EC460

NEURAL NETWORKS & DEEP LEARNING

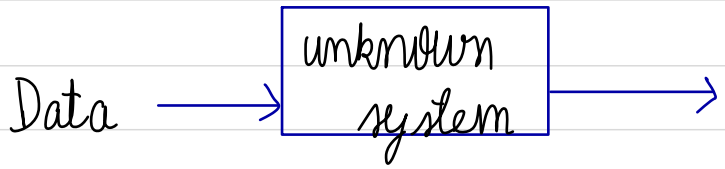
- M MANVITH PRABHU
211EC228

NEURAL NETWORKS AND DEEP LEARNING

Introduction

Consider the data $(x_1, y_1) = (1, 1)$, $(x_2, y_2) = (2, 2)$, $(x_3, y_3) = (3, 2)$

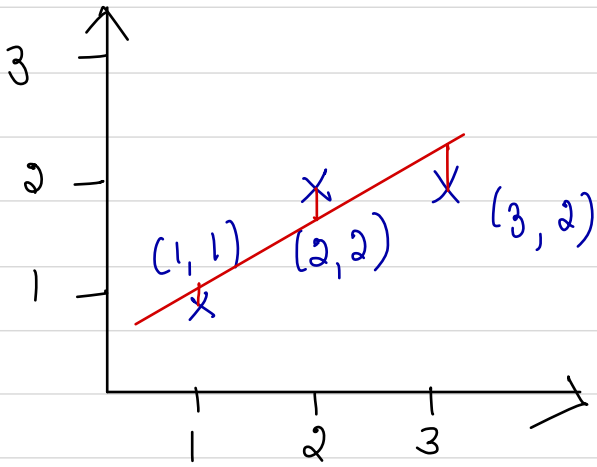
- Here x_1, x_2 and x_3 are inputs and y_1, y_2, y_3 are output.
- Also x_1, x_2, x_3 are called feature vectors.
- And y_1, y_2, y_3 are target values.



- Initially the system gives random output. We then iteratively change the system to get desired output.

Fitting a line to data (Linear regression)

$y = f(x)$
$= ax + b$



Here $a(1) + b = 1$
$a(2) + b = 2$
$a(3) + b = 2$

$$\Rightarrow \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

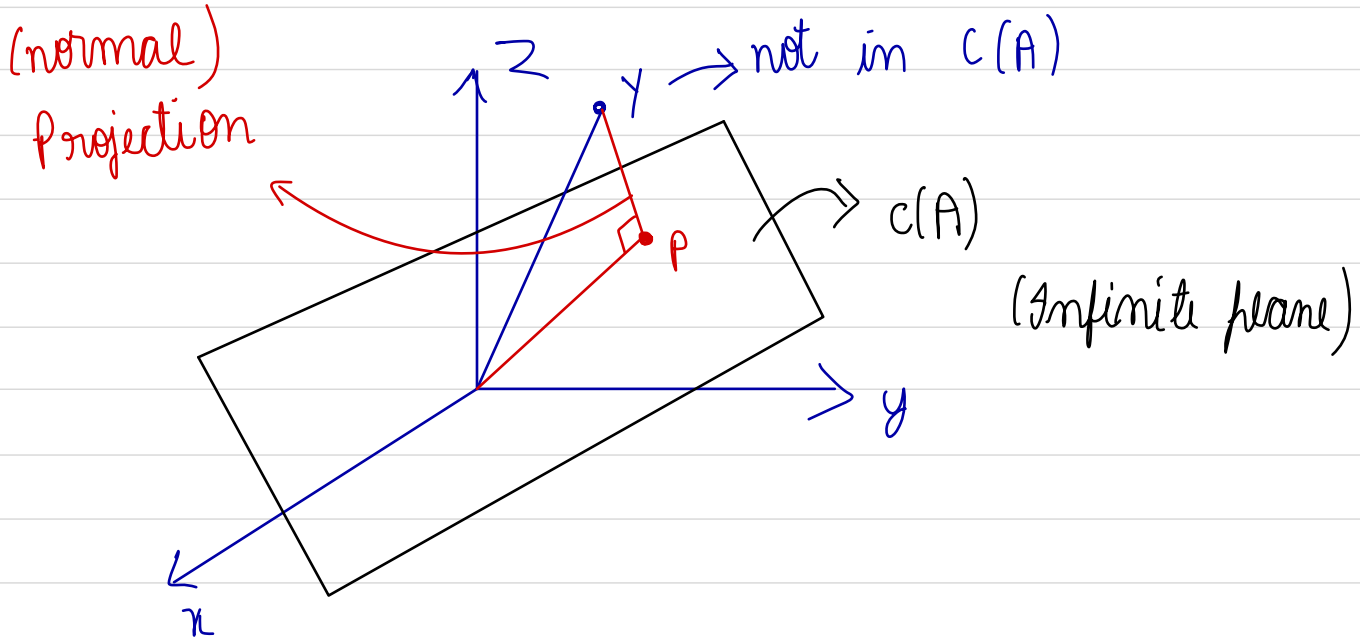
$A \quad X \quad = \quad Y$

$C(A)$ i.e column space of A

$$C(A) = a\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + b\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Here $a, b \in \mathbb{R}$

