Lecture 29

# Foundation and Uses of Deep Learning

**Mohammad Sabik Irbaz**

Data Scientist, Leadbook Pte. Ltd.

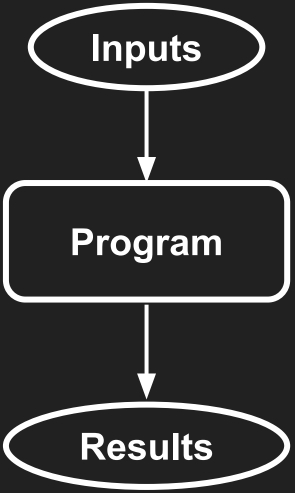
Former Lead ML Engineer, Omdena & Pioneer Alpha

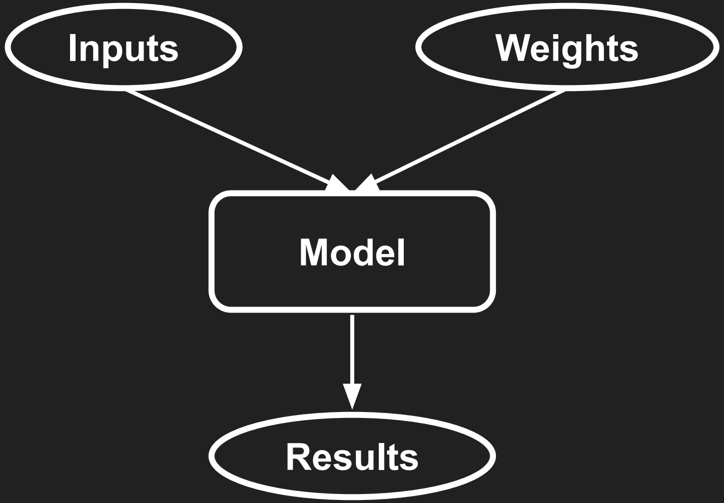
sabikirbaz@iut-dhaka.edu

## Dokkho Data Science Career Program

By MasterCourse

# Traditional Programming
# vs.
# Machine Learning





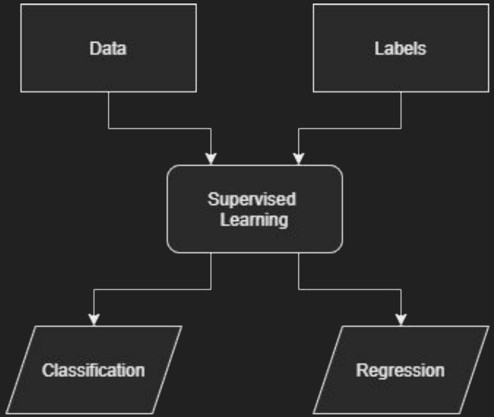
Traditional Programming

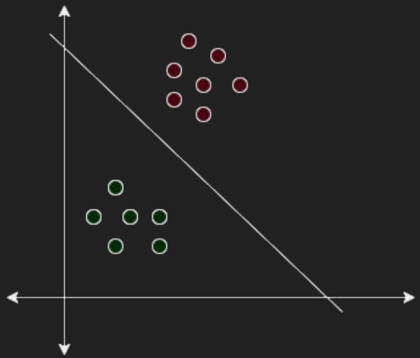
Machine Learning

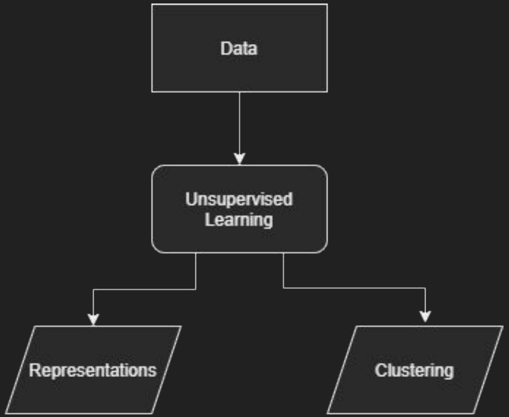
# Types of ML Algorithms

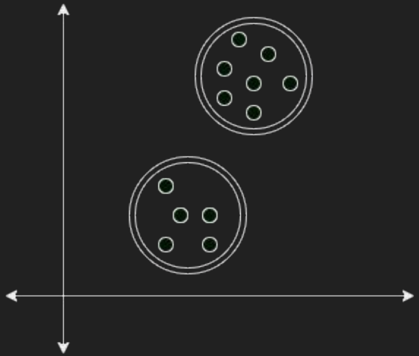
- ➔ Supervised Learning
- ➔ Unsupervised Learning
- ➔ Semi-Supervised Learning
- ➔ Self-Supervised Learning (SSL)
- ➔ Reinforcement Learning (RL)
- ➔ Active Learning

