Gradient Descent

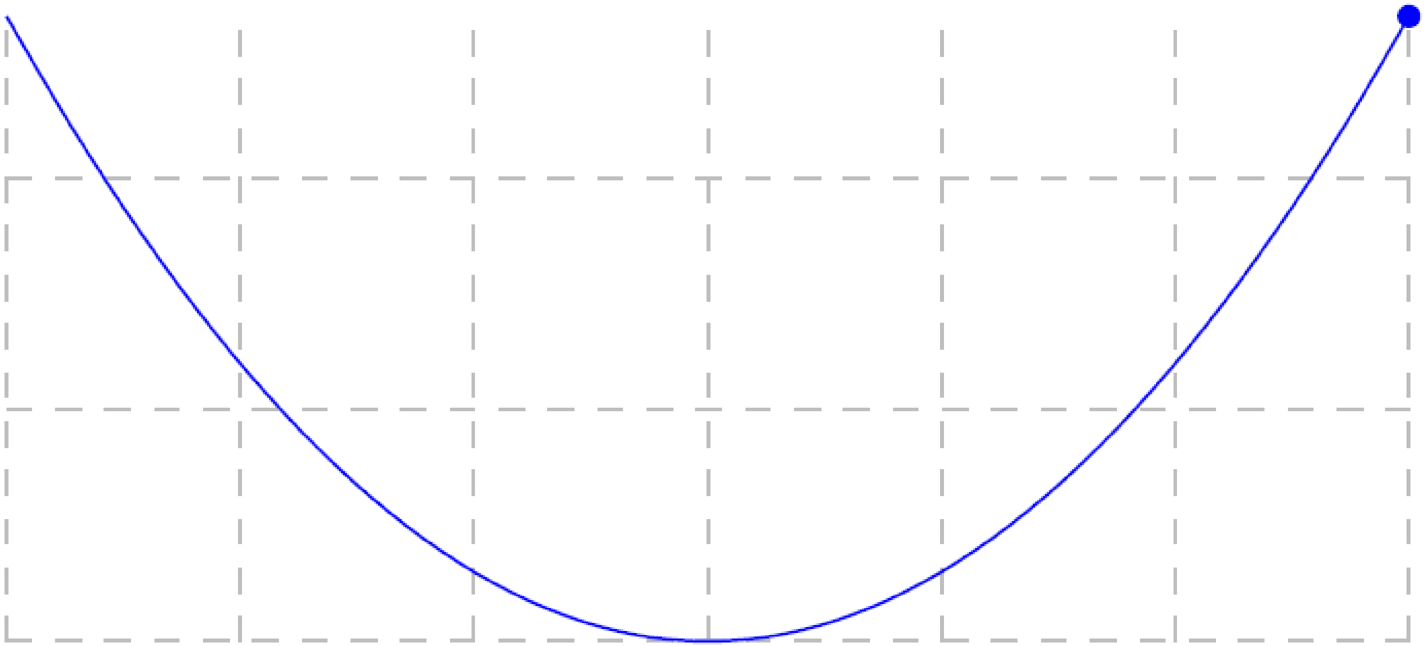
CS-309

What is Gradient Descent?

- Gradient descent is an optimization algorithm used to minimize a function.
- It is commonly used to optimize the parameters of a machine learning model to minimize a loss function.

