# Deep Learning

## 3.3 Gradient Descent

Dr. Konda Reddy Mopuri
kmopuri@iittp.ac.in
Dept. of CSE, IIT Tirupati

# Training an ML model

1. Finding the parameters that minimize the training loss

$$W^*, \mathbf{b}^* = \underset{W,\mathbf{b}}{\operatorname{argmin}} \mathcal{L}(f(\cdot; W, \mathbf{b}); \mathcal{D})$$

# Training an ML model

1. Finding the parameters that minimize the training loss

$$W^*, \mathbf{b}^* = \underset{W, \mathbf{b}}{\operatorname{argmin}} \, \mathcal{L}(f(\cdot; W, \mathbf{b}); \mathcal{D})$$

2. How do we find these optimal parameters?
    1. Closed form solution (e.g. linear regression)
    2. Ad-hoc recipes (e.g. Perceptron, K-NN classifier)

# Training an ML model

1. Finding the parameters that minimize the training loss

$$W^*, \mathbf{b}^* = \underset{W,\mathbf{b}}{\operatorname{argmin}} \mathcal{L}(f(\cdot; W, \mathbf{b}); \mathcal{D})$$

2. How do we find these optimal parameters?
   1. Closed form solution (e.g. linear regression)
   2. Ad-hoc recipes (e.g. Perceptron, K-NN classifier)
   3. What if the loss function can't be minimized analytically?

3. General minimization method used in such cases is the 'Gradient Descent'.

# Gradient

1. Given a function

$$
\begin{aligned}
f : & \mathcal{R}^D \rightarrow \mathcal{R} \\
& x \rightarrow f(x_1, x_2, \ldots, x_D)
\end{aligned}
$$

# Gradient

1. Given a function

$$
\begin{aligned}
f :& \mathcal{R}^D \to \mathcal{R} \\
& x \to f(x_1, x_2, \ldots, x_D)
\end{aligned}
$$

2. Its gradient is the mapping

$$
\begin{aligned}
\nabla f :& \mathcal{R}^D \to \mathcal{R}^D \\
& x \to \left( \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_D} \right)
\end{aligned}
$$

# Gradient

1. Given a function

$$
\begin{aligned}
f : & \mathcal{R}^D \to \mathcal{R} \\
& x \to f(x_1, x_2, \ldots, x_D)
\end{aligned}
$$

2. Its gradient is the mapping

$$
\begin{aligned}
\nabla f : & \mathcal{R}^D \to \mathcal{R}^D \\
& x \to \left( \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_D} \right)
\end{aligned}
$$

3. It computes how much each input component influences the value of $f$ locally.

# Gradient

1. Given a function

$$\begin{aligned} f :& \mathcal{R}^D \to \mathcal{R} \\ & x \to f(x_1, x_2, \ldots, x_D) \end{aligned}$$

2. Its gradient is the mapping

$$\begin{aligned} \nabla f :& \mathcal{R}^D \to \mathcal{R}^D \\ & x \to \left( \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_D} \right) \end{aligned}$$

3. It computes how much each input component influences the value of $f$ locally.
4. The gradient vector is interpreted as the direction and rate of fastest increase.

# Gradient Descent in ML

1. Goal is to minimize the error (or loss): determine the parameters $\theta$ that minimize the loss $\mathcal{L}(\theta)$

# Gradient Descent in ML

1. Goal is to minimize the error (or loss): determine the parameters $\theta$ that minimize the loss $\mathcal{L}(\theta)$
2. Gradient points uphill $\rightarrow$ negative of gradient points downhill

# Gradient Descent in ML

1. Goal is to minimize the error (or loss): determine the parameters $\theta$ that minimize the loss $\mathcal{L}(\theta)$
2. Gradient points uphill $\rightarrow$ negative of gradient points downhill
3. 
    1. Start with an arbitrary initial parameter vector $\theta_0$
    2. Repeatedly modify it via updating in small steps
    3. At each step, modify in the direction that produces steepest descent along the error surface

# Gradient Descent in ML

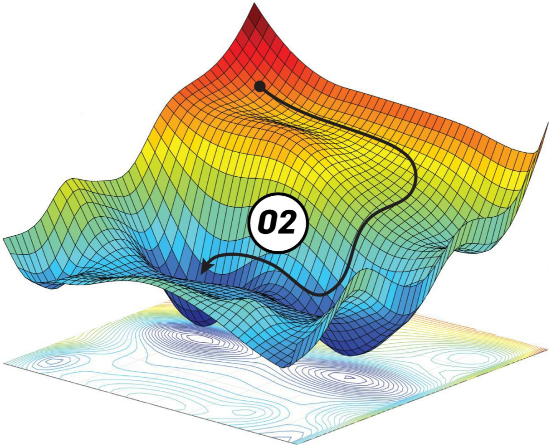


Figure credits:Ahmed Fawzy Gad

# Gradient Descent in ML

1. Start with an arbitrary initial parameter vector $\theta_0$
2. Repeatedly modify it via updating in small steps
3. At each step, modify in the direction that produces steepest descent along the error surface

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t)$$

# Gradient Descent in ML

1. Start with an arbitrary initial parameter vector $\theta_0$
2. Repeatedly modify it via updating in small steps
3. At each step, modify in the direction that produces steepest descent along the error surface

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t)$$

4. Almost always ends in a local minimum, choice of parameters $\theta_0$ and $\eta$ are important.

# Gradient descent example

1. Logistic regression (we will work it out on whiteboard)