of 1.

Q2. *Using the gap penalties you determined from question 1, which of the provided scoring matrices performs the best, in terms of false positive rate (at a true positive rate of 0.7)? What are the performance rates of each of the matrices? Create a Receiver Operator Curve (ROC) graph which shows the fraction of true positives on the Y axis and the fraction of false positives on the X axis. Include on the graph data for each of the provided matrices. Please take care to make your ROC graphs square,*

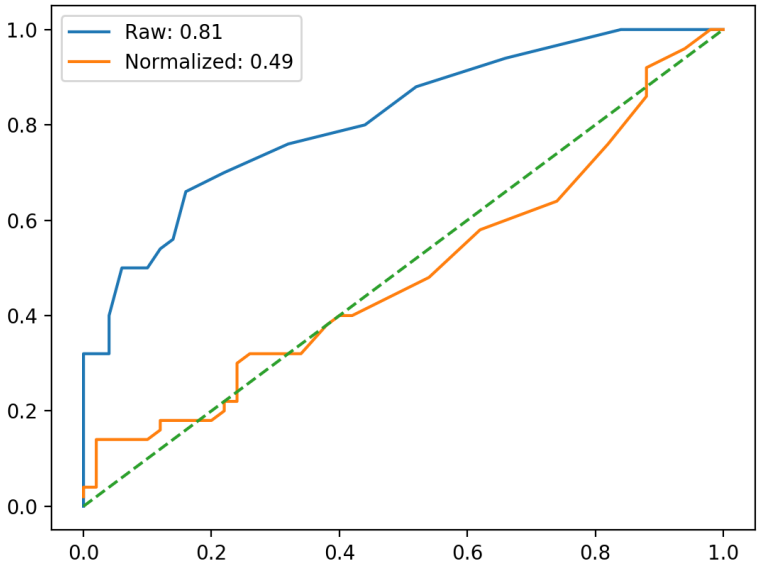
with both X and Y axes limited to the range [0:1]. Note, you can download ROC code from here: http://www.jainlab.org/Public/ucsf-roc.zip. It is not guaranteed to be bug free but it might save you some time.

matrix	FP rate
BLOSUM50	0.16
BLOSUM62	0.5
MATIO	0.38
PAM100	0.36
PAM250	0.3



The FP rate at a true positive rate of 0.7 is shown above for the 5 matrices provided using a gap open penalty of 11 and a gap extend penalty of 1. Using these values and the provided pairs, the matrix with the best FP rate was BLOSUM50, which makes sense as the penalties were optimized on this matrix. The second best was PAM250, which actially had a higher AUC on the ROC curve.

Q3. *How does the performance change if you normalize the Smith-Waterman scores by the length of the shortest sequence in a pair (i.e. divide the raw score by the min length)? Show the ROC curves for your best matrix and for the same matrix with normalized scores. Are the false positive rates better or worse? Why do you think this is so?*



Shown above is the ROC curve for BLOSUM50 with both raw and normalized scores. The AUC drops significantly when divided by the length of the shorted sequence. This is likely because longer sequences are more likely to have some good local alignments, even by chance, compared to shorter sequences.

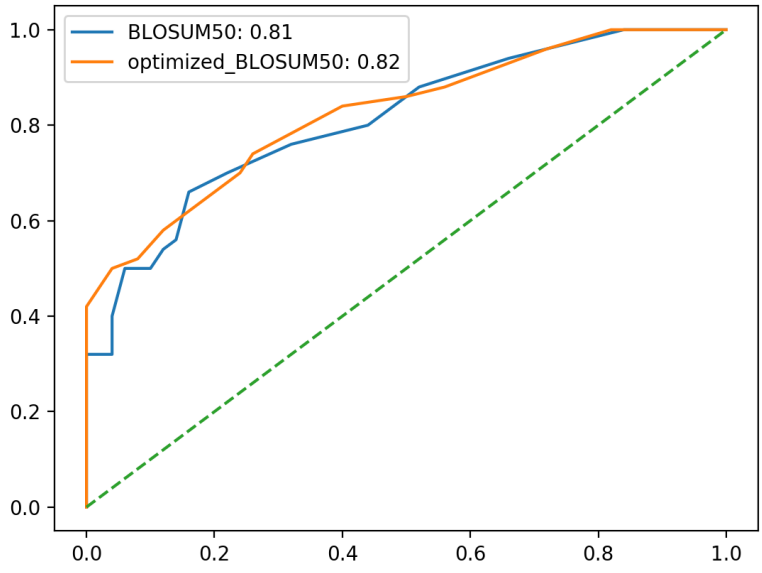
2 Optimization

Q1. *Devise an optimization algorithm to modify the values in a starting score matrix such as to maximize the following objective function: sum of TP rates for FP rates of 0.0, 0.1, 0.2, and 0.3. The maximum value for the objective function is 4.0 (where you are getting perfect separation of positive and negative pairs even at the lowest false positive rate). You should use the gap and extension penalties derived from Part 1. Remember, you must maintain symmetry in your matrix. You can make use of real-valued scores in the matrices if desired (this is probably a good idea).*

See Q7, and functions optimize_score_matrix, genetic_loop, mutate_matrix, loss_function, located in smith_waterman/__main__.py