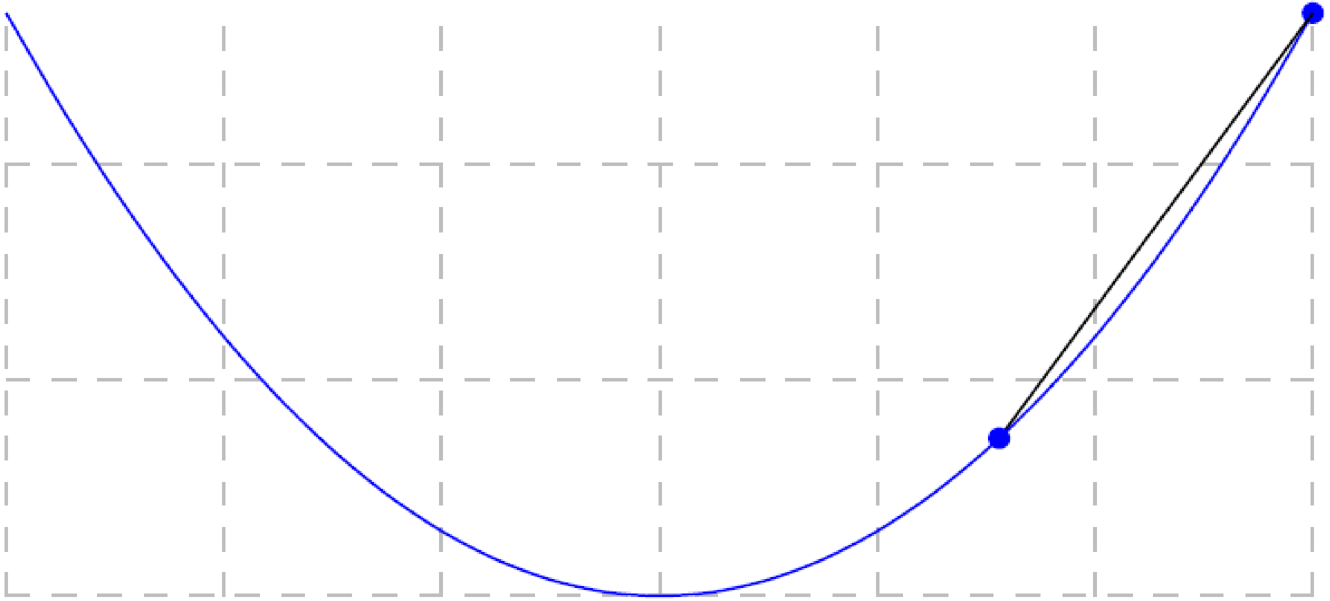
How Does it Work?

- **Initialization**: Gradient descent starts by initializing the parameters (weights) of the model to some arbitrary values.
- **Compute the Gradient**: At each iteration, the algorithm computes the gradient of the loss function with respect to the model parameters.
- The gradient essentially measures the slope of the loss function at the current parameter values and indicates the direction of the steepest increase.
- **Update the Parameters**: The parameters are updated in the direction opposite to the gradient to minimize the loss function.



