LINEAR REGRESSION

Regression

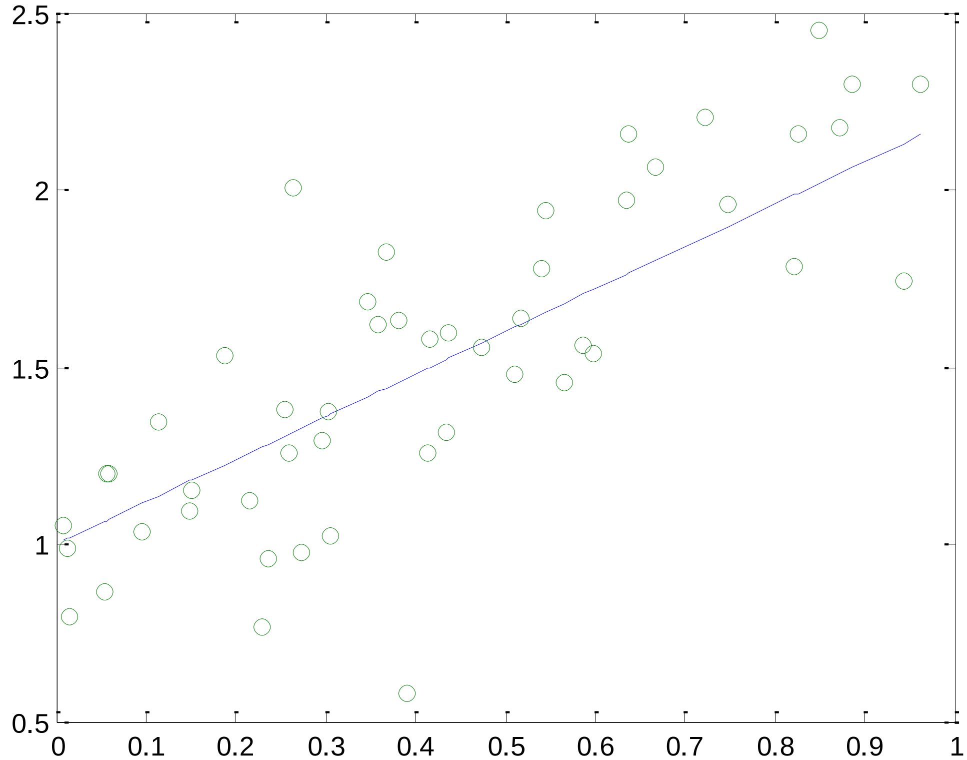
Regression is the other main supervised learning problem besides classification. We have jointly distributed variables (X, Y) where

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

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

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

A regression model is a collection of candidates for f. We'll begin with the case where $f(x) = w^T x + b$ for some $w \in \mathbb{R}$, $b \in \mathbb{R}$.



Least Squares

A common performance measure is the mean squared error

$$R(f) = \mathbb{E}_{X,Y}\left[(Y - f(X))^2\right]$$

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

$$R(w,b) = \mathbb{E}_{X,Y}\left[(Y - w^T X - b)^2\right].$$

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

$\hat{}$

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

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

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

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:

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

$$\Updownarrow$$

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

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

where $\bar{y} = \frac{1}{n}\sum x_i$, $\bar{x} = \frac{1}{n}\sum x_i$. Plugging this in, the objective function becomes

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

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

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

Therefore the objective function is

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

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

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

$$=:\frac{1}{2}w^TAw+b^Tw+c=:J(w)$$

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

$$\nabla J(w^*) = A w^* + b = 0$$

If $A \succ 0$, then

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

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

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

where now

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

$$\begin{bmatrix} y_n \end{bmatrix} \qquad \begin{bmatrix} 1 & x_n^{(1)} & \cdots & x_n^{(d)} \end{bmatrix}$$

Similar to above, we find

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

$\boxed{\text{Large Scale Ridge Regression}}$

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

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

iteratively using gradient descent:

