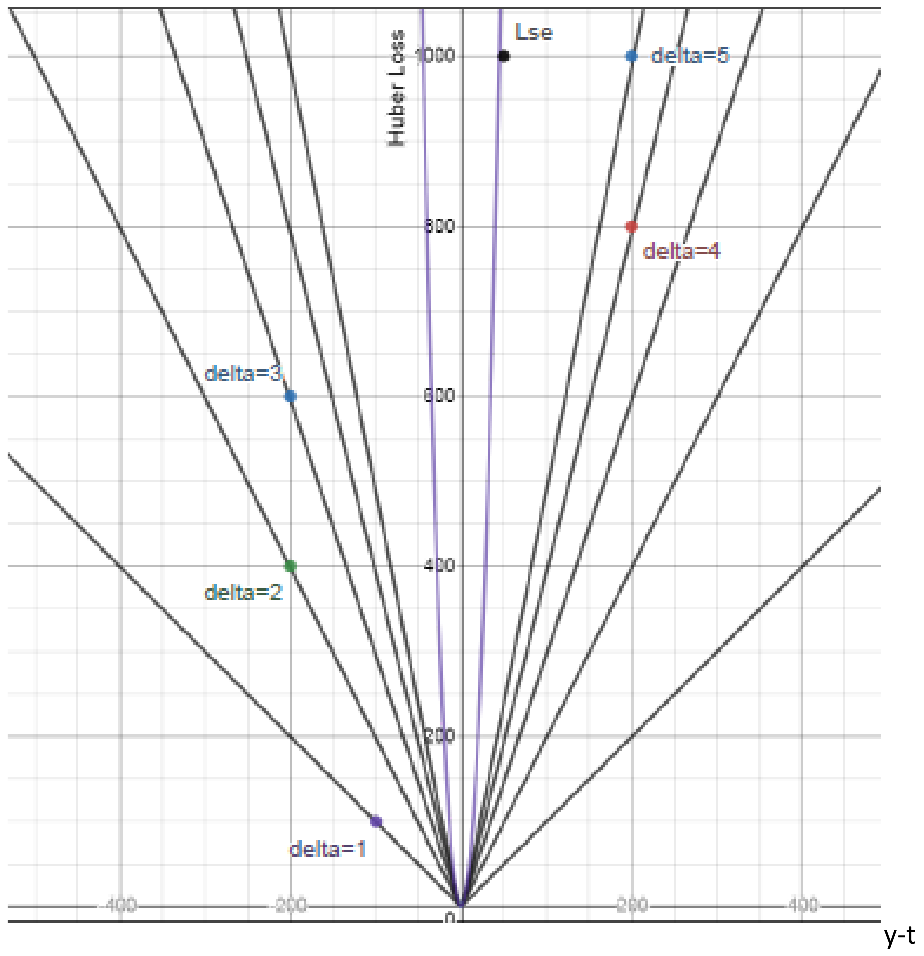
1. a)

Below is the graph of the Huber loss and squared error loss for t = 0. The different deltas for the Huber loss are displayed with their corresponding graph. There is one graph in the middle, the squared error loss, and is labelled as such.



Automatically, we can spot that the squared error loss increases exponentially faster than the Huber loss. This means that the loss function based upon squared error would attribute a much greater loss for outliers (y values that are large) than the Huber loss function, therefore influencing the gradient descent a lot more than the Huber loss which is what we do not want. We want outliers to be outliers and not affect our models which is where the Huber loss does better than the squared error loss.

b)

1. b). $\frac{\partial H}{\partial a_i} = \left\{ \begin{array}{ll} a & , |a| \le \delta \\ \delta \cdot \text{sign}(a) & , |a| > \delta \end{array} \right\}$

