Machine Learning (CS 306)
Machine Learning Lab (CS 360)

Instructor: Dr. Moumita Roy

Teaching Assistants: Indrajit Kalita, Veronica Naosekpam

Email-ids:

moumita@iiitg.ac.in

veronica.naosekpam@iiitg.ac.in

indrajit.kalita@iiitg.ac.in

Mobile No: +91-8420489325 (only for emergency quires)

Machine Learning (ML)

- Machine learning is the science of getting computers to act without being explicitly programmed
- It means “computer can learn something apart from for which it can be programmed”

Formally

- Suppose, we have design a model to carry out some task (e.g. for object recognition)

➢ <u>What is the role of ML here?</u>

Study the algorithms that

- improve their performance P
- at some task T
- with experience E

Well defined learning task has three tuples: <P, T, E>

Where ML is necessary?

- Human expertise does not possible
- Humans are unable to explain their expertise (speech recognition)
- Solution changes in time
- Solution needs to be adapted to particular cases (user biometrics)

Straight forward mathematical model not available or input /output relation not known a proiri

Application of ML

- Self-driving car (run-time learning requires)
- Speech recognition
- Earth observation
- Efficient web search

ML helps us to move towards human-level AI

Task

Handwritten character recognition

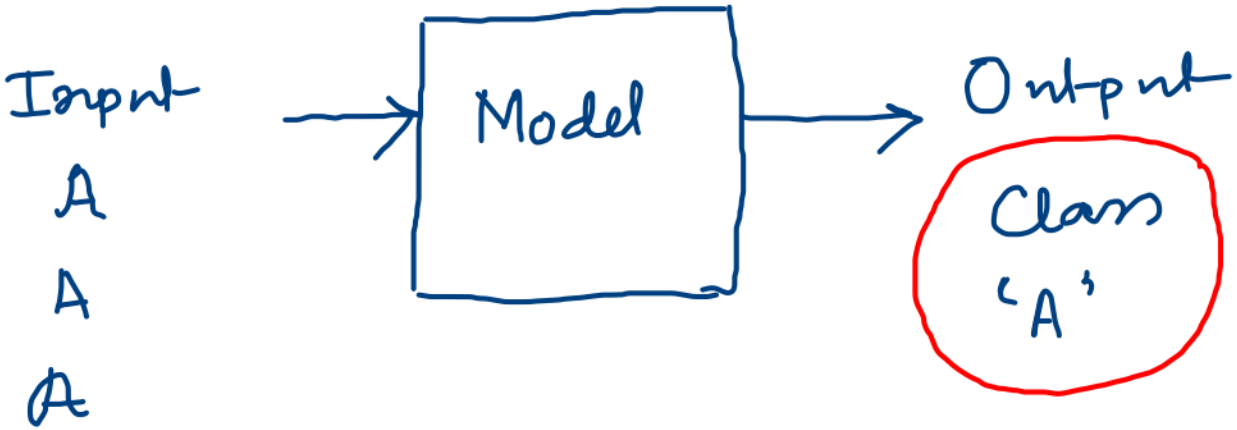
What are the challenges?

Handwritten character recognition

Student 1 A, B, C, D

Student 2 A, B, C, D

Student 3 A, B, C, D



Challenges

- How can we feed Handwritten character in the model?
- Input-output relation is not straight forward. For different handwritten format of A as input , the model should recognize it as A.
- No fixed mathematical model exists.
- How can we design such model?

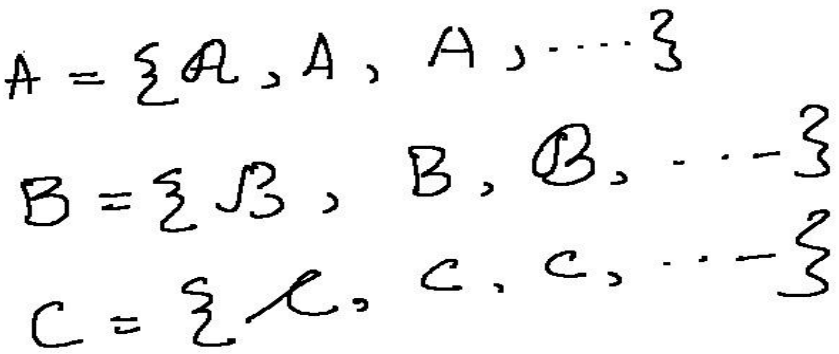
We need to enter the scenario of approximation. The designed model should able to say “This input handwritten character is most similar to A”

Solution

- Collect different set of handwritten samples
- Make model learn from such samples (learning phase/training phase)
- What is the meaning of learning?
  - Extract knowledge from available information
  - The model should able to learn from experience
  - Here, the experience can gather by learning from different set of handwritten samples
  - Different model (learning algorithms) have different capability to learn (like different human)
  - We need to design acceptable learning algorithm

Training samples

Designing a good learning algorithm is a open scope of research



Learning algorithm should able to improve its performance from experience (extracted from training samples)

Testing phase/generalization phase

- If we have a trained model, then we need to pass the unknown samples to the model
- Model can provide a score for each of the possible categories
- Finally, we need to assign the best matched category to the samples

Model is now capable to learn the input-output relation

Performance measure

- Designing of learning model is mostly motivated from human learning strategy
- For same task, we need to apply different learning models to choose the best one
- Need to check the performance of the model
- There are different ways to measure the performance

Pattern and Feature

- Need to input handwritten character (named as pattern/object) in model
- Extract some useful information from pattern to describe it to the model
- It means that we need to find some features to represent the pattern
- Here, features may be the pixel values of the image corresponding to handwritten character.

New challenges

- Design learning algorithm
- Collect suitable features to represent the pattern
- Choose the measuring indices to assess the performance of model (after and during training)
- Collect samples (data-driven model) to learn the behavior of patterns

Major Topics

- Introduction to Machine Learning
- Polynomial curve fitting problem
- Linear regression
- Gradient descent algorithm
- Logistic regression
- Regression vs. classification
- K-fold cross validation (train, test, validation set)
- Introduction to artificial neural networks
- Singlelayer perceptron
- Multilayer perceptron
- Supervised vs. unsupervised learning

(Continued…)

Major Topics

- Clustering algorithms (hard and soft clustering)
- Elbow method for k means clustering
- Different evaluation matrices for clusters (Silhouette Coefficient)
- Self-organizing feature maps neural networks
- Decision Trees
- K-nearest neighbor classifier and Min distance classifier
- Bayesian classifier and Bayesian networks
- Gaussian mixture models and EM algorithm
- Support vector machine
- Radial basis function neural networks
- Dimensionality reduction and principal component analysis
- Statistical significance test: T test
- Introduction to Markov decision process

Regression

Machine Learning (CS 306)

Instructor: Dr. Moumita Roy

Teaching Assistants: Indrajit Kalita, Veronica Naosekpam

Email-ids:

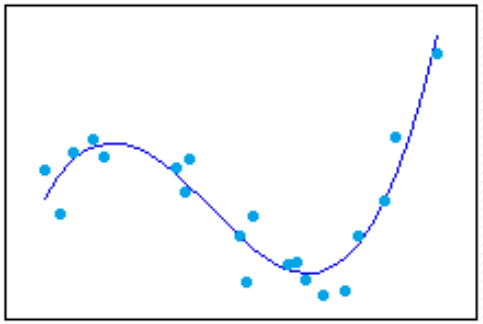
moumita@iiitg.ac.in

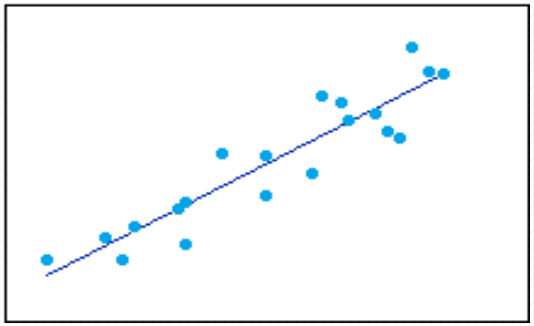
veronica.naosekpam@iiitg.ac.in

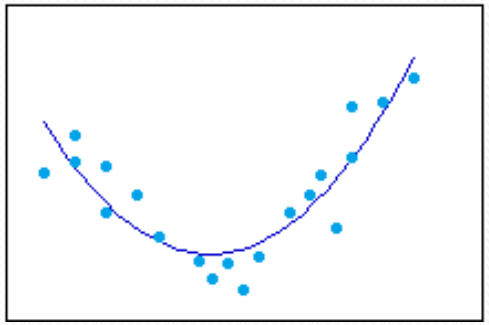
indrajit.kalita@iiitg.ac.in

Mobile No: +91-8420489325 (only for emergency quires)

Polynomial Curve fitting

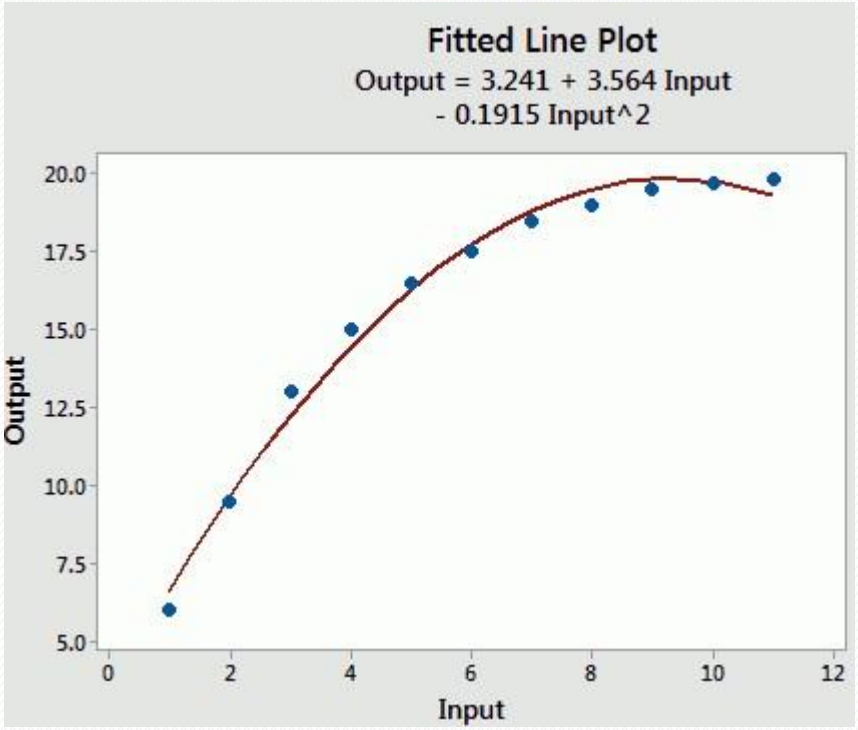






Polynomial Curve fitting (with known function)

Independent variable (Input/x)
Dependent variable (Output/y)



Structure of the polynomial (Not known)

Try to approximate polynomial structure with some (training) samples $<x_i, y_i>$

x — Independent variable / Input

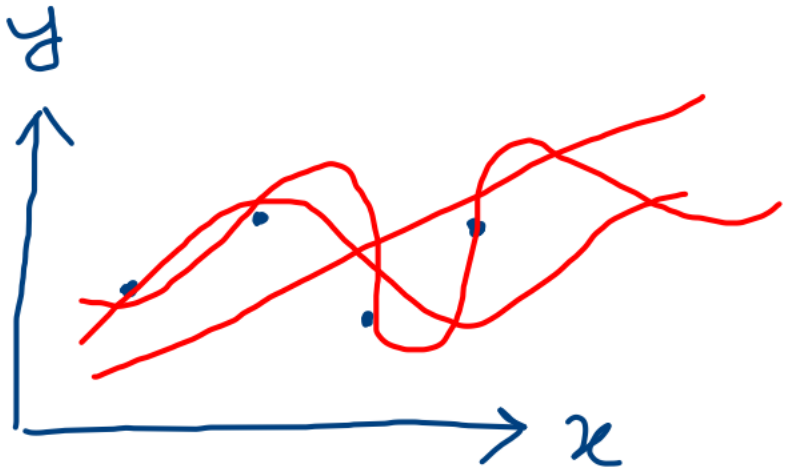
y → Dependent variable / Output

Suppose, Some set of (x, y) is provided,

x	y
0.1	5
0.6	2
0.9	1.2

Solution?

Solution



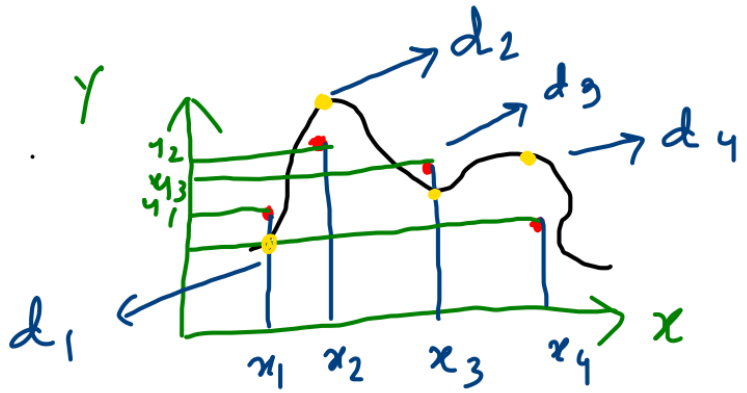
make a guess about the equation

many options??

Which one is better??

Draw curves

- find best-fit ones



x	Actual Y	Predicted d
.1	5	4
.6	2	1
.9	1.2	6
1.2	3	2

predicted value for the approximate polynomial structure ??

m = no of training samples

d_1, d_2, d_3, d_4

$$\text{Error} = \sum_{i=1}^{m} (y_i - d_i)^2$$

Sum-of-square error

Try to find a polynomial structure (equation) which minimize (Error)

To sum up

- The structure of the polynomial is not known (one independent and one dependent variables).
- Instead of it, we have some samples to approximate the structure of the polynomial
- There are many possible structure (curves/equations)

  Like, $y=a+bx$, $y=ax^2$, $y=a+bx^2+cx^3$……
- We can assume a structure and calculate error.
- Try to find a polynomial that minimizes the error (best fit curve)

→proceed with trial and error approach

Supervised Learning

- There is a set of training samples $<x_i, y_i>$
- The goal is to learn function/hypothesis $h(x_i)$ so that it is a good predictor for the corresponding value of y using the training samples

$$\boldsymbol{h: x \rightarrow y}$$

- If y is continuous, then the problem is called regression
- If y is discrete, then the problem is called classification.
- Aim-> to design learning algorithm to approximate hypothesis structure

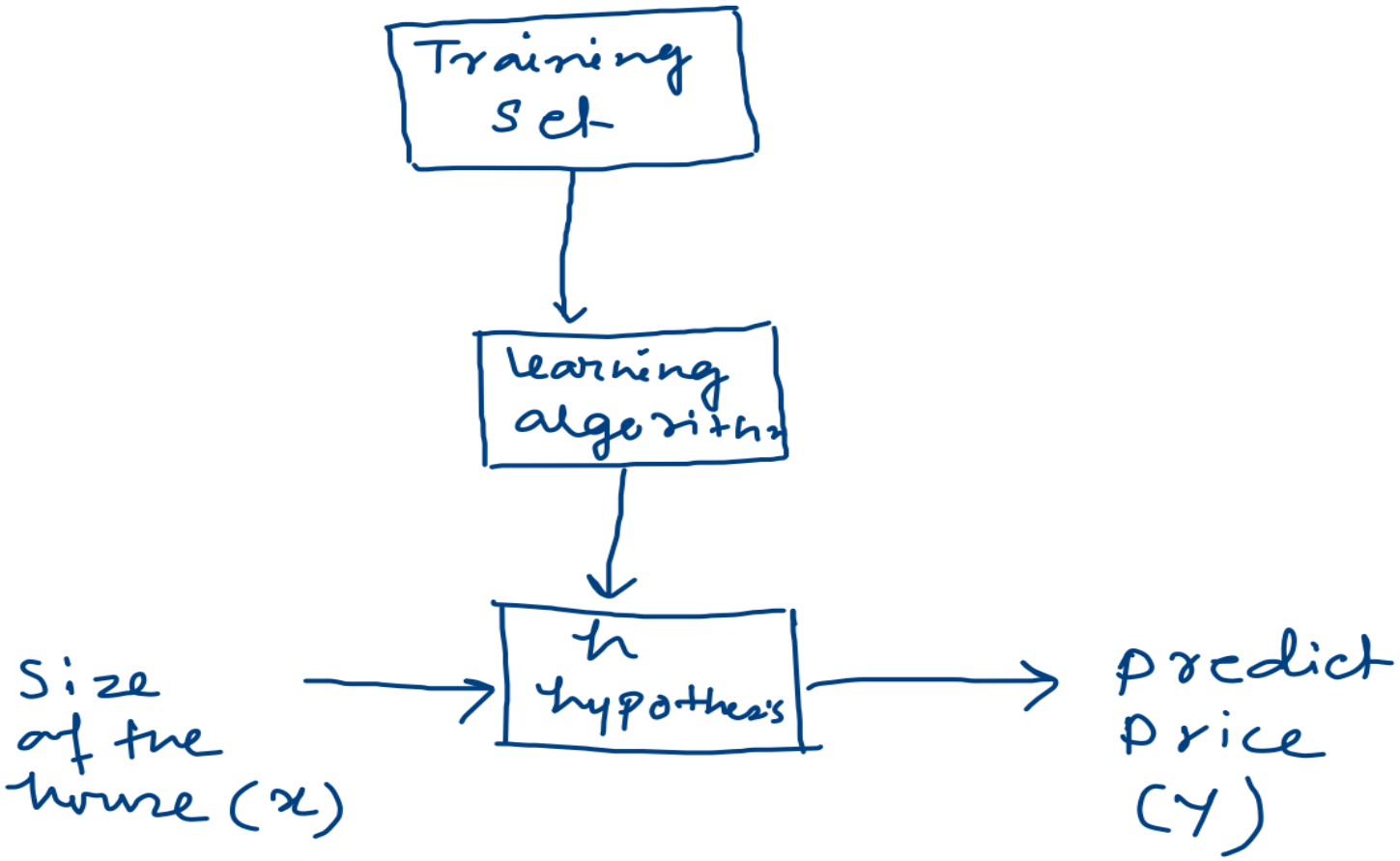
Some application for Regression

Size of house (in sq.ft)	Price (Lakh)
600	15
1120	30
2000	40
700	30

Predict the price of the house

H.W: Find some more applications of regression problem

Graphical Abstract



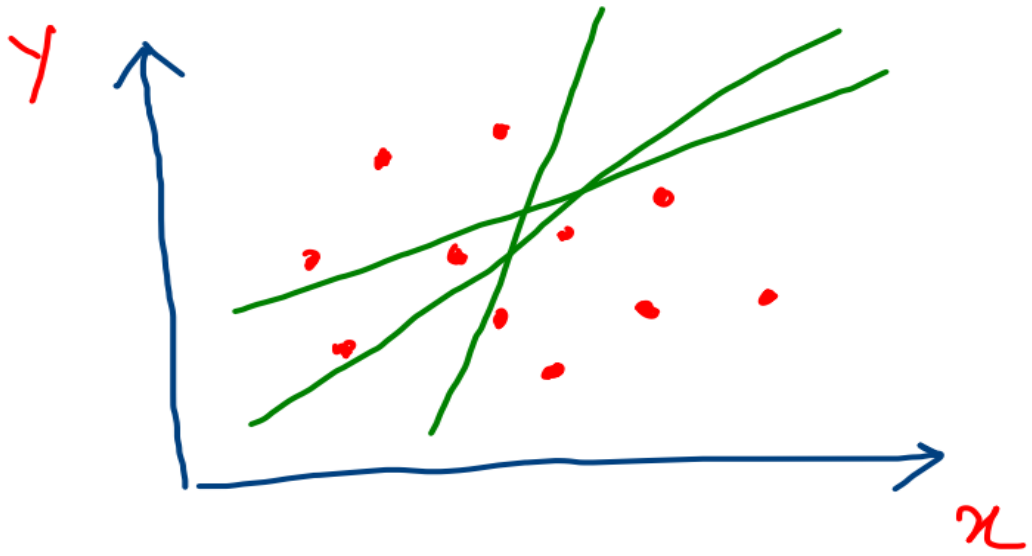
Linear Regression (univariate)

- Assume, y is a linear function of x

$$h(x)=w_0+w_1 * x$$

- Learn $h(x)$ using the training set $<x_i, y_i>$
- m=number of training sample, number of feature =1

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



Linear Regression (multiple features)

Age of the house (year)	No. of bedrooms	Size of house (in sq.ft)	Price (Lakh)
2	2	600	15
10	3	1120	30
6	4	2000	40
7	2	700	30

Linear Regression (multiple features)

$$x_i = [x_{i,1} \quad x_{i,2} \quad x_{i,3} \quad \dots \quad x_{i,n}]$$

n = number of features (independent variables)

$$h(x_i) = w_0 + w_1 x_{i,1} + w_2 x_{i,2} + w_3 x_{i,3} + \dots + w_n x_{i,n}$$

$$= \sum_{j=0}^{n} w_j x_{i,j} \quad | \quad x_{i,0} = 1$$

Training samples: $\langle x_i, y_i \rangle$ — n number of features

$$W = [w_0 \quad w_1 \quad w_2 \quad \dots \quad w_n]$$

$$J(W) = \frac{1}{2} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

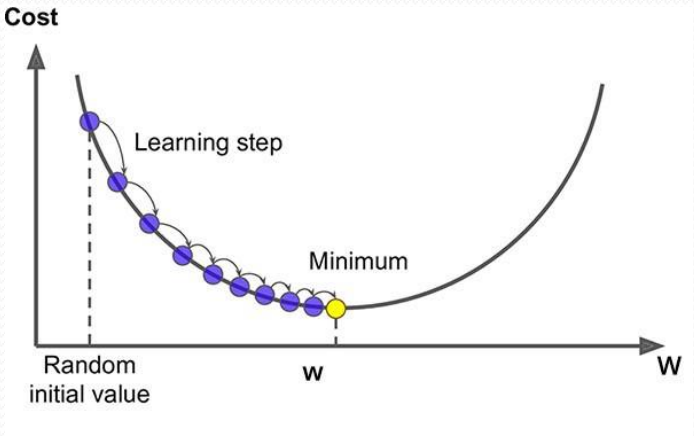
↳ minimize

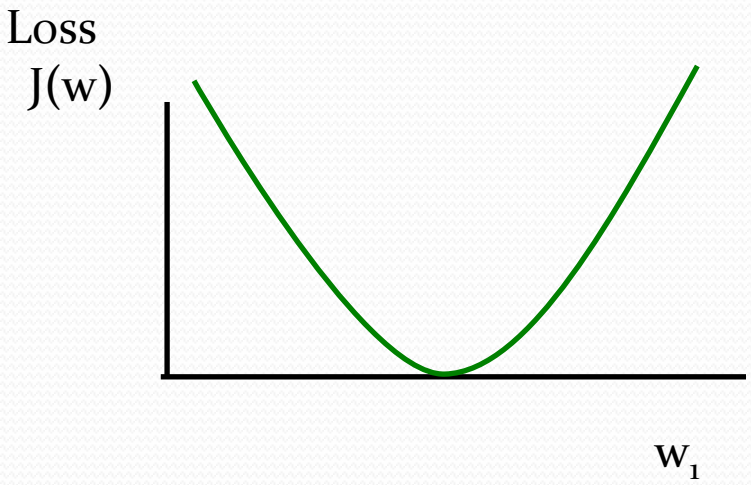
Aim

- To obtain parameter $w_0, w_1, w_2, \ldots, w_n$ for good approximation of y
- Check random initialization of w and find the best fit one by trial and error approach
- Solve the problem using linear algebra (numerical complexity)
- Fit model by minimizing sum of squared errors (least squares)

Finding the minimum

Finding the minimum

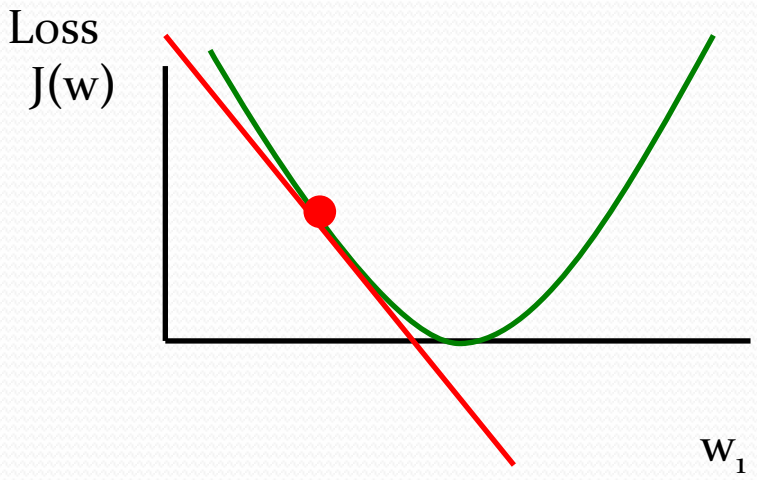




How do we do this for a function?

