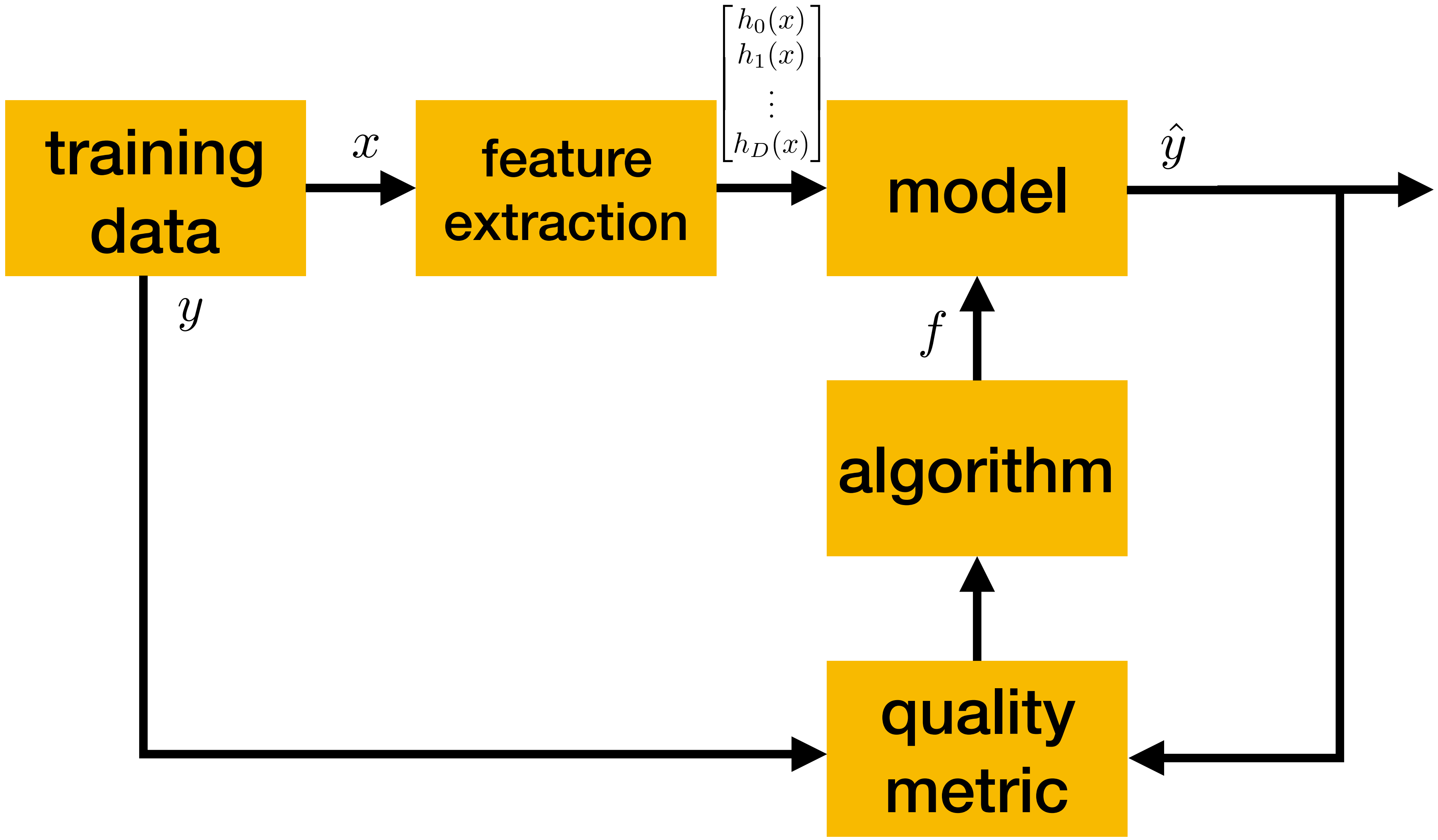
Non-quadratic Loss

Sewoong Oh

CSE/STAT 416
University of Washington

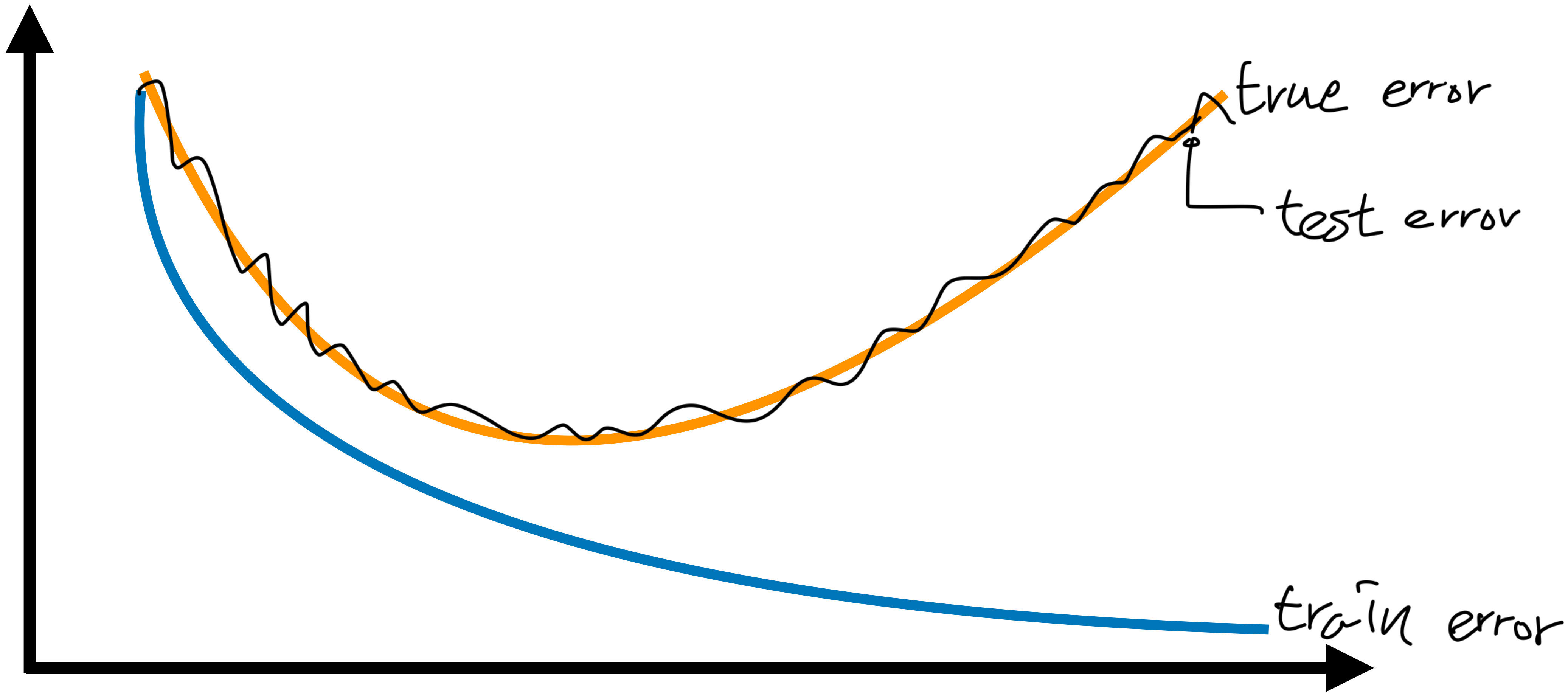
Recap



In [1. Regression], we studied linear regression with L2 loss:

$$\text{minimize}_w \quad \underbrace{\sum_{i=1}^{N} \Big(\underbrace{\hat{y}_i}_{\text{linear model:} w^T h(x_i)} - y_i \Big)^2}_{\text{quality metric: quadratic loss (i.e. L2 loss, squared loss)}}$$

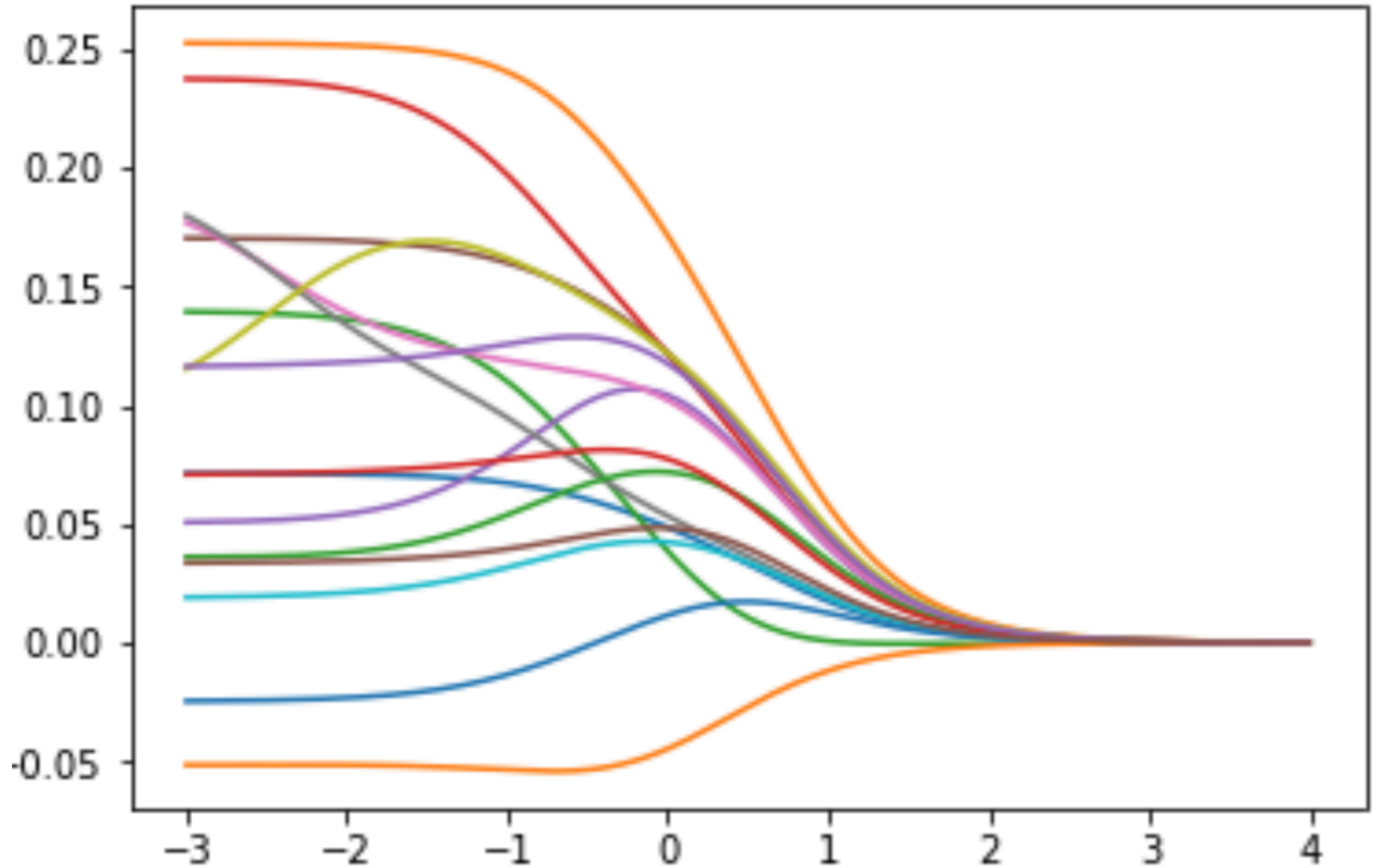
Recap



$$\text{minimize}_w \quad \underbrace{\sum_{i=1}^{N} \Big(\underbrace{\hat{y}_i}_{\text{linear model:} w^T h(x_i)} - y_i \Big)^2}_{\text{quality metric: quadratic loss (i.e. L2 loss, squared loss)}}$$

