50.007 Machine Learning, Fall 2021
Homework 1

Due 7 Oct 2021, 11:59 pm

This homework will be graded by TA Zhang, Qi

1. Classification [20 points]

Consider data points from a 2-d space where each point is of the form $x = (x_1, x_2)$. You are given a dataset with two positive examples: (1, 1) and (2, 2), and two negative examples (-1, 1) and (1, -1). For each of the following hypothesis spaces, find the parameters of a classifier (a member of the hypothesis space) that can correctly classify all the examples in the dataset, or explain why no such classifier exists.

(a) [10 points] Inside or outside of an origin-centered circle with radius r (r is the parameter).

(b) [10 points] Above or below a line through the origin with normal vector $\theta = (\theta_1, \theta_2)$ (or $[\theta_1, \theta_2]^T$).

(a) The training set is ((1, 1), +1), ((2, 2), +1), ((-1, 1), -1), ((1, -1), -1).

i. The decision boundary is given by $x : \|x\| = r$. The classifier h$_r(x)$ is:

$$h_r(x) = \begin{cases} +1, & \text{if } \|x\| > r. \\ -1, & otherwise. \end{cases} \tag{1}$$

Because (1, 1) and (-1, 1) are the same distance from the origin but have opposite labels, $h_r(x)$ will not be able classify the training examples correctly.

ii. A line through the origin with normal w. The resulting linear decision boundary is defined by $x : w \cdot x = 0$. We can define a classifier $h_w(x)$as:

$$h_w(x) = \begin{cases} +1, & \text{if } w \cdot x \geq 0. \\ -1, & \text{otherwise.} \end{cases} \tag{2}$$

Because the two negative examples belong to the decision boundary normal to $w = (1, 1)$ that includes the origin, it is not possible to separate all training examples accurately according to the above definition of the classifier. However, if you define the classifier as follows:

$$h_w(x) = \begin{cases} +1, & \text{if } w \cdot x > 0. \\ -1, & \text{otherwise.} \end{cases} \tag{3}$$

Then the answer is yes as the two negative points which are exactly on the decision boundary are classified correctly as negative. We will check your answer for understanding and award the points accordingly.

2. Linear classification [20 points]

Automatic handwritten digit recognition is an important machine learning task. The US Postal Service Zip Code Database (http://www.unitedstateszipcodes.org/zip-code-database/) provides 16×16 pixel images preprocessed from scanned handwritten zip codes (US zip codes are the analogues of Singapore postal codes). The task is to recognize the digit in each image. We shall consider the simpler goal of recognizing only two digits: 1 and 5. To simplify our task even further, let's consider only two features: intensity and symmetry. Digit 5 generally occupies more black pixels and thus have higher average pixel intensity than digit 1. Digit 1 is usually symmetric but digit 5 is not. By defining asymmetry as the average difference between an image and its flipped versions, and symmetry as the negation of asymmetry, we can get higher symmetry values for digit 1.

Write an implementation of the perceptron algorithm. Train it on the training set (`train_1_5.csv`), and evaluate its accuracy on the test set (`test_1_5.csv`). The training and test sets are posted on eDimension. csv stands for comma-separated values. In the files, each row is an example. The first value is the symmetry, the second is the average intensity, and the third is the label.

Note: please do NOT shuffle the data. Visit the instances sequentially in the training set when running the perceptron algorithm.

(a) [5 points] Run the perceptron algorithm with offset on the training data for 1 epoch (i.e., traversing the training set 1 time), report the θ, offset and accuracy on the test set.

(b) [5 points] Run the perceptron algorithm with offset on the training data for 5 epochs, report the θ, offset and accuracy on the test set.

(c) [10 points] Submit your code together with crystal clear instructions to run the code. The code must be ready to run code without requiring any changes. The TA will follow the instructions to run your code and grade accordingly.

Please check the ipynb/pdf.

3. Linear and polynomial regression [40 points]

For this exercise, you will experiment with linear and polynomial (in features) regression on a given data set. The inputs are in the file `hw1x.dat` and the desired outputs in `hw1y.dat`. Load the data and add a column vector of 1s to the inputs.

(a) [10 points] Write a function implementing the closed form linear regression formula discussed in class to obtain the weight vector θ and report it. Plot both the linear regression line and the data on the same graph. Write a function that will evaluate the training error in terms of empirical risk of the resulting fit and report the error.

(b) [20 points] Write a function to calculate the weight vector θ using batch gradient descent algorithm (where batch size is equal to number of training examples). Consider learning rate as $\eta = 0.01$, update θ and calculate the training error (empirical risk) for the batched data, 5 times. Report θ for the minimum training error, the minimum training error itself and plot the linear regression line along with the data on the same graph. Repeat the same with stochastic gradient descent algorithm for 5 epochs (i.e. random sampling with replacement for 5 x number of training examples).

(c) [10 points] Write a function called PolyRegress(x,y,d) which adds the features $x^2, x^3, ...x^d$ to the inputs and performs polynomial regression using closed form solution. Use your function to get a quadratic fit of the data. Plot the data and the fit. Report the training error. Repeat the same for 3rd order fit to 15th order fit. After which order fit does the error get worse?

Please check the ipynb/pdf.

4. Ridge regression [20 points]

In this problem, we will explore the effects of ridge regression on generalization. We will use `hw1_ridge_x.dat` as the inputs and `hw1_ridge_y.dat` as the desired output. Please note that a column vector of 1s is already added to the inputs. Recall from Lecture Notes 4, the optimal weight for ridge regression is given by

$$\hat{\theta} = \left(n\lambda I + X^T X\right)^{-1} X^T Y \tag{4}$$

To find a suitable value for λ, we will set aside a small subset of the provided data set for estimating the test loss. This subset is called *validation set*, which we use to compute *validation loss*. The remainder of the data will be called the *training set*. Let the first 10 entries of the data set be the validation set, and the last 40 entries be the training set. Concatenate their features into matrices vX and tX, and their responses into vectors vY and tY.

(a) [10 points] Write a function *ridge_regression*(tX, tY, l) that takes the training features, training responses and regularizing parameter λ, and outputs the exact solution θ for ridge regression. Report the resulting value of θ for $\lambda = 0.15$.

(b) [10 points] Use the following code to plot graphs of the validation loss and training loss as λ varies on logarithmic scale from $\lambda = 10^{-5}$ to $\lambda = 10^{0}$. Write down the value of λ that minimizes the validation loss.

```
import matplotlib.pyplot as plt

tn = tX.shape[0]
vn = vX.shape[0]
tloss = []
vloss = []
index = -np.arange(0,5,0.1)

for i in index:
    w = ridge_regression(tX,tY,10**i)
    tloss = tloss+[np.sum((np.dot(tX,w)-tY)**2)/tn/2]
    vloss = vloss+[np.sum((np.dot(vX,w)-vY)**2)/vn/2]
```

plt.plot(index,np.log(tloss),'r')
plt.plot(index,np.log(vloss),'b')

Please check the ipynb/pdf.