One approach: gradient descent

Partial derivatives give us the slope (i.e. direction to move) in that dimension (suppose w_1)



One approach: gradient descent

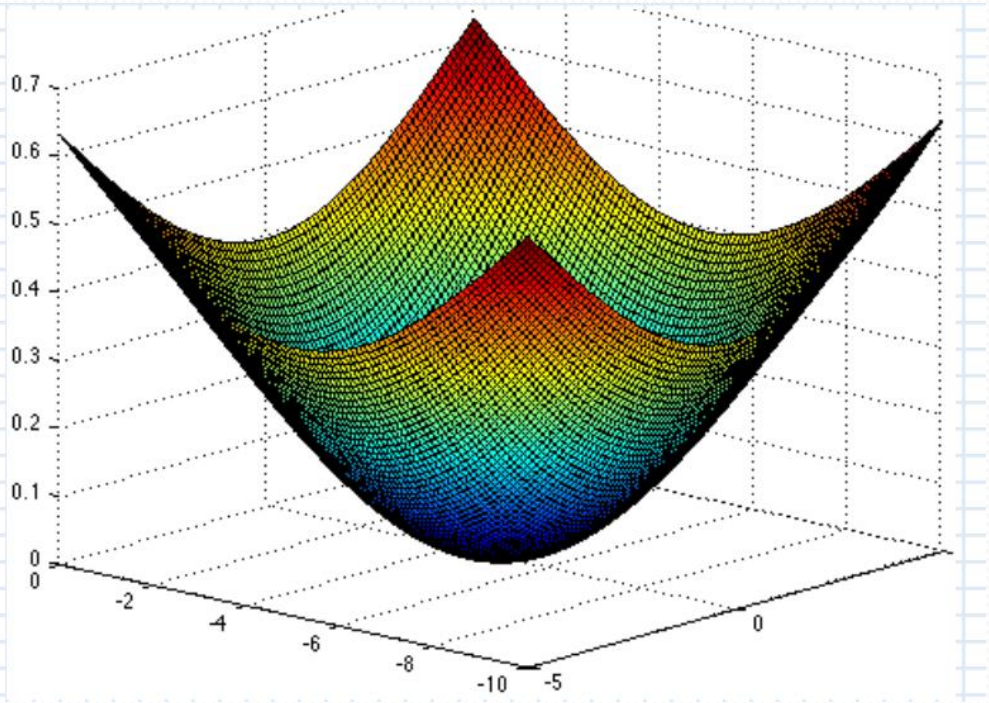
Partial derivatives give us the slope (i.e. direction to move) in that dimension

Approach:

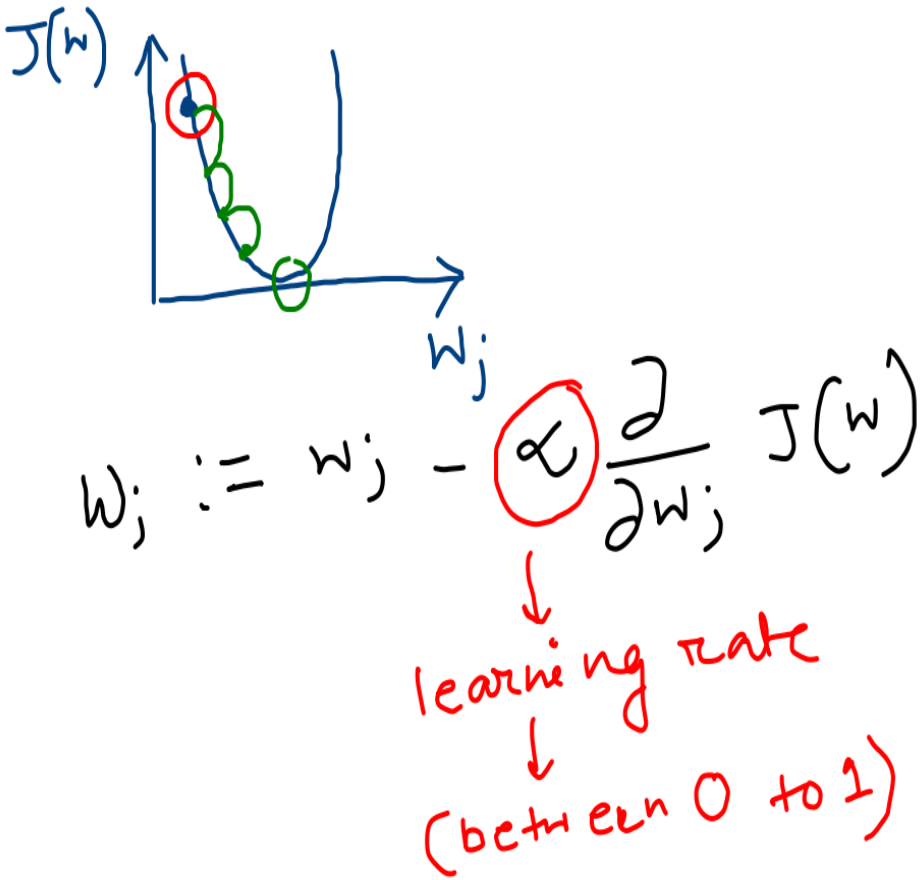
- pick a starting point (w)
- repeat:
  - pick a dimension
  - move a small amount in that dimension towards decreasing loss (using the derivative)/opposite direction of slope

Gradient Descent for Linear Regression

- Convex loss function
- Geometrically: Error surface is bowl shaped



Gradient Descent



- Simultaneously update for $j=1,2,\ldots,n$

Gradient Descent for Linear Regression

$W = [w_0 \; w_1 \; w_2 \; \dots \; w_n]$

- randomly initialized between $[-3 \text{ to } +3]$

n = number of features

$X_1 \quad X_2 \quad \dots \quad X_m$

m = number of patterns

$X_i = [x_{i,1} \quad x_{i,2} \quad \dots \quad x_{i,j} \quad \dots \quad x_{i,n}]$

$$h(x_i) = w_0 + w_1 x_{i,1} + w_2 x_{i,2} + \dots + w_n x_{i,n}$$

$$= \sum_{j=0}^{n} w_j x_{i,j}$$

$h(x_1) \rightarrow$

$h(x_2) \rightarrow$

$\vdots$

$h(x_m) \rightarrow$

$$J(W) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(x_i) - y_i \right)^2$$

MSE

↳ mean

Gradient Descent for Linear Regression

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w)$$

Update

w_1

w_2

w_3

$\vdots$

w_n

} Simultaneously

Gradient Descent for Linear Regression

$$\frac{\partial}{\partial w_j} J(w) = \frac{\partial}{\partial w_j} \left[\frac{1}{2m} \sum_{i=1}^{m} \left(h(x_i) - y_i \right)^2 \right]$$

$$\vdots$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(h(x_i) - y_i \right) x_{i,j}$$

$$w_j := w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(x_i) - y_i \right) x_{i,j}$$

$$= w_j + \alpha \frac{1}{m} \sum_{i=1}^{m} \left(y_i - h(x_i) \right) x_{i,j}$$

- if we have large number of training samples!?

Algorithmic representation for gradient descent

Step 1: Randomly initialize $w_0, w_1, ..., w_n$ and learning rate

Step 2: Calculate $h(X_i)$ for all training samples

Step 3: Calculate $J_{itr}(W)$ for itr^{th} epoch

Step 4: Update all parameters $w_0, w_1, ..., w_n$

Step 5: Repeat steps 2-5 until

$$| J_{itr}(W) - J_{itr-1}(W)| > \rho$$

Step 6: Report MSE $J_{itr}(W)$

ρ: small value (may be *0.0001*)

Epoch/training step: one time updating considering all the training samples

Gradient Descent for Linear Regression

- **Batch gradient descent:**

➢ all the training samples are taken into consideration for a single updating of all parameters (after each epoch).

➢ Problem for huge number of training samples

- **Stochastic gradient descent:**

➢ Update parameters by considering just one training sample at time (repeat epoch by epoch till convergence)

$$w_j := w_j + \alpha (y_i - h(x_i)) x_{i,j}$$

↳ simultaneously for all $w_1, w_2 \dots w_n$

Algorithmic representation for stochastic gradient descent

Step 1: Randomly initialize $w_0, w_1, \ldots, w_n$ and learning rate

Step 2: Initialize $itr = 0$ (itr stands for epoch)

Step 3: $J_{itr}(W)=0$

Step 5: Repeat Steps 6-8 for all the training samples

Step 6: Calculate $h(X_i)$ for i^{th} training sample

Step 7: Update $J_{itr}(W)= J_{itr}(W) +(h(X_i)-Y_i)^2$

Step 8: Update all parameters $w_0, w_1, \ldots, w_n$

Step 9: $J_{itr}(W)=(1/2m)\ J_{itr}(W)$

Step 10: Check convergence if $| J_{itr}(W) - J_{itr-1}(W)|<\rho$ then stop and report MSE $J_{itr}(W)$; else go to step 11

Step 11: $itr=itr+1$ and repeat steps 3-10

ρ: small value (may be *0.0001*)

Exercise 1

Design a linear regression model using (batch mode) gradient descent approach by considering the following dataset and use this model to predict the output for the input pattern [3 12]

Feature 1	Feature 2	Output
1	2	6
2	10	24

For demonstration, note the followings:

1. Learning rate: 0.1
2. Number of training steps/epochs:2
3. All parameters are initialized as 1

Parameters vs. HyperParameters

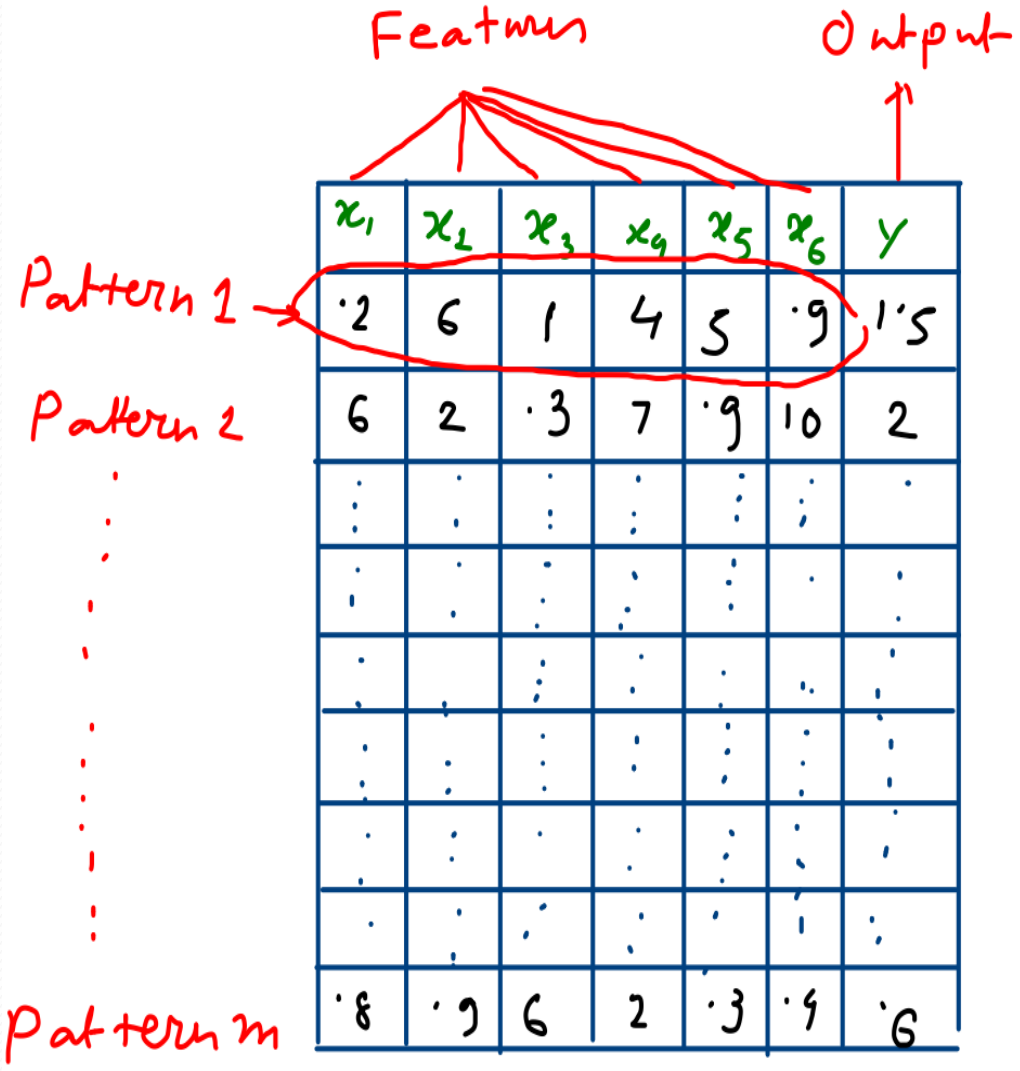
- **Parameter:** A model parameter is a variable of the selected model which can be estimated by fitting the given data to the model.

  Example: $w_0, w_1, w_2, \ldots, w_n$

- **HyperParameter:** A model hyperparameter is the parameter whose value is set before the model start training. They cannot be learned by fitting the model to the data.

  Example: learning rate, number of epochs

Training, Test, Validation Set



Split randomly

training set (20%.)

validation Set (10%.)

test set (70%.)

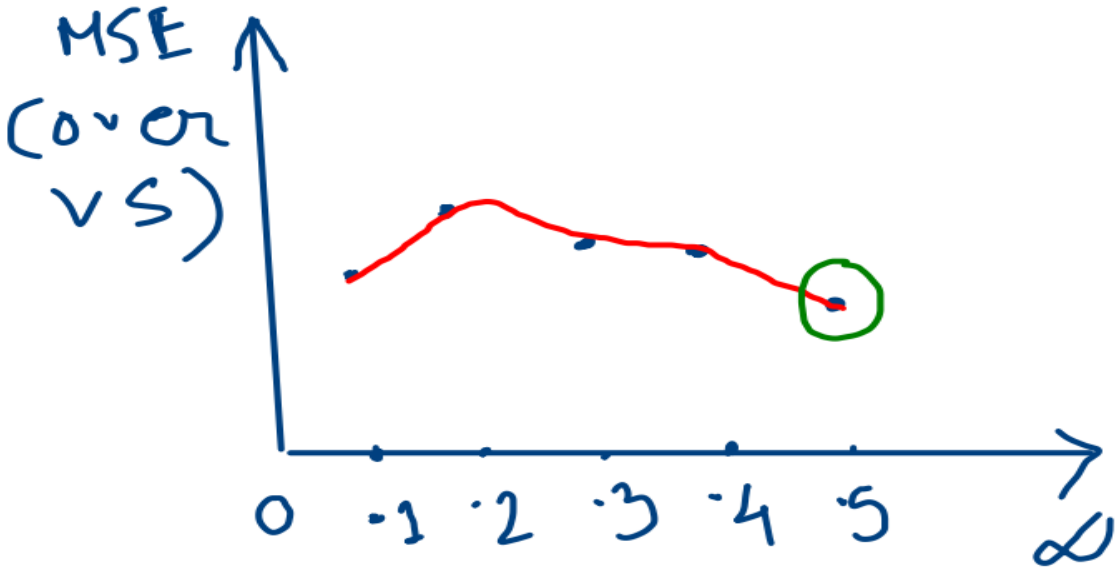
Training, Test, Validation Set

- **Training set (TS)**: samples/patterns are used to train the model (by updating the parameters).

- **Validation set (VS):** samples/patterns are used to evaluate the performance of models with different values of hyperparameters (hyperparameter tuning/model configuration). It's also used to detect overfitting during the training stages.

- **Test set (TeS):** samples/patterns are used to evaluate the final performance (Mean square error) of the models (after hyperparameter tuning).

Example of Hyperparameter Tuning



TS → 20%

VS → 10%

Calculate MSE over 70% (TES)
using $\alpha = .5$

Final Result

Training samples (%)	α	Train MSE	Test MSE
10%	.5	?	?
40%	.6	?	?
60%	.3	?	?
80%	.8	?	?

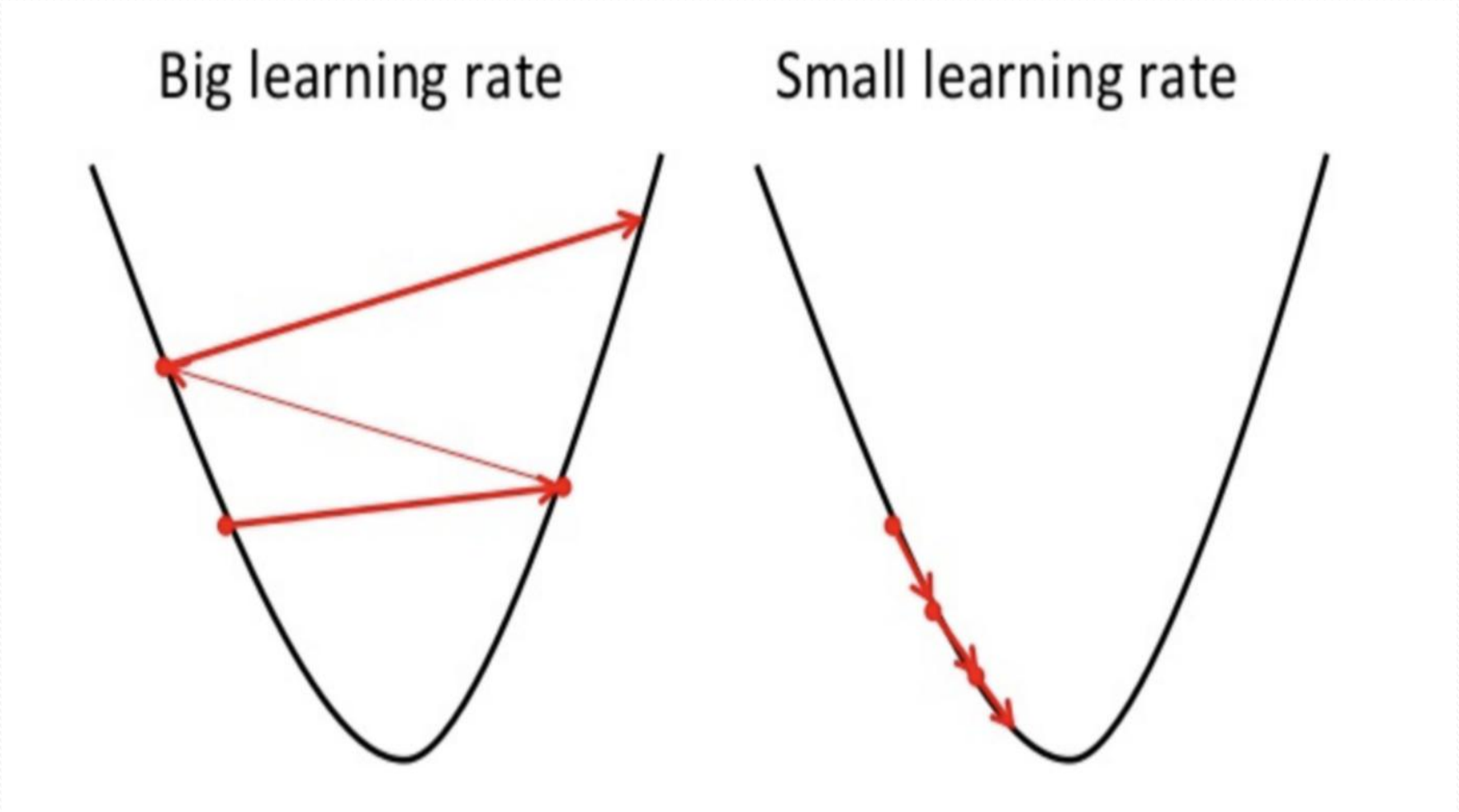
↓

After Convergence

Some observations

- Is there any affect of initial parameter values in performance?
- Is there any affect of different random set of training samples (considering same percentage) in performance of the model?
- How to check these issues during experimentation?

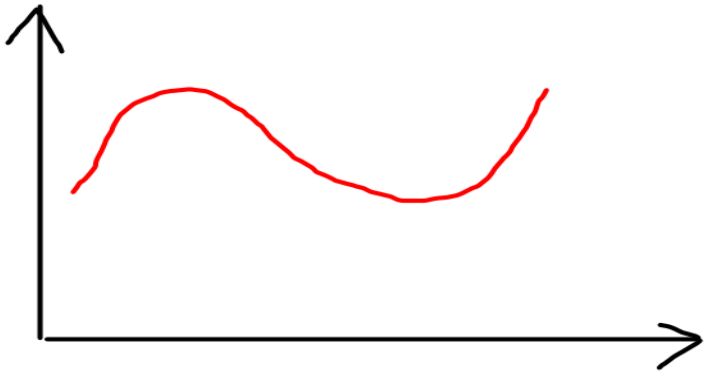
Setting of learning rate



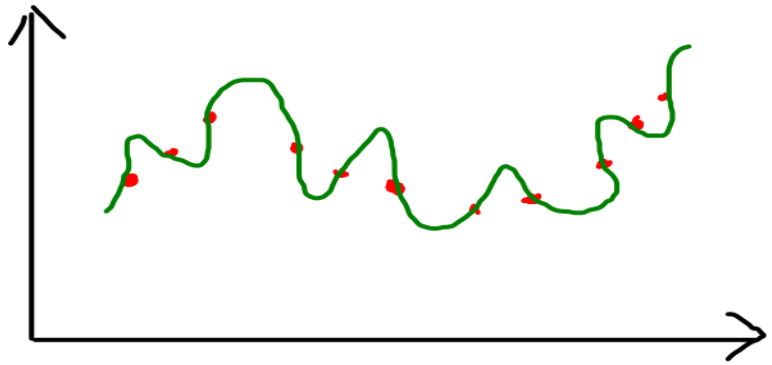
Overfitting and Underfitting

- Overfitting: model performs well using training samples but does not perform well on the testing samples. The validation set is also used to detect overfitting during the training stages.

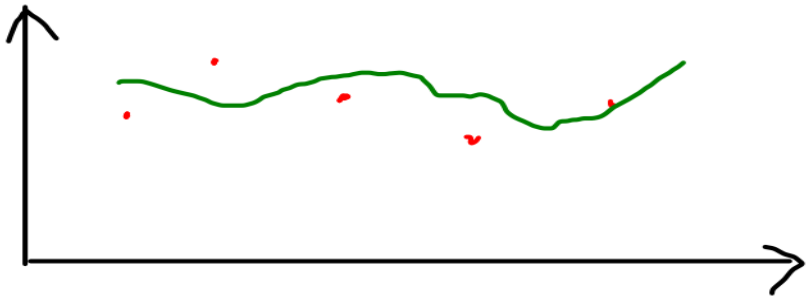
- Underfitting: model does not perform well on training samples as well as testing samples (may be due to very insufficient training samples).



→ actual curve
(we have training samples; not exact polynomial structure)

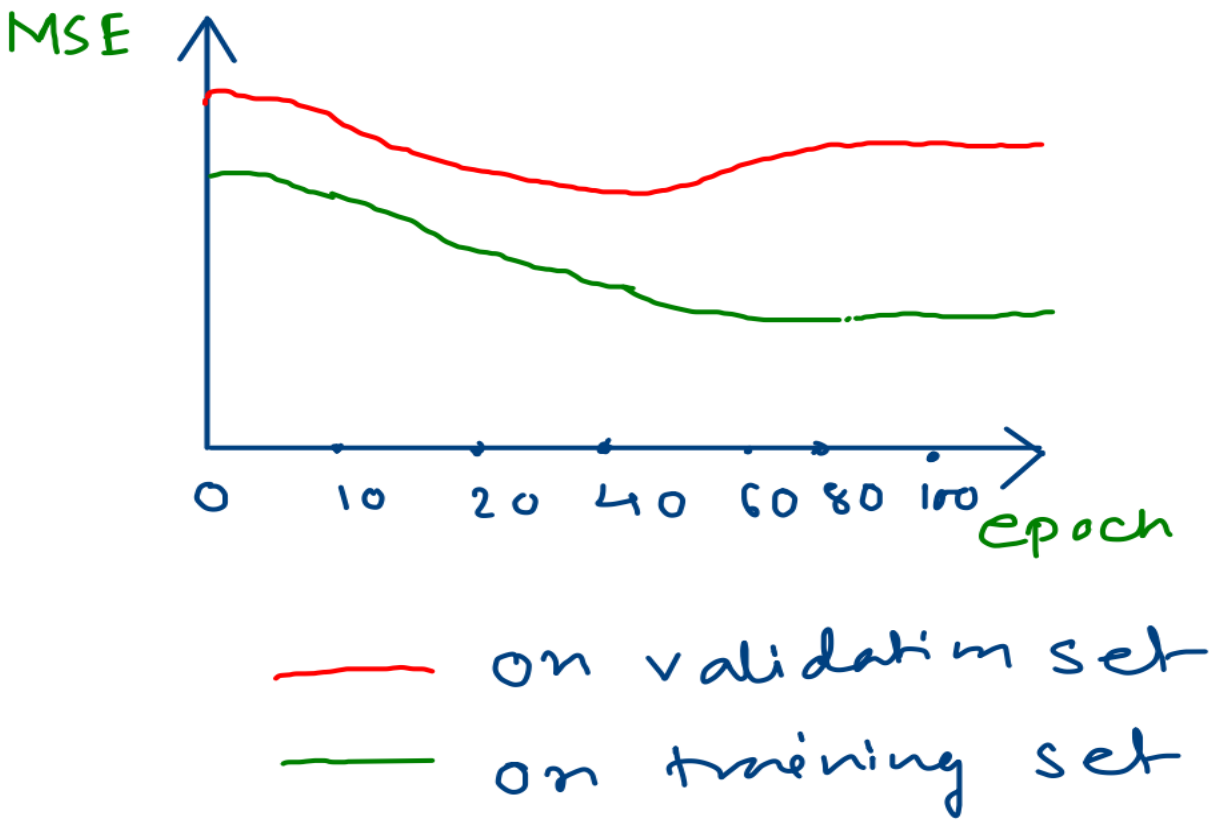


→ overfitting
→ training error in zero
→ but not able to fit the curve



→ underfitting
→ poor performance in training / testing

Checking overfitting



Feature Scaling (Normalization)

- Feature scaling is a method used to normalize the range of independent variables or features of data.
- It can speed up the gradient descent.
- Absolute maximum: Find maximum value in the dataset (considering features only) and divide all the features by the maximum value.

x_1	x_2	y
1·2	200000	1·2
1·3	400 00	6
1·4	50 000	1·2
2	10 0000	1·6

Logistic Regression

Machine Learning (CS 306)

Instructor: Dr. Moumita Roy

Teaching Assistants: Indrajit Kalita, Veronica Naosekpam

Email-ids:

moumita@iiitg.ac.in

veronica.naosekpam@iiitg.ac.in

indrajit.kalita@iiitg.ac.in

Mobile No: +91-8420489325 (only for emergency quires)

Why use logistic regression?

- There are many important research topics for which the dependent variable is "limited."
- For example: voting, spam or not spam, and participation data is not continuous.
- Binary logistic regression is a type of regression analysis where the dependent variable is a dummy variable: coded as 0 (did not vote) or 1(did vote)

Classification

- Email: Spam/not spam
- Land-cover: Water-cover/not water-cover
- Customer Behavior Prediction: Sad/Happy
- Tumor: Malignant/not Malignant

$Y \in \{0,1\}$

- coded as binary dependent variable (two-class problem)

  0→Negative class

  1 →Positive class

Linear Regression (Revisit)

$$x_i = [x_{i,1} \quad x_{i,2} \quad x_{i,3} \quad x_{i,4} \quad \cdots \quad x_{i,n}]$$

$$h(x_i) = \sum_{j=0}^{n} w_j x_{i,j} \qquad x_{i,0} = 1$$

$$w_j = [w_0 \quad w_1 \quad w_2 \quad \cdots \quad w_n]$$

$$h(x_i) = w^T x_i$$

$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

↓ cost function

$$w_j := w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h(x_i) - y_i)\, x_{i,j}$$

↓ parameter updation using GD

Important Points

- Can we use solution of linear regression problem directly for classification?

