인공지능과 수학적 배경

Linear Regression

Multiple variables

- **Data may have K even M features/attributes (curse of dimensionality).**

d=1 variable

	Size x	Price y
n data	31	124
	54	156
	…	…

Prediction function:

$$f_w(x) = w_0 + w_1 x$$

d=3 variables

	Size $x_{(1)}$	#Rooms $x_{(2)}$	Age $x_{(3)}$	Price y
n data	31	2	6	124
	54	3	12	156
	…	…	…	…

Prediction function:

$$f_w(x_{(1)}, x_{(2)}, x_{(3)}) = w_0 + w_1 x_{(1)} + w_2 x_{(2)} + w_3 x_{(3)}$$

Notation: $x_{(j)}$ = j^{th} variable of x data

Linear Regression
Data matrix X

- **Notation:**
  - ***n*** **= number of training data**
  - ***d*** **= number of variables/features, dim(x)**
  - **x_i = i^{th} training data, size(x_i) = *d* x 1**
  - **$x_{(j)}$ = j^{th} feature of training data x, scalar**
  - **X = data matrix, size(X)= *n* x *d***
  - **X_{ij} = $x_{i(j)}$ = j^{th} variable of i^{th} training data**
- **Example:**

$$x_1 = \begin{bmatrix} 2 \\ 1 \\ 7 \end{bmatrix} \updownarrow \text{d variables}$$

d **x 1**

$$x_1^T = [2 \quad 1 \quad 7]$$

1 x *d*

$$\begin{array}{cc} \overset{\text{d variables}}{\longleftrightarrow} & \\ X = \begin{bmatrix} — x_1^T — \\ \vdots \\ — x_n^T — \end{bmatrix} & \updownarrow \text{n data} \end{array}$$

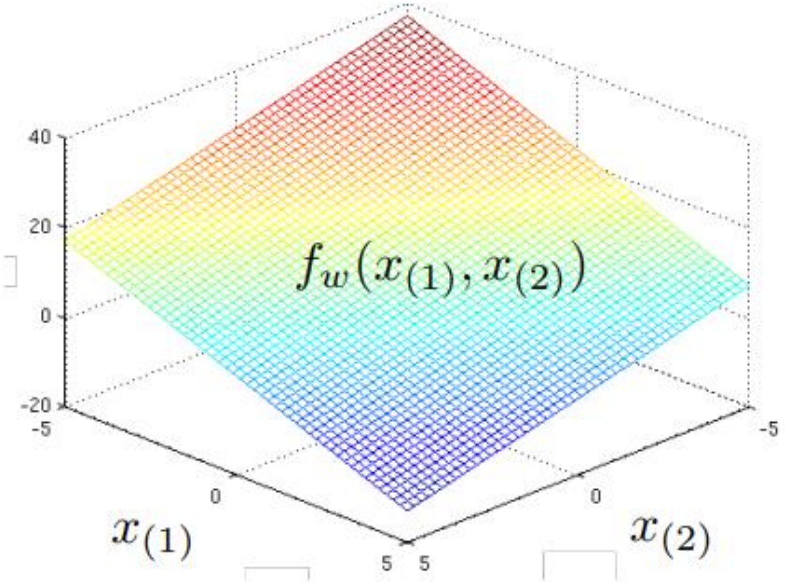
Data matrix
n **x *d***

All training data in a single matrix
(very efficient for linear algebra computations)

Linear Regression

Prediction function with *d* variables

- ***d*=1 variable:** $f_w(x) = w_0 + w_1 x$
- ***d* variables:** $f_w(x_{(1)}, \ldots, x_{(d)}) = w_{(0)} + w_1 x_{(1)} + \ldots + w_d x_{(d)}$
- **Example: *d*=2** $f_w(x_{(1)}, x_{(2)}) = w_{(0)} + w_1 x_{(1)} + w_d x_{(2)}$



$$f_w(x_{(1)}, x_{(2)}) = -0.3 + 1.2x_{(1)} + 9.3x_{(2)}$$

Linear Regression

Vector representation

- **Define the vectors:**

$$x = \begin{bmatrix} 1 \\ x_{(1)} \\ x_{(2)} \\ \vdots \\ x_{(d)} \end{bmatrix} \qquad w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

