SYSC4906 Introduction to Machine Learning Fall 2019

Assignment 1

Q1) Calculate the gradient of the following function:

$$f(x,z) \stackrel{\text{def}}{=} \sqrt{5x^3 + z^2 + 4xz + 11x + 5}$$

Q2) Create a python notebook which loads the `sklearn` breast cancer dataset (see `sklearn.datasets.load_breast_cancer`). This dataset has 2 classes of breast tumor biopsies: malignant (target=0) and benign (target=1). There are 569 samples (357 benign, 212 malignant) with 30 features each.

a) Split the data, using 75% for training and 25% for test. Make sure you use stratified sampling.

b) Train and test a logistic regression classifier. How accurate is your classifier?

c) Repeat part b), but using only the first two features from the dataset. Was the classifier accuracy impacted?

d) Using the 2-feature classifier from part c), create two subplots using the first two features from the data set.

i) On the first, plot the decision boundary and the training data. Use green for malignant (target==0) and blue for benign (target==1).

ii) On the second, plot the decision boundary and the test data. Use the same colours (blue/green), but highlight all misclassified test points (from either class) in red.

Q3) Linear regression. Download the file “Assig1Q3.csv” from CULearn under “Assignments”. The first column represents the X values, while the second column represents the Y values.

1. Plot the data

We are going to use linear regression to fit a linear and a quadratic model to these data. **Without using sklearn.linear_model** (or any other linear regression libraries), write your own python code to implement the least squares solution for linear regression. That is:

$$\beta = (X^T X)^{-1} X^T y$$

2. Assuming the model $y = mx + b$, use your code to best-fit the parameters m and b to the data. Report your optimal parameter values.

   Hints:

   a. *recall that you must create the ‘augmented’ feature vector X from the given x data (add a column of 1’s).*

   b. *look at numpy.T(), numpy.matmul(), numpy.dot(), and numpy.linalg.inv()*

3. Plot your line of best fit on top of the data
4. Calculate the sum of square residuals, or mean squared error, as in:

$$MSE(\beta) = \sum_{i=1}^{N} (y - X\beta)^2$$

5. Assuming the model $y = ax^2 + bx + c$, repeat steps 2-4 using this new model (i.e. estimate the optimal values for a,b,c; report those estimates; plot the line of best fit; report the MSE).
6. Briefly discuss which model would you prefer for these data?
7. Why is best-fitting the second (quadratic) model still considered <u>linear</u> regression?

Q4) Create a Jupyter Notebook based on `Lecture5.ipynb` to use `make_classification` to create a linearly separable dataset, with 2 classes, 2 informative features, 1000 samples per class, using a class_sep=2.0, and a random_state of 3. Generate some random noise of the same shape as your feature data, drawn from a **standard normal distribution** (see `numpy.random`) and a random_state of 2. Create four datasets: 1) no noise, 2) data + 0.5 * noise, 3) data + 1.0 * noise, and 4) data + 2.0 * noise.

i) For all four datasets, plot the data, labelling each (sub)plot by the degree of noise added (i.e. 0, 0.5, 1.0, and 2.0)

ii) For each dataset, create training and test data using a 70/30 train/test split (see `train_test_split`).

iii) For each dataset, train and test an SVM classifier with a polynomial kernel with degree=2, and C=1.0. Report the test score for each. How does prediction accuracy change with noise level?

iv) For a noise level of 0.5, train and test SVM classifiers using the following values for C: {0.001, 0.01, 0.1, 1, 10, 100}. Report the test accuracy for each. How does performance vary with C? Briefly describe what the `C` controls for sklearn.svc. *Hint: look at the documentation for sklearn.svc rather than the class notes here...*