

Machine Learning: Homework #1

Due on March 15, 2018 at 12:00 pm

*Professor ***

Name: Zhang ye di

ID: **

Problem 1

Part One

- Example 1: **Recommendation system**, like "NetEase Cloud Music" app. Take Content Based Recommendation(CBR) for an example. Such CBR provides personalized recommendation by matching users interests with description and attributes of items. For CBR, we can use standard Machine Learning techniques like Logistic Regression, SVM, Decision tree etc. based on user and item features for making predictions for eg: extent of like or dislike. Then, we can easily convert the result to ranked recommendation.
- Example 2: **Disease diagnosis**. There is a strong need for diagnostic imaging to accurately treat patients. Some works have successfully embedded mobile devices with some specific systems for the purpose of identifying specific diseases. Such specific systems are coupled with machine learning techniques, which are an assortment of mathematical algorithms capable of identifying patterns in data and performing predictions on new information. For example, researchers investigating the detection of retinal diseases combined a smart phone and microscopic lens to perform eye examinations and disease diagnosis. By capturing retinal images and analyzing them, normal or infected conditions could be identified. Similar research involving the detection of ultrasound kidney abnormalities also made use of a smart phone camera and machine learning algorithms for diagnostic analysis. In both cases, the disease detection rates were competitive with accuracies above 80%.

Part Two

Let $g : R \rightarrow R$ be defined as $g(\alpha) = f(\mathbf{x} + \alpha \mathbf{d})$, ($\alpha > 0$), then $g'(\alpha) = \mathbf{d}^T$. With Taylor expansion, we can get:

$$g(\alpha) = g(0) + g'(0)\alpha + o(\alpha)$$

It equals to:

$$f(\mathbf{x} + \alpha \mathbf{d}) = f(\mathbf{x}) + \alpha \cdot \nabla^T f(\mathbf{x}) \cdot \mathbf{d} + o(\alpha)$$

equals to :

$$\frac{f(\mathbf{x} + \alpha \mathbf{d}) - f(\mathbf{x})}{\alpha} = \nabla^T f(\mathbf{x}) \cdot \mathbf{d} + \frac{o(\alpha)}{\alpha}$$

Since $\exists \bar{\alpha}$ such that for $\forall \alpha \in (0, \bar{\alpha})$, we have $\frac{|o(\alpha)|}{\alpha} < \frac{1}{2} |\nabla^T f(\mathbf{x}) \cdot \mathbf{d}|$. Since we have $\nabla f(\mathbf{x})^T \cdot \mathbf{d} < 0$, we conclude that for $\forall \alpha \in (0, \bar{\alpha})$:

$$f(x + d) - f(x) < \frac{1}{2} \nabla^T f(x)^T \cdot \mathbf{d} \cdot \alpha < 0$$

So we can say, direction \mathbf{d} is a descent direction.

Problem 2

Part One

This question can be reduced to such a problem:

If f has positive semidefinite Hessian, then for all \mathbf{x}, \mathbf{y} in the domain, and $t \in [0, 1]$, we have:

$$f(t\mathbf{x} + (1-t)\mathbf{y}) \leq tf(\mathbf{x}) + (1-t)f(\mathbf{y})$$

To reduce it to the one-dimensional case, fix \mathbf{x} and \mathbf{y} and look at the function restricted to the line segment connecting those points. That is, define the one-dimensional function:

$$g(t) = f(t\mathbf{x} + (1-t)\mathbf{y})$$

Then we can compute the derivatives of g :

$$g'(t) = (\mathbf{x} - \mathbf{y})^T \nabla f(t\mathbf{x} + (1-t)\mathbf{y})$$

$$g''(t) = (\mathbf{x} - \mathbf{y})^T \nabla^2 f(t\mathbf{x} + (1-t)\mathbf{y})(\mathbf{x} - \mathbf{y})$$

Since the Hessian is positive semidefinite, we have $g''(t) \geq 0$ for all t . Then we use this with Taylor's theorem to prove that:

$$g(0) \geq g(t) + g'(t)(-t)$$

$$g(1) \geq g(t) + g'(t)(1-t)$$

Then if $t \in [0, 1]$, these can be combined to give:

$$g(t) \leq tg(1) + (1-t)g(0)$$

which is equivalent to the inequality we want to prove.

Part Two

If $\nabla f(x^*) \neq 0$, then $d = -\nabla f(x^*)$ is a descent direction, whereby x^* can not be a global minimizer for this convex function.

