Programming Assignment 1

Pocket Algorithm

1. Refer to the documentation, what is the functionality of the tol parameter in the Perceptron class? (2 marks)

   The tol parameter stand for the 'tolerance' which essentially can be considered as the stopping criteria. The process halts when there is no improvement in accuracy (when you're close to the desired value), therefore reaching tol.

2. If we set max iter=5000 and tol=1e-3 (the rest as default), does this guarantee that the algorithm will pass over the training data 5000 times? If not, which parameters (and values) should we set to ensure that the algorithm will pass over the training data 5000 times? (2 marks)

   No, this is because the algorithm may stop before making 5000 iterations. As mentioned previously, if the accuracy isn't changing and therefore the misclassifications are repeatedly zero then the tol value is getting nearer - resulting in the process to be halted. In order to ensure that the algorithm passes over the training data 5000 times, tol should be equal to None. The algorithm will make 5000 iterations regardless of accuracy or misclassifications.

3. How can we set the weights of the model to a certain value? (2 marks)

   The weights of a model can be set to a certain value using the 'coef_' attribute of the Perceptron class. Weights can be assigned to its subsequent features via this attribute.

4. How close is the performance (through confusion matrix) of your NumPy implementation in comparison to the existing modules in the scikit-learn library? (2 marks)

   At times both the Numpy and scikit-learn library produced an exact match via the confusion matrix. However, for most test cases performed, either the first or second row of both matrices were the same while the other was vastly different.

Linear Regression

1. When we input a singular matrix, the function linalg.inv often returns an error message. In your fit LinRegr(X train, y train) implementation, is your input to the function linalg.inv a singular matrix? Explain why. (2 marks)

In fit LinRegr(), the input is not a singular matrix generally, and hence why it is able to have an MSE computed to be a value as 2838.2 for example whereas for subtestFn() we are inputting a singular matrix – which doesn't have an inverse that exists because of a lower number of linearly independent "n" in the nxm matrix size and determinant 0. This error spurs prior to the linalg.inv operation when we compute the dot product of Xt and X – since it results in a square matrix whose inverse is non-existent for subtestFn()'s test matrix.

2. As you are using linalg.inv for matrix inversion, report the output message when running the function subtestFn(). We note that inputting a singular matrix to linalg.inv sometimes does not yield an error due to numerical issue. (1 marks)

   The reported error message is "ERROR"

3. Replace the function linalg.inv with linalg.pinv, you should get the model's weight and the "NO ERROR" message after running the function subtestFn(). Explain the difference between linalg.inv and linalg.pinv, and report the model's weight. (2 marks)

   Model Weight: [2.0000000e-01 4.0000000e-01 -8.8817842e-16]

   The difference between (inv) and (pinv) is that (inv) directly computes the inverse of the square matrix/singular matrix whereas linalg.pinv - the pseudo inverse calculates an approximation/generalization of the singular matrix despite it being singular. This is why we have very large decimal values in the weight for example: 2.000000e-01.

4. How close is the performance of your implementation in comparison to the existing modules in the scikit-learn library? Place this comment at the end of the code file.

   Asnwer added in code file.