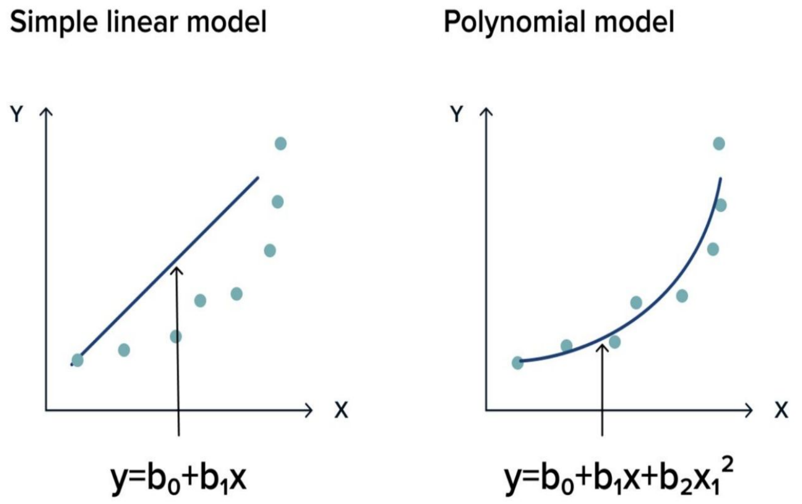
Tutorial 1

Linear Regression

Regression

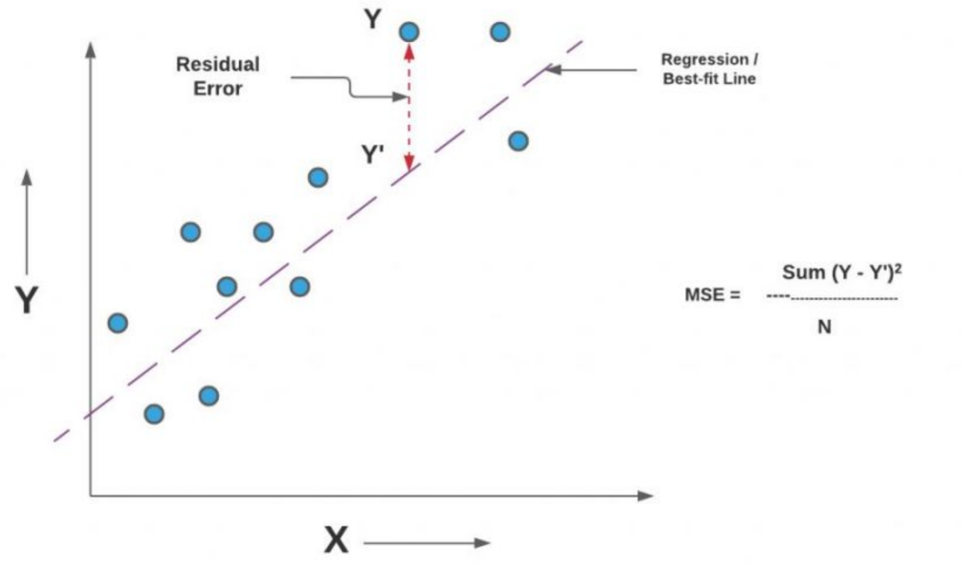
- Supervised Learning Algorithm - given pairs of $(X^{(i)},Y^{(i)})$, find the line that best fits the given data.
- Types -
  - Based on number of variables -
    - **Univariate : $y_{pred} = b_0 + b_1x$**
    - **Multivariate : $y_{pred} = b_0 + b_1x_1 + b_2x_2$**
  - Based on degree of the function -
    - **Linear : $y_{pred} = b_0 + b_1x$**
    - **Polynomial : $y_{pred} = b_0 + b_1x + b_2x^2$**



Mean Squared Error (MSE) - Loss function

Sum of squared error between predicted and actual target values

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$



Univariate Linear Regression

- $y_{pred} = b_0 + b_1x$
- Given $D = \{(x_1,y_1),(x_2,y_2),...(x_n,y_n)\}$
- Find optimal values of b_0 and b_1 that best fits the given data D
- The loss at each value of $x^{(i)}$ is : $(y^{(i)}-y^{(i)}_{pred})^2$
- Total loss for all the instances is : $(1/n)\Sigma(y^{(i)}-y^{(i)}_{pred})^2$
- We need to minimize this loss function to get the optimal values

