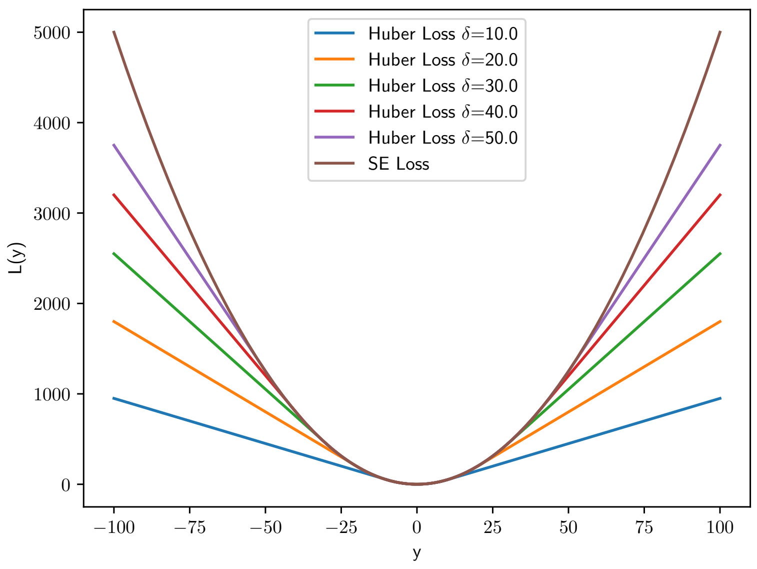
CSC411 Fall 2018: Homework 3

Niloufar Afsariardchi

March 9, 2020

1 Problem 1

a) Here is how these to loss functions look like:



Outliers usually have high errors. Huber loss is more robust to outliers because as can be seen above it penalizes high errors less than SE loss. For small errors below δ, the loss is the same, but above that it changes linearly instead of quadratic. By changing δ, we can control the behaviour of huber loss. As we increase δ, we increase the degree of penalization on high errors.

b) The derivative of the Huber function is

$$\frac{dH_\delta(a)}{da} = \begin{cases} a, & \text{if } |a| < \delta \\ \delta, & \text{if } a > \delta \\ -\delta, & \text{if } a < -\delta \end{cases} \tag{1}$$

Therefore,

$$\frac{\partial L_\delta(\mathbf{y}-\mathbf{t})}{\partial \mathbf{w}} = \frac{\partial L_\delta(\mathbf{X}\mathbf{w}+b-\mathbf{t})}{\partial \mathbf{w}} \tag{2}$$

$$= \frac{\partial L_\delta(a)}{\partial a}\frac{\partial a}{\partial \mathbf{w}} \tag{3}$$

$$= \mathbf{X}^T H'(\mathbf{y}-\mathbf{t}) \tag{4}$$

where $\mathbf{X}$ is the design matrix of size $N \times m$ for N number of training examples and m features. Similarly,

$$\frac{\partial L_\delta(\mathbf{y}-\mathbf{t})}{\partial b} = \frac{\partial L_\delta(\mathbf{X}\mathbf{w}+b-\mathbf{t})}{\partial b} \tag{5}$$

$$= \frac{\partial L_\delta(a)}{\partial a}\frac{\partial a}{\partial b} \tag{6}$$

$$= \mathbf{1}_{1\times N} H'(\mathbf{y}-\mathbf{t}) \tag{7}$$

c) Please find q1.py file.

2 Problem 2

a) The weighted loss function is

$$L = 0.5\sum_{i=1}^{N} a^{(i)}(y^{(i)} - \sum_{j=1}^{D} w_j x_j^{(i)})^2 + 0.5\lambda \sum_{j=1}^{D} w_j^2 \tag{8}$$

Therefore,

$$\frac{\partial L}{\partial w_j} = -\sum_{i=1}^{N} a^{(i)}(y^{(i)} - \sum_{j'=1}^{D} w_{j'} x_{j'}^{(i)})x_j^{(i)} + \lambda w_j \tag{9}$$

Equating $\frac{\partial L}{\partial w_j} = 0$ gives

$$\sum_{i=1}^{N} a^{(i)} y^{(i)} x_j^{(i)} = +\sum_{j'=1}^{D}\sum_{i=1}^{N} a^{(i)} w_{j'} x_{j'}^{(i)} x_j^{(i)} + \lambda w_j \tag{10}$$

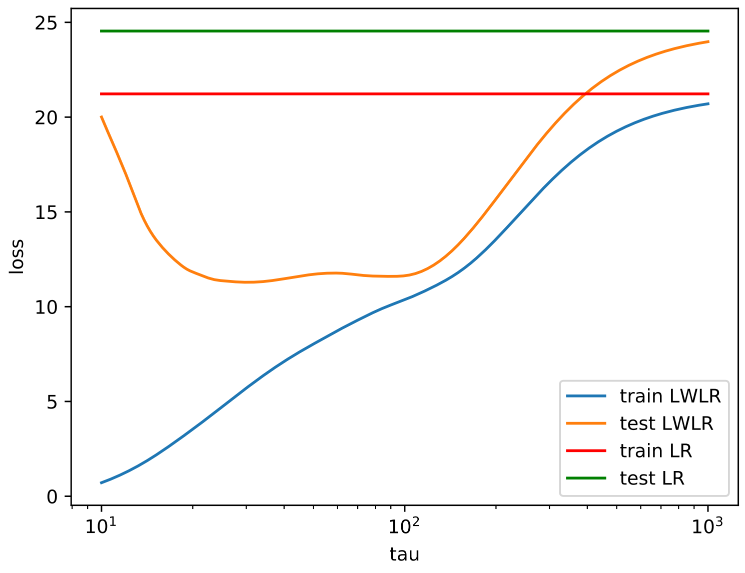
considering the sizes of each matrix, we then vectorize two sides of the equation,

$$\mathbf{X}^T\mathbf{A}\ \mathbf{y} = (\mathbf{X}^T\mathbf{A}\mathbf{X} + \lambda\mathbf{I})\mathbf{w} \tag{11}$$

$$(\mathbf{X}^T\mathbf{A}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{A}\ \mathbf{y} = \mathbf{w} \tag{12}$$

b) Please see q2.py file for the implementation of LWLR.

c) Below you see the squared errors of validation and training sets in two cases of locally weighted linear regression (LWLR) and normal linear regression (LR).



The latter I implemented for comparison purposes. As we can see, LWLR outperforms LR for lower τs. Also there is an optimum range for τ that gives low losses for the validation set.

d) As we increase τ, our bell-shaped weight distribution function around a datum x_0 becomes broader. Note that the weights are normalized so that their sum for different $x^{(i)}$s is equal to one. This means that as $\tau \to \infty$, the weights associated with $x^{(i)}$s become nearly equal as the bell-shaped function is at the broadest possible shape. With equal weights there are indeed no penalties on further training data points in the cost function, and thus the algorithm approaches a normal linear regression method. In the plot above, I showed the performance of a linear regression with weights equal to one which confirms the earlier statement. Then when $\tau \to 0$, the weight function looks like a delta dirac function which has a value only at the query location x_0. The weights everywhere else are zero. While this works good for training set because the query point has been already seen and the algorithm basically returns the target of an already-seen data point, it will not work well for the test set, because the training data points in general have some distance with the query point x_0 and since the weights approaches zero everywhere except at the location of x_0, the output of the algorithm would not use the training data at all (except in the case of an already-seen datum). This is why the validation loss is large for low values of τ in the plot above.