MAC 0460 / 5832
Introduction to Machine Learning

04 – Linear regression

• MSE cost/loss function • analytical solution • gradient descent •

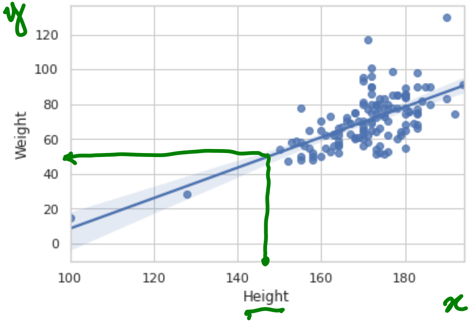
IME/USP

Recall: **The linear regression problem**

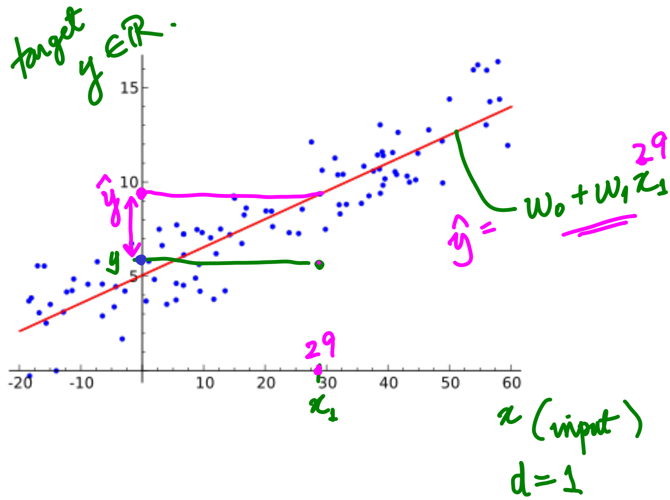
Suppose I need to record the height and weight of a person.

However, my

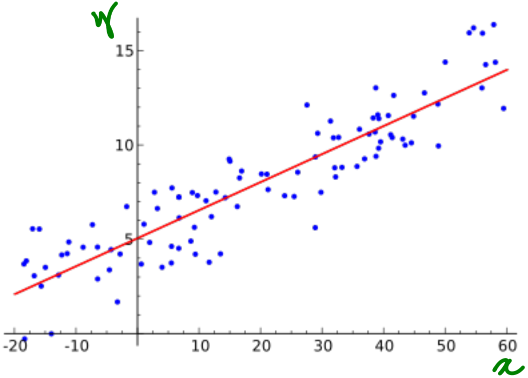
is broken !