Finding the Optimal Parameters for Univariate Linear Regression

Minimize the loss function,

$$J(b_0, b_1) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - y^{(i)}_{pred}\right)^2$$

$$= \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - b_0 - b_1 x^{(i)}\right)^2$$

To minimize this, set partial derivatives to 0,

$$\frac{\partial J(b_0, b_1)}{\partial b_0} = 0 \quad \text{and} \quad \frac{\partial J(b_0, b_1)}{\partial b_1} = 0$$

$$\Rightarrow \frac{1}{2n}\cdot 2\sum\left(y^{(i)} - b_0 - b_1 x^{(i)}\right)\cdot(-1) = 0$$

$$\Rightarrow \sum y^{(i)} - b_0 \sum 1 - b_1 \sum x^{(i)} = 0$$

$$\Rightarrow \sum y^{(i)} - n b_0 - b_1 \sum x^{(i)} = 0$$

$$\Rightarrow b_0 = \frac{\sum y^{(i)} - b_1 \sum x^{(i)}}{n} \rightarrow (1)$$

$$\Rightarrow \frac{1}{2n}\cdot 2\sum\left(y^{(i)} - b_0 - b_1 x^{(i)}\right)\cdot\left(-x^{(i)}\right) = 0$$

$$\Rightarrow \sum x^{(i)} y^{(i)} - b_0 \sum x^{(i)} - b_1 \sum x^{(i)2} = 0$$

Putting value of b_0 here,

$$b_1 = \frac{n\sum x^{(i)} y^{(i)} - \sum x^{(i)} \sum y^{(i)}}{n\sum x^{(i)2} - \left(\sum x^{(i)}\right)^2} \rightarrow (2)$$

Try the same for bivariate case where we have to find b_0, b_1 and b_2.

What would you do for multivariate case with n variables and n+1 parameters?

Closed Form Solution for N variables

- For n variables we would have to solve n+1 such simultaneous equations which is not feasible
- Use the closed form solution which finds the optimal parameters using matrix algebra
- A set of linear simultaneous equations can be represented as a matrix product solution

$$
\begin{aligned}
b_0 + b_1 x_1^{(1)} + b_2 x_2^{(1)} + \cdots b_n x_n^{(1)} &= y^{(1)} \\
b_0 + b_1 x_1^{(2)} + b_2 x_2^{(2)} + \cdots + b_n x_n^{(2)} &= y^{(2)} \\
\vdots \qquad \vdots \qquad & \vdots \qquad \vdots \\
b_0 + b_1 x_1^{(m)} + b_2 x_2^{(m)} + \cdots + b_n x_n^{(m)} &= y^{(m)}
\end{aligned}
$$

This can be represented in matrix form

$$
\Rightarrow
\begin{bmatrix}
1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\
1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\
\vdots & \vdots & & & \vdots \\
1 & x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)}
\end{bmatrix}
\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}
=
\begin{bmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}
$$

Thus, the system of linear equations reduces to -

$$XB = Y \quad \text{where } X \in R^{m \times (n+1)},\ B \in R^{n+1},\ Y \in R^{m}$$

- We can find the approximate value of **B** by using matrix inverse property

So we can find B using matrix inverse

i.e. $B = X^{-1}Y$

But X is often a non-square matrix whose inverse doesn't exist!

So, we use the pseudo inverse of X (see references)

$\therefore$ $B = X^{*}Y$ where X^{*} is the pseudoinverse

X^{*} is defined as –

i) $X^{*} = (X^TX)^{-1}X^T$ if $m > n$ (more equations than unknowns)

ii) $X^{*} = X^T(XX^T)^{-1}$ if $m < n$ (more unknowns than equations)

