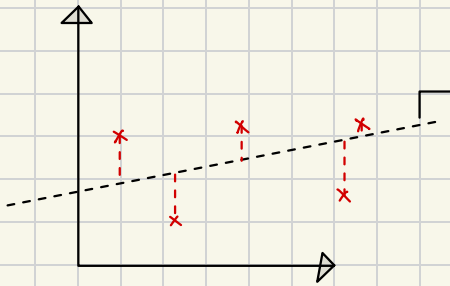
Linear Regression with GD



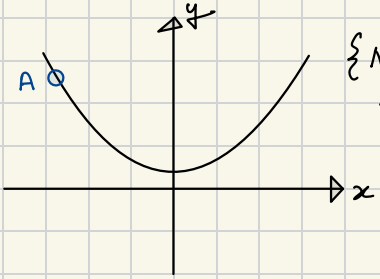
error = y − guess

(input, output)

x $\quad$ y

Cost: $\sum_{i=0}^{n} (\hat{y}_i - y_i)^2$ ⟶ We want to minimise this function.

$\therefore$ Cost = $\sum (\hat{y}_i - y_i)^2$ this is kind of like $y = x^2$



{ Now, we need to find the value of x for that y is lowest.

Suppose, we're at point A & if we want to reduce the function, we need to find its gradient. But how do we go down?

⟶ We calculate the slope.

$m = m + \Delta m$

$b = b + \Delta b$

$\therefore J(m,b) = \frac{1}{n} \sum_{i=0}^{n} (mx + b - y)^2$

$\frac{\delta}{\delta m} J = 2(mx + b - y)m$ & $\frac{\delta}{\delta b} J = 2(mx + b - y)(1)$

* $\frac{1}{n}$ in $J(m,b) = \frac{1}{n}\sum(\hat{y}_i - y_i)$?

→ We want to take the average error & minimise it.

→ Makes it scale irrelevant so that same learning rate works in different datasets.

We also sometimes take $\frac{2}{n}$, this is purely because of algebraic convenience.

Update rule: $m^{(t+1)} = m - \alpha \frac{\partial}{\partial m} J$

$b^{(t+1)} = b - \alpha \frac{\partial}{\partial b} J$

* Example:

$(x, y) = \{(1,2), (2,3), (3,5)\}$ predict y when $x = 7$

here, $n = 3$, model, $\hat{y} = mx + b$ let, $\hat{y}_1 = \hat{y}_2 = \hat{y}_3 = 0$ { because, we assumed $m = 0$ & $b = 0$ }

$$J = \frac{1}{n}\sum_0^n (\hat{y}_i - y_i)^2 = \{(0-2)^2 + (0-3)^2 + (0-5)^2\}/3$$

$$= \frac{4+9+25}{3} = \frac{38}{3} \approx 12.67$$

let, $\alpha = 0.01$

Now, we'll minimise loss by updating m & b

Given, $J = \frac{1}{n}\sum_0^n (mx + b - y_i)^2$

$$\therefore \frac{\delta J}{\delta m} = \frac{1}{n}\sum(mx_i + b - y_i) \times 2 \times (x_i)$$

$$\& \quad \frac{\delta J}{\delta x} = \frac{2}{n}\sum(mx_i + b - y_i)(1)$$

$m^{(t+1)} = m^t + \alpha \frac{\delta}{\delta m} J$

$b^{(t+1)} = b^t + \alpha \frac{\delta}{\delta b} J$

1st step,

$$\frac{\delta J}{\delta m} = \frac{1}{3}\{(0 \times 1 + 0 - 2) \times 2(1) + (0 \times 2 + 0 - 3)(2 \times 2) + (-5)(2 \times 3)\} = -15.33$$

$$\therefore m^1 = m^0 - \alpha\left(\frac{\delta J}{\delta m}\right) = 0 - 0.01(-15.33) = 0.1533$$

