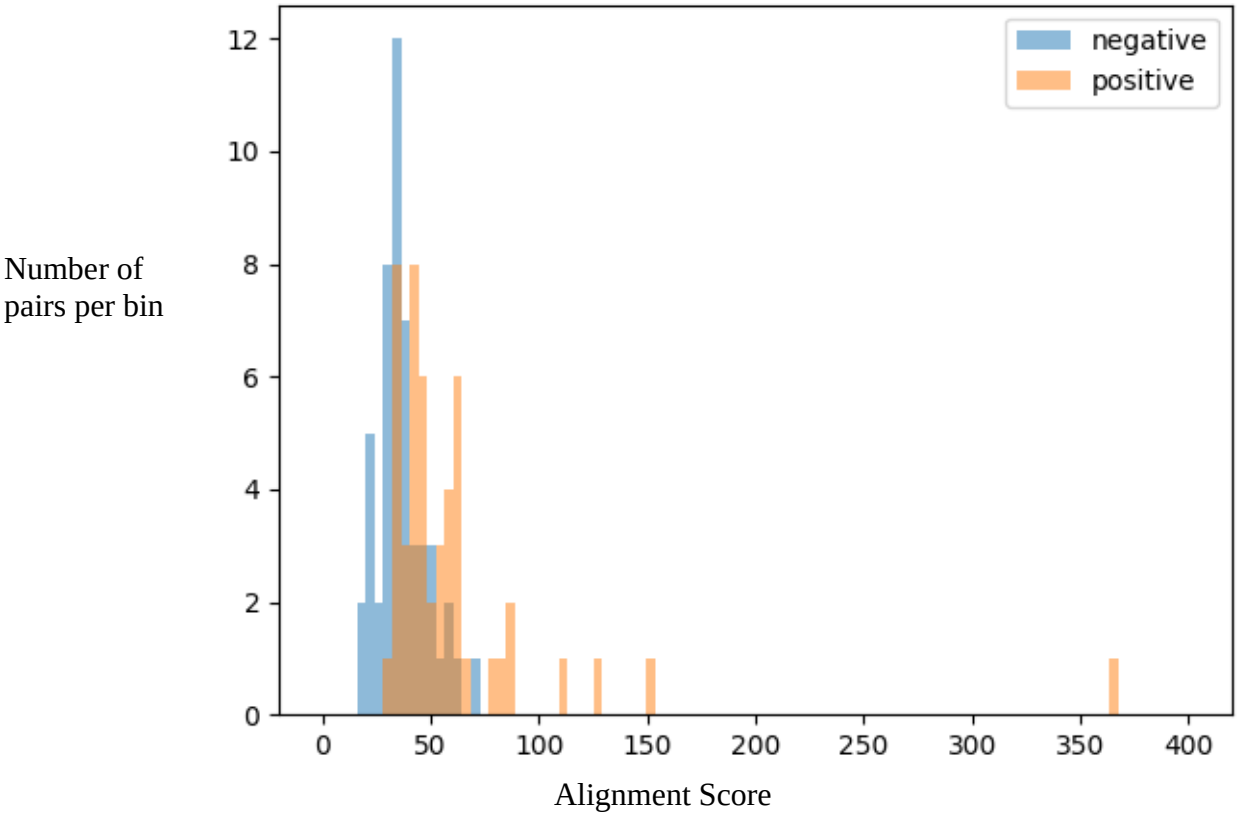
Benjamin Wheeler
BMI 203
Project 1 Write Up

1.)For both Smith-Waterman and Needleman-Wunsch algorithms: a) What are the parameters and variables required for algorithm initialization, execution, and termination? b) What quantities are returned? c) What is the runtime complexity?