We have **training data**, $(x^{(n)}, y^{(n)})$ índice

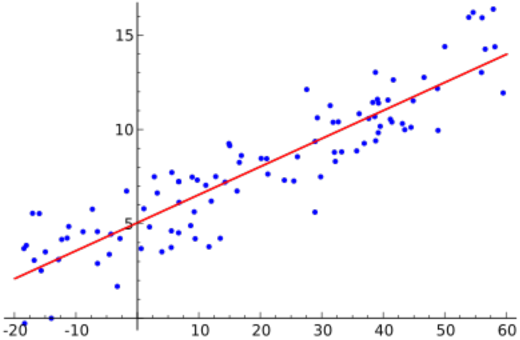


There is a **linear relation** between x and y

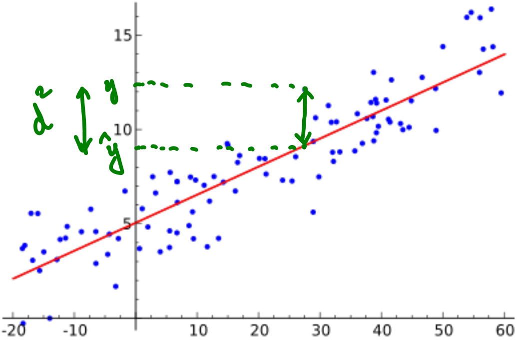


$$\text{parameters}$$

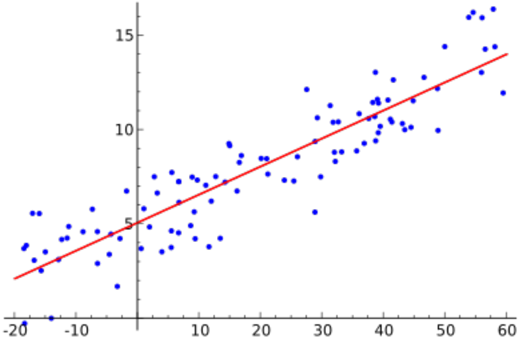
Hypothesis space: $\underbrace{h(x) = w_0 + w_1\, x}$



Cost function: $J(w_0, w_1) = \frac{1}{N}\sum_{n=1}^{N}\Big(\underbrace{\hat{y}^{(n)}}_{h(x^{(n)})=w_0+w_1\,x^{(n)}} - y^{(n)}\Big)^2$



Goal: Find w_0 and w_1 ($h(x) = w_0 + w_1 x$) that minimizes $J(w_0, w_1)$

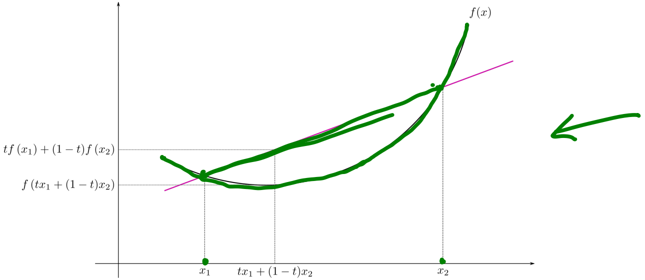


The cost function is quadratic, so it is convex

$$J(w_0, w_1) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{y}^{(n)} - y^{(n)}\right)^2$$

$w_0 + w_1 x^{(n)}$

Convex means:



Source: wikipedia

Solution: $d = 1$

Partial derivatives:

$$\frac{\partial J(w_0, w_1)}{\partial w_0} = 2\sum_{n=1}^{N}\left(w_0 + w_1 x^{(n)} - y^{(n)}\right) = 0$$

$$\frac{\partial J(w_0, w_1)}{\partial w_1} = 2\sum_{n=1}^{N}\left(w_0 + w_1 x^{(n)} - y^{(n)}\right) x^{(n)} = 0$$

Minimum point of $J(w_0, w_1)$:

$$w_0 = \overline{y} - w_1 \overline{x}$$

$$w_1 = \frac{\sum_{n=1}^{N}(x^{(n)} - \overline{x})(y^{(n)} - \overline{y})}{\sum_{n=1}^{N}(x^{(n)} - \overline{x})^2}$$

Solution: $\boxed{d = 1}$

$$\boxed{J(w_0, w_1) = \sum_{n=1}^{N} \left(w_0 + w_1 x^{(n)} - y^{(n)} \right)^2}$$

Partial derivatives:

$$\frac{\partial J(w_0, w_1)}{\partial w_0} = 2 \sum_{n=1}^{N} (w_0 + w_1 x^{(n)} - y^{(n)}) = 0$$

$$\frac{\partial J(w_0, w_1)}{\partial w_1} = 2 \sum_{n=1}^{N} (w_0 + w_1 x^{(n)} - y^{(n)})\, x^{(n)} = 0$$

Minimum point of $J(w_0, w_1)$:

$$\begin{cases} w_0 = \overline{y} - w_1 \overline{x} \\ w_1 = \dfrac{\sum_{n=1}^{N} (x^{(n)} - \overline{x})(y^{(n)} - \overline{y})}{\sum_{n=1}^{N} (x^{(n)} - \overline{x})^2} \end{cases}$$

$$x = \begin{bmatrix} 1 \\ x_1 \\ \\ x_d \end{bmatrix}$$

Notations: d-dimensional case ($d > 1$)

$$\mathbf{x}^{(n)} = (1, x_1, x_2, \ldots, x_d) \in \{1\} \times \mathbb{R}^d \longrightarrow \texttt{array (d+1,1)}$$