$$a_i = y^{(i)} - t^{(i)}$$

$$= \sum_{1}^{N} w_j x_j^{(i)} + b^{(i)} - t^{(i)}$$

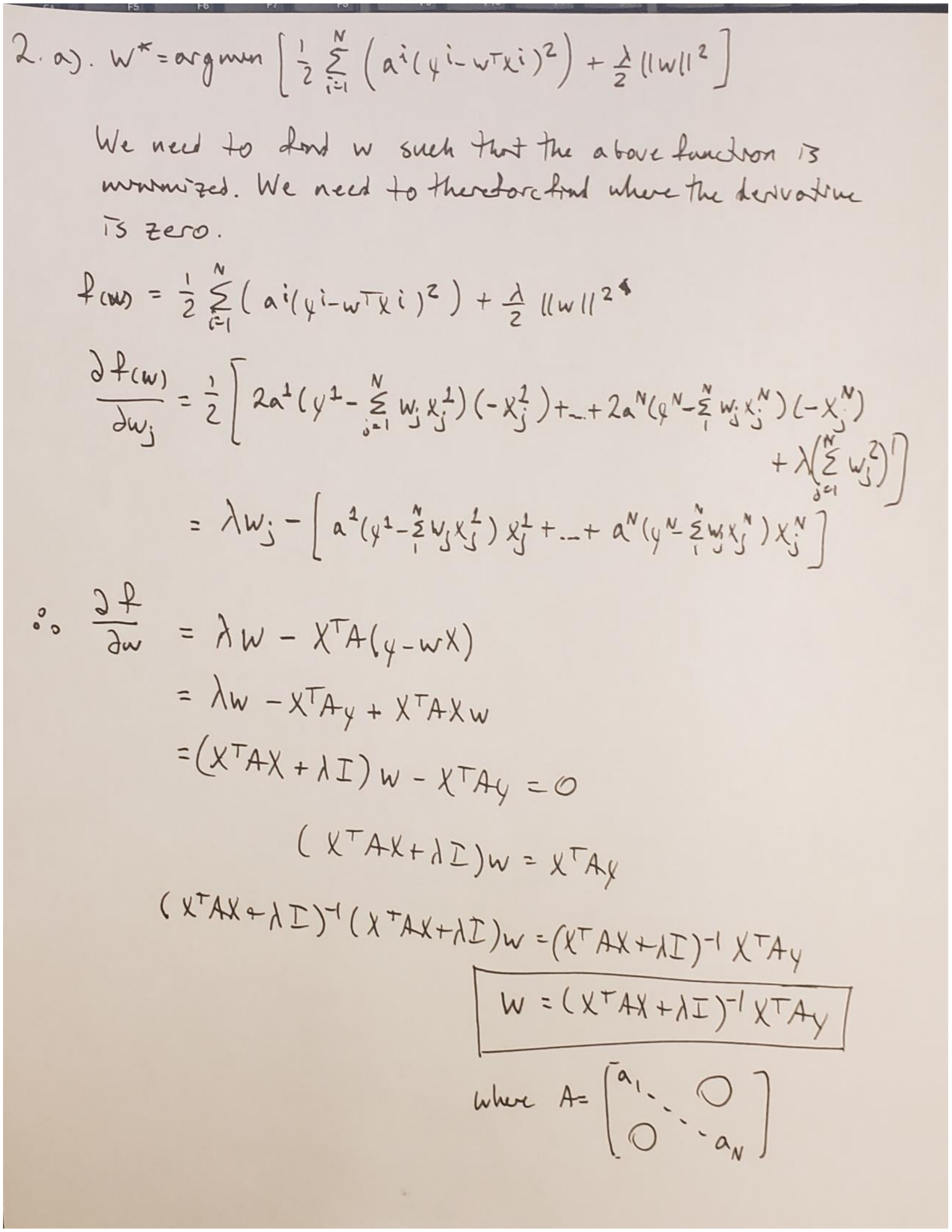
$$\frac{\partial a_i}{\partial w_j} = +x_j^{(i)} \quad , \quad \frac{\partial a_i}{\partial b^{(i)}} = 1$$

$$\therefore \frac{\partial H}{\partial w_j} = \frac{\partial H}{\partial a} \cdot \frac{\partial a}{\partial w_j} = \left\{ \begin{array}{ll} (y^{(i)} - t^{(i)}) x_j^{(i)} & , |y^{(i)} - t^{(i)}| \le \delta \\ \delta \cdot \text{sign}(y^{(i)} - t^{(i)}) x_j^{(i)} & , |y^{(i)} - t^{(i)}| > \delta \end{array} \right.$$

$$\frac{\partial H}{\partial b} = \frac{\partial H}{\partial a} \cdot \frac{\partial a}{\partial b} = \left\{ \begin{array}{ll} (y^{(i)} - t^{(i)}) & , |y^{(i)} - t^{(i)}| \le \delta \\ \delta \cdot \text{sign}(y^{(i)} - t^{(i)}) & , |y^{(i)} - t^{(i)}| > \delta \end{array} \right.$$

c). In q1.py

2. a)



2. a). $w^* = \arg\min \left[\frac{1}{2}\sum_{i=1}^{N} \left(a^i(y^i - w^T x^i)^2\right) + \frac{\lambda}{2}\|w\|^2 \right]$

We need to find w such that the above function is minimized. We need to therefore find where the derivative is zero.