In [2. Validation], we studied tradeoffs between test error, training error, sample size, model complexity; and cross validation:

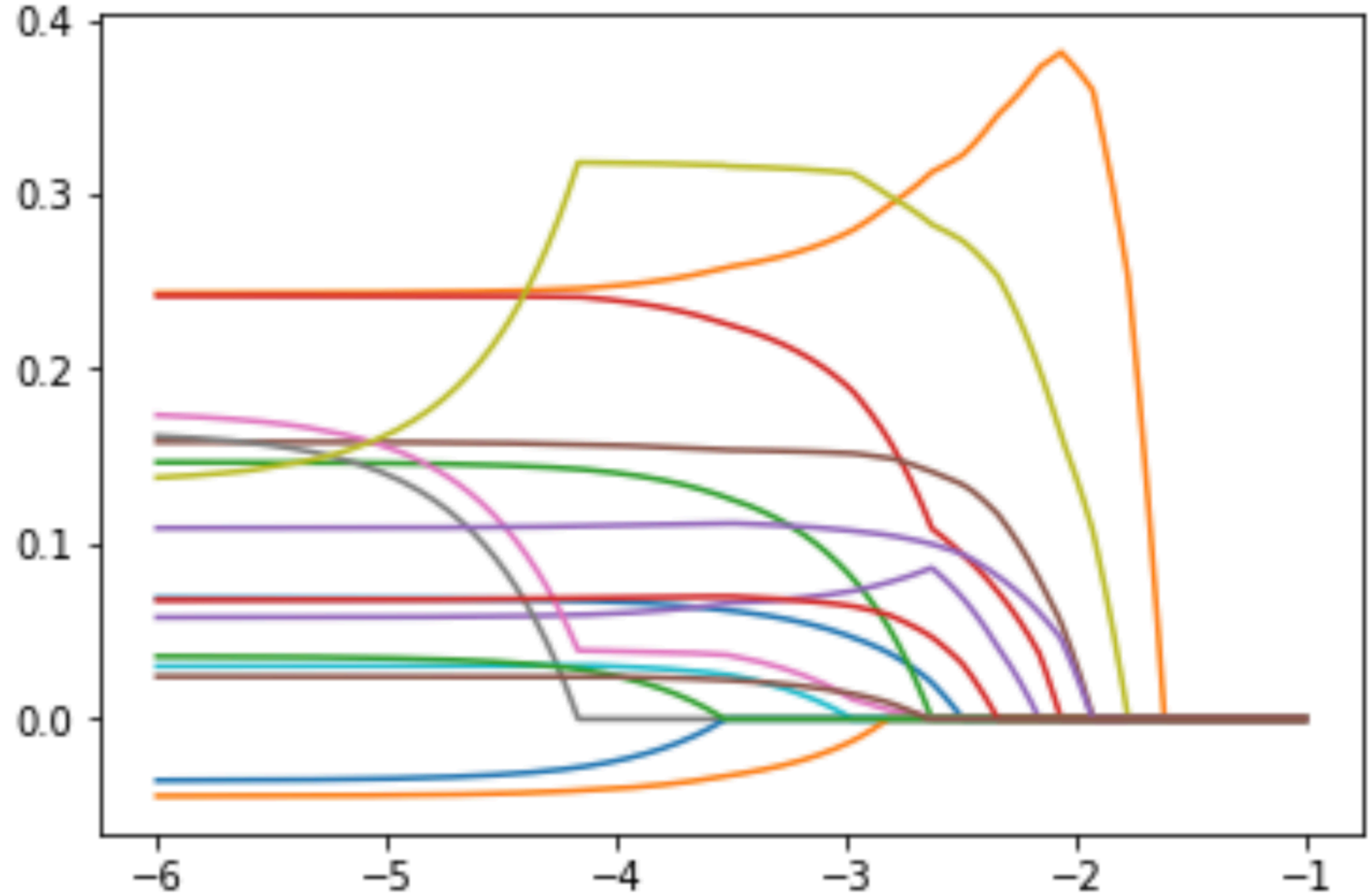
Recap



In [3. Regularization], we studied Ridge regression:

$$\text{minimize}_w \underbrace{\sum_{i=1}^{N} \Big(\underbrace{\hat{y}_i}_{w^T h(x_i)} - y_i \Big)^2}_{\text{L2 loss}} + \lambda \underbrace{r(w)}_{\|w\|^2 = \sum_{j=1}^{d} w_j^2}$$

Recap



In [4. Non-quadratic Regularization], we studied Lasso regression:

$$\text{minimize}_w \underbrace{\sum_{i=1}^{N} \Big(\underbrace{\hat{y}_i}_{w^T h(x_i)} - y_i \Big)^2}_{\text{L2 loss}} + \lambda \underbrace{r(w)}_{\|w\|_1 = \sum_{j=1}^{d} |w_j|}$$

This lecture…

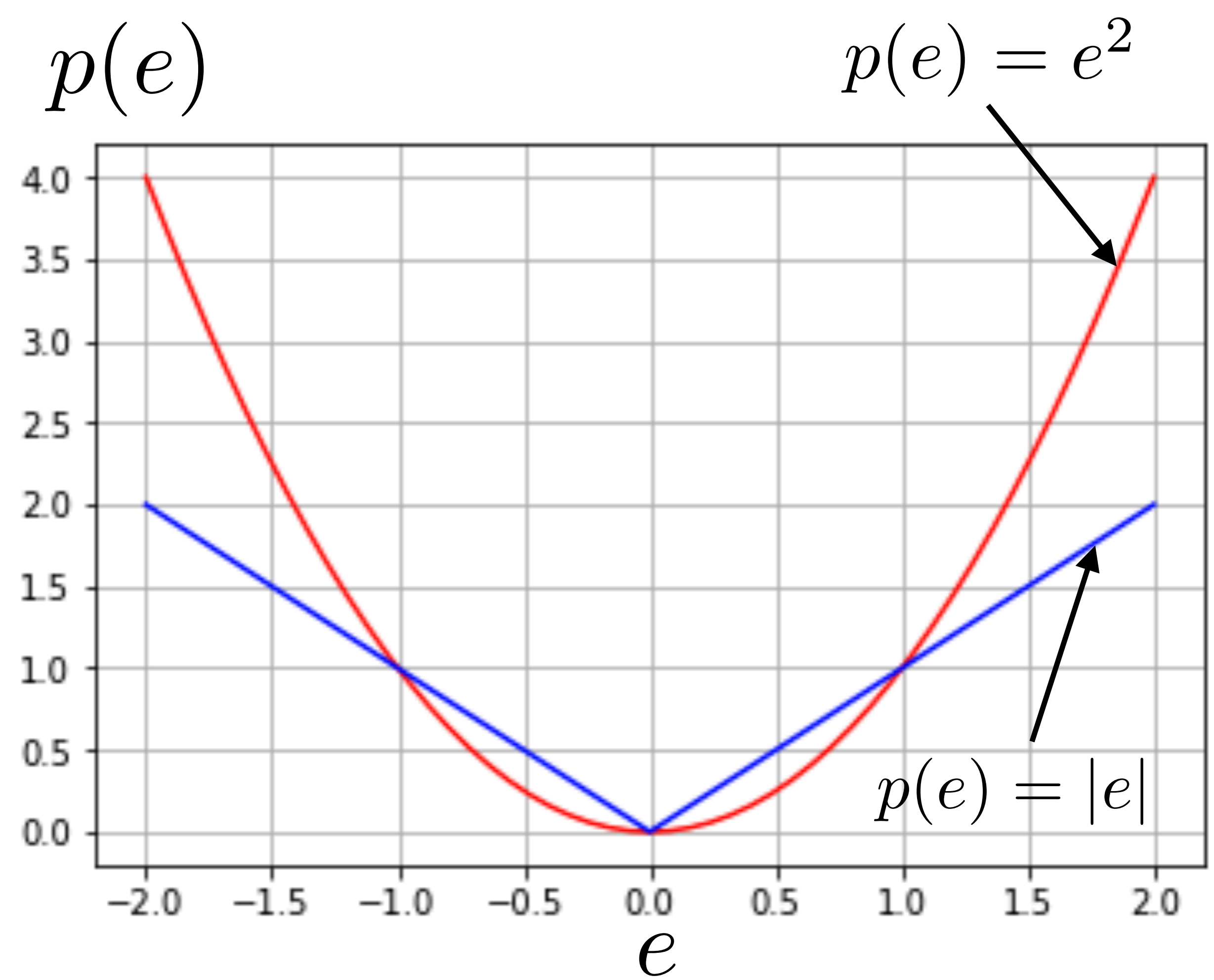
$$\text{minimize}_w \quad \underbrace{\sum_{i=1}^{N} p\Big(\underbrace{\hat{y}_i}_{w^T h(x_i)} - y_i \Big)}_{\text{generic loss}}$$

- we study generic loss defined by the penalty function

$$p(\hat{y} - y)$$

for example

$$\text{L2 loss: } p(\hat{y} - y) = (\hat{y} - y)^2$$
$$\text{L1 loss: } p(\hat{y} - y) = |\,\hat{y} - y\,|$$



Loss and penalty functions

- In training a linear model, we minimize an average loss:

$$\mathcal{L}(w) \;=\; \frac{1}{n}\sum_{i=1}^{n} \ell(\underbrace{w^T x_i}_{\hat{y}_i}, y_i)$$

- here, $\ell(\hat{y}, y)$ is called the **loss function**, and penalizes the deviation between the predicted value $\hat{y}$ and the observed value y

