

# LINEAR REGRESSION

## Regression

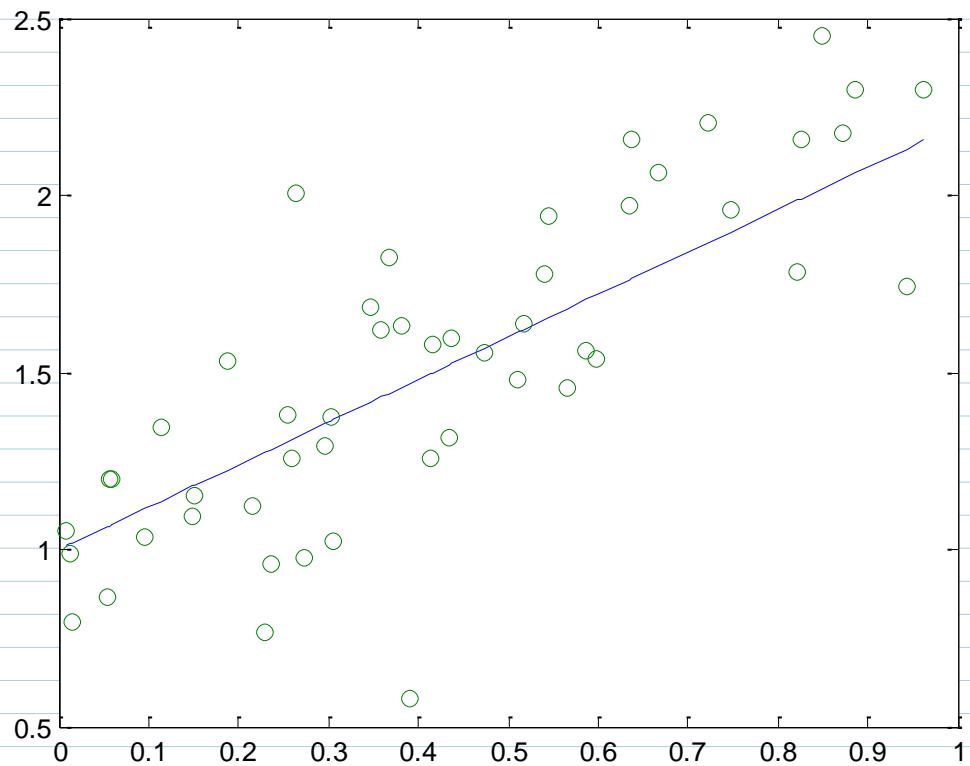
Regression is the other main supervised learning problem besides classification. We have jointly distributed variables  $(X, Y)$  where

$$X \in \mathbb{R}^d, \quad Y \in \mathbb{R}$$

and the goal is to predict  $Y$  from  $X$  using a function  $f: \mathbb{R}^d \rightarrow \mathbb{R}$ .

In practice we don't have access to the joint distribution and must estimate the optimal  $f$  using training data  $(x_1, y_1), \dots, (x_n, y_n)$ .

A regression model is a collection of candidates for  $f$ . Well begin with the case where  $f(x) = w^\top x + b$  for some  $w \in \mathbb{R}^d, b \in \mathbb{R}$ .



## Least Squares

A common performance measure is the mean squared error

$$R(f) = \mathbb{E}_{x,y} [(y - f(x))^2]$$

or in the case of linear regression, we can just write

$$R(w,b) = \mathbb{E}_{x,y} [(y - w^T x - b)^2].$$

Although the joint distribution of  $(X,Y)$  is unknown, we can estimate it via

$$R(w, b) = \frac{1}{n} \sum_{i=1}^n (y_i - w^T x_i - b)^2$$

Adding a regularization term for greater generality, we will obtain a regression estimate by solving

$$\min_{w, b} \frac{1}{n} \sum (y_i - w^T x_i - b)^2 + \lambda \|w\|^2$$

when  $\lambda = 0$  the method is called least squares regression or ordinary least squares, and for  $\lambda > 0$  it is called ridge regression.

