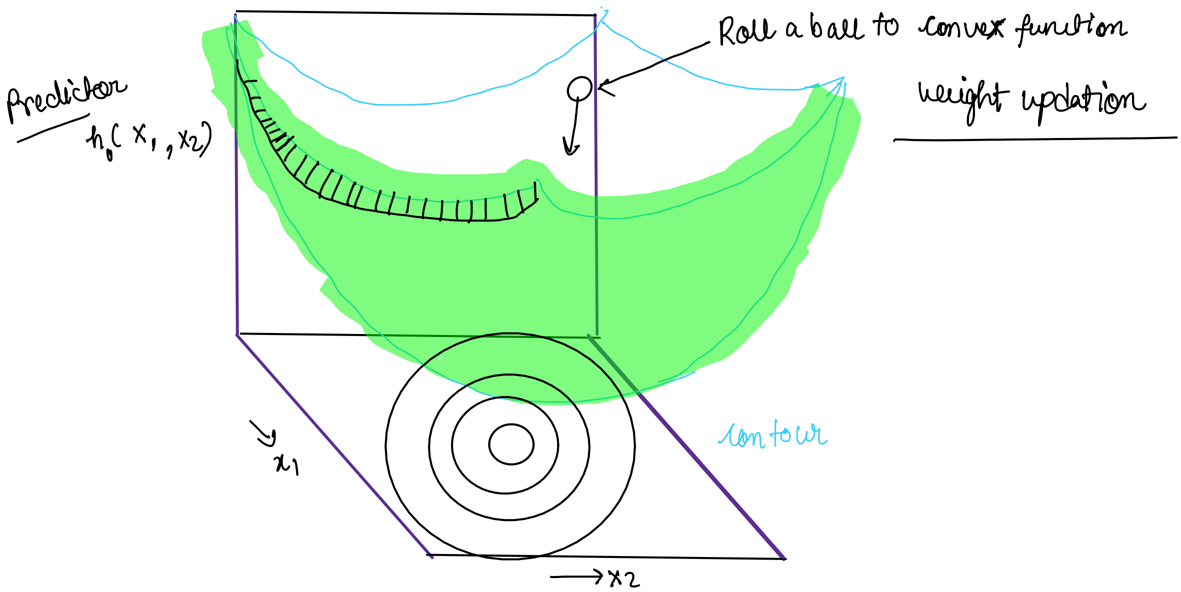
Gradient Descent algorithm as optimizer

Monday, 11 March 2024   10:50 AM

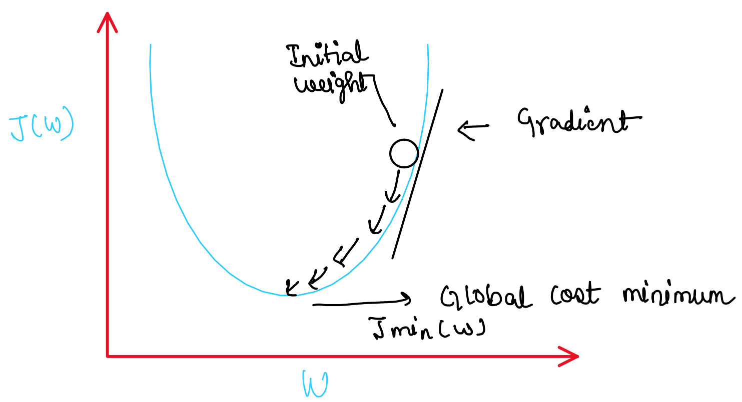
We have studied Normal Equation and its matrix form this is analytical method, which is good when the number of features X are not large

for large number of features, we need the algorithmic tool based on Gradient Descent Algorithm, it is called optimizer.

When we apply on a convex quadratic functions, the global optimum solution is guaranteed



Example of a convex function



$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_w(x_i^{(i)}) - y^{(i)}\right)^2$$

for $i=1 \rightarrow J(w) = \frac{1}{2}\left(h_w(x_i) - y\right)^2 \rightarrow (1)$

for n no. of features

$$h_w(x)_{j=1,2,\dots,n} = W_0 X_0 + W_1 X_1 + W_2 X_2 + \dots + W_n X_n$$

$$= w_j^T x_j \qquad w_j = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \in R^{n+1} \quad \& \quad x_j = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in R^{n+1} \rightarrow (2)$$

for optimization, we need to find

$$\frac{\partial J(w)}{\partial w_j}$$

Now let us substitute $S = h_w(x_j) - Y \rightarrow (3)$

for (2) $\rightarrow S = w_j^T x_j - Y$

$$\boxed{\frac{\partial S}{\partial w_j} = x_j \rightarrow (4)}$$

from (1) & (3) $\rightarrow J(w) = \frac{1}{2} S^2$

Hence, $\frac{\partial J(w)}{\partial S} = \frac{1}{2} * 2 * S = S$ — (5)

$$\frac{\partial J(w)}{\partial w_j} = \frac{\partial J(w)}{\partial S} \cdot \frac{\partial S}{\partial w_j} = S . x_j \text{ (from (4) and (5))}$$

$$= (h_w(x_j) - Y) . x_j$$

for 'm' samples $\boxed{\frac{\partial J(w)}{\partial w_j} = \sum_{i=1}^{m} (h_w(x_j^{(i)}) - Y^{(i)}) . X_j^{(i)}}$

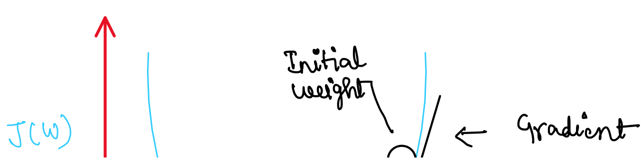
(i) ← feature size
(i) = 0, 1, 2, -- n
(j) ⇐ no. of samples