- so far, we use L2 or quadratic loss

$$\ell(\hat{y}, y) \;=\; (\hat{y} - y)^2$$

- Typically we use loss function of the form

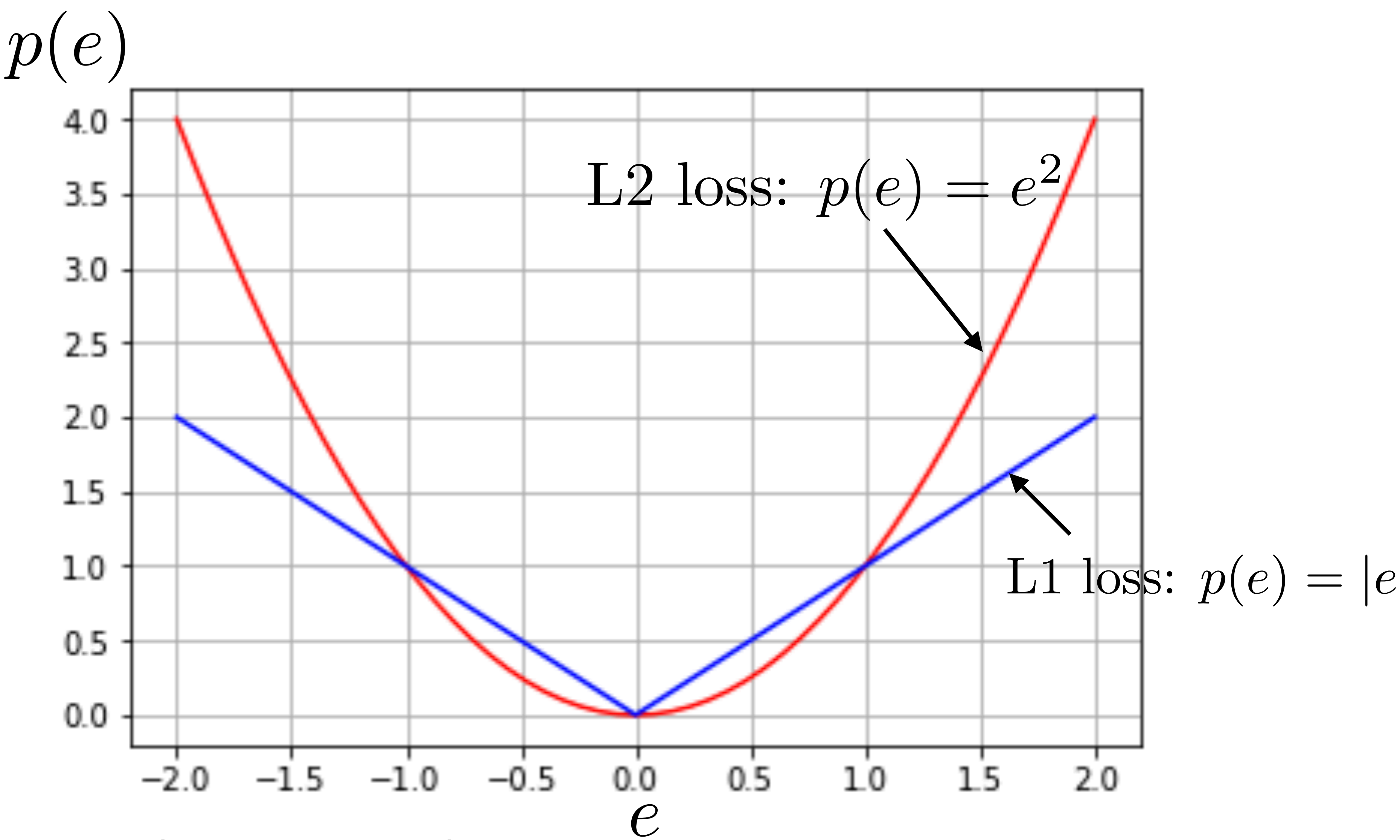
$$p(\hat{y} - y)$$

  where p( ) is called the **penalty function**

- e= $\hat{y} - y$ Is called the **residual** or **prediction error**

Predictor and choice of penalty function

- Choice of penalty function depends on how you want to penalize large, small, positive, negative errors
- Different choice of penalty functions yield different predictor parameters ***w***



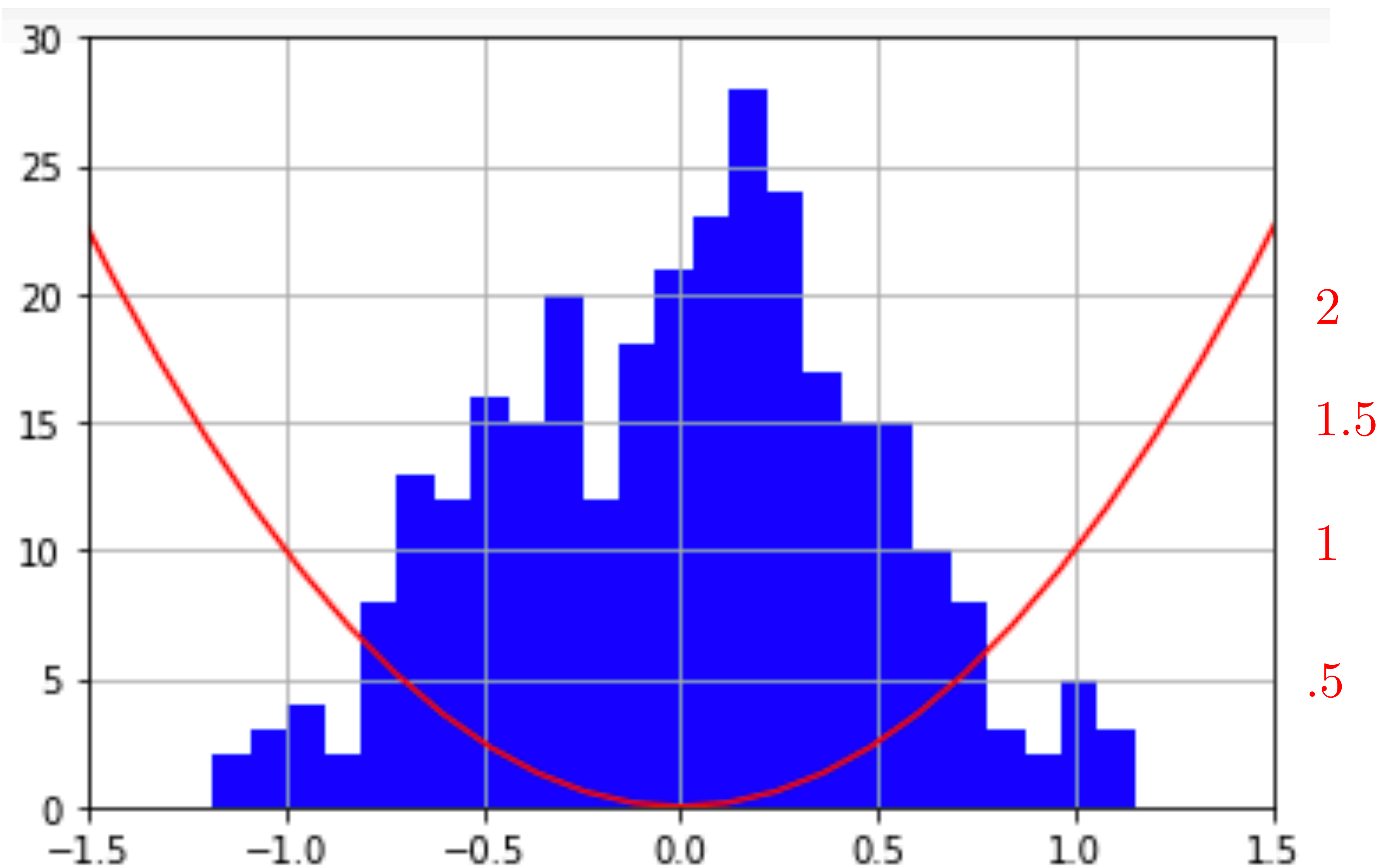
- Square penalty (a.k.a. L2 loss) is very large for large error, and very small for small
- Absolute penalty (a.k.a. L1 loss) is smaller for large error, and larger for small error

- Choice of penalty function **shapes** the histogram of prediction errors:

$$\{\hat{y_1} - y_1, \cdots, \hat{y_N} - y_N\}$$

Blue bars are the histogram of errors after training

Red line shows the penalty function of L2 loss: $p(e) = e^2$

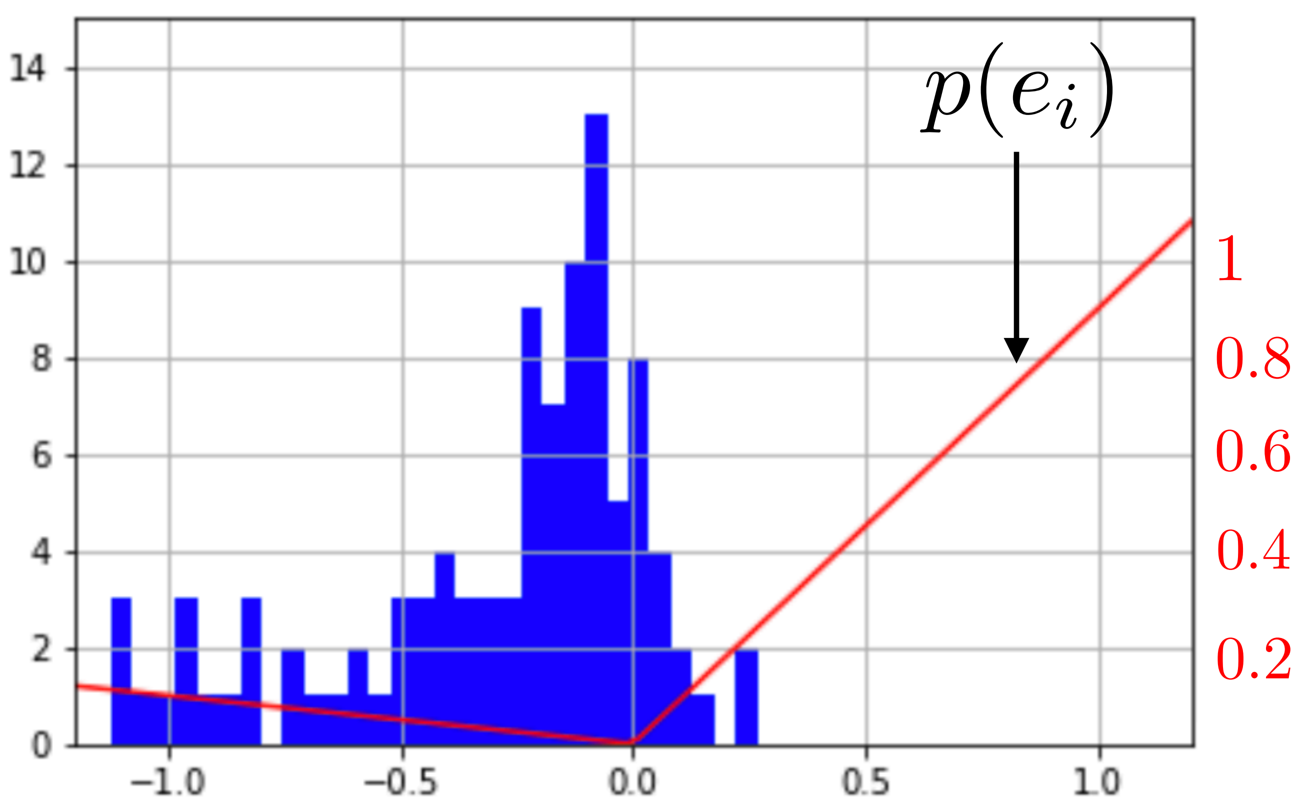


$$e_i = \hat{y}_i - y_i$$

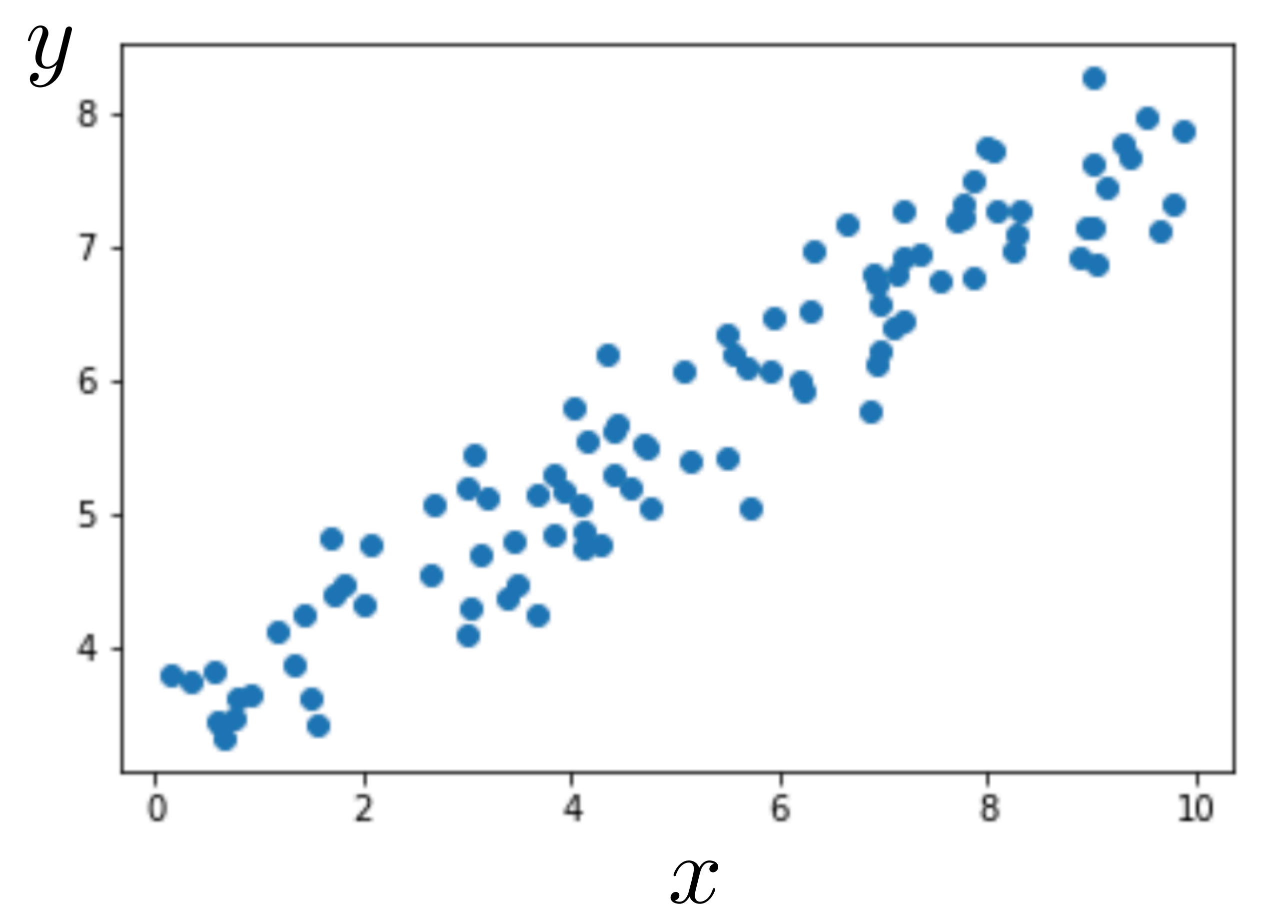
- Choice of penalty function **shapes** the histogram of prediction errors:

$$\{\hat{y_1} - y_1, \cdots, \hat{y_N} - y_N\}$$

- p(e) can be asymmetric, in which case we care more about over-estimating or under-estimating



$$e_i = \hat{y}_i - y_i$$

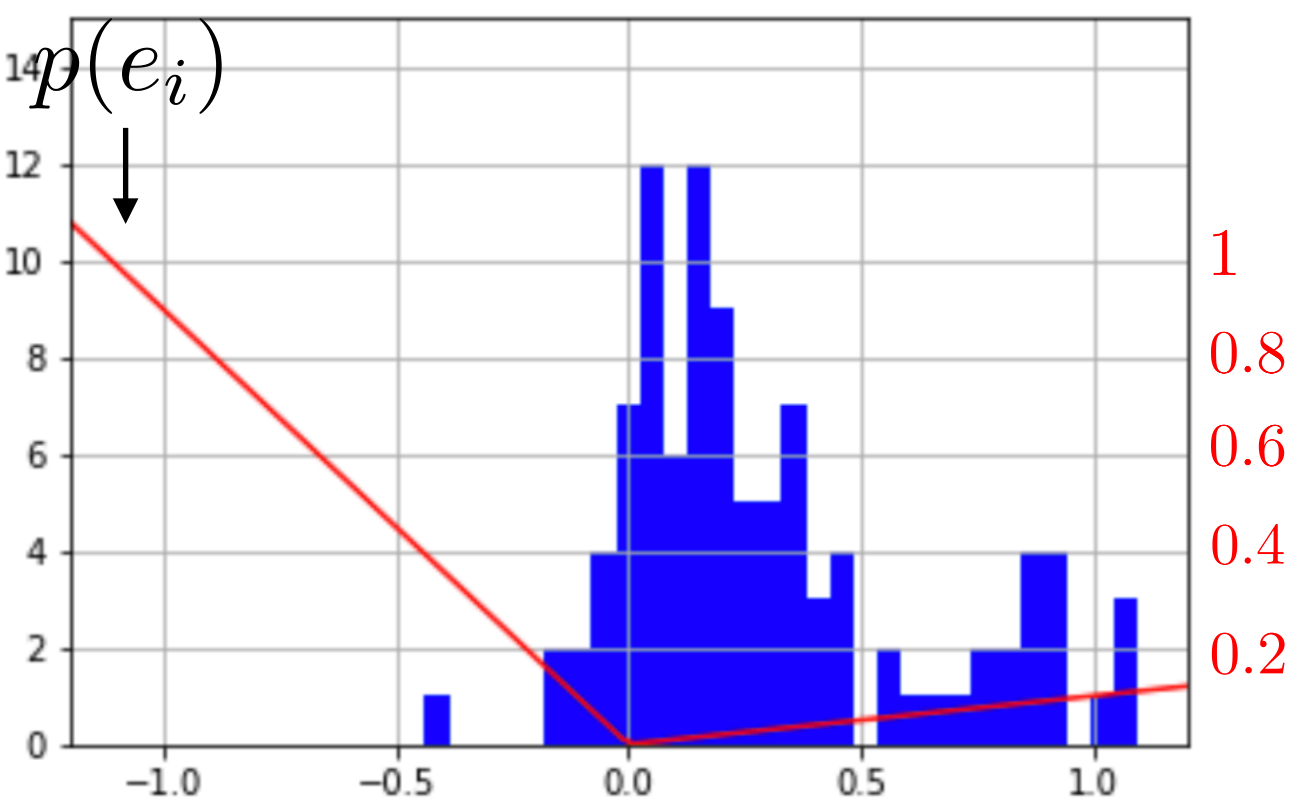


- Larger slope on the positive side means we penalize over-estimation heavily

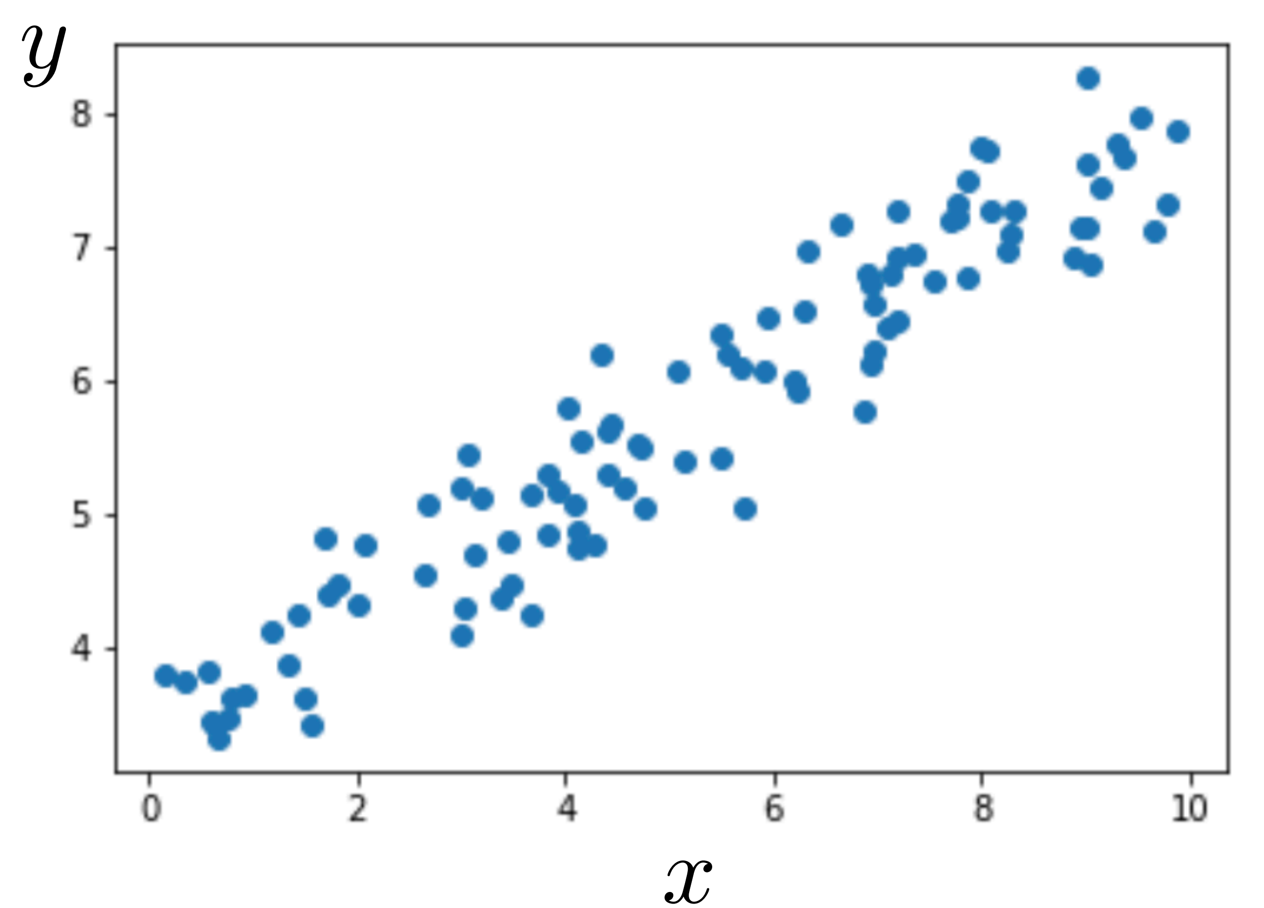
- Choice of penalty function **shapes** the histogram of prediction errors:

$$\{\hat{y_1} - y_1, \cdots, \hat{y_N} - y_N\}$$

- p(e) can be asymmetric, in which case we care more about over-estimating or under-estimating



$$e_i = \hat{y}_i - y_i$$



- Larger slope on the negative side means we penalize under-estimation heavily

Robust fitting

- **Outliers**
  - Commonly, **a few** data points are ‘**way off**’ in real datasets
  - Even just a few outliers in a data set can make prediction poor

- one simple method for fighting outliers
  - 1. Train a predictor
  - 2. Flag data points with large prediction errors as outliers
  - 3. Remove them from data set and retrain

- We want a principled method that can fight outliers

Robust penalty function

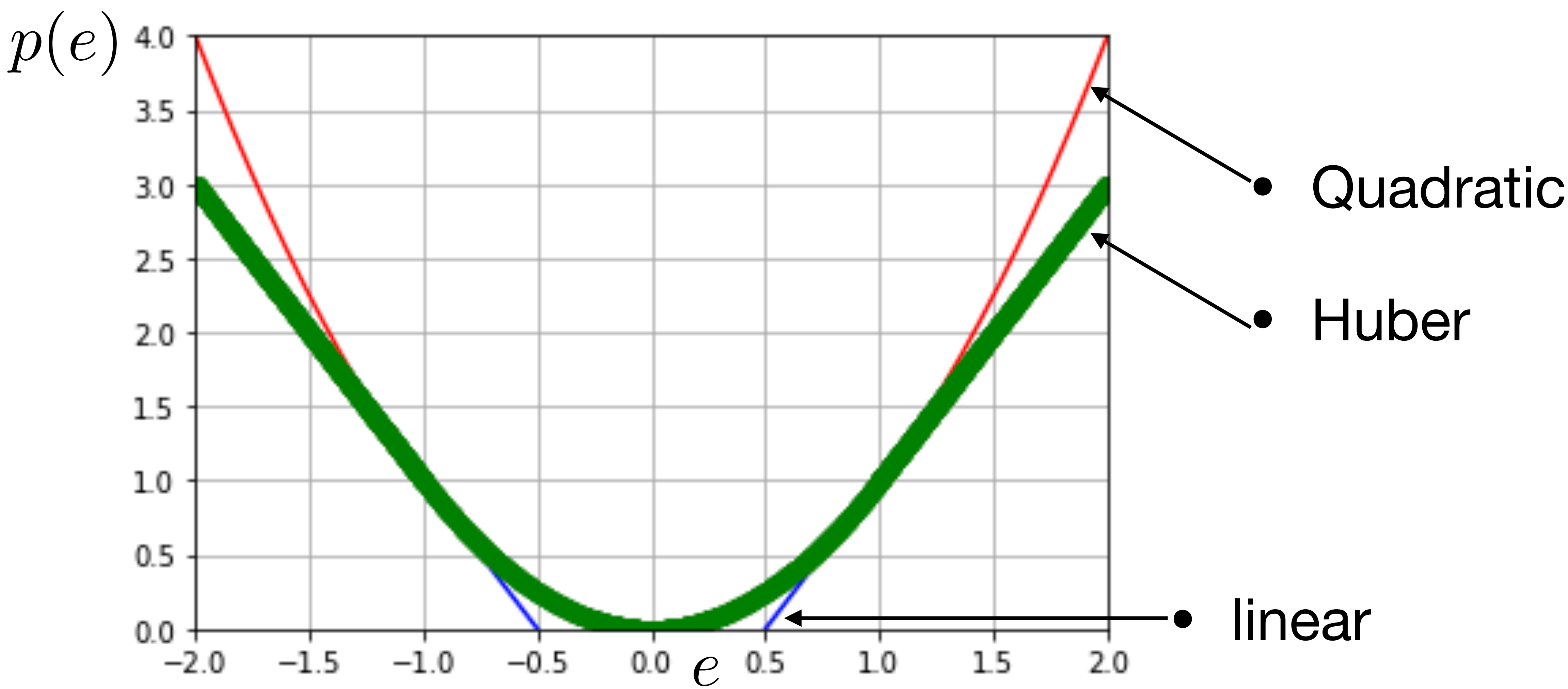
- We say a penalty function is **robust** if it has low sensitivity to outliers
- Robust penalty functions grow more slowly compared to the typical square penalty function
- This allows the predictor to tolerate a few large prediction errors (presumably for the outliers)
- So they handle outliers more gracefully
- For example, a **robust predictor** might fit 98% of the data very well
- Whereas a square penalty might try to fit 100% of data well (and fail if there are outliers)