Problem 3

Part One

Linear Function of x :

$$h_\theta(x) = \theta_0 + \theta_1 x, \theta = [\theta_0, \theta_1]^T$$

And the error/cost function J on the training set is:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^i) - y^i)^2$$

The termination criterion is : converged($J - e \leq \text{accuracy}$) or iteration times gets to an upper bound $iter$). J is the error in the new iteration, and e is the error in the last(previous) iteration.

Part Two

The result is as follows:

$$\theta_0 = 5.7667322265 \quad \theta_1 = -0.300668940342$$

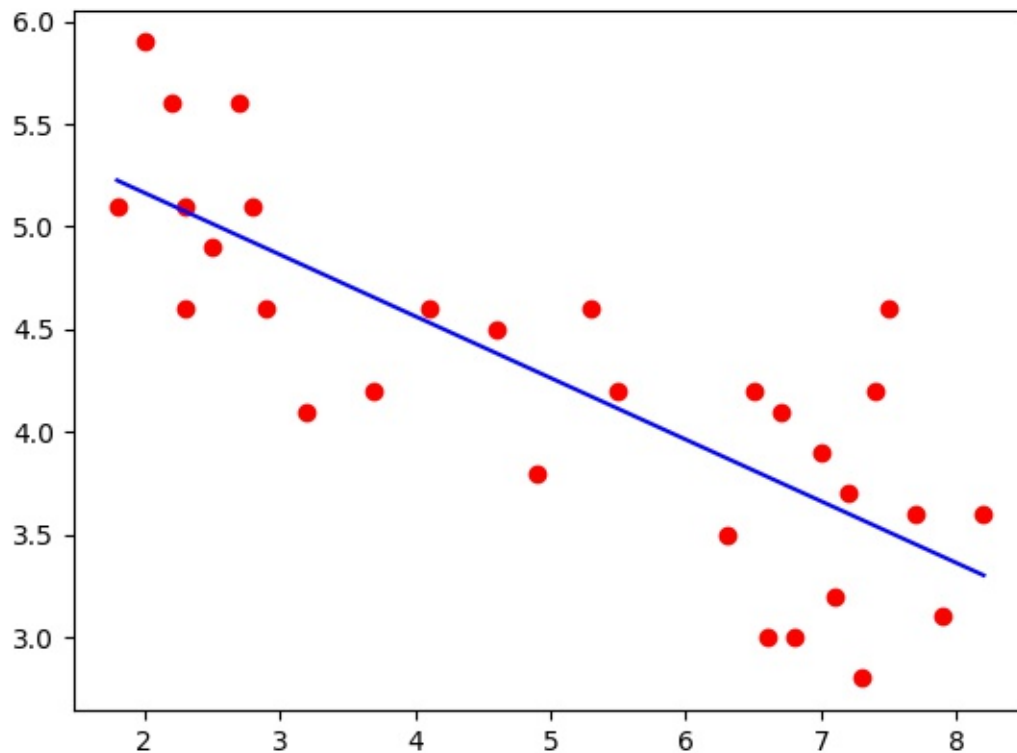


Figure 1: plot graph

```
Converged, iterations: 7506 !!!  
theta0 is: 5.7667322265 and theta1 is: -0.300668940342 and the error is: 6.93340244834  
Have done!!!
```

Figure 2: result

Problem 4

Solution:

Here we introduce two methods to implement robust regression:

1. RANSAC: Random sample consensus
2. Huber Loss

RANSAC

In the first method we use "Random sample consensus(RANSAC)" method to improve our algorithm by observing and deleting outliers from original training set, i.e. for each iteration, we find out a better inliers data to get a better simulation result. It is also can be interpreted as an outlier detection method. So the loss function doesn't change, is also

square loss function. However, since we omit outliers, which means **the new loss function will not consider these outliers**. So the new error function will be like:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^i) - y^i)^2, \text{ and } (x^i, y^i) \notin \text{outliers}$$

