# CSC411 HW3 (corrected)

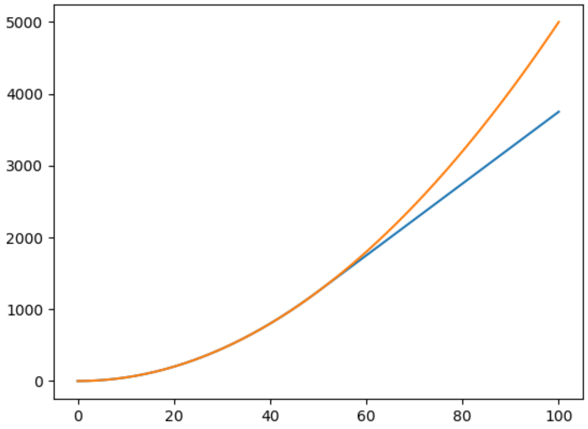
Casey Juanxi Li - 998816973

October 2018

## 1 Robust Regression.

a) Sketch the Huber Loss. Based on your sketch, why would you expect the Huber loss to be more robust to outliers?

Here's a sketch of Huber loss for an arbitrarily chosen $\delta = 50$, halfway between the plot's range for y (0 to 100):



Note that at the breakpoint of the piecewise function, $\delta = a$, the function values are equal, and

$$\frac{d(\frac{1}{2}a^2)}{da} = a, \quad \frac{d(\delta(a - \frac{1}{2}\delta))}{da} = \delta, \quad \delta = a$$

So the slopes are equal too. Basically from $\delta$ onwards, the loss function takes on and maintains the slope of the squared error loss at $\delta$.

Outliers - things that are far away from the target, as in past $\delta$ - will only increase the error in a linear fashion, as opposed to the polynomial increase of the SE loss.

b) Formulas for partial derivatives of Huber Loss. Correction made here. Be more careful with derivatives involving absolute values.

$$\frac{\partial L_\delta}{\partial w} = \begin{cases} \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w}, & \text{if } |y-t| \leq \delta \\ \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w}, & \text{if } |y-t| > \delta \end{cases}$$

$$\frac{\partial L_\delta}{\partial w} = \begin{cases} ((w^T x + b) - t) \cdot x, & \text{if } |y-t| \leq \delta \\ \delta \cdot x \cdot sign(w^T x + b - t), & \text{if } |y-t| > \delta \end{cases}$$

$$\frac{\partial L_\delta}{\partial b} = \begin{cases} \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial b}, & \text{if } |y-t| \leq \delta \\ \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial b}, & \text{if } |y-t| > \delta \end{cases}$$

$$\frac{\partial L_\delta}{\partial b} = \begin{cases} ((w^T x + b) - t) \cdot 1, & \text{if } |y-t| \leq \delta \\ \delta \cdot 1 \cdot sign(w^T x + b - t), & \text{if } |y-t| > \delta \end{cases}$$

c) Done in `q1.py` using a $m \times N$ random matrix as X.