(d+1) x 1 (d+1) x 1

- **Re-write the prediction function (as vector-vector multiplication):**

$$f_w(x) = w_0.1 + w_1 x_{(1)} + \ldots + w_d x_{(d)}$$

$$= w^T x$$ One line of code

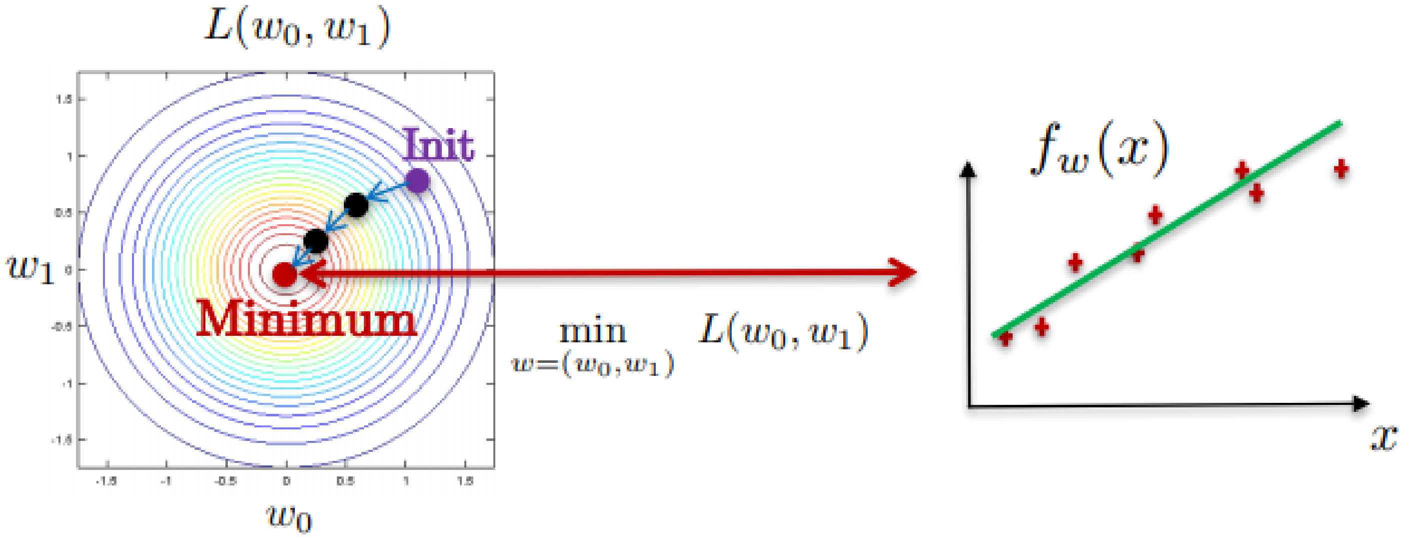
1 x (d+1) (d+1) x 1

f is called multivariate linear regression function.