$y = 0.3x^2, x_0 = 3, \alpha = 0.8$

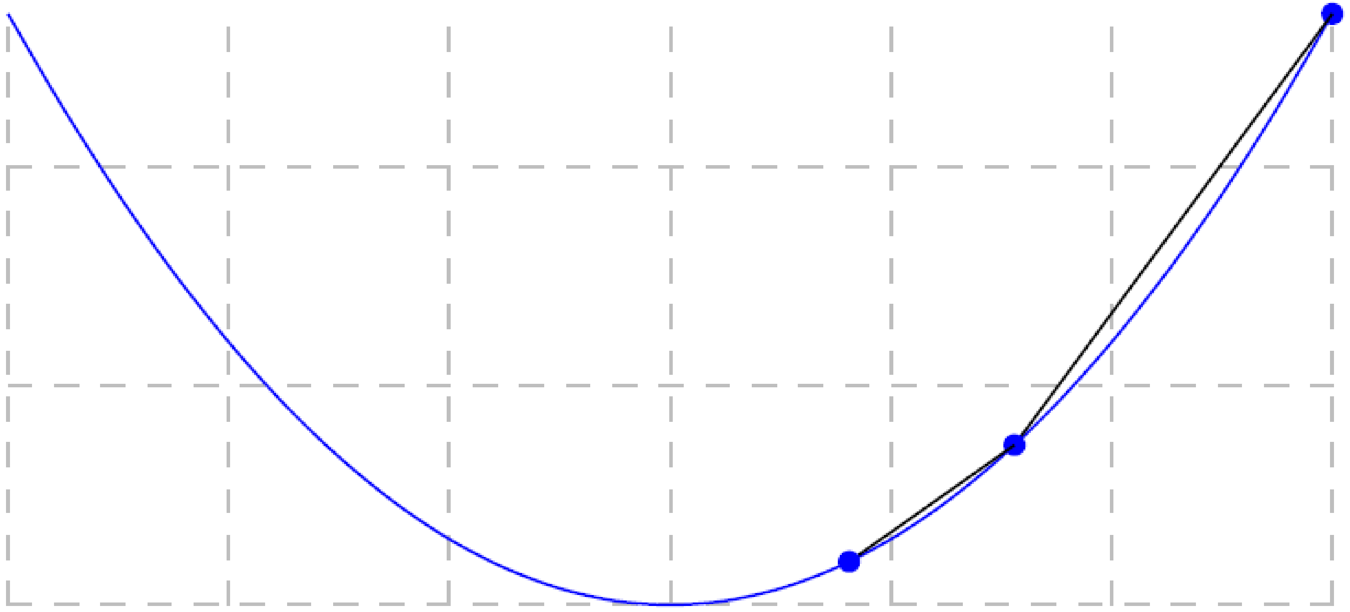
gradient=1.80002



$y = 0.3x^2, x_0 = 3, \alpha = 0.8$

gradient=1.80002

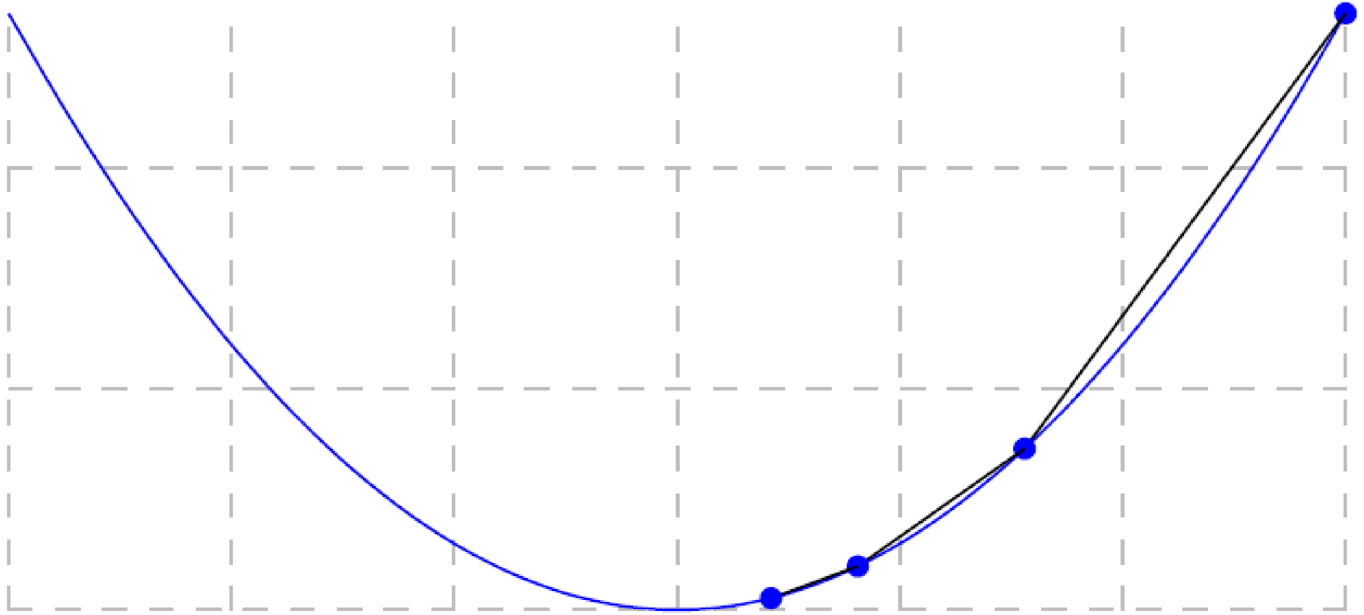
x_{new}=1.56001



$y = 0.3x^2, x_0 = 3, \alpha = 0.8$

gradient=0.936

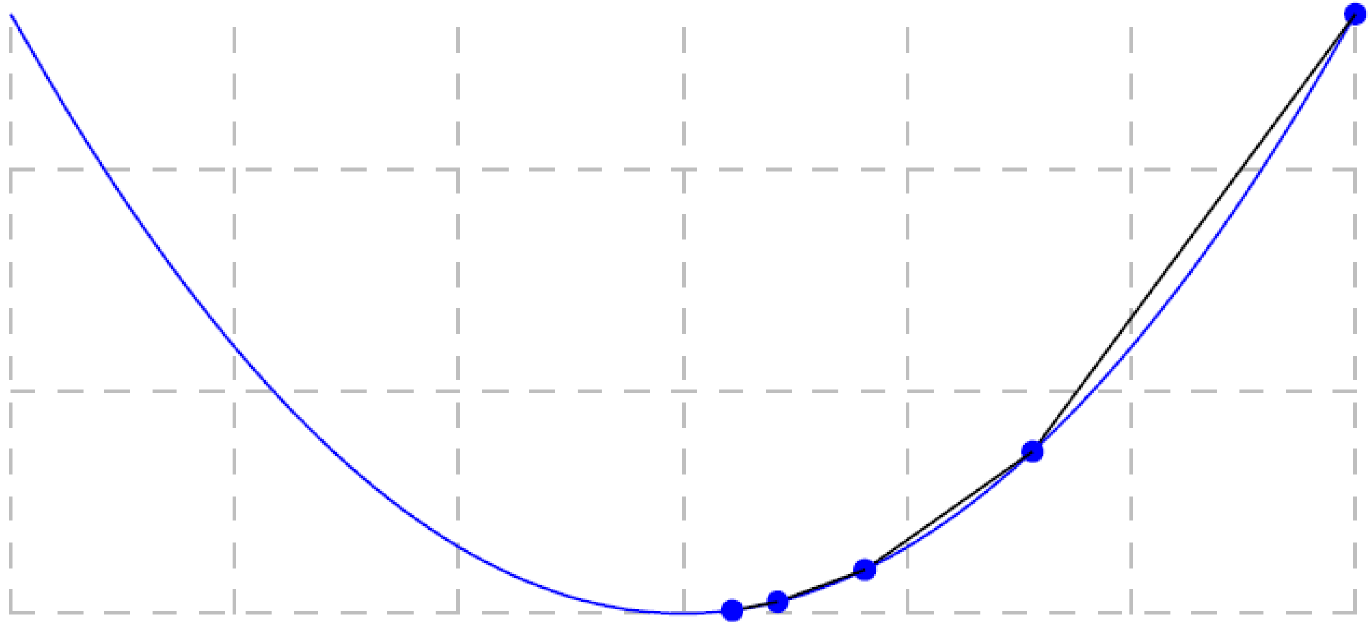
x_{new}=0.81122



$y = 0.3x^2, x_0 = 3, \alpha = 0.8$

gradient=0.48672

x_{new}=0.42184



$y = 0.3x^2$, $x_0 = 3$, $\alpha = 0.8$

gradient=0.2531

x_{new}=0.21938

Why do we need Gradient Descent in Linear Regression?

- Linear regression can indeed be solved using a closed-form solution, known as the ordinary least squares (OLS) solution.
- **Computational Efficiency**: The closed-form solution involves matrix inversions, which can be computationally expensive and numerically unstable.
- **Scalability:** Gradient descent is highly scalable and can handle large datasets that may not fit into memory.
- **Flexibility:** Gradient descent is a more flexible optimization technique that can be adapted to different loss functions and regularization penalties.

Regression as Parameter Fitting

We seek coefficients that minimize the sum of squared error of the points over all possible coefficients.