Huber loss

- The **Huber** penalty function is

$$p(e) = \begin{cases} e^2 & \text{if } |e| \leq a \\ a(2|e| - a) & \text{if } |e| > a \end{cases}$$

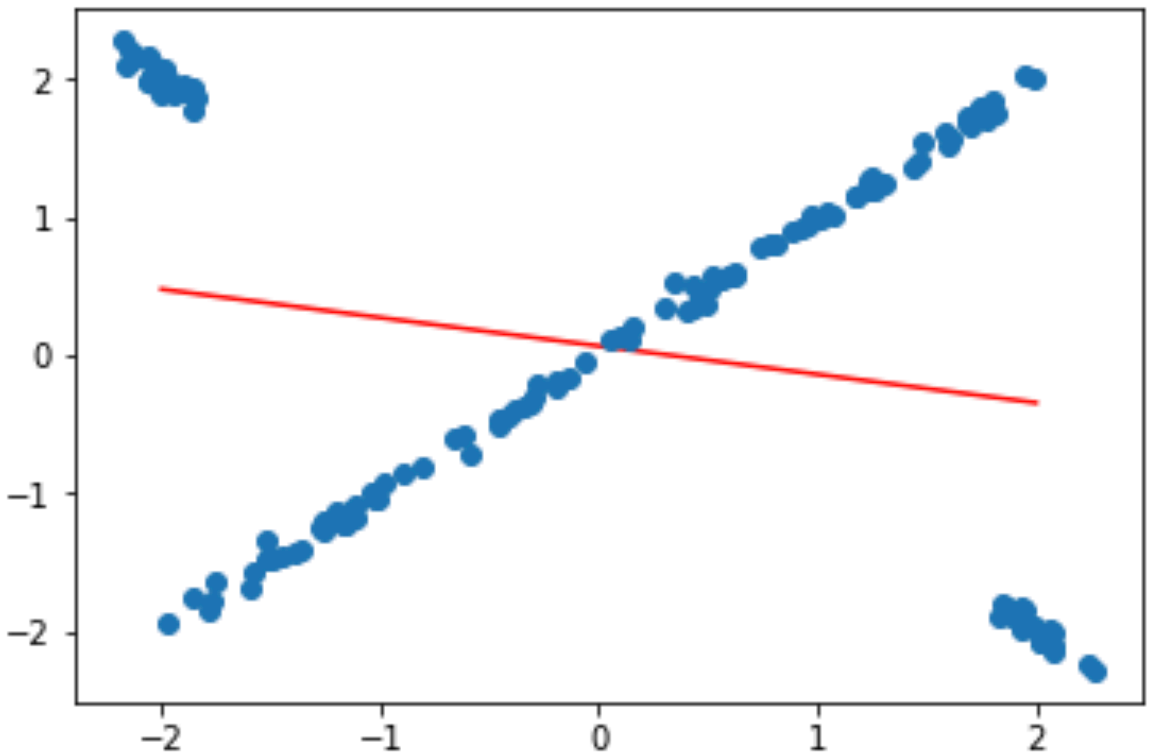
- ***a*** is a parameter one can choose
- It is quadratic for small ***e*** and linear for large ***e***