Linear Regression

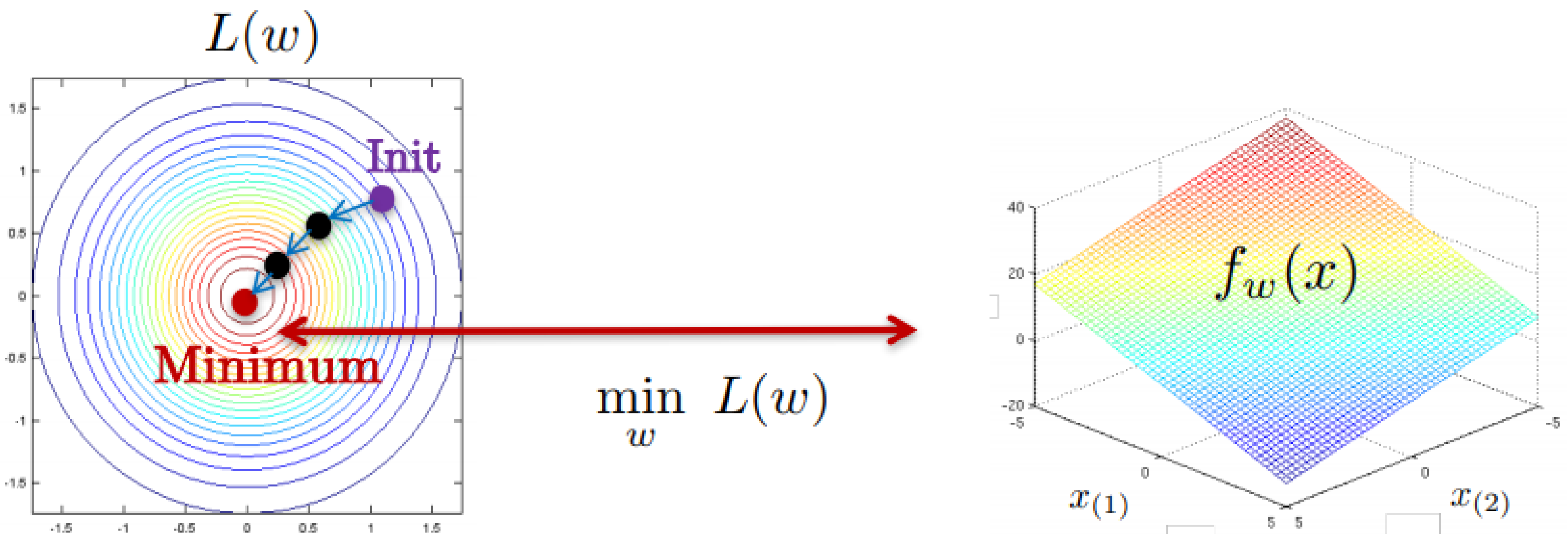
Gradient descent with d=1 variable

- **Prediction function:** $f_w(x) = w_0 + w_1 x$
- **Parameters:** w_0, w_1
- **Loss function:** $L(w_0, w_1) = \frac{1}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_i - y_i\right)^2$
- **Optimization:** $\min_{w=(w_0,w_1)} L(w_0, w_1)$
- **Gradient descent:** $w_j \leftarrow w_j - \tau \frac{\partial}{\partial w_j} L(w)$



Gradient descent with *d* variables

- **Prediction function:** $f_w(x) = w^T x = w_0 + w_1 x_{(1)} + \dots + w_d x_{(d)}$
- **Parameters:** $w = [w_0, w_1, \dots, w_d]$
- **Loss function:** $L(w) = \dfrac{1}{n} \sum_{i=1}^{n} \left(w^T x_i - y_i \right)^2$
- **Optimization:** $\min_w \; L(w)$
- **Gradient descent:** $w_j \leftarrow w_j - \tau \dfrac{\partial}{\partial w_j} L(w)$



Linear Regression

Gradient descent equations

- *d*=1 (one variable):

$$\frac{\partial}{\partial w_j}L(w_0, w_1) = \frac{\partial}{\partial w_j}\left[\frac{1}{n}\sum_{i=1}^{n}\left(w_0.1 + w_1 x_i - y_i\right)^2\right]$$

$$w_0 \leftarrow w_0 - \tau\frac{2}{n}\sum_{i=1}^{n}(w_0 + w_1 x_i - y_i).1$$

$$w_1 \leftarrow w_1 - \tau\frac{2}{n}\sum_{i=1}^{n}(w_0 + w_1 x_i - y_i).x_i$$

- *d* variable

$$\frac{\partial}{\partial w_j}L(w) = \frac{\partial}{\partial w_j}\left[\frac{1}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_{i(1)} + \ldots + w_d x_{i(d)} - y_i\right)^2\right]$$

