Snow Naing
BMI203
HW-3 re-do: Smith-waterman algorithm implementation and optimization algorithm

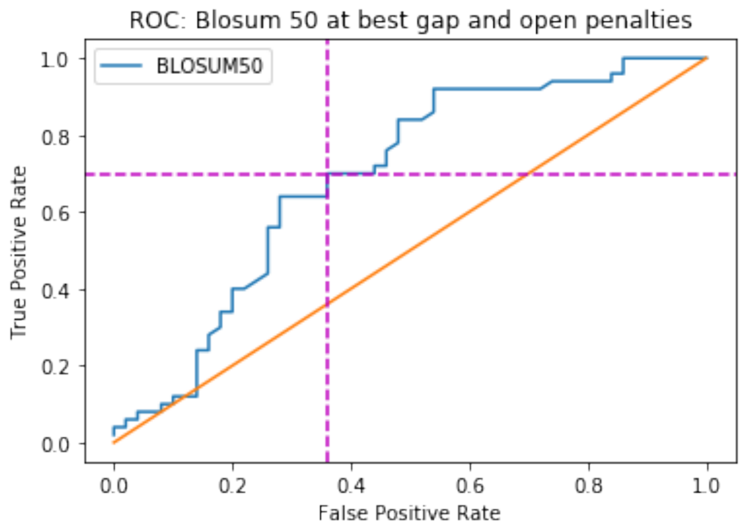
Part 1

Smith-Waterman Algorithm Implementation

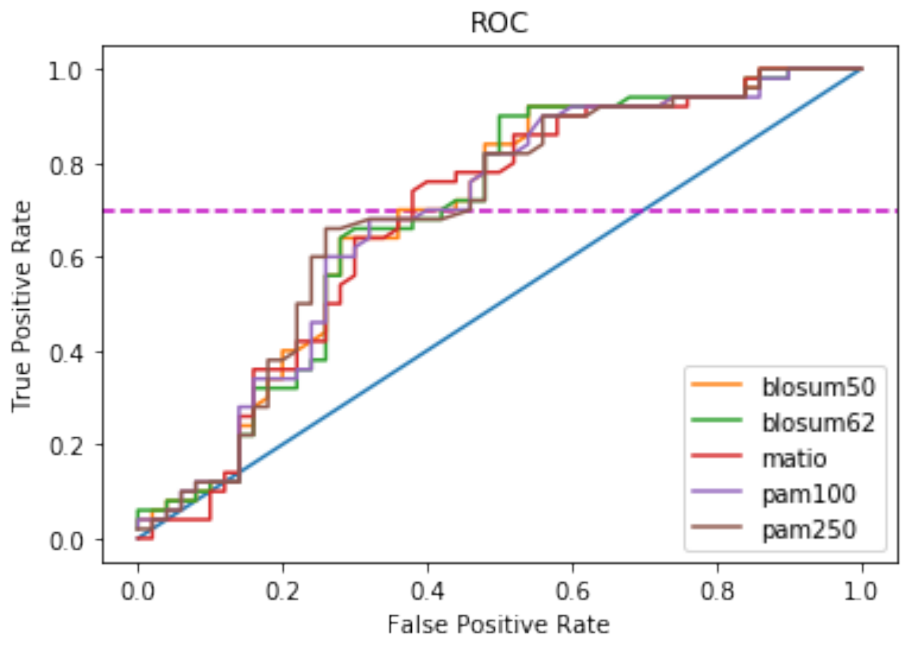
My codes for smithwaterman can be found in "alignment/smithwaterman.py". Given two sequences, penalty scores for opening a gap and extending a gap along with a substitution matrix, my smith-waterman algorithm outputs an alignment score matrix. The output alignment score matrix is built based on the calculation of insertion, deletion or match/mismatch scores of left, up and diagonal wells in the matrix.

Questions

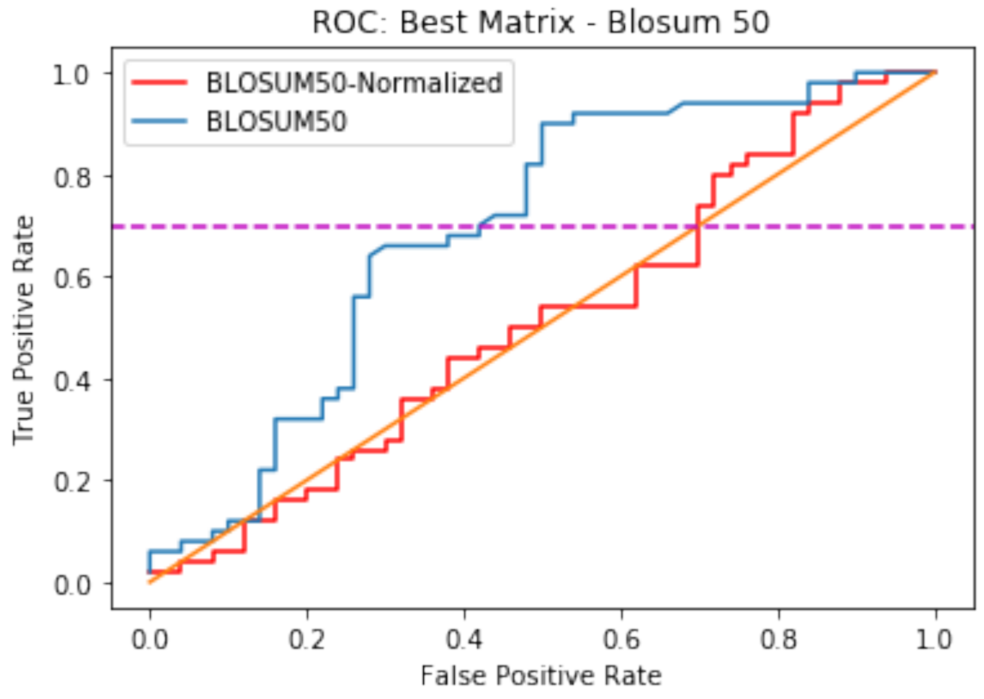
1. Using every possible gap penalties, I found that at true positive rate of 0.7, the best false positive rate achieved is 0.36 at best open and extension gap penalty scores at 1 and 1, respectively. ROC graph drawn using best gap penalty combination with BLOSUM50 matrix is shown below.