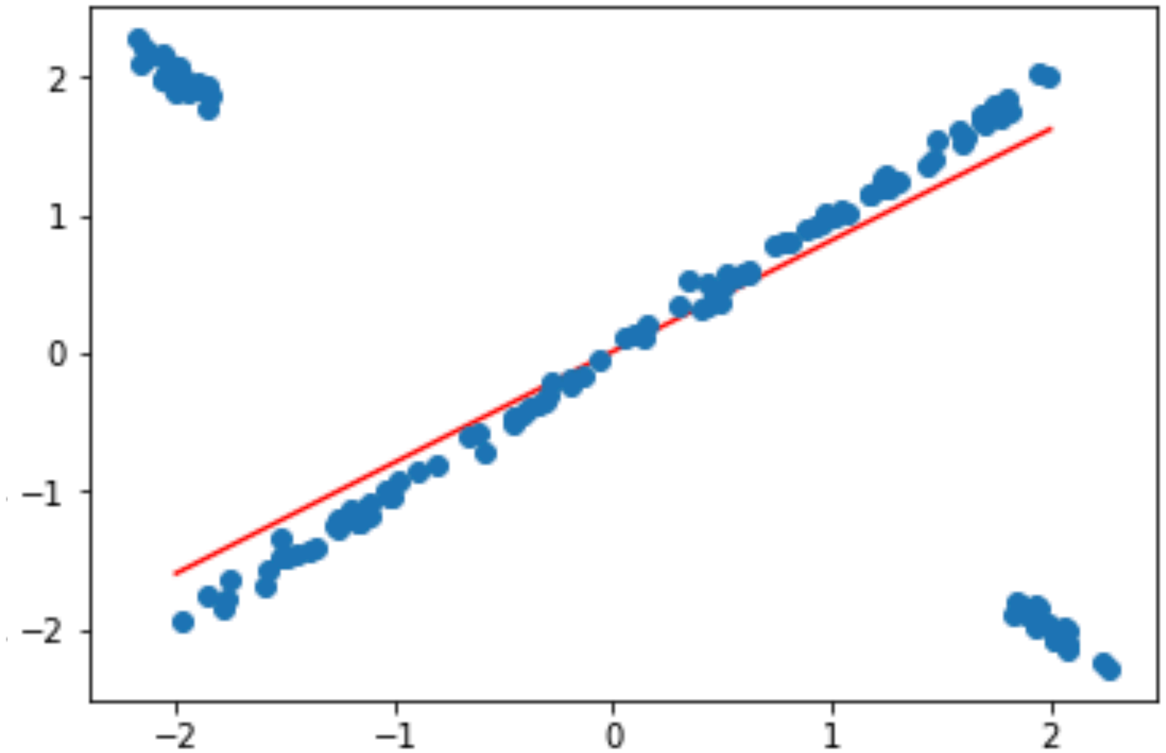
Huber loss

- Linear growth for large r makes it less sensitive to outliers

- Quadratic



- Huber

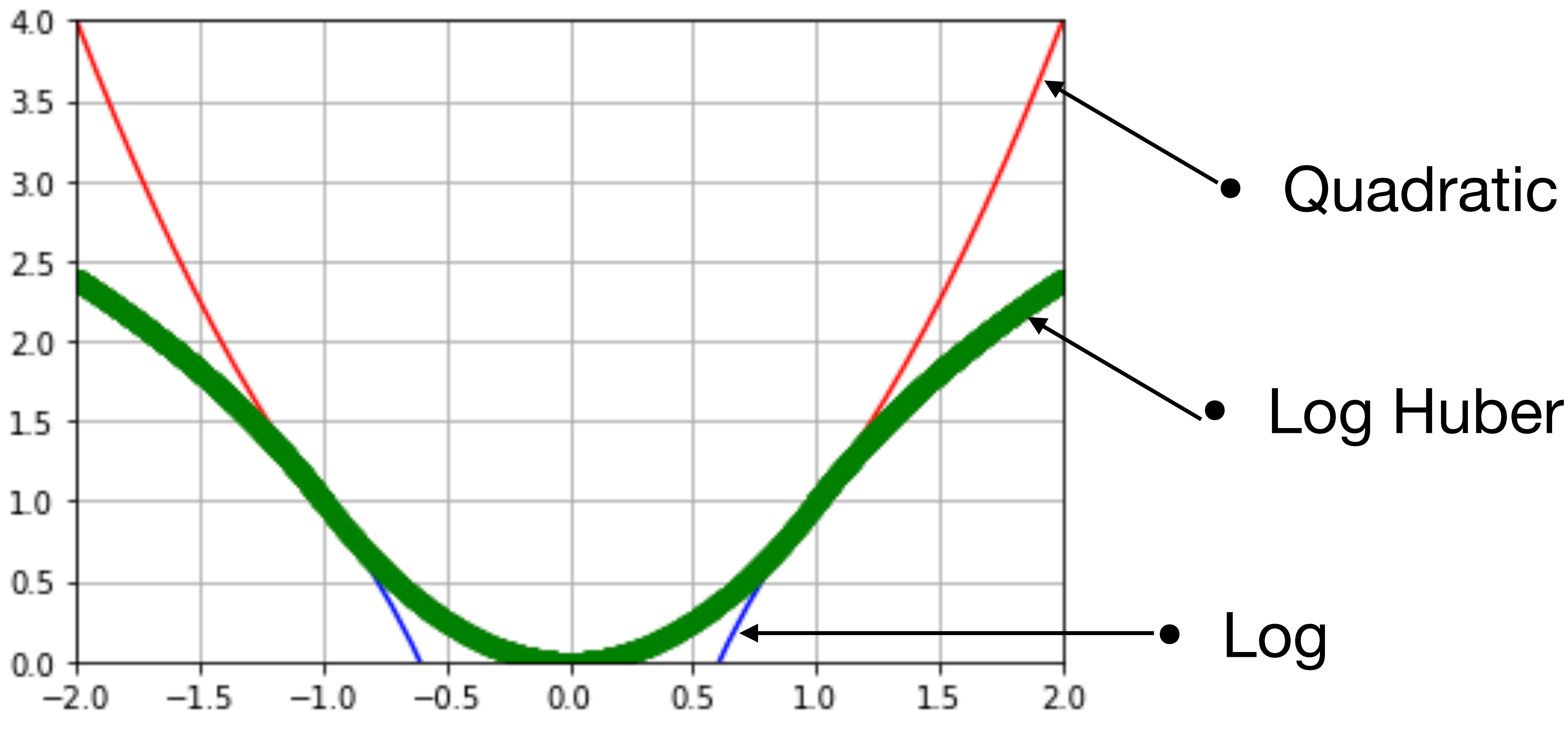


Log Huber

- Quadratic for small r, logarithmic for large r

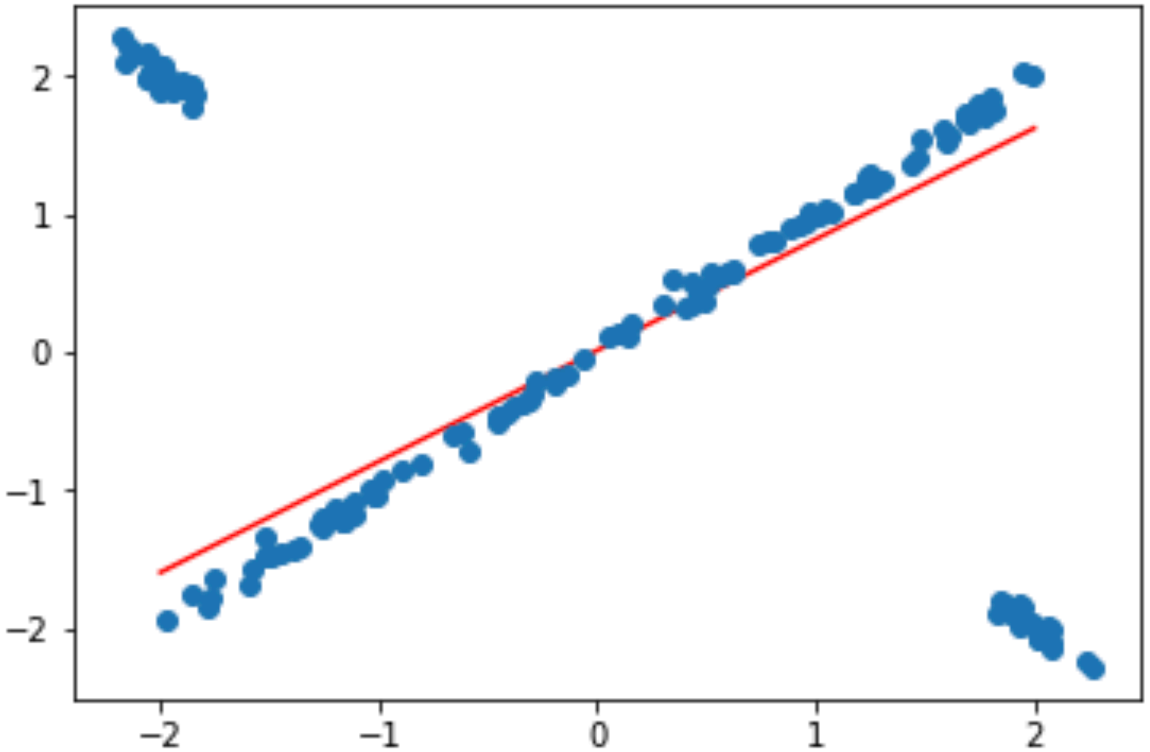
$$p(e) = \begin{cases} e^2 & \text{if } |e| \leq a \\ a^2(1 - 2\log(a) + \log(e^2)) & \text{if } |e| > a \end{cases}$$

- Diminishing incremental penalty for large ***e***
- non-convex

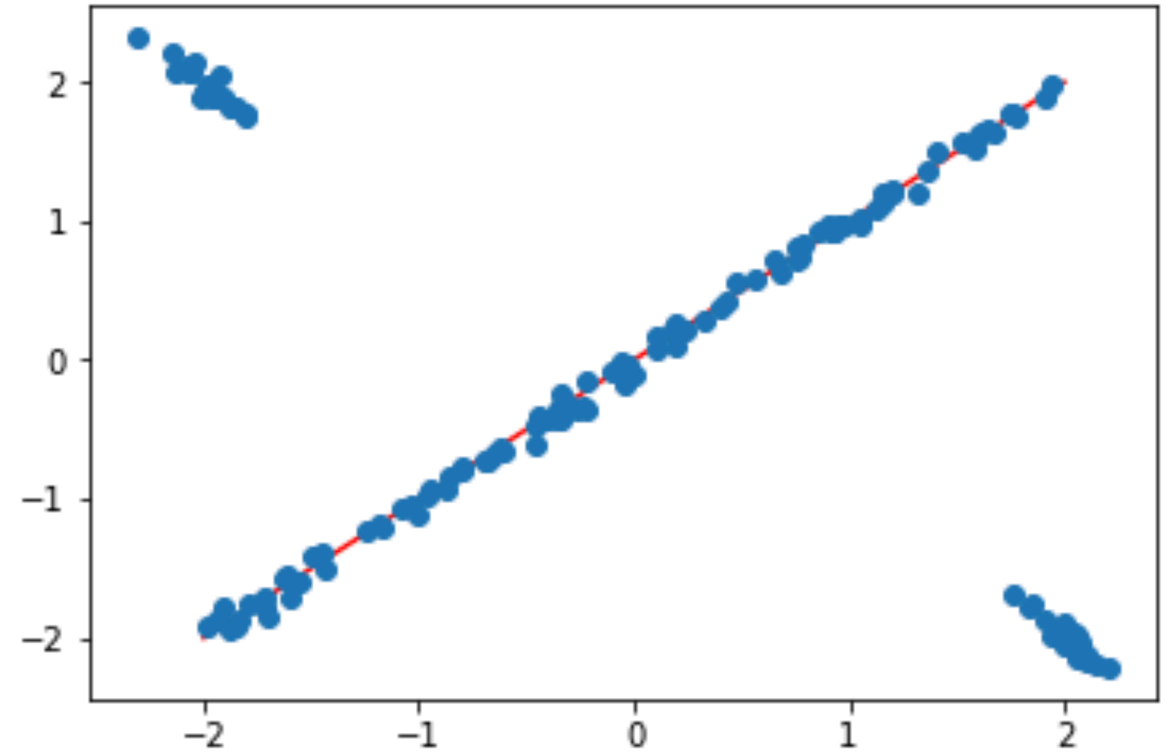


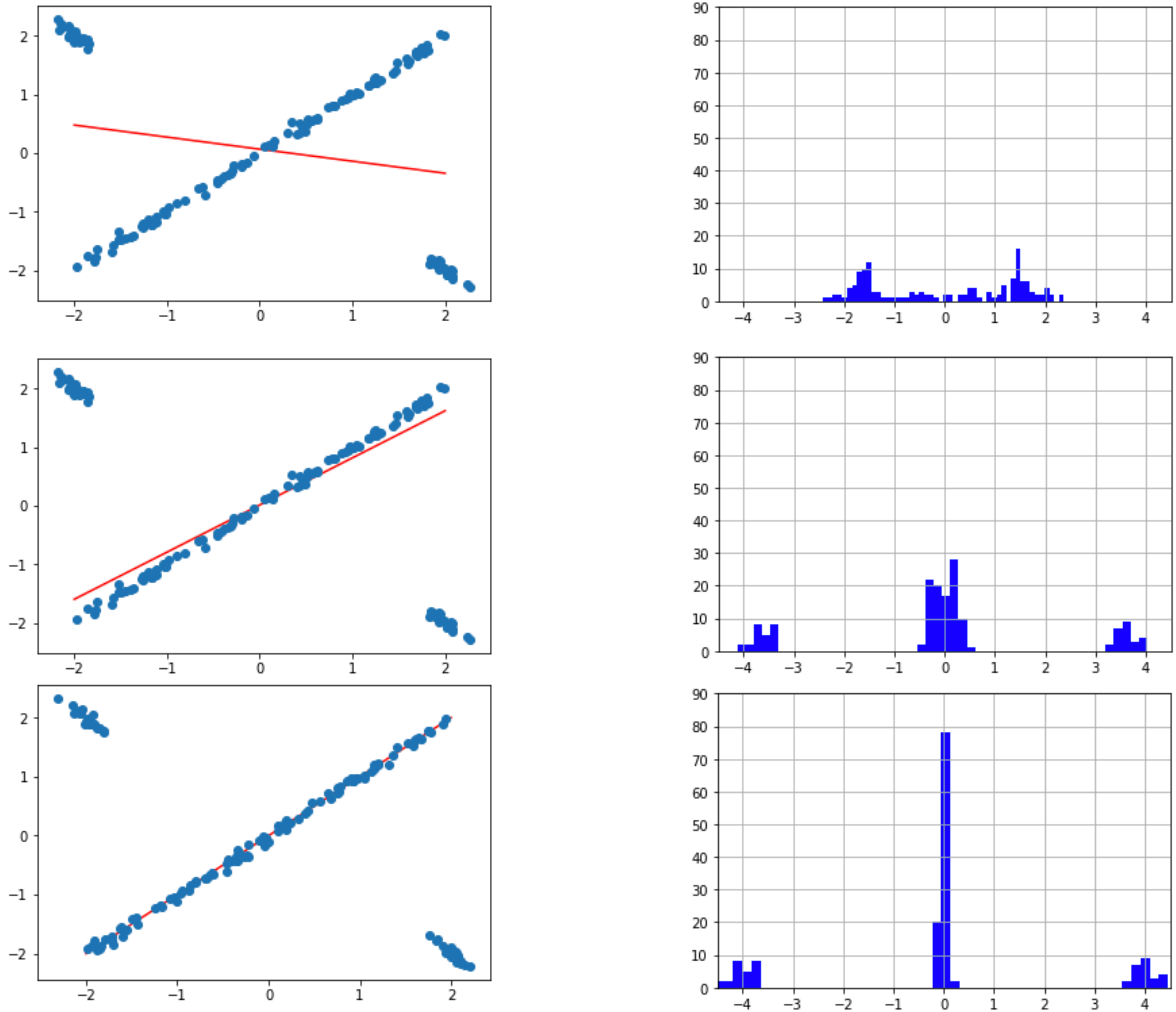
- **Log Huber** is even less sensitive to outliers than **Huber**

- Huber



- Log Huber





Quantile regression

Properties of absolute penalty

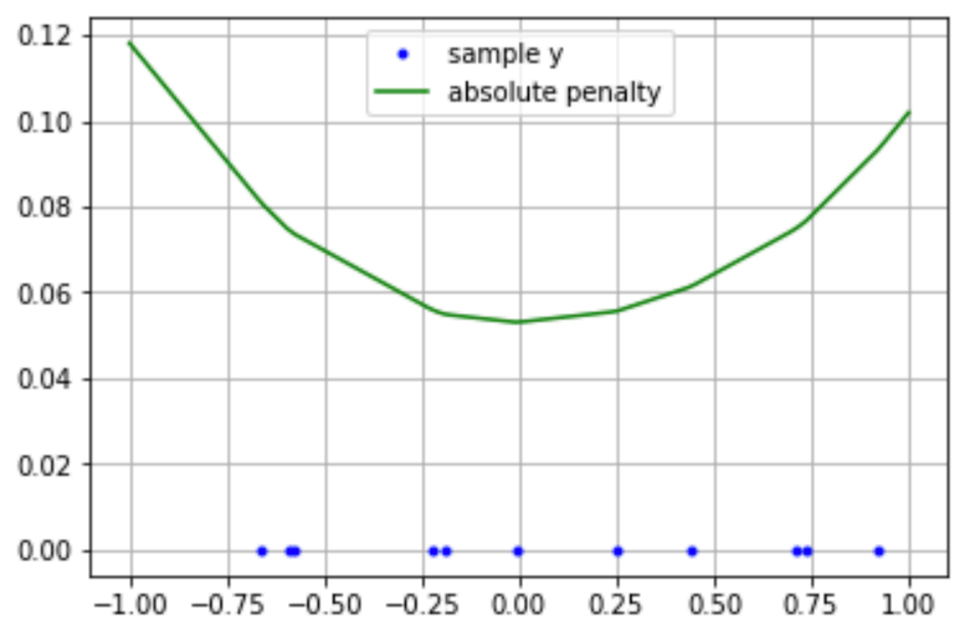
- Consider the absolute penalty p(e) = |e|
- And consider a constant predictor of the form h(x)=1
- Then the best constant predictor for absolute penalty is the solution of the following minimization

$$\text{minimize}_{w_0} \quad \frac{1}{n} \sum_{i=1}^{N} |w_0 - y_i|$$

the optimal solution of this minimization (which is the best predictor for this loss and this model) is

