Introduction to Machine Learning

Problem Set 2: Regression and Gradient Descent

Problem 1 (20 points)

Suppose we have a dataset giving the living prices, number of rooms, and prices of a certain number of houses in Portland, Oregon. Below is a selection of the dataset[1].

Living Area (ft^2)	No. of Bedrooms	Prices (in $1000s)
1600	3	330
2400	3	369
1416	2	232
3000	4	540

Table 1: Housing Prices of Portland, Oregon

Using this dataset, we wish to model the prices of houses in Portland, as a linear function of the sizes of their living areas and the number of bedrooms, $h_w(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2$. Here $x_1^{(i)}$, $x_2^{(i)}$ is the living area and number of bedrooms of the *i*th house in the training set respectively, $y^{(i)}$ is the price of the *i*th house in the training set.

1. Under this formulation, what is the cost function $J(\mathbf{w}) = J(w_1, w_2, w_3)$ for this data set? Your cost function should only have real numbers and weights w_1, w_2, w_3.

2. Suppose we are going to apply gradient descent to minimize $J(\mathbf{w})$. Assume we use $\alpha = 0.1$ and we initialize the values of w_0, w_1, and w_2 as 0s. After the first iteration, what are the new values of w_0, w_1, and w_2 ?

Problem 2 (60 points)

Suppose you are selling your house and you want to know what a good market price would be. One way to do this is to first collect information on recent houses sold and make a model of housing prices. The file housing.txt contains a training set of housing prices in Portland, Oregon. The first column is the size of the house (in square feet), the second column is the number of bedrooms, and the third column is the price of the house.

A. Feature normalization

[1] Data source: http://cs229.stanford.edu/

By looking at the values in housing.txt, you will note that house sizes are about 1000 times the number of bedrooms. When features differ by orders of magnitude, first performing feature scaling can make gradient descent converge much more quickly.

(a) Given a set of numbers $x_1,\ldots,x_m$, write down the equations for the mean and standard deviations of these numbers.

The standard deviation is a way of measuring how much variation there is in the range of values of a particular feature (most data points will lie within 2 standard deviations of the mean); this is an alternative to taking the range of values (max-min).

(b) Write a python function that takes in a list of real numbers and returns the mean and standard deviation for that list.

(c) Write a program that takes as input housing.txt and creates a file called normalized.txt. To create normalized.txt, subtract the mean value of each feature from the dataset. After subtracting the mean, additionally scale (divide) the feature values by their respective standard deviations.

Implementation Note: When normalizing the features, it is important to store the values used for normalization - the mean value and the standard deviation used for the computations. After learning the parameters from the model, we often want to predict the prices of houses we have not seen before. Given a new x value (living room area and number of bedrooms), we must first normalize x using the mean and standard deviation that we had previously computed from the training set.

B. Gradient Descent to Find Weights (Parameters)

Our hypothesis (also called the model) will take the form $y = f(x_1,x_2) = w_0 + w_1x_1 + w_2x_2$ where x_1 is the normalized size of the house, x_2 is the normalized number of bedrooms, and y is the predicted price of the house. In this problem, your goal is to find the values of w_0, w_1, and w_2 that minimize the sum of the squared errors (MSE).

(a) Suppose there are m examples. Write down the formula for the loss function $J(\mathbf{w})$ using the sum of the squared errors. Be sure to include a $1/2m$ term.

(b) Implement gradient descent to find the values w_0, w_1, and w_2 that minimize $J(\mathbf{w})$. Apply your code to the normalized data set using the learning rates $\alpha = 0.01, 0.1, 0.3$.

A good way to verify that gradient descent is working correctly is to look at the value of $J(\mathbf{w})$ and check that it is decreasing with each step. Assuming you have implemented gradient descent correctly and your learning rate is not too big, your value of $J(\mathbf{w})$ should never increase, and should converge to a steady value by the end of the algorithm. Plot $J(\mathbf{w})$ for 10, 20,30,40,50, 60,70, 80 iterations for each of your α values.

(d) Do the same for learning rates $\alpha = 0.05$ and 0.3. Comment on which of the three learning rates gives the best result.

Implementation Note: If your learning rate is too large, J(**w**) can diverge and `blow up', resulting in values which are too large for computer calculations.

C. Predicting housing prices

You will now use the **w** you obtained in Part 2 to predict the housing prices. Predict the price of a house with 1650 square feet and 3 bedrooms. Don't forget to normalize your features when you make this prediction!

D. Stochastic Gradient

Using a learning rate of $\alpha = 0.1$ to find the values w_0, w_1, and w_2 using stochastic gradient descent (SGD). Make three passes through the data set, and provide J(**w**) after each pass. For the second and third passes, randomly shuffle the data set. Comment how the J(**w**) you obtain with SGD with 3 passes compares with the J(**w**) you obtain gradient descent with 80 passes. Also comment on the computation time of the two approaches.

Problem 3 (20 points)

In order to sharpen your skills with vector algebra, rigorously prove that the perceptron algorithm will have at most R^2/γ^2 mistakes. Justify all steps in your proof.