U-4 SML

12/11

(i) SVM

↳ support vector machines
↳ maximizes the boundaris b/w various points of data
↳ can be used at any type of data
~~Application~~: Text classification
Quality control of DNA

Classification: on their working principle.
(i) Maximum margin classifier
(ii) support vector classifer
(iii) support vector maching

(i) Maximum margin classifier
→ It is used to draw infinite hyper planes to classify the set of data
→ The hyperplane with maximum margin of separation will be an ideal hyperplane
→ Hyperplane is a straight line which seperates the 2-Dimensional space into two halves
→ Mathematical Rep:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

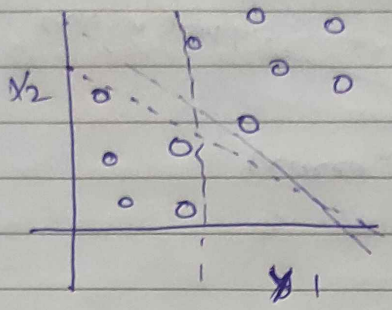
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 < 0$$

→ Constraint 1: $\sum_{j=1}^{n} \beta_j^2 = 1$

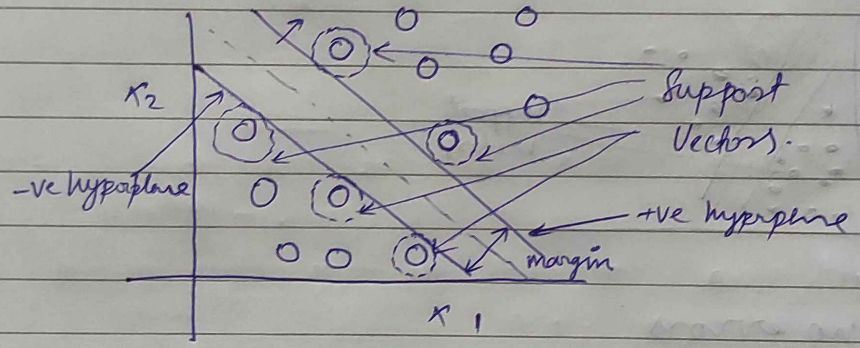
→ constraint 2: $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_n x_{in}) \geq M$

→ Infinite Hyperplanes



Used to separate two classes

→ Maximum Margin Classifier



used to fit the points in plane

(4) (ii) Support vector Classifier

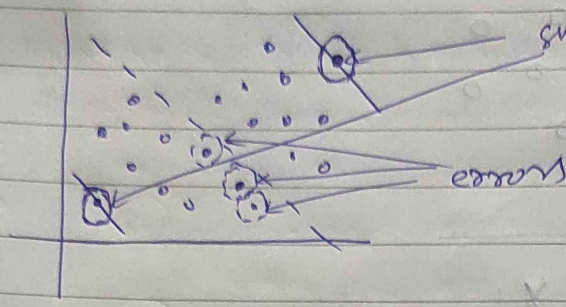
→ it is extended version of maximum margin classifier
→ used to create best fit and with less errors
→ Mathematical Rep:

constraint 1: $\sum_{j=1}^{n} \beta_j^2 = 1$

constraint 2: $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_n x_{in}) \geq M(1 - \varepsilon_i)$

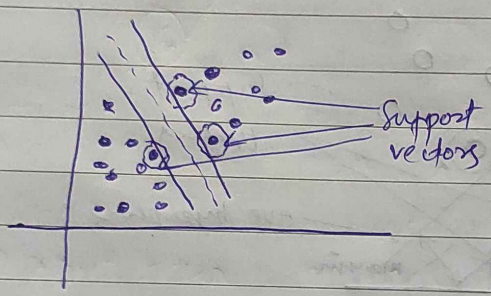
constraint 3: $\varepsilon_i \geq 0$

→ Support Vector Classifier with large value of C



more tolerating
has space for errors.