Let's get b^1

$$\frac{\delta J}{\delta b} = \frac{2}{n}\sum_{0}^{n}(mx_i + b - y_i) = \frac{2}{3}\left(-2-3-5\right) = -6.667$$

$$\therefore b^1 = b^0 - \alpha\left(\frac{\delta J}{\delta b}\right) = 0 - 0.0667 = 0.0667$$

So, model updates to,

$$\hat{y}_i = 0.1533\,x_i + 0.0667$$

$$\therefore \hat{y}_1 = 0.22 \quad \hat{y}_2 = 0.3733 \quad \hat{y}_3 = 0.5266$$

Current error: $J = \frac{1}{n}\sum_{0}^{n}(\hat{y}_i - y_i)^2$

$$= \frac{1}{3}\left((0.22-2)^2 + (0.3733-5)^2 + (0.5266-5)^2\right)$$

$$= \{(-1.78)^2 + (-4.6267)^2 + (-4.4734)^2\}/3$$

$$= 10.022067$$

So, we've reduced error in step 1 but we have two more steps to go. It took a lot of manual calculations, let's simplify a bit using matrices.

* Dataset $\{(1,2), (2,3), (3,5)\}$ & prediction $\hat{y}_i = mx_i + b$

$\theta = \begin{pmatrix} m \\ \theta \end{pmatrix}$ so, $\overline{\hat{y}} = \overline{x}\,\overline{\theta}$

* Design matrix:

$$\overline{x} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} \quad \overline{\theta} = \begin{pmatrix} m \\ \theta \end{pmatrix} \quad \overline{\hat{y}} = \overline{x}\,\overline{\theta} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix}\begin{pmatrix} m \\ \theta \end{pmatrix}$$

$$= \begin{pmatrix} m+\theta \\ 2m+\theta \\ 3m+\theta \end{pmatrix}$$

as $m, \theta = 0, 0$ $\qquad n = 3$

$\therefore \bar{\hat{y}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ $\qquad$ Given, $\bar{y} = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$

$\bar{e} = \bar{\hat{y}} - \bar{y} = \begin{pmatrix} -2 \\ -3 \\ -5 \end{pmatrix}$ $\qquad$ $\bar{e}^2 = (\bar{\hat{y}} - \bar{y})^2 = \|\bar{\hat{y}} - \bar{y}\|_2$

$= (\bar{\hat{y}} - \bar{y})^T (\bar{\hat{y}} - \bar{y})$

$= (2 \quad 3 \quad 5) \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$

$= (4 + 9 + 25) = 38$

* Cost Function,

$J = \frac{\bar{e}^2}{n} = 38/3 = 12.667$

* Gradient,

$J(\theta) = 1/n \left(\bar{\hat{y}} - \bar{y}\right)^T \left(\bar{\hat{y}} - \bar{y}\right) = \frac{1}{n} (\bar{x}\theta - \bar{y})^T (\bar{x}\theta - \bar{y})$

$= \frac{1}{n} (\bar{x}^T\theta^T - \bar{y}^T)(\bar{x}\theta - \bar{y})$

$= \frac{1}{n} (\theta^T\bar{x}^T\bar{x}\theta - \bar{x}^T\theta^T\bar{y} - \bar{y}^T\bar{x}\theta - \bar{y}^T\bar{y})$

$= \frac{1}{n} (\theta^T\bar{x}^T\bar{x}\theta - 2\bar{x}^T\theta^T\bar{y} - \bar{y}^T\bar{y})$

$J(\theta) = \frac{1}{n} (\theta^T\bar{x}^T\bar{x}\theta - 2y^T\bar{x}\theta - \bar{y}^T\bar{y})$

* Matrix Calculus Identities

1. $\nabla_\theta (\theta^T A \theta) = 2A\theta$ {When A is symmetric & if $A = x^Tx$ then A is always symmetric}
2. $\nabla_\theta (c^T\theta) = c$
3. $\nabla_\theta (\text{constant}) = 0$

