ECE M146 HW#2
Melody Chen
#705120273

1. In class, we see that the solution of the least square problem satisfies the normal equation $X^TXw = X^Ty$, where $X \in \mathbb{R}^{N\times M}$ and $y \in \mathbb{R}^N$. Prove the following statement:

   The Gram matrix X^TX is nonsingular if and only if X has linearly independent columns.

   With $N >> M$, there will likely be M linearly independent vectors x_n. Therefore, the Gram matrix X^TX is likely to be invertible.

   Hint: X has linearly independent columns if $Ax = 0$ only for $x = 0$. You may want to use the following properties of (non)singular matrix. The equation $Ax = 0$ has only the trivial solution $x = 0$ if and only if A is nonsingular. If A is singular, there exist $z \neq 0$ such that $Az = 0$.

Show $X^TXv = 0$ only when $v=0$ if X has linearly independent columns.
$Xv = 0$ only when $v=0$ since X is linearly independent.
We multiply v^T on both sides of $X^TXv = 0$.

$$v^TX^TXv = v^T \cdot 0$$
$$(Xv)^T(Xv) = \|Xv\|^2 = 0$$

only true if $v=0$, since X has lin. independent columns.

Show X has linearly independent columns when X^TX is nonsingular.
X^TX is nonsingular $\Rightarrow X^TXv = 0$ only when $v=0$.
Suppose X is linearly dependent.
$\Rightarrow Xz = 0$ when $z \neq 0$.
which means that $X^TXv = 0$ when $z \neq 0$.
This is a contradiction as X^TX is nonsingular, so $X^TXv=0$ only when $v=0$.
Thus, X has to be linearly independent.

2. Consider the hat matrix $H = X(X^TX)^{-1}X^T$, where $X \in \mathbb{R}^{N\times M}$ and X^TX is invertible.

   (a) Show that H is symmetric.

   (b) Show that $H^K = H$ for any positive integer K.

   (c) If I is the identity matrix of size N, show that $(I-H)^K = I-H$ for any positive integer K.

   (d) Show that $Trace(H) = M$, where the trace is the sum of diagonal elements. Hint:$Trace(AB) = Trace(BA)$.

a) Matrix H is symmetric when $H^T = H$.

$$H^T = (X(X^TX)^{-1}X^T)^T$$
$$= X((X^TX)^{-1})^TX^T$$
$$= X((X^TX)^T)^{-1}X^T$$
$$= X(X^TX)^{-1}X^T = H$$

$*\ (A^T)^{-1} = (A^{-1})^T$, $(AB)^T = B^TA^T$

b) $K=1$, $H^1 = H$.

$K=2$, $H^2 = H\cdot H = [X(X^TX)^{-1}X^T][X(X^TX)^{-1}X^T]$
$= X(X^TX)^{-1}X^TX(X^TX)^{-1}X^T$ $\quad * A^{-1}A = I$
$= X\cdot I\cdot(X^TX)^{-1}X^T = X(X^TX)^{-1}X^T = H$

$H^3 = H^2\cdot H = \underbrace{H\cdot H}_{\text{shown above}} = H$

For $K=k-1$, we have $H^{k-1} = H$, then $H^k = H^2 = H$.
By induction, $H^K = H$ for any positive integer K.

c) $K=1$, $(I-H)^1 = I-H$
$K=2$, $(I-H)^2 = I^2 - IH - HI + H^2$
$= I^2 - 2H + H^2 \leftarrow H^2 = H$
$= I^2 - H = I - H$
For $K=k-1$, we have $(I-H)^{k-1} = (I-H)$.
Then, $(I-H)^k = (I-H)(I-H) = I-H$.
By induction, $(I-H)^K = (I-H)$ for any positive integer K.

d) $Trace(X(X^TX)^{-1}X^T)$
$= Trace[(X^T)(X(X^TX)^{-1})]$
$= Trace[(X^TX)(X^TX)^{-1}]$
$= Trace(I_M)$ $\quad *(X^TX)(X^TX)^{-1}$ has $D = M\times M$
$= M$

3. Consider a linear regression problem in which we want to "weigh" different training instances differently because some of the instances are more important than others. Specifically, suppose we want to minimize

$$J(w_0, w_1) = \sum_{n=1}^{N} \alpha_n (w_0 + w_1 x_{n,1} - y_n)^2. \quad (1)$$

Here $\alpha_n > 0$. In class we worked out what happens for the case where all weights (the $\alpha_n's$) are the same. In this problem, we will generalize some of those ideas to the weighted setting. Calculate the gradient by computing the partial derivative of J with respect to each of the parameters (w_0, w_1). Comment on how the $\alpha_n's$ affect the linear regression problem. For example, what happens if $\alpha_i = 0$ for some i? Qualitatively describe what will happen in gradient descent if α_j is much greater than other $\alpha_{j'}, j' \neq j$.

