1. Let y be a linear predictor with form $y = w^T x$ with $w \in \mathbb{R}^D$

Let $x_1, x_2 \in \mathbb{R}^D$, $a \in \mathbb{R}$

$x_1 = (x_{11}, x_{12}, \dots, x_{1D})$, $x_2 = (x_{21}, x_{22}, \dots, x_{2D})$, $x_{ij} \in \mathbb{R}$ for $i \in \{1,2\}, j \in \{1, \dots, D\}$

Then $y(x_1) = w^T x_1 = \sum_{j=1}^{D} w_j x_{1j}$ $\quad y(x_2) = w^T x_2 = \sum_{j=1}^{D} w_j x_{2j}$

Then $y(x_1) + y(x_2) = \sum_{j=1}^{D} w_j x_{1j} + \sum_{j=1}^{D} w_j x_{2j} = \sum_{j=1}^{D} w_j (x_{1j} + x_{2j}) = y(x_1 + x_2)$

So $y(x_1) + y(x_2) = y(x_1 + x_2)$

And $y(ax_1) = \sum_{j=1}^{D} a w_j x_{1j} = a \sum_{j=1}^{D} w_j x_{1j} = a y(x_1)$

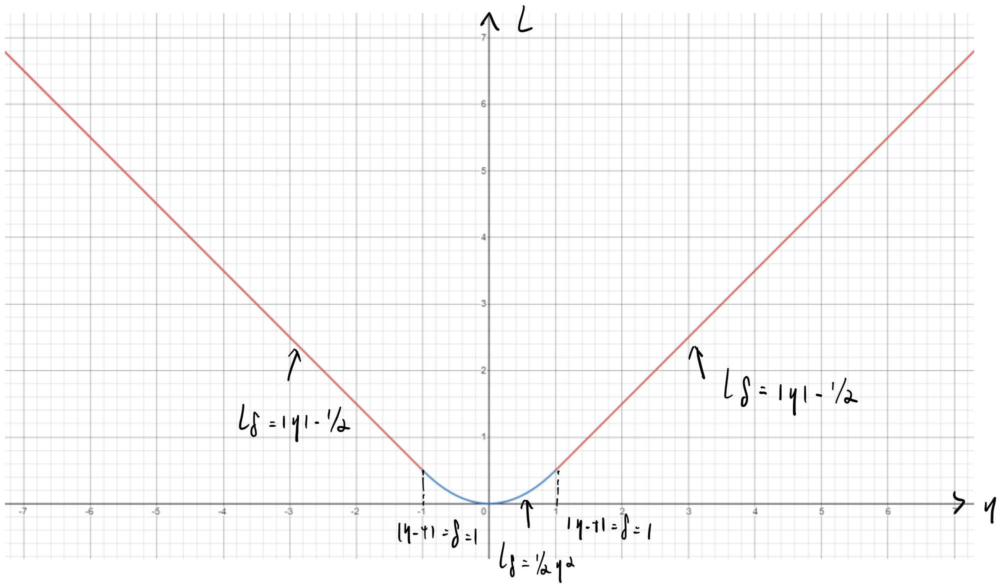
Since x_1, x_2, a are fixed arbitrarily

So $\forall x_1, x_2 \in \mathbb{R}^D$, $\forall a \in \mathbb{R}$ $y(x_1 + x_2) = y(x_1) + y(x_2)$, $y(ax_1) = a y(x_1)$