- For the perfect solution Y should lie in $C(A)$.
- But here Y does not lie in $C(A)$
- So no perfect solution exists.



Also $A^T A X = A^T Y$

$$\Rightarrow \hat{X} = (A^T A)^{-1} A^T Y$$

The best solution is given by

$$f(x) = \min_{X} \|AX - Y\|^2 \quad \rightarrow \text{(Scalar valued function)}$$

$f: \mathbb{R}^n \rightarrow \mathbb{R}$

So to find X: $\nabla \|AX - Y\|^2 = 0$

Solution is $X = (A^T A)^{-1} A^T Y$

If Y takes only 2 values i.e $(-1 \& 1)$ or $(0 \& 1)$ then it becomes binary classification. And $f(x)$ can act as linear discriminant function.

$$\rightarrow \min_{a,b} (1 - (a+b))^2 + (2 - (2a+b))^2 + (2 - (3a+b))^2$$

$\Rightarrow$ the solution to this is a^*, b^*

$$\Rightarrow \min_X \|AX - Y\|^2 = \min_X (AX-Y)^T (AX-Y)$$

$$= \min_X ((AX)^T - Y^T) \cdot (AX - Y)$$

$$= \min_X X^T A^T AX - X^T A^T Y - Y^T AX + Y^T Y$$

$$= \min_X \left(X^T A^T AX - 2A^T YX + Y^T Y\right)$$

Now taking gradient with respect to X and equating to 0.

$$2A^T AX - 2A^T Y = 0$$

$$\therefore \hat{X} = (A^T A)^{-1} A^T Y //$$

Q> Find a, b for least square solution from given example.

Ans $f(x) = \min_{a,b} (1-(a+b))^2 + (2-(2a+b))^2 + (2-(3a+b))^2$

$\frac{\partial f}{\partial a} = 0$

$\Rightarrow -2(1-(a+b)) - 2\times 2(2-(2a+b)) - 2\times 3(2-(3a+b)) = 0$

$1 - a - b + 4 - 4a - 2b + 6 - 9a - 3b = 0$

$\Rightarrow 14a + 6b = 11$

$\frac{\partial f}{\partial b} = 0$

$\Rightarrow -2(1-(a+b)) - 2(2-(2a+b)) - 2(2-(3a+b)) = 0$

$1 - a - b + 2 - 2a - b + 2 - 3a - b = 0$

$6a + 3b = 5$

$14a + 6b = 11$

$-(12a + 6b = 10)$

$2a = 1 \Rightarrow \hat{a} = 1/2$

$\Rightarrow 3 + 3b = 5 \quad \therefore \hat{b} = 2/3$ //

$\therefore f(x) = \frac{x}{2} + \frac{2}{3}$ //

Linear discriminant function:

If 2 classes are -1 & 1 i.e $-1 \rightarrow$ class 1 and $1 \rightarrow$ class 2.

Then $\hat{y} = \hat{f}(x) = a^* x + b^* > 0$

then $X \in$ class 2

otherwise $X \in$ class 1

$y = f(x) = a_1 x_1 + a_2 x_2 + b$ can be written as:

$$y = W^T X + b$$

where $W = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$ $\quad X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Also $y = \bar{W}^T \bar{X}$

where $\bar{W} = \begin{bmatrix} b \\ a_1 \\ a_2 \end{bmatrix}$ $\quad \bar{X} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$

Eg: Consider the points $(1.1, 2, 1.5)$, $(1.5, 2.5, 2)$ $(2, 2.5, 3)$ & $(2.2, 3.2, 3.3)$, find $f(x)$:

Ans $Y = f(x) = a_1 x_1 + a_2 x_2 + b$

$$\Rightarrow \begin{aligned} a_1(1.1) + a_2(2) + b &= 1.5 \rightarrow (1) \\ a_1(1.5) + a_2(2.5) + b &= 2 \rightarrow (2) \\ a_1(2) + a_2(2.5) + b &= 3 \rightarrow (3) \\ a_1(2.2) + a_2(3.2) + b &= 3.3 \rightarrow (4) \end{aligned}$$