Derivative of J w.r.t w_0

$$\frac{\partial(J(w_0,w_1))}{\partial w_0} = \sum_{n=1}^{N} 2\alpha_n (w_0 + w_1 x_{n,1} - y_n)$$

Derivative of J w.r.t w_1

$$\frac{\partial(J(w_0,w_1))}{\partial w_1} = \sum_{n=1}^{N} 2\alpha_n (w_0 + w_1 x_{n,1} - y_n) \cdot x_{n,1}$$

If $\alpha_i = 0$, the data point at n is irrelevant to the direction of the gradient descent, will not be affected by value of data point in any way.

If α_j is much greater than $\alpha_{j'}$ and $j \neq j'$, this means that difference between $w_0 + w_1 x_{j,1}$ and y_i will be enlarged and algorithm will take a large step. If α_j is much greater than the rest of α_n, the algorithm will likely converge to a line passing through $x_{j,1}$ and y_j.

4. In this exercise, you will develop a stochastic gradient descent (SGD) view of the perceptron algorithm.

(a) Find the gradient $\frac{\partial J(w)}{dw}$ for the following loss function:

$$J(w) = -\sum_{i \in \mathcal{M}} w^T x_i y_i,$$

where $w \in \mathbb{R}^N, x_i \in \mathbb{R}^N$, $i = 1, \cdots, M$ and $y_i \in \{-1, 1\}$. The set $\mathcal{M}$ denote the index set of misclassified data points, i.e., $\mathcal{M} = \{i | \text{sign}(w^T x_i) \neq y_i\}$. You may assume $w^T x_i \neq 0, \forall i$ in this question.

(b) Find the gradient $\frac{\partial J(w)}{dw}$ for the following loss function:

$$J(w) = \sum_{i=1}^{M} \max[0, -w^T x_i y_i],$$

where $w \in \mathbb{R}^N, x_i \in \mathbb{R}^N$, and $y_i \in \{-1, 1\}$. You may assume $w^T x_i \neq 0, \forall i$ in this question.

(c) Compare your answers from (a) and (b), are they equivalent? Describe the update rule if you use the SGD algorithm to minimize these loss functions. With the learning rate $\eta = 1$, are they equivalent to the perceptron algorithm?

$w^T x_i y_i$ always negative.

a) $$\frac{\partial J(w)}{\partial w} = -\sum_{i \in M} x_i y_i$$

b) $$\frac{\partial J(w)}{\partial w} = \sum_{i=1}^{M} \max[0, -x_i y_i]$$

$-x_i y_i$ always positive.

$$= -\sum_{i=1}^{M} (x_i y_i)$$

c) Yes they're equivalent. The update rule with SGD algorithm:

$$w_{t+1} = w_t - \eta(-x_i y_i)$$

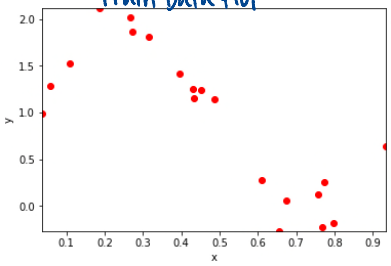
Yes with $\eta = 1$, it is equivalent to perceptron algorithm.

$$w_{t+1} = w_t + x_i y_i$$

5. In this exercise, you will work through linear and polynomial regression. Our data consists of inputs $x_n \in \mathbb{R}$ and outputs $y_n \in \mathbb{R}, n \in \{1, \cdots, N\}$, which are related through a target function $f(x)$. Your goal is to learn a predictor $h_w(x)$ that best approximates $f(x)$.

   (a) **Visualization** We provide you two sets of data, the training data and the testing data in the two files, *regression_train.csv* and *regression_test.csv*. In each file, the first column is the input and the second column is the output. In MATLAB, visualize the training data by plotting the input v.s. the output. What do you observe? For example, can you make an educated guess on the effectiveness of linear regression in predicting the data?