$$\mathbf{w} = (w_0, w_1, w_1, \ldots, w_d) \in \mathbb{R}^{d+1} \longrightarrow \texttt{array (d+1,1)}$$

$$h_{\mathbf{w}}(\mathbf{x}^{(n)}) = \sum_{i=0}^{d} w_i\, x_i = \begin{bmatrix} w_0 & w_1 & \ldots & w_d \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \ldots \\ x_d \end{bmatrix} = \mathbf{w}^T\, \mathbf{x}^{(n)}$$

$$\boxed{J(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \left(h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)} \right)^2}$$

In sample error

The expression for E_{in}

$h_w(x^{(n)})$

J

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n - y_n)^2$$

$$= \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2$$

where $\mathrm{X} = \begin{bmatrix} \text{—}\mathbf{x}_1^{\mathsf{T}}\text{—} \\ \text{—}\mathbf{x}_2^{\mathsf{T}}\text{—} \\ \vdots \\ \text{—}\mathbf{x}_N^{\mathsf{T}}\text{—} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$

Solution based on matrix algebra

Cost function: $J(\mathbf{w}) = \dfrac{1}{N}\sum_{n=1}^{N}\Big(\underbrace{h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)}}_{R}\Big)^2$

Residuals:

$$\begin{array}{c} 1 \\ \vdots \\ \\ N \end{array}\begin{bmatrix} h_{\mathbf{w}}(\mathbf{x}^{(1)}) - y^{(1)} \\ h_{\mathbf{w}}(\mathbf{x}^{(2)}) - y^{(2)} \\ \vdots \\ h_{\mathbf{w}}(\mathbf{x}^{(N)}) - y^{(N)} \end{bmatrix}$$

Solution based on matrix algebra

Cost function: $J(\mathbf{w}) = \dfrac{1}{N} \displaystyle\sum_{n=1}^{N} \left(h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)} \right)^2$

Residuals:

$$\begin{bmatrix} h_{\mathbf{w}}(\mathbf{x}^{(1)}) - y^{(1)} \\ h_{\mathbf{w}}(\mathbf{x}^{(2)}) - y^{(2)} \\ \vdots \\ h_{\mathbf{w}}(\mathbf{x}^{(N)}) - y^{(N)} \end{bmatrix} = \begin{bmatrix} h_{\mathbf{w}}(\mathbf{x}^{(1)}) \\ h_{\mathbf{w}}(\mathbf{x}^{(2)}) \\ \vdots \\ h_{\mathbf{w}}(\mathbf{x}^{(N)}) \end{bmatrix} - \underbrace{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}}_{\mathbf{y}}$$

Solution based on matrix algebra

Cost function: $J(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)}\right)^2$

Residuals:

$$\begin{bmatrix} h_{\mathbf{w}}(\mathbf{x}^{(1)}) - y^{(1)} \\ h_{\mathbf{w}}(\mathbf{x}^{(2)}) - y^{(2)} \\ \vdots \\ h_{\mathbf{w}}(\mathbf{x}^{(N)}) - y^{(N)} \end{bmatrix} = \begin{bmatrix} h_{\mathbf{w}}(\mathbf{x}^{(1)}) \\ h_{\mathbf{w}}(\mathbf{x}^{(2)}) \\ \vdots \\ h_{\mathbf{w}}(\mathbf{x}^{(N)}) \end{bmatrix} - \underbrace{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}}_{\mathbf{y}} = \begin{bmatrix} \mathbf{w}^T\mathbf{x}^{(1)} \\ \mathbf{w}^T\mathbf{x}^{(2)} \\ \vdots \\ \mathbf{w}^T\mathbf{x}^{(N)} \end{bmatrix} - \mathbf{y}$$

Solution based on matrix algebra

Cost function: $J(\mathbf{w}) = \dfrac{1}{N}\displaystyle\sum_{n=1}^{N}\left(h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)}\right)^2$