Least square solution is given by $\hat{W} = (A^T A)^{-1} A^T Y$

NOTE:
- For this solution to exist $A^T A$ should be invertible. $\Rightarrow$ A is full column ranked i.e rank $(A) = n$ for $A_{m \times n}$
- Also $[(A^T A)^{-1} A^T] A = I$ $\rightarrow$ this is left inverse
  $A [A^T (A A^T)^{-1}] = I$ $\rightarrow$ This is right inverse

$$A^T A = \begin{bmatrix} 1.1 & 1.5 & 2 & 2.2 \\ 2 & 2.5 & 2.5 & 3.2 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1.1 & 2 & 1 \\ 1.5 & 2.5 & 1 \\ 2 & 2.5 & 1 \\ 2.2 & 3.2 & 1 \end{bmatrix}$$

$$A^TA = \begin{bmatrix} 12.3 & 17.9 & 6.8 \\ 17.9 & 26.74 & 10.2 \\ 6.8 & 10.2 & 4 \end{bmatrix}$$

$$(A^TA)^{-1}A^T = \begin{bmatrix} 3.22 & -2.47 & 0.82 \\ -2.47 & 3.26 & -4.12 \\ 0.82 & -4.12 & 9.36 \end{bmatrix} \begin{bmatrix} 1.1 & 1.5 & 2 & 2.2 \\ 2 & 2.5 & 2.5 & 3.2 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -0.57 & -0.52 & 1.09 \\ -0.3 & 0.33 & -0.9 \\ 2.02 & 0.29 & 0.703 \end{bmatrix}$$

$$\therefore \hat{W} = (A^TA^{-1})A^TY = \begin{bmatrix} 1.38 \\ 0.4 \\ -0.93 \end{bmatrix}$$

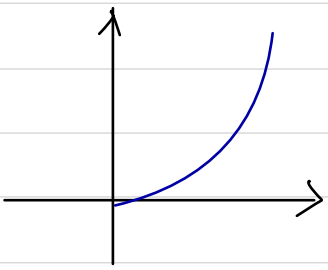
$\therefore \quad a_1 = 1.38 \qquad a_2 = 0.4 \qquad b = -0.93$ //

$\therefore \quad f(x) = 1.38x_1 + 0.4x_2 - 0.93$ //

Perceptron:

It is a system that can generate a function $f(x)$ of form $f(x) = W^TX + b = a_1x_1 + a_2x_2 + \dots + b$

Consider the function:



We can use $f(x) = b + a_1x + a_2x^2$

Here basis is $1, x, x^2$. But x^2 needs to be provided. Advantage with neural networks is that x^2 can be generated, i.e basis can be generated.

Eg: Consider the points $(1,1)$, $(2,2)$ & $(3,2)$ and $f(x) = b + a_1 x + a_2 x^2$

$$f(1) = b + a_1 + a_2 = 1$$
$$f(2) = b + 2a_1 + 4a_2 = 2$$
$$f(3) = b + 3a_1 + 9a_2 = 2$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \begin{bmatrix} b \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

$$\hat{W} = (A^T A)^{-1} A^T Y = \begin{bmatrix} -1 \\ 2.5 \\ -0.5 \end{bmatrix}$$

$\therefore f(x) = 2.5x - 0.5x^2 - 1$

- Now consider $(5,1)$ which is a test data.
- But $(5,1)$ does not fit into $2.5 - 0.5x^2 - 1$.

⇒ This is because of overfitting.

To prevent overfitting, we use regularisation.

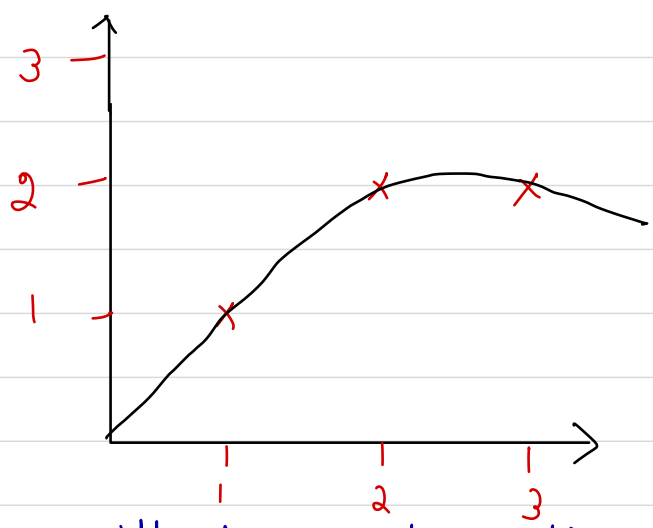
i.e $\min_W \|AW - Y\|^2 + \lambda \|w\|_2$