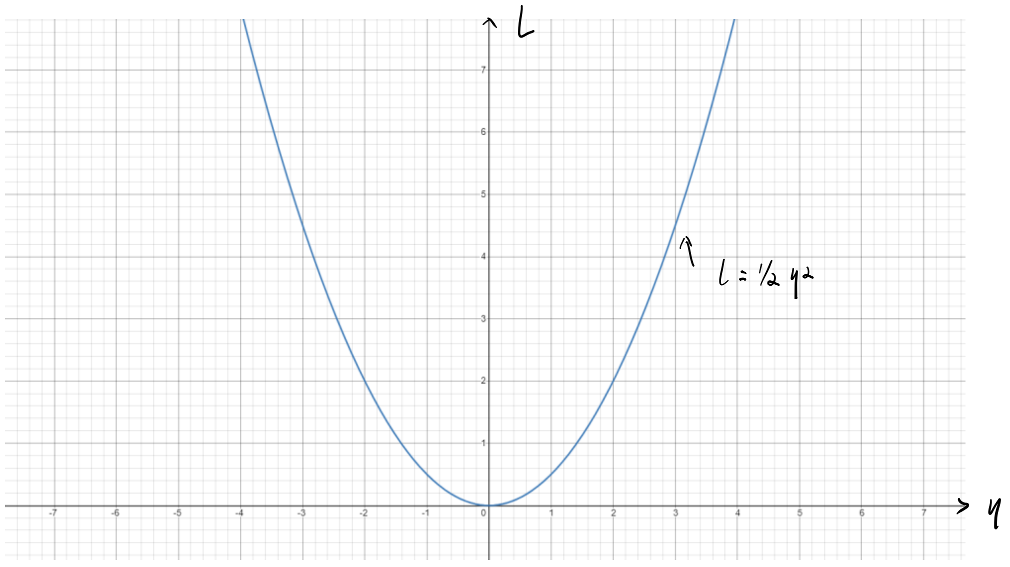
So y is a linear function

2a) Take delta=1

Huber Loss when $\delta = 1$:



Squared Error Loss:



From the graph, we can see that the loss of the Huber loss is less than the squared error loss when $|y-t| > 1$, where we consider the label t as an outlier. So, the Huber loss is more robust to the outliers.

2b) · First, want to find $H'_\delta(a)$

$$H_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{if } |a| \leq \delta \\ \delta(|a| - \frac{1}{2}\delta) & \text{if } |a| > \delta \end{cases}$$

Since $\lim_{|a| \to \delta^-} H_\delta(a) = \lim_{|a| \to \delta^+} H_\delta(a) = \frac{1}{2}\delta^2$

so $H_\delta(a)$ is differentiable at $|a| = \delta$

There are 2 cases

Case 1: $|a| \leq \delta$

Then $H'_\delta(a) = a$

Case 2: $|a| > \delta$

Then $H'_\delta(a) = \delta$ if $a > 0$

$H'_\delta(a) = -\delta$ if $a < 0$

$$\text{Then } H_\delta'(a) = \begin{cases} -\delta & \text{if } a < -\delta \\ a & \text{if } -\delta \leq a \leq \delta \\ \delta & \text{if } a > \delta \end{cases}$$

$$\text{so } H'_\delta(y - t) = \begin{cases} -\delta & \text{if } y - t < -\delta \\ y - t & \text{if } -\delta \leq y - t \leq \delta \\ \delta & \text{if } y - t > \delta \end{cases}$$

Since $H'_\delta(y-t) = \begin{pmatrix} H'_\delta(y^{(1)} - t^{(1)}) \\ \vdots \\ H'_\delta(y^{(N)} - t^{(N)}) \end{pmatrix}$ And $\hat{R} = \frac{1}{N}\sum_{i=1}^{N} H_\delta(y^{(i)} - t^{(i)})$

$$\text{so } \frac{\partial \hat{R}}{\partial y} = \begin{pmatrix} \partial \hat{R}/\partial y^{(1)} \\ \vdots \\ \partial \hat{R}/\partial y^{(N)} \end{pmatrix} = \begin{pmatrix} \frac{1}{N} H'_\delta(y^{(1)} - t^{(1)}) \\ \vdots \\ \frac{1}{N} H'_\delta(y^{(N)} - t^{(N)}) \end{pmatrix} = \frac{1}{N} H'_\delta(y - t)$$

