CS 3120 / IS 3117 / SCS 3201

Machine Learning and Neural Computing

Kasun Gunawardana

University of Colombo School of Computing

Linear Regression

Multivariate Linear Regression

Multivariate

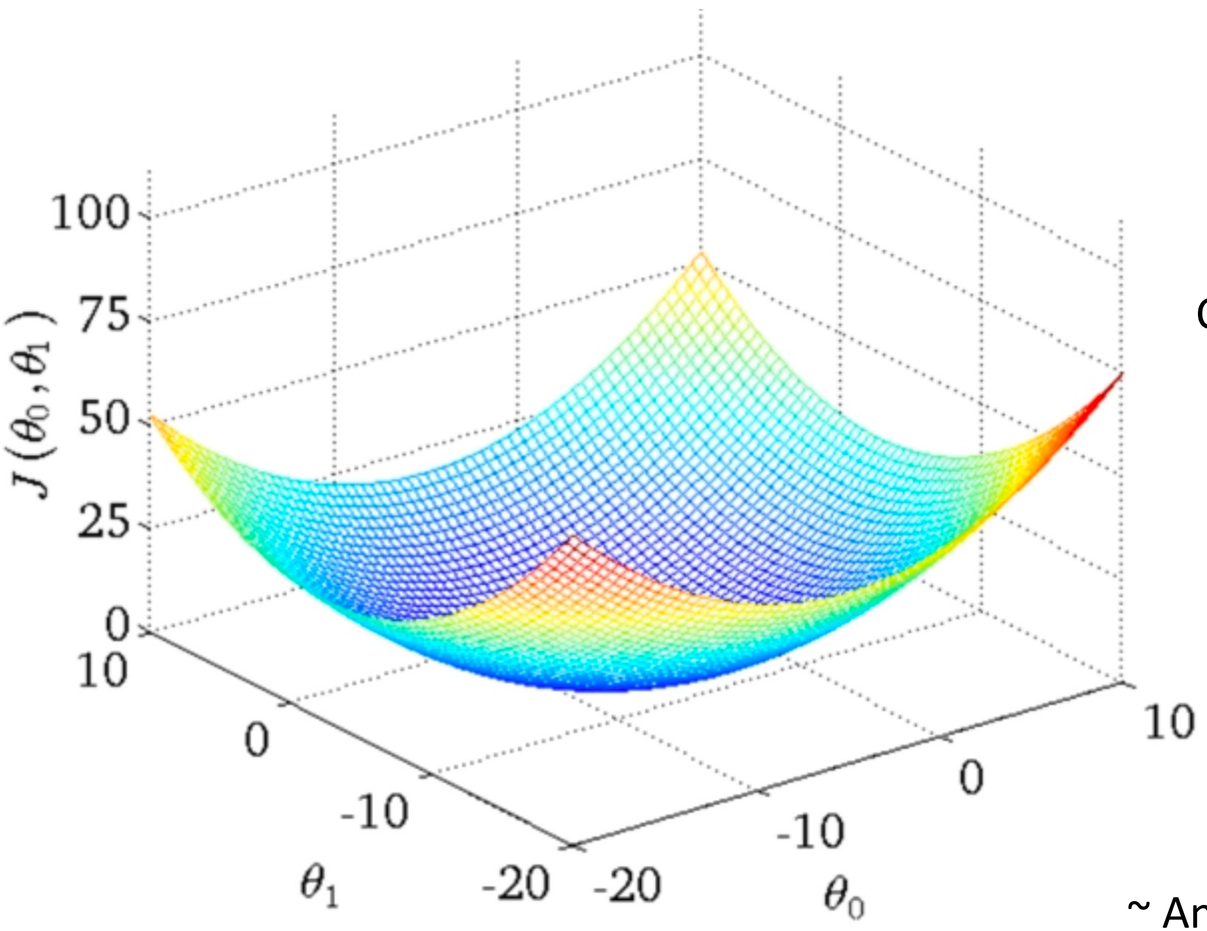
- Student data point

Age	Gender	Program	3-4 Year	GPA	Awards	Project	Programming	Union Member	Club Member	Discipline Issues	Recommendations	Language Proficiency	Pursuing Position	Extra Curricular
24	1	1	0	3.12	0	3	8	0	1	0	3	3	2	1

From the Last Lesson

- Univariate Linear Regression
- Cost Function
  - Mean Squred Error Function
- Gradient Descent Algorithm