→ Support vector classifier with small value of C

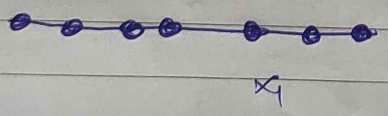


no errors

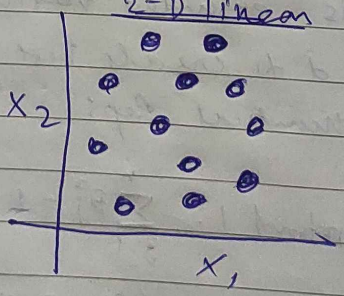
(iii) Support Vector Machines

→ used when decision boundary is non-linear
→ we cannot classify SVM using cost

1-D linear



2-D linear

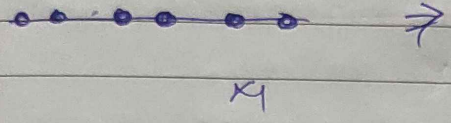


→ Kernel method is used to handle the data in 1-D and 2-D space

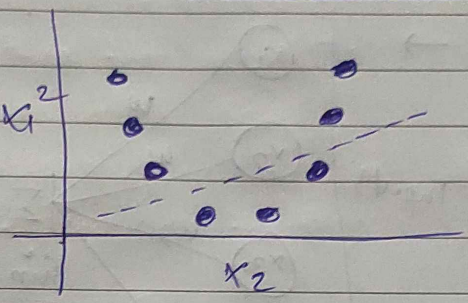
→ Apply a polynomial kernal with degree 2, transforms the data from 1D to 2D

→ helps to make data linear seperable.

1-D

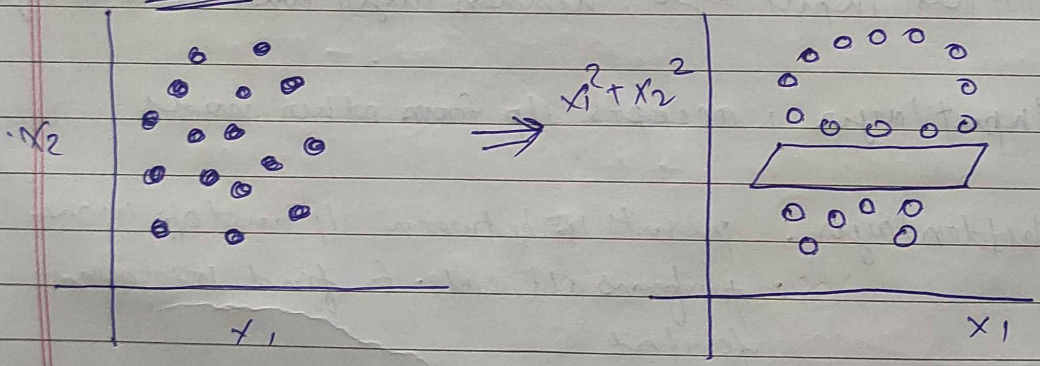


2D with kernel degree 2



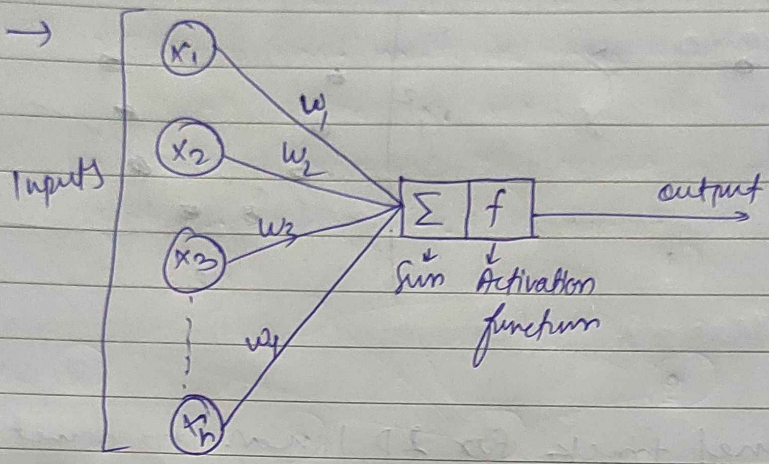
→ Applying kernel trick for 2D linear to convert it to Poly. kernel with Degree 2