↓ regularisation parameter

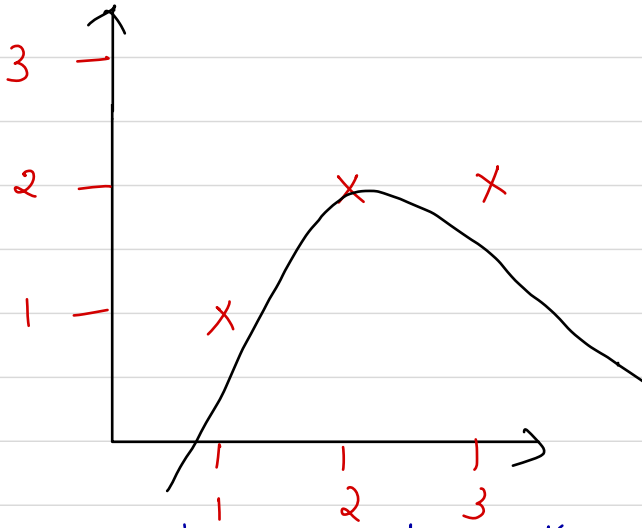
→ If we use $\|w\|_1$, i.e L1 norm, it becomes sparse regularisation.

Solution of $\min_W \|AW - Y\|^2 + \lambda \|w\|_2$

is $\hat{W} = (A^T A + I\lambda)^{-1} A^T Y$



without regularisation



with regularisation.

In Neural networks, drop out layers are used.

For one dimensional data, $X \in \mathbb{R}$, general

$f(x) = a_0 + a_1 X + a_2 X^2 + a_3 X^3 + \ldots$

For $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $X \in \mathbb{R}^2$,

$f(x) = a_0 + a_1 x_1^2 + a_2 x_2^2 + a_3 x_1 x_2 + a_4 x_1 + a_5 x_2$

$f(x) = X^T A X + C^T X + a_0$.

where A is positive definite matrix.

Positive definite matrix: $X^T A X > 0$ for every X then A is positive definite matrix.

Both λ_1, λ_2 (eigenvalues of A) must be positive.

$X^T A X + C^T X + a_0 = 1$ will give ellipse.

In higher dimension it will give ellipsoid.

If A is not positive definite matrix then it will be hyperbola.

For n dimension, $f(x) = X^T A X + C^T X + A_0$

Linear programming

$f(x) = q_2(x) - q_1(x)$ where the $q_2(x)$ is posterior probability for class 2.

where the q_1 is posterior probability of class 1.

If $f(x) > 0$ then $X \in$ class 2
otherwise $X \in$ class 1.

$q_1(x) = p(X/C_1)\, p(C_1) \rightarrow$ Bayes' classifier.

NOTE: → Theoretically Bayes' classifier is best.
→ However $p(X|C_1)$, $p(C_1)$, $p(X|C_2)$, $p(C_2)$ is not easily available and should be approximated. Hence Bayes' classifier is not generally used.

To estimate density function:
Implicitely: i> PCA
ii> Autoencoders
iii> Variational autoencoders.

Explicitely: i> Parametric. (estimate μ, σ in Gaussian etc.)
ii> Non parametric
(i. Parzan window)

$$p(x/C_1) = \frac{1}{\sqrt{2\pi}\,\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}}$$

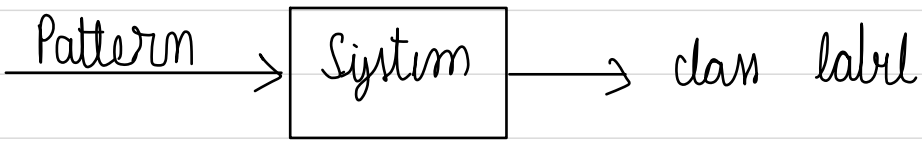
$$p(x/C_2) = \frac{1}{\sqrt{2\pi}\,\sigma_2} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}$$

$f(x)$ becomes linear when $\sigma_1 = \sigma_2 = \sigma$

Linear classifiers : 1> SVM with linear kernel.
2> Neural Networks
3> Random forest