Q2. *Beginning from the best matrix from above (that which produced the alignments), run your optimization algorithm to maximize the fitness of the new matrix. How much improvement do you see in the fitness? Show the full ROC curves for the original matrix and the optimized matrix. What happens when you now realign the sequences using the new matrix and rescore? Show the new ROC curve following realignment on the same graph as above. Qualitatively discuss how your matrix and your alignments change following optimization.*

Using the BLOSUM50 matrix, the initial objective function is 2.26. I optimized until I achieved a matrix with a score greater than 2.5. After 25 generations, a matrix was generated that had an objective score of 2.52. However, you could potentially achieve a more optimal matrix by setting a higher goal or running for a greater number of generations. The ROC curve for this optimized matrix as well as the initial BLOSUM50 matrix are shown below. The optimized matrix ensures a lower FP rate at lower thresholds, which makes sense given the optimization function.

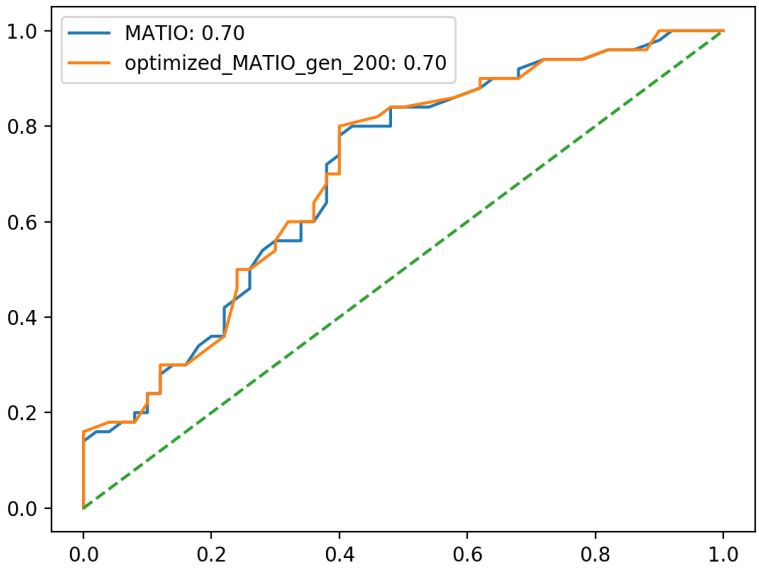


Looking at the alignments, overall the scores for each are higher than the basic BLOSUM50 matrix, with an average increase in score per alignment of 8.8. However, the scores for the positive pairs

increased more overall (12.7) than those for the negative pairs (5.0), leading to a higher true positive rate at low false positive rates. The optimized matrix is likely more permissive of certain mismatches, leading to longer alignments and correspondingly higher scores.

Q3. *Beginning from the MATIO matrix, but using the same initial sequence alignments, re-run the optimization. Show the same ROC plots as for (2). Discuss the relationship between the results you see here and the results you saw for (2).*

The initial score of the objective function using the MATIO scoring matrix was 1.32. After 200 generations, the best score achieved by an optimized matrix was 1.38, which was first observed in the 36th generation and was never improved further. This limited improvement is also seen from the ROC curve, where the optimized matrix has better enrichment at low FP thresholds, but is not overall very much better than the initial MATIO matrix.



Unlike BLOSUM50, the scores decreased overall using the optimized MATIO (-1.5), though similarly the negative pair scores decreased more than the positive pairs (-2.6 and -0.4 respectively). Part of this is likely that the MATIO matrix was very permissive of wrong pairs to begin with, leading to high initial scores.

Q4. *Describe your optimization algorithm briefly. How might you improve it?*

My optimization is an evolutionary algorithm. Starting from a base matrix (BLOSUM50 or MATIO), the remaining members of the current "population" are created by selecting new scores from a normal distribution centered on the value in the starting matrix. The score of each matrix in the population is taken using the described objective function, and the two matrices with the best scores are saved. These are then combined to make "offspring" by randomly selecting amino acids and switching the scores for those amino acids. The offspring are further modified by mutating a small number of scores, again selecting from within a normal distribution centered on the current score. The two offspring are then added to the population, replacing the matrices that scored the worst in the previous generation. This is repeated until we find a matrix that scores better than the goal, or if no goal is provided, better than the starting matrix.

There are a few ways that this optimization algorithm could potentially be improved. For one, I did not test a variety of mutation rates or sigmas for generating new scores during random initialization and mutation. Another similar change could be adding in a simulated annealing component, wherein

the mutation rate and the range of the normal distribution both decrease as the generation increases. Finally, another possible approach would be to introduce a Monte Carlo minimization component, keeping a worse matrix than one of the parents with some small probability, in order to avoid getting stuck in local minima, as I saw when optimizing the MATIO matrix.

Q5. *What would be required in order to make a convincing case that an optimized matrix will be of general utility and will actually be beneficial for people to use in searching databases?*

This matrix was optimized on a small training set of sequences. In order to prove its wider application, and that it is not overfitted,the matrix should be used on a test set with sequences not present in the training set.

Github: `https://github.com/laurashub/BMI203HW3`.