# Supervised vs Unsupervised Learning

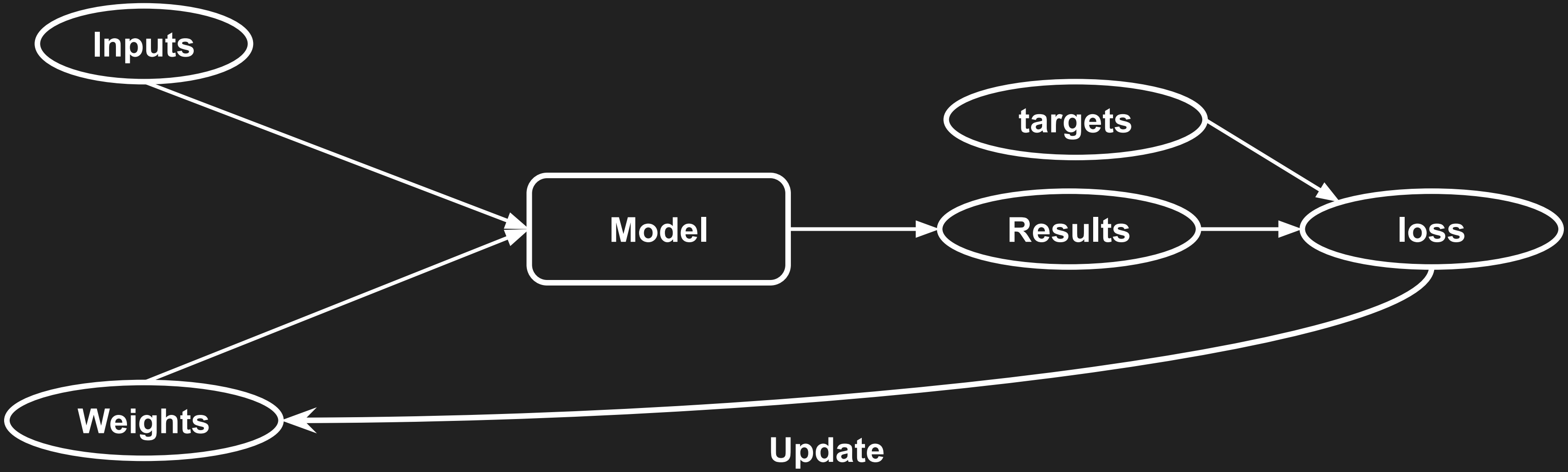






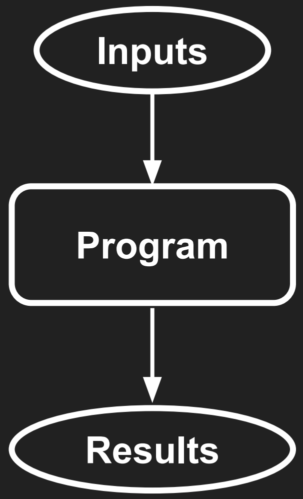


# Training ML Models

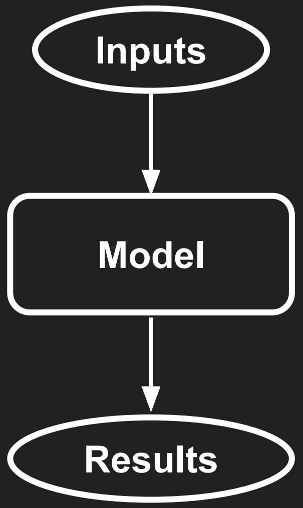


Generally loss = difference between targets  and results

# During Inference



Traditional Programming



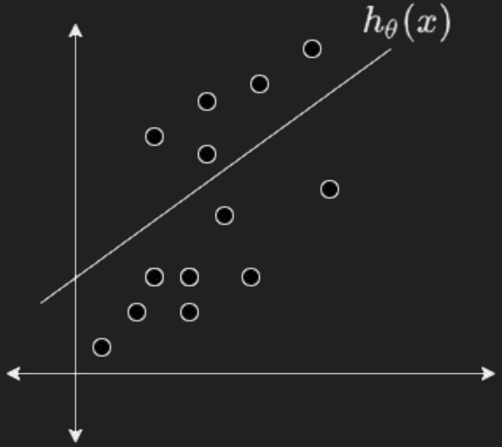
Machine Learning

# Hypothesis

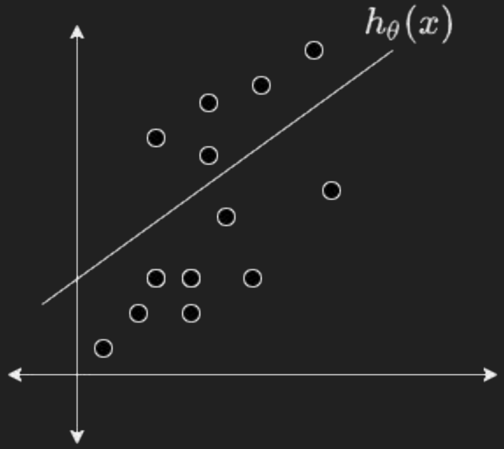
## Representation of a line : y = mx + c

## Let’s redefine it

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



# Mean Squared Error (MSE)



=> Imagine you have ‘m’ data points

=> You drew a line

=> MSE calculates how far is that line from data points in average.

=> Mean is for the average

=> Square is for the Euclidean Distance

# Cost Function

Cost Function in case of Linear Regression is nothing but MSE

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

m = #samples

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

Our Objective is to minimize the cost.

How do we do that?

# Cost Function

Cost Function in case of Linear Regression is nothing but MSE

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

m = #samples

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

```
Our Objective is to minimize the cost.
```

How do we do that?

# Gradient Descent

Images taken from Google Photos

# Gradient Descent (cont.)

Visualization by
Lili Jiang, Head  of Data Science, Quora

# Gradient Descent (cont.)

**Repeat until Convergence**

$$\theta_j = \theta_j - \alpha * \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

# Gradient Descent (cont.)

**Repeat until Convergence**

$$\theta_j = \theta_j - \boxed{\alpha} * \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

**Learning Rate**

# Gradient Descent (cont.)

Repeat until Convergence

$$\theta_j = \theta_j - \underbrace{\boxed{\alpha}}_{\text{Learning Rate}} * \underbrace{\boxed{\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)}}_{\text{Partial Derivative of the Cost Function}}$$

# Where To Use Deep Learning

- Computer Vision
- Natural Language Processing
- Medicine
- Biology
- Text to Image
- Text to Speech
- RecSys
- Robotics
- Time Series Forecasting

Most of the objective tasks that does not require critical thinking

# Where We Cannot Use DL

If we do not have enough data.
Data is the raw material for Deep Learning Models.

Where the decisions are subjective.
For example: Court Rulings in different countries

We should not use DL in applications that are straightforward.
DL should be explored after we exhaust traditional programming