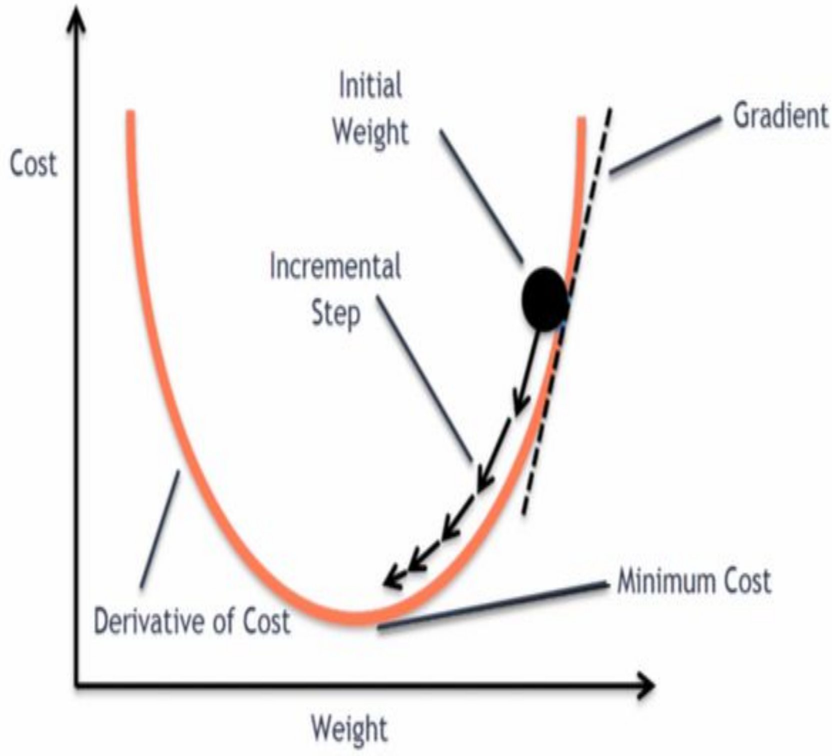
iii) $X^{*} = X^{-1}$ if $m = n$ (Same no of unknowns and equations)

In most cases we have $m > n$

$\therefore$ $B = (X^TX)^{-1}X^TY$ is the closed form sol^n for multivariate linear regression

Solving optimal parameters using Gradient Descent

- For large datasets containing thousands of rows and columns, finding the inverse of such massive matrices is not feasible as it requires huge amount of memory
- Solution - use iterative gradient descent algorithm
- The loss function is convex and we need to find its global minima
- We compute the gradients at each iteration and go down the hill using the direction of the gradients till we reach the lowest point on the curve



Explanation

- Consider a loss function which only depends on one parameter
- We start with random initialization of parameter b and we land on some point on the loss function
- Calculate the gradient (slope in case of a single parameter)
- If slope is +ve we need to decrease the value of b (i.e. go left)
- If slope is -ve we need to increase the value of b (i.e. go right)
- Thus we need to go in the direction opposite to the slope in small steps
- When we reach the minima, slope is 0 and we don't need to move any further
- The update rule at step t is -

$$b_t = b_{t-1} - \alpha *(dJ(b)/db)$$

where α is the learning rate often in range of (0,1)

Gradient Descent for N variables

- For n variables, we have n+1 parameters and the loss surface is n+1 dimensional paraboloid
- We need to update simultaneously all the parameters to go downhill in all directions

$b_{0,t} = b_{0,t-1} - \alpha *(\delta J(B)/\delta b_0)$

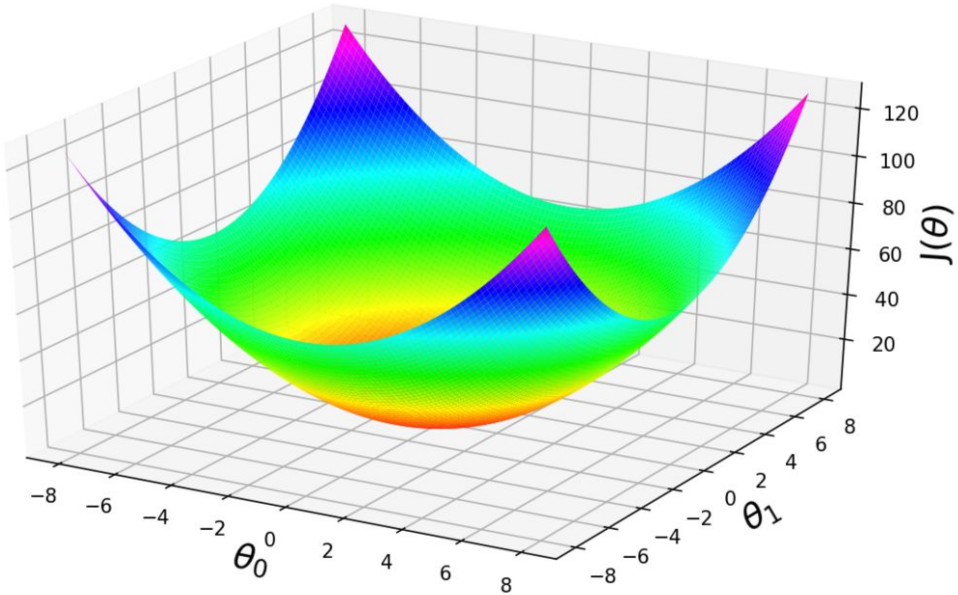
$b_{1,t} = b_{1,t-1} - \alpha *(\delta J(B)/\delta b_1)$

.