→ Appendix 1

* Applying these in the cost function,

1. $\nabla_\theta (\theta^T\bar{x}^T\bar{x}\theta) = 2\bar{x}^T\bar{x}\theta$
2. $\nabla_\theta (2\bar{y}^T\bar{x}\theta) = 2\bar{x}^Ty$
3. $\nabla_\theta (\bar{y}^T\bar{y}) = 0$

$\therefore \nabla_\theta J = (2\bar{x}^T\bar{x}\theta - 2\bar{x}^T\bar{y})/n$

$= \frac{2\bar{x}^T}{n} \underbrace{(\bar{x}\theta - \bar{y})}_{\text{Error Vector}}$

↑ Gradient (w.r.t m & b)

Update rule : $\theta := \theta - \alpha \nabla_\theta J$

Let's verify !

$\{(1,2),(2,3),(3,5)\}$

$\bar{x} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix}$ A $\quad \bar{\theta} = \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ B $\quad \bar{y} = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$ C $\quad \hat{y} = \bar{x}\bar{\theta} \quad \bar{e} = \hat{y} - y$

$y_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

$\bar{e} = \begin{pmatrix} -2 \\ -3 \\ -5 \end{pmatrix}$

*Cost Calc

$J(\theta) = \frac{1}{n}(\bar{x}\theta - y)^T(\bar{x}\theta - y)$

$= \frac{1}{3}\begin{pmatrix} -2 & -3 & -5 \end{pmatrix}\begin{pmatrix} -2 \\ -3 \\ -5 \end{pmatrix} = \frac{1}{3}(2^2 + 3^2 + 5^2) = 38/3 = 12.67$

* $\nabla_\theta J(\theta) = \frac{2}{n}\bar{x}^T(\bar{x}\theta - y)$

$= \frac{2}{3}\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}\left(\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}\right)$

$= \frac{2}{3}\begin{pmatrix} -23 \\ -10 \end{pmatrix} = \begin{pmatrix} -46/3 \\ -20/3 \end{pmatrix} = \begin{pmatrix} -15.33 \\ -6.667 \end{pmatrix}$

$\therefore \nabla_\theta J(\theta) = \begin{pmatrix} -15.33 \\ -6.667 \end{pmatrix}$

$\therefore \theta^1 = \theta - \alpha \begin{pmatrix} -15.33 \\ -6.667 \end{pmatrix} \{\alpha = 0.01\}$

$= \begin{pmatrix} 0.1533 \\ 0.0667 \end{pmatrix}$

$$\hat{y}_1 = \bar{x}\,\bar{\theta}$$

$$= \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} \begin{pmatrix} 0.1533 \\ 0.0667 \end{pmatrix} = \begin{pmatrix} 0.22 \\ 0.3733 \\ 0.5266 \end{pmatrix}$$

$$e^1 = \begin{pmatrix} -1.78 \\ -2.626 \\ -4.9739 \end{pmatrix}$$

$$\theta^{(2)} = \theta^{(1)} - \alpha\left(\nabla_\theta J(\theta)\right) = \theta^{(1)} - \alpha\left(\frac{2}{n} x^T (x\theta^{(1)} - y)\right)$$

$$= \theta^{(1)} - 0.01\left(\frac{2}{3}\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}\left(\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix}\begin{pmatrix} 0.1533 \\ 0.0667 \end{pmatrix} - \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}\right)\right)$$

$$= \theta^{(1)} - 0.01\left(\frac{2}{3}\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} -1.78 \\ -2.626 \\ -4.9739 \end{pmatrix}\right) e^1$$

$$\therefore \theta^{(2)} = \begin{pmatrix} 0.289698 \\ 0.125896 \end{pmatrix}$$

$e = x\theta - y$

$$\theta^3 = \theta^{(2)} - 0.01 \times \frac{2}{3}\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} -1.584 \\ -2.274 \\ -4.005 \end{pmatrix}$$

$$= \begin{pmatrix} 0.289698 \\ 0.125896 \end{pmatrix} - \begin{pmatrix} -0.121263 \\ -0.052563 \end{pmatrix} = \begin{pmatrix} 0.410912 \\ 0.1734 \end{pmatrix}$$

$$J(\theta)^{(3)} = \left|(\bar{x}\,\theta^{(3)} - y)\right|_2 \times 1/n \quad (n=3)$$

$$= 6.29$$

* How long will we go?