2D



(2) Artificial Neural Network

(Ans) ↳ gives the relationship between i/p signal and o/p signal



→ It has input layer, hidden layer and output layer

→ Input layer : accepts i/p from other nods

→ hidden layer : presents in between i/p and o/p layer
: perform all calc. to find hidden feature

→ output layer : gives the result from the hidden layer

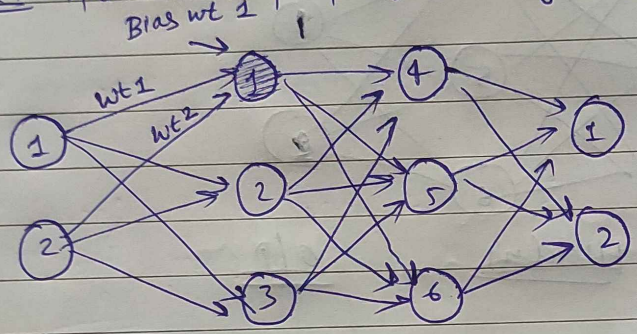
Application : To identify images or videos
: Text processing
: Speech Recognition

→ Forward propagation

↳ features are input to the network and fed through the layers to produce the o/p

→ The Activation of hidden layer is obtained by combination of bias weight 1 and weighted combination of I/p vars

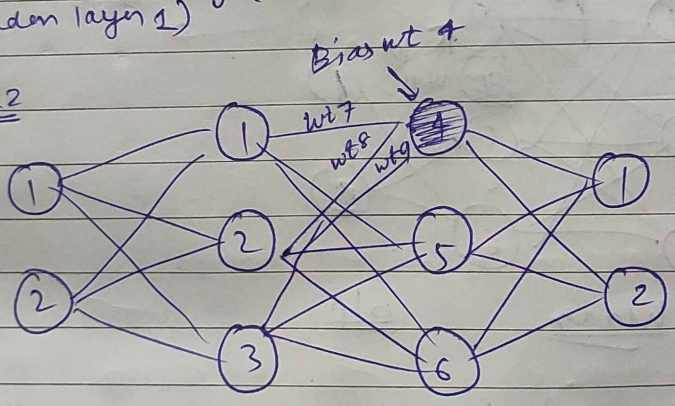
Case 1: forward propagation of layer 1 neurons



Input layer Hidden layer o/p layer

Activation (Hidden layer 1) = g (Bias wt 1 + wt 1 * input 1 + wt 2 * input 2)

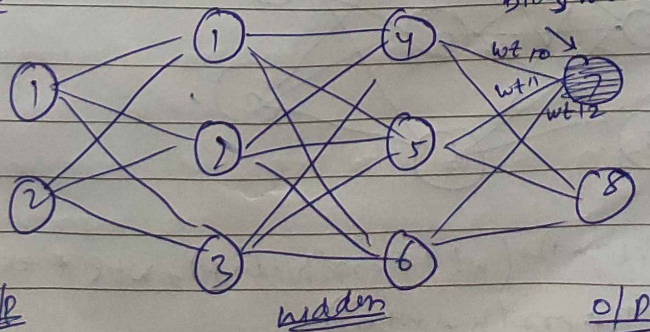
Case 2



Input layer Hidden layer O/p layer

Activation (hidden 2) = g (Bias wt 4 + wt 7 * hidden 1 + wt 8 * hidden 2 + wt 9 * hidden 3)

Case 3



I/p Hidden O/p

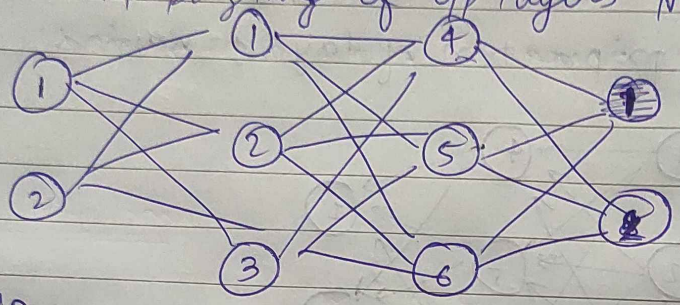
Activation = g(Bias wt 7 + wt 10 * hidden 4 + wt 11 * hidden 5 + wt 12 * hidden 6)
(output layer)