# 2 Locally Weighted Regression.

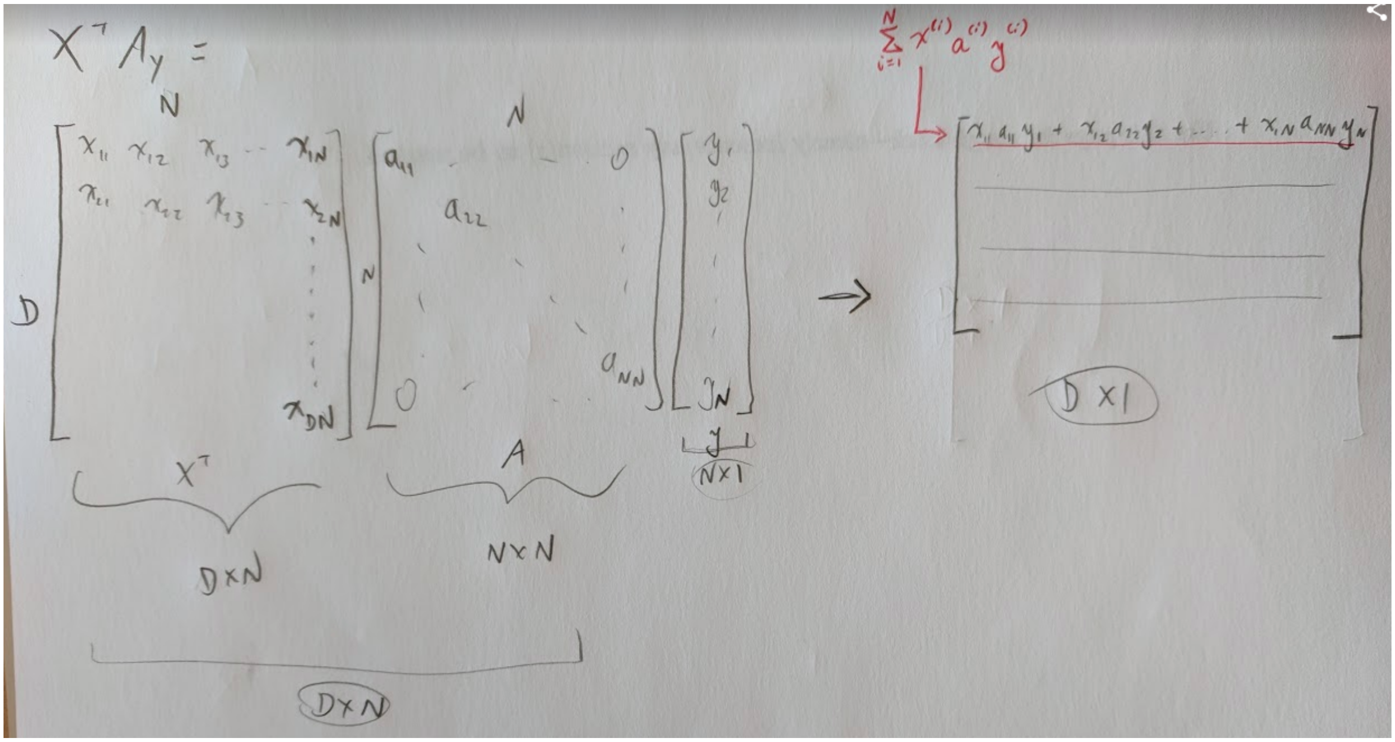
a) Direct solution for **w** that minimizes regularized weighted least squares loss:

$$\begin{aligned}\text{L} &= \frac{1}{2}\sum_{i=1}^{N} a^{(i)}\left(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \\ &= \frac{1}{2}\sum_{i=1}^{N} a^{(i)}\left(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}\right)^2 + \frac{\lambda}{2}(\mathbf{w}^T\mathbf{w})\end{aligned}$$