- It may be formulated like this (threshold the output):

$h(X_i) >= 0.5$, predict Y_i as 1

$h(X_i) < 0.5$, predict Y_i as 0

- Problem in the present approach:

I. The output $h(X_i)$ is unbounded

$h(X_i) > 1$ or < 0

II. In this case, fixing threshold at 0.5, is reasonable or not?

Solution: search a function which can bound the output between 0 to 1

Logistic Regression (Classification)

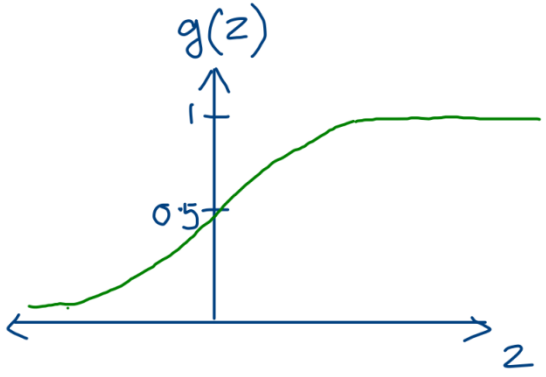
- Aim is to range the output: $0 <= h(X_i) <= 1$

$$h(x_i) = g(w^T x_i)$$

- $g(...)$ as sigmoid function/logistic function

$$g(z) = \frac{1}{1+e^{-z}}$$

$$h(x_i) = \frac{1}{1+e^{-w^T x_i}}$$



$g(z) \to 0$ if $z \to -\infty$

$g(z) \to 1$ if $z \to \infty$

Interpretation of Hypothesis output

- $h(X_i)$=estimated probability that $Y_i=1$ for a given X_i
- Suppose, $h(X_i)=0.7$

  interpret as, there is a 70% chance that X_i belongs to $Y_i=1$; alternatively there is a 30% chance that X_i belongs to $Y_i=0$ (if estimate in the scale of 100)

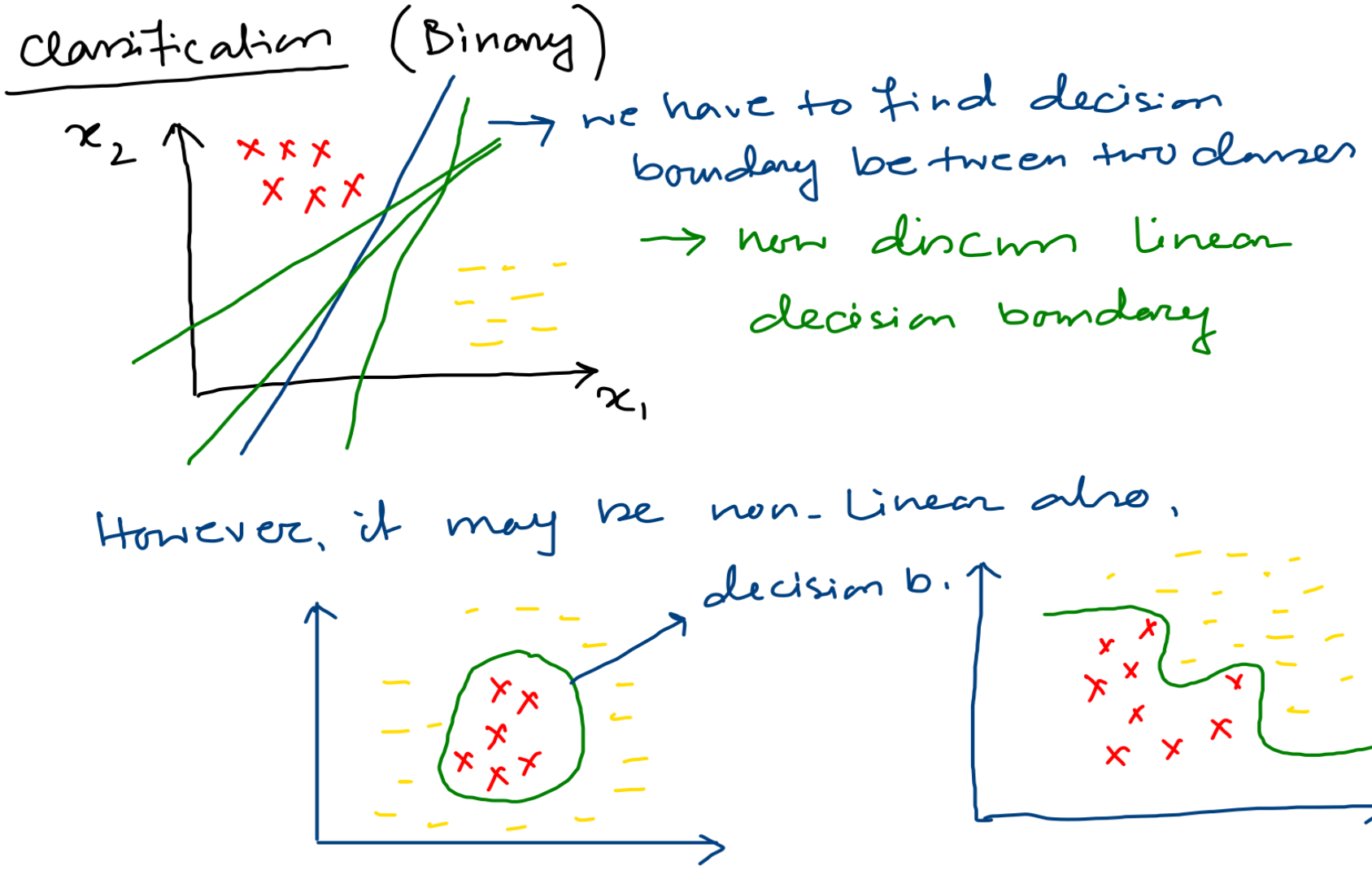
- $h_w(X)=P(Y=1|X;W)$ [probability that $Y=1$ for given X; parameterized by W]

- Since, Y is either 0 or 1

$P(Y=1|X;W)+ P(Y=0|X;W)=1$

$P(Y=0|X;W)=1-P(Y=1|X;W)$

Classification (Decision Boundary)



Simple case (two features)

$ax + by = c$

DB : Line

$w_1 x_1 + w_2 x_2 = w_0$

Aim: Find the values of parameters

$w = [w_0\ w_1\ w_2]$ using training

Samples $\langle x_i, y_i \rangle$

$w_1 x_1 + w_2 x_2 - w_0 \geq 0$ (One side of line)

↳ for such samples $Y = 1$ (positive class)

$w_1 x_1 + w_2 x_2 - w_0 < 0$ (Other side of line)

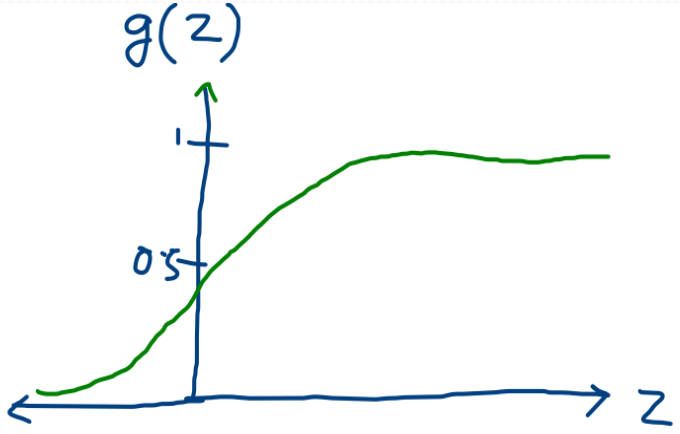
↳ for such samples $Y = 0$ (negative class)

Logistic Regression

$h(x_i) = g(w^T x_i)$

$h(x) = g(w^T x)$

$g(z) = \frac{1}{1+e^{-z}}$



→ $g(z) \geq 0.5$ when $z \geq 0$

Suppose, predict $y=1$; $h(x) \geq 0.5$

$w^T x \geq 0$ [Same as the previous example for two feature case]

predict $y=0$; $h(x) < 0.5$

$w^T x < 0$

Data set for classification

x_1	x_2	y
2	0.6	spam
1	5	not spam
6	2.8	not spam
3	1.4	spam

Format 1

class label
1
2
2
1

Format 2

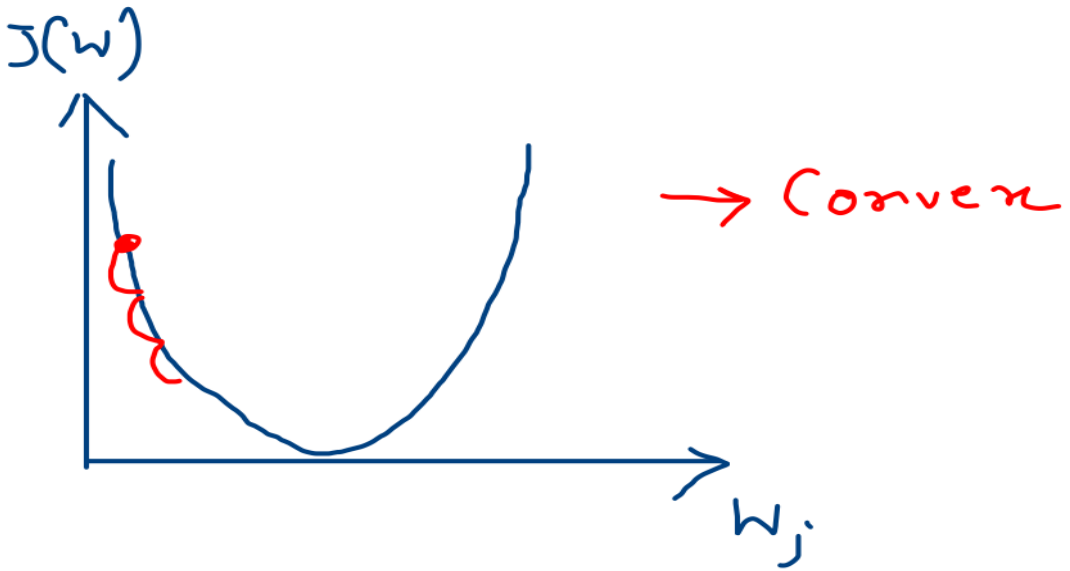
y
0
1
1
0

After re-write the class label

Logistic Regression (Cost Function)

Linear regression

$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(x_i) - y_i \right)^2$$



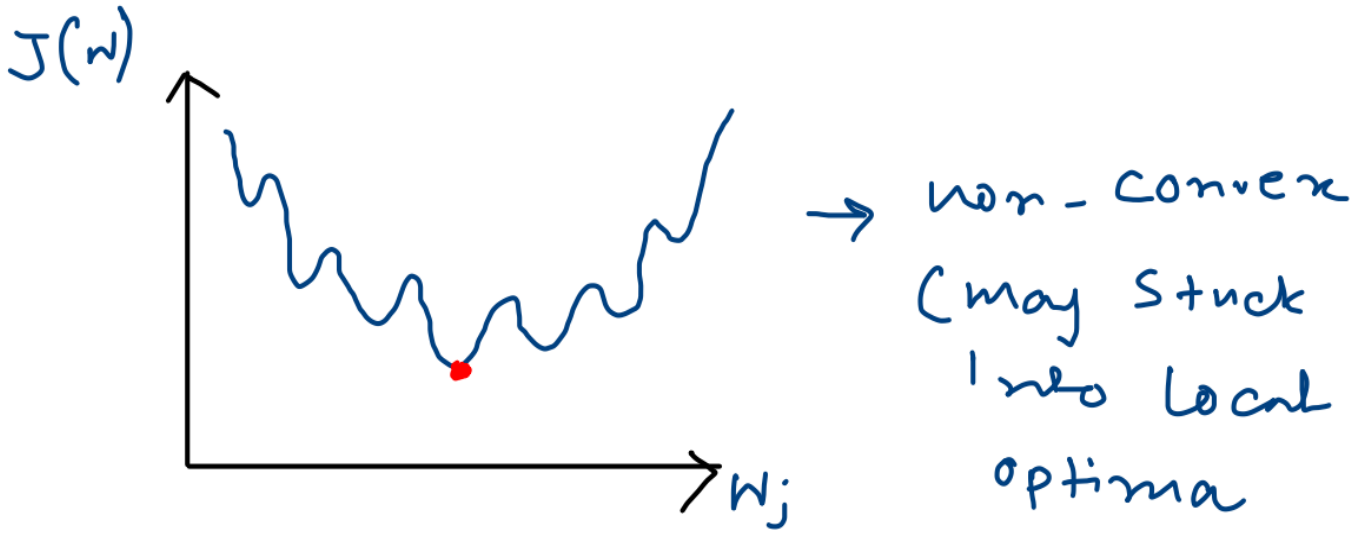
Logistic Regression

$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(x_i) - y_i\right)^2$$ → Is it convex included here

$$h(x_i) = g(w^T x_i)$$

$$= \frac{1}{1 + e^{-w^T x_i}}$$

} non-linearity involves



→ non-convex (may stuck into local optima

Logistic regression with MSE

$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(x_i) - y_i \right)^2$$

$$h(x_i) = g(w^T x_i)$$

$$g(z) = \frac{1}{1+e^{-z}}$$

$$g'(z) = g(z)\left(1 - g(z)\right)$$

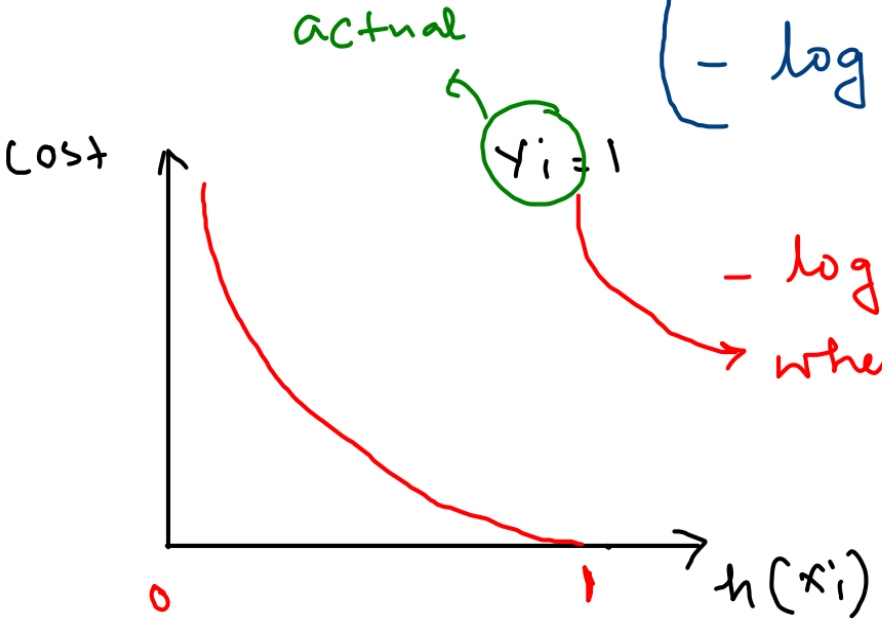
SGD: $w_j := w_j - \frac{\partial}{\partial w_j} J(w)$

$$w_j := w_j + \alpha \left(y_i - h(x_i) \right) x_{i,j} \underbrace{h(x_i) \cdot \left(1 - h(x_i)\right)}$$

Cost function for logistic regression

$$J(w) = \frac{1}{m} \sum_{i=1}^{m} \text{cost}\left(h(x_i), y_i\right)$$

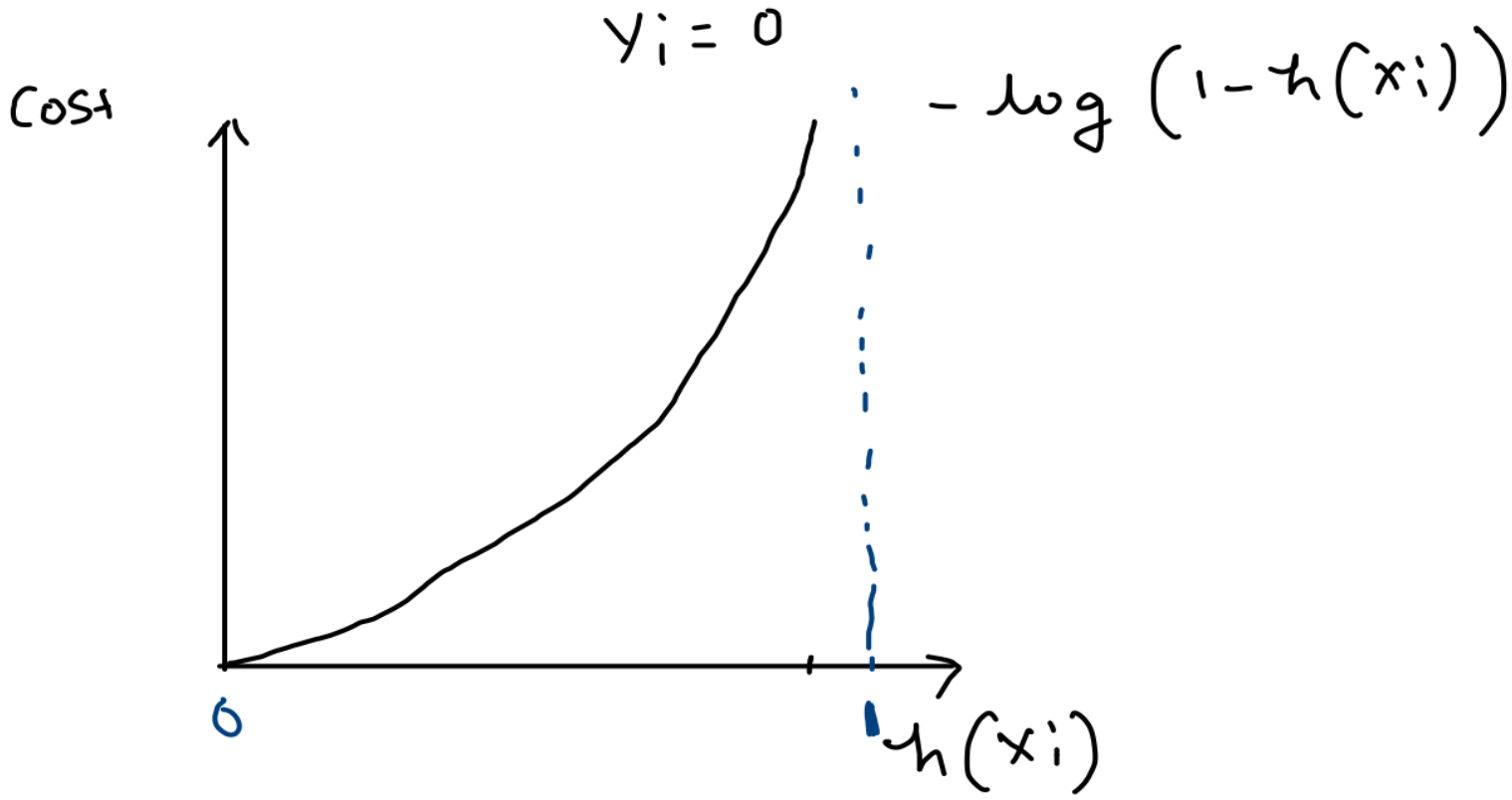
$$\text{Cost}\left(h(x_i), y_i\right) = \begin{cases} -\log\left(h(x_i)\right) & \text{if } y_i = 1 \\ -\log\left(1 - h(x_i)\right) & \text{if } y_i = 0 \end{cases}$$



$-\log(h(x_i))$ when, $h(x_i) = 1$ (predicted)

$\text{cost}(h(x_i), y_i) = 0$ if $h(x_i) \to 0$

cost $\to \infty$



if $y_i = 0$, $h(x_i) = 0$, Cost = 0

$h(x_i) \to 1$

Cost $\to \infty$

Simplified Cost function

$$\text{Cost}\left(h(x_i), y_i\right)$$

$$= -y_i \log\left(h(x_i)\right) - (1-y_i) \log\left(1 - h(x_i)\right)$$

$$y_i = 0 \text{ or } 1$$

when $y_i = 0$

$$\text{Cost}\left(h(x_i), y_i\right) = -\log\left(1 - h(x_i)\right)$$

when $y_i = 1$

$$\text{Cost}\left(h(x_i), y_i\right) = -\log\left(h(x_i)\right)$$

Cost function for logistic regression

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}\left(h(x_i), y_i\right)$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left[-y_i \log\left(h(x_i)\right) - (1-y_i)\log\left(1-h(x_i)\right)\right]$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log\left(h(x_i)\right) + (1-y_i)\log\left(1-h(x_i)\right)\right]$$

→ minimize $J(w)$ to obtain

$w = [w_0\ w_1\ \ldots\ w_n]$

$h(x_i) = \frac{1}{1+e^{-w^T x_i}}$

GD: $w_j := w_j - \alpha\left(h(x_i) - y_i\right) x_{i,j}$

→ Same updation rule as linear regression

Algorithmic representation for gradient descent

Step 1: Randomly initialize $w_0, w_1, ..., w_n$ and learning rate

Step 2: Calculate $h(X_i)$ for all training samples

Step 3: Calculate $J_{itr}(W)$ for itr^{th} epoch

Step 4: Update all parameters $w_0, w_1, ..., w_n$

Step 5: Repeat steps 2-5 until

$$| J_{itr}(W) - J_{itr-1}(W)| > \rho \text{ or number of epochs} > no$$

Step 6: Apply threshold to take final decision for binary classification

Step 6: Report the value of $J_{itr}(W)$ and classification accuracy

ρ: small value (may be *0.0001*)

***no*: sufficiently large so that model can learn**

Can we stop using validation set?

Epoch/training step: one time updating considering all the training samples

GD for Logistic regression

Training samples

x_1	x_2	Class Label
1.2	3	1
3	4	2
6	7	1
9	10	1

$W = [1 \quad 1 \quad 1]$

$\alpha = .1$

$$g(z) = \frac{1}{1+e^{-z}}$$

1. Normalized the dataset
2. Split train set / Test set / VS

TS

x_0	x_1	x_2	y
1	1.2	3	1
1	3	4	0
1	6	7	1
1	9	10	1

Epoch 1 $h(x_i) = g(w^T x_i) = g(w_0 x_0 + w_1 x_1 + w_2 x_2)$

TS1 : $h(x_1) = g(w_0 \times 1 + w_1 \times 1 \cdot 2 + w_2 \times 3)$

$= \cdot 6$ (not exact here / you need to calculate the exact one)

TS2 : $h(x_2) = \cdot 3$

TS3 : $h(x_3) = \cdot 5$

TS4 : $h(x_4) = \cdot 2$

Cost function → MSE (non-convex)

→ log loss (Convex)

MSE

$$J(w) = \frac{1}{2m} \sum (h(x_i) - y_i)^2$$

$$J(w) = \frac{1}{2m}\left[(\cdot 6 - 1)^2 + (\cdot 3 - 0)^2 + (\cdot 5 - 1)^2 + (\cdot 2 - 1)^2 \right]$$

$$w_j := w_j - \alpha \frac{\partial J(w)}{\partial w_j} \longrightarrow h(x_i) \text{ includes sigmoid function}$$

$$w_j := w_j + \alpha \frac{1}{m} \sum \left[(y_i - h(x_i)) x_{i,j} \right.$$

$$\left. h(x_i)(1 - h(x_i)) \right]$$

$$w_0 := w_1 + \alpha \frac{1}{m} \sum \left[\left((1 - \cdot 6) \times 1 \times \cdot 6 \times (1 - \cdot 6) \right) + \right.$$

$$w_1 := \quad (\quad w \quad) + (\quad t \quad)$$

$$w_2 := \quad + (\quad t \quad) \left. \right]$$

log loss

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log\left(h(x_i)\right) + (1-y_i)\log\left(1-h(x_i)\right)\right]$$

$$w_j := w_j - \alpha \frac{\partial J(w)}{w_j}$$

$$w_j := w_j + \alpha \frac{1}{m}\sum_{i=1}^{m}\left(y_i - h(x_i)\right) x_{i,j}$$

$w_0 = ?$

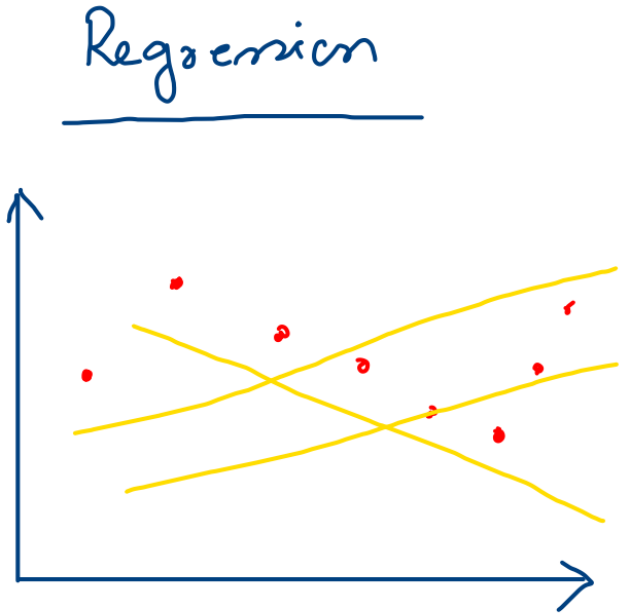
$w_1 = ?$

$w_2 = ?$

Repeat the training epoch by epoch until convergence (?)