→ Back propagation

↳ It is used to check the error in Network in backward order

↳ Error is calculated by taking the derivative of output and multiply to error component

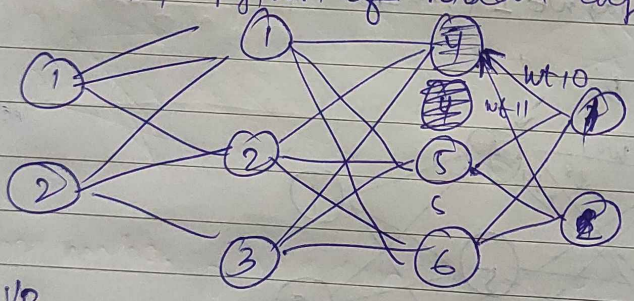
Case1 Backpropagating of o/p layers Neurons



I/P Hidden O/P

Output(1) = g'(output1) * (True1 - output1)

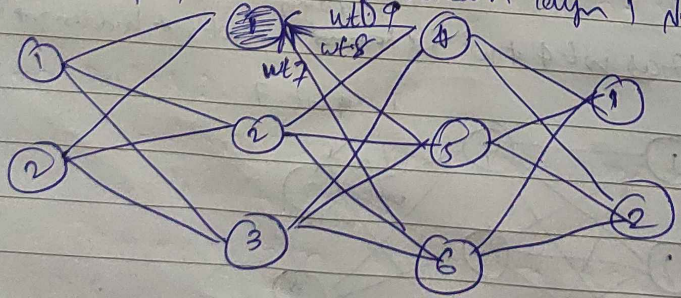
Case2 Backpropgation of hidden layer 2 Neurons



I/P Hidden O/P

error(hidden4) = g'(hidden4) * (wt10 * err(o/p1) + wt11 * err(o/p2))

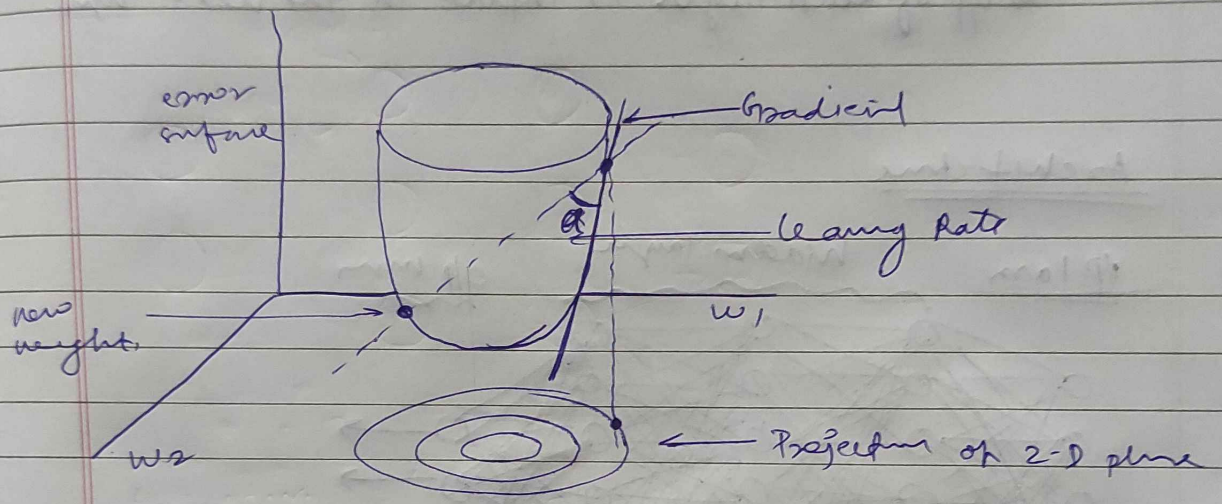
Case3 Backpropation of hidden layer 1 Neurons



error(hidden1) = g'(hidden1) * (wt7 * err(hidden4) + wt8 * err(hidden5) + wt7 * err(hidden6))