$y = Xw$ with $X = \begin{pmatrix} x^{(1)T} \\ \vdots \\ x^{(N)T} \end{pmatrix}$ $w = \begin{pmatrix} w_1 \\ \vdots \\ w_D \end{pmatrix}$

For arbitrary $i \in \{1, 2, \dots, D\}$, $k \in \{1, 2, \dots, N\}$

$$\partial H_\delta(y^{(k)} - t^{(k)}) / \partial w_i = \partial H_\delta(y^{(k)} - t^{(k)}) / \partial y^{(k)} \cdot \partial y^{(k)} / \partial w_i$$

$$= H'_\delta(y^{(k)} - t^{(k)}) \cdot X_{ki}$$ (X_{ki} is the ki th entry of X)

So $\partial H_\delta(y^{(k)} - t^{(k)}) / \partial w = X_k^T H'_\delta(y^{(k)} - t^{(k)})$ (X_k is the kth row of X)

So $\partial \hat{R} / \partial w = \partial / \partial w \left(1/N \sum_{i=1}^{N} H_\delta(y^{(i)} - t^{(i)}) \right)$

$$= 1/N \; \partial / \partial w \left(\sum_{i=1}^{N} H_\delta(y^{(i)} - t^{(i)}) \right)$$

$$= 1/N \sum_{i=1}^{N} X_i^T H'_\delta(y^{(i)} - t^{(i)})$$ (X_i is the ith row of X)

$$= 1/N \; X^T \begin{pmatrix} H'_\delta(y^{(1)} - t^{(1)}) \\ \vdots \\ H'_\delta(y^{(N)} - t^{(N)}) \end{pmatrix}$$