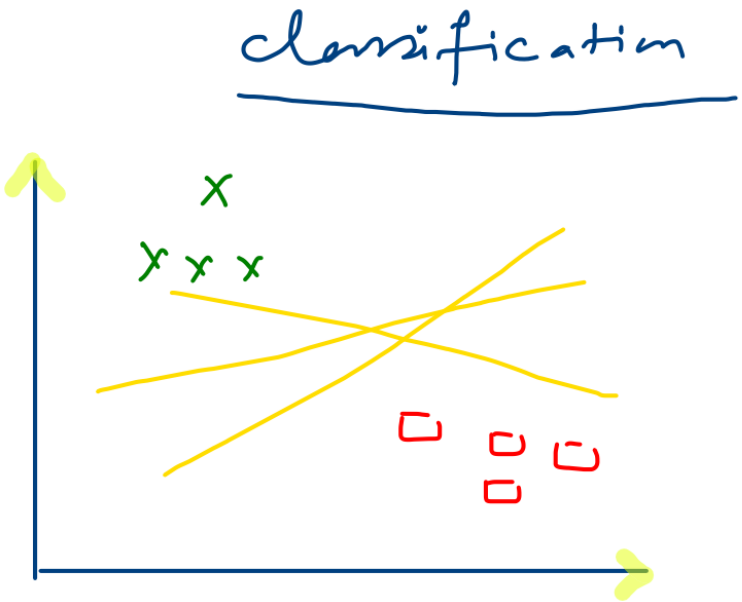
→ Finally report test accuracy after convergence of training

Regression vs. Classification



Linear Relation between independent and dependent variables

Try to find best fit one by minimizing cost function



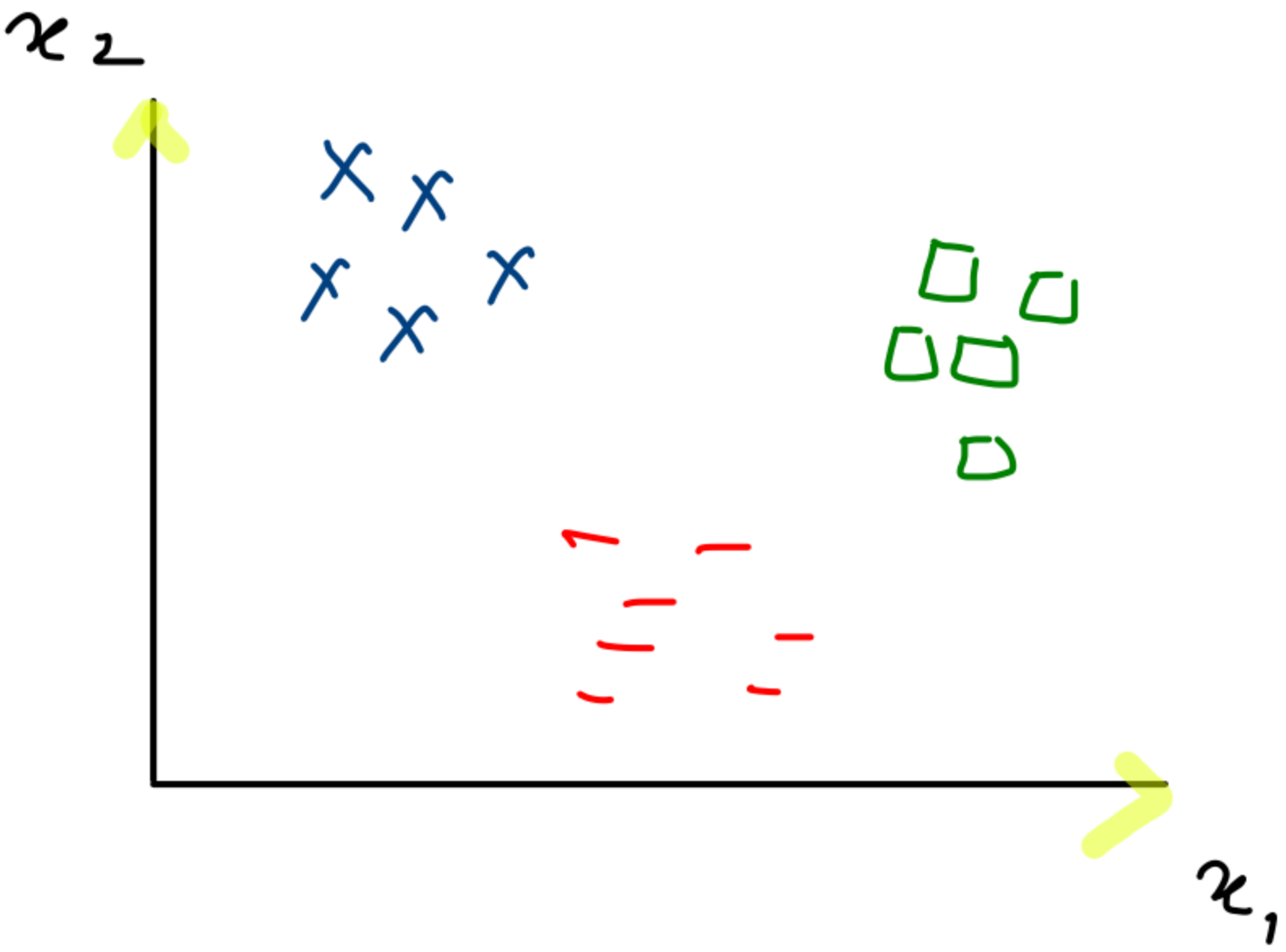
Data set is linearly separable

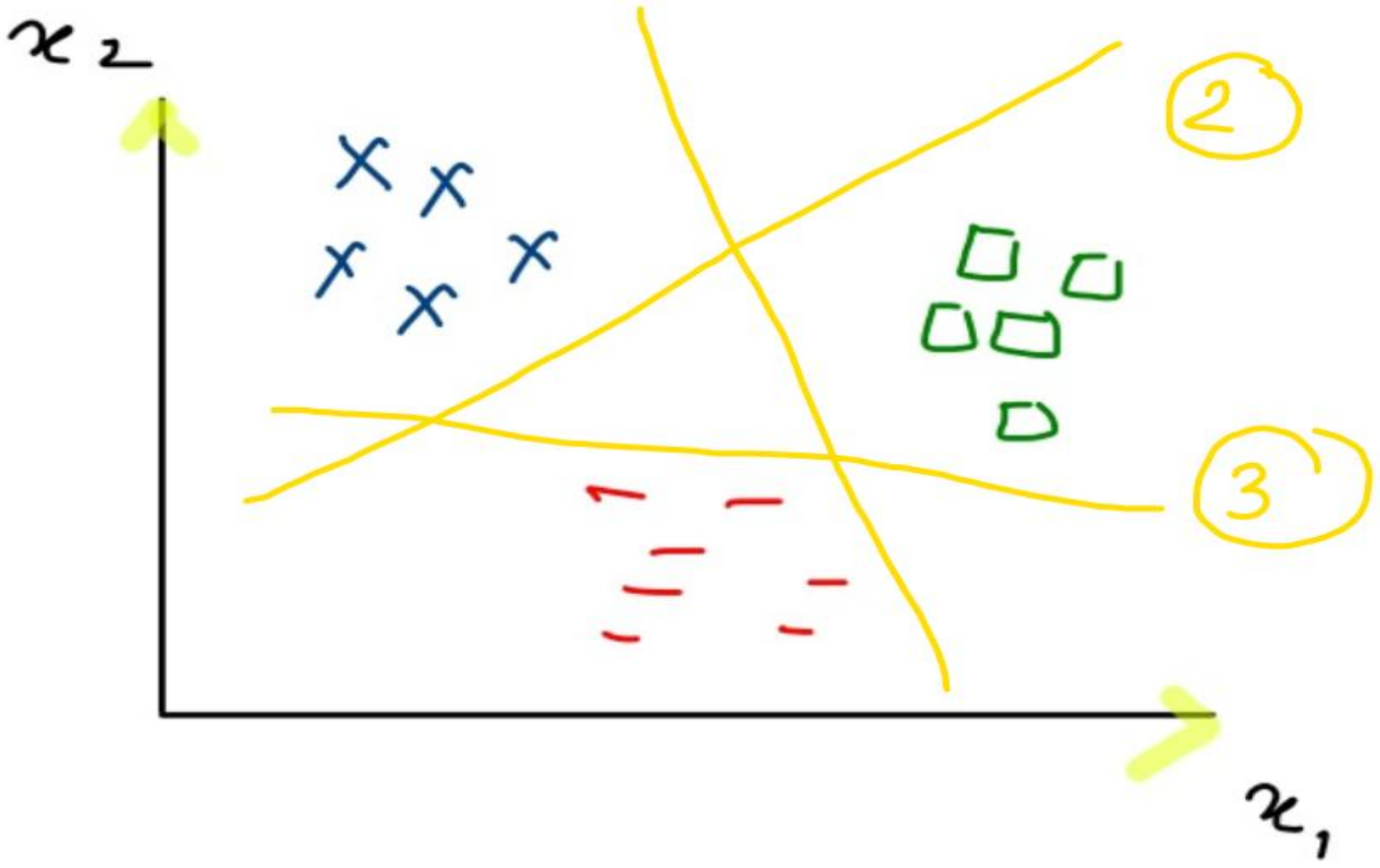
Try to find linear DB by minimizing cost function

Consider binary classification

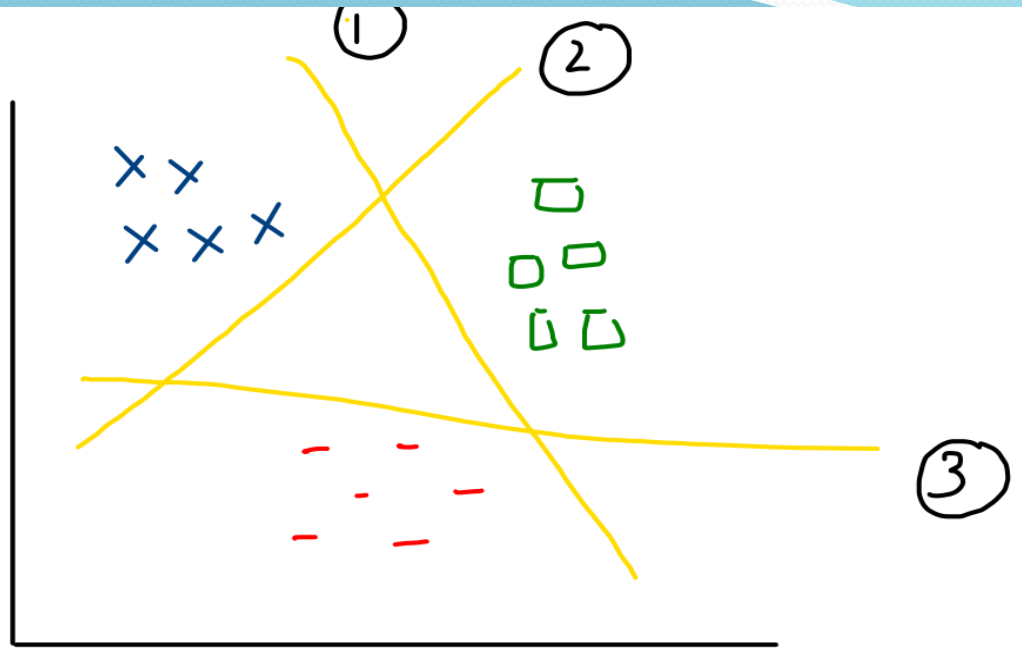
Multi-class Logistic Regression

x_1	x_2	class label	
2	3	1	
4	5	2	
6	7	3	→ Three classes
1·2	3	1	

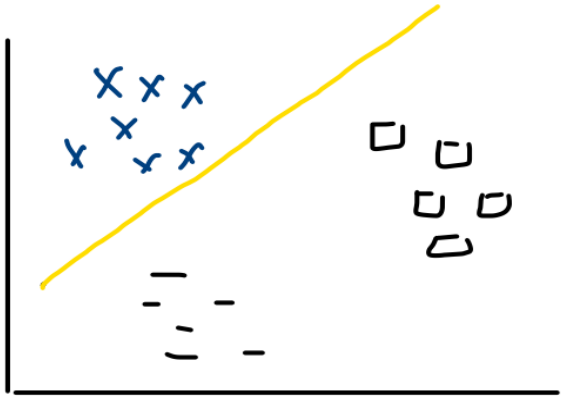

x_2
x_1



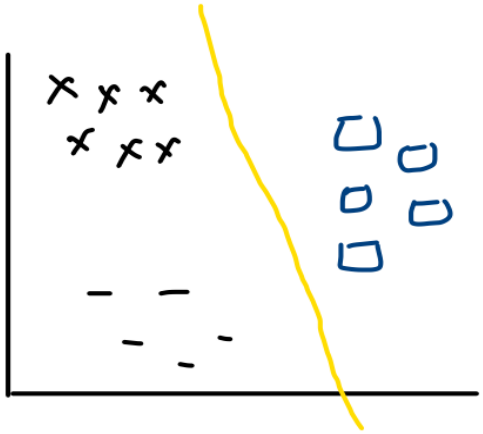
One vs. All



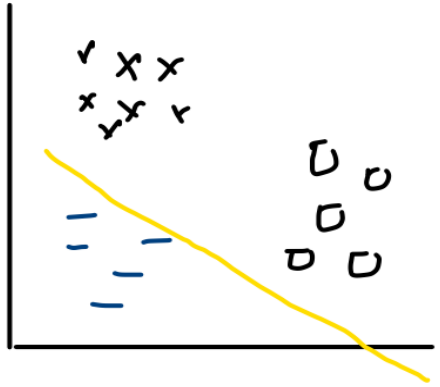
Multi-class, linearly separable



Classifier model 1



Classifier model 2



Classifier model 3

Dataset preparation??

Original Dataset (Training samples)

x_1	x_2	class label
2	3.4	1
6	1.2	2
3	4.5	2
6	1	3

Dataset to training classifier 1

x_1	x_2	Y
2	3.4	1
6	1.2	0
3	4.5	0
6	1	0

Same way prepare for classifier 2 and 3

m = number of pattern

$X_1 \quad X_2 \quad \cdots \quad X_i \quad \cdots \quad X_m$

n = number of features

$X_i = [x_{i,1} \quad x_{i,2} \quad \cdots \quad x_{i,j} \quad \cdots \quad x_{i,n}]$

c = number of classes

$k \in \{1, 2, \ldots, c\}$

One-vs-all

Train a logistic regression model $h^k(x_i)$ for each class k to predict probability that $y_i = k$ for a given x_i

→ Testing (Class label for unknown pattern)??

Testing phase

$x_i \rightarrow$ class label ?? $\rightarrow$ predicted class label in K

$h^1(x_i) \rightarrow$

$h^2(x_i) \rightarrow$

$h^3(x_i) \rightarrow$

$\vdots$

$h^K(x_i) \rightarrow$

$\vdots$

$h^C(x_i) \rightarrow$

$\Big\}$ – Choose K $\max_K h^K(x_i)$

↳ parameters are already estimated for all cases.

Some observations

- Is there any affect of initial parameter values in performance? (Hyperparameter Tuning/Validation Set)
- ***Is there any affect of different random set of training samples (considering same percentage) in performance of the model?***
- How to check these issues during experimentation?

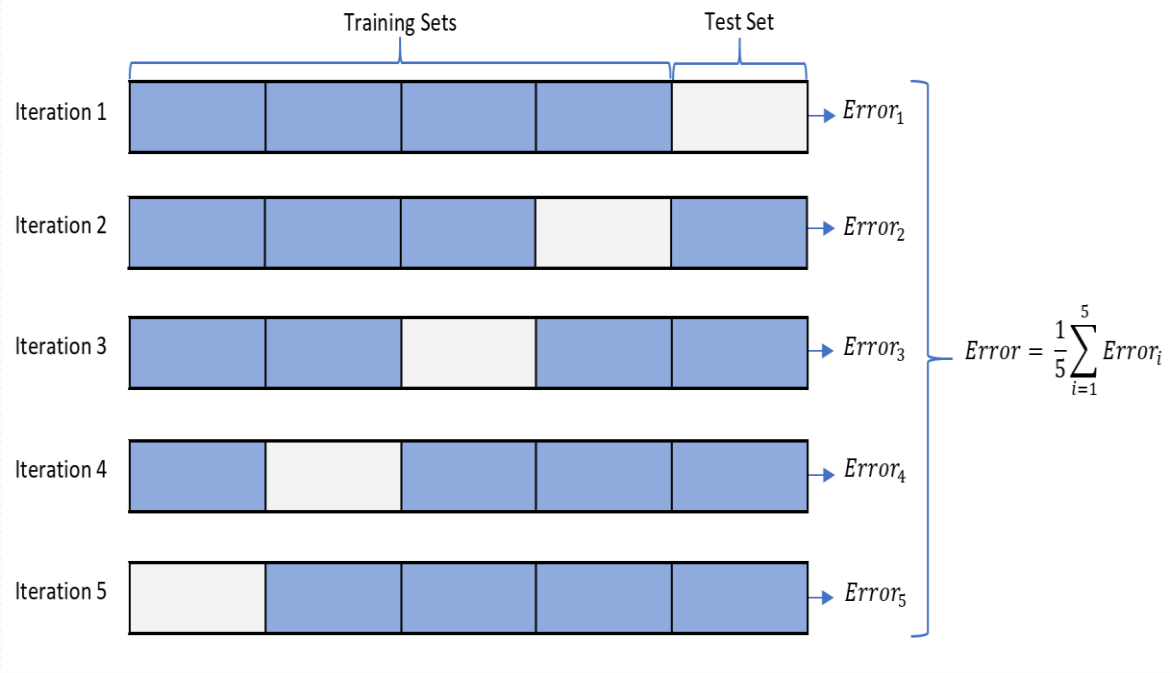
K-fold Cross-Validation

- Cross-validation is a statistical method used to estimate the skill of machine learning models.
- It is a resampling procedure used to evaluate the performance of learning models on different train/test split.
- The model is expected to perform well in generalization when used to make predictions for the samples not used during the training of the model.
- Here, the dataset is divided in train set and test set only.

Procedure

- Shuffle the dataset randomly (Why?)
- Split the dataset into k groups/folds (approximately equal size)
- For each group:
  - Take the group as a hold out or test data set
  - Take the remaining groups as a training data set
  - Fit a model on the training set and evaluate it on the test set
  - Retain the evaluation score and discard the model
- Report the performance score of the model on each group (test set) and average of all the performance score

Validation set??


Training Sets
Test Set
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
$Error_1$
$Error_2$
$Error_3$
$Error_4$
$Error_5$
$Error = \frac{1}{5}\sum_{i=1}^{5} Error_i$

Observation

- Some portion of training set can be used for validation set (for example 10%)
- The value of K: mostly use 5 or 10 for experimentation
- Each pattern is assigned to an individual group and stays in that group for the duration of the procedure
- This means that each sample is given the opportunity to be used in the hold out set/test set 1 (one) time and used to train the model *K-1* times.

Confusion Matrix for Classification

- Confusion Matrix is used to know the performance of a Machine learning classification. It is represented in a matrix form.

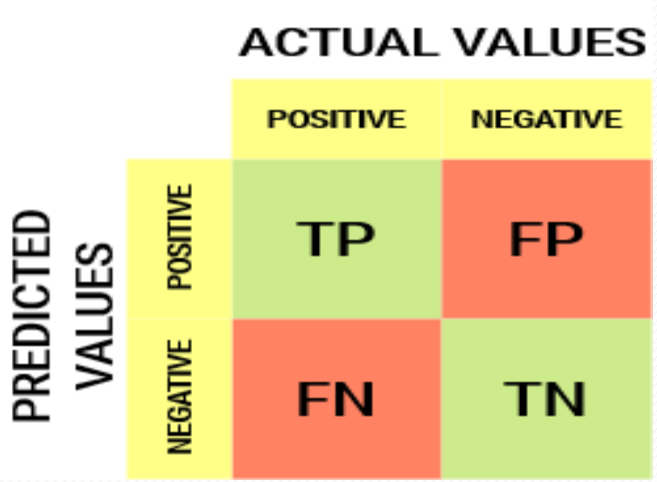
- Confusion Matrix gives a comparison between actual and predicted values.

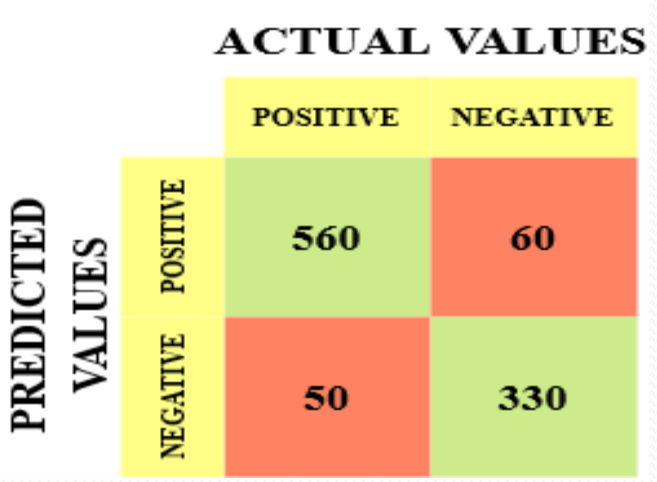
- The confusion matrix is a N x N matrix, where N is the number of classes.

  For 2 class-problem ,we get 2 x 2 confusion matrix.

  For 3 class-problem ,we get 3 X 3 confusion matrix.

Confusion Matrix for Classification

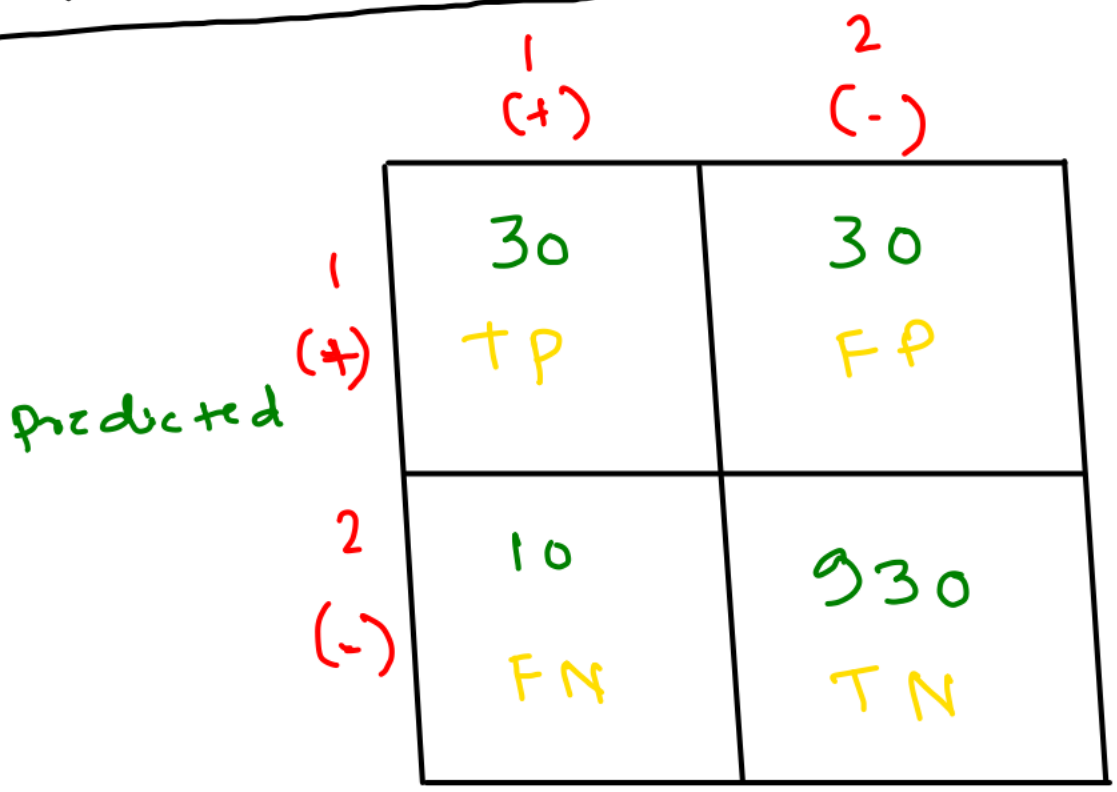




- The target variable has two values: **Positive** or **Negative**
- The **columns** represent the **actual values** of the target variable
- The **rows** represent the **predicted values** of the target variable

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Confusion matrix



Observation

$$\text{Accuracy} = \frac{30 + 930}{30 + 30 + 10 + 930} = .96$$

good one but not provide the complete idea

$$\text{Accuracy}_{\text{class 1}} = \frac{30}{40} = .75$$

$$\text{Accuracy}_{\text{class 2}} = \frac{930}{960} = .96$$

Suppose, class 1 is more important than class 2

Precision vs. Recall

- **Precision:** Out of all the positive predicted samples how many are truly positive.

$$Precision = \frac{TP}{TP + FP}$$

The precision value lies between 0 and 1.

- **Recall:** How many of the actual positive cases we were able to predict correctly with our model.

$$Recall = \frac{TP}{TP + FN}$$

The recall value lies between 0 and 1.