Train Data Plot



There is a linear trend for the data points, so we can use a linear regression.

Based on the way the data points are scattered, a linear regression is likely not the most effective regression method. Maybe higher order polynomial will fit data points better.

   (b) **Linear Regression: closed-form solution** Let us start by considering a simple linear regression model:

$$h_w(x) = w^T x = w_0 + w_1 x.$$

Recall that linear regression attempts to minimize the objective function

$$J(w) = \sum_{n=1}^{N} (h_w(x_n) - y_n)^2 = \|Xw - y\|^2,$$

where

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, X = \begin{bmatrix} 1, x_1 \\ 1, x_2 \\ \vdots, \vdots \\ 1, x_n \end{bmatrix}, w = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}.$$

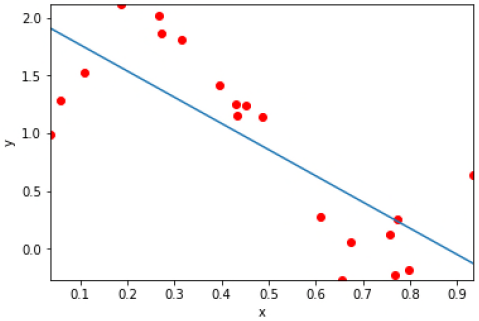
Note that to take into account the intercept term w_0, we can add an additional "feature" to each instance and set it to one, e.g., $x_{i,0} = 1$. This is equivalent to adding an additional first column to X and setting it to all ones.

In class we learned that the closed-form solution to linear regression is:

$$w^* = (X^T X)^{-1} X^T y.$$

Implement this closed-form solution in MATLAB using the training data and report w and $J(w)$. Also generate a plot depicting your training data and the fitted line.

```
w vector:
[[ 1.99196163]
 [-2.27048372]]
J(w): 4.603634906406957
```



(c) **Linear Regression: gradient descent** Another way to solve linear regression is through gradient descent (GD). In gradient descent, each iteration performs the following update:

$$w_j := w_j - \eta \sum_{n=1}^{N} (h_w(x_n) - y_n) x_{n,j} \quad \text{(simultaneously update } w_j \text{ for all } j\text{)}.$$

With each iteration of gradient descent, we expect our updated parameters w to come closer to the optimal w^* and therefore achieve a lower value of $J(w)$.

Implement the gradient descent algorithm in MATLAB using all of the following specifications for the gradient descent algorithm:

- Initialize w to be the all 0s vector.
- Run the algorithm for 10000 iterations.
- Terminate the algorithm earlier if the value of $J(w)$ is unchanged across consecutive iterations. In this exercise, we say $J(w)$ is unchanged if $|J(w)_{t-1} - J(w)_t| < 0.0001$.
- Use a fixed learning rate η.

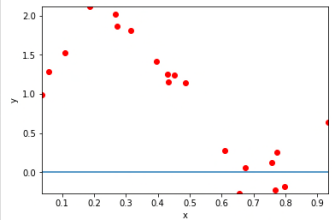
For different $\eta = 0.05, 0.001, 0.0001, 0.00001$, report the final $J(w)$ and the number of iterations until convergence (the number will be 10000 if the algorithm did not converge within 10000 iterations). Discuss how does the learning rate η affect the GD algorithm.

```
Eta: 0.05
w vector:
[[ 1.98016789]
 [-2.24681457]]
J(w): 4.6044056136703855
Iterations till Convergence: 83

Eta: 0.001
w vector:
[[ 1.90162541]
 [-2.08918575]]
J(w): 4.648852786068523
Iterations till Convergence: 2420

Eta: 0.0001
w vector:
[[ 1.55932339]
 [-1.40221141]]
J(w): 5.640772178144737
Iterations till Convergence: 10000

Eta: 1e-05
w vector:
[[0.76703186]
 [0.04168683]]
J(w): 12.249367268440261
Iterations till Convergence: 10000
```

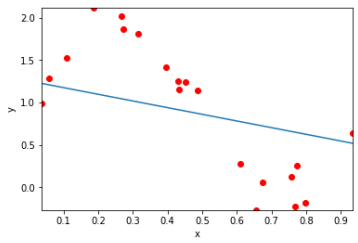
If the learning rate, η, is very small, the gradient descent algorithm will not converge within 10000 iterations. This is because the steps taken are very small, so the learning will be very slow.