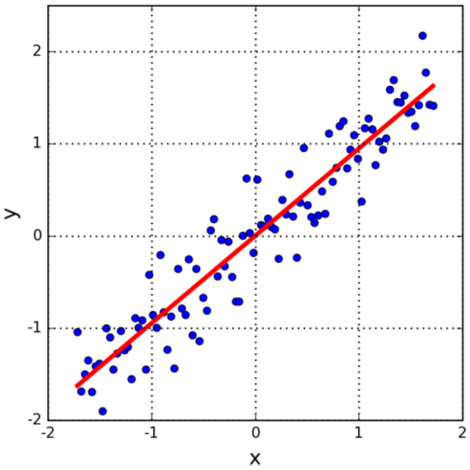
$$J(w_0, w_1) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

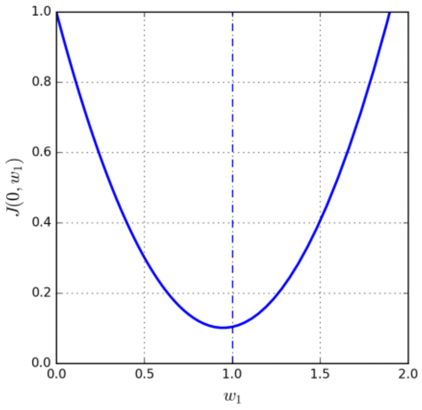
Here the regression line is:

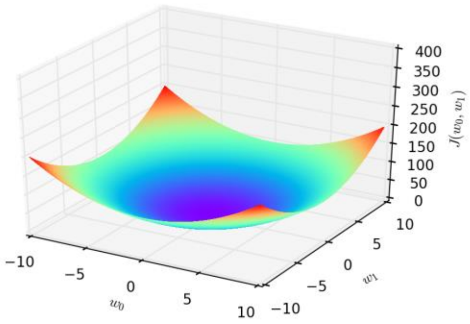
$$f(x) = w_0 + w_1 x$$

Lines in Parameter Space

The error function J(w0,w1) is convex, making it easy to find the single local/global minima.







Gradient Descent Search

A space with only one local/global minima is called convex.

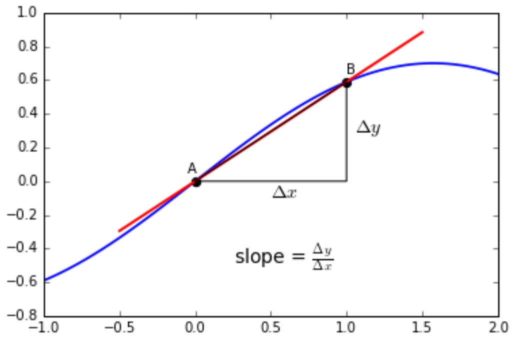
When a search space is convex, it is easy to find the minima: just keep walking down.

The fastest direction down is defined by the slope or tangent at the current point.

The Fastest Way Down

The direction down at a point is given by its derivative, which specified by its tangent line:

This *could* be approximately computed by finding the point (x+dx,y(x+dx)) and fitting the line with (x,y(x))



Gradient Descent for Regression

Gradient descent algorithm

$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\
&\qquad (\text{for } j = 1 \text{ and } j = 0) \\
&\}
\end{aligned}
$$

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- $J(\theta_0, \theta_1)=\frac{1}{2m}\sum_{i=1}^{m}(h_{\theta(x^{(i)})} - y^{(i)})^2$

$$=\frac{1}{2m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}((\theta_0 + \theta_1 x^{(i)})^2 - 2(\theta_0 + \theta_1 x^{(i)})\, y^{(i)} + (y^{(i)})^2$$

$$=\frac{1}{2m}\sum_{i=1}^{m}(A + B + c)$$

A= $(\theta_0 + \theta_1 x^{(i)})^2$,   B= $-2(\theta_0 + \theta_1 x^{(i)})\, y^{(i)}$, C=$(y^{(i)})^2$

- A= $(\theta_0 + \theta_1 x^{(i)})^2$

  $= \theta_0^2 + 2\theta_0\theta_1 x^{(i)} + (\theta_1 x^{(i)})^2$

  $\frac{\partial A}{\partial \theta_0} = 2\theta_0 + 2\theta_1 x^{(i)}$, $\frac{\partial A}{\partial \theta_1} = 2\theta_0 x^{(i)} + 2\theta_1 (x^{(i)})^2$

  B= $-2(\theta_0 + \theta_1 x^{(i)})\ y^{(i)}$

  $\frac{\partial B}{\partial \theta_0} = -2y^{(i)}$, $\frac{\partial B}{\partial \theta_1} = -2x^{(i)} y^{(i)}$

  C=$(y^{(i)})^2$, $\frac{\partial C}{\partial \theta_0} = 0$, $\frac{\partial B}{\partial \theta_1} = 0$.