Residuals:

$$\begin{bmatrix} h_{\mathbf{w}}(\mathbf{x}^{(1)}) - y^{(1)} \\ h_{\mathbf{w}}(\mathbf{x}^{(2)}) - y^{(2)} \\ \vdots \\ h_{\mathbf{w}}(\mathbf{x}^{(N)}) - y^{(N)} \end{bmatrix} = \begin{bmatrix} h_{\mathbf{w}}(\mathbf{x}^{(1)}) \\ h_{\mathbf{w}}(\mathbf{x}^{(2)}) \\ \vdots \\ h_{\mathbf{w}}(\mathbf{x}^{(N)}) \end{bmatrix} - \underbrace{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}}_{\mathbf{y}} = \begin{bmatrix} \mathbf{w}^T\mathbf{x}^{(1)} \\ \mathbf{w}^T\mathbf{x}^{(2)} \\ \vdots \\ \mathbf{w}^T\mathbf{x}^{(N)} \end{bmatrix} - \mathbf{y} =$$

$$\begin{bmatrix} w_0 + w_1 x_1^{(1)} + \ldots + w_d x_d^{(1)} \\ w_0 + w_1 x_1^{(2)} + \ldots + w_d x_d^{(2)} \\ \vdots \\ w_0 + w_1 x_1^{(N)} + \ldots + w_d x_d^{(N)} \end{bmatrix} - \mathbf{y}$$

Solution based on matrix algebra

Cost function: $J(\mathbf{w}) = \dfrac{1}{N}\sum_{n=1}^{N}\left(h_{\mathbf{w}}(\mathbf{x}^{(n)}) - y^{(n)}\right)^2$

Residuals:

$$
R = \begin{bmatrix} h_{\mathbf{w}}(\mathbf{x}^{(1)}) - y^{(1)} \\ h_{\mathbf{w}}(\mathbf{x}^{(2)}) - y^{(2)} \\ \vdots \\ h_{\mathbf{w}}(\mathbf{x}^{(N)}) - y^{(N)} \end{bmatrix} = \begin{bmatrix} h_{\mathbf{w}}(\mathbf{x}^{(1)}) \\ h_{\mathbf{w}}(\mathbf{x}^{(2)}) \\ \vdots \\ h_{\mathbf{w}}(\mathbf{x}^{(N)}) \end{bmatrix} - \underbrace{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}}_{\mathbf{y}} = \begin{bmatrix} \mathbf{w}^T\mathbf{x}^{(1)} \\ \mathbf{w}^T\mathbf{x}^{(2)} \\ \vdots \\ \mathbf{w}^T\mathbf{x}^{(N)} \end{bmatrix} - \mathbf{y} =
$$

$$
\begin{bmatrix} w_0 + w_1 x_1^{(1)} + \ldots + w_d x_d^{(1)} \\ w_0 + w_1 x_1^{(2)} + \ldots + w_d x_d^{(2)} \\ \vdots \\ w_0 + w_1 x_1^{(N)} + \ldots + w_d x_d^{(N)} \end{bmatrix} - \mathbf{y} = \underbrace{\underbrace{\begin{bmatrix} 1 & x_1^{(1)} & \ldots & x_d^{(1)} \\ 1 & x_1^{(2)} & \ldots & x_d^{(2)} \\ \vdots & & & \\ 1 & x_1^{(N)} & \ldots & x_d^{(N)} \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_N \end{bmatrix}}_{w} - \underbrace{\mathbf{y}}_{y}}_{\mathbf{X}\mathbf{w} - \mathbf{y}}
$$

Thus, the vector of residuals can be expressed as

$$\begin{bmatrix} h_{\mathbf{w}}(\mathbf{x}^{(1)}) - y^{(1)} \\ h_{\mathbf{w}}(\mathbf{x}^{(2)}) - y^{(2)} \\ \vdots \\ h_{\mathbf{w}}(\mathbf{x}^{(N)}) - y^{(N)} \end{bmatrix} = \mathbf{X}\mathbf{w} - \mathbf{y}$$

