Introduction to Machine Learning

Problem Set 2: Regression and Gradient Descent

Xinyue Chen NetID: xc1305

**Problem 1**

Handwritten in “writeup\_supplement\_xc1305.pdf”

**Problem 2**

1. **Feature normalization**

(a) Given a set of numbers *x*1*, …, xm*, write down the equations for the mean and standarddeviations 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).

Handwritten in “writeup\_supplement\_xc1305.pdf”

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

standard deviation for that list.

Written in “code\_forP2.A.(b,c)-normalization.py”

(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 Notes*: 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.

Written in “code\_forP2.A.(b,c)-normalization.py”

**B. Gradient Descent to Find Weights (Parameters)**

(a) Suppose there are m examples. Write down the formula for the loss function J(w) using

the sum of the squared errors. Be sure to include a 1/2m term.

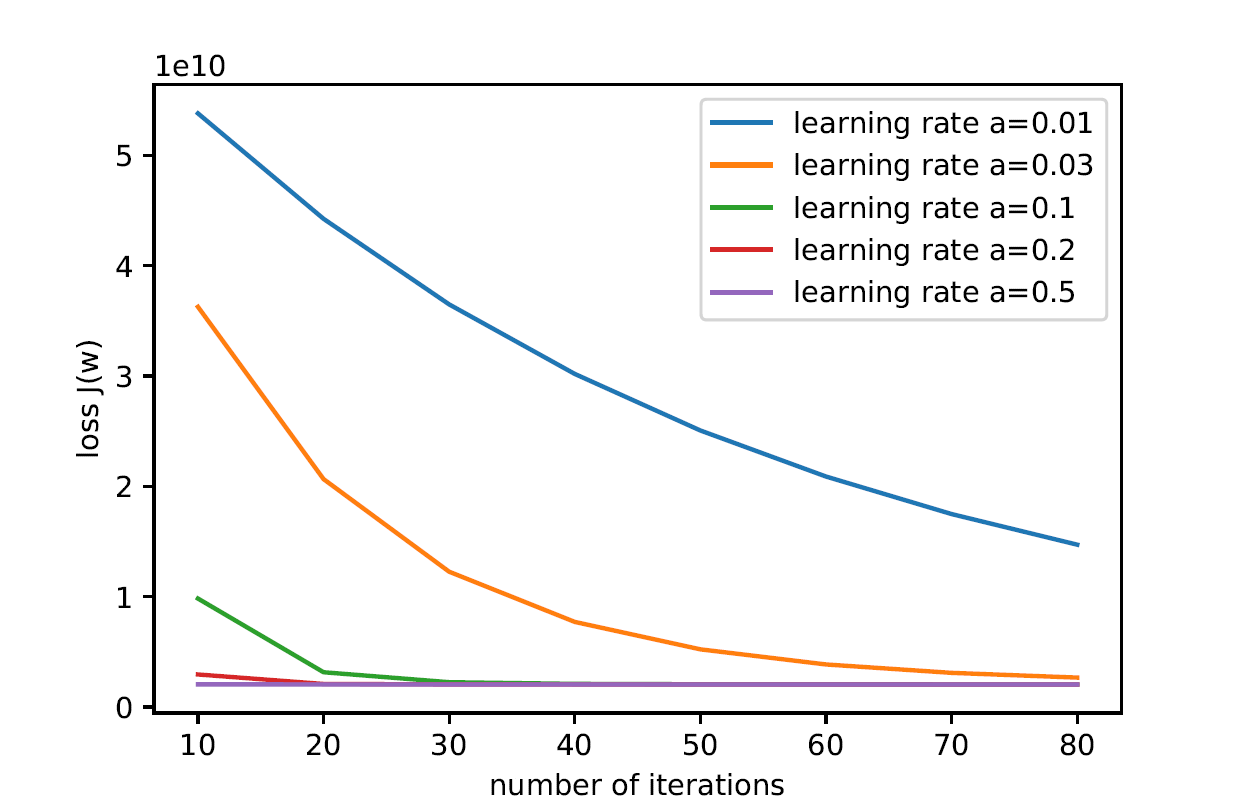
Handwritten in “writeup\_supplement\_xc1305.pdf”

1. Implement gradient descent to find the values w0, w1, and w2 that minimize J(w). Apply your code to the normalized data set using the learning rates ɑ = 0.01, 0.1, 0.2.

A good way to verify that gradient descent is working correctly is to look at the value of J(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(w) should never increase, and should converge to a steady value by the end of the algorithm. Plot J(w) for 10, 20,30,40,50,60,70,80 iterations for each of your values.

1. Do the same for learning rates ɑ = 0.03 and 0.5. Comment on which of the five learning rates gives the best result.

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

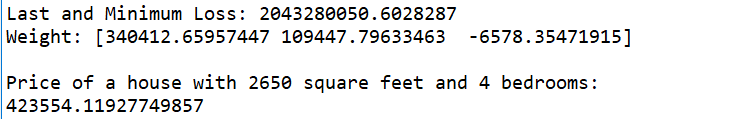


Written in “code\_forP2.B.(b,c)-gradient\_descent.py”

Learning rate ɑ = 0.5 gives the best result, since it descents the fastest and gives the smallest loss.

**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 2650 square feet and 4 bedrooms. Don’t forget to normalize your features when you make this prediction!



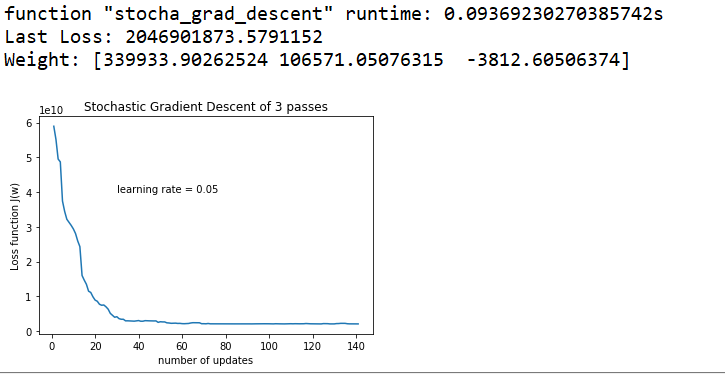
Written in “code\_forP3-predict.py”

**D. Stochastic Gradient**

Using a learning rate of ɑ = 0.05 to find the values w0, w1, and w2 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.

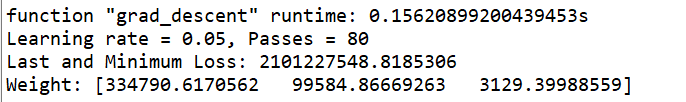
Comment how the J(w) you obtain with SGD with 3 passes compares with the J(w) you obtain gradient descent with 80 passes. Also comment on the computation time of the two approaches.

**SGD** (Output is random)**:**



Written in “code\_forP4-stochastic\_gradient\_descent.py”

**GD:**



Written in “code\_forP4-compare\_gradient\_descent.py”

The J(w) I obtain with SGD with 3 passes is 2046901873.5791152. The J(w) I obtain gradient descent with 80 passes is 2101227548.8185306. Stochastic gradient descent gets nearly the same minimum loss as gradient descent, and its loss function takes to converge in the last several updates.

The computation time of SGD with 3 passes (about 0.09s) is less than that of gradient descent with 80 passes (about 0.15s). Because by computing complexity, SDG with 3 passes have 3\*O(mn) computation; GD with 80 passes have 80\*O(mn) computation.

**Problem 3**

Handwritten in “writeup\_supplement\_xc1305.pdf”