Precision vs. Recall (Multiclass)

- Calculate precision and recall for each of the classes
- *precision* for the Cat class is the number of correctly predicted Cat photos (4) out of all predicted Cat photos (4+3+6=13), which amounts to 4/13
- the *recall* for Cat is the number of correctly predicted Cat photos (4) out of the number of actual Cat photos (4+1+1=6), which is 4/6

		True/Actual		
		Cat	Fish	Hen
Predicted	Cat	4	6	3
	Fish	1	2	0
	Hen	1	2	6

Some issues related to training samples

- Training samples in one class are significantly fewer than the samples in other classes (class imbalance)

- Sufficient training samples are not available (data scarcity)

Artificial Neural Networks (ANN)

Machine Learning (CS 306)

Instructor: Dr. Moumita Roy

Teaching Assistants: Indrajit Kalita, Veronica Naosekpam

Email-ids:

moumita@iiitg.ac.in

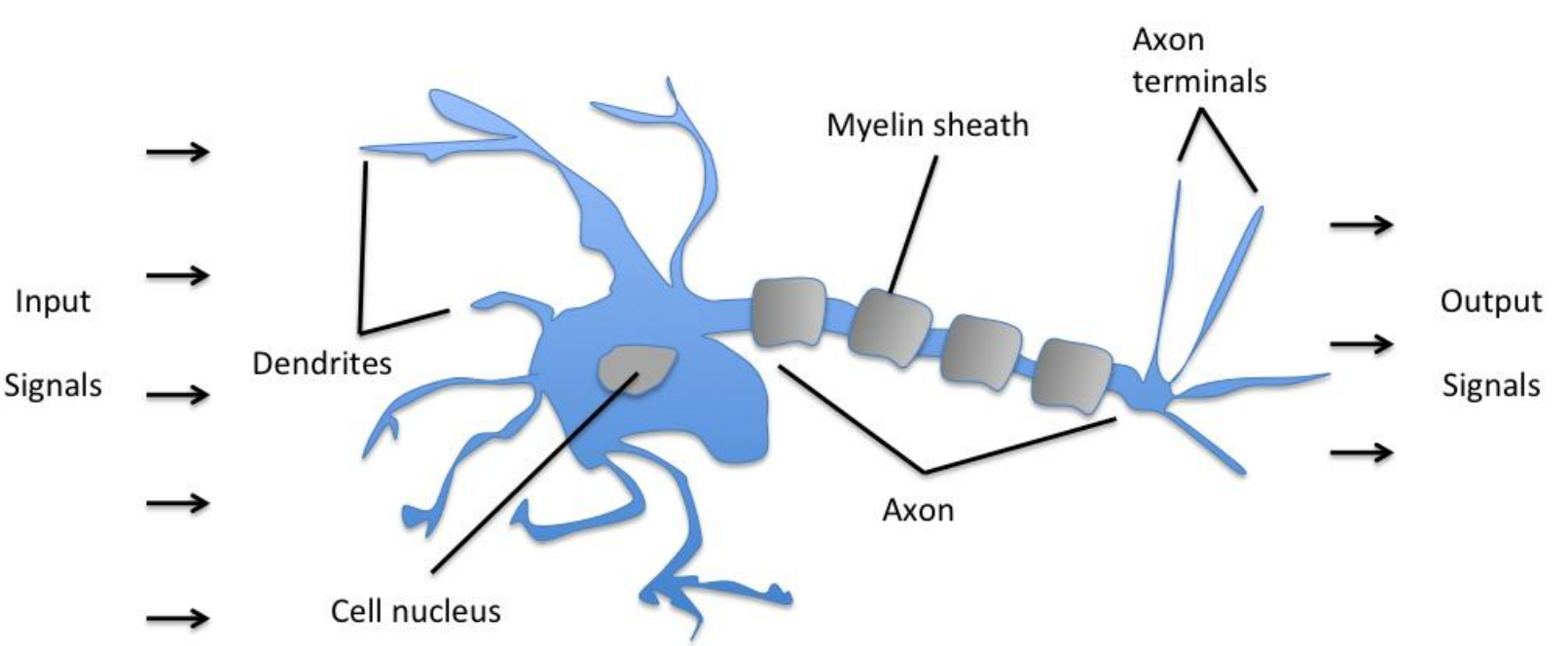
veronica.naosekpam@iiitg.ac.in

indrajit.kalita@iiitg.ac.in

Mobile No: +91-8420489325 (only for emergency quires)

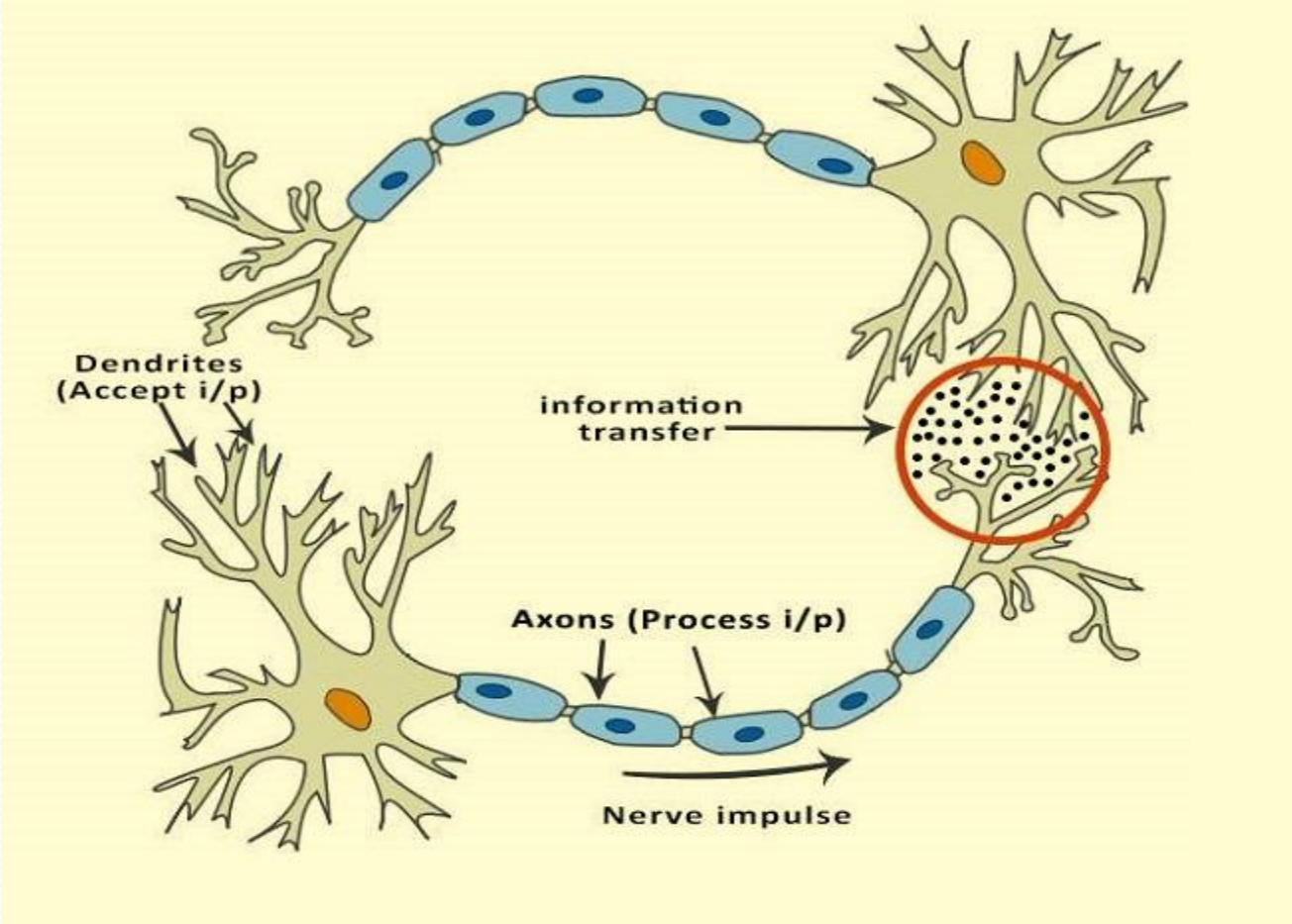
Reference: https://www.cse.iitm.ac.in/~miteshk/CS6910/Slides/Lecture2.pdf

Biological Neuron



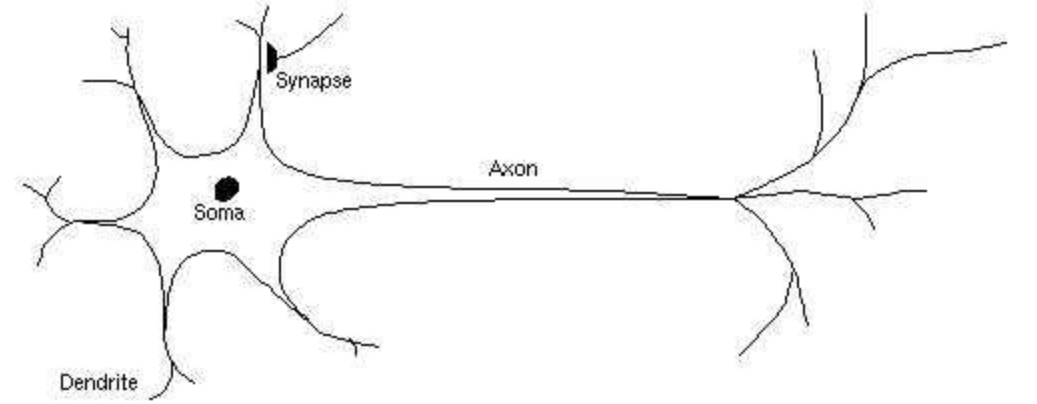
Schematic of a biological neuron.

Biological Neural network



Working principle of biological neuron

- **Dendrite**: receives signals from the axons of other neurons
- **Synapses**: placed between the dendrite and axons, signals are modulated in various amounts (synaptic weights)
- **Soma/Cell body**: Signals from the dendrites are joined and passed on (functional component of the neuron)
- **Axon**: carries nerve impulses away from the cell body. It connects one neuron with other neurons



Formal Definition

A artificial neural network is a massively parallel distributed processor, made up of a simple processing unit, which can store the knowledge and making it available for use.

It resembles the brain in two aspects:

1. Knowledge is acquired by the network from its environment through the learning process.
2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

Characteristic of ANN

- Massive parallel
- Learning ability
- Adaptability
- Distributed representation
- Fault tolerance

McCulloch-Pitts (MP) Neuron (1943)

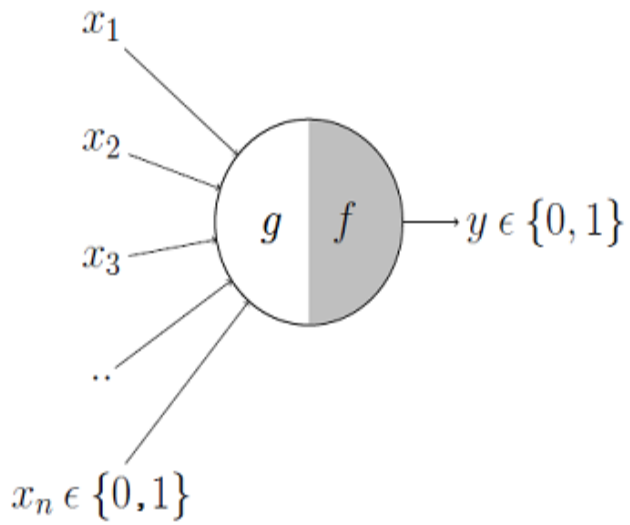
- McCulloch (neuroscientist) and Pitts (logician) proposed a highly simplified computational model of the neuron (1943)
- *g* aggregates the inputs and the function *f* takes a decision based on this aggregation (considering no inputs are inhibitory)

$$g(x_1, x_2, x_3, \cdots, x_n) = g(x) = \sum_{i=1}^{n} x_i$$

$$y = f(g(x)) = 1 \quad \text{if } g(x) \geq \theta$$

$$= 0 \quad \text{if } g(x) < \theta$$

θ = Thresholding parameter

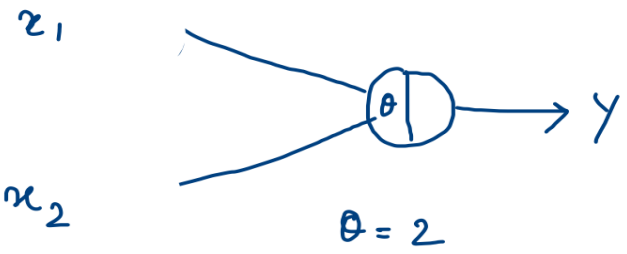


Boolean function representation

AND Function

Truth Table

	x_1	x_2	y
1.	1	0	0
2.	0	1	0
3.	1	1	1
4.	0	0	0



1. $g(x) = 1 + 0 = 1$

   $y = 0 \mid g(x) < \theta$

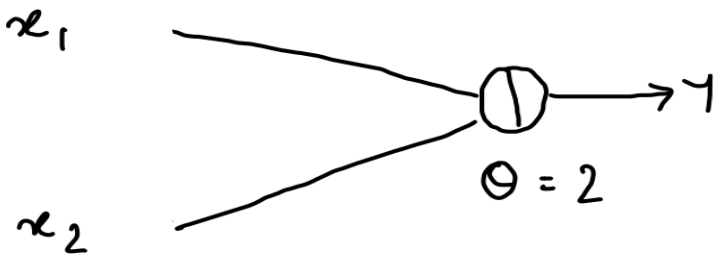
2. $g(x) = 0 + 1 = 1$

   $y = 0 \mid g(x) < \theta$

3. $g(x) = 0 + 0 = 0 \mid y = 0, g(x) < \theta$

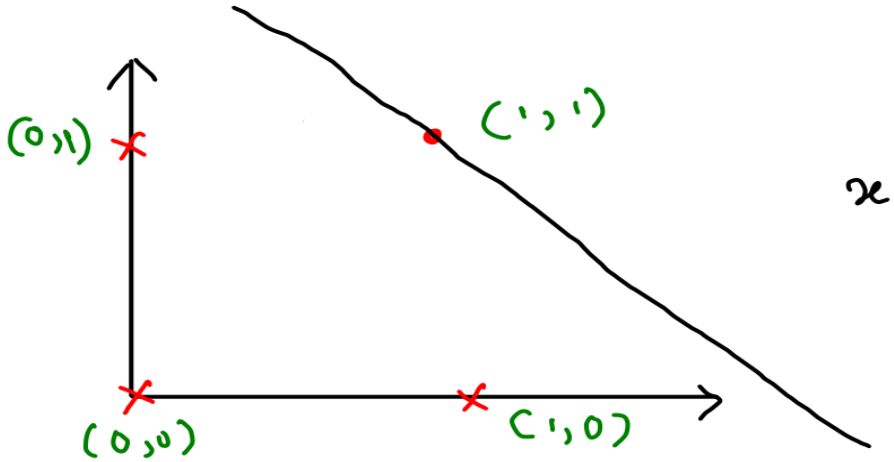
4. $g(x) = 1 + 1 = 2 \mid y = 1, g(x) \geq \theta$

Geometrical interpretation of the model



AND function

$x_1 + x_2 \geq 2 \rightarrow$ then $y = 1$
else $y = 0$



$x_1 + x_2 = \Theta$

$x_1 + x_2 = 2$

Check for other Boolean functions

- OR
- NAND
- NOR

→MP model can be used for linearly separable Boolean function

→What about real inputs?

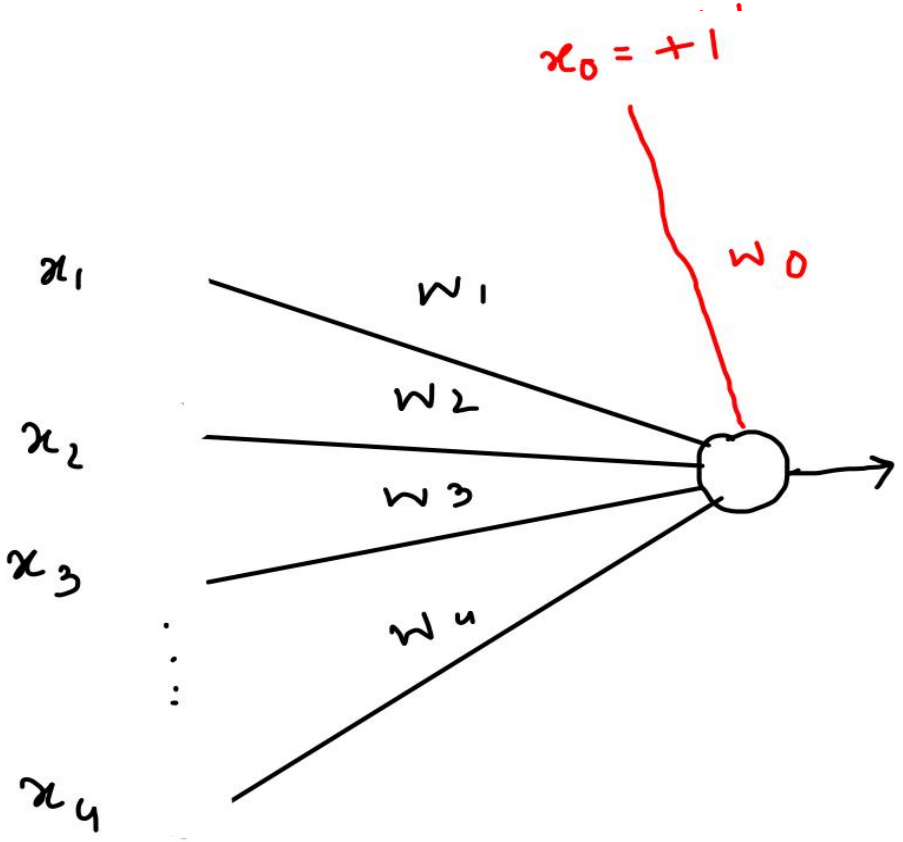
→ Always hand-coded the threshold?

→want to assign more weight (importance) to some inputs?

→functions which are not linearly separable?

Perceptron (Artificial Neuron)

- Frank Rosenblatt, an American psychologist, proposed the classical perceptron model (1958)
- Introduce numerical weights for inputs and a mechanism for learning these weights (same as synaptic weights in biological neuron)
- Inputs are no longer limited to Boolean values
- Refined and carefully analyzed by Minsky and Papert (1969) - their model is referred to as the perceptron model



$y = 1$ if $w_i x_i \geq \theta$

$= 0$ if $w_i x_i < \theta$

Rewrite:

$y = 1$ if $w_i x_i - \theta \geq 0$

$= 0$ if $w_i x_i - \theta < \theta$

Rewrite:

$x_0 = 1$

$w_0 = -\theta$

$$\begin{cases} y = 1 \text{ if } \sum_{i=0}^{n} w_i x_i \geq 0 \\ = 0 \text{ if } \sum_{i=0}^{n} w_i x_i < 0 \end{cases}$$

Observation

- A single perceptron can only be used to implement linearly separable functions (start with binary classification)
- However, the weights (including threshold) can be learned and the inputs can be real valued
- Revisit the Boolean function with perceptron and then start learning of weights

AND function

$y = 1 \; ; \; w^T x \geq 0$

$= 0 \; ; \; w^T x < 0$

x_0	x_1	x_2	y	
1	0	0	0	$\rightarrow w_0 < 0$ — (1) ✓
1	0	1	0	$\rightarrow w_0 + w_2 < 0$ — (2) $-2+1<0$ ✓
1	1	0	0	$\rightarrow w_0 + w_1 < 0$ — (3) $-2+1<0$ ✓
1	1	1	1	$\rightarrow w_0 + w_1 + w_2 \geq 0$ — (4) $-2+1+1 \geq 0$ ✗

Can we have any solution to these inequalities?

$w_0 = -2, \; w_1 = 1, \; w_2 = 1$

Final one, $w^T x = 0$

or $w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$

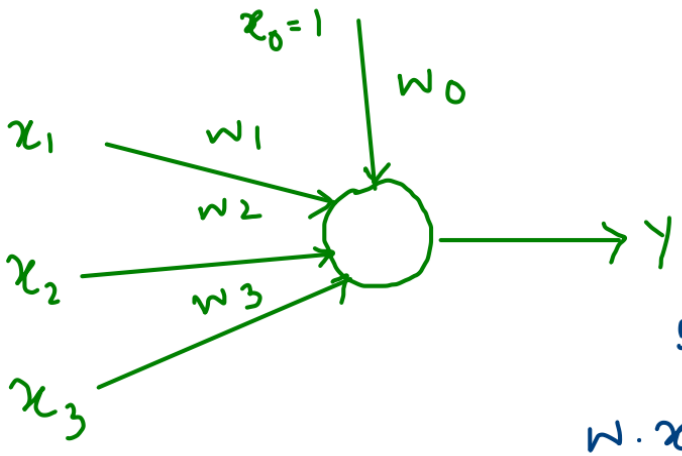
or $x_1 + x_1 = 2$

Same as MP neuron?

Perceptron Learning Algorithm

Algorithm: Perceptron Learning Algorithm

$P \leftarrow inputs \quad with \quad label \quad 1;$
$N \leftarrow inputs \quad with \quad label \quad 0;$
Initialize $\mathbf{w}$ randomly;
while $!convergence$ **do**
    Pick random $\mathbf{x} \in P \cup N$;
    if $\mathbf{x} \in P \quad and \quad \mathbf{w}.\mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
    end
    if $\mathbf{x} \in N \quad and \quad \mathbf{w}.\mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
    end
end
//the algorithm converges when all the inputs are classified correctly



$w = [w_0 \; w_1 \; w_2 \; w_3]$

$x = [x_1 \; x_2 \; x_3 \; x_4]$

$w \rightarrow$ randomly initialized

pick any pattern x

$$w \cdot x = w^T x = \sum_{i=0}^{n} w_i x_i$$

we want;

$y = 1$; $w^T x \geq 0$

$y = 0$; $w^T x < 0$

interested to find line

$$w^T x = 0$$

train epoch by epoch until all the training samples are correctly classified

There is a proof of convergence of perceptron learning algorithm.

AND function

	x_0	x_1	x_2	y
TS1	1	0	0	0
TS2	1	0	1	0
TS3	1	1	0	0
TS4	1	1	1	1

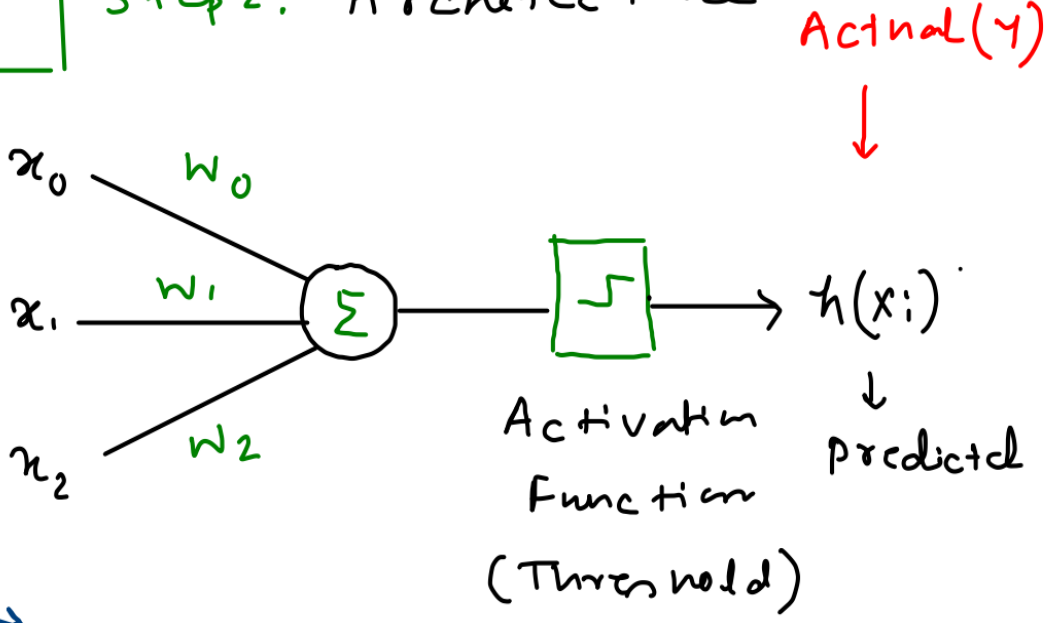
Dataset

Step 1. Randomly initialized weights (Small value)

$W = [W_0 \quad W_1 \quad W_2]$

$= [0.1 \quad 0.1 \quad 0.1]$

Step 2: Architecture



(SLP)

Single perceptron / Single layer Perceptron

Step3: Learning (perceptron learning algorithm) $\alpha = 1$

Epoch 1

Training sample 1:

$Y < 0$ (N)

$$h(TS1) = g(w_0 x_0 + w_1 x_1 + w_2 x_2) > 0$$

$$= g(0.1 \times 1 + 0.1 \times 0 + 0.1 \times 0)$$

$$= g(0.1) = 1$$

$$w_0 = w_0 - x_0 = w_0 + \alpha(y - h(x_i))x_0 = 0.1 - 1$$

$$= w_0 + 1(0 - 1)x_0 = -0.9$$

$$w_1 = w_1 - x_1 = 0.1 - 0 = 0.1$$

$$w_2 = w_2 - x_2 = 0.1 - 0 = 0.1$$

Training sample 2 (∈ N⁻) $y = 0$

$$h(TS2) = g(w_0 x_0 + w_1 x_1 + w_2 x_2) < 0$$

$$= g(-0.9 \times 1 + .1 \times 0 + .1 \times 1)$$

$$= g(-0.9 + .1 + .1) = g(-0.7)$$

No updation

Training sample 3

$h(TS3)$ .....

Training sample 4

...

Epoch 2 :

So on

Continue the process epoch by epoch till convergence (no misclassification)

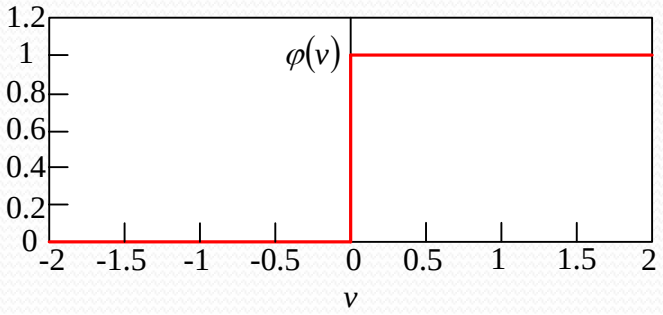
Important Aspects

- Architecture (Feed-forward/signal can move in forward direction)
- Activation function (Threshold)
- Learning process (Perceptron learning algorithm)

Types of Activation function

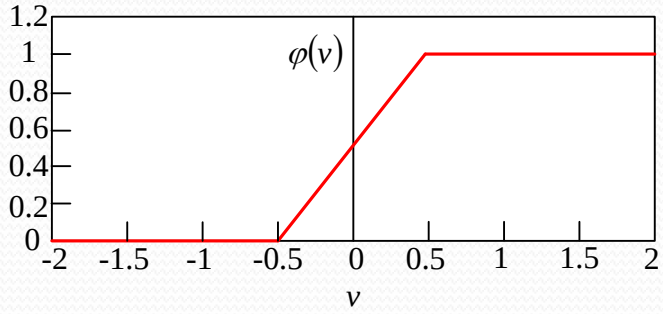
- Threshold function

$$\varphi(v) = \begin{cases} 1 & if\ v \geq 0 \\ 0 & if\ v < 0 \end{cases}$$



- Piecewise-Linear Function

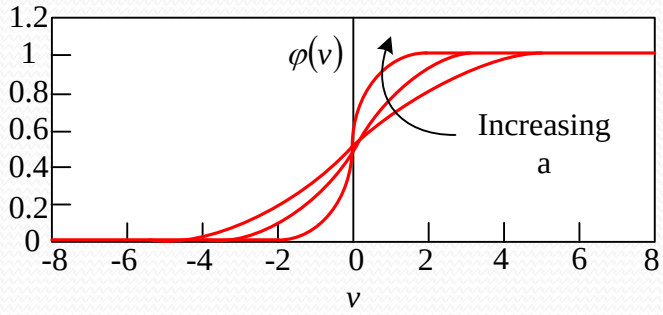
$$\varphi(v) = \begin{cases} 1 & v \geq 1/2 \\ v & -1/2 < v < 1/2 \\ 0 & v \leq -1/2 \end{cases}$$



- Sigmoid Function/logistic function

$$\varphi(v) = \frac{1}{1+\exp(-av)}$$

a is the slope parameter



Sigmoid (logistic) neuron (binary classification/linearly separable)

- A perceptron neuron will fire if the weighted sum of its input is greater than a threshold ($-w_0$)
- The threshold activation function is very harsh (behave like a step function)
- We want some smoother decision function (gradually moves from 0 to 1)
- Introduce sigmoid neuron using one form of the sigmoid function (i.e., logistic function) as activation function/ output function
- Now there is no sharp transition around the threshold.