⑤ Stochastic Gradient descent - SGD

↳ way to optimize the neural network
↳ used to minimize the objective function
↳ the imp parameter here is the size of steps, which is calculated by learning rate
↳ If learning rate is small, then so many iteration possible and if it is slow, then algo will give optimal value



Types: ~~Batch~~ gradient: all training observation utilized
SGD: one observation per iteration
Mini batch: 50 observation each iteration

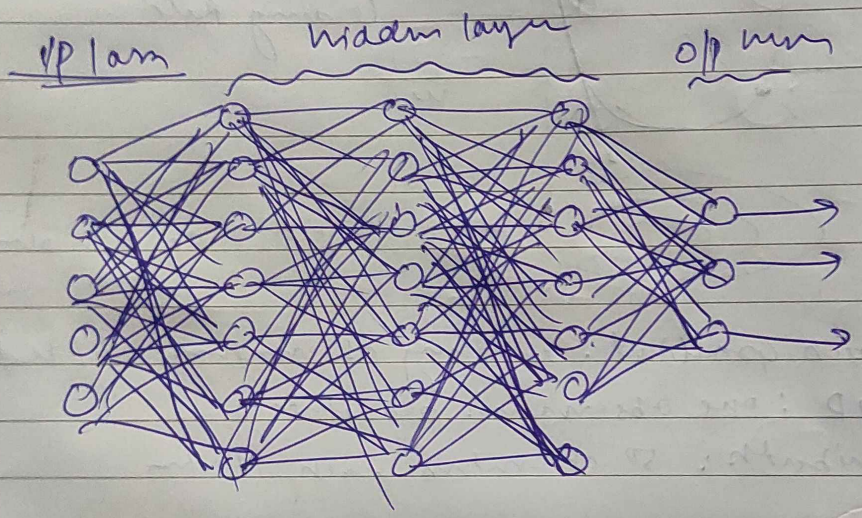
Adv: speed
: memory efficiency
: efficient

Disadv: slow
: Noise
: less accurate

④ Deep Architecture of Neural Networks

- deep learning is the type of ML algo
- used to build models to solve supervised and unsupervised problems.
- Deep neural network consist of multiple hidden layers between input & o/p layers
- each layer is fully connected with layers
- o/p of each layer is input in the next layer

Architecture



More number of hidden layers are present.

→ Here, backpropagation is used to solve ~~deep~~ the error by calculating error of network at o/p and propagates through layers to update the weights

→ Three Rules:

(i) All hidden layer should have same no. of neurons per layer
(ii) Two hidden layer are used to solve problems
(iii) Normalization is used to improve convergence effectively
(iv) Reduction of steps in each iteration improves convergence.

→ Deep learning software:

Qm

→ Keras model is used to develop the model
→ Keras models are easy to understand
→ other softwares are also used:
  * Theano : python based library
  * Tensorflow : google's dl library
  * Keras : faster model
  * caffe : dl library

Architecture:

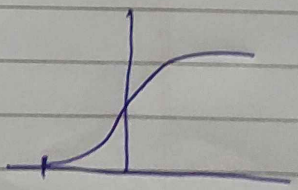
Keras
↓
Theano | TensorFlow
↓
Python

Qm

① Activation fns: used to process the info & pass through the network.

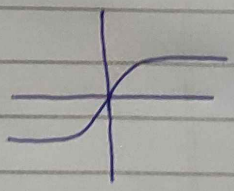
(1) Sigmoid

→ $A = \frac{1}{1+e^{-x}}$, Range is between 0 & 1



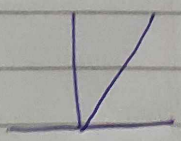
(2) Tanh

→ $\tanh(x) = \frac{2}{1+e^{-2x}} - 1$, Range : -1 to +1



(3) Relu

→ Rectified linear unit, $A(x) = \max(0, x)$, Range : $[0, \inf]$



(4) Linear Activation fns

→ $f(x) = x$,