Linear Regression

Gradient descent equations

- **Gradient:**

$$\frac{\partial}{\partial w_j} L(w) = \frac{\partial}{\partial w_j}\left[\frac{1}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_{i(1)} + \ldots + w_d x_{i(d)} - y_i\right)^2\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial w_j}\left[\left(w_0 + w_1 x_{i(1)} + \ldots + w_d x_{i(d)} - y_i\right)^2\right]$$

$$= \frac{2}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_{i(1)} + \ldots + w_d x_{i(d)} - y_i\right).$$

$$\frac{\partial}{\partial w_j}\left(w_0 + w_1 x_{i(1)} + \ldots + w_d x_{i(d)} - y_i\right)$$

$$= \frac{2}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_{i(1)} + \ldots + w_d x_{i(d)} - y_i\right) x_{i(j)}$$

- **Gradient descent:**

$$w_j \leftarrow w_j - \tau \frac{2}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_{i(1)} + \ldots + w_d x_{i(d)} - y_i\right).x_{i(j)}$$

Data matrix X_{ij}

Linear Regression

Matrix-vector representation

- Vectorize gradient descent scheme:

$$w_j \leftarrow w_j - \tau \frac{2}{n} \sum_{i=1}^{n} (w_0 + w_1 x_{i(1)} + \dots + w_d x_{i(d)} - y_i).x_{i(j)}$$

$$w_j \leftarrow w_j - \tau \frac{2}{n} \sum_{i=1}^{n} X_{ij}.(x_i^T w - y_i)$$

$$\underset{(d+1) \text{ x } 1}{x_i} = \begin{bmatrix} 1 \\ x_{i(1)} \\ \vdots \\ x_{i(d)} \end{bmatrix} \qquad \underset{(d+1) \text{ x } 1}{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

$$w_j \leftarrow w_j - \tau \frac{2}{n} X_j^T (Xw - y)$$

for w_0

$$\underset{n \text{ x } (d+1)}{X} = \begin{bmatrix} 1 & -x_1^T- \\ & \vdots \\ 1 & -x_n^T- \end{bmatrix} \qquad \underset{n \text{ x } 1}{X_j} = \begin{bmatrix} x_{1(j)} \\ \vdots \\ x_{n(j)} \end{bmatrix}$$

Data matrix

$$\underset{(d+1) \text{ x } 1}{w} \leftarrow w - \tau \frac{2}{n} \underbrace{X^T}_{(d+1) \text{ x } n} \underbrace{(Xw - y)}_{n \text{ x } 1}$$