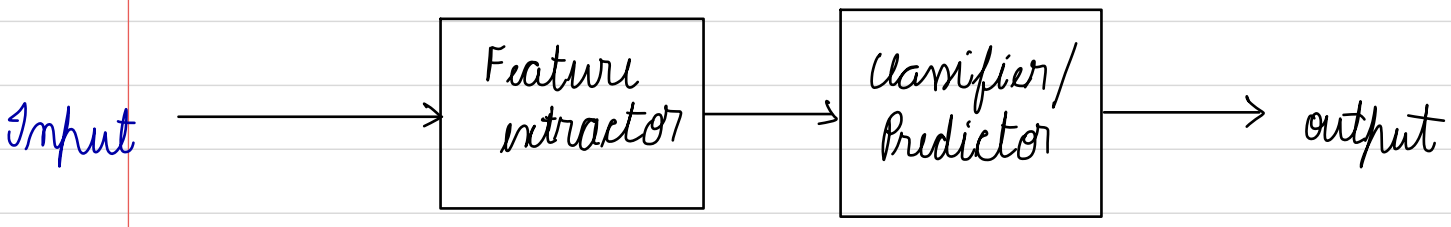
Non linear classifiers : 1> Neural networks
2> SVM with other kernels

Recognition system



Eg : 1> Reading
2> Speech recognition
3> Face recognition.

Block diagram of system that learns and recognises task

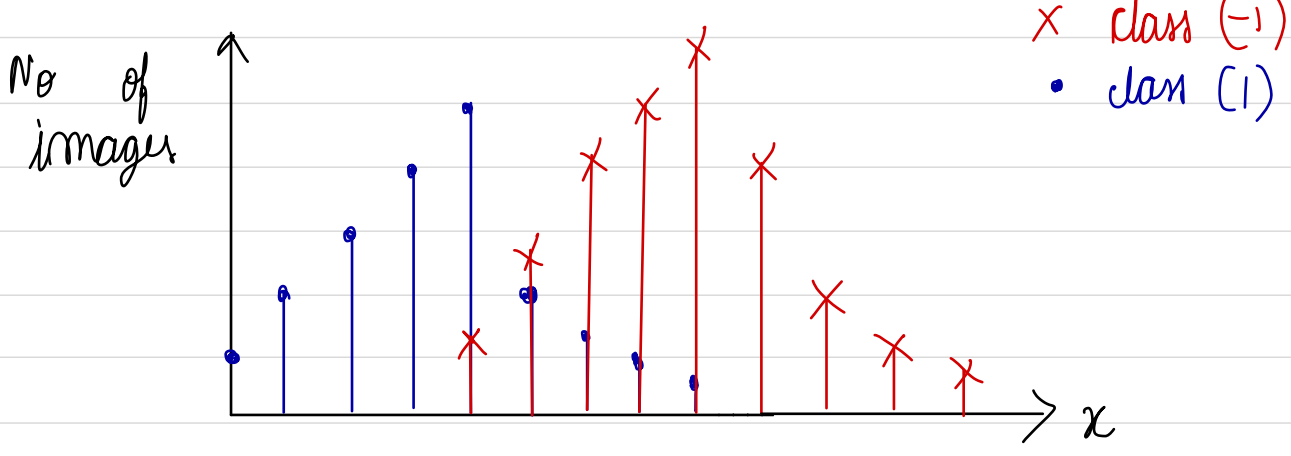


Why classification is so difficult ?:

Consider we have 1000 images out of which 500 are benign ($y=1$ class) & 500 are malignant ($y=-1$ class)

Taking x = mean of pixel values of image.

Eg: Histogram of x:

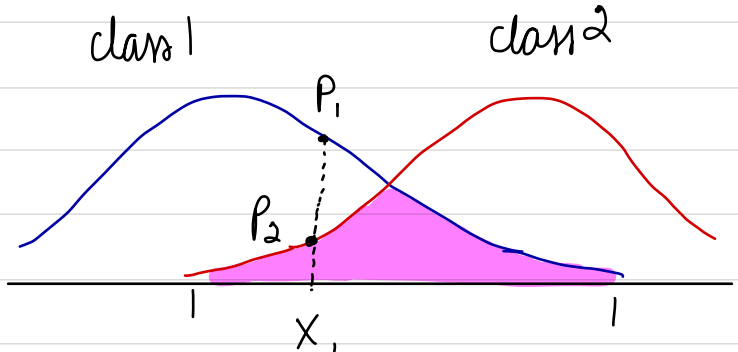


1> This shows that there is a huge variability in feature vector values within the class which makes the classification difficult.
2> Feature vector of pattern / signal from different classes can be arbitarily close or overlapped.

3> Noisy measurements.

Bayes' classifier:

class conditional density function : prior probability



↳ if we choose a sample here it may belong to class 1 or class 2.