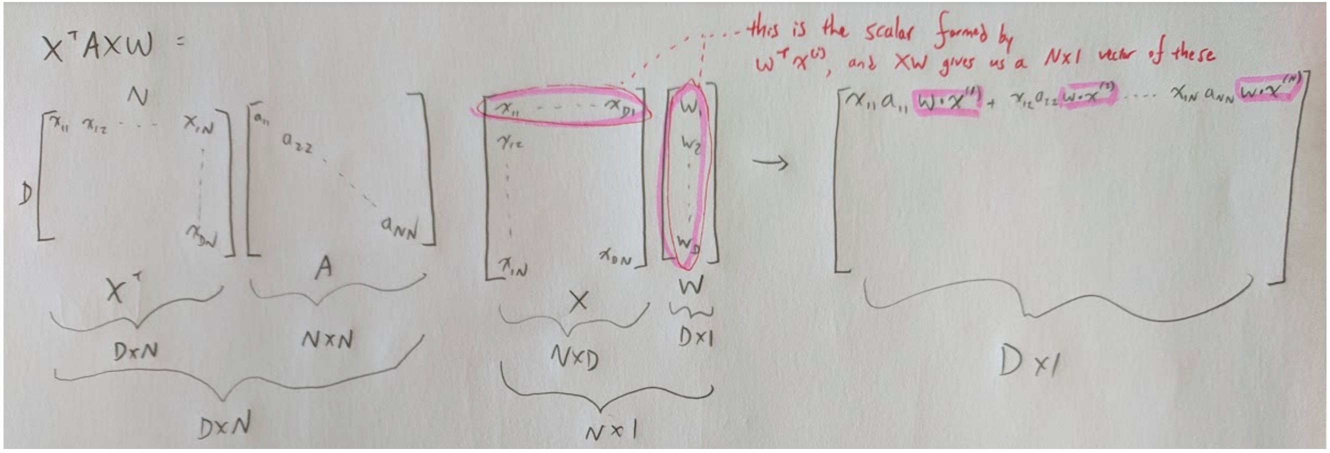
$$\begin{aligned}\frac{\partial L}{\partial w} &= \frac{1}{2}\sum_{i=1}^{N} a^{(i)}\cdot 2\cdot x^{(i)}\left(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)}\right) + \frac{\lambda}{2}\mathbf{w}\cdot 2 \\ &= \sum_{i=1}^{N} a^{(i)}x^{(i)}y^{(i)} - \sum_{i=1}^{N} a^{(i)}x^{(i)}\cdot\mathbf{w}^T\mathbf{x}^{(i)} + \lambda\mathbf{I}\mathbf{w}\end{aligned}$$

Note that $\sum_{i=1}^{N} a^{(i)}x^{(i)}y^{(i)}$ becomes $\mathbf{X}^T\mathbf{A}\mathbf{y}$ when $\mathbf{X}$ is a $N\times D$ matrix with each row being an $x^{(i)}$. Sorry, not good enough with LaTeX to typeset this:

$$\sum_{i=1}^{N} a^{(i)}x^{(i)}y^{(i)} = a^{(1)}x^{(1)}y^{(1)} + a^{(2)}x^{(2)}y^{(2)} + ... + a^{(N)}x^{(N)}y^{(N)}$$


$X^T A_y =$
N
$x_{11}$ $x_{12}$ $x_{13}$ $x_{1N}$
$x_{21}$ $x_{22}$ $x_{23}$ $x_{2N}$
D
$x_{DN}$
N
$a_{11}$ 0
$a_{22}$
N
0 $a_{NN}$
$y_1$ $y_2$ $y_N$
$X^T$
$D\times N$
A
$N\times N$
y
$N\times 1$
$D\times N$
$\sum_{i=1}^{N} x^{(i)} a^{(i)} y^{(i)}$
$x_{11}a_{11}y_1 + x_{12}a_{22}y_2 + \dots + x_{1N}a_{NN}y_N$
$D\times 1$

And note that $\sum_{i=1}^{N} a^{(i)}x^{(i)} \cdot \mathbf{w}^T\mathbf{x}^{(i)}$ becomes $\mathbf{X}^T\mathbf{A}\mathbf{X}\mathbf{w}$:



$X^T A X W =$
this is the scalar formed by $w^T x^{(i)}$, and $XW$ gives us a $N \times 1$ vector of these
N
D
$x_{11}$ $x_{12}$ $x_{1N}$ $x_{DN}$
$a_{11}$ $a_{22}$ $a_{NN}$
$x_{11}$ $x_{D1}$ $x_{12}$ $x_{1N}$ $x_{DN}$
$w_1$ $w_2$ $w_D$
$x_{11}a_{11}\,w\cdot x^{(1)} + x_{12}a_{22}\,w\cdot x^{(2)} \cdots x_{1N}a_{NN}\,w\cdot x^{(N)}$
$X^T$
A
X
W
D×N
N×N
N×D
D×1
D×N
N×1
D×1

And $\lambda\mathbf{I}\mathbf{w}$ is simply the $D \times 1$ matrix $\mathbf{w}$ scaled by $\lambda$. $\mathbf{I}$ is the $D \times D$ identity matrix.