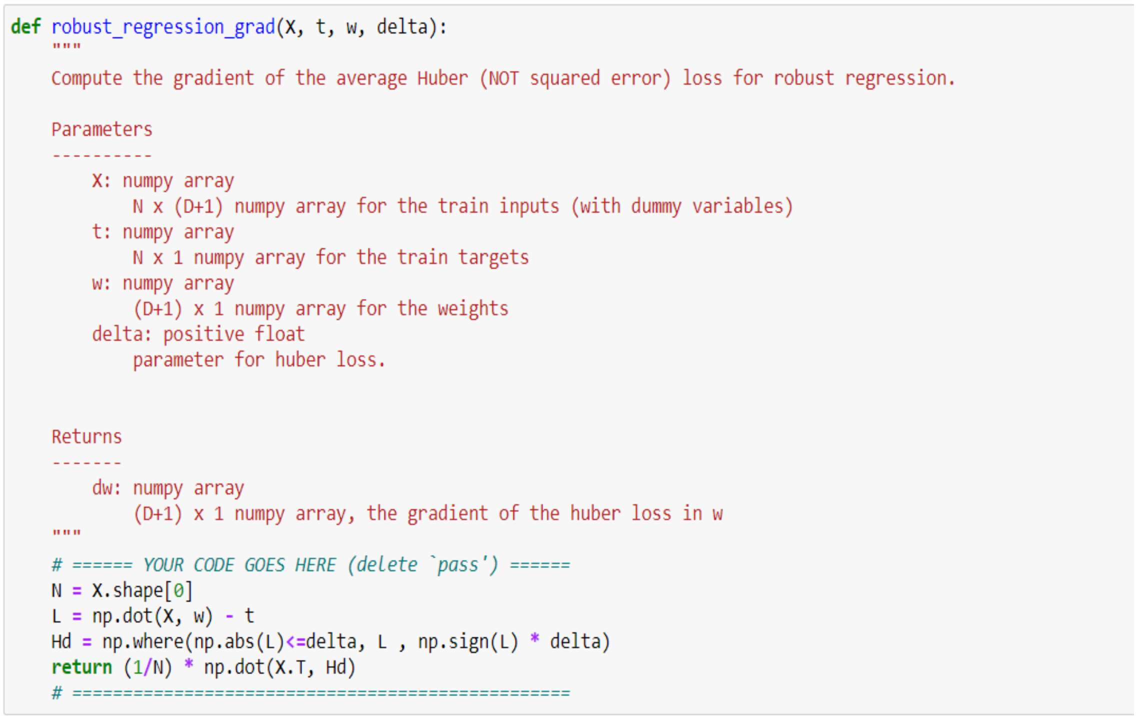
$$= 1/N \; X^T H'_\delta(y - t)$$

As a result, $y = \begin{pmatrix} x^{(1)T} \\ \vdots \\ x^{(N)T} \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_D \end{pmatrix}$

$$\partial \hat{R} / \partial y = 1/N \; H'_\delta(y - t)$$

$$\partial \hat{R} / \partial w = 1/N \; X^T H'_\delta(y - t)$$

2c)

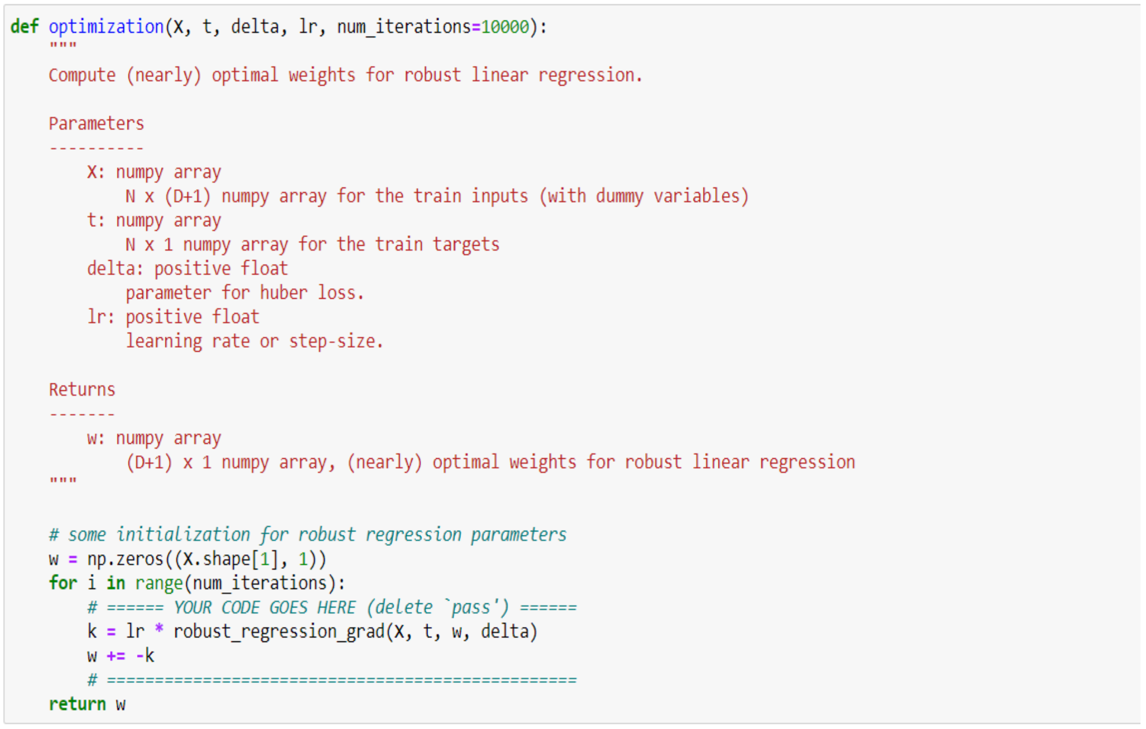


```python
def robust_regression_grad(X, t, w, delta):
    """
    Compute the gradient of the average Huber (NOT squared error) loss for robust regression.

    Parameters
    ----------
        X: numpy array
            N x (D+1) numpy array for the train inputs (with dummy variables)
        t: numpy array
            N x 1 numpy array for the train targets
        w: numpy array
            (D+1) x 1 numpy array for the weights
        delta: positive float
            parameter for huber loss.


    Returns
    -------
        dw: numpy array
            (D+1) x 1 numpy array, the gradient of the huber loss in w
    """
    # ====== YOUR CODE GOES HERE (delete `pass') ======
    N = X.shape[0]
    L = np.dot(X, w) - t
    Hd = np.where(np.abs(L)<=delta, L , np.sign(L) * delta)
    return (1/N) * np.dot(X.T, Hd)
    # =================================================
```