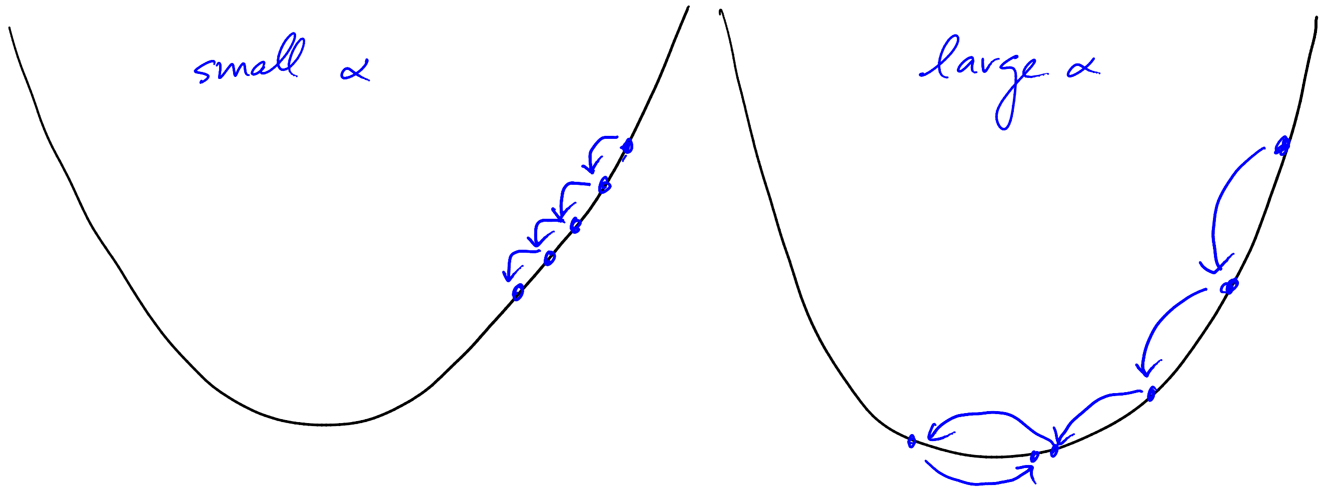
- Initialize w_0
- For $t = 1, \dots,$ max_iter

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

  If convergence condition satisfied, exit.

  End

α is called the learning rate or step-size parameter



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

The gradient is

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

Computational complexity: $O(nd)$ per iteration

Stochastic gradient descent is the following variation on gradient descent:

- Initialize w^0

called for by theory

- Initialize w^0
- $t \leftarrow 0$
- For $j = 1, \ldots,$ max_iter

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

  called for by theory, and also avoids cycles

  $$w_{t+1} \leftarrow w_t - 2\alpha_j \left[\hat{x}_i (w_t^T \hat{x}_i - \tilde{y}_i) + \lambda w_t \right]$$

  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:

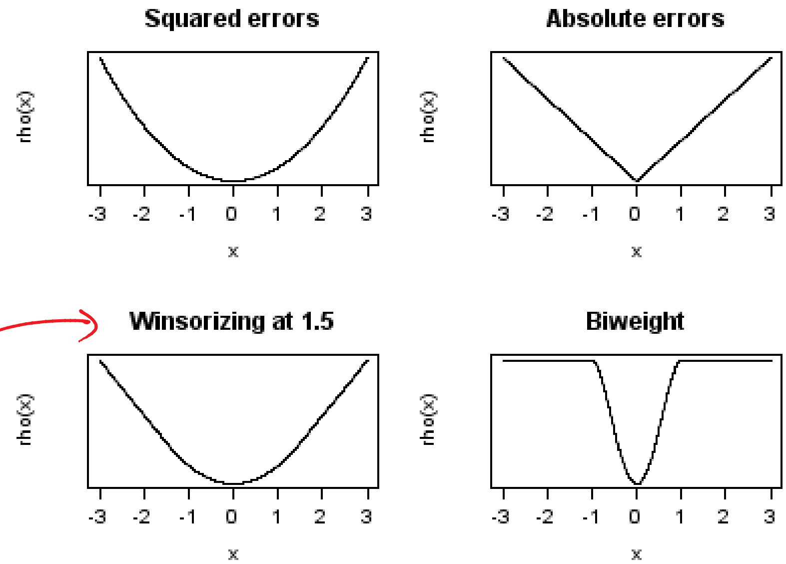
Least squares is not robust to outliers because errors get squared, which means the LS solution is very sensitive to outliers.