Setting $\frac{\partial L}{\partial w}$ to zero we have:

$$\begin{aligned}
0 &= \mathbf{X}^T\mathbf{A}\mathbf{y} - \mathbf{X}^T\mathbf{A}\mathbf{X}\mathbf{w} + \lambda\mathbf{I}\mathbf{w} \\
\mathbf{X}^T\mathbf{A}\mathbf{X}\mathbf{w} - \lambda\mathbf{I}\mathbf{w} &= \mathbf{X}^T\mathbf{A}\mathbf{y} \\
(\mathbf{X}^T\mathbf{A}\mathbf{X} - \lambda\mathbf{I})\mathbf{w} &= \mathbf{X}^T\mathbf{A}\mathbf{y} \\
\mathbf{w} &= (\mathbf{X}^T\mathbf{A}\mathbf{X} - \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{A}\mathbf{y}
\end{aligned}$$

which is the claim given in the question.

b) and c) are implemented in `q2.py`. Various plots are shown in d) - please see those.
Assumptions:

- Used my train/test split function from HW1 - unsure if we could use `sklearn.model_selection.train_test_split`
- Training and validation loss for question c is defined as the average of $\frac{1}{2}(y_{pred} - y_{target})^2$

d) As $\tau \to \infty$, each $a^{(i)}$ approaches $\frac{1}{N}$ - as if there is no local weighting happening at all and we are running a normal linear regression:

$$\begin{aligned}
\frac{e^{-(\|x-x^{(i)}\|/2\tau^2)}}{\sum_j e^{-(\|x-x^{(j)}\|/2\tau^2)}} &= \frac{\frac{1}{e^{(\|x-x^{(i)}\|/2\tau^2)}}}{\sum_j \frac{1}{e^{(\|x-x^{(j)}\|/2\tau^2)}}} \to_{\tau\to\infty} \frac{\frac{1}{e^{(\|x-x^{(i)}\|/2\infty^2)}}}{\sum_j \frac{1}{e^{(\|x-x^{(j)}\|/2\infty^2)}}} \\
&= \frac{\frac{1}{e^0}}{\sum_j \frac{1}{e^0}} = \frac{1}{\sum_j 1} = \frac{1}{N} \qquad \text{(where N is number of samples in the training set)}
\end{aligned}$$

The asymptotic behaviour as $\tau \to 0$ is that at the extreme, we'd assign all of the weight of the error calculation (for the purposes of finding $\mathbf{w*}$) to our test datum's closest training point. I found this a little tougher to prove, however:

$$\begin{aligned}
\frac{e^{-(\|x-x^{(i)}\|/2\tau^2)}}{\sum_j e^{-(\|x-x^{(j)}\|/2\tau^2)}} &= \frac{\frac{1}{e^{(\|x-x^{(i)}\|/2\tau^2)}}}{\sum_j \frac{1}{e^{(\|x-x^{(j)}\|/2\tau^2)}}} \to_{\tau\to 0} \frac{\frac{1}{e^{(\|x-x^{(i)}\|/2\cdot 0^2)}}}{\sum_j \frac{1}{e^{(\|x-x^{(j)}\|/2\cdot 0^2)}}} \\
&= \frac{\frac{1}{\text{VERY LARGE THING}}}{\sum_j \frac{1}{\text{VERY LARGE THING}}} = \frac{0}{\sum_j 0} = \frac{0}{0} \qquad \text{(?! Well, that's no good)}
\end{aligned}$$

Instead, intuitively consider the two broad cases we can have:

- $x$ is close to $x^{(i)}$. In the most extreme case of closeness, $\|x - x^{(i)}\| = 0$.
- Or, $x$ is not close to $x^{(i)}$, and $\|x - x^{(i)}\|$ = some positive quantity.

