Machine Learning PS2: Regression

Template Write-Up

NOTES:

- Due to the nature of Q1 & Q2 (it is difficult to write mathematical expressions in MS Word), it is fine to submit a physical (handwritten) copy of your answers, if that is what you prefer. However, it must be scanned and attached along with your code on NYU Classes, and the answers to the appropriate questions must be clearly labeled. Write "see attached document [document name]" for every question you do this for in this template document.

- Note that Q1, part 1 has a typo:
  - ~~$J(\mathbf{w}) = J(w_1, w_2, w_3)$~~ → $J(\mathbf{w}) = J(w_0, w_1, w_2)$

- Note that for Q2B part (c), $J(\mathbf{w})$ decreases with each pass of gradient descent, but will never actually converge to zero. Find a workaround – either implement a pass limit, or stop doing passes if the loss function $J(\mathbf{w})$ isn't changing much from pass to pass (define a threshold).

PROBLEM 1 (20 points)

Q1-1. What is the cost function $J(\mathbf{w}) = J(w_0, w_1, w_2)$ for the 4-item dataset in Problem 1?

$$J(\vec{w}) = \frac{1}{2}w_0^2 + 4831264w_1^2 + 9.5w_2^2 + 2104w_0w_1 + 3w_0w_2 + 6708w_1w_2 - 367.75w_0 - 840528w_1 - 1180.3w_2 - 147621.25$$

Q1-2. What are the values of (w_0, w_1, w_2) after one iteration of gradient descent? Comment on whether you normalized the data or not for this pass by hand.

$$\vec{w} = (w_1, w_2, w_3) = (36.775, 84052.8, 118.03)$$

PROBLEM 2 (60 points)

* Make sure that all your relevant code / implementation is attached. You must implement the relevant functions (mean, standard deviation, partial derivatives, the loss function, vanilla and stochastic gradient descent, etc.) yourself.

Q2A-a. Given a set of numbers $x_1,\ldots,x_m$, write down the equations for the mean and the standard deviation of these numbers. USE POPULATION STDEV (***not*** simple stdev)

The mean $\mu = \sum_{i=1}^{m} x_i$

The standard deviation $\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2}$

Q2B-a. 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. Is it different from your answer to Q1-1?

$$\mathrm{J}(\vec{w}) = \frac{1}{2\mathrm{m}}\sum_{i=1}^{m}\left(f_w\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$= \frac{1}{2\mathrm{m}}\sum_{i=1}^{m}\left(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} - y^{(i)}\right)^2$$

They appear different but the answer to Q1-1 is simply an expansion of the formula above using the provided data.

Q2B-b. Derive the partial derivatives of J(**w**) with respect to w_0, w_1, and w_2.

$$\frac{\partial \mathrm{J}(\vec{w})}{\partial w_0} = \frac{1}{2\mathrm{m}}\sum_{i=1}^{m}\left(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} - y^{(i)}\right)$$