(d) **Gradient Descent Visualization** Repeat the exercise in (c) with only $\eta = 0.05$ for only 40 iterations, with w initialized to the all 0s vector. Report w, $J(w)$ and plot the fitted line (along with the data) for iteration 0, 10, 20, 30 and 40. Compare your fitted lines and $J(w)$(s) to what you get in (b). What do you observe over different iterations?

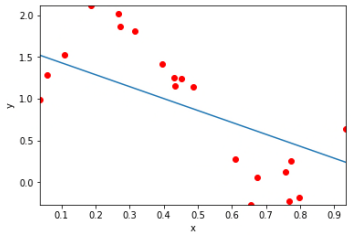
```
Iteration: 0
w vector:
[[0.]
 [0.]]
J(w): 28.655524112125004
```



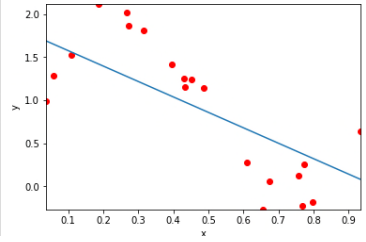
```
Iteration: 10
w vector:
[[ 1.25300519]
 [-0.78745477]]
J(w): 7.629319845754809
```



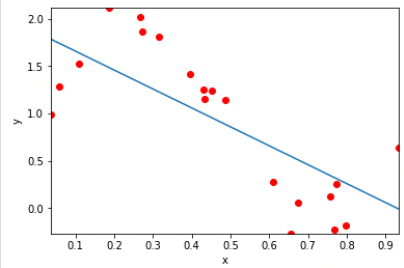
```
Iteration: 20
w vector:
[[ 1.57272545]
 [-1.42910834]]
J(w): 5.5775115336820615
```



```
Iteration: 30
w vector:
[[ 1.75411397]
 [-1.79314142]]
J(w): 4.917096388407238
```



```
Iteration: 40
w vector:
[[ 1.85702216]
 [-1.99967039]]
J(w): 4.7045286931001815
```



Over increasing iterations, the weights get closer to the weights found in part B. J(w) decreases as the iterations of gradient descent increases. Thus, with more iterations, the fitted line looks more like fitted line in part B.

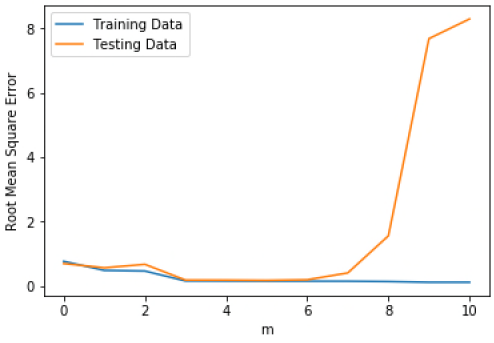
(e) **Polynomial Regression** Next let us consider the more complicated case of polynomial regression, where our hypothesis is

$$h_w(x) = w^T\phi(x) = w_0 + w_1x + w_2x^2 + \cdots + w_mx^m.$$

Note that the function is linear in the coefficient w. We can therefore extend the result of linear regression by replacing the input matrix X with

$$\Phi = \begin{bmatrix} \phi(x_1)^T \\ \phi(x_2)^T \\ \vdots \\ \phi(x_N)^T \end{bmatrix}$$

where $\phi(x)$ is a function such that $\phi_j(x) = x^j$ for $j = 0, \cdots, m$.
For $m = 0, \cdots, 10$, use the closed-form solution to determine the best-fit polynomial regression model on the training data. With this model, calculate the RMSE (Root-Mean-Square error),i.e., $E_{RMS} = \sqrt{J(w)/N}$, on both the training data and the test data. Generate a plot depicting how RMSE (both training and testing) varies with model complexity (polynomial degree m). Which degree of polynomial would you say best fits the data? Was there evidence of under/over-fitting the data? Use your plot to justify you answer.



Polynomial of order 3 to 6 appears to best fit the data as both the RSME for training and testing data is low.
A small amount of underfitting can be observed from both the training and testing data for order 0-2 as RSME for both is higher than other parts.
A large amount of overfitting, can be observe for order 7-10 as RSME for testing data is significantly high.