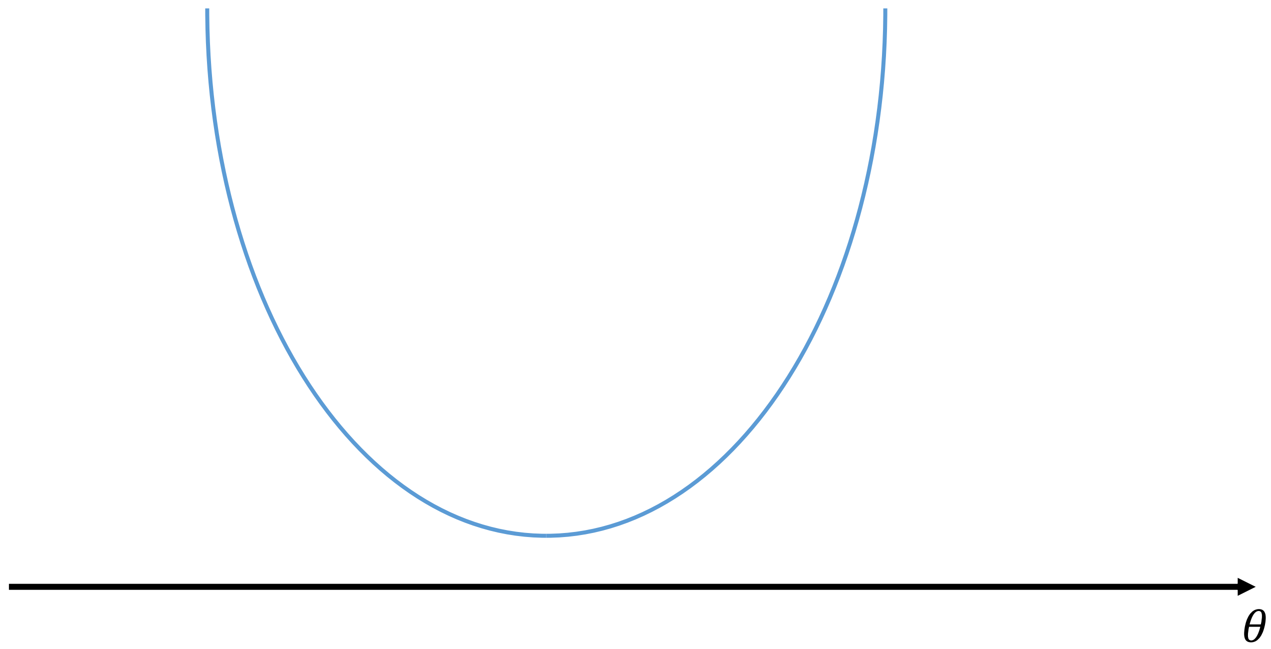
Cost Function in Linear Regression



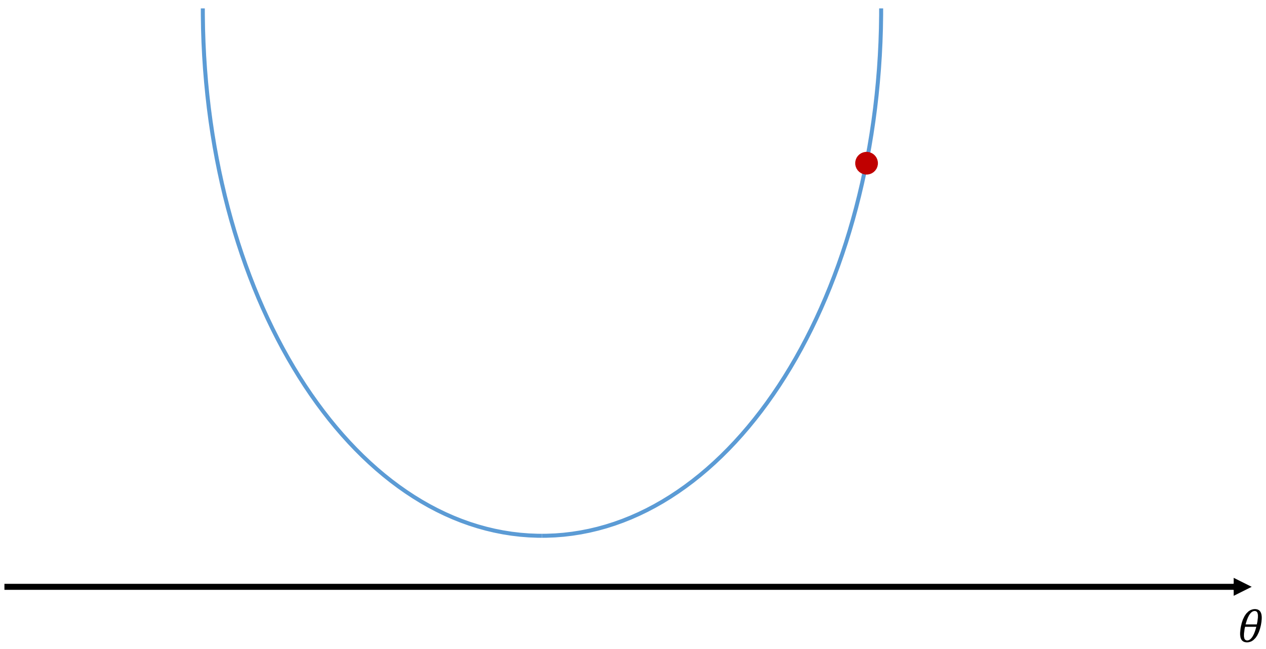
Convex Function

~ Andrew Ng

Cost Function - MSE

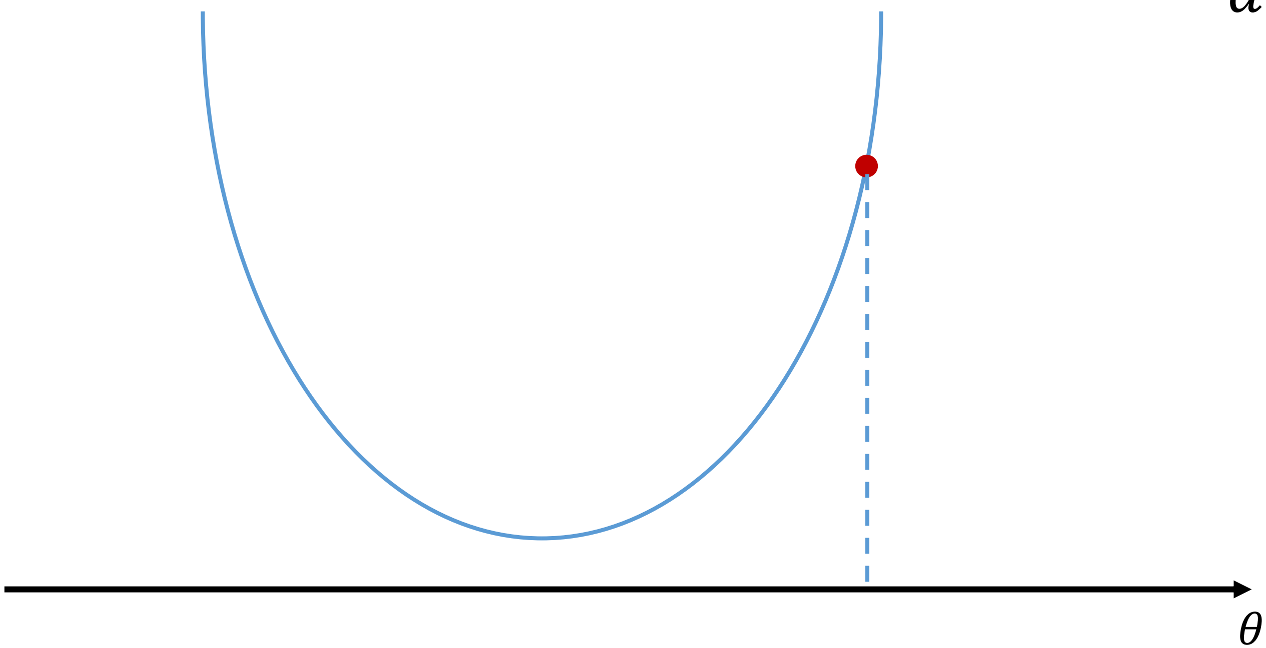


Gradient Descent - MSE



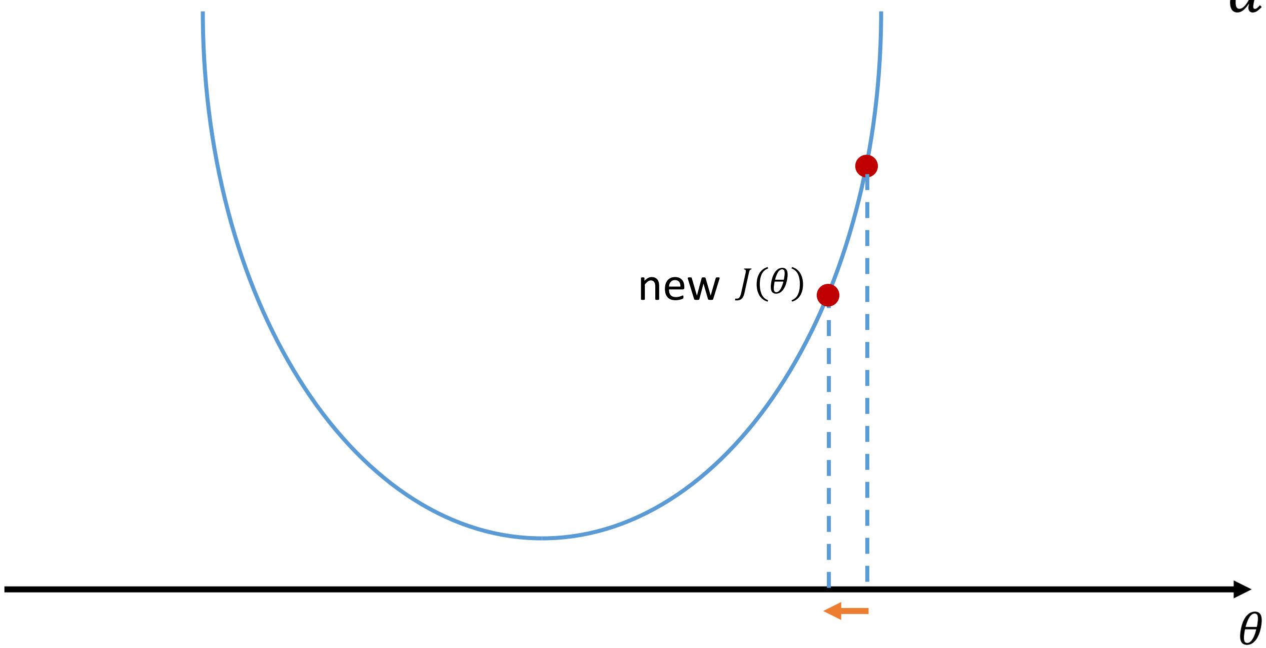
Gradient Descent - MSE

$$\theta := \theta - \alpha \frac{d}{d\theta} J(\theta)$$



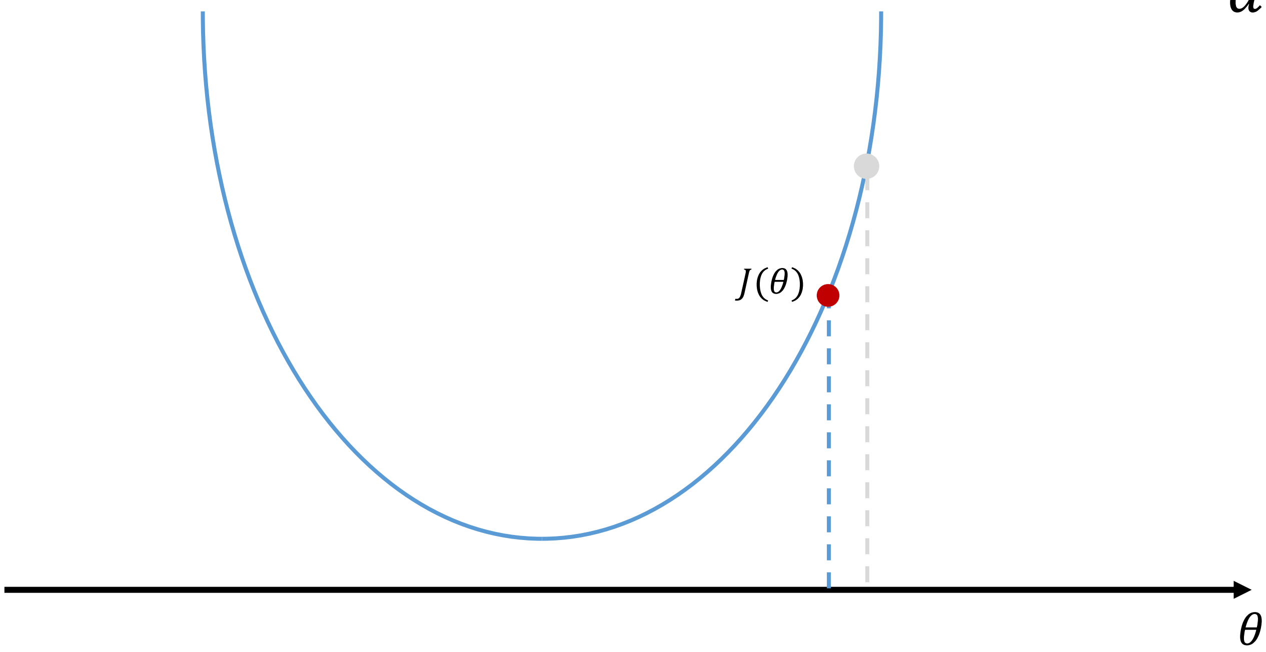
Gradient Descent - MSE

$$\theta := \theta - \alpha \frac{d}{d\theta} J(\theta)$$



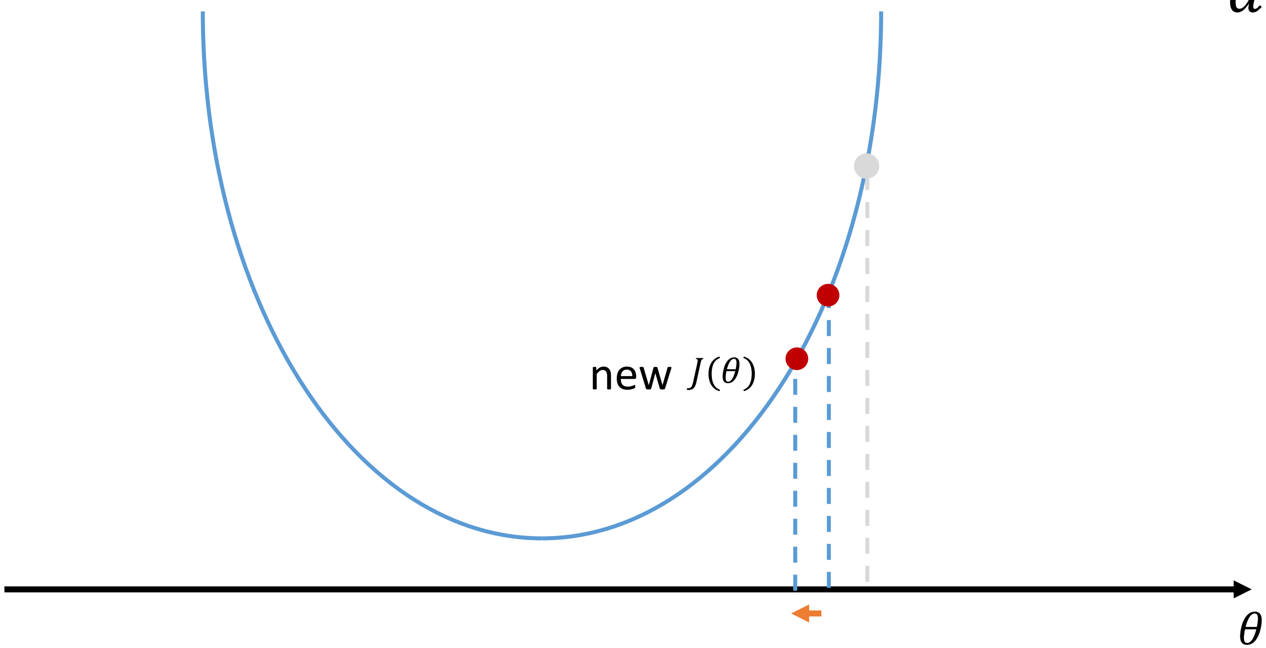
Gradient Descent - MSE

$$\theta := \theta - \alpha \frac{d}{d\theta} J(\theta)$$



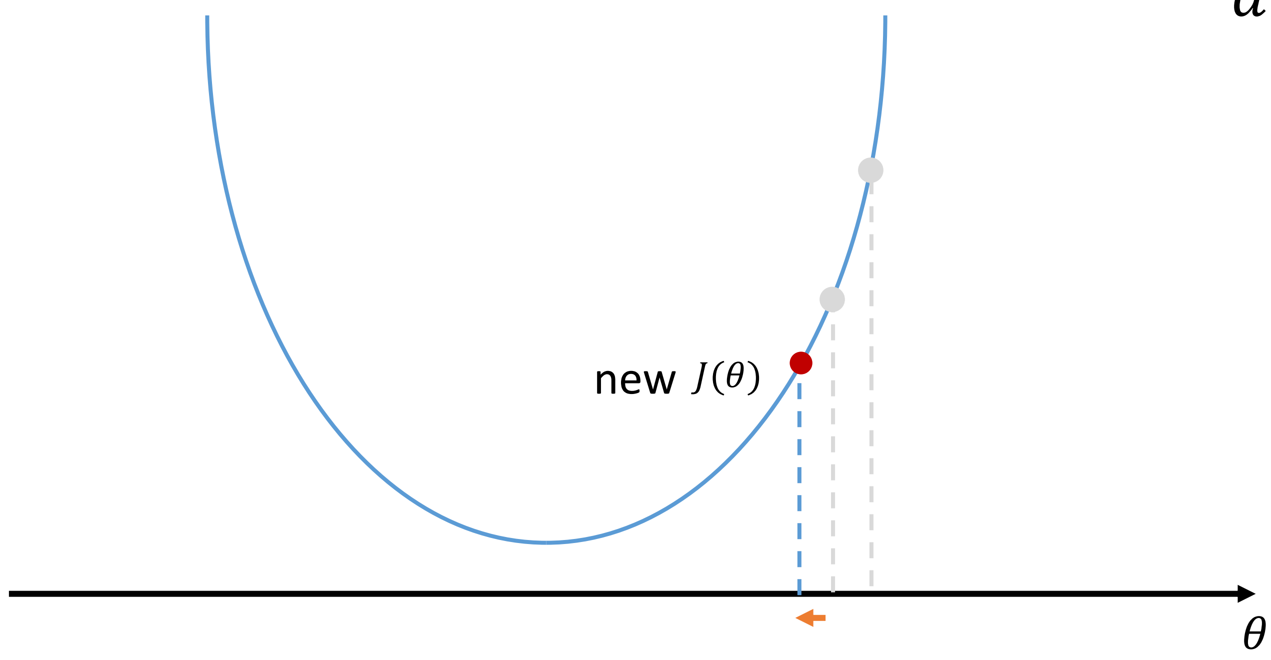
Gradient Descent - MSE