1.)a) Both algorithms require the following variables: two sequences to be aligned. In addition the following parameters must be provided: a matrix defining the score for a match between two given amino acids, the gap opening penalty, and the gap extension penalty.
b) The algorithms will return in each case the ideal alignment(s) and the score associated with that alignment
c)The complexity is proportional to the product of the length of each string to be aligned. O(N*M) where N and M are the lengths of the sequences to be aligned.
2.) What functionalities in initialization, execution and termination are shared between these algorithms? Which are not shared?
The algorithms share the initialization of defining the sequences to be aligned and the score matrices for a given character pair. Howevever, when initializing the result matrix the first column and row for the local(smith-waterman) algorithm negative values for the gaps are implemented. For the global(Needleman-wunsch) algorithm zeros values are assigned for the gaps.
In execution the algorithms are exceedingly similar in that the matrix is stepped through and a score is assigned to each box based upon the conditions of the algorithm. The essential difference here is that the local alignment method does not allow negative values but instead places zeros where the other options would be negative. In traceback they are also exceedingly similar. The differences are in the initiation and termination of step back, but the act of following the path determined during step through is the same. In initiating step back smith-waterman finds the highest value contained in the matrix, and needleman-wunsch always starts at the bottom right cell of the matrix. In the case of smith-waterman the step back continues until a zero value is reached, while in needleman-wunsch the step back is carried on until the top left corner of the matrix is reached.
3.) In a sense linear gap penalty is a special case of affine gap penalty in which there is no penalty for gap opening. To say it differently there is “memory” involved in affine gap alignment. To implement linear gap alignment, a single gap penalty is referenced in determining whether to open or extend a gap. However, in affine gap alignment two new matrices must be used to track where the optimal preceding sequence contains gaps. Two are needed to account for gaps in either sequence 1 or sequence 2.
4.) see readme.md

Part 2 Questions
1.) I would describe this distribution as a chi-squared distribution. Many of the scores fall into a tight distribution on the lower end of the score with a long tail describing the very high scoring matches.

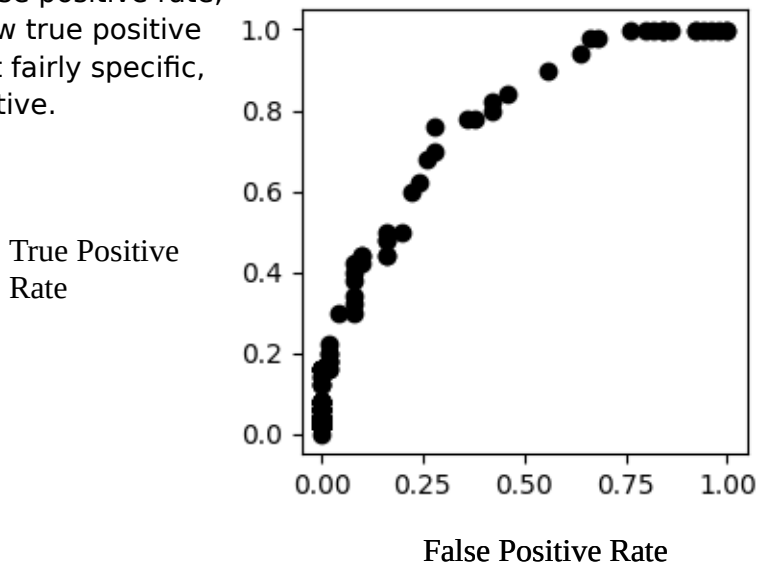



2.) The threshold value is 49.13, as determined by taking the average of all the scores. The confusion matrix is:

| | |
|---|---|
| 42 | 8 |
| 28 | 22 |

This suggests that the algorithm has a fairly low false positive rate, but also a fairly low true positive rate. This makes it fairly specific, but not very sensitive.

3.)



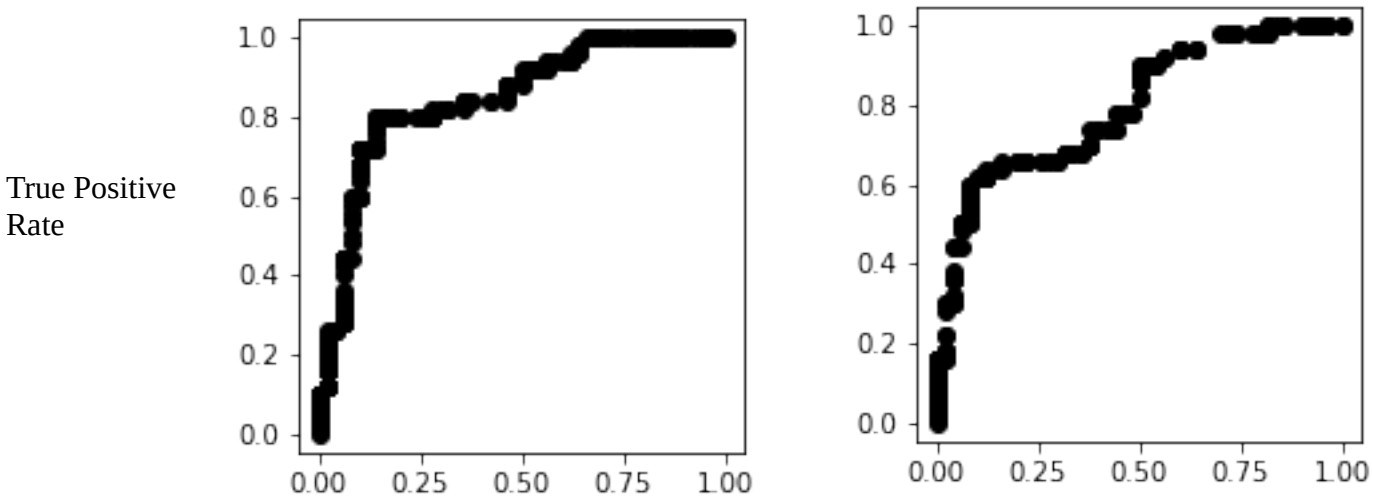
4.) The AUROC is 0.7858 for these parameters. The ideal value for this metric is 1 and the value for this metric in a random algorithm is 0.5. So this algorithm performs better than chance, but not “ideally”. This value alone is not sufficient to make judgments about the algorithm. Ultimately, the user will have to set the threshold of their choosing and this value is a proxy for the trade offs they will have to weigh. Further, it is best used in refinement of the algorithm to compare and tune parameters use, to achieve a more ideal state.