Let's determine the solution. First we can eliminate  $b$ :

$$\frac{\partial}{\partial b} = -\frac{2}{n} \sum_{i=1}^n (y_i - w^T x_i - b) = 0$$



$$b = \frac{1}{n} \sum (y_i - w^T x_i)$$

$$= \bar{y} - w^T \bar{x}$$

where  $\bar{y} = \frac{1}{n} \sum y_i$ ,  $\bar{x} = \frac{1}{n} \sum x_i$ . Plugging this in,

the objective function becomes

$$\frac{1}{n} \sum (y_i - \bar{y} - w^T(x_i - \bar{x}))^2 + \lambda \|w\|^2$$

So let's denote  $\tilde{y}_i = y_i - \bar{y}$ ,  $\tilde{x}_i = x_i - \bar{x}$ .

Next, observe

$$\sum (\tilde{y}_i - w^T \tilde{x}_i)^2 = \|\tilde{y} - Xw\|^2$$

where

$$\tilde{y} = \begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_n \end{bmatrix} \quad \tilde{X} = \begin{bmatrix} \tilde{x}_1^{(1)} & \dots & \tilde{x}_1^{(d)} \\ \vdots & \ddots & \vdots \\ \tilde{x}_n^{(1)} & \dots & \tilde{x}_n^{(d)} \end{bmatrix}$$

Therefore the objective function is

$$\frac{1}{n} \|\tilde{y} - \tilde{X}w\|^2 + \lambda \|w\|^2$$

$$\propto (\tilde{y} - \tilde{X}w)^T (\tilde{y} - \tilde{X}w) + n\lambda w^T w$$

$$= w^T (\tilde{X}^T \tilde{X} + n\lambda I) w^T - 2\tilde{y}^T \tilde{X} w + \tilde{y}^T \tilde{y}$$

$$=: w^T B w - 2C^T w + d =: J(w)$$

Notice that  $B \succeq 0$  (positive semi-definite), and  $B \succ 0$  if  $\lambda > 0$ . Therefore  $w^*$  is a global minimizer iff

$$\nabla J(w^*) = 2Bw^* - 2c = 0$$

If  $B \succeq 0$ , then

$$w^* = B^{-1}c = \boxed{(X^T X + n\lambda I)^{-1} X^T \tilde{y}}$$

is the unique minimizer.

### Alternate Derivation for OLS

If  $\lambda=0$ , there is an alternate (but equivalent) solution.

Instead of first eliminating  $b$ , we can solve for

$$\theta = \begin{bmatrix} b \\ w \end{bmatrix} \text{ at once.}$$

Just observe  $\sum (y_i - w^T x_i - b)^2 = \|y - X\theta\|^2$

where now

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_1^{(d)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^{(1)} & \dots & x_n^{(d)} \end{bmatrix}$$

Similar to above, we find

$$\hat{\theta} = (X^T X)^{-1} X^T y.$$

## Large Scale Ridge Regression

Note that  $X^T X$  is  $d \times d$ . Suppose  $d$  is so large that inverting a  $d \times d$  matrix is computationally prohibitive. An alternative is to minimize

$$J(w) = w^T (X^T X + \eta I) w - 2\hat{y}^T X w + \hat{y}^T \hat{y}$$

iteratively using gradient descent:

- Initialize  $w_0$

- $t \leftarrow 0$

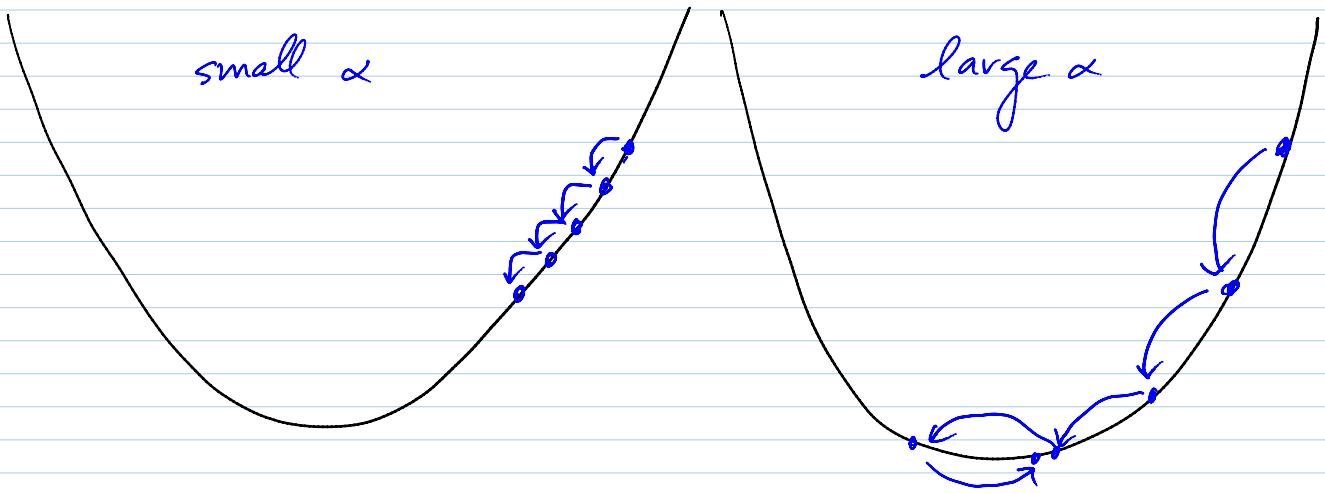
- Repeat

$$w_{t+1} \leftarrow w_t - \alpha \nabla J(w_t)$$

$$t \leftarrow t + 1$$

until convergence condition satisfied

$\alpha_t$  is called the learning rate or step-size parameter.



For convergence, need  $\alpha = \alpha_t \downarrow 0$ , otherwise  $w_t$  keeps oscillating around minimizer.

The gradient is

$$\begin{aligned}\nabla J(w) &= 2(X^T X + n\lambda I)w - 2X^T \tilde{y} \\ &= 2X^T(Xw - \tilde{y}) + 2n\lambda Iw \\ &= 2 \sum_{i=1}^n [\tilde{x}_i(w^T \tilde{x}_i - \tilde{y}_i) + \lambda w]\end{aligned}$$

Computational complexity:  $O(nd)$  per iteration

Stochastic gradient descent is the following variation on gradient descent:

- Initialize  $w^0$

- $t \leftarrow 0$
- While 1

For  $i = 1, \dots, n$  in random order

$$w_{t+1} \leftarrow w_t - 2\alpha_t [\hat{x}_i (w_t^\top \hat{x}_i - \hat{y}_i) + \lambda w_t]$$