$$\hat{y} \; = \; w_0 \; = \; \text{median}(\{y_1, \cdots, y_N\})$$

- mathematically, this follows from

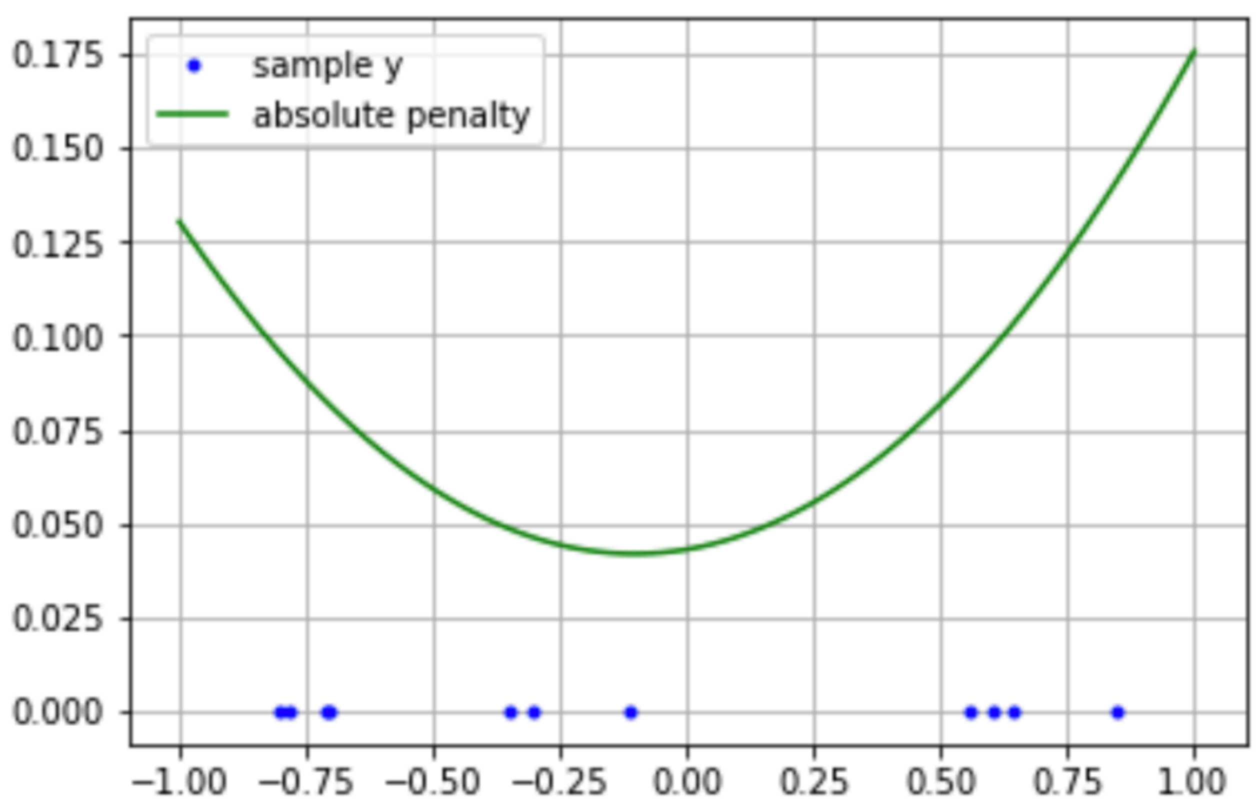


$$\frac{d}{dw_0} \sum_{i=1}^{N} |w_0 - y_i| \;\; = \;\; (\#\text{ of y's} < w_0) - (\#\text{ of y's} > w_0)$$

cf. the best constant predictor with square loss is

$$\hat{y} = w_0 = \text{mean}(\{y_1, \cdots, y_N\}) = \frac{1}{N}\sum_{i=1}^{N} y_i$$

- mathematically, this follows from



$$\frac{d}{dw_0}\sum_{i=1}^{N}(w_0 - y_i)^2 = 2\sum_{i=1}^{N}(w_0 - y_i)$$

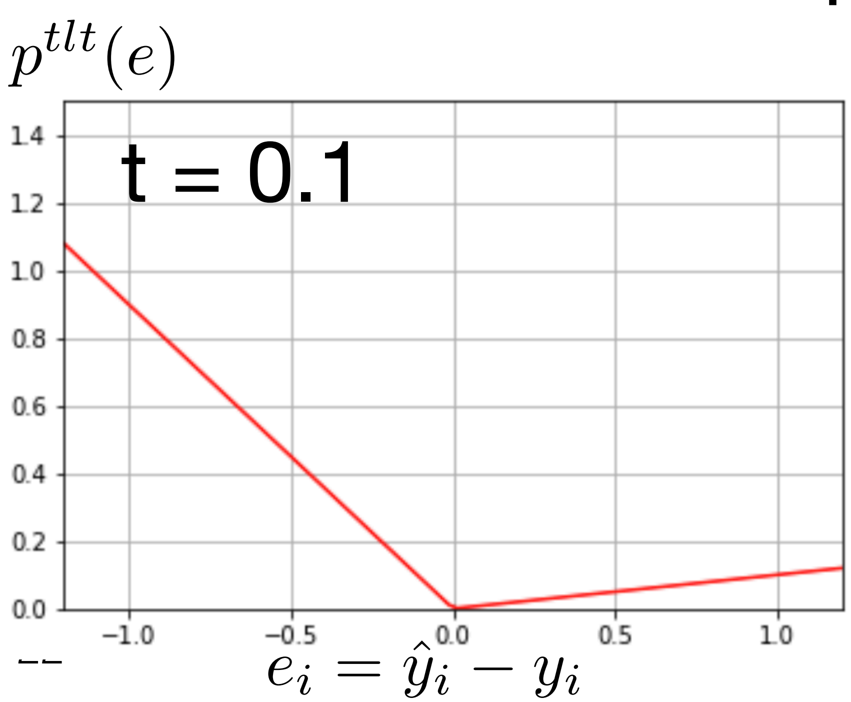
Tilted absolute penalty

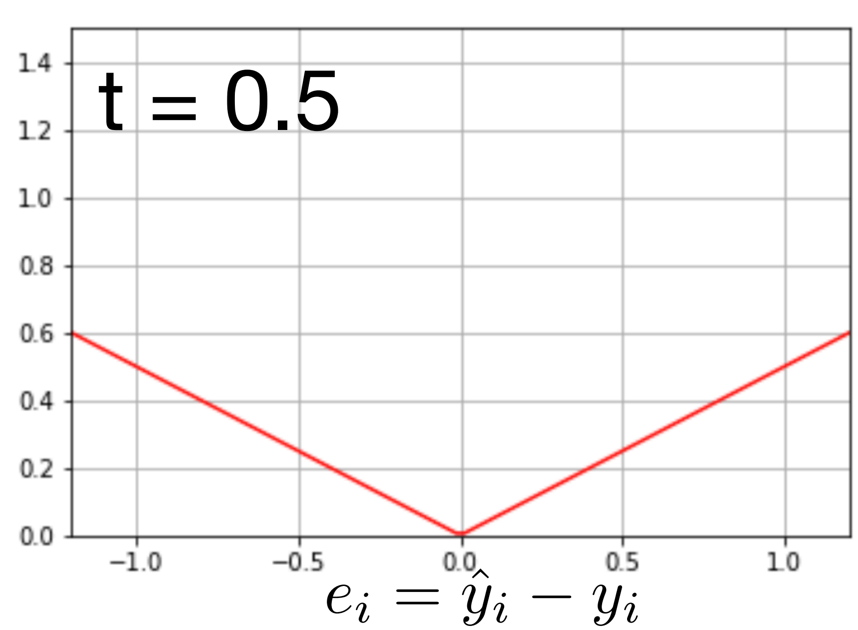
- **Tilted absolute penalty:** for some 0<t<1

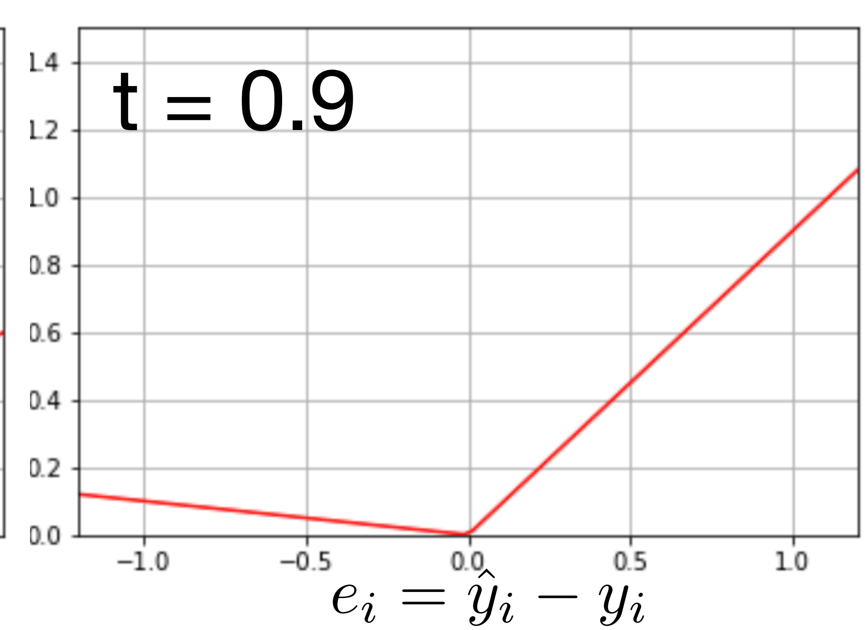
$$p^{tlt}(e) \;=\; t[e]^{+} + (1-t)[e]^{-} \;=\; (1/2)|e| + (t - 1/2)e$$

- where $[e]^{+} = \max(0,e)$ and $[e]^{-} = \max(0,-e)$

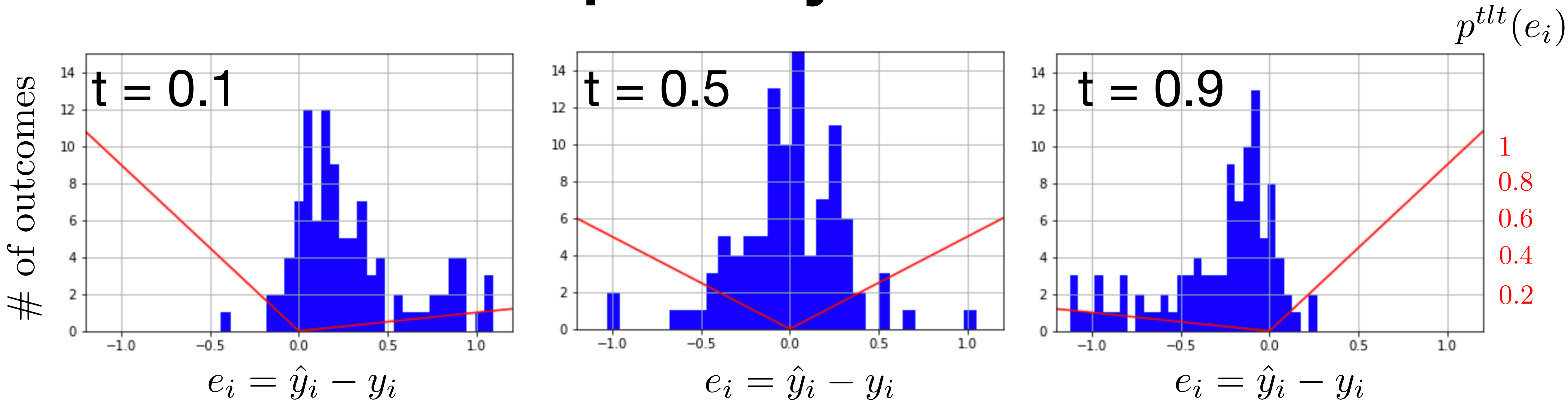
- t = 0.5: equal penalty for over- and under-estimating
- t = 0.1: 9x more penalty for under estimating
- t = 0.9: 9x more penalty for over-estimating