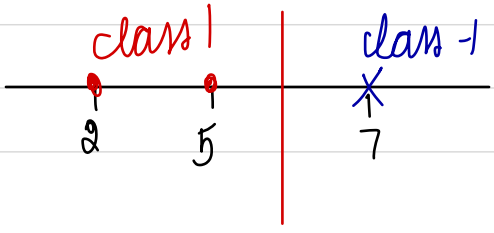
We have to use minimum error classifier.

Consider 3 samples: (2, 1), (7, -1), (5, 1)
class 1 class -1 class 1

As the samples are very less, Bayes classifier cannot be implemented

so linear or non linear classifier can be used.

Q> Design a linear classifier using Least square method:



$$f(x) = a_0 + a_1 x$$

$$2a_1 + a_0 = 1$$
$$5a_1 + a_0 = 1$$
$$7a_1 + a_0 = -1$$

$$\hat{W} = (A^T A)^{-1} A^T Y$$

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 5 \\ 1 & 7 \end{bmatrix} \qquad Y = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

$$\hat{W}^T = \begin{bmatrix} \frac{78}{38} & \frac{-14}{38} \end{bmatrix}$$

$$\therefore y = \frac{39}{19} - \frac{7}{19}x$$

$\therefore$ If $x < \frac{39}{7}$, $x \in$ class 1 else belongs to class -1.

This is overfitting.

Q> Design a non linear classifier using Least square method for (1, 1), (2, -1) & (3, 1)

Ans Using $f(x) = a_0 + a_1 x + a_2 x^2$

Using least square method:

$y = f(x) = 7 - 8x + 2x^2$,

if $y > 0$ then $x \in$ class 1 else it belongs to class -1

Here training error = 0, $\Rightarrow$ overfitting.

Linear classifier may have training error but may perform better on test set.

Hence we cannot say that non linear is always better than linear.

Bayes' classifier: (two class classifier)

$h(x) = 1$ if $q_1(x) > q_2(x)$

$= 0$ or (-1), otherwise

where $q_1(x) = P(C_1|x) = \dfrac{p(x|C_1)\,P(C_1)}{p(x|C_1)\cdot p(C_1) + p(x|C_2)\,p(C_2)}$

$$P(x) = \sum_{i=1}^{2} P(x, C_i)$$

$$= P(x, C_1) + P(x, C_2)$$

$$= P(x|C_1)\,P(C_1) + P(x|C_2)\,P(C_2)$$

$$q_2(x) = \frac{P(x|C_2)\,P(C_2)}{P(x)}$$

$q_1(x)$ & $q_2(x)$ are posterior probabilities of class 1 and class 2.

How do we implement the Bayes' classifier:

Data: { Training data - 1000 images
        Test data

↘

500 - Benign

500 - Malignant

If $P(C_1) = P(C_2) = 1/2$ then Bayes' estimation becomes Maximum likelihood estimation.

To implement Bayes' classifier we need:
1> Priors $(P(C_1), P(C_2))$
2> class condiontal probability : $P(x|C_1)$, $P(x|C_2)$ etc.

Parametric method:

Gaussian distribution: $\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

μ_1 is mean value of the 500 mean values of pixels. σ_1 is the standard deviation of the 500 mean values of class 1.
For class 1:
Random variables $\left\{ \begin{array}{ll} \hat{\mu}_1 & \text{for } \mu_1 \\ \hat{\sigma}_1 & \text{for } \sigma_1 \end{array} \right\}$

$\hat{\mu}_1$ estimator for μ_1
$\hat{\sigma}_1$ estimator for σ_1

$$P(x|C_1) = \frac{1}{\sqrt{2\pi}\,\hat{\sigma}_1} e^{-\frac{(x-\hat{\mu}_1)^2}{2\hat{\sigma}_1^2}}$$

$$\hat{\mu}_1 = \frac{1}{500} \sum_{i=1}^{500} x_i \qquad x_i \in \text{class 1}$$

] maximum likelihood approach.