An alternative is to solve

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

where ρ is a robust loss function.

aka Huber loss



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 $J_t(w,b)$ such that

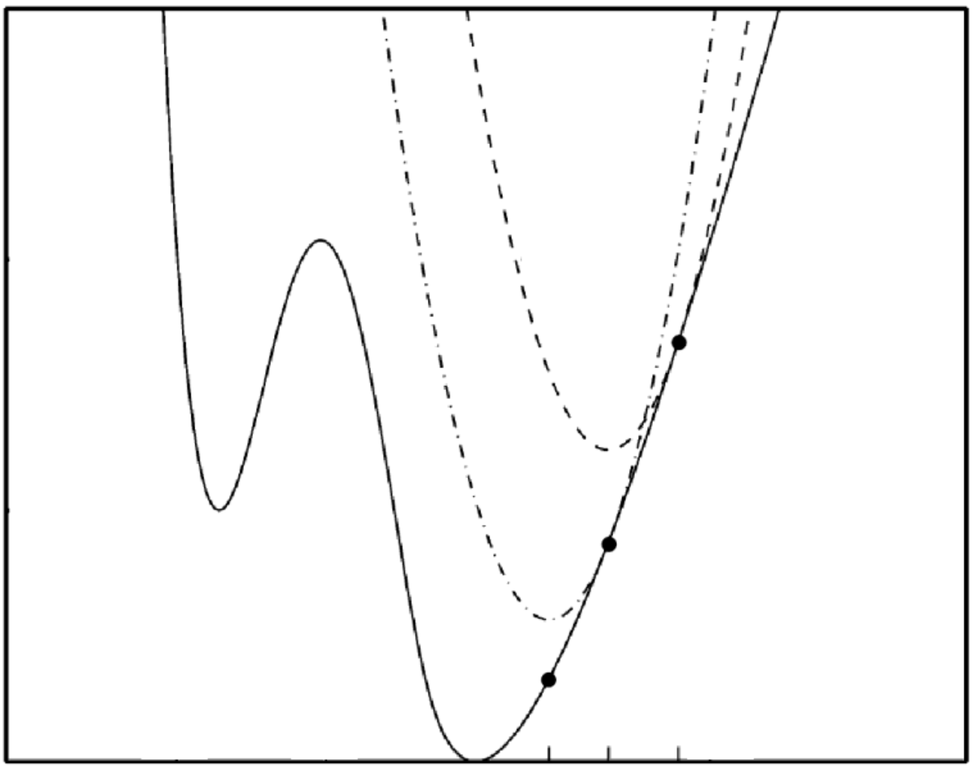
$$\left.\begin{aligned} J(w_t, b_t) &= J_t(w_t, b_t) \\ J(w,b) &\le J_t(w,b) \quad \forall w,b \end{aligned}\right\} \text{majorize}$$

- $(w_{t+1}, b_{t+1}) \leftarrow \arg\min_{w,b} J_t(w,b)$

minimize

- $t \leftarrow t+1$

Until converged



θ_{t+2} θ_{t+1} θ_t

The idea is to choose J_t such that minimization can be carried out efficiently. We will select J_t to have the form

$$J_t(w,b) = \sum_{i=1}^{n} \rho_t\left(y_i - w^T x_i - b\right)$$

where ρ_t is a majorizing function for ρ.

Let us introduce the notation

$$\psi(r) := \rho'(r)$$
$$\varphi(r) := \frac{\psi(r)}{r}, \quad r \neq 0$$

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 ρ.

$\boxed{\text{Lemma}}$ Assume $\rho(r)$ is symmetric and differentiable, nondecreasing for $r > 0$, $\frac{\psi(r)}{r}$ is nonincreasing for $r > 0$, $\varphi(0) := \lim_{r \to 0} \varphi(r)$ exists and φ is continuous.

Define

$$J_t(w,b) = \sum_{i=1}^{n} \rho_t(y_i - w^T x_i - b)$$

where

$$\rho_t(r) = \rho(r_{t,i}) - \frac{1}{2} r_{t,i} \psi(r_{t,i}) + \frac{1}{2} \cdot \frac{\psi(r_{t,i})}{r_{t,i}} \cdot r^2.$$

Then J_t majorizes J.

<u>Proof</u> The proof is left as a supplemental homework problem.

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

With this majorizing function, the iterative update has the form

$$(w_{t+1}, b_{t+1}) = \arg\min_{w,b} \sum_{i=1}^{n} c_{t,i} \left(y_i - w^T x_i - b \right)^2$$

where

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

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.

Follow this link to learn more about different choices for ρ.

Survey of robust losses and their properties