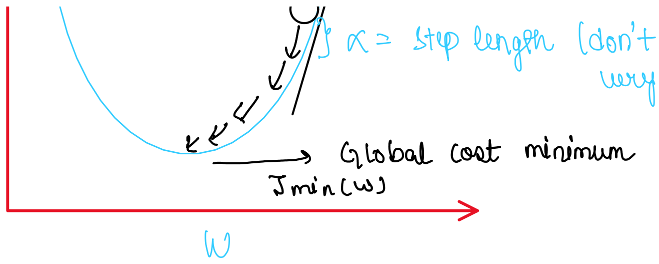
for Gradient Descent Algorithm

$$\boxed{w_j = w_j - \alpha . \frac{1}{m} \frac{\partial J(w)}{\partial w_j}}$$

hyper parameter
α = learning Rate
plays a important Role
how model parameter
will converge

$$\boxed{w_j = w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_w(x_j)^{(i)} - Y^{(i)}) . X_j^{(i)}}$$





α = Step length (don't take very high because if it is very high it will miss Global cost minima

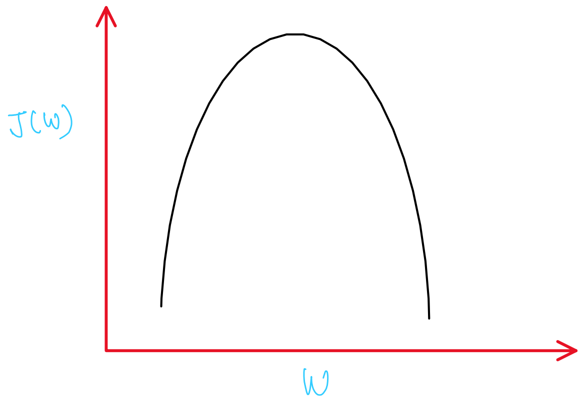
for Gradient **Ascent** Algorithm

$J(W)$ = Profit function $(w_1, w_2, w_3 \;---\; w_n)$

$$\boxed{w_j = w_j + \alpha \cdot \frac{1}{m} \frac{\partial J(w)}{\partial w_j}}$$

hyper parameter
α = learning Rate
plays a important Role how model parameter will converge

$$\boxed{w_j = w_j + \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_w(x_j)^{(i)} - y^{(i)}\right) \cdot x_j^{(i)}}$$



Batch / vanilla Gradient Descent Algorithm

Repeat {

$$w_j = w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_w(x)^{(i)} - y^{(i)}\right) \cdot x_j^{(i)}$$

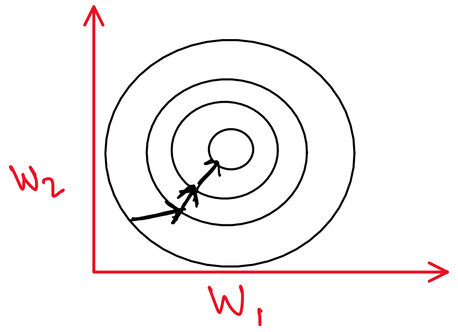
$$w_0 := w_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_w(x)^{(i)} - y^{(i)}\right) \cdot x_0^{(i)}$$

$$w_1 := w_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_w(x)^{(i)} - y^{(i)}\right) \cdot x_1^{(i)}$$

$\vdots$

$$w_n := w_n - \alpha \frac{1}{m} \sum_{i=1} (h_w(x) - y) \cdot x_n$$

}



Contour plot

GDA optimizer

m is very large
n is also large 1000 or higher

↓ We shift to

Stocastic Gradient Descent Algorithm

1. Randomly shuffle training samples.
2. Repeat {

for $i = 1, 2, \ldots m$ {

$$w_j = w_j - \boxed{\alpha (h_w(x)^{(i)} - y^{(i)}) \cdot x_j^{(i)}} \leftarrow \frac{\partial J(w)}{\partial w_j}$$

$$w_0 := w_0 - \alpha (h_w(x)^{(i)} - y^{(i)}) \cdot x_0^{(i)}$$

$$w_1 := w_1 - \alpha (h_w(x)^{(i)} - y^{(i)}) \cdot x_1^{(i)}$$

$$\vdots$$

$$w_n := w_n - \alpha (h_w(x)^{(i)} - y^{(i)}) \cdot x_n^{(i)}$$

}

}

Mini-Batch Gradient Descent Algorithm

① Create a Batch size, b

② Randomly shuffle the batches

Repeat {

for $i = 1, 2, \ldots b$ {

$$w_j := w_j - \frac{\alpha}{b} \overbrace{\sum_{i=1}^{b} \left(h_w(x^{(i)}) - y^{(i)}\right) \cdot x_j^{(i)}}^{\frac{\partial J(w)}{\partial w_j}}$$

$$w_0 := w_0 - \frac{\alpha}{b} \sum_{i=1}^{b} \left(h_w(x^{(i)}) - y^{(i)}\right) \cdot x_0^{(i)}$$

$$w_1 := w_1 - \frac{\alpha}{b} \sum_{i=1}^{b} \left(h_w(x^{(i)}) - y^{(i)}\right) \cdot x_1^{(i)}$$

$$\vdots$$

$$w_n := w_n - \frac{\alpha}{b} \sum_{i=1}^{b} \left(h_w(x^{(i)}) - y^{(i)}\right) \cdot x_n^{(i)}$$

}

}

b = hyper parameter

How to choose batch size - b ?

b may be 16, 32, 64, 128

$2^4, 2^5, 2^6, 2^7$