Module 1

Supervised Learning:

- Labelled Data
- Goal is to Learn mapping functions
- Examples: Linear Regression, Logistic Regression, Decision Trees, Support Vector Machines, Neural Networks

Unsupervised Learning:

- Un-labelled Data
- Goal is to learn underlying patterns or hidden structure in data
- Examples: Clustering(K means, Hierarchical Clustering) , Dimensionality Reduction- Reducing Features(PCA, t-SNE)

Semi-Supervised Learning:

- Small amount of labelled data and large amount of unlabeled data
- Goal is to improve learning accuracy in both types of data
- Example: Training a spam filter with some labelled emails and then a large amount of unlabeled ones

Reinforcement Learning:

- Algorithm learns by interacting with an environment and receiving rewards and penalties.
- Goal: Learn a policy that maximizes cumulative reward over time
- Key Concepts : Agent, Environment , State , Action , Reward
- Example: Self driving cars

MLE:

We don't have any beliefs. We only trust the data that we saw. So if we saw 10 coin tosses and 7 heads, we assume that the probability of heads is 0.7 (we assume its a biased coin due to the experiment)

Pick the parameter that makes the data you saw the most likely to happen

- $$M \quad amaP(\quad)$$
- Likelihood is:
$$L(\mu) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x_i - \mu)^2}{2\sigma^2}\right]$$
- take log for simplicity:
$$\log L(\mu) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2$$

MAP: Maximum A Posteriori Estimation

Now suppose you have some *prior knowledge*.

You think:

> “Most coins are pretty fair — probably close to p=0.5”

This is your **prior belief** — your gut feeling *before* looking at data.

MAP says:

> "Let's combine what the data tells us *and* what we already believed before."

So you take:

- The **likelihood** (what the data suggests)
- The **prior** (your belief before seeing the data)

and multiply them together.

The new combined curve (posterior) will lean a little toward 0.5 (your prior belief).

Concept	What it Means	Uses What?	Analogy
MLE	Choose the parameter that makes the data most likely	Only the data	"I trust what I saw."
MAP	Choose the parameter that's most likely given both the data and prior beliefs	Data + prior belief	"I trust my data, but also my experience."

Gradient Descent:

- Method to minimize loss function

$$: - (()$$

Guess - learning rate x differential of loss function

Symbol	Meaning
$()$	The current guess (like your current position on the mountain)
$()$	The learning rate — how big your steps are

Symbol	Meaning
$(())$ \	The slope (gradient) — tells you which direction is uphill

- **Batch Gradient Descent:
  - Very stable, but can be slow if you have a huge dataset.
  - Analogy: You check the whole mountain before every step — careful but time-consuming.

Stochastic Gradient Descent (SGD):

Stochastic Means Random

- Uses **one data point at a time**.
- Faster, but your steps are a bit noisy — you may wiggle left and right before reaching the bottom.
- Analogy: You look at just one rock under your foot before moving — fast, but shaky.

Mini-Batch Gradient Descent:

- Uses **a small group of data points** (e.g., 32 or 64 samples) per step.
- It's a compromise between the two — fast and relatively stable.
- Analogy: You check a handful of rocks before moving — balanced and efficient.

Regression:

Concept	Formula	Shape	When to Use
Linear Regression	$(y \quad +)$	Straight line	Relationship looks linear

Concept	Formula	Shape	When to Use
Polynomial Regression	$(y \quad + \ +_2{}^2 + \ +_n{}^n)$	Curved line	Data has bends or nonlinear trend

Least Squares Linear Regression:

- Least Square means we are trying to minimize the squared values of an equation.
- This equation is (Actual Output-The output we predicted)
- We do the squared thing so that we can maximize the large differences /variance.
- $$\sum(yatal - yeite)^2$$
- Least squares linear regression is a way to draw the best straight line through data by minimizing how far the points are from the line — it helps us see relationships and make predictions.

Normal Equation and Closed Form Solution:

Aspect	Normal Equation (Closed Form)	Gradient Descent
Method	Solve $(\quad y)$ directly	Iteratively update weights using the gradient of the error
Formula	$(\ ()^{-}y)$	$(: \ - \ )$
Computational cost	Expensive if number of features (n) is large (involves inverting $(n \times n)$ matrix)	Scales better for large datasets; works with any number of features
Convergence	Immediate (direct solution)	Depends on learning rate $()$ and number of

Aspect	Normal Equation (Closed Form)	Gradient Descent
		iterations
Limitations	Fails if $()$ is not invertible	Always works; may need more iterations or regularization

When to use each

U wanna one shot this shit in one go with math, you go with normal equation.

If its impossible to figure out those matrix shit u go with gradient descent

- **Normal Equation**:
  - Small to medium datasets
  - Few features (columns in XXX)
  - Want an exact, closed-form solution
- **Gradient Descent**:
  - Very large datasets
  - Many features
  - When computing $() - ()^{-}() -$ is expensive or impossible