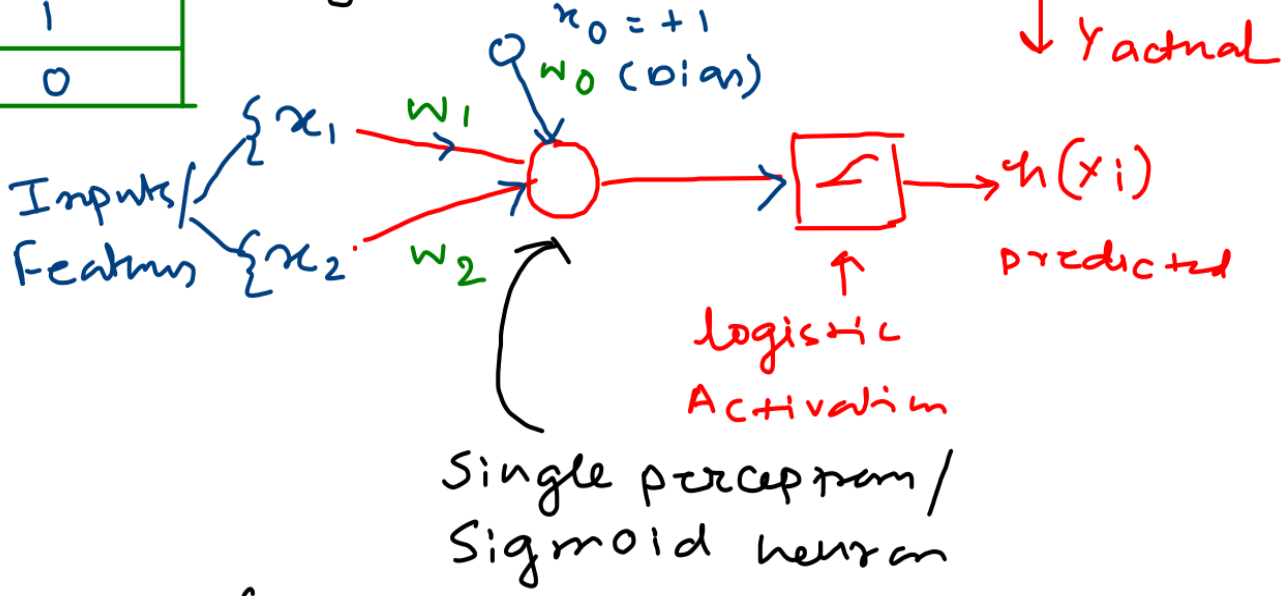
Dataset

x_0	x_1	x_2	Y
1	0.2	3	1
1	0.5	1	0
1	1.6	2	1
1	2.3	4	0

→ Training Samples

Step1: Normalization and split dataset (train, test, validation set)

Step2: Architecture (SLP/ Single layer perceptron) with Sigmoid neuron



Step3: Train using GD learning algorithm

Step4: After convergence, test for unknown patterns

Single perceptron to Singlelayer Perceptron (SLP)

- A singlelayer perceptron (**SLP**) is a feed-forward neural network
- Generally, it **(single perceptron)** has following properties:

  1. one perceptron neuron (threshold activation function)
  2. classifier model for binary classification of linearly separable patterns
- However, some modifications have been incorporated.

  1. Sigmoid neuron
  2. Can we use same architecture for multi-class classification problem ? (same one-vs.-all approach)
  3. Can we use similar architecture for regression problem?
  4. **SLP: Single layer of perceptrons/artificial neurons (to handle multi-class classification in one architecture)**

Single Perceptron for multi-class classification

x_0	x_1	x_2	y
1	.6	1	1
1	.2	3	2
1	.4	5	2
1	6	.2	3

Original dataset

class 1

x_0	x_1	x_2	y
1	.6	1	1
1	.2	3	0
1	.4	5	0
1	6	.2	0

Data set for SLP1

class 2

x_0	x_1	x_2	y
1	.6	1	0
1	.2	3	1
1	.4	5	1
1	6	.2	0

Dataset for SLP2

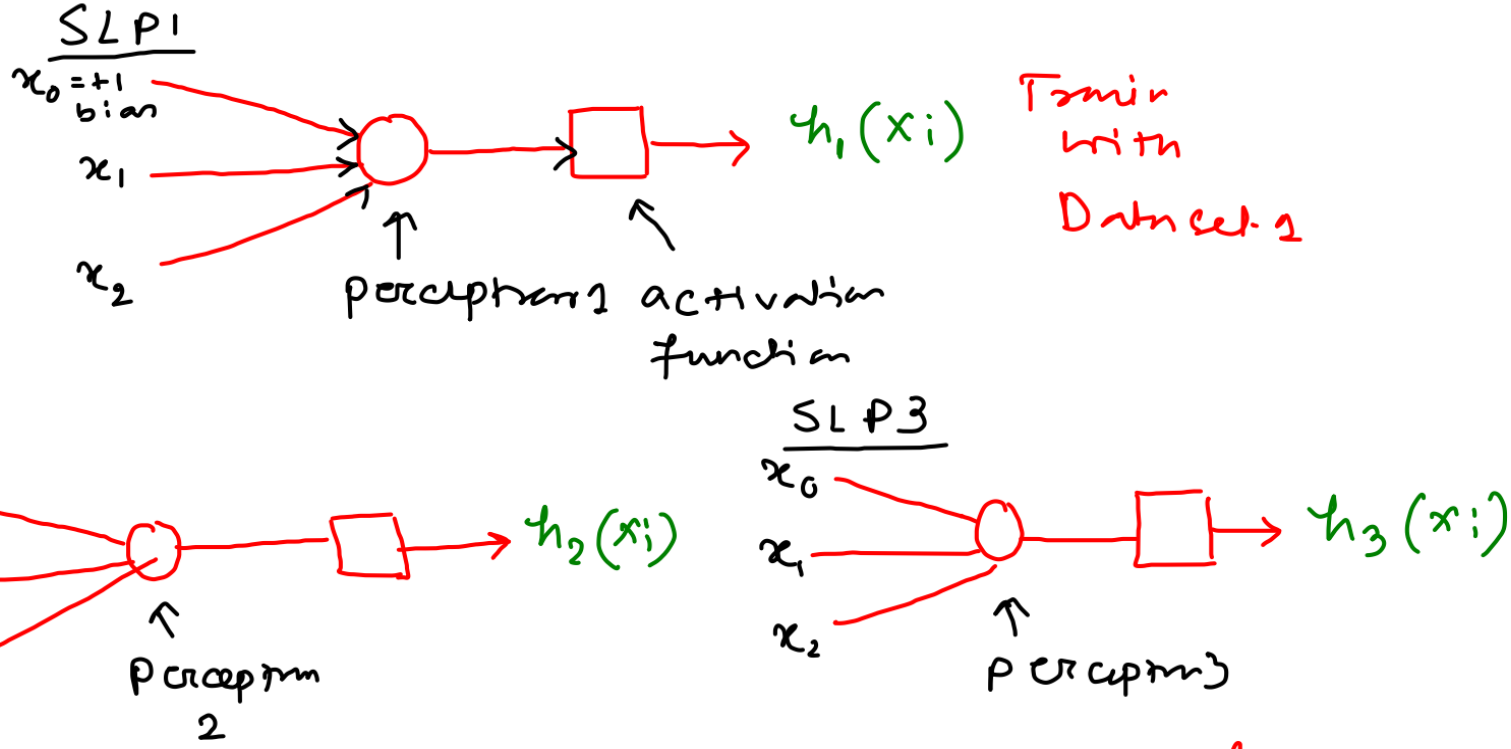
class 3

x_0	x_1	x_2	y
1	.6	1	0
1	.2	3	0
1	.4	5	0
1	6	.2	1

Data Set for SLP3

Step1 Training phase

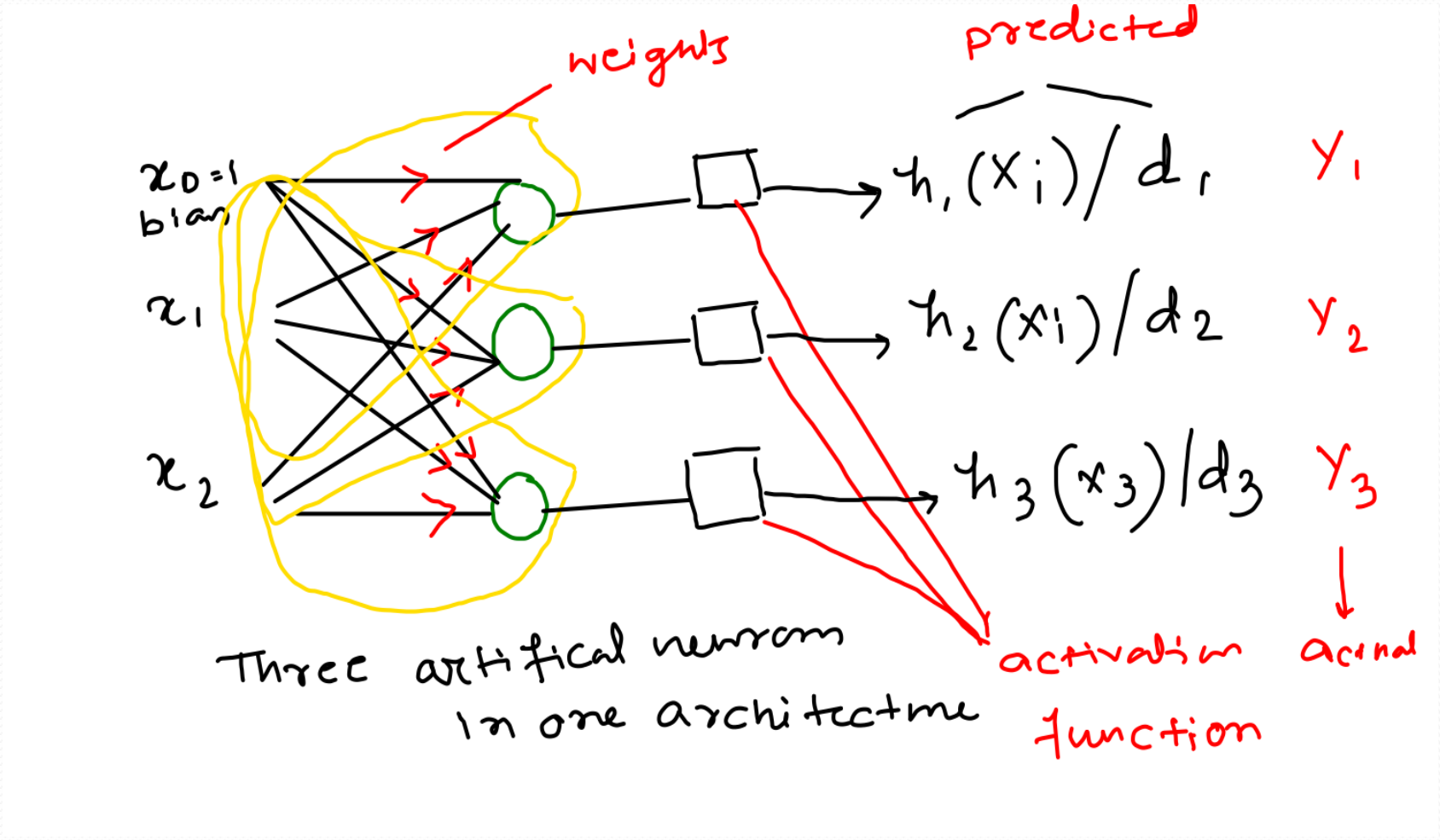
[train 3 SLPs for 3 class problem for seperate datasets]



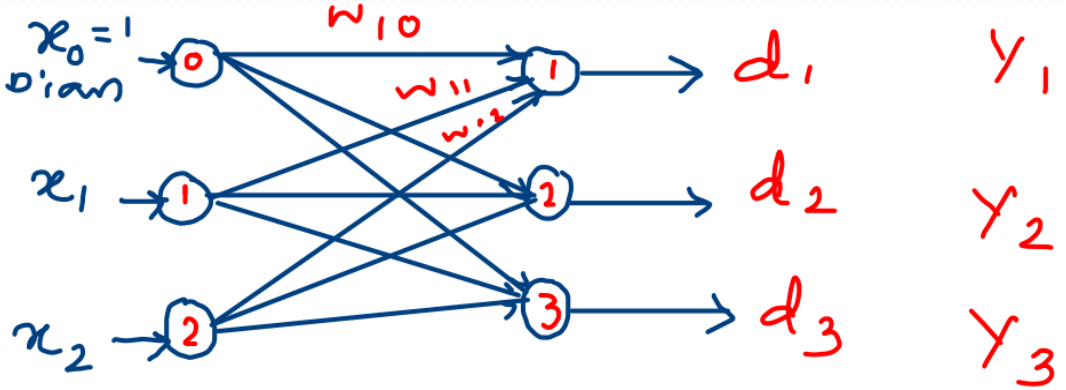
Step2 Choose k with $\max_k h_k(x_i)$

Singlelayer Perceptron (SLP)

Multi-class classification



Architecture for SLP



Inputs / Input neuron

Input layer

Output neurons

Output layer

$$W = \begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \\ w_{30} & w_{31} & w_{32} \end{bmatrix}$$

→ weights between output neuron i with input neurons / inputs

Singlelayer percepron (SLP)

- It is a feed-forward neural network.
- It has a single layer of perceptrons or artificial neurons.
- It can handle multiclass problem in one architecture.
- Generally, from architecture point of view, it has two layers:

1. Input layer: number of inputs = number of features+1
2. Output layer: number of output neurons/perceptron/sigmoid neuron=number of classes

- Each input is connected to all the output neurons.

One hot encoding

First step, pre-process the dataset (label) using one hot encoding to prepare it for the multi-class classification

$Y = 0 / 1$ — binary classification

$Y \longrightarrow Y_1 \quad Y_2 \quad Y_3$

One hot encoding

x_0	x_1	x_2	class label	y_1	y_2	y_3
1	0.3	1	1	1	0	0
1	0.4	6	2	0	1	0
1	0.2	1.2	1	1	0	0
1	6.2	7	2	0	1	0
1	6.5	6	3	?	?	?

↓

Each output neuron represents one class

Learning process of SLP

Target:

- Each output neuron is trained to activate for exactly one class.
- The correct output for SLP : one output neuron is activated (what indicates the classification decision), others are not activated.
- Perceptron learning algorithm can be used for threshold activation function.
- Gradient descent algorithm can be used for sigmoid activation function.

Incremental GD for SLP (with Sigmoid neuron)

Epoch 1 error = $E_{itr} = 0$

Training Sample 1

$j \rightarrow$ output neuron / class

$i \rightarrow$ input neuron / feature

$g(\cdot) \rightarrow$ activation function

all weights are initialized to 0.1

$$d_j = g\left(\sum w_{ji}\, x_i\right)$$

$$d_1 = g(0.1 * 1 + 0.1 * 0.3 + 0.1 * 1) = .6$$ Suppose

$$d_2 = g(0.1 * 1 + 0.1 * 0.3 + 0.1 \times 1) = .6$$

$$d_3 = g(0.1 * 1 + 0.1 * 0.3 + 0.1 \times 1) = .6$$

$$E_{itr} = E_{itr} + \frac{1}{2}\left[(d_1 - y_1)^2 + (d_2 - y_2)^2 + (d_3 - y_3)^2\right]$$

$$E_{itr} = E_{itr} + \frac{1}{2}\sum_{j=1}^{3}(d_j - y_j)^2$$

$$= 0 + \frac{1}{2}\left[(.6 - 1)^2 + (.6 - 0)^2 + (.6 - 0)^2\right]$$

Weight update

$\alpha = 0.5$

$$w_{ji} := w_{ji} + \alpha (y_j - d_j) x_i \boxed{d_j (1 - d_j)}$$

↳ if sigmoid activation

$$w_{10} = 0.1 + 0.5 (1 - .6) * 1 * 0.6 (1 - 0.6)$$

$$w_{11} = 0.1 + 0.5 (1 - .6) * 0.3 * 0.6 (1 - 0.6)$$

$w_{12} = ?$ $w_{20}, w_{21}, w_{22}, w_{30}, w_{31}, w_{32} = ?$

Training sample2 : Same procedure.

$E_{tr} = ?$

$w_{10} = ?$ for all weights ??

$w_{11} = ?$

$w_{12} = ?$

Repeat same process for all training samples

Repeat Epoch 1, Epoch 2, ..., Epoch 3

itr itr+1 itr+2 . . .

Epoch 1 Complete

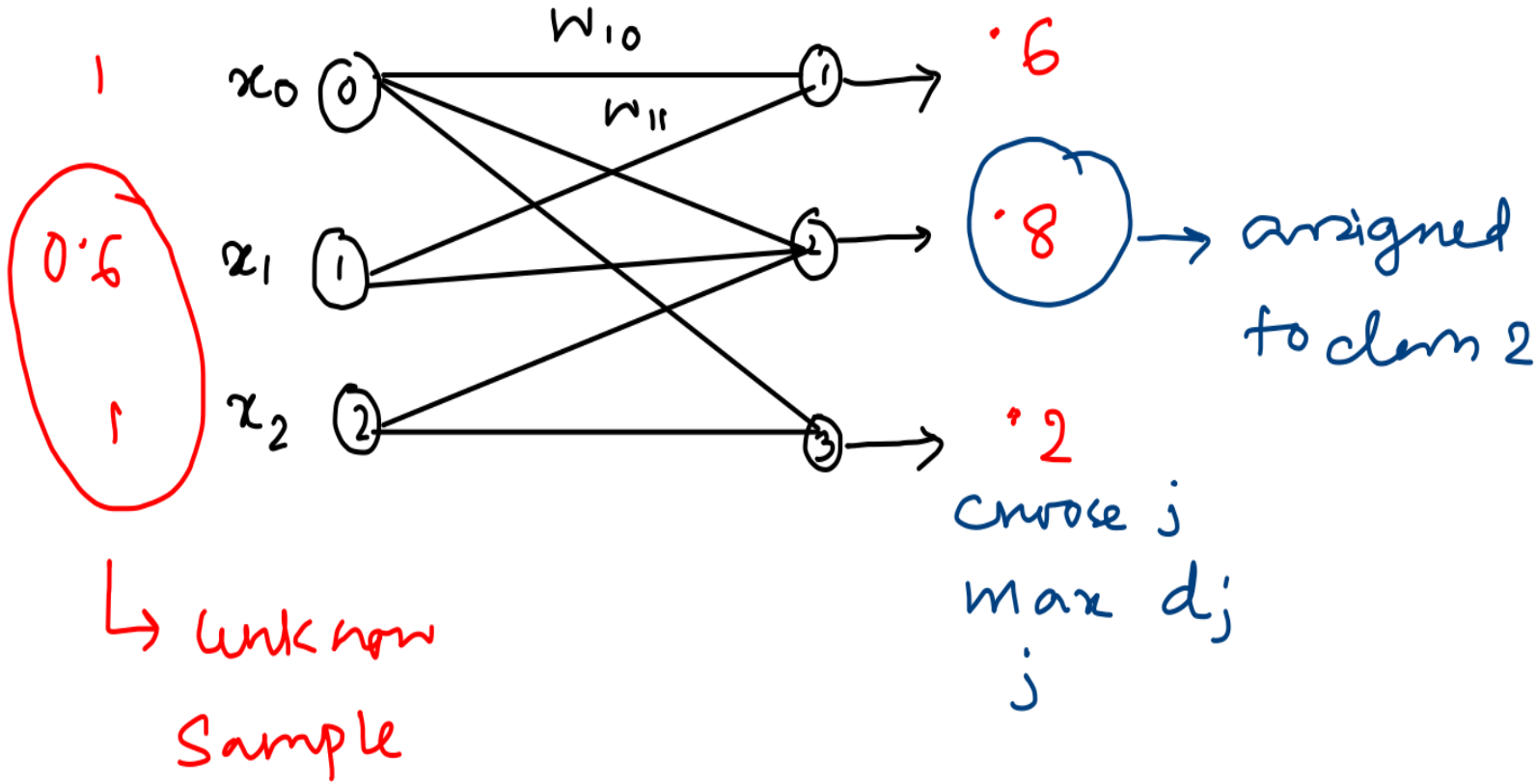
$E_{itr} = \frac{1}{m} E_{itr}$

Convergence:

$|E_{itr} - E_{itr-1}| <$ small value or

number of epochs > large value

Testing phase



Question

x_1	x_2	Class Label
1	2	1
0.2	3	2
4	1	2
6	1	3

Design SLP using the given training samples.

Note the following during demonstration:

1. Threshold activation function
2. Number of epochs:2
3. All weights are initialized as 0.1.

Predict the class label for test sample [3 0.1].

Non-linear Hypothesis
Introduction of Multilayer Perceptron

Machine Learning (CS 306)

Instructor: Dr. Moumita Roy

Teaching Assistants: Indrajit Kalita, Veronica Naosekpam

Email-ids:

moumita@iiitg.ac.in

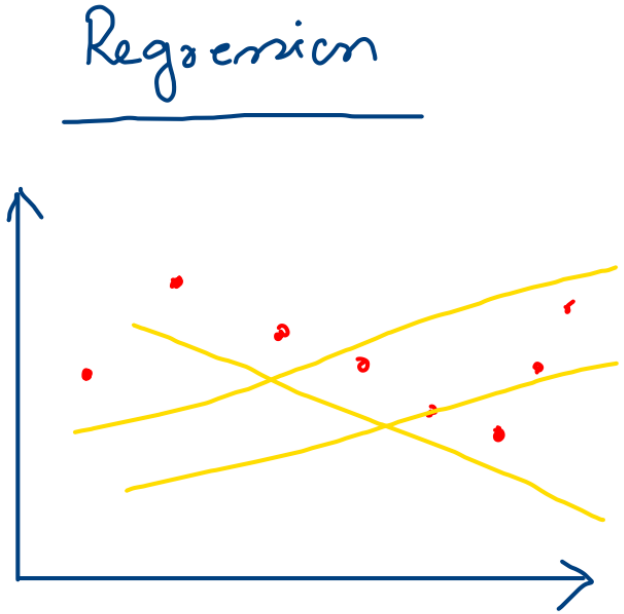
veronica.naosekpam@iiitg.ac.in

indrajit.kalita@iiitg.ac.in

Mobile No: +91-8420489325 (only for emergency quires)

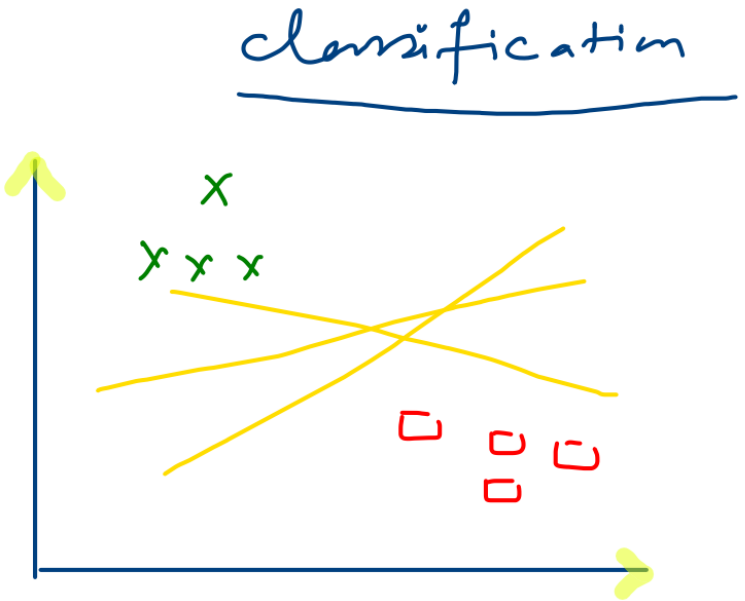
Reference: https://www.cse.iitm.ac.in/~miteshk/CS6910/Slides/Lecture2.pdf

Regression vs. Classification (considering linear hypothesis)



Linear Relation between independent and dependent variables

Try to find best fit one by minimizing cost function

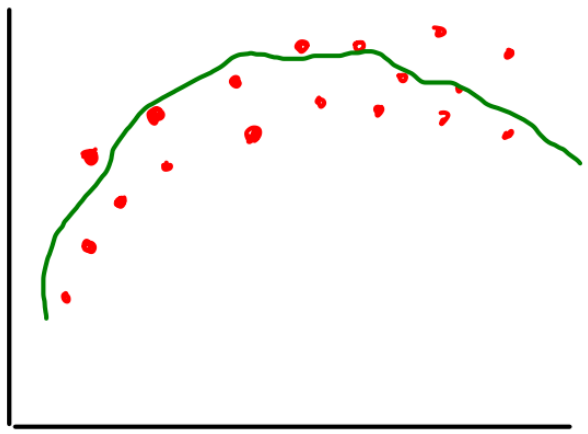


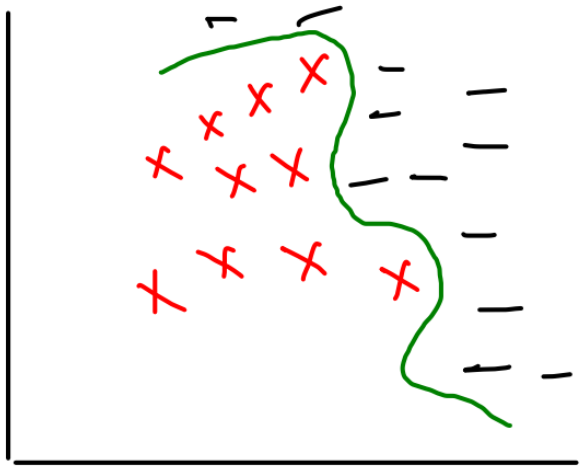
Data set is linearly separable

Try to find linear DB by minimizing cost function

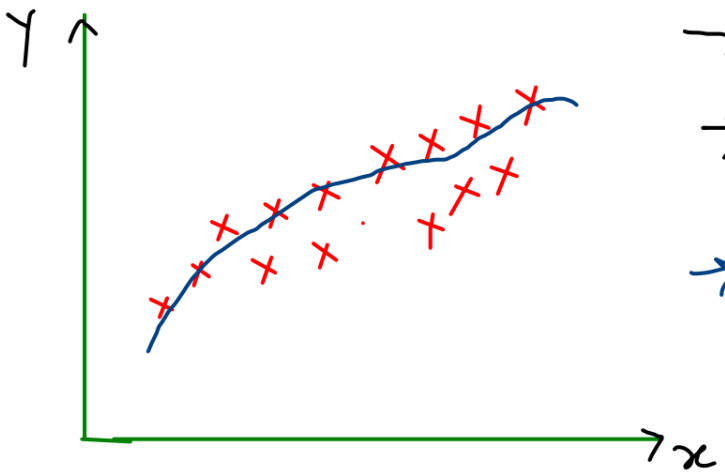
Consider binary classification

Scenarios (Non-linear hypothesis)





Polynomial Regression



$\rightarrow h(x) = w_0 + w_1 x$

$\rightarrow h(x) = w_0 + w_1 x + w_2 x^2$

$\rightarrow h(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

If, we consider, $h(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

Can we use Linear regression to Solve the problem?