$$f(w) = \frac{1}{2}\sum_{i=1}^{N}\left(a^i(y^i - w^T x^i)^2\right) + \frac{\lambda}{2}\|w\|^2$$

$$\frac{\partial f(w)}{\partial w_j} = \frac{1}{2}\Big[2a^1\big(y^1 - \sum_{j=1}^{N} w_j x_j^1\big)(-x_j^1) + \dots + 2a^N\big(y^N - \sum_{1}^{N} w_j x_j^N\big)(-x_j^N) + \lambda\big(\sum_{j=1}^{N} w_j^2\big)'\Big]$$

$$= \lambda w_j - \Big[a^1\big(y^1 - \sum_{1}^{N} w_j x_j^1\big)x_j^1 + \dots + a^N\big(y^N - \sum_{1}^{N} w_j x_j^N\big)x_j^N\Big]$$

$$\therefore \frac{\partial f}{\partial w} = \lambda w - X^T A(y - wX)$$

$$= \lambda w - X^T A y + X^T A X w$$

$$= (X^T A X + \lambda I)w - X^T A y = 0$$

$$(X^T A X + \lambda I)w = X^T A y$$

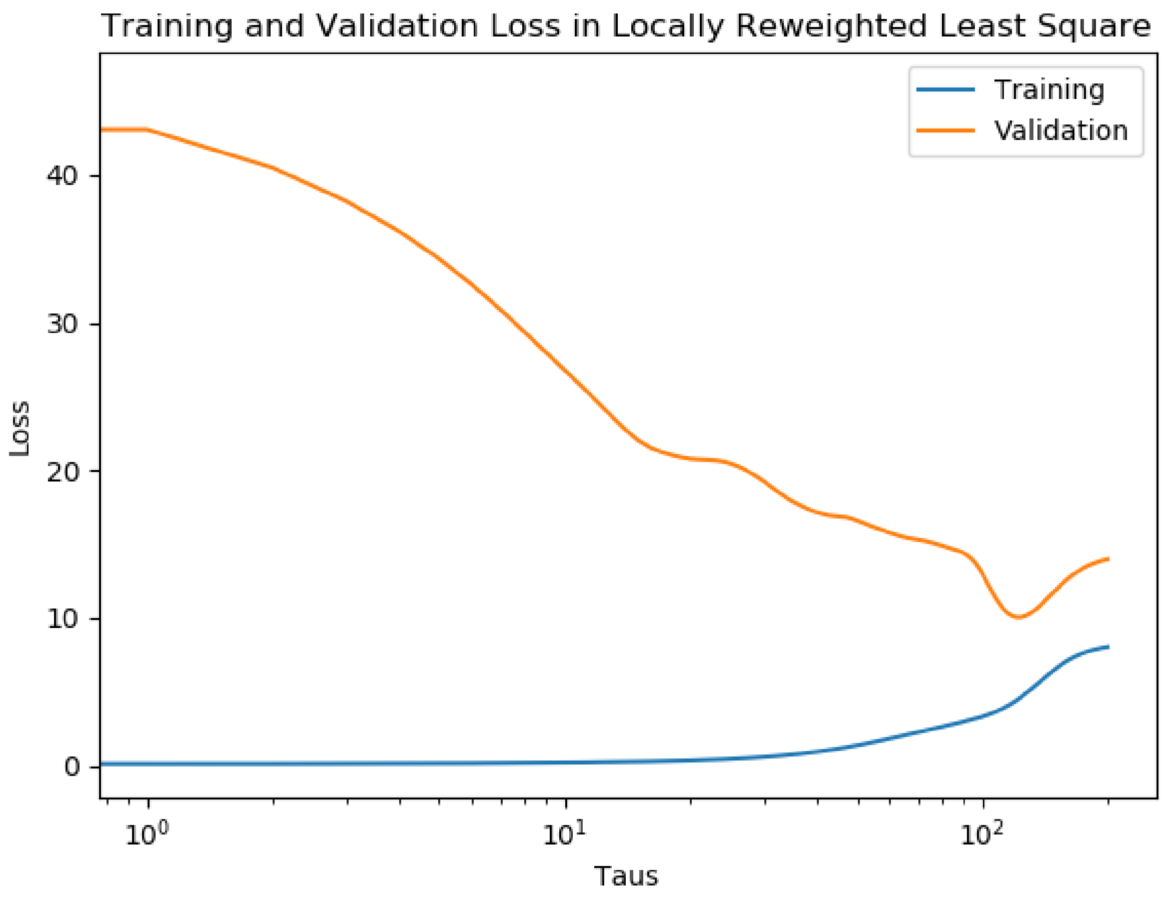
$$(X^T A X + \lambda I)^{-1}(X^T A X + \lambda I)w = (X^T A X + \lambda I)^{-1} X^T A y$$

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

where $A = \begin{bmatrix} a_1 & & 0 \\ & \ddots & \\ 0 & & a_N \end{bmatrix}$

b). In q2.py

c).



d).

As tau goes to infinity, you could see the graph levelling off, or in other words, the datasets converging to specific values. The loss would remain constant. This is what is actually happening in the graph above. As it goes to zero, we would see 0 training loss, and maximum validation loss. This is what is happening in the graph above. When tau goes to infinity, the A matrix becomes a constant because the exponentials tau contributes to become e^0 = 1 which is what is happening above.