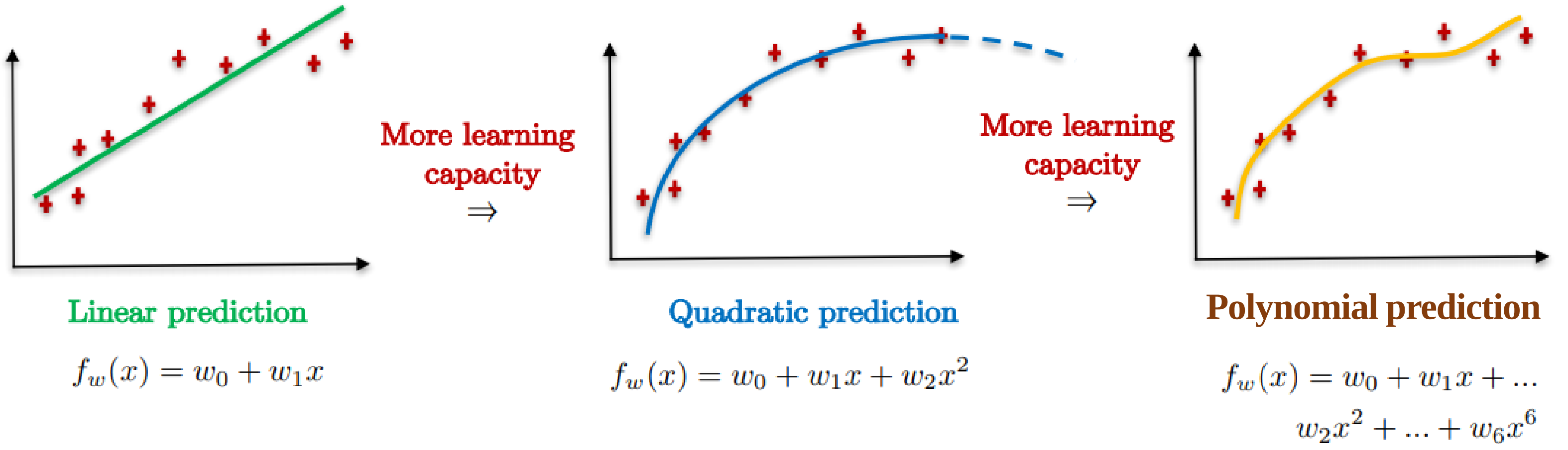
1 line of code

$$\underset{n \text{ x } 1}{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Linear Regression

Beyond linear regression

- **Example:** **Housing prices prediction function**



- **Handcrafted prediction function:** **Domain expertise allows to define better families of predictive regression functions. Example:**

$$f_w(x) = w_0 + w_1 \sqrt{x} + w_2 e^{-x}$$

- **Best non-linear regression technique:** **Neural networks.**

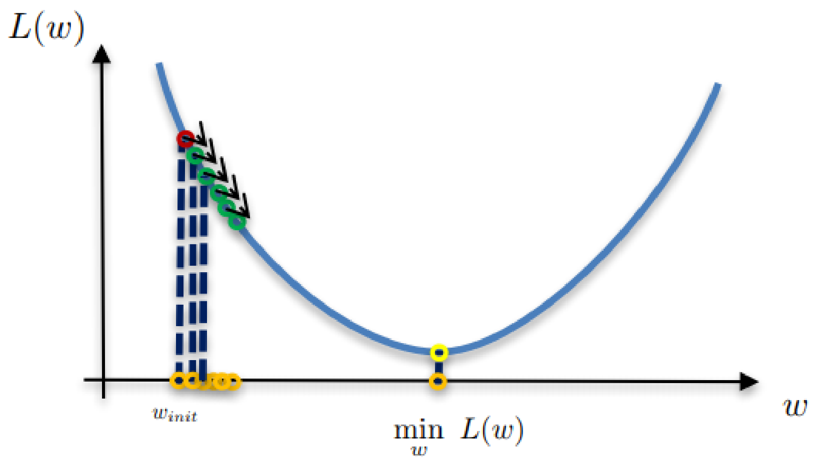
Beyond gradient descent

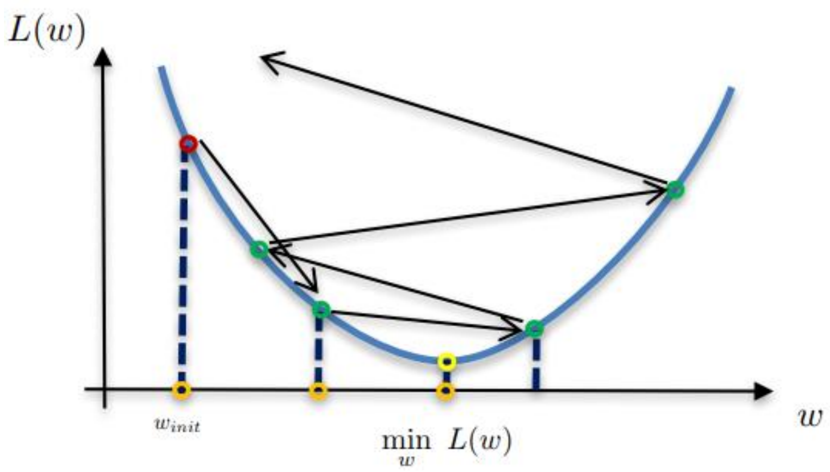
- **Gradient descent is the most generic optimization technique.**

$$\min_w \; L(w)$$

$$w_j \leftarrow w_j - \tau \frac{\partial}{\partial w_j} L(w)$$

- **But it has limitations:**
  - **Choice of learning rate τ**
  - **Convergence speed (even with optimal τ)**