R

We need the square of the residuals:

$$\begin{bmatrix} (h_{\mathbf{w}}(\mathbf{x}_1) - y_1)^2 \\ (h_{\mathbf{w}}(\mathbf{x}_2) - y_2)^2 \\ \vdots \\ (h_{\mathbf{w}}(\mathbf{x}_N) - y_N)^2 \end{bmatrix} = (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y})$$

It can be expressed as:

$$||\mathbf{X}\mathbf{w} - \mathbf{y}||^2$$

Minimizing E_{in}

J

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{2}{N}\mathrm{X}^{\intercal}(\mathrm{X}\mathbf{w} - \mathbf{y}) = \mathbf{0}$$

$$\mathrm{X}^{\intercal}\mathrm{X}\mathbf{w} = \mathrm{X}^{\intercal}\mathbf{y}$$

$\boxed{\mathbf{w} = \mathrm{X}^{\dagger}\mathbf{y}}$ where $\mathrm{X}^{\dagger} = (\mathrm{X}^{\intercal}\mathrm{X})^{-1}\mathrm{X}^{\intercal}$

$\mathrm{X}^{\dagger}$ is the '**pseudo-inverse**' of X

The pseudo-inverse

$$\mathrm{X}^{\dagger} = (\mathrm{X}^{\intercal}\mathrm{X})^{-1}\mathrm{X}^{\intercal}$$

$$\underbrace{\left(\underbrace{\begin{bmatrix} \quad \end{bmatrix}}_{d+1 \,\times\, d+1}\right)^{-1} \underbrace{\begin{bmatrix} \qquad\qquad \end{bmatrix}}_{d+1 \,\times\, N}}_{d+1 \,\times\, N}$$

X

N x (d+1)

$w = (w_0, w_1, \ldots, w_d)$

The linear regression algorithm

1: Construct the matrix X and the vector $\mathbf{y}$ from the data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$ as follows

$$\underbrace{\mathrm{X} = \begin{bmatrix} \text{—}\mathbf{x}_1^\intercal\text{—} \\ \text{—}\mathbf{x}_2^\intercal\text{—} \\ \vdots \\ \text{—}\mathbf{x}_N^\intercal\text{—} \end{bmatrix}}_{\text{input data matrix}}, \qquad \underbrace{\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}}.$$

$(N, d+1)$

2: Compute the pseudo-inverse $\mathrm{X}^\dagger = (\mathrm{X}^\intercal\mathrm{X})^{-1}\mathrm{X}^\intercal$.

3: Return $\boxed{\mathbf{w} = \mathrm{X}^\dagger\mathbf{y}}$.

$d+1$

$f(x) = w_0 + w_1 x_1 + \cdots + w_d x_d$

$w \rightarrow (1, x_1, \ldots, x_d)$

Computational cost

Solution: $\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$

We need to compute the inverse of $X^T X$ (dimension $(d+1) \times (d+1)$) $\longrightarrow$ expensive!

Complexity of matrix inversion: cubic

Computation of $X^T X$ is also expensive (N could be very large)

Due to the expensive computational cost, **gradient descend** based solution might be preferable

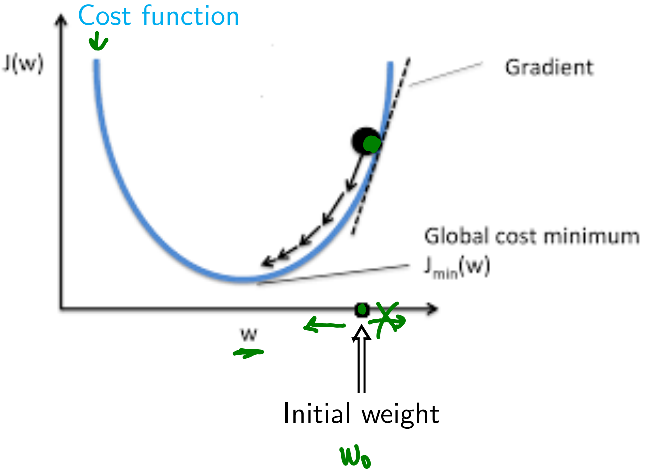
Gradient descent

Let $J(\mathbf{w})$ be the cost function to be minimized

Algorithm – pseudo code

- Initialize $\mathbf{w}$ (typically with small random values)
- Iterate until some stop criteria is met
  - Compute the gradient of J at $\mathbf{w}$ ("direction of fastest increase")
  - Update $\mathbf{w}$ in the negative direction of the gradient