Estimator is the sample mean
↳ random variable

$$\hat{\sigma}_1^2 = \frac{1}{500} \sum_{i=1}^{500} (x_i - \hat{\mu}_1)^2 \quad , \quad x_i \in \text{class 1}$$

$\rightarrow h(x) = 1$ if

$$\log(p(x|C_1)) > \log(p(x|C_2))$$

i.e $$\log\left(\frac{1}{\sqrt{2\pi}\,\hat{\sigma}_1} e^{-\frac{(x-\hat{\mu}_1)^2}{2\hat{\sigma}_1^2}}\right) > \log\left(\frac{1}{\sqrt{2\pi}\,\hat{\sigma}_2} e^{-\frac{(x-\hat{\mu}_2)^2}{2\hat{\sigma}_2^2}}\right)$$

$$-\ln\hat{\sigma}_1 - \frac{1}{2}\ln(2\pi) - \frac{(x-\hat{\mu}_1)^2}{2\hat{\sigma}_1^2} > -\ln\hat{\sigma}_2 - \frac{1}{2}\ln(2\pi) - \frac{(x-\hat{\mu}_2)^2}{2\hat{\sigma}_2^2}$$

$\therefore$ $h(x) = 1$ ($x \in$ class 1 implies):

$$\frac{x^2}{2}\left(\frac{1}{\hat{\sigma}_1^2} - \frac{1}{\hat{\sigma}_2^2}\right) + x\left(\frac{\hat{\mu}_1}{\hat{\sigma}_1^2} - \frac{\hat{\mu}_2}{\hat{\sigma}_2^2}\right) + \frac{1}{2}\left(\frac{\hat{\mu}_1^2}{\hat{\sigma}_1^2} - \frac{\hat{\mu}_2^2}{\hat{\sigma}_2^2}\right) + \ln\left(\frac{\hat{\sigma}_1}{\hat{\sigma}_2}\right) > 0$$

if $\hat{\sigma}_1 = \hat{\sigma}_2 = \hat{\sigma}$

$$x\,\frac{(\hat{\mu}_1 - \hat{\mu}_2)}{\hat{\sigma}^2} + \frac{1}{2}\,\frac{(\hat{\mu}_1 - \hat{\mu}_2)(\hat{\mu}_1 + \hat{\mu}_2)}{\hat{\sigma}_2^2} > 0$$

$\therefore$ $x < \frac{\hat{\mu}_1 + \hat{\mu}_2}{2}$ then $h(x) = 1$ otherwise 0/-1

For the available data we cannot find the true densities.

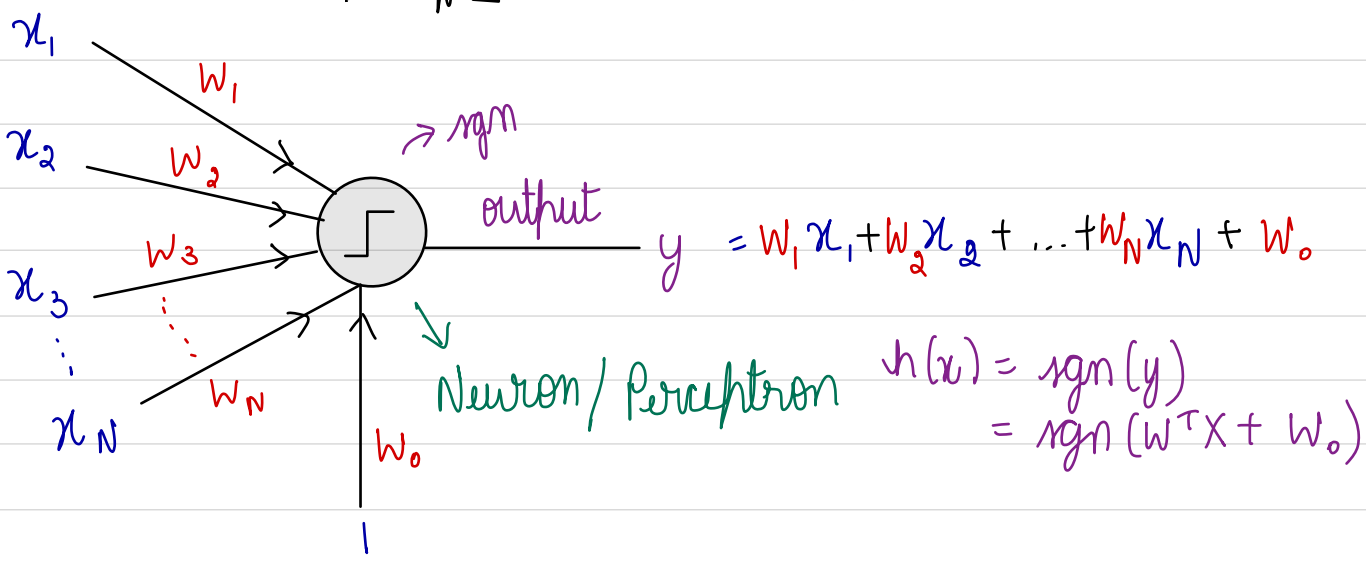
Perceptron model

Linear classifier: $h(x) = 1$ if $W^T X + a_0 > 0$
$h(x) = 0$ (or -1), otherwise

where $W = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_N \end{bmatrix}$ $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix}$

