COMP 411, Homework 5: *LMS Regression and the Perceptron*
Due by 11:59pm on October 21, 2021

---

1. Least Mean Squares Regression

Problem 1. *Use Least Mean Squares Regression to make a prediction about a continuous-valued feature in your data set. Remember that LMS Regression assumes a linear relationship between inputs and outputs. Think about what features you should use as inputs: try using only a few features vs. many features. First use gradient descent to minimize the cost function for LMS regression, then use incremental/stochastic gradient descent. Also try different learning rates. Do you notice any difference in convergence speed or prediction accuracy (computed using root mean squared error, RMSE* $= \sqrt{\frac{1}{N}\sum_i^N (y_i - \hat{y}_i)^2}$*) on test data?*

Problem 2. *Using Scikit, train a ridge regression model (a regularized version of linear regression) using the same data as the previous problem. How does the performance differ on the testing data?*

Note that you can either use the closed form solver:

```
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha=1, solver="cholesky")
```

or use stochastic gradient descent (specifying the L2 regularizer):

```
sgd_reg = SGDRegressor(penalty="l2")

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html
```

You will want to normalize your input features in some way if the values are widely varying for different features.

2. Perceptron

Problem 3. *Use a Perceptron to make a prediction about a feature in your data set whose values can be split into two classes. Remember that a Perceptron's weights will only converge if the data is linearly separable. Think about what features you should use as inputs: again try using only a few features vs. many features. Use the perceptron algorithm descent to update weights. Do you notice any difference in convergence speed or prediction accuracy (computed using precision and accuracy) on test data?*

You may want to fix the maximum number of iterations to run or the maximum number of misclassifications allowed to determine the stopping point for your implementation. You will also want to normalize your input features in some way if the values are widely varying for different features.

Precision is defined as the fraction of guessed positives which were actually positive while recall is defined as the fraction of actual positives which were guessed as positive.

PROBLEM 4. *Try Scikit's perceptron implementation on your data and see how your results compare with the previous problem. See:*

```
sgd_reg = SGDClassifier(loss="perceptron", learning_rte="constant", eta0=1, penalty=None)
```

```
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
```

Notice the learning rate is 1 since the number of mistakes that the perceptrons makes isn't affected by the learning rate.

3. Feedback on slides

Let me know what you found easy and hard to understand on the slides. Any comments or criticisims are appreciated.

4. Thoughts on Other Topics to Cover Later in the Course (Optional)

If there are specific topics related to neural networks that you would like to see covered, let me know. If there are other topics you might be interested in seeing covered, also tell me, though it may be less likely that I can cover them.