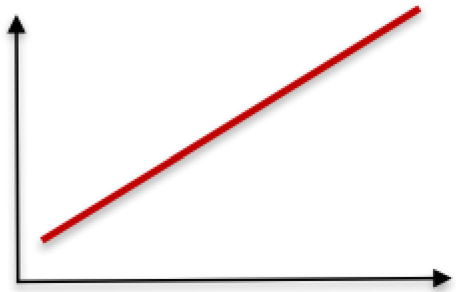




Beyond gradient descent

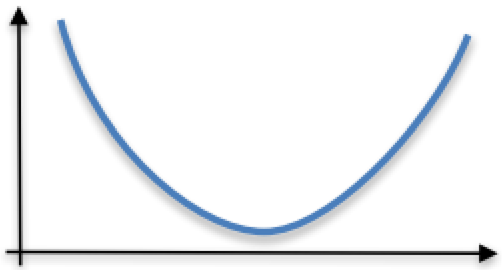
- We can leverage some mathematical properties to speed up the optimization.
  - **Prediction** function is **linear**:

$$f_w(x) = w^T x = w_0 + w_1 x_{(1)} + \ldots + w_d x_{(d)}$$



  - **Loss** function is **convex** (quadratic):

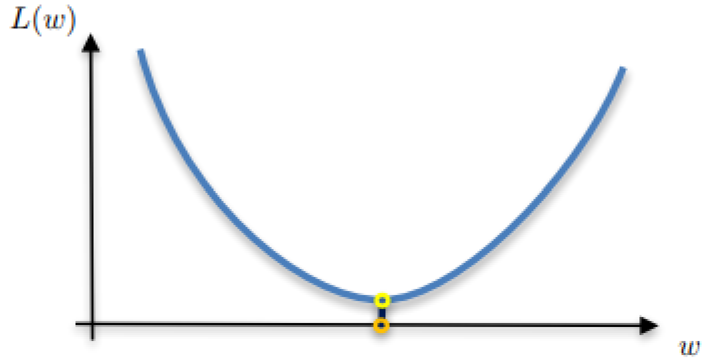
$$L(w) = \frac{1}{n} \sum_{i=1}^{n} \left(f_w(x_i) - y_i \right)^2$$



Normal equation for d=1 and n=1

- **Normal equation: Solution of mean square error loss**

$$\min_{w} \left\{ L(w) = (wx - y)^2 \right\}$$



- **The minimum is obtained when the gradient/slope is zero:**

$$\frac{\partial}{\partial w} L(w) = 0$$

$$\frac{\partial}{\partial w}(wx - y)^2 = 2x(wx - y) = 0 \Rightarrow w = x^{-1}y$$

Solution

One line of code

Normal equation for d=1 and n data

- **Loss L:**

$$L(w) = \frac{1}{n}\sum_{i=1}^{n}\left(wx_i - y_i\right)^2$$

- **Gradient of the loss L w.r.t. w:**

$$\frac{\partial}{\partial w}L(w) = 0 \;\Rightarrow \min_{w}\; L(w)$$

$$\begin{aligned}
\frac{\partial}{\partial w}\left[\frac{1}{n}\sum_{i=1}^{n}\left(wx_i - y_i\right)^2\right] &= \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial w}\left[\left(wx_i - y_i\right)^2\right] \\
&= \frac{2}{n}\sum_{i=1}^{n}(wx_i - y_i)\frac{\partial}{\partial w}(wx_i - y_i) \\
&= \frac{2}{n}\sum_{i=1}^{n}(wx_i - y_i)x_i \\
&= \frac{2}{n}w\sum_{i=1}^{n}x_i^2 - \frac{2}{n}\sum_{i=1}^{n}y_i x_i \\
&= 0 \Rightarrow \underset{\text{Solution}}{w = \frac{\sum_{i=1}^{n} y_i x_i}{\sum_{i=1}^{n} x_i^2}}
\end{aligned}$$