→ Our error term is reducing, so we can go as low as we want till error ≈ 0

But, we can assign few other stopping conditions

1. if $J(\theta) \approx 0$
2. iteration count limit
3. $\|e\|_2 \approx 0$
4. $\| J(\theta)^{t+1} - J(\theta)^{(1)} \| < \epsilon$ or ≈ 0

But no matter what we do, we need computational power because datasets can become very large!

→ We'll use python to create our linear regress with GD algorithm, calculate accuracy.

* This part is partly rough derivations *

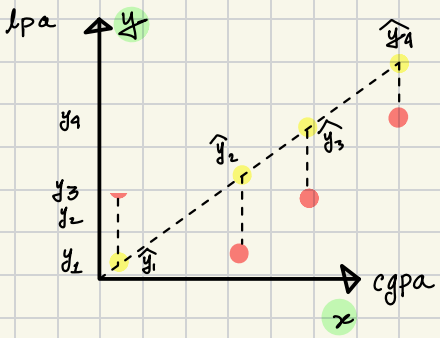
* No need to go through if you're not intereste *

A bit simpler

Gradient Descent is an optimisation algorithm for finding a local minimum of a differentiable function. The key idea is that we take repeated steps in the opposite direction of the gradient of the function at the current point, because this is the direction of the steepest descent.

Gradient Descent is a general algorithm that is used in linear regression to deep learning algorithms.

*Intuition



$n = 4$

$$RSS = \text{Loss Func} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \sum_{1} (y_1 - (mx_i + b))^2$$

Here $\hat{y}_i$ is a predicted linear function with a slope 'm'.

y_i's are the given values.

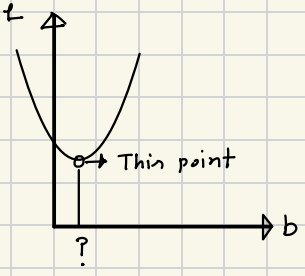
b → is the intercept.

Loss Funct (m, b).

We basically try to find m & b

Suppose, we know 'm', then we'll need to find a value of 'b' for that L is minimised.

If we plot a 'L' vs 'b' graph, it'll be like



Algorithm

Step 1: Select a random 'b'.

Step 2: Calculate slope at that point.

Step 3: Slope < 0 ; increase b

$$b_{new} = b_{old} - \text{slope}$$

$$b_{new} = b_{old} - \eta \, \text{slope}$$

η → Learning Rate

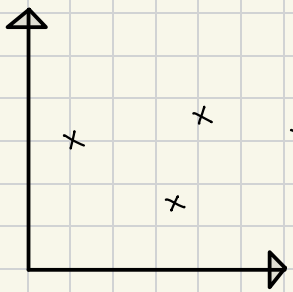
η → Learning rate tuning is very important.

When to stop?

$b_{new} - b_{old} \approx 0$ or limit iterations.

epoch in ML lingo

* Mathematical Foundation



At first we'll assume we know 'b'.

Step 1: Select a random b.