Tilted absolute penalty



for tilted absolute penalty with t,
the best constant predictor minimize

$$\text{minimize}_{w_0} \frac{1}{N} \sum_{i=1}^{N} p^{tlt}(w_0 - y_i)$$

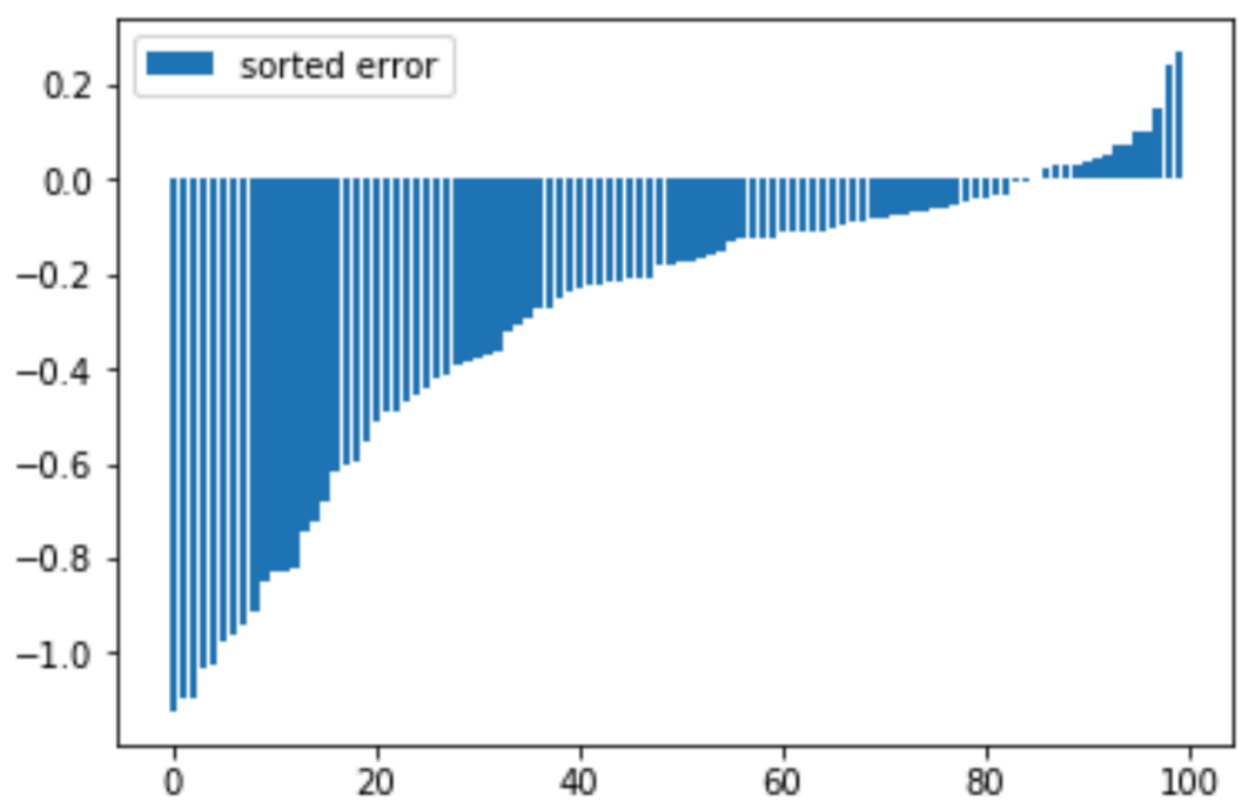
optimal when fration t of training data satisfies $w_0 < y_i$

t-quantile of training residual is zero

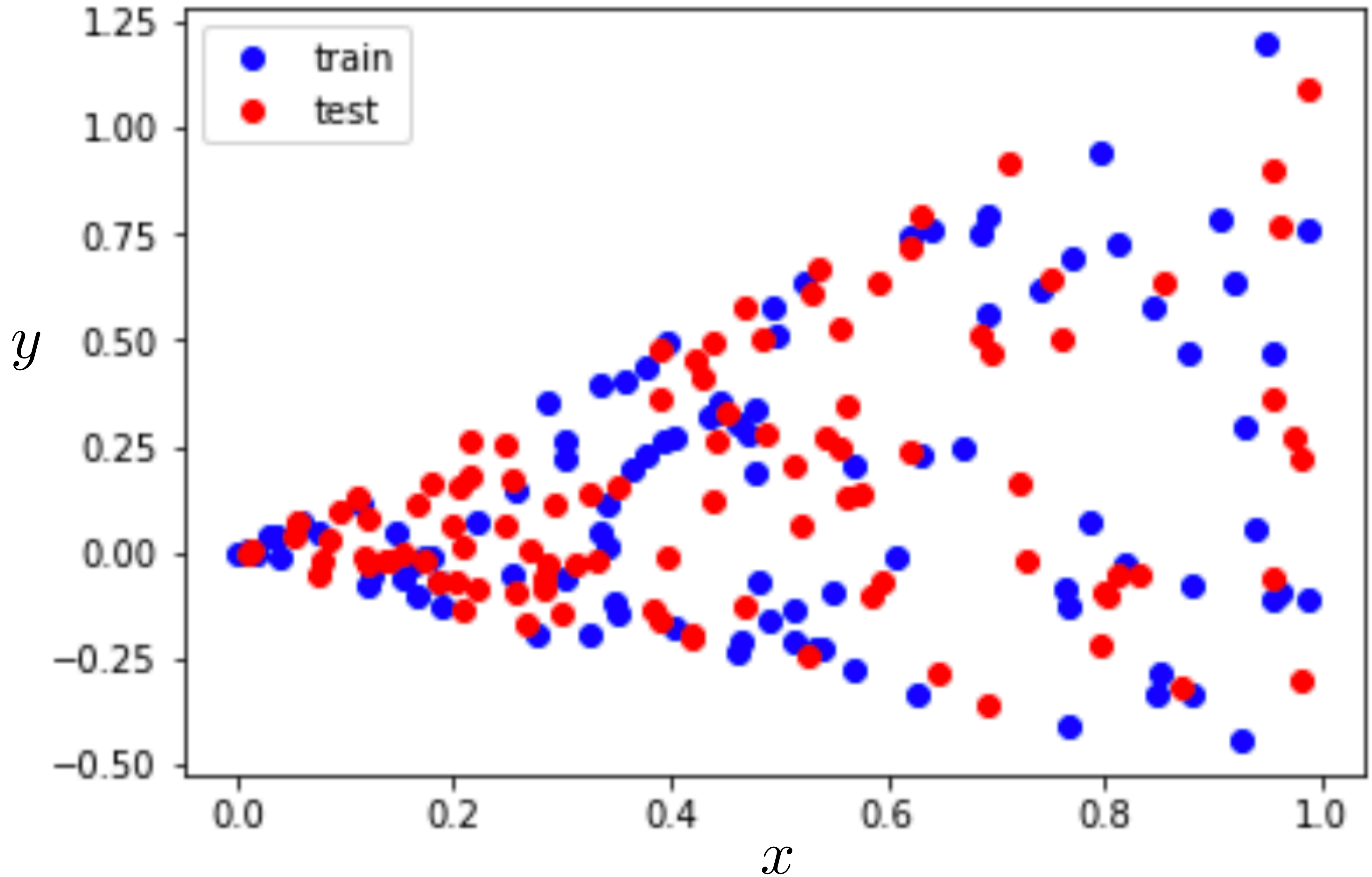
$$\hat{y} = w_0 = \text{ the } (1-t)\text{−quantile of } \{y_1, \ldots, y_N\}$$

Quantile regression

- **Quantile regression** uses the loss with $p^{tlt}(\hat{y}_i - y_i)$
- But with general regression models (not necessarily a constant model)
- In general, the resulting t-quantile of residual errors is zero
- Hence the name quantile regression
- Sorted residual error t = 0.9

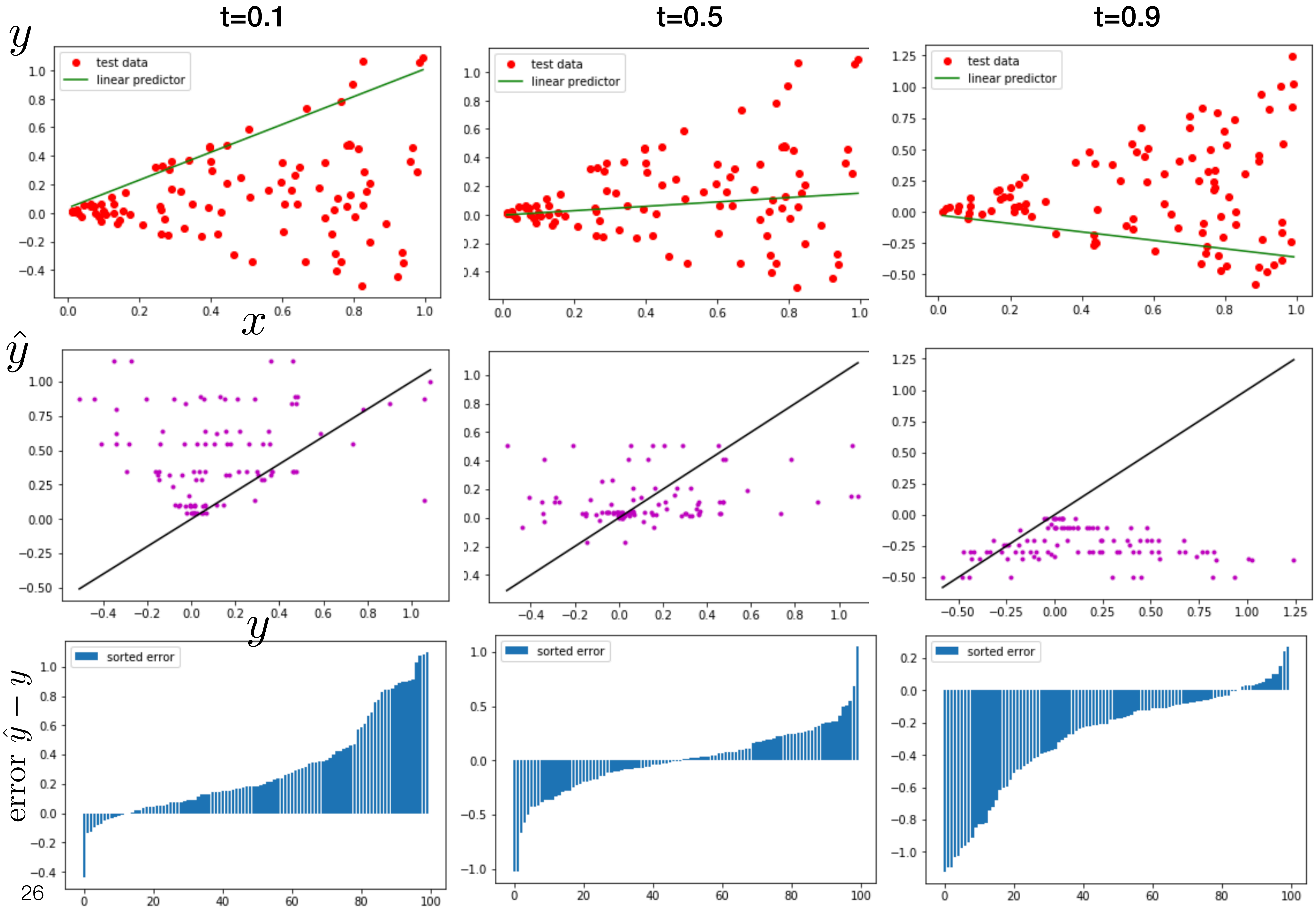


Example: quantile regression



- Fit training data with penalty function: $p^{tlt}(\hat{y}_i - y_i)$
- Consider t=0.1, 0.5, 0.9

Example: quantile regression



Example: quantile regression

- t-quantile of the error (a.k.a. residual) is zero