$$= \mathrm{w}_0 + \mathrm{w}_1\bar{x}_1 + \mathrm{w}_2\bar{x}_2 - \bar{y}$$

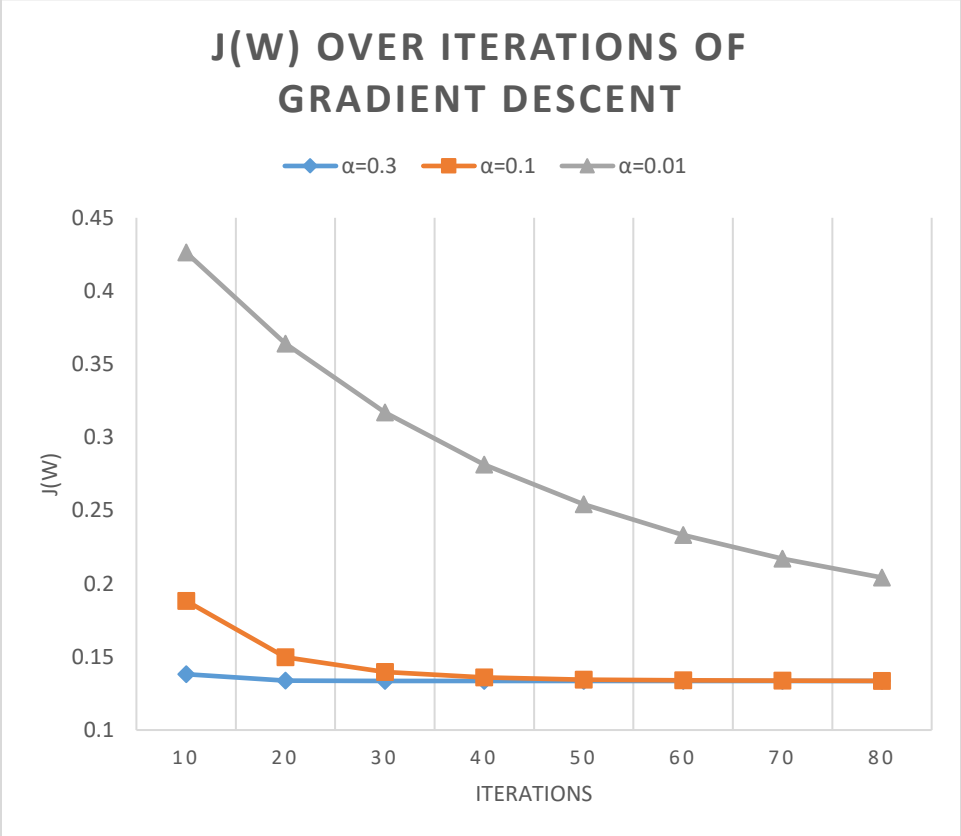
$$\frac{\partial \mathrm{J}(\vec{w})}{\partial w_1} = \frac{1}{2\mathrm{m}}\sum_{i=1}^{m}\left(w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} - y^{(i)}\right) x_1^{(i)}$$

$$= w_0\bar{x}_1 + \frac{w_1}{m}\sum\left(x_1^{(i)}\right)^2 + \frac{w_2}{m}\sum\left(x_1^{(i)}x_2^{(i)}\right) - \frac{1}{m}\sum\left(x_1^{(i)}y^{(i)}\right)$$

$$\frac{\partial J(\vec{w})}{\partial w_2} = \frac{1}{2m}\sum_{i=1}^{m}\left(w_0 + w_1x_1^{(i)} + w_2x_2^{(i)} - y^{(i)}\right)x_2^{(i)}$$

$$= w_0\bar{x}_2 + \frac{w_1}{m}\sum\left(x_1^{(i)}x_2^{(i)}\right) + \frac{w_2}{m}\sum\left(x_2^{(i)}\right)^2 - \frac{1}{m}\sum\left(x_2^{(i)}y^{(i)}\right)$$

Q2B-c. For learning rates α = 0.01, 0.1 and 0.3. Plot the J(**w**) values for [10,80] for each of the three learning rates. Comment on which gives the best result.



Learning rate of 0.01 does not allow convergence for 80 iterations. For 0.1, it reaches convergence by the 40th iteration. In the plot for 0.3 we can see that it converges at the 20th iteration.

Q2B-d. For learning rates α = 0.05, 0.5 and 1.5. Comment on which of the three learning rates gives the best result.

0.05 did not converge soon enough. 0.5 results in a quick convergence at around the

20th iteration, but 1.5 results in an even quicker convergence by the 10^{th} iteration. So $\alpha = 1.5$ is the best. It seems that if α becomes higher than 1.7~1.9, it'll take longer to converge or not converge at all.

Q2C. Use the w you obtain in Q2B from $\alpha = 0.3$ after 80 passes to predict housing prices. Predict the price of a house with 1650 sq. ft. area and 3 bedrooms. Don't forget to normalize the features when you make this prediction!

$293081.51138

Q2D. Use learning rate $\alpha = 0.1$ and compare the result of vanilla gradient descent using 80 passes over the data – that is, the w0, w1, and w2 values – to the result of stochastic gradient descent with $\alpha = 0.1$ and 3 full passes over the data. Provide the value of J(w) after each pass of SGD. Does the computation time significantly differ between the two methods?

Vanilla: [2.1023537520799205e-13, 0.87354202898599675, -0.042213526742056431]
Stochastic: [0.00039382465871208583, 0.2141775352629012, 0.096824333790158049]

J(w) values:
step 1 , 0.41986103549
step 2 , 0.359670747835
step 3 , 0.314208429249

Yes, it differs by more than 10 folds.

PROBLEM 3 (20 points)

Q3. Prove that the perceptron algorithm will have at most R^2/γ^2 mistakes. Justify all steps in your proof.

Please refer to attached file: Proof of bounded behavior.pdf