- $$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_0} = \frac{1}{2m} \sum_{i=1}^{m} \frac{\partial A}{\partial \theta_0} + \frac{\partial B}{\partial \theta_0} + \frac{\partial C}{\partial \theta_0}$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \left(2\theta_0 + 2\theta_1 x^{(i)}\right) + \left(-2y^{(i)}\right) + (0)$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(\left(\theta_0 + \theta_1 x^{(i)}\right)\right) - \left(y^{(i)}\right)$$

$$= \frac{1}{m} \sum_{i=1}^{m} (h_{\theta} x^{(i)} - y^{(i)})$$

- $$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1} = \frac{1}{2m} \sum_{i=1}^{m} \frac{\partial A}{\partial \theta_1} + \frac{\partial B}{\partial \theta_1} + \frac{\partial C}{\partial \theta_1}$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \left(2\theta_0 x^{(i)} + 2\theta_1 (x^{(i)})^2\right) - \left(2x^{(i)} y^{(i)}\right) + (0)$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(x^{(i)}\right). \left((\theta_0 + \theta_1 x^{(i)}) - y^{(i)}\right)$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(x^{(i)}\right).(h_\theta(x^{(i)}) - y^{(i)})$$

Update Rules

- $\theta_0 = \theta_0 - \alpha \dfrac{\partial J(\theta_0, \theta_1)}{\partial \theta_0}$

  $= \theta_0 - \dfrac{\alpha}{m} \sum_{i=1}^{m} (h_\theta x^{(i)} - y^{(i)})$

  $\theta_1 = \theta_1 - \alpha \dfrac{\partial J(\theta_0, \theta_1)}{\partial \theta_1}$

  $= \theta_1 - \dfrac{\alpha}{m} \sum_{i=1}^{m} \left(x^{(i)}\right).(h_\theta(x^{(i)}) - y^{(i)})$

Example

- Consider the following pair (*x*, *y*) of points - (1, 2), (2, 4), (3, 6), (4, 8)
- Let us try to fit a curve as follows y = βx where β is initialized with 4, learning rate (α) as 0.1
- MSE as cost function. Derivative will be $\frac{1}{4}\sum_{i=1}^{4}(x_i).(\beta x_i - y_i)$

- Step Derivative New β

Step	Derivative	New β
1	15	2.5
2	3.75	2.13
3	0.94	2.03
4	0.23	2.01
5	0.06	2.00

Stochastic gradient descent (SGD)

- Stochastic gradient descent is an optimization method for unconstrained optimization problems. In contrast to (batch) gradient descent, SGD approximates the true gradient of $J(\theta_0,\theta_1)$ by considering a single training example at a time.

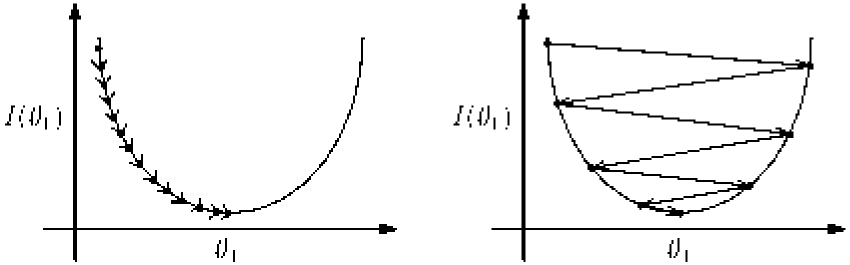
Example of SGD

- Consider the following pair (x, y) of points - (1, 2), (2, 4), (3, 6), (4, 8)
- Let us try to fit a curve as follows $y = \beta \times x$ where β is initialized with 4, learning rate as 0.1
- MSE as cost function. Derivative will be $x(\beta \times x - y)$