Step 2: for i in epochs

$b_{new} = b_{old} - \eta \times slope$; $\eta = 0.01$

Now, slope for b,

$$L = \sum_{1}^{n} (y_i - \hat{y}_i)^2 = \sum_{1}^{n} (y_1 - mx_i - b)^2$$

$$\therefore \frac{dL}{db} = 2 \sum_{1}^{n} (y_1 - mx_i - b)(0 - 0 - 1)$$

$$= -2 \sum_{1}^{n} (y_1 - mx_i - b)$$

$$= -2 \sum_{1}^{n} (y_1 - mx_i - 0) \quad \{b = 0\}$$

Given,

$m = -78.35$

$$\therefore b_{new} = b_{old} - \frac{\partial L}{\partial b} \times \eta$$

* This part follows lecture on Stanford CS229 *
* No need to go through if you've understood the concepts well *

Supervised Learning Setup

- Regression (Where does this data fit?) → Discrete
- Classification (What type of data is this?) → Classification

$x \rightarrow y$
(Input) (Output)

$n \rightarrow$ number of examples in training set.
(x, y)

$d \rightarrow x \in \mathbb{R}^d$ dimension of input.

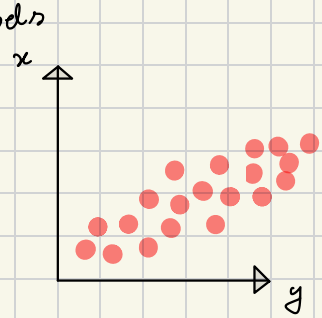
$x^{(i)} \rightarrow$ ith example input.

$y^{(i)} \rightarrow$ ith example output. (label / ground truth)

$(x^{(i)}, y^{(i)}) \rightarrow$ ith example.

* Regression

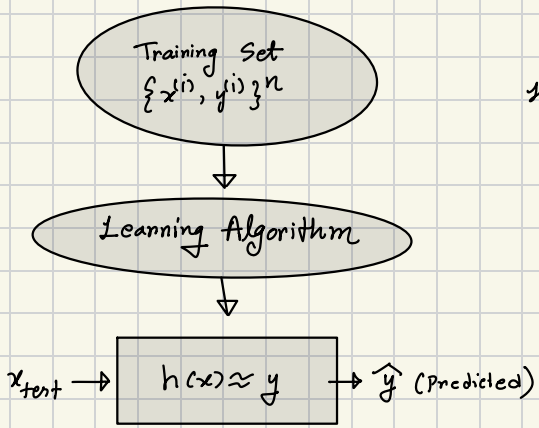
x	y
2104	400
1600	330
2400	369
3000	540



Goal : $f(x) \approx y$ → Learnt Model

Or find the best fit line

* Pipeline



* Linear Regression:

$x \in \mathbb{R}^d$; $y \in \mathbb{R}$, n such examples

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d$$

$$h_\theta(x) = \sum_{i=1}^{d} \theta_i x_i + \theta_0 \;;$$

$x_0 = 1 \rightarrow$ intercept term

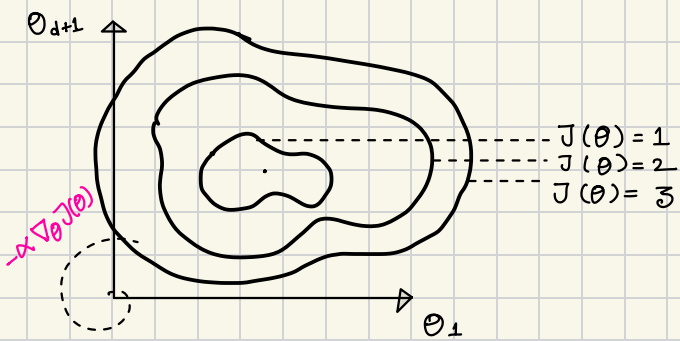
$$h_\theta(x) = \theta_0 x_0 + \dots + \theta_d x_d$$

$$= \sum_{i=0}^{d} x_i \theta_i = \theta^T x$$

Cost Function: $J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2$ | $J(\theta) = f(x)$

$\hat{\theta} \rightarrow \arg\min_{\theta} J(\theta) \rightarrow \arg\min_{\theta} \sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2$