.

.

$b_{n,t} = b_{n,t-1} - \alpha *(\delta J(B)/\delta b_n)$



Probabilistic Interpretation of Linear Regression

- The predicted value of $y^{(i)}$, given an $x^{(i)}$ and B is actually sampled from a normal distribution

The actual data contains some error 'ϵ'

Thus, $y^{(i)} = B^T x^{(i)} + \epsilon^{(i)}$ where $B = [b_0 \ b_1 \cdots b_n]$

$\Rightarrow \epsilon^{(i)} = y^{(i)} - B^T x^{(i)} \longrightarrow ①$ $\quad x^{(i)} = [1 \ x_1^{(i)} \ x_2^{(i)} \cdots x_n^{(i)}]$

Linear Regression assumes that $\epsilon^{(i)} \sim N(0, \sigma^2)$

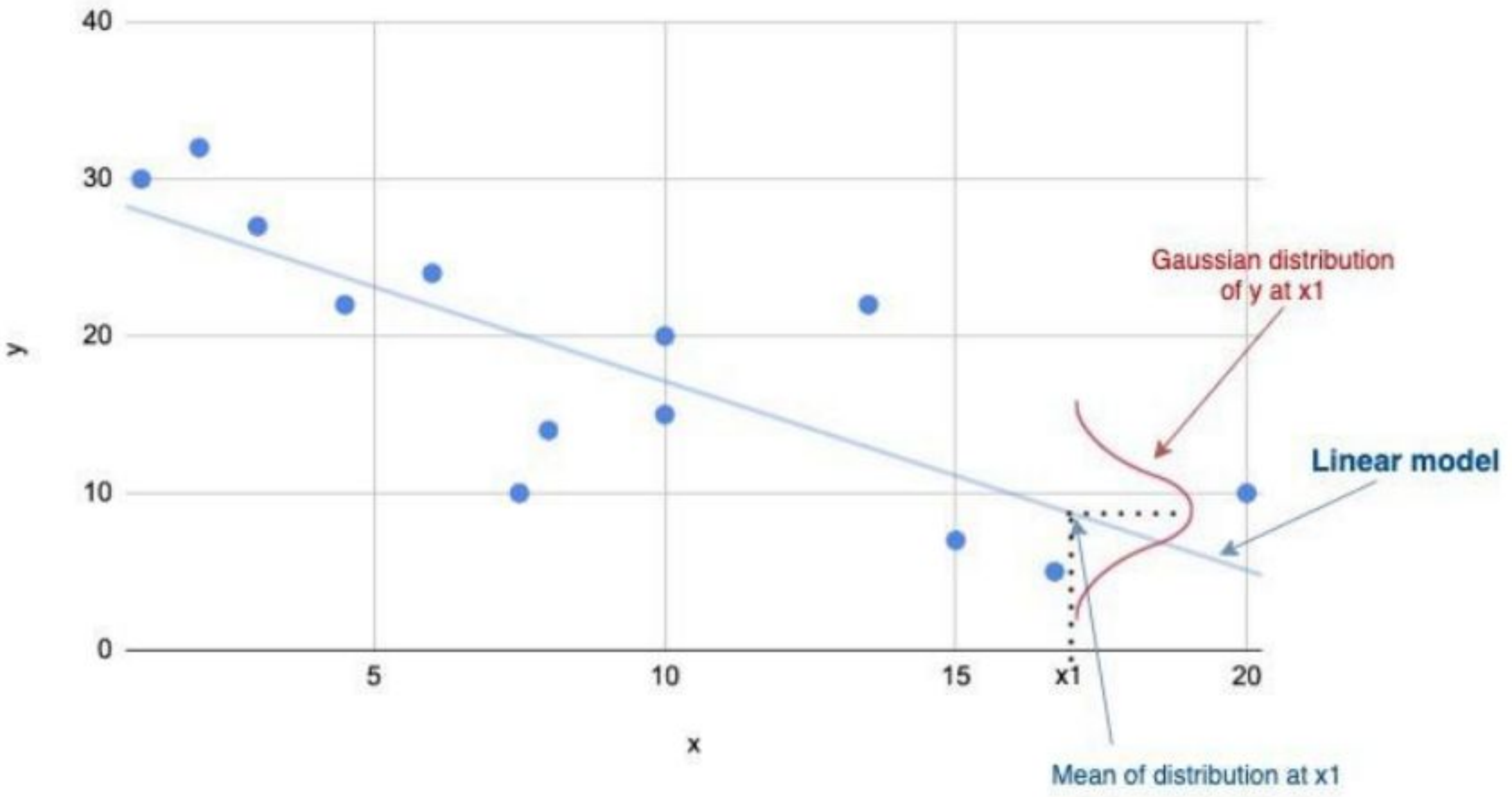
i.e. error is sampled from a normal distribution with 0 mean and variance σ^2