Step	Point	Derivative	New β
1	(1,2)	1*(4.0*1-2)=2.0	3.80
2	(2,4)	2*(3.8*2-4)=7.2	3.08
3	(3,6)	3*(3.1*3-6)=9.7	2.11
4	(4,8)	4*(2.1*4-8)=1.7	1.94
5	(1,2)	1*(1.9*1-2)=-0.1	1.94
6	(2,4)	2*(1.9*2-4)=-0.2	1.97
7	(3,6)	3*(2.0*3-6)=-0.3	1.99
8	(4,8)	4*(2.0*4-8)=-0.1	2.00
9	(1,2)	1*(2.0*1-2)=0.0	2.00

Effect of Learning Rate / Step Size

- Taking too small steps results in slow convergence to the optima.
- But too large a step overshoots the goal.



What is the Right Learning Rate?

Monitor the value of the loss function J() over the course of optimization.

If progress is too slow, increase by a multiplicative factor (say 3) or accept.

If J gets larger, the step size is too large, decrease by a multiplicative factor (say ⅓).

The Adam optimizer is an algorithm for this.

- Cost-function= $\prod_{i=1}^{n}(p_i{}^{y_i}(1-p_i)^{(1-y_i)})$
- Where $p_i = \frac{e^{x_i{}^T\beta}}{1+e^{x_i{}^T\beta}} = \frac{1}{1+e^{-z}}$ where z= $x_i{}^T\beta = \beta_0 + \sum_{i=1}^{m}\beta_i X_i$
- Z=-Log(Cost-function)=$\sum_{i=1}^{n}(-y_i).(log p_i) - (1 - y_i).(\log(1-p_i))$
- A=- $y_i\ (log p_i)$ B= $-(1-yi).(\log(1-p_i))$

- A=- y_i $(log p_i)$
- $\frac{\partial A}{\partial \beta_i} = \frac{-y_i}{p_i} . \frac{\partial p_i}{\partial z} . \frac{\partial z}{\partial \beta_i}$

$$= \frac{-y_i}{p_i} . \frac{e^{-z}}{(1+e^{-z})^2} .X_i$$

$$= \frac{-y_i}{p_i} . \frac{1}{(1+e^{-z})} . \frac{e^{-z}}{(1+e^{-z})} .X_i$$

$$= \frac{-y_i}{p_i} . p_i .(1- p_i). X_i$$

$$=- y_i X_i .(1- p_i)$$

- B= $-(1-y_i).(\log(1-p_i))$
- $\frac{\partial B}{\partial \beta_i} = \frac{(1-y_i)}{(1-p_i)} . \frac{\partial p_i}{\partial z} . \frac{\partial z}{\partial \beta_i}$

$= \frac{(1-y_i)}{(1-p_i)}. p_i.(1- p_i).$ x_i

$=$ x_i $. \ p_i \ . \ (1-y_i)$

$\frac{\partial Z}{\partial \beta_i} = \sum_{i=1}^{n} \left(\frac{\partial A}{\partial \beta_i}\right) + \left(\frac{\partial B}{\partial \beta_i}\right)$

$=\sum_{i=1}^{n}(- y_i$ x_i $.(1- p_i)) + ($ x_i $. \ p_i \ . \ (1-y_i))$

$=\sum_{i=1}^{n}($ x_i $). (p_i - y_i)$

Disadvantages Of Gradient Descent

- **Sensitivity to Learning Rate**: The choice of learning rate can be critical in Gradient Descent since using a high learning rate can cause the algorithm to overshoot the minimum, while a low learning rate can make the algorithm converge slowly.
- **Slow Convergence:** Gradient descent takes large time to converge for large dataset.
- **Local Minima:** Gradient Descent can get stuck in local minima if the cost function has multiple local minima.