$$\theta := \theta - \alpha \frac{d}{d\theta} J(\theta)$$



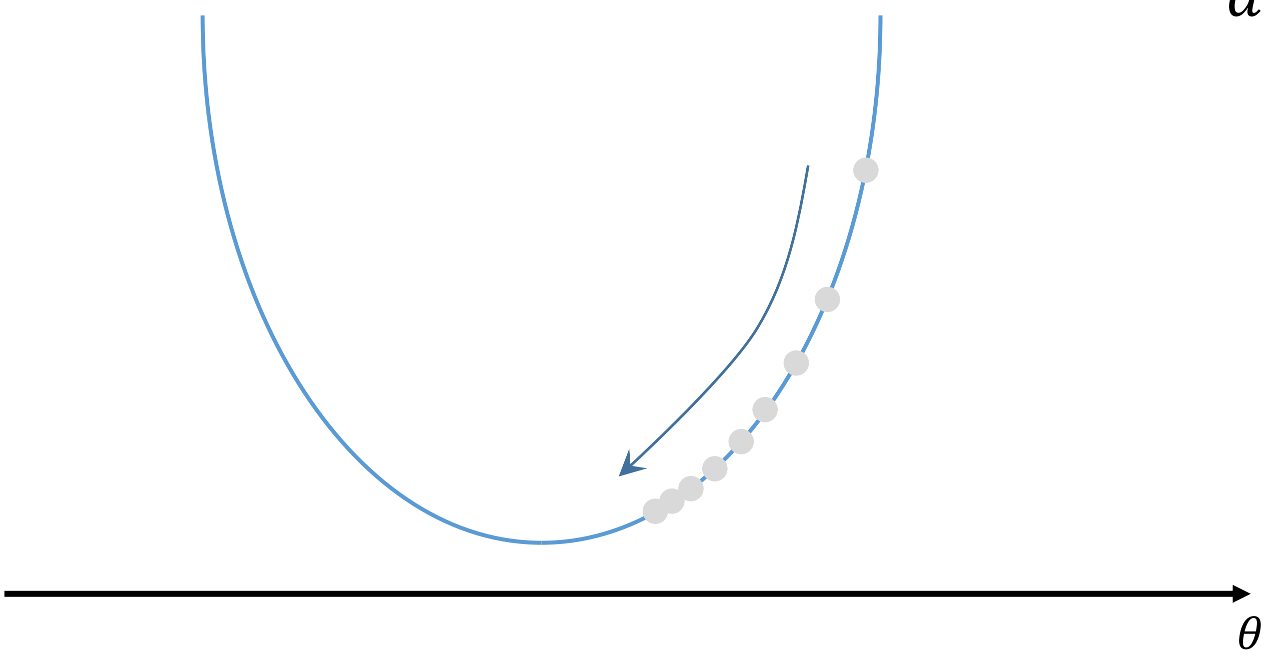
Gradient Descent - MSE

$$\theta := \theta - \alpha \frac{d}{d\theta} J(\theta)$$



Gradient Descent - MSE

$$\theta := \theta - \alpha \frac{d}{d\theta} J(\theta)$$



Notation

- x – input (multivariate data point/ vector)
- d – number of features (dimensions)
- x_i – i^{th} data point
- x_i^k – k^{th} feature of i^{th} data point

Multivariate Linear Function

- Univariate Linear Regression

$$g(x) = \theta^0 + \theta^1 x$$

- Multivariate Linear Regression

$$g(x) = \theta^0 + \theta^1 x^1 + \theta^2 x^2 + \theta^3 x^3 + \ldots + \theta^d x^d$$

- For convenience, define x^0 such that $x^0 = 1$

$$g(x) = \theta^0 x^0 + \theta^1 x^1 + \theta^2 x^2 + \theta^3 x^3 + \ldots + \theta^d x^d$$

Simplification

- $x = \begin{bmatrix} x^0 \\ x^1 \\ x^2 \\ x^3 \\ \vdots \\ x^d \end{bmatrix} \in \mathbb{R}^{d+1}$ $\qquad \theta = \begin{bmatrix} \theta^0 \\ \theta^1 \\ \theta^2 \\ \theta^3 \\ \vdots \\ \theta^d \end{bmatrix} \in \mathbb{R}^{d+1}$

- $g(x) = \theta^0 x^0 + \theta^1 x^1 + \theta^2 x^2 + \theta^3 x^3 + \ldots + \theta^d x^d$
- $g(x) = \theta^T x$

Cost Function

- Cost Function

$$J(\theta^0, \theta^1, \dots, \theta^d) = \frac{1}{2N}\sum_{i=1}^{N}(h(x_i) - y_i)^2$$

$$J(\theta^0, \theta^1, \dots, \theta^d) = \frac{1}{2N}\sum_{i=1}^{N}(\theta^T x_i - y_i)^2$$

Gradient Descent in Linear Reg.