The improved robust result and plot graph is as follows (red for **square**, blue for **RANSAC**):

$$\theta'_0 = 5.7384847 \quad \theta'_1 = -0.292283$$

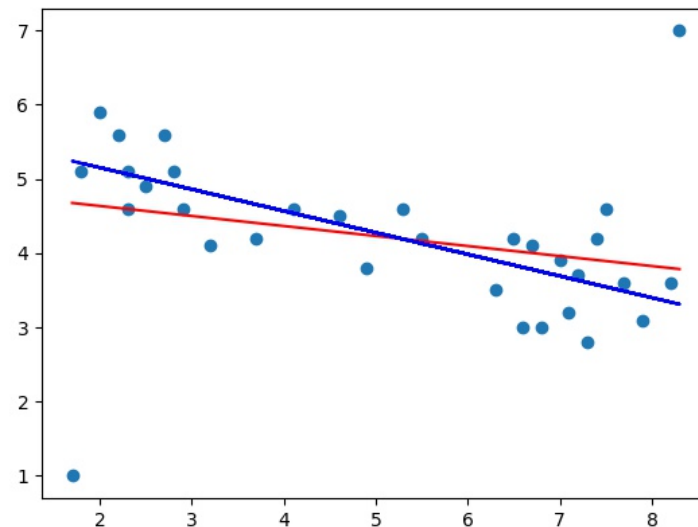


Figure 3: Ransac implement result 1

When applying to the test-data, result is as follows:

$$\text{square-error} = 1.75992 \quad \text{ransac-error} = 0.41787$$

```

gradient_descent 3  gradient_descent ransac
D:\anaconda\python.exe "G:/dd-resource/slides/homework/ML_project/gradient_descent ransac.py"
this theta0 is: 4.90627078185 and this theta1 is: -0.134975820788 and this error is: 34.5656467306
better theta0 is: 5.73848472559 and better theta1 is: -0.292283463248 and better error is: 2.07683053037
linear square error is: 1.75992349089
robust error is: 0.417876218584
Have done!!!
Process finished with exit code 0

```

Figure 4: Ransac implement result 2

Huber

In the second method we use "Huber loss", a loss function used in robust regression, that is less sensitive to outliers in data than the squared error loss. The Huber loss function is defined as follows:

$$L_{\delta}(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leq \delta, \\ \delta |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

Figure 5: huber loss function

$$\begin{aligned} \theta_0: \\ \frac{\partial J}{\partial \theta_0} &: \frac{1}{m} \cdot \sum_i \begin{cases} \frac{1}{2}(\theta_0 + \theta_1 x^i - y^i) & , \quad |y^i - (\theta_0 + \theta_1 x^i)| \leq \delta \\ \frac{1}{2} \cdot \delta & , \quad \theta_0 + \theta_1 x^i - y^i > \delta \\ -\frac{1}{2} \cdot \delta & , \quad \theta_0 + \theta_1 x^i - y^i < -\delta \end{cases} \\ \\ \theta_1: \\ \frac{\partial J}{\partial \theta_1} &: \frac{1}{m} \cdot \sum_i \begin{cases} \frac{1}{2}(\theta_0 + \theta_1 x^i - y^i) \cdot x^i & , \quad |y^i - (\theta_0 + \theta_1 x^i)| \leq \delta \\ \frac{\delta}{2} \cdot x^i & , \quad \theta_0 + \theta_1 x^i - y^i > \delta \\ -\frac{\delta}{2} \cdot x^i & , \quad \theta_0 + \theta_1 x^i - y^i < -\delta \end{cases} \end{aligned}$$

Figure 6: huber loss function

First, we test a series of δ value, and find out the best case is $\delta = 0.5$. (shown in Figure 7)

The improved robust result and plot graph (Figure 8) is as follows (red for **square**, blue for **Huber Loss**):

$$\theta'_0 = 5.297371 \quad \theta'_1 = -0.216245$$

When applying to the test-data, result (Figure 9) is as follows:

$$\text{square-error} = 1.75992 \quad \text{huber-error} = 0.43864$$

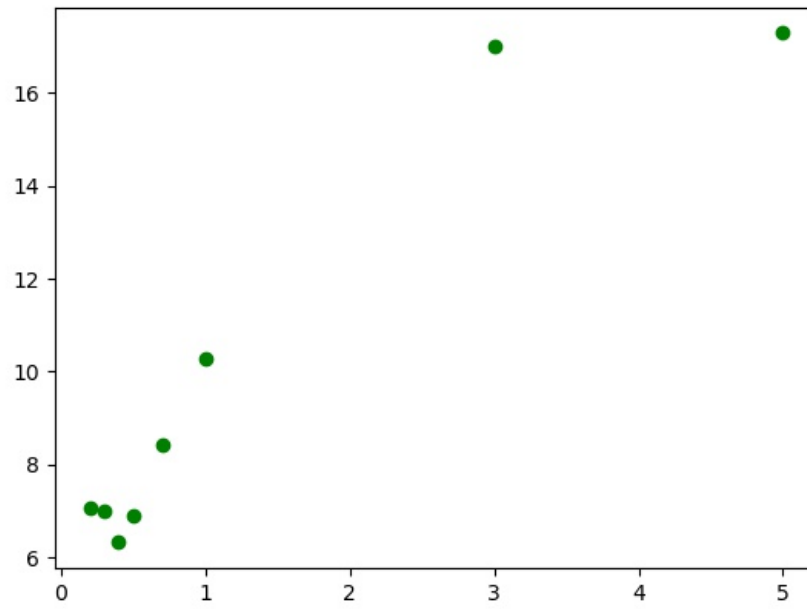


Figure 7: huber loss plot graph

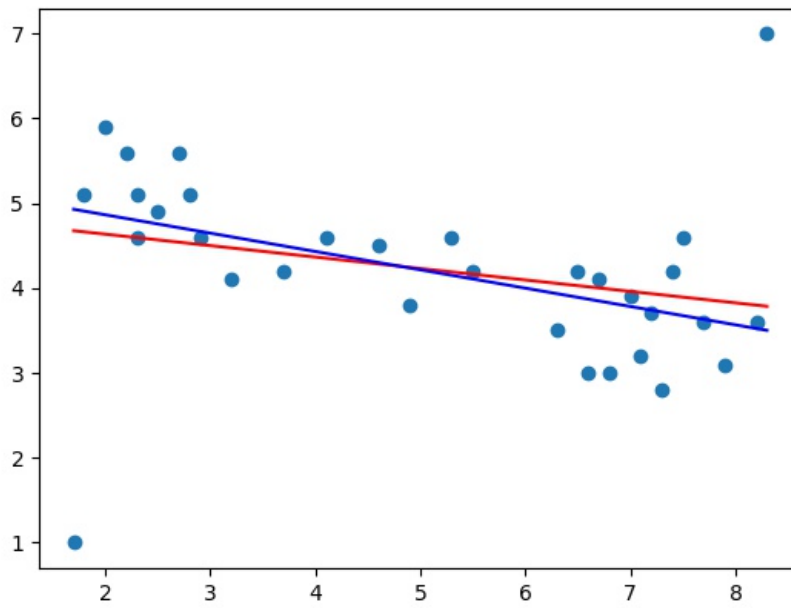
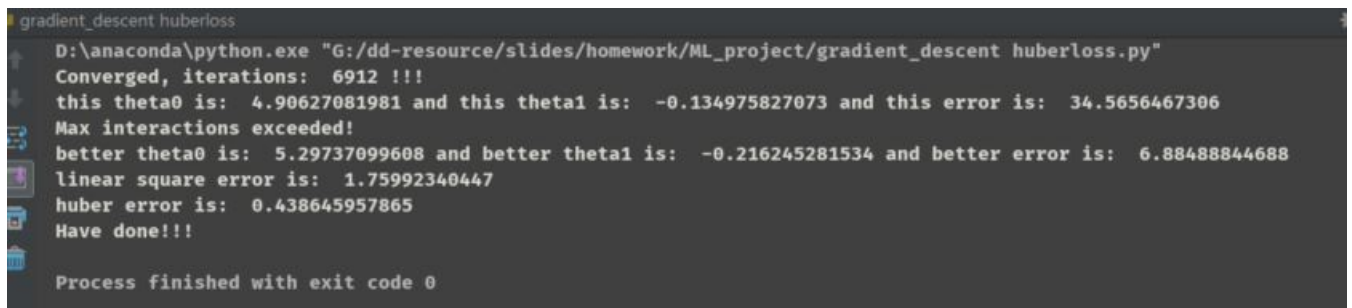


Figure 8: huber loss plot graph



```
gradient_descent huberloss
D:\anaconda\python.exe "G:/dd-resource/slides/homework/ML_project/gradient_descent huberloss.py"
Converged, iterations: 6912 !!!
this theta0 is: 4.90627081981 and this theta1 is: -0.134975827073 and this error is: 34.5656467306
Max interactions exceeded!
better theta0 is: 5.29737099608 and better theta1 is: -0.216245281534 and better error is: 6.88488844688
linear square error is: 1.75992340447
huber error is: 0.438645957865
Have done!!!
Process finished with exit code 0
```

Figure 9: huber loss implement result 2