2. Using the best gap penalties combination, Blosum50 matrix performs the best as it has the lowest false positive rate at a true positive rate of 0.7. The false positive rates at a true positive rate of 0.7 for blosum50, blosum62, matio, pam100 and pam250 are 0.36, 0.42,0.38, 0.39 and 0.43 respectively. This information can also be observed from the graph below.



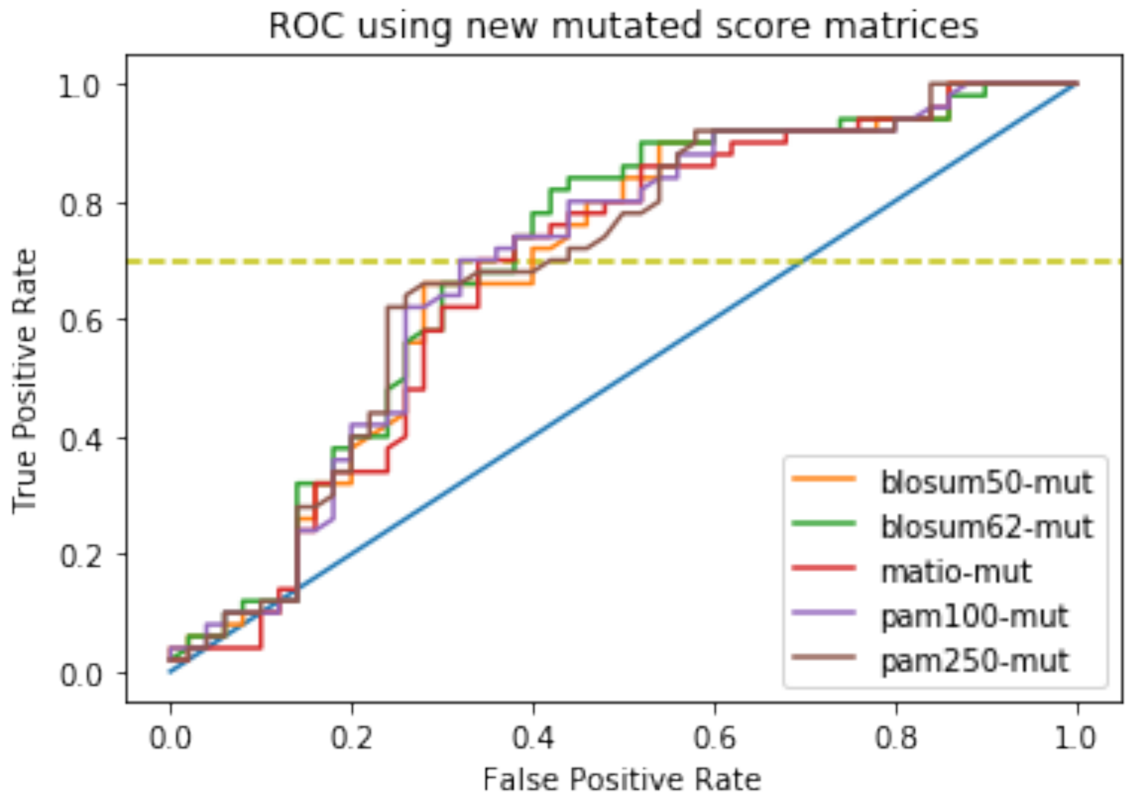
3. False positive rate performance is worse when the Smith-Waterman scores are normalized by the length of the shorter sequence in a pair. As seen in the ROC graph below, the best performing scoring matrix – BLOSUM50 outputs very high false positive rates when normalized. Local alignment scores show how similar two sequences are, which is independence of the length. When normalizing the score by dividing the similarity score by the shorter length of the sequence, the score just gets minimized and hence, poorer performance.



Part 2 - Optimization

For my optimization algorithm to modify the scoring matrix, I randomly switched 4 indices of the scoring matrices, giving a new score for those switched amino acids.

When calculating the false positive rates using the new matrices, I found that the best matrix is now the new PAM100 matrix. To optimize, I should think more about the chemical property of the amino acid instead of randomly switching and assigning new values. All the new matrices are stored in matrices_mut folder.



My optimization algorithm needs a lot of work. For the optimized matrix to be of general utility, the score output has to make sense biologically. For example, we can give more match score values to certain amino acids with higher active site functionality and less mismatch score if the two amino acids are of the same biochemical functionality. We can maybe also try to get a feedback loop and see what combination of amino acids with match/mismatch score will get us the best false positive rates. This way, we can have a scoring matrix that has better biological meanings rather than randomly assigning scoring values.