Ilustration of the gradient descent technique



Example: MSE cost function

$$J(\mathbf{w}) = \boxed{\frac{1}{N}} \sum_{n=1}^{N} \Big(\underbrace{h_{\mathbf{w}}(\mathbf{x}^{(n)})}_{\hat{y}^{(n)} = \mathbf{w}^T \mathbf{x}^{(n)}} - y^{(n)} \Big)^2$$

Some notes

- Prof. Abu-Mostafa denotes J as E_{in}
- Sometimes $\frac{1}{N}$ is replaced with $\boxed{\frac{1}{2}}$ (or it even does not show up)

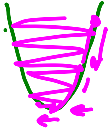
Gradient vector of J:

$$\nabla J(\mathbf{w}) = \left[\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}, \ldots, \frac{\partial J}{\partial w_d}\right]^T$$

$$\begin{aligned}
\boxed{\frac{\partial J}{\partial w_j}} &= \frac{\partial}{\partial w_j}\frac{1}{2}\sum_n(\hat{y}^{(n)} - y^{(n)})^2 \\
&= \frac{1}{2}\sum_n \frac{\partial}{\partial w_j}(\hat{y}^{(n)} - y^{(n)})^2 \\
&= \frac{1}{\cancel{2}}\sum_n \cancel{2}(\hat{y}^{(n)} - y^{(n)})\frac{\partial}{\partial w_j}(\hat{y}^{(n)} - y^{(n)}) \\
&= \sum_n(\hat{y}^{(n)} - y^{(n)})\frac{\partial}{\partial w_j}\left(\underline{(w_0 + w_1x_1^{(n)} + \ldots + w_jx_j^{(n)} + \ldots + w_dx_d^{(n)})} - y_n\right) \\
&= \boxed{\sum_n(\hat{y}^{(n)} - y^{(n)})\, x_j^{(n)}}
\end{aligned}$$

(Component j of the gradient vector depends on $(\hat{y}^{(n)} - y^{(n)})$ and the components j of all examples $\mathbf{x}^{(n)}$)

Gradient descent technique



Gradient of J: $\nabla J(\mathbf{w}) = \left[\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}, \ldots, \frac{\partial J}{\partial w_d}\right]$

$$\frac{\partial J}{\partial w_j} = \sum_n (\hat{y}^{(n)} - y^{(n)})\, x_j^{(n)}$$

learning rate

Initial weight: $\mathbf{w}(0)$

Weight update rule (iteration r):

$$\mathbf{w}(r+1) = \mathbf{w}(r) + \eta \Delta \mathbf{w}(r)$$

$$\Delta \mathbf{w}(r) = -\nabla J(\mathbf{w}), \qquad \boxed{\Delta w_j(r) = \sum_n (y^{(n)} - \hat{y}^{(n)})\, x_j^{(n)}}$$

η : learning rate (usually a small value, e.g, 0.001)

0.1, 0.01

Batch gradient descent

Algorithm 1 GradientDescent

Input: D, η, *epochs*

Output: $\mathbf{w}$

$\mathbf{w} \leftarrow$ small random value

repeat

  $\Delta w_j \leftarrow 0, \quad j = 0, 1, 2, \ldots, d$

  for all $(\mathbf{x}, y)$ in D **do**

    compute $\hat{y} = \mathbf{w}^T \mathbf{x}$

    $\boxed{\Delta w_j} \leftarrow \Delta w_j + (y - \hat{y})\, x_j, \quad j = 0, 1, 2, \ldots, d$

