

---

The NPTEL logo is centered in the background. It features a circular emblem with a stylized flower or star shape in the center. The emblem is composed of multiple overlapping layers of petals or segments in shades of orange, yellow, and pink. Below the emblem, the word "NPTEL" is written in a bold, orange, sans-serif font.

# Gradient Descent Algorithms

NPTEL

---

# Gradient Descent

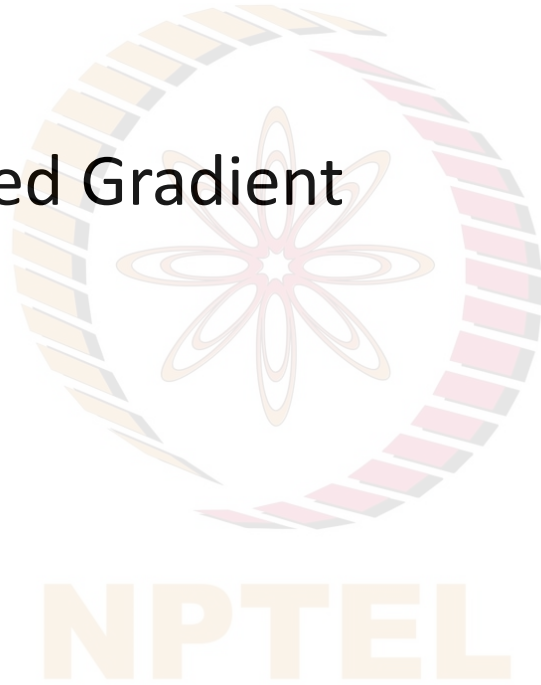
- Batch gradient descent
  - Makes a parameter update i.e. calculates gradient of cost function, using the entire training data set
  - No online update is possible
- Stochastic gradient descent
  - Parameter updates are done for every training sample
  - Online learning is possible
  - Causes large oscillations in objective function due to frequent updates
- Mini-batch gradient descent
  - Combination of the above
  - Performs update for a mini-batch of training data

$$w = w - \alpha \frac{\partial J}{\partial w}$$

# Algorithms

---

- Momentum
- NAG- Nesterov Accelerated Gradient
- Adagrad
- RMS-Prop



# Momentum

- Adds a fraction of the previous update to the current update
- Helps to take larger steps in the relevant direction, preventing oscillations that are common in vanilla SGD

$$\underline{\delta w_n} = \underline{\gamma \cdot \delta w_{n-1}} + \left[ \alpha \nabla_w J(w) \right]$$

$$\gamma = 0.9$$

$$w_n = w_{n-1} + \delta w_n$$

# Nesterov Accelerated Gradient (NAG)

- We can compute the updated parameter value using the current gradient and treat it like a look ahead
- Evaluate gradient at the new parameter values and then perform an update to the parameters.

$$\underline{\delta w_n} = \gamma \cdot \underline{\delta w_{n-1}} + \alpha \nabla_w J(\underline{w_{n-1}} + \delta w_n)$$

$$w_n = w_{n-1} + \delta w_n$$

$$\underline{w_n = w_{n-1} + 2 \nabla J}$$

# Adagrad

- Use a different learning rate for different parameters
- Compute update for every parameter using gradient
- Scale general learning rate at a particular iteration using accumulated squared gradients from previous iterations
- Leads to rapidly diminishing updates

$$w_{n,i} = w_{n-1,i} + \frac{\alpha}{\sqrt{G_{n,i} + \varepsilon}} g_{n,i}$$

$G_{1,i} = G_{2,i} = \frac{g_{1,i}^2 + g_{2,i}^2}{2}, \quad G_{4,i} = \frac{g_{1,i}^2 + g_{2,i}^2 + g_{3,i}^2}{3}$

$i \rightarrow$  parameter index  
 $g_{n,i} \rightarrow \nabla(w_i(J))$   
timestep  $n$

# Adadelta

- Extension of Adagrad
- Instead of storing running sum of squared gradients, use weighted average of squared gradients
  - i.e. the current average depends on the average in the previous iteration and the square of the gradient in the current iteration

$$\delta w_{n,i} = \frac{\alpha}{\sqrt{E[g^2]_n + \varepsilon}} g_{n,i}$$

$$E[g^2]_n = \gamma \cdot E[g^2]_{n-1} + (1 - \gamma) g_n^2$$

$$w_n = w_{n-1} + \delta w_n$$

$$g_n \rightarrow \nabla_w J$$

$$w_i$$
$$\gamma = 0.9$$