For the first case, $\|x - x^{(i)}\| = 0$: make a simplifying assumption that every other $x^{(j)}$ for $j \neq i$ is some positive distance away from $x$:

$$\frac{e^{-(\|x-x^{(i)}\|/2\tau^2)}}{\sum_j e^{-(\|x-x^{(j)}\|/2\tau^2)}} = \frac{\frac{1}{e^{(0/2\tau^2)}}}{\sum_j \frac{1}{e^{(\|x-x^{(j)}\|/2\tau^2)}}} \rightarrow_{\tau\rightarrow 0} \frac{\frac{1}{e^0}}{\sum_j \frac{1}{e^{(\|x-x^{(j)}\|/2\cdot 0^2)}}} = \frac{1}{\sum_j \frac{1}{e^{(\|x-x^{(j)}\|/2\cdot 0^2)}}}$$

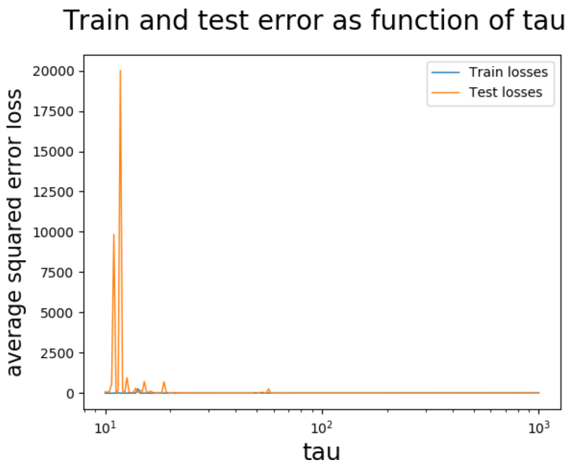
Note that the sum in the denominator can be split into the contribution from the element $j = i$ (which is 1, we already calculated it on top), and the contribution from all the other terms. These will be very small in comparison since $\|x - x^{(j)}\| > 0$ for $j \neq i$, and $\tau \rightarrow 0$ blows up the exponential as long as it has some non-zero norm to work with:

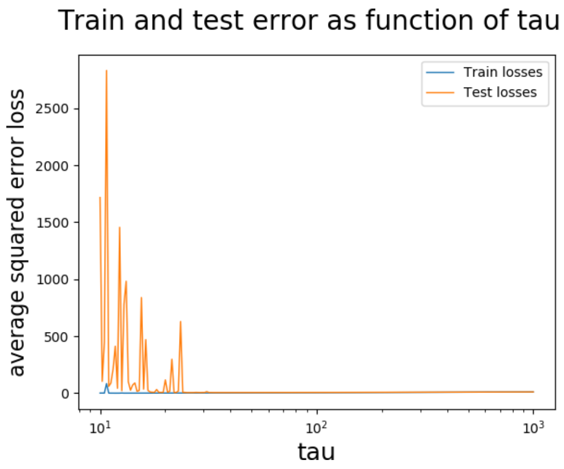
$$a^{(i)} \rightarrow_{\tau\rightarrow 0} \frac{1}{1 + \sum_{j\neq i} \frac{1}{\text{VERY LARGE THING}}} = \frac{1}{1 + (\text{lots of very small things})} \rightarrow \frac{1}{1}$$