Introduce more features

$x_1 = x$

$x_2 = x^2$

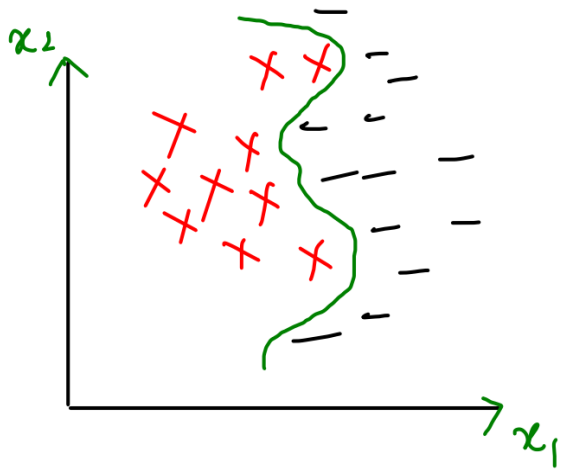
$x_3 = x^3$

Rewrite the hypothesis,

$$h(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$

Can we solve now?!

Non-linear classification



$$\rightarrow h(x) = g(w_0 + w_1 x_1 + w_2 x_2)$$

$$\rightarrow h(x) = g(w_0 + w_1 x_1^2 + w_2 x_2^2 + w_1 x_1 x_2 + w_2 x_1 x_2^2 + \cdots)$$

Introduce more features

$$x_1' = x_1^2$$
$$x_2' = x_2^2$$
$$x_3' = x_1 x_2$$
$$\vdots$$
So many ....

$$h(x) = g(w_0 + w_1 x_1' + w_2 x_2' + w_3 x_3' + \cdots)$$

Can we solve this problem??

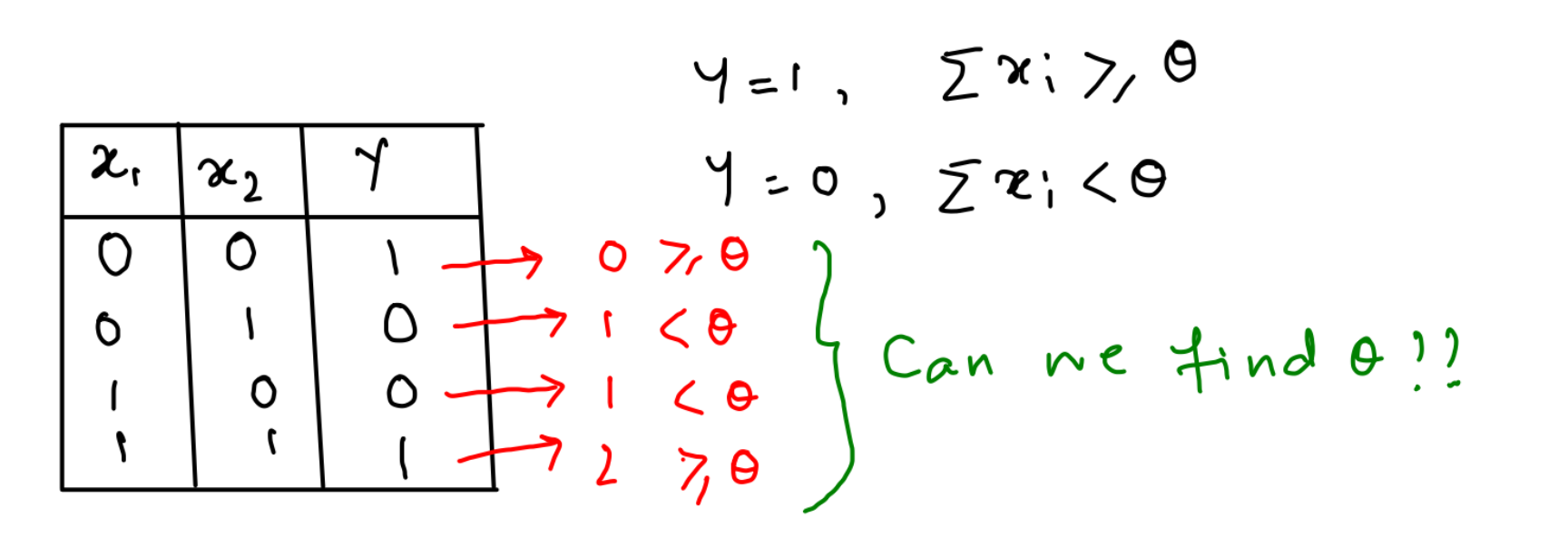
Observations

- Introducing more features (mainly polynomial terms; complex non-linear hypothesis)
- These features help the model to use existing learning algorithm to handle the problem (linear regression/logistic regression)
- Such hand-crafted feature engineering is not desirable option to solve non-linear problems (with more features)

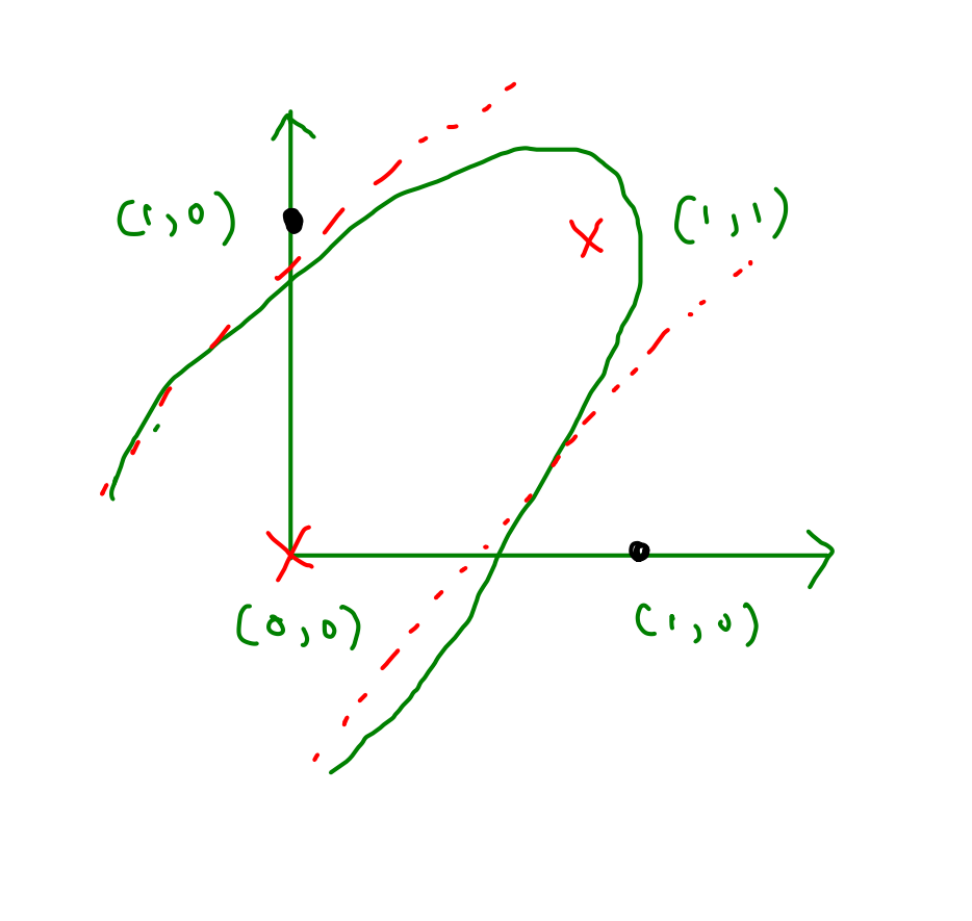
ANN

- ANN is a very powerful and widely used model to learn a complex non-linear hypothesis
- ANN Representation helps us to automatically extract such features
- Solve XOR problem with ANN topics discussed so far (MP neuron, Perceptron, Sigmoid neuron)

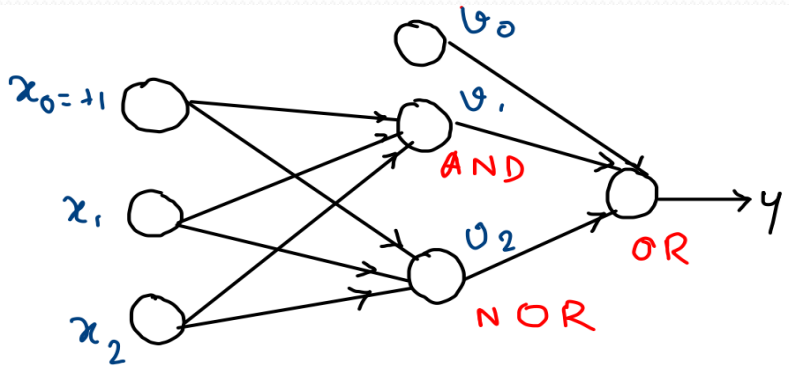
XOR Problem (using MP neuron)



Geometrical representation for XOP Problem



Introduce multiple layers in ANN



x_0	x_1	v_1	v_2	y
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Hidden features (not always easy to demonstrate)

Input layer — Hidden layer — Output layer

XOR → Simple problem (Solve by two lines)
we can solve it by dividing it into two linear problem (AND and NOR ; then OR)

Multilayer Perceptron (MLP)

- It has one input layer, one output layer, and multiple hidden layers (feed-forward architecture).
- Number of input neurons= number of features+1
- Number of output neurons=number of classes
- Number of hidden layers and hidden neurons in each layer are fixed experimentally (application depended)
- We consider, number of hidden layers=1
- Activation function has been considered in each hidden neuron and output neuron.

Neural representation for non-linear hypothesis

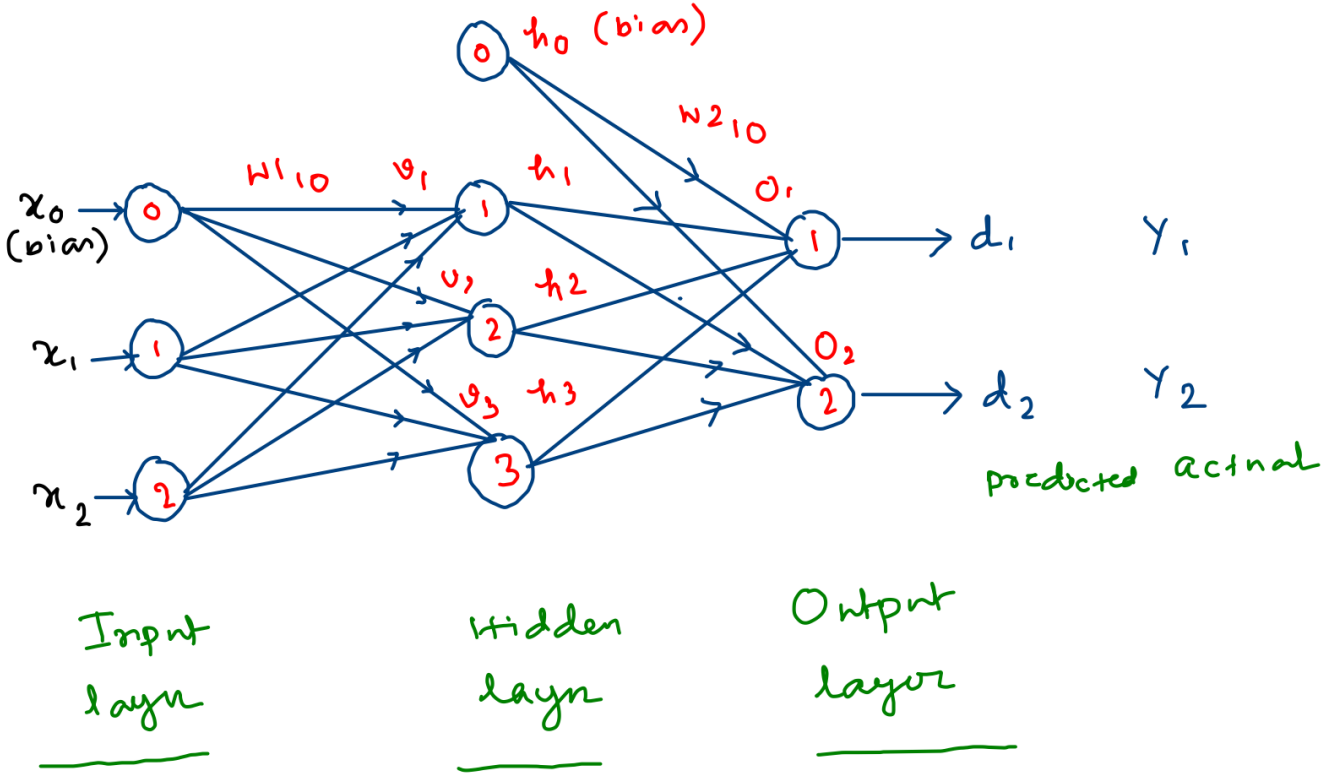
- MLPs can be trained to implement any given nonlinear input-output mapping (hypothesis).
- Here, multiple hidden neurons (with non-linear activation function) are helpful to extract hidden features automatically.
- MLP is basically transformation of the original input space into a new one, where the classification task becomes linearly separable.
- In this regard, Cover's theorem may be referred:

  A complex pattern-classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space.

Training Process of MLP

- Design a MLP to solve a non-linearly separable problem (2 features and two classes)
  - Data preprocessing (one hot encoding)
  - Architecture
  - Training algorithm (?)
  - Cost function: MSE
  - Activation function: Sigmoid/logistic function

Architecture



Start training (using incremental GD)

Random initialized the weights

Input to hidden layer

$$v_1 = W1_{10} x_0 + W1_{11} x_1 + W2_{12} x_2$$

$v_2 =$ same ...

$v_3 =$ same ...

$$h_1 = g(v_1) = \frac{1}{1 + \exp(-v_1)}$$

$h_2 =$ same ...

$h_3 =$ same ....

Hidden to output

$$O_1 = W2_{10} h_0 + W2_{11} h_1 + W2_{12} h_2 + W2_{13} h_3$$

O_2 = Same ....

$$d_1 = g(O_1)$$

$$d_2 = g(O_2)$$

Training GD

$$E_{itr} = \frac{1}{2}\sum_{j=1}^{2}(y_j - d_j)^2$$

error in each output neuron.

$$e_1 = y_1 - d_1$$

$$e_2 = y_2 - d_2$$

$$w_{ji} := w_{ji} + \alpha (y_j - d_j)\, d_j (1 - d_j)\, x_i$$

Update the weight between hidden to output:

$$w2_{10} = w2_{10} + \alpha (y_1 - d_1)\, d_1 (1 - d_1)\, h_0$$

⋮

Same for others

Update weight between input to hidden

$$w'_{10} = w_{10} + \alpha \underbrace{(? - h_1)}_{?} h_1(1-h_1) x_0$$

? error (not known)

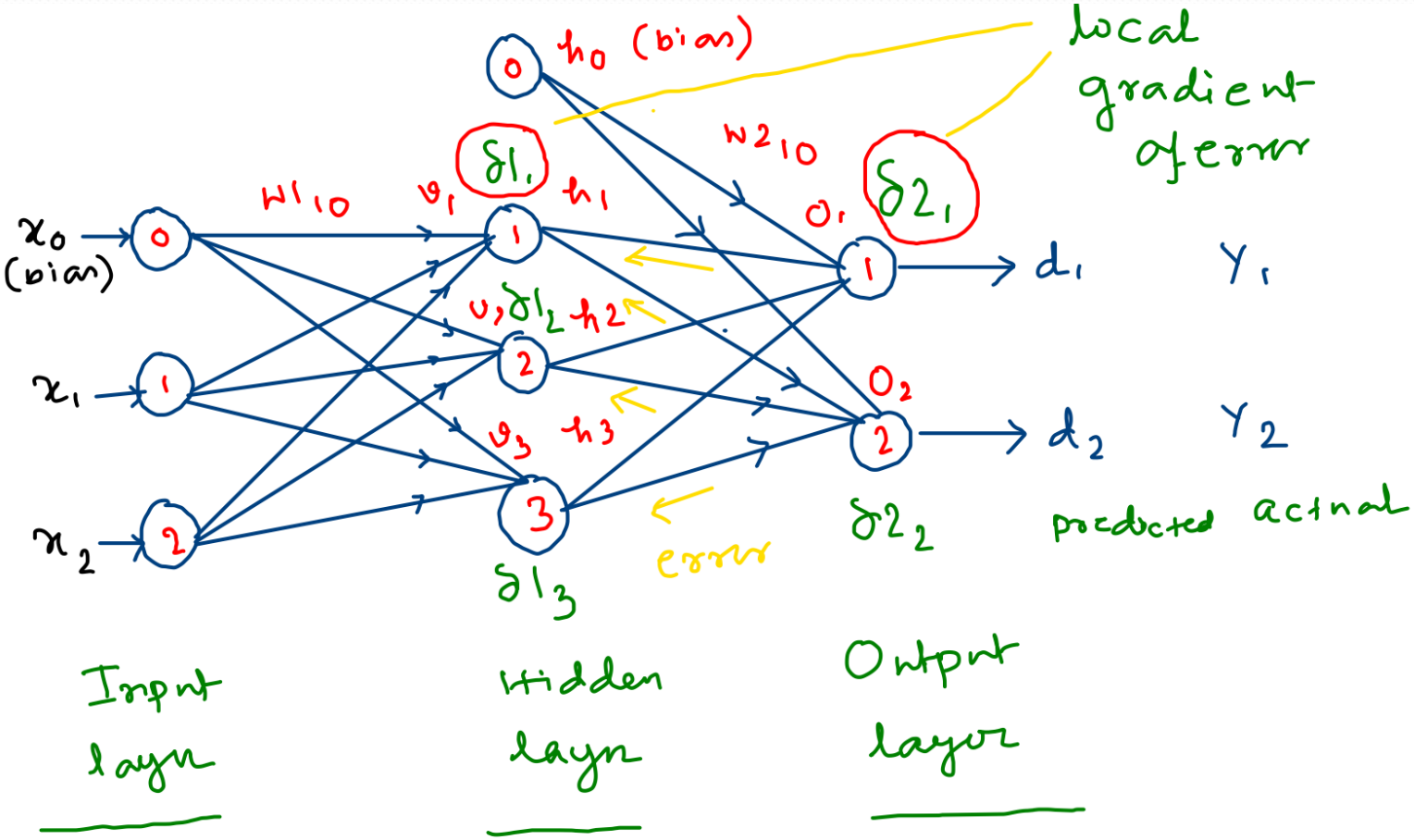
Solution: back-propagating errors for output neurons to hidden neurons

Back-propagation learning algorithm

Step1: Forward pass

Step2: Backward pass

Back-propagation algorithm



Back-propagation algorithm

Forward Pass (updation for one sample)

Input to hidden layer

$$v_1 = w1_{10} x_0 + w1_{11} x_1 + w2_{12} x_2$$

$v_2 =$ same ...

$v_3 =$ same ...

$$h_1 = g(v_1) = \frac{1}{1+\exp(-v_1)}$$

$h_2 =$ same ...

$h_3 =$ same ...

Hidden to output

$$O_1 = W2_{10} h_0 + W2_{11} h_1 + W2_{12} h_2 + W2_{13} h_3$$

O_2 = Same ....

$$d_1 = g(O_1)$$

$$d_2 = g(O_2)$$

Backward pass

Error calculation for the pattern

$$E_{ptn} = \frac{1}{2} \sum_{j=1}^{2} (d_j - y_j)^2$$

weight updation in GD

$$w_{ji} := w_{ji} + \alpha (y_j - d_j)(1 - d_j) d_j x_i$$

error in each output node

$$e_1 = (y_1 - d_1)$$

$$e_2 = (y_2 - d_2)$$

local gradient of error in each output node

$$\delta 2_1 = d_1(1-d_1)(Y_1-d_1)$$

$$\delta 2_2 = d_2(1-d_2)(Y_2-d_2)$$

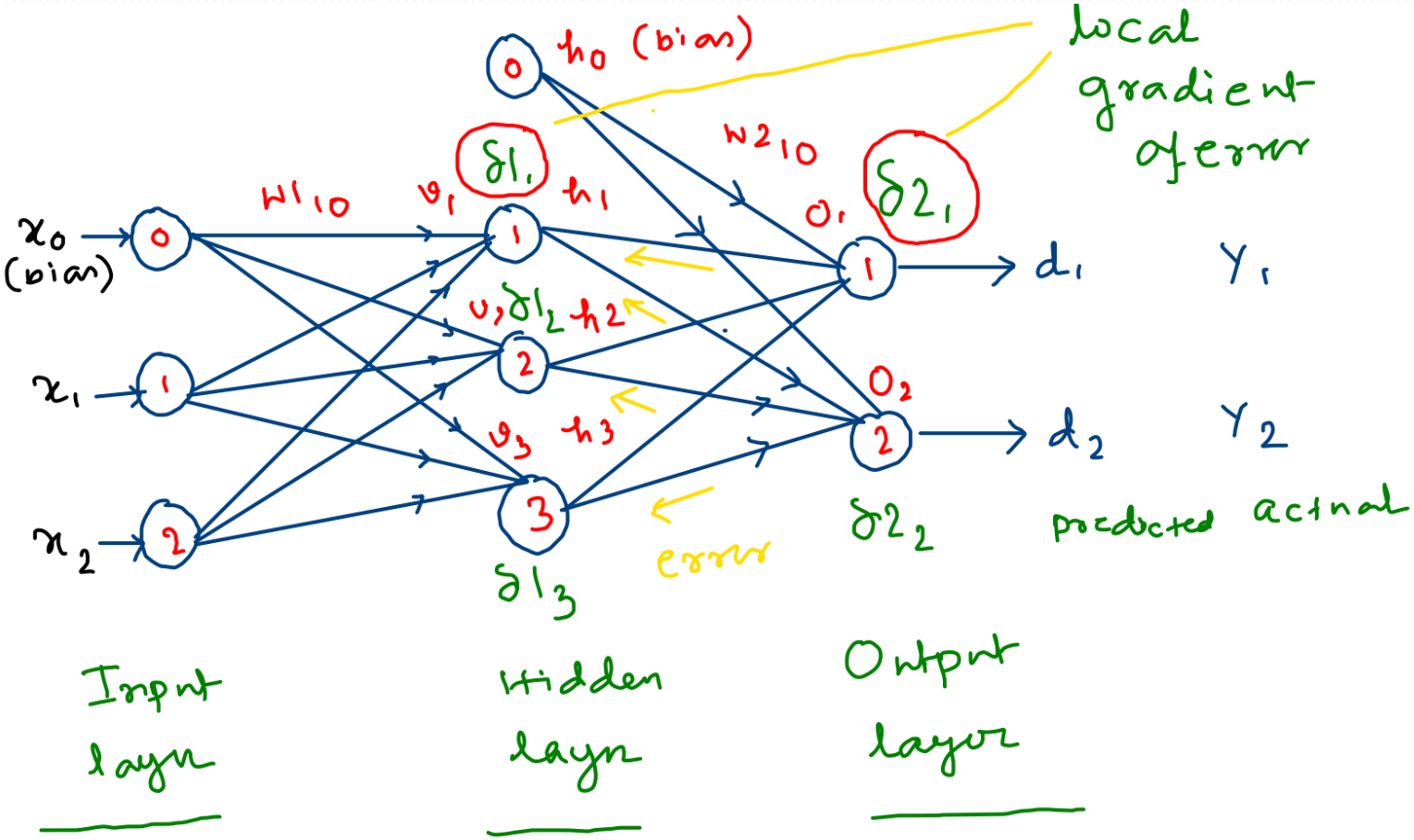
local gradient of error in each hidden node

$$\delta 1_1 = (\delta 2_1 * w2_{11} + \delta 2_2 * w2_{21})\, h_1(1-h_1)$$

$$\delta 1_2 = (\delta 2_1 * w2_{12} + \delta 2_2 * w2_{22})\, h_2(1-h_2)$$

$$\delta 1_3 = ?$$

Back-propagation algorithm



Updation of weights

$$w_{ji} := w_{ji} + \alpha (y_j - d_j) d_j (1 - d_j) x_i$$

hidden to output

$$w2_{10} := w2_{10} + \alpha * \delta 2_1 * h_0$$

$$w2_{11} := w2_{11} + \alpha * \delta 2_1 * h_1$$

$$\vdots$$

Input to hidden

$$W1_{10} := W1_{10} + \alpha * \delta 1_1 * x_0$$

$$W1_{11} := W1_{11} + \alpha * \delta 1_1 * x_1$$

$$\vdots$$

Stopping criteria

no. of samples → m, 2 ← no. of classes

Epoch 1: $E_{itr} = \sum_{i=1}^{m} \sum_{j=1}^{2} (y_j - d_j)^2$

Update weights for each sample

Epoch 2: $E_{itr+1} = \sum_{i=1}^{m} \sum_{j=1}^{2} (y_j - d_j)^2$

Until

$|E_{itr-1} - E_{itr}| >$ small value or

no. of epochs < large value

Clustering Techniques

Machine Learning (CS 306)

Instructor: Dr. Moumita Roy

Teaching Assistants: Indrajit Kalita, Veronica Naosekpam

Email-ids:

moumita@iiitg.ac.in

veronica.naosekpam@iiitg.ac.in

indrajit.kalita@iiitg.ac.in

Mobile No: +91-8420489325 (only for emergency quires)

Reference: slides from Dr. Debasis Samanta (IIT Kharagpur)

- Such problems are already introduced in Artificial Intelligence course (under the topics of local search)

Complete state-formulation (Clustering)

Formulate the following problem for intelligent agent:

Divide the N students in k groups based on their marks in AI such as diversity in each group is minimized.