5.) For this optimization I found a gap opening penalty of 4 and a gap extension penalty of 4 to provide the highest AUROC value. These value indicate that gap extension is almost as costly as gap opening. Further, this gap opening penalty is lower than assumed a priori and very close to many mismatch amino acid penalties. This suggests that the likelihood of an indel is fairly high, but the gaps created were not very large.

6.)

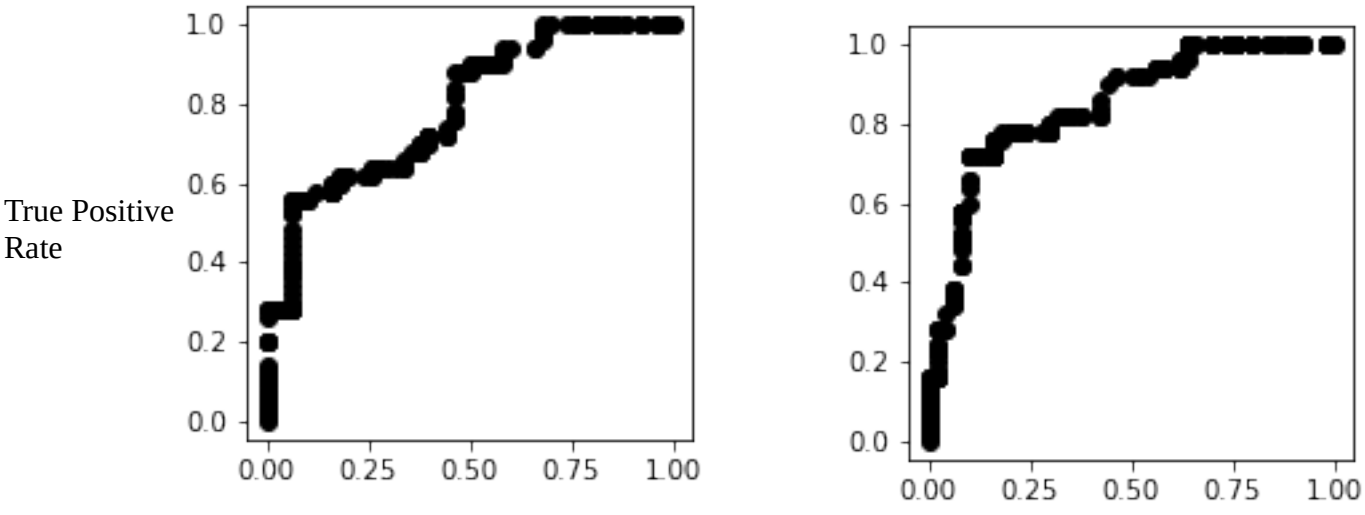
Blosum50

Blosum62




PAM100

PAM250



7.) By AUROC the BLOSUM50 matrix performs the best.

8.) For the sequences provided the best algorithm is one which accounts for a greater amount of evolutionary time between the sequences. Further, when the cost for indels is lesser, a better alignment is achieved. This suggests that the sequences have undergone frequent mutations, and have diverged significantly. This can also be seen by the lack of good separation between the negative and positive pairs.