  end for

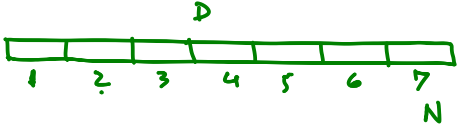
  $w_j \leftarrow w_j + \eta \Delta w_j, \quad j = 0, 1, 2, \ldots, d$

until number of iterations = *epochs*

Stochastic gradient descent

Algorithm 2 Stochastic GradientDescent

Input: D, η, *ephocs*
Output: $\mathbf{w}$
$\mathbf{w} \leftarrow$ small random value
repeat
  for all $(\mathbf{x}, y)$ in D **do**
    compute $\hat{y} = \mathbf{w}^T \mathbf{x}$
    $w_j \leftarrow w_j + \eta(y - \hat{y})\, x_j, \quad j = 0, 1, 2, \ldots, d$
  end for
until number of iterations = *epochs*



Batch gradient descent

$$\Delta w_j(r) = \sum_n \underbrace{\left(y^{(n)} - \hat{y}^{(n)}\right) x_j^{(n)}}$$

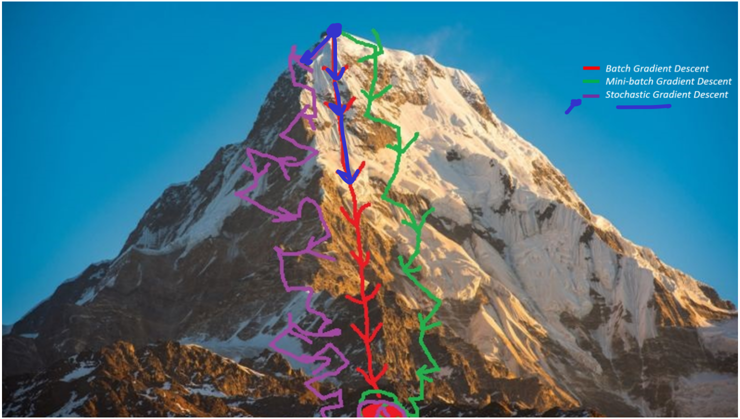
Stochastic gradient descent

$$\Delta w_j(r) = \underbrace{\left(y^{(n)} - \hat{y}^{(n)}\right) x_j^{(n)}}$$

Mini-batch gradient descent

In-between both

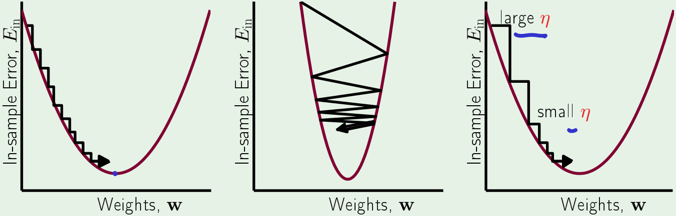
Gradient descent



https://imaddabbura.github.io/post/gradient_descent_algorithms/

Fixed-size step?

How η affects the algorithm:



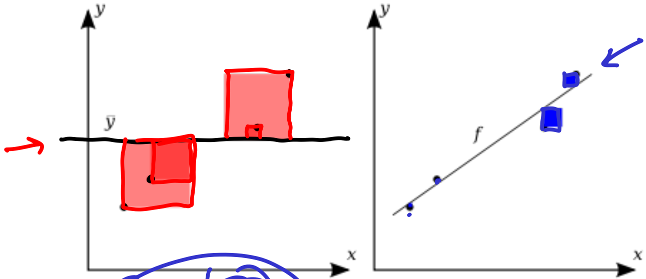
η too small $\qquad$ η too large $\qquad$ variable η – just right

η should increase with the slope

Goodness of fit

R^2

Coefficient of determination (wikipedia)



$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

~ 0

$R^2 \sim 1$

The better the linear regression (on the right) fits the data in comparison to the simple average (on the left graph), the closer the value of R^2 is to 1. The areas of the blue squares represent the squared residuals with respect to the linear regression. The areas of the red squares represent the squared residuals with respect to the average value.