For demonstration, you may use:

N=5, k=2, set of marks={50, 20,10, 5, 15, 45} (out of 60)

- State representation:
- Initial state:
- Goal state:
- Successor function:
- Objective function:

Complete state-formulation (Version 1) (Clustering)

Set of students/marks $A=\{a_1, a_2, a_3, \ldots\ldots, a_N\}= \{50, 20, 10, 5, 15, 45\}$

- State representation:

  $S=[C_1\ C_2\ C_3\ \ldots\ldots\ C_N]$

  if a_j belongs to i^{th} group, then $C_j=i$, where $i \in \{1, 2, 3, \ldots., k\}$

- Initial state: (example for demonstration)

  $S^0=[1\ 2\ 2\ 2\ 2\ 1]$ → better representation is needed

  $\mathbf{F(S^0)}=(50-45)^2 + [(20-10)^2+(20-5)^2+(20-15)^2+(10-5)^2+(10-15)^2+(5-15)^2]$

- Goal state/test:

  No binary goal state

  report the state when we have the minimum value of objective function

- Successor function: Successor is generated by changing the group of one student at a time
- Objective function: summation of diversity in each cluster

Complete state-formulation (Version 2) (Clustering)

Set of students/marks $A=\{a_1, a_2, a_3,\ldots\ldots,a_N\}= =\{50, 20,10, 5, 15, 45\}$

- State representation:

  $S=[C_1\ C_2\ C_3\ \ldots\ldots\ C_k]$

  if C_j is representative of the j^{th} group, then $C_j \in \{a_1, a_2, a_3,\ldots,a_N\}$

- Initial state: (example for demonstration)

  S^0=[50 20] (assigned in a group with nearest representative; then same as [1 2 2 2 2 1])

  $F(S^0)=[(50-50)^2+(50-45)^2]+[(20-20)^2+(20-10)^2+(20-5)^2+(20-15)^2]$

  $\mathbf{F(S^0)=(50-45)^2 + [(20-10)^2+(20-5)^2+(20-15)^2+(10-5)^2+(10-15)^2+(5-15)^2]}$

- Goal state/test:

  No binary goal state

  report the state when we have the minimum value of objective function
- Successor function: Successor is generated by changing a group representative at a time
- Objective function: summation of diversity in each cluster

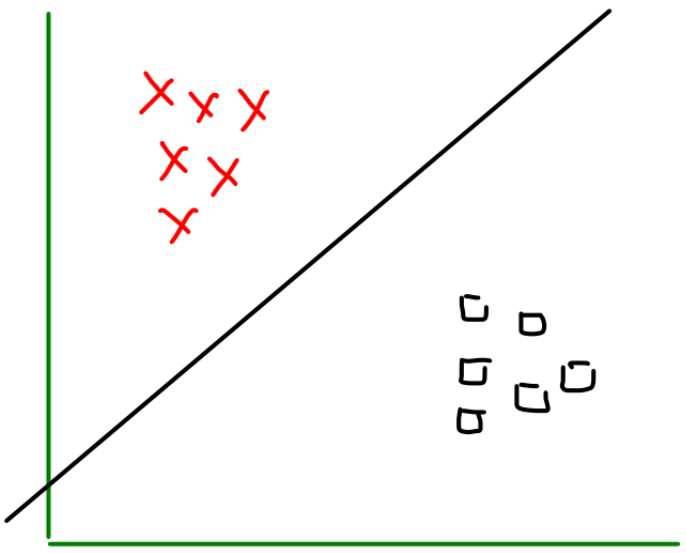
Observation

- We need some strategy to generate successors
- State space may be continuous or discrete
- Solution for such problems are addressed using the following concepts:
  - Genetic algorithms
  - Particle swarm optimization

Clustering techniques

- Clustering is a process of grouping a set of objects (into the groups of similar objects).
- It is most common form of unsupervised learning (learning from raw data without label).
- Groups are called clusters.
- Objects in the same cluster are somehow more similar to each other than the ones in the different clusters.

Classification

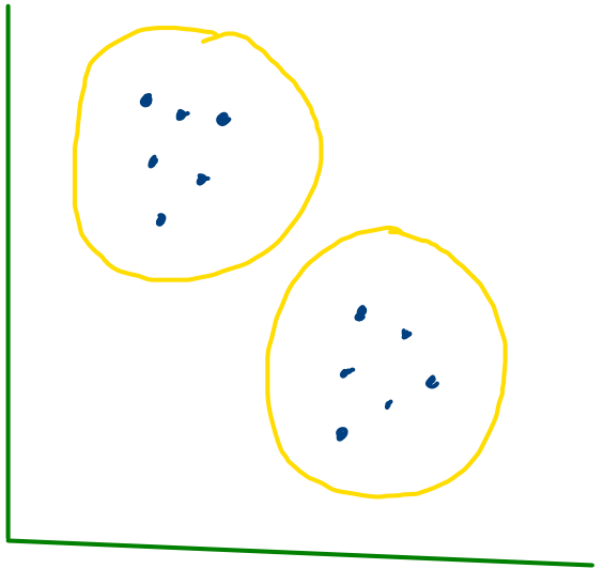


Supervised learning

Training Samples:

$\{(x_1, y_1), (x_2, y_2) \cdots (x_i, y_i)\}$

Clustering



Unsupervised learning

Samples:

$\{x_1, x_2, \cdots x_i\}$

Goal of Clustering

- Organizing samples into clusters such that there is
  - **high intra-cluster similarity**
  - **low inter-cluster similarity**

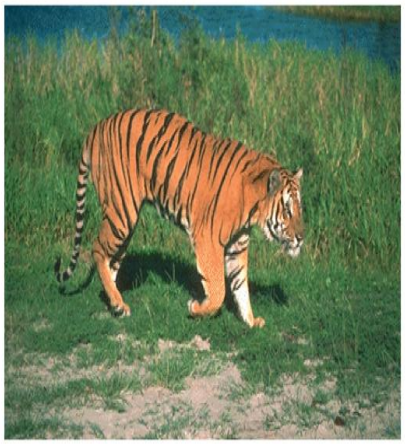
- Requires the definition of a similarity measure
- Try to gain some insight into the structure of the data/samples

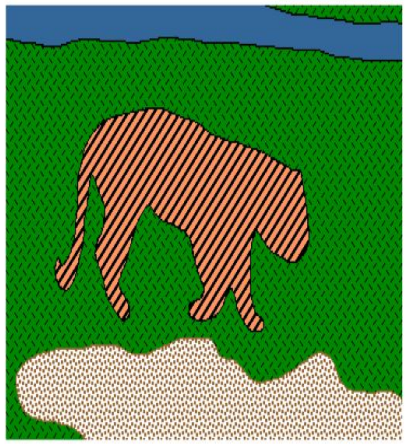
Application of clustering techniques

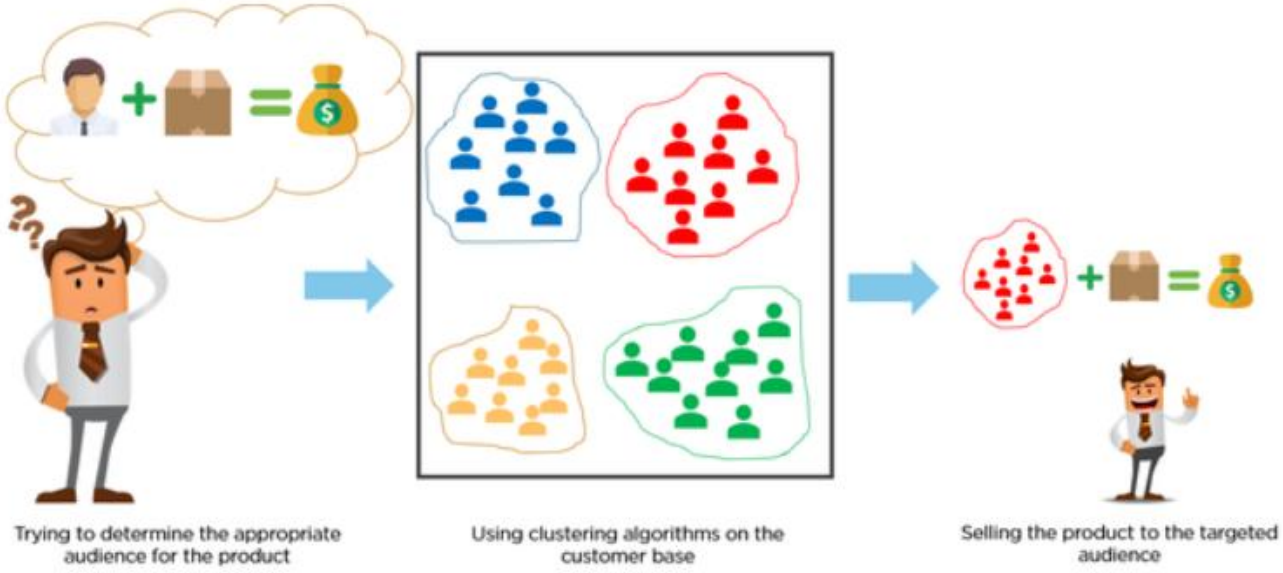
- Marketing policy (customer segmentation): Discover distinct groups of customers by analyzing the customer behavior and develop different marketing strategy for different groups

- Clustering in social network analysis: Discovery of clusters or communities (i.e. a collection of individuals with dense/sparse friendship patterns).

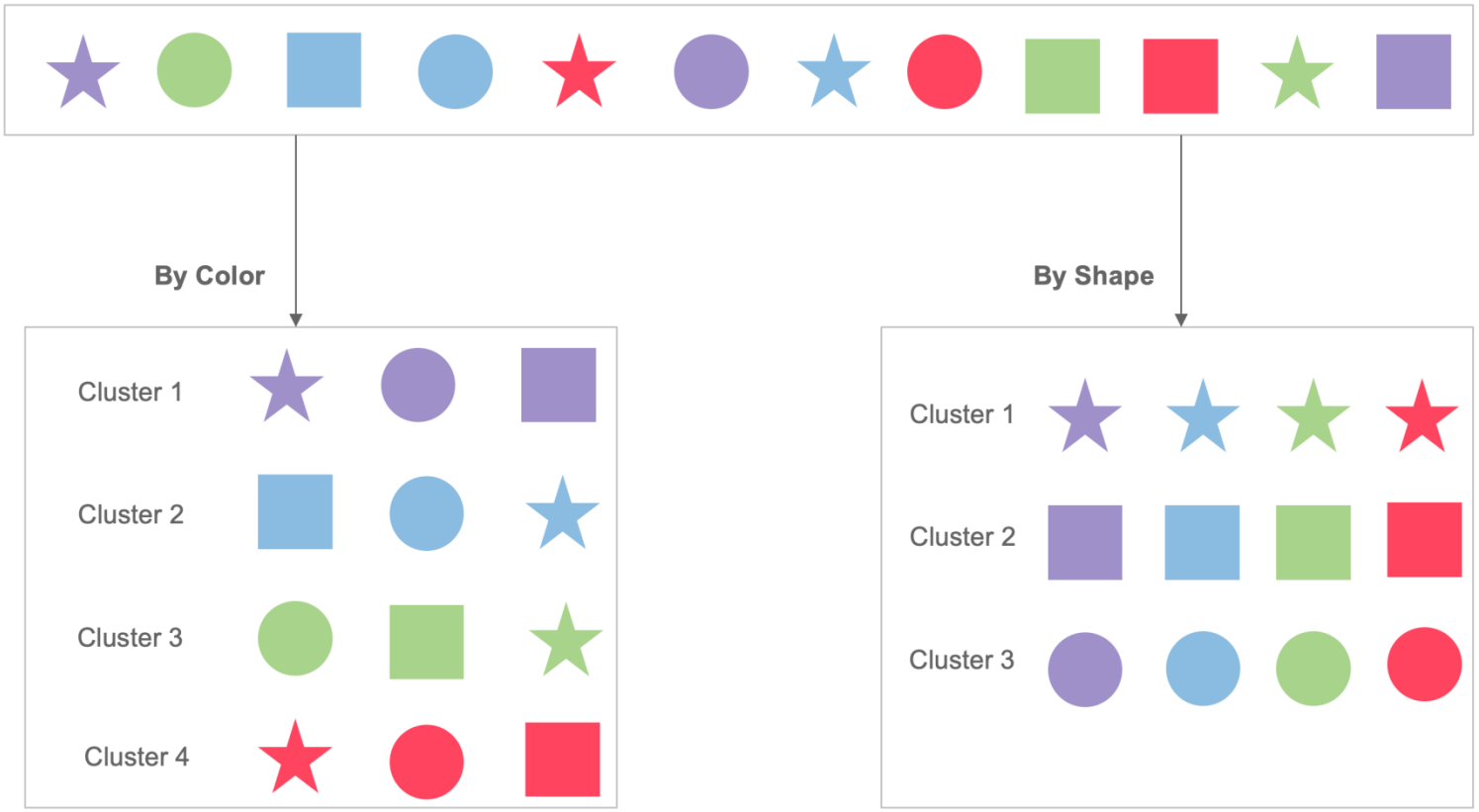
- Land-use monitoring (image segmentation): Identify areas of similar land-use






Trying to determine the appropriate audience for the product
Using clustering algorithms on the customer base
Selling the product to the targeted audience

Clustering (subjective)



Clustering algorithms

- Hard clustering: Each pattern exclusively belongs to a single cluster.
  - Partitioning algorithm (K-means, K-medoids)
  - Density based methods (DBSCAN, CLIQUE)
  - Graph theoretical based methods (MST, OPOSSUM)
  - Model-based methods (SOFM, EM algorithm)
  - Genetic clustering algorithm
  - PSO based clustering
  - Hierarchical clustering (Agglomerative and divisive)
- Soft clustering: Each pattern may belongs to more than one clusters
  - Fuzzy clustering
  - Probabilistic clustering

What is Similarity?

The quality or state of being similar; likeness; resemblance; as, a similarity of features.
Webster's Dictionary

Similarity is hard to define, but…
"*We know it when we see it*"

The real meaning of similarity is a philosophical question. We will take a more pragmatic approach.

Similarity and Dissimilarity Measures

- In clustering techniques, similarity (or dissimilarity) is an important measurement.
- Informally, similarity between two objects (e.g., two images, two documents, two records, etc.) is a numerical measure of the degree to which two objects are alike.
- The dissimilarity on the other hand, is another alternative (or opposite) measure of the degree to which two objects are different.
- Both similarity and dissimilarity also termed as proximity.

Note:

- Frequently, the term distance (for example, Euclidian distance) is used as a synonym for dissimilarity
- In fact, it is used to refer as a special case of dissimilarity.

Partitioning algorithm: Basic concepts

- Partition of n objects into k-clusters by optimizing some partitioning criteria.

- One approach: examine all possible partition (too expensive)

- Heuristic based: K-means algorithm and K-medoids algorithm

k-Means Algorithm

- k-Means clustering algorithm proposed by J. Hartigan and M. A. Wong [1979].
- Given a set of *n* distinct objects/patterns, the k-Means clustering algorithm partitions the objects into *k* number of clusters such that intracluster similarity is high but the intercluster similarity is low.
- In this algorithm, we need to specify *k*, the number of clusters and consider the objects are defined with numeric attributes and thus using any one of the distance metric to demarcate the clusters.

k-Means Algorithm

The algorithm can be stated as follows.

- First it selects *k* number of objects at random from the set of n objects. These *k* objects are treated as the centroids of *k* clusters.
- For each of the remaining objects, it is assigned to one of the closest centroid. Thus, it forms a collection of objects assigned to each centroid and is called a cluster.
- Next, the centroid of each cluster is then updated (by calculating the mean values of attributes of each object).
- The assignment and update procedure is until it reaches some stopping criteria (such as, number of iteration, centroids remain unchanged or no assignment, etc.)

k-Means Algorithm

Algorithm: k-Means clustering

Input: D is a dataset containing *n* objects/patterns, *k* is the number of cluster

Output: A set of *k* clusters

Steps:

1. Randomly choose *k* objects from D as the initial cluster centroids.
2. **For** each of the objects in D **do**
   - Compute distance between the current objects and *k* cluster centroids
   - Assign the current object to that cluster to which it is closest.
3. Compute the "cluster centers" of each cluster. These become the new cluster centroids.
4. Repeat step 2-3 until the convergence criterion is satisfied
5. Stop

Important Points

- Distance measure: Usually Euclidian distance is used

- Convergence criteria:

  - maximum number of iteration
  - no change of centroid values in any cluster
  - no change in cluster assignment of the patterns
  - cluster quality reaches a certain level of aceptance

Cluster quality

- There is some objective function to be met as a goal of clustering. One of such objective function is sum-square-error (denoted by SSE).

$$SSE = \sum_{i=1}^{K} \sum_{P \in C_i} |d(P, m_i)|^2$$

$C_i \rightarrow$ denotes i^{th} cluster

$P \rightarrow$ denotes the pattern belongs to any cluster

$m_i \rightarrow$ denotes the centroid of i^{th} cluster

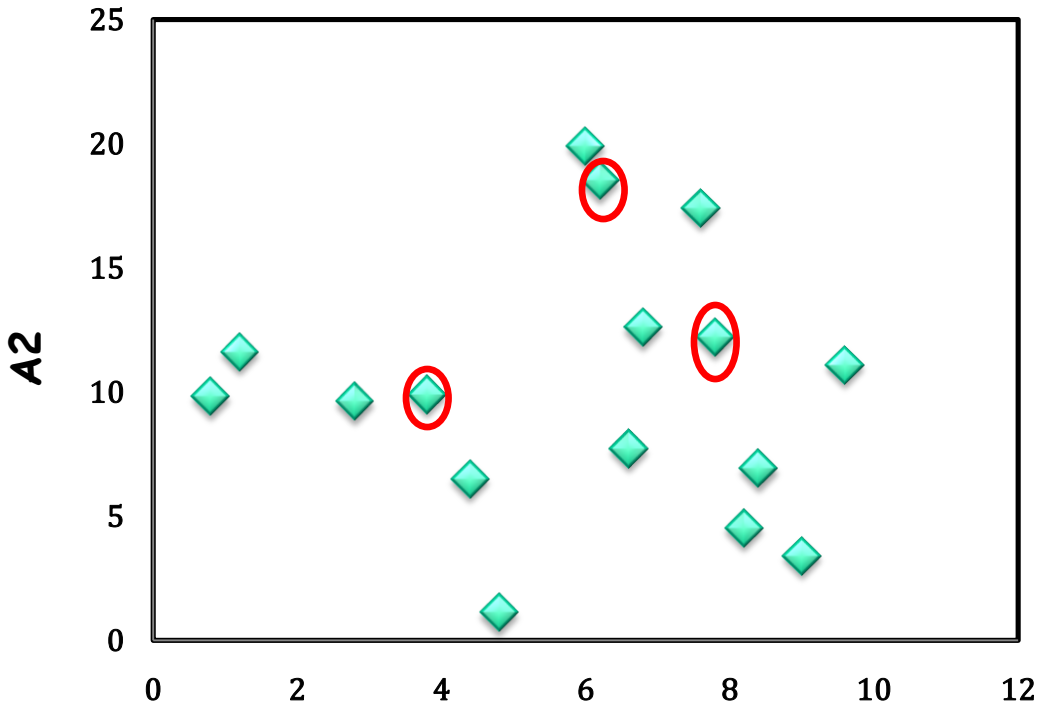
$d(P, m_i) \rightarrow$ denotes distance measure between any pattern and the corresponding cluster

Demonstration of k-Means clustering algorithms

Dataset

A_1	A_2
6.8	12.6
0.8	9.8
1.2	11.6
2.8	9.6
3.8	9.9
4.4	6.5
4.8	1.1
6.0	19.9
6.2	18.5
7.6	17.4
7.8	12.2
6.6	7.7
8.2	4.5
8.4	6.9
9.0	3.4
9.6	11.1

Plotting data (n=?,k=?, Feature=?)



Centroid	Objects	
	A1	A2
c_1	3.8	9.9
c_2	7.8	12.2
c_3	6.2	18.5

Initial Centroids chosen randomly

Demonstration of k-means clustering algorithms

- Suppose, k=3. Three objects are chosen at random shown as red circled. These three centroids are shown below.

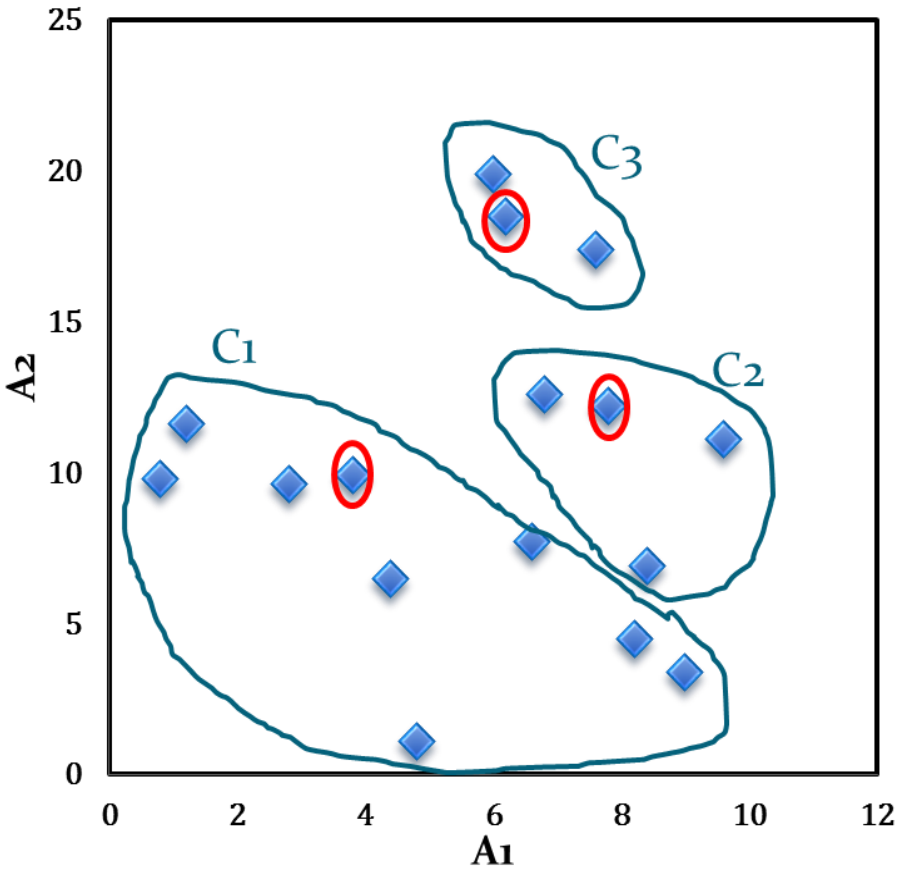
- Let us consider the Euclidean distance measure (L_2 Norm) as the distance measurement in our illustration.
- Let d_1, d_2 and d_3 denote the distance from an object to c_1, c_2 and c_3 respectively.
- Assignment of each object to the respective centroid is shown in the right-most column (in the next table).

Cluster Assignment (after Iteration 1)

Distance calculation

A_1	A_2	d_1	d_2	d_3	cluster
6.8	12.6	4.0	1.1	5.9	2
0.8	9.8	3.0	7.4	10.2	1
1.2	11.6	3.1	6.6	8.5	1
2.8	9.6	1.0	5.6	9.5	1
3.8	9.9	0.0	4.6	8.9	1
4.4	6.5	3.5	6.6	12.1	1
4.8	1.1	8.9	11.5	17.5	1
6.0	19.9	10.2	7.9	1.4	3
6.2	18.5	8.9	6.5	0.0	3
7.6	17.4	8.4	5.2	1.8	3
7.8	12.2	4.6	0.0	6.5	2
6.6	7.7	3.6	4.7	10.8	1
8.2	4.5	7.0	7.7	14.1	1
8.4	6.9	5.5	5.3	11.8	2
9.0	3.4	8.3	8.9	15.4	1
9.6	11.1	5.9	2.1	8.1	2

Cluster formation (iteration 1)



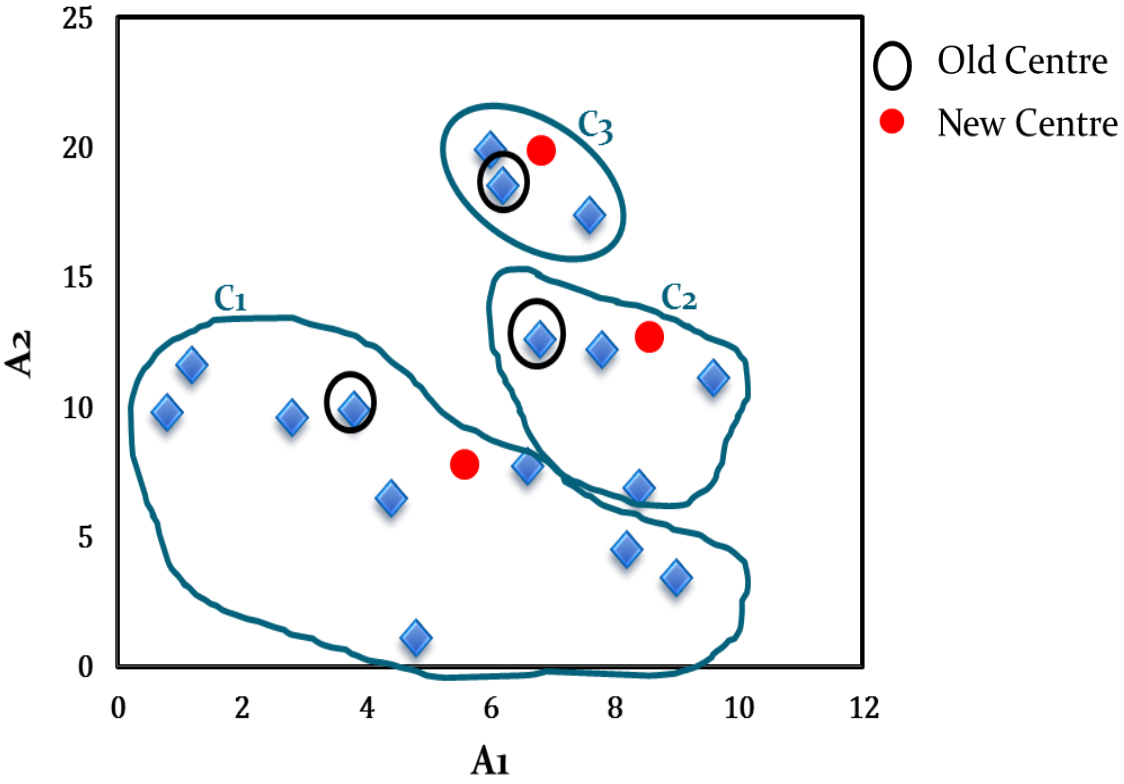
Updating centroids (iteration 1)

The calculation new centroids of the three cluster using the mean of attribute values of A_1 and A_2 is shown in the Table below. The cluster with new centroids are shown in Fig 16.3.

Calculation of new centroids

New Centroid	Objects	
	A1	A2
c_1	4.6	7.1
c_2	8.2	10.7
c_3	6.6	18.6

Repeat the steps until convergence



Initial cluster with new centroids

Comments on K-means clustering

- Simple clustering technique
- Specify the value of k: Elbow Method for optimal value of k
- Choose initial centroids: usually terminate with any initial choice of the cluster centroids; initial choice influences the ultimate cluster quality (trapped in local optimum); choose initial centroids in multiple runs
- Sensitive to outliers: When the SSE is used as objective function, outliers can unduly influence the cluster that are produced. More precisely, in the presence of outliers, the cluster centroids, in fact, not truly as representative as they would be otherwise. It also influence the SSE measure as well; (one solution may be choose centroids from samples)

K-medoids clustering algorithm

Motivation: The k-Medoids algorithm aims to diminish the effect of outliers. Medoids are similar in concept to means or centroids, but medoids are always restricted to be members of the data set.

Basic concepts:

- The basic concepts of this algorithm is to select an object as a cluster center (one representative object per cluster) instead of taking the mean value of the objects in a cluster (as in k-Means algorithm).
- We call this cluster representative as a cluster medoid or simply medoid.

1. Initially, it selects a random set of *k* objects as the set of medoids.
2. Then at each step, all objects from the set of objects, which are not currently medoids are examined one by one to see if they should be medoids.

This version of K-medoids clustering algorithm is also called partition around medoids (PAM)

Partition around medoids (PAM)