One can use the homogeneous co-ordinates:
$h(x) = 1$ if $\bar{W}^T \bar{X} > 0$
$h(x) = 0$ otherwise

where $\bar{W} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}$ $\bar{X} = \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}$ where $a_i, x_i \in \mathbb{R}$



Perceptron model was created by Rosenblatt in 1962

To train, we can use following linear classifiers:
1) Least square method
2) Perceptron algorithm
3) Fischer linear discriminant analysis

However Perceptron model works only if data is completely separable (linearly separable).

The training set $\{(x_i, y_i)\}$, $i = 1, 2, \ldots, n$, such a set of pattern is said to be linearly separable.

If there exists w^* such that

$x^T w^* > 0$ if $y_i = 1$

$x^T w^* < 0$ if $y_i = 0$ or $y_i = -1$.

→ So perceptron algorithm converges if data is linearly separable as it is an iterative algorithm.

→ The aim of the algorithm is to find w^*. w^* is a hyperplane that separates 2 classes

→ Let us say $w(k)$ denotes the weight vector at k^{th} iteration. At each iteration we update the w.

$w(k+1) = w(k)$, if classification is correct

or $w(k+1) = w(k) + x(k)$, if $w^T x \leq 0$ & $y = 1$

or $w(k+1) = w(k) - x(k)$, if $w^T x \geq 0$ & $y = 0$ or (-1)

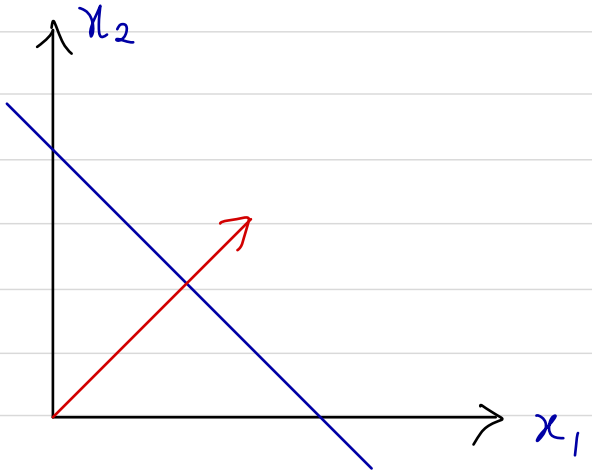
But it is not clear how the algorithm works:
i) There is no guarantee that $w^T(k+1)x(k)$ has correct sign.
ii) Let $w(k)$ classify $x(k-1)$, $x(k-2)$ correctly, $w(k+1)$ might not classify them correctly.

This algorithm will converge for linearly seperable data, only if $k \to \infty$, $w(k+1) \to w^*$

$$f(x, w) = w^T x + w_0 \quad \to \quad \bar{w}^T \bar{x} > 0 \quad \to \quad \bar{w}^T \bar{x} = 0$$

↓

plane of seperation



$f(x) = 0$ is also called contour or level set or hyperplane

Consider $w_2 x_2 + w_1 x_1 - w_0 = 0$

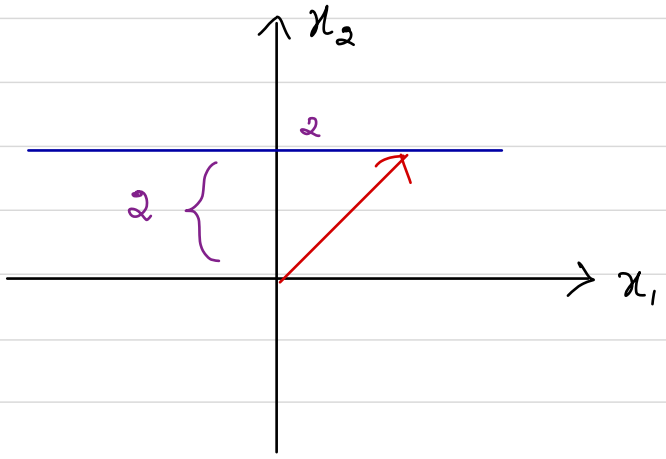
$$\frac{w_1 x_1 + w_2 x_2}{\sqrt{w_1^2 + w_2^2}} = \frac{w_0}{\sqrt{w_1^2 + w_2^2}}$$

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \perp w^T x + w_0 = 0$$

$$\frac{\perp}{\|w\|} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \to \text{unit vector}$$