$$\therefore P(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{1}{2}\left(\frac{\epsilon^{(i)}}{\sigma}\right)^2\right)}$$

$$= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{y^{(i)} - B^T x^{(i)}}{\sigma}\right)^2}$$

This is equivalent to $y^{(i)}$ being sampled from a normal distribution with mean $\mu = B^T x^{(i)}$ and variance σ^2

$$\therefore P(y^{(i)} | x^{(i)}; B) \sim N(B^T x^{(i)}, \sigma^2)$$


Gaussian distribution
of y at x1
Linear model
Mean of distribution at x1
40
30
20
10
0
y
5
10
15
x1
20
x

Maximum Likelihood Estimation for Linear Regression

- In probabilistic interpretation of linear regression, we need to maximize the probability of each output $y^{(i)}$ being correctly generated given input $x^{(i)}$ and parameters B. This is called maximum likelihood estimation.
- MLE can be proved to be equivalent to minimizing the Loss Function J(B)

Probability of $y^{(i)}$ being generated b the linear model is

$$P(y^{(i)}) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{1}{2}\left(\frac{y^{(i)} - B^T x^{(i)}}{\sigma}\right)^2}$$

We need to maximize this probability for all the outputs $[y^{(0)}, y^{(1)}, \dots y^{(m)}]$

ie. $\underset{B}{\operatorname{argmax}} \prod_{i=1}^{m} P(y^{(i)})$ which is the likelihood function $L(B)$

$$\therefore \; L(B) = \underset{B}{\operatorname{argmax}} \prod_{i=1}^{m} P(y^{(i)} \mid x^{(i)}; B)$$

Taking log on both sides, we get the log likelihood function

$$\log(L(\beta)) = \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{m} \log\left(P(y^{(i)} \mid x^{(i)}; \beta)\right)$$

$$= \underset{\beta}{\operatorname{argmax}} \sum_{i=1}^{m} \log\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{y^{(i)} - \beta^T x^{(i)}}{\sigma}\right)^2}\right)$$

$$= \underset{\beta}{\operatorname{argmax}} \log\frac{1}{\sqrt{2\pi}\sigma} + \sum_{i=1}^{m} \log e^{-\frac{1}{2}\left(\frac{y^{(i)} - \beta^T x^{(i)}}{\sigma}\right)^2}$$

$$= \underset{\beta}{\operatorname{argmax}} \log\frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2}\sum_{i=1}^{m} (y^{(i)} - \beta^T x^{(i)})^2$$

$$= \underset{\beta}{\operatorname{argmax}} -\frac{1}{2}\sum_{i=1}^{m} (y^{(i)} - \beta^T x^{(i)})^2 \quad \left[\text{ignoring the constant} - \log\left(\frac{1}{\sqrt{2\pi}\sigma}\right)\right]$$

$$= \underset{\beta}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{m} (y^{(i)} - \beta^T x^{(i)})^2$$

$$= \underset{\beta}{\operatorname{argmin}} J(\beta)$$

Thus maximising the log likelihood is equivalent to minimizing the loss function. This is true for all other models too like Logistic regression.

Maximum A Posteriori Interpretation of Linear Regression

References:

http://seismo.berkeley.edu/~kirchner/eps_120/Toolkits/Toolkit_10.pdf

https://towardsai.net/p/machine-learning/linear-regression-complete-derivation-with-mathematics-explained

https://ocw.mit.edu/courses/18-05-introduction-to-probability-and-statistics-spring-2014/4a8de32565ebdefbb7963b4ebda904b2_MIT18_05S14_Reading10b.pdf

https://bookdown.org/peter_neal/math4081_notes/MLE.html