1. $repeat$
2. $\theta^j := \theta^j - \alpha \frac{\partial}{\partial \theta^j} \frac{1}{2N} \sum_{i=1}^{N} (h(x_i) - y_i)^2$
3. $until\ converge$

- j = 0 $\frac{\partial}{\partial \theta_0} J(\theta^0, \theta^1, \dots, \theta^d) = \frac{1}{N} \sum_{i=1}^{N} (h(x_i) - y_i) \,.\, x_i^0$
- j = 1 $\frac{\partial}{\partial \theta_1} J(\theta^0, \theta^1, \dots, \theta^d) = \frac{1}{N} \sum_{i=1}^{N} (h(x_i) - y_i). x_i^1$
- j = d $\frac{\partial}{\partial \theta_d} J(\theta^0, \theta^1, \dots, \theta^d) = \frac{1}{N} \sum_{i=1}^{N} (h(x_i) - y_i). x_i^d$

Gradient Descent in Linear Reg.

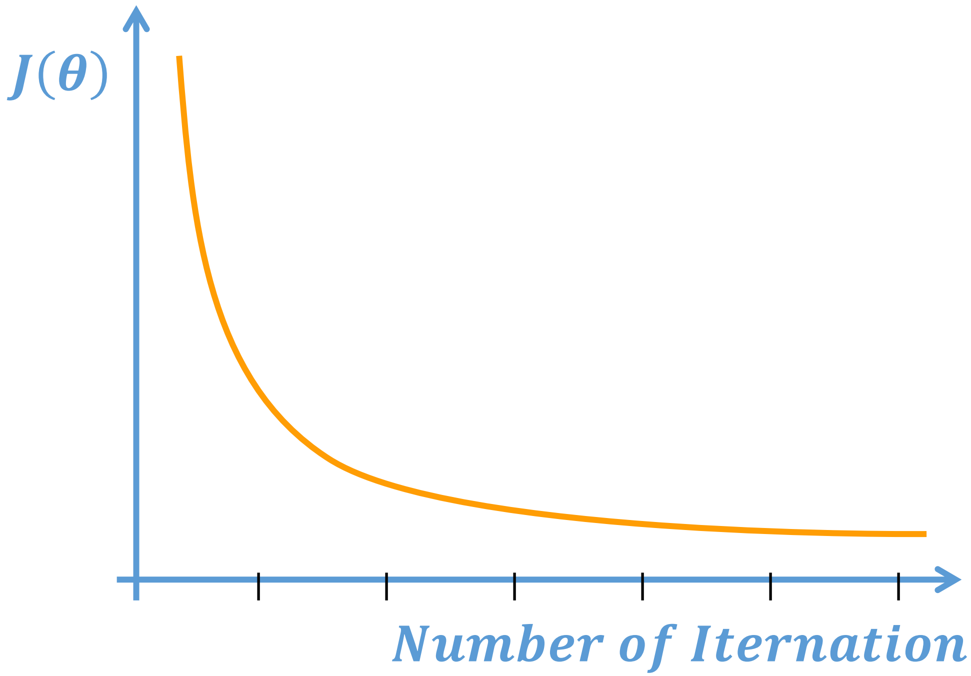
1. $repeat$
2. $$\theta^0 := \theta^0 - \alpha \frac{1}{N}\sum_{i=1}^{N}(h(x_i) - y_i) \,.\, x_i^0$$
3. $$\theta^1 := \theta^1 - \alpha \frac{1}{N}\sum_{i=1}^{N}(h(x_i) - y_i) \,.\, x_i^1$$

   ....

4. $$\theta^d := \theta^d - \alpha \frac{1}{N}\sum_{i=1}^{N}(h(x_i) - y_i) \,.\, x_i^d$$
5. $until\ converge$

Convergence

- Plot cost function and identify error minimized and stabled
- Predefine tolerance level for cost change



Normal Equations

- Gradient Descent – Iterative process.
- Normal Equations – Solve a mathematical equation and finds optimal values for parameters (θ^k).
- Directly finds the value of θ without Gradient Descent iterative process.
- Effective when the data set has less features.

Normal Equations

- Take the partial derivative for each θ^k and equals it to zero.
- Solve the equation and find θ^k for each parameter

Cost Function

- Our cost function

$$J(\theta^0, \theta^1, \dots, \theta^d) = \frac{1}{2N}\sum_{i=1}^{N}(\theta^T x_i - y_i)^2$$

$$J(\theta^0, \theta^1, \dots, \theta^d) = \frac{1}{2N} \|X\theta - y\|^2$$

Cost Function

- Our cost function

$$J(\theta^0, \theta^1, \dots, \theta^d) = \frac{1}{2N}\sum_{i=1}^{N}(\theta^T x_i - y_i)^2$$

$$(\theta^T x_1 - y_1)^2 + (\theta^T x_2 - y_2)^2 + \cdots + (\theta^T x_N - y_N)^2$$

$$(x_1^T \theta - y_1)^2 + (x_2^T \theta - y_2)^2 + \cdots + (x_N^T \theta - y_N)^2$$

Representation

- If we consider matrix X and vector θ,

$$\begin{bmatrix} \dots. & x_1^T & \dots. \\ \dots. & x_2^T & \dots. \\ & \vdots & \\ \dots. & x_N^T & \dots. \end{bmatrix} \begin{bmatrix} \theta^1 \\ \theta^2 \\ \vdots \\ \theta^d \end{bmatrix}$$

- $X \ - N \times d$
- $\theta \ - d \times 1$

Representation

- If we consider two matrix X and θ,

$$\begin{bmatrix} \dots. & x_1^T & \dots. \\ \dots. & x_2^T & \dots. \\ & \vdots & \\ \dots. & x_N^T & \dots. \end{bmatrix} \begin{bmatrix} \theta^1 \\ \theta^2 \\ \vdots \\ \theta^d \end{bmatrix} = \begin{bmatrix} x_1^T\theta \\ x_2^T\theta \\ \vdots \\ x_N^T\theta \end{bmatrix}$$

- $X \ - N\times d$
- $\theta \ - d\times 1$
- $Output - \ N\times 1$

Representation

$$\begin{bmatrix} \dots & x_1^T & \dots \\ \dots & x_2^T & \dots \\ & \vdots & \\ \dots & x_N^T & \dots \end{bmatrix} \begin{bmatrix} \theta^1 \\ \theta^2 \\ \vdots \\ \theta^d \end{bmatrix} = \begin{bmatrix} x_1^T\theta \\ x_2^T\theta \\ \vdots \\ x_N^T\theta \end{bmatrix}$$

- Now the error,

$$\begin{bmatrix} x_1^T\theta \\ x_2^T\theta \\ \vdots \\ x_N^T\theta \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_1^T\theta - y_1 \\ x_2^T\theta - y_2 \\ \vdots \\ x_N^T\theta - y_N \end{bmatrix}$$

$$= X\theta - y$$

Representation

- What we want is,

$$(x_1^T\theta - y_1)^2 + (x_2^T\theta - y_2)^2 + \cdots + (x_N^T\theta - y_N)^2$$