*Gradient Descent



Contour plot of the cost function

$\theta^{(0)}$ = Initialisation

$$\theta_j^{(1)} = \theta_j^{(0)} - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$$

$$\therefore \theta^{(1)} = \theta^{(0)} - \alpha\nabla_\theta J(\theta^{(0)})$$

↳ In vector form (we repeat this till convergence)

so, $\theta^{(t+1)} = \theta^{(t)} - \alpha\nabla_\theta J(\theta^{(0)})$ ← Gradient

Gradient Descent on Linear Regression

$\theta^{(t+1)} = \theta^t - \alpha\nabla_\theta$ [Cost function of 1R]

$= \theta^t - \alpha\nabla_\theta\left(\frac{1}{2}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2\right)$

$= \theta^t - \alpha\nabla_\theta\left(\frac{1}{2}\sum_{i=1}^{n}(\theta^T x^{(i)} - y^{(i)})^2\right)$

$= \theta^t - \alpha\left(\frac{1}{2}\sum_{i=1}^{n}2(\theta^T x^{(i)} - y^{(i)})x^{(i)}\right)$

$\theta^{t+1} = \theta^t - \alpha\left(\sum_{i=1}^{n}(\theta^T x^{(i)} - y^{(i)})x^{(i)}\right)$

Scalar, Vector, Vector → Vector

How to check for convergence

1) $\|\theta^{(t)} - \theta^{(t+1)}\| \approx 0$

or, 2) $\|\nabla_\theta J(\theta^t)\| \approx 0$

or, 3) $|J(\theta^{t+1}) - J(\theta^t)| \approx 0$

θ → Set of all parameters.

*Stochastic Gradient Descent - SGD (I find it stupid)

→ For each small progress we need to scan through all of the dataset.

→ To make it a bit simple we can do,

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla_\theta \tilde{J}(\theta)$$

$$\tilde{J}(\theta) = \tfrac{1}{2}(\theta^T x^{(u)} - y^{(u)})^2$$

* Only linear regression has a closed form solution of GD. In other cases, the solution varies with the problem.

We defined $J(\theta)$ as,

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (\theta^T x^{(i)} - y^{(i)})^2$$

Design Matrix

$$\underset{n \times d+1}{X} = \begin{pmatrix} x^{(i)} \\ \vdots \\ x^{(i+1)} \\ \vdots \\ x^{n} \end{pmatrix} \qquad \underset{}{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \qquad \theta = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_{d+1} \end{pmatrix}$$

$$X\theta - y$$

$$\mathbb{R}^{n \times d+1} \; \mathbb{R}^{d+1} - \mathbb{R}^n = \mathbb{R}^n$$

$$\therefore X\theta - y = \begin{pmatrix} x^{(1)T}\theta - y^{(1)} \\ x^{(i)T}\theta - y^{(2)} \\ x^{(n)T}\theta - y^{(n)} \end{pmatrix}$$

so, $J(\theta) = \frac{1}{2} (X\theta - y)^T (X\theta - y)$

Now, $\nabla_\theta J(\theta) = 0$

$$\nabla_\theta \tfrac{1}{2} (X\theta - y)^T (X\theta - y)$$

$$= \nabla_\theta \tfrac{1}{2} \left((X\theta)^T (X\theta) - (X\theta)^T y - y^T (X\theta) - y^T y \right)$$

$$= \nabla_\theta \tfrac{1}{2} \left(\theta^T (X^T X) \theta - 2\theta^T (X^T y) + y^T y \right)$$

$$= \frac{1}{2} \left(2(X^T X)\theta - 2X^T y \right) = 0$$

$$\Rightarrow X^T X \theta = X^T y \quad \text{or,} \quad \hat{\theta} = (X^T X)^{-1} X^T y$$

↳ Normal Expression

{As long as X^T is invertible}

*Probabilistic Interpretation

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$$

$$\varepsilon \approx N(0, \sigma^2)$$