If convergence condition satisfied, exit

$$t \leftarrow t + 1$$

End

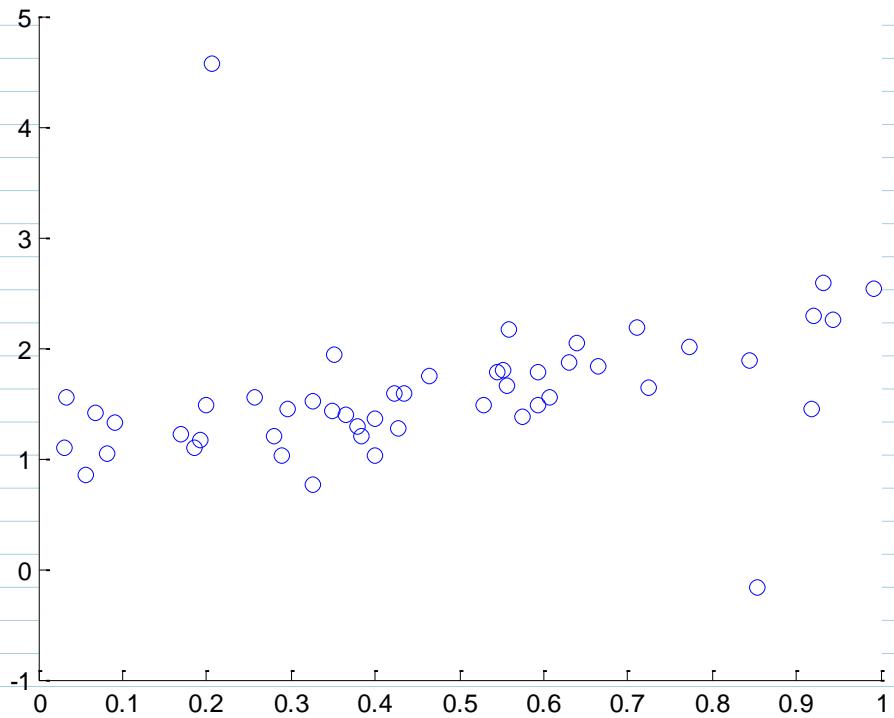
End

SGD can converge much faster than GD, and is particularly useful when the full gradient is expensive to compute/store.

In practice, an extension of gradient descent called conjugate gradient descent is often used.

## Robust Regression

Least squares is nice, but it also has some disadvantages. Consider the following data:

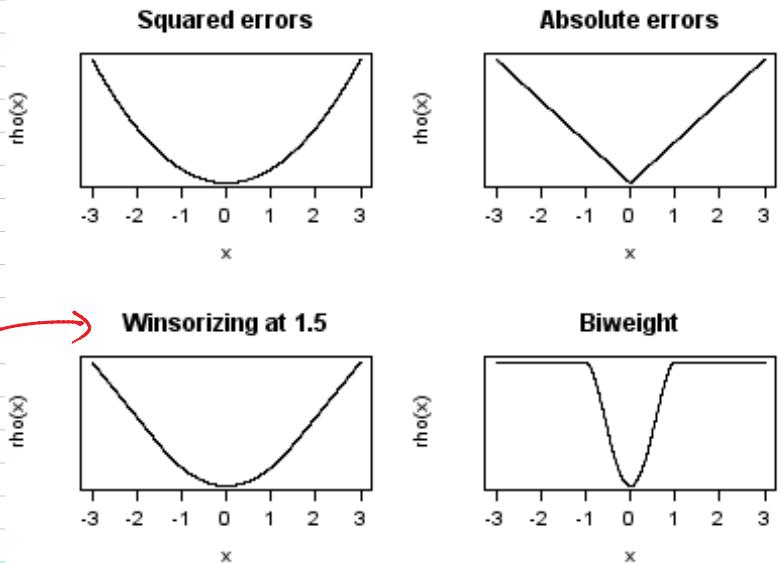


Least squares is not robust to outliers because errors get squared, which means the LS solution is very sensitive to outliers.

An alternative is to solve

$$\min_{w, b} \frac{1}{n} \sum_{i=1}^n p(y_i - w^T x_i - b)$$

where  $p$  is a robust loss function.



aka Huber loss → Winsorizing at 1.5

Unlike least squares, there is no closed form solution. To minimize

$$J(w, b) = \frac{1}{n} \sum \rho(y_i - w^T x_i - b)$$

numerically we will employ the majorize/minimize (MM) technique. Here's the idea:

- Initialize  $w_0, b_0$

- $t \leftarrow 0$

- Repeat

- Find a function  $\bar{J}_t(w, b)$  such that

$$J(w_t, b_t) = \bar{J}_t(w_t, b_t)$$

$$J(w, b) \leq \bar{J}_t(w, b) \quad \forall w, b$$

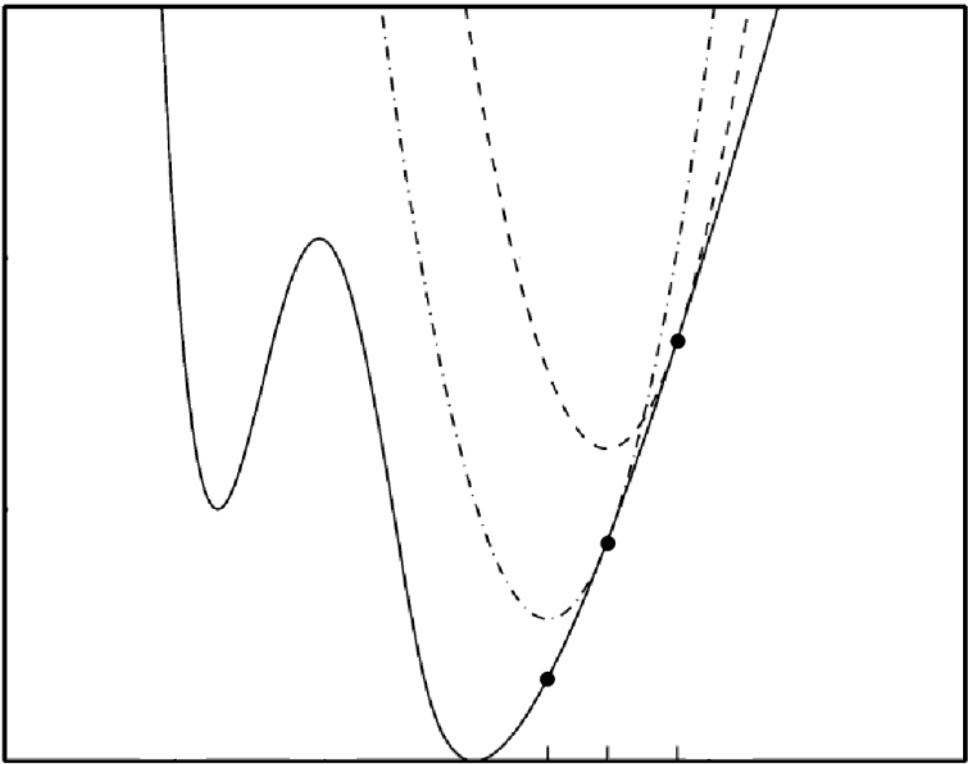
} majorize

minimize

$$\bullet (w_{t+1}, b_{t+1}) \leftarrow \arg \min_{w, b} \bar{J}_t(w, b)$$

$$\bullet t \leftarrow t + 1$$

Until converged



$$\theta_{t+2} \theta_{t+1} \theta_t$$

The idea is to choose  $\bar{J}_t$  such that minimization can be carried out efficiently. We will select

$\bar{J}_t$  to have the form

$$\bar{J}_t(w, b) = \sum_{i=1}^n \bar{p}_t(y_i - w^T x_i - b)$$

$$1 \quad 1 \quad 0$$

where  $\bar{p}_t$  is a majorizing function for  $p$ .

Let us introduce the notation

$$\psi(r) := p'(r)$$

$$q(r) := \frac{\psi(r)}{r}, \quad r \neq 0$$

and

$$r_{t,i} = y_i - w_t^T x_i - b_t.$$

The following result provides a majorizing function for a broad class of  $p$ .

Lemma Assume  $p(r)$  is symmetric and differentiable, nondecreasing for  $r > 0$ ,  $\frac{\psi(r)}{r}$  is nonincreasing for  $r > 0$ ,  $q(0) := \lim_{r \rightarrow 0} q(r)$  exists and  $q$  is continuous.

Define

$$\tilde{J}_t(w, b) = \sum_{i=1}^n \bar{p}_t(y_i - w^T x_i - b)$$

where

$$\bar{p}_t(r) = p(r_{t,i}) - \frac{1}{2} r_{t,i} \psi(r_{t,i}) + \frac{1}{2} \cdot \frac{\psi(r_{t,i})}{r_{t,i}} \cdot r^2$$

. = -

Then  $J_t$  majorizes  $J$ .

Proof] The proof is left as a supplemental homework problem.

The conditions of the lemma are satisfied by Huber's  $\rho$  and the biweight function, as well as several others.

With this majorizing function, the iterative update has the form

$$\min_{w, b} \sum_{i=1}^n c_{t,i} (y_i - w^T x_i - b)^2$$

where

$$c_{t,i} = \frac{\psi(r_{t,i})}{r_{t,i}}.$$

The algorithm is known as iteratively reweighted least squares. Weighted least squares can

be solved by a slight modification of ordinary LS.

This adaptation is left as an exercise.