Vectorization

- **Loss L:**

$$L(w) = \frac{1}{n}\sum_{i=1}^{n}\left(wx_i - y_i\right)^2$$

$$= \frac{1}{n}\underbrace{(wx-y)^T}_{1 \times n}\underbrace{(wx-y)}_{n \times 1}$$

$$\underset{n \times 1}{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$\underset{n \times 1}{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Vectorization

- **Gradient of the loss *L* w.r.t. *w*:**

$$\frac{\partial}{\partial w} L(w) = 0 \ \Rightarrow \min_{w} \ L(w)$$

$$\begin{aligned}
\frac{\partial}{\partial w}\left[\frac{1}{n}(wx-y)^T(wx-y)\right] &= \frac{1}{n}\frac{\partial}{\partial w}\left[\left(wx^T-y^T\right)(wx-y)\right] \\
&= \frac{2}{n}x^T(wx-y) \\
&= \frac{2}{n}\left(wx^T x - x^T y\right) \\
&= 0 \Rightarrow w = \left(x^T x\right)^{-1} x^T y
\end{aligned}$$

One line of code

Linear Regression

Normal equation for *d* features and *n* data

- **Loss *L*:**

$$L(w_0, \dots, w_d) = \frac{1}{n} \sum_{i=1}^{n} \left(w_0 + w_1 x_{i(1)} + \dots + w_d x_{i(d)} - y_i \right)^2$$

- **Gradient of the loss *L* w.r.t. *w*:**

$$\frac{\partial}{\partial w_j} L(w_0, \dots, w_d) = 0 \quad \forall j$$

$$\frac{\partial}{\partial w_j} L(w) = \frac{\partial}{\partial w_j} \left[\frac{1}{n} \sum_{i=1}^{n} \left(w_0 + w_1 x_{i(1)} + \dots + w_d x_{i(d)} - y_i \right)^2 \right]$$

$$= \frac{2}{n} \sum_{i=1}^{n} \left(w_0 + w_1 x_{i(1)} + \dots + w_d x_{i(d)} - y_i \right) x_{i(j)}$$

$$= 0 \Rightarrow w_j = \frac{\sum_{k \neq j} \sum_i w_k x_{i(k)} x_{i(j)}}{\sum_i x_{i(j)}^2} \qquad \text{with } x_{i(0)} = 1$$

Solution

Linear Regression
Vectorization

◆ **Loss L:**

$$L(w_0, ..., w_d) = \frac{1}{n}\sum_{i=1}^{n}\left(w_0 + w_1 x_{i(1)} + ... + w_d x_{i(d)} - y_i\right)^2$$

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

$$L(w) = \frac{1}{n}\left(Xw - y\right)^T\left(Xw - y\right)$$

$$\underset{(d+1)\times 1}{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix} \quad \underset{(d+1)\times 1}{x_i} = \begin{bmatrix} x_{i(0)} \\ \vdots \\ x_{i(d)} \end{bmatrix} \quad \underset{n\times(d+1)}{X} = \begin{bmatrix} — x_1^T — \\ \vdots \\ — x_n^T — \end{bmatrix} \quad \underset{n\times 1}{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

with $x_{i(0)} = 1$

Data matrix

Linear Regression

Vectorization

- **Gradient of the loss *L* w.r.t. *w*:**

$$\begin{aligned}
\frac{\partial}{\partial w}L(w) &= \frac{1}{n}\frac{\partial}{\partial w}\left[\left(Xw-y\right)^T\left(Xw-y\right)\right] \\
&= \frac{1}{n}\frac{\partial}{\partial w}\left[\left(w^TX^T-y^T\right)\left(Xw-y\right)\right] \\
&= \frac{2}{n}X^T\left(Xw-y\right) \\
&= \frac{2}{n}\left(X^TXw-X^Ty\right) \\
&= 0 \Rightarrow w=(X^TX)^{-1}X^Ty
\end{aligned}$$

One line of code