2d)



```python
def optimization(X, t, delta, lr, num_iterations=10000):
    """
    Compute (nearly) optimal weights for robust linear regression.

    Parameters
    ----------
        X: numpy array
            N x (D+1) numpy array for the train inputs (with dummy variables)
        t: numpy array
            N x 1 numpy array for the train targets
        delta: positive float
            parameter for huber loss.
        lr: positive float
            learning rate or step-size.

    Returns
    -------
        w: numpy array
            (D+1) x 1 numpy array, (nearly) optimal weights for robust linear regression
    """

    # some initialization for robust regression parameters
    w = np.zeros((X.shape[1], 1))
    for i in range(num_iterations):
        # ====== YOUR CODE GOES HERE (delete `pass') ======
        k = lr * robust_regression_grad(X, t, w, delta)
        w += -k
        # =================================================
    return w
```

2e) The output is:

```
linear regression validation loss: 28.442028967907067
delta: 0.1, valid. squared error loss: 4.909228730093668, train squ
ared error loss: 38.04144442529113
delta: 0.5, valid. squared error loss: 4.0189727619053155, train sq
uared error loss: 38.311166891032116
delta: 1, valid. squared error loss: 4.200164017924187, train squar
ed error loss: 37.993439863363456
delta: 5, valid. squared error loss: 23.636105018763626, train squa
red error loss: 28.84756778475285
delta: 10, valid. squared error loss: 26.420467729491516, train squ
ared error loss: 27.79829743150903
```

When delta is small, the training squared loss is larger since for the Huber loss model, the loss brought by most of the outlier has been reduced, so there is a small impact of the outlier on w. While the training squared loss will still take the outlier into account, so the training loss is larger for the smaller delta. Besides, the larger delta is, less impact brought by the outliers will be reduced, so that the training squared error of the Huber loss will be like the error of the linear regression model. In terms of the validation squared loss, it is minimized when delta=0.5. Since in the validation set, we assume there is no outlier. So, we need to reduce the impact of the outlier from the training sets by fit the Huber loss model. In that case we need a smaller delta (not too small, since in that case points that are not outliers will be considered as one) to reduce the validation squared error.

3a) When x=-1 and x=3, they have the same t value, 1. Since if two points have the same prediction and the data set is linearly separable, then all points on the line segment connecting them also have the same prediction. Since 1 is on the line segment between -1 and 3, so when x=1, t should be 1. However, x=1, t=0. So, this data set is not linearly separable.

since $1 = \frac{1}{2} \times (-1) + (1 - \frac{1}{2}) \times 3$

3b) Take $a = 0$

Then $z = w^T \varphi(x)$

$= w^T \begin{pmatrix} x \\ x^2 \end{pmatrix}$

$= w_1 x + w_2 x^2$

since $x = -1$, $T = 1$, $-w_1 + w_2 \geq 0$

$x = 1$, $T = 0$, $w_1 + w_2 < 0$

$x = 3$, $T = 1$, $3w_1 + 9w_2 \geq 0$

so $\begin{cases} w_1 < -w_2 \\ w_1 \leq w_2 \\ w_1 \geq -3w_2 \end{cases}$

One possible pair is $(w_1, w_2) = (-2, 1)$