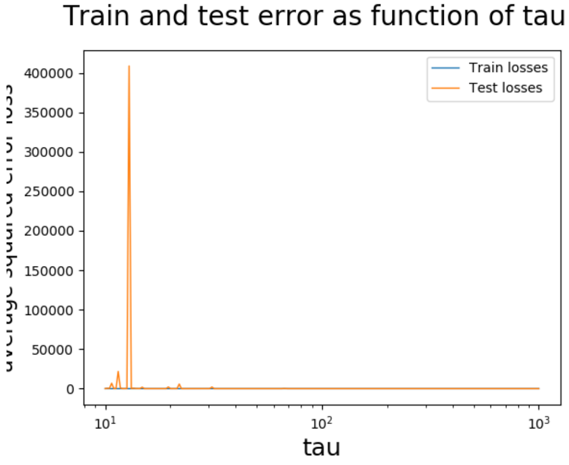
What about the weight $a^{(j)}$ for points that are far away? The sum in the denominator is a constant for every given test datum $x$ and its training points $(x^{(0)}...x^{(j)})$. For a far away point with a positive norm, the numerator would be something very small:

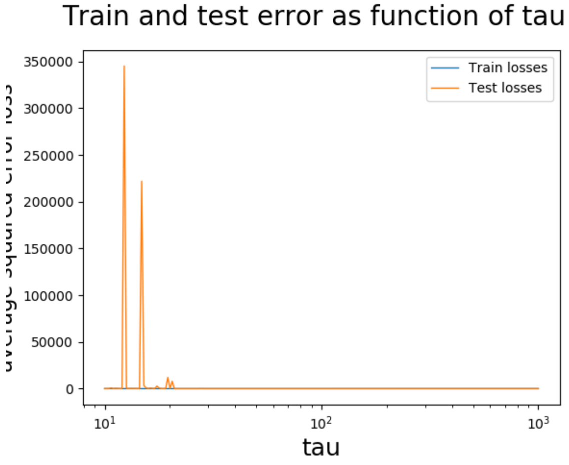
$$a^{(j\neq i)} \rightarrow_{\tau\rightarrow 0} \frac{(\text{one of those very small things})}{1 + (\text{lots of very small things})} = \text{something much less than } 1$$

I've hand-waved a bit, but as $\tau$ approaches zero, the influence of points close to test datum x grow disproportionately large. In the extreme, we'd end up calculating our optimal weights and predicting using a regression that only considers error of the **training point closest to x**. This causes very unstable behaviour in the validation set, as we can see for plots of various seed values in the train/val split:

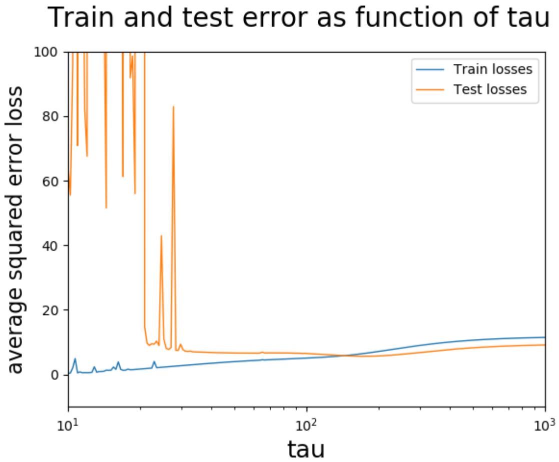

Train and test error as function of tau


Train and test error as function of tau


Train and test error as function of tau


Train and test error as function of tau

Zoom in on one of them, since the large validation error for small $\tau$ blows up the y axis:



We can see that training error approaches zero as $\tau \to 0$, as expected. If every point in the training set is predicted on the basis of the point closest to it (itself), everything will obviously be classified correctly.

This is of course at the expense of insane validation error, which fluctuates and goes up as the model is atrociously overfit for small $\tau$. If all we're considering is the nearest training point, there are infinitely many regressions which will predict it correctly, very few of which are likely to generalize well to the rest of the data.

The element of randomness in choosing the train/val split is also evident in the varying results: these figures actually change substantially with each new random seed. Any test datum from the validation set which, by chance, is far away from the $d$-dimensional cluster of training points would have a relatively equal Euclidean distance to everything in the training set, meaning that we'd largely lose any benefits to be had from using distance as a similarity measure.