- What we have is,

$$X\theta - y = \begin{bmatrix} x_1^T\theta - y_1 \\ x_2^T\theta - y_2 \\ \vdots \\ x_N^T\theta - y_N \end{bmatrix}$$

$$\|X\theta - y\| = \sqrt{(x_1^T\theta - y_1)^2 + (x_2^T\theta - y_2)^2 + \cdots + (x_N^T\theta - y_N)^2}$$

Representation

- So, if we take norm squared,

$$\|X\theta - y\|^2 = (x_1^T\theta - y_1)^2 + (x_2^T\theta - y_2)^2 + \cdots + (x_N^T\theta - y_N)^2$$

- Therefore,

$$J(\theta^0, \theta^1, \dots, \theta^d) = \frac{1}{2N}\|X\theta - y\|^2$$

Minimizing $J(\theta^0, \theta^1, \dots, \theta^d)$

$$J(\theta^0, \theta^1, \dots, \theta^d) = \frac{1}{2N} \|X\theta - y\|^2$$

- Take the derivative, and make it equal to zero

$$\nabla J(\theta) = \frac{1}{N} X^T (X\theta - y)$$

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

$$X^T (X\theta - y) = 0$$

$$X^T X\theta = X^T y$$

Solving the equation

- Solve this to find θ

$$X^T X\theta = X^T y$$

- Solve this

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

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

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

$I - Identity\ Matrix$

Solving the equation

- Solve this to find θ

$$X^T X\theta = X^T y$$

- Solve this

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

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

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

$$(X^T X)^{-1} X^T\ \ - pseudo\ inverse\ of\ X$$

Normal Equation Method

- Construct the matrix X from the sample data
- Construct the vector y from the available labels
- Compute the $pseudo\ inverse$ of X
- Compute θ by

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

Gradient Descent Vs Normal Eq.

- Learning rate in Gradient Descent
- Gradient Descent requires a number of iterations
- Matrix inversion calculation is a computationally intensive task, so NE method may be computationally inefficient if the number of features relatively large.
- NE method can be used to initialize parameters for classification counter part of linear problems.

Logistic Regression

Linear Regression for Classification

- In our linear regression model, the hypothesis is,

$$g(x) = \theta^T x$$

- It gives real valued output for any given input.
- Linear regression doesn't suit with classification problems.

Classification

- What we want is a hypothesis that restricts the output.

$$0 \le h_\theta(x) \le 1$$

  - $0\ - Negative\ Class$
  - $1 - Positive\ Class$

Logistic Regression

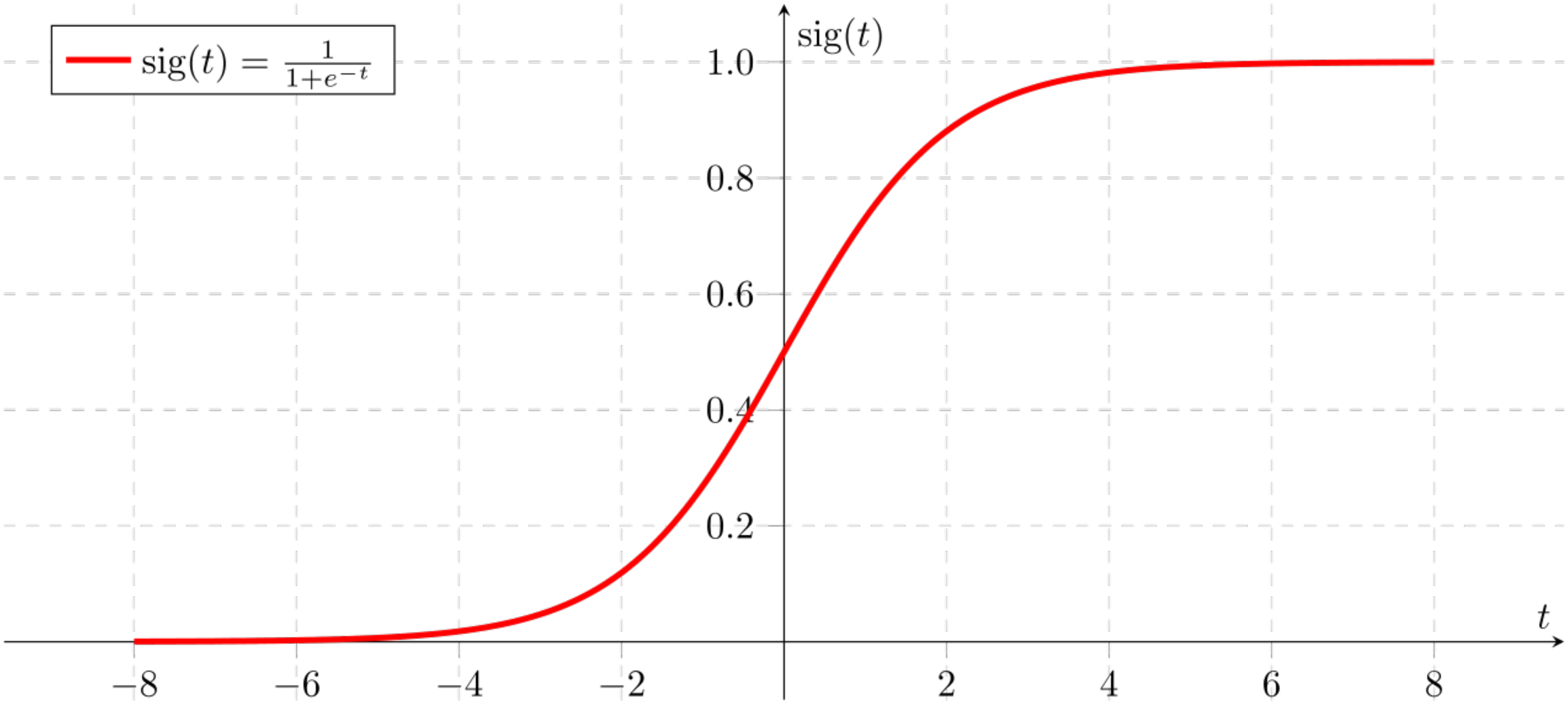
- Uses the linear regression hypothesis $h(x)$
- Apply restrictive function $g(x)$ on $h(x)$
- It will be the classification hypothesis $k(x)$

$$k(x) = g(h(x))$$

- We use

$$g - Logistic\ Function$$

Logistic (Sigmoid) Function



Logistic Function

- Also known as Sigmoid Function

$$Sig(x) = \frac{1}{1 - e^{-x}}$$

- In Linear Regression

$$g_\theta(x) = \theta^T x$$

- Therefore, our sigmoid function becomes,

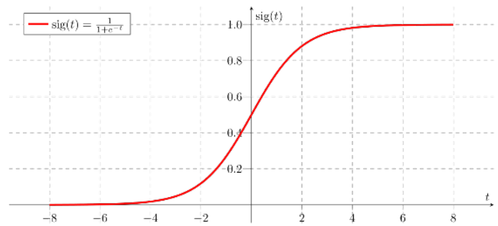
$$s_\theta(x) = \frac{1}{1 - e^{-\theta^T x}}$$

- Learning Task - Fit parameters θ, to this function

Logistic Function

- Always provides output $0 \leq s_\theta(x) \leq 1$
- If $s_\theta(x) = 0.82$
  - Can use a threshold 0.5 and declare it is class 1
  - Can state probability of being class 1 is 82%
  - Probability of being class 0 is 18%

Decision Boundary



- Let's assume Logistic Regression hypothesis $h_\theta(x)$

$$h_\theta(x) = s\big(g_\theta(x)\big)$$

- Where $g_\theta(x)$ is the linear regression hypothesis

$$g_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

- According to the Sigmoid function
  - Output = 1 when $\theta_0 + \theta_1 x_1 + \theta_2 x_2 \; > 0$
  - Output = 0 when $\theta_0 + \theta_1 x_1 + \theta_2 x_2 \; < 0$

$\theta_0 + \theta_1 x_1 + \theta_2 x_2 \; = 0$ is the decision boundary

Parameter Fitting – Cost Function

- In linear regression

$$J(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(h(x_i) - y_i)^2$$

- If the same cost function is used, gradient descent tend to end up with local minimum.
- Mean squared error function is not a convex function in this problem.

Cost Function – Logistic Regression

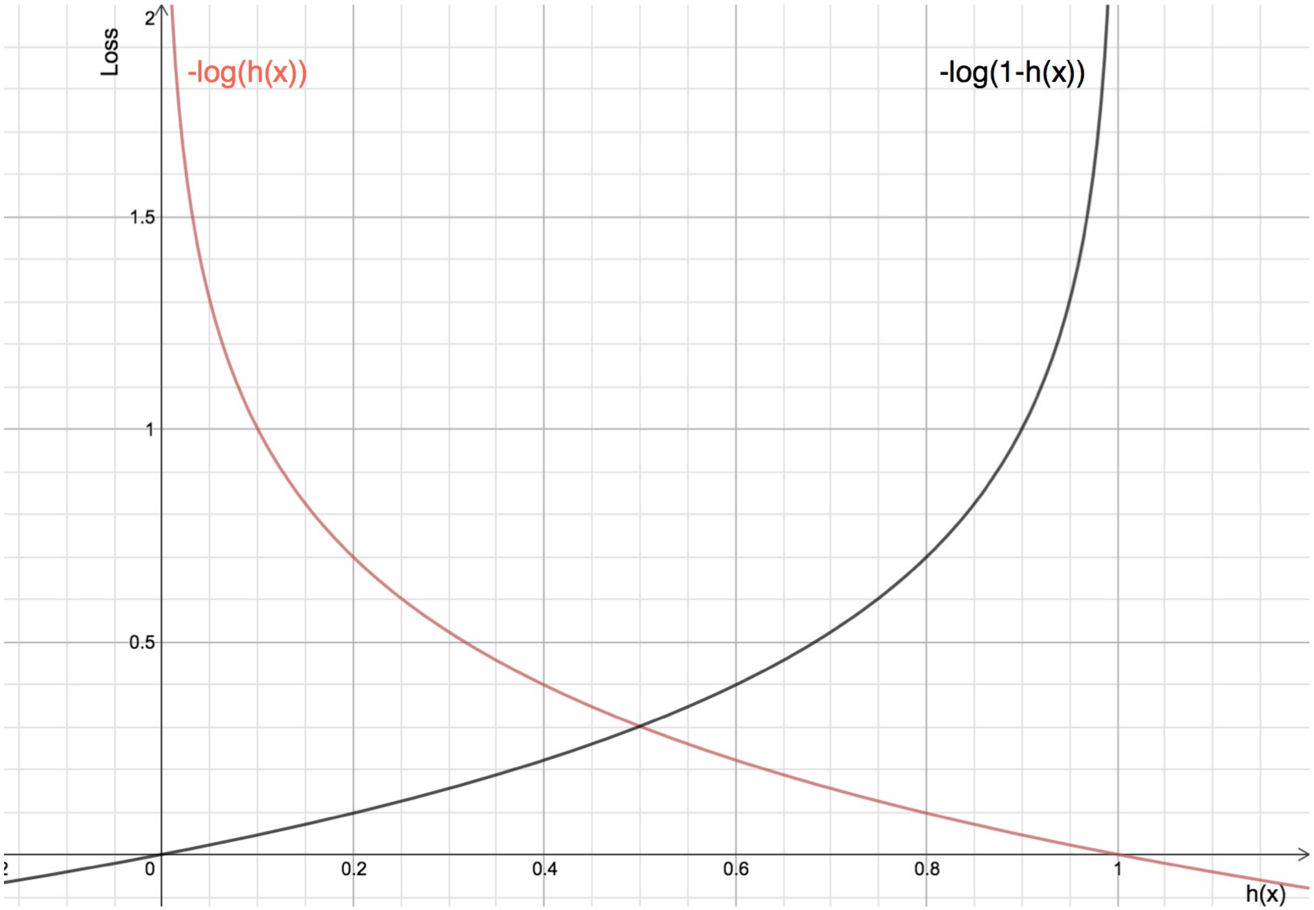
- Cost function

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} Cost(h_\theta(x_i), y_i)$$

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & y = 1 \\ -\log(1 - h_\theta(x)), & y = 0 \end{cases}$$

$$Cost(h_\theta(x), y) = -y\log\big(h_\theta(x)\big) - (1-y)\log(1 - h_\theta(x))$$

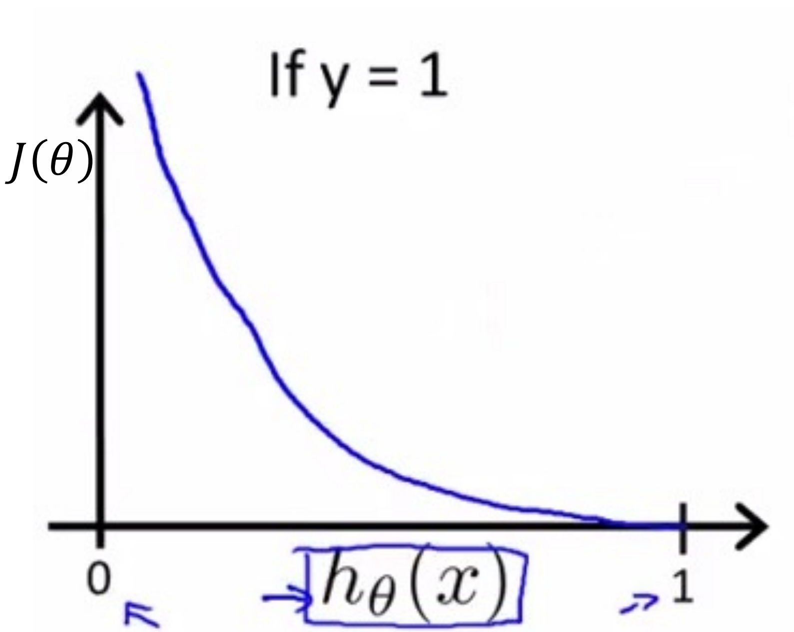
Behaviour of Log Functions

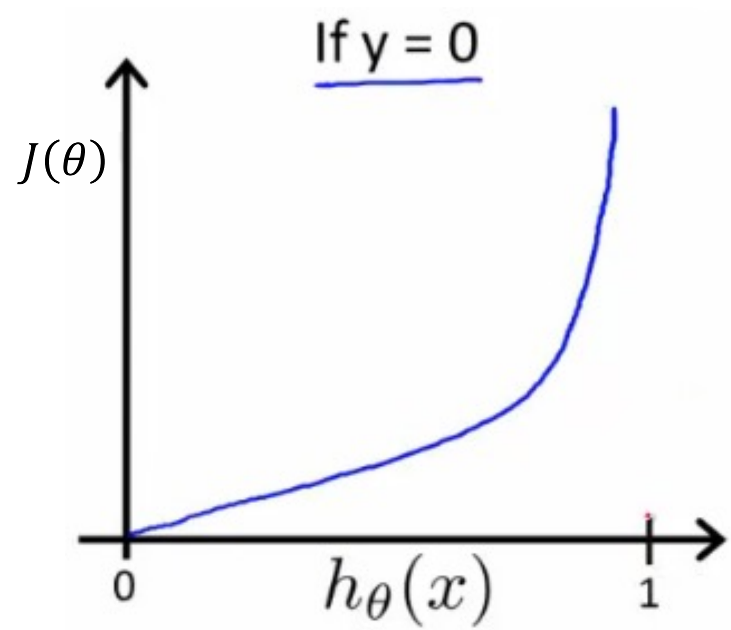


https://www.analyticsvidhya.com/blog/2020/11/binary-cross-entropy-aka-log-loss-the-cost-function-used-in-logistic-regression/

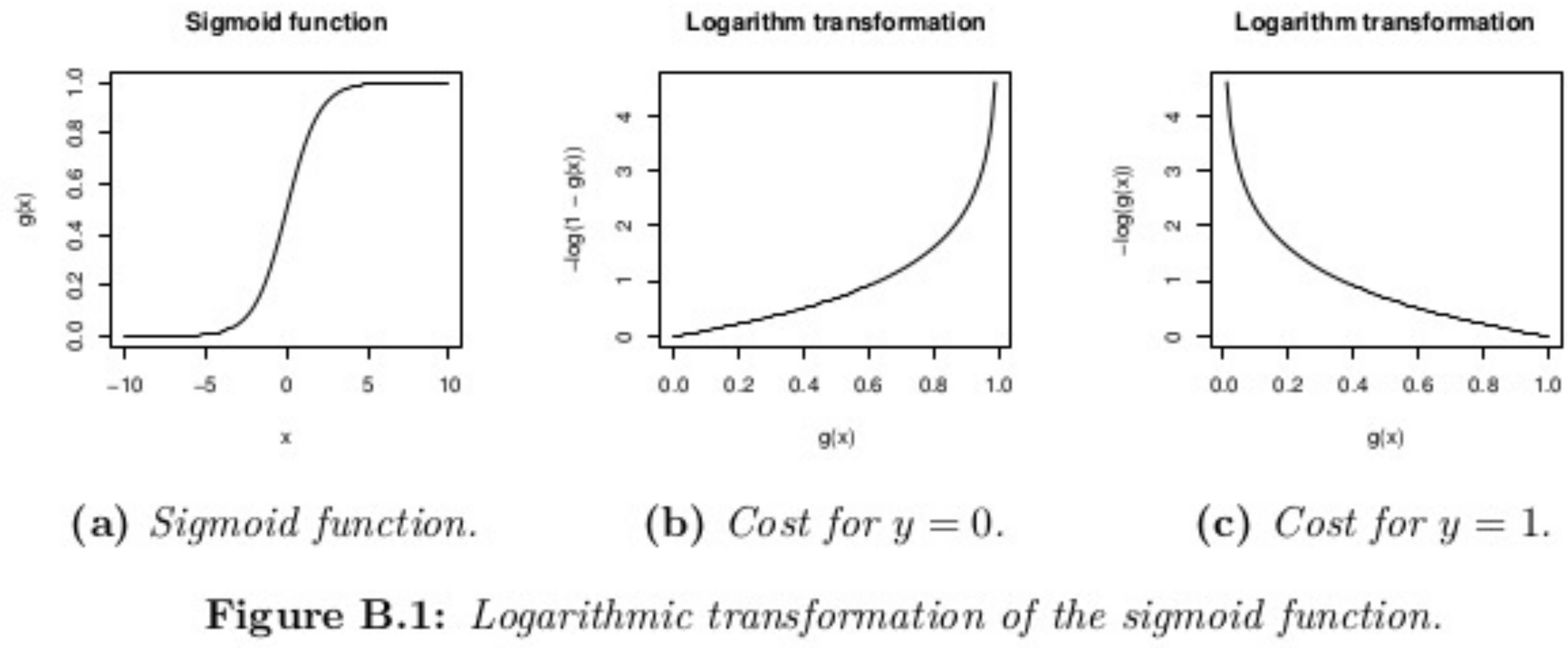
Cost Function – Logistic Regression

- $J(\theta)$ $- \ Cost\ Function$
- $h_\theta(\theta)$ $- \ Hypothesis\ of\ Logistic\ Regression$





Cost Function – Logistic Regression



Cost Function – Logistic Regression

- Cost function

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} Cost(h_\theta(x_i), y_i)$$

$$Cost(h_\theta(x), y) = -y\log\big(h_\theta(x)\big) - (1-y)\log(1-h_\theta(x))$$

$$J(\theta) = -\frac{1}{N}\left[\sum_{i=1}^{N} y_i \log\big(h_\theta(x_i)\big) + (1-y_i)\log(1-h_\theta(x_i))\right]$$

Derivative of the Cost Function

- The cost function $J(\theta)$

$$J(\theta) = -\frac{1}{N}\left[\sum_{i=1}^{N} y_i \log\big(h_\theta(x_i)\big) + (1 - y_i)\log(1 - h_\theta(x_i))\right]$$

- The derivative of $J(\theta)$ is,

$$\frac{\partial}{\partial \theta^j} J(\theta) = \frac{1}{N}\sum_{i=1}^{N}(h_\theta(x_i) - y_i).x_i^j$$

Derivation of the derivative - https://medium.com/analytics-vidhya/derivative-of-log-loss-function-for-logistic-regression-9b832f025c2d

Gradient Descent

1. $repeat$
2. $\theta^j := \theta^j - \alpha \frac{\partial}{\partial \theta^j} J(\theta)$
3. $until\ converge$

- For any j

$$\frac{\partial}{\partial \theta^j} J(\theta) = \frac{1}{N} \sum_{i=1}^{N} (h_\theta(x_i) - y_i).\, x_i^j$$

- Where $h_\theta(x_i) = \frac{1}{1 - e^{-\theta^T x}}$

Parameters θ

- To minimize the cost to fit the parameters θ,

$$min \; \mathrm{J}(\theta)$$

- Finally find the $g(x)$ out of set of $h(x)$ that gives the minimum $J(\theta)$,

$$g_\theta(x) = \frac{1}{1 - e^{-\theta^T x}}$$

Read

- https://www.internalpointers.com/post/cost-function-logistic-regression
- https://www.analyticsvidhya.com/blog/2020/11/binary-cross-entropy-aka-log-loss-the-cost-function-used-in-logistic-regression/

Assignment - 1

Task

- Pick a dataset from UCI Machine Learning Repository
- Do Multivariate Linear Regression with Colab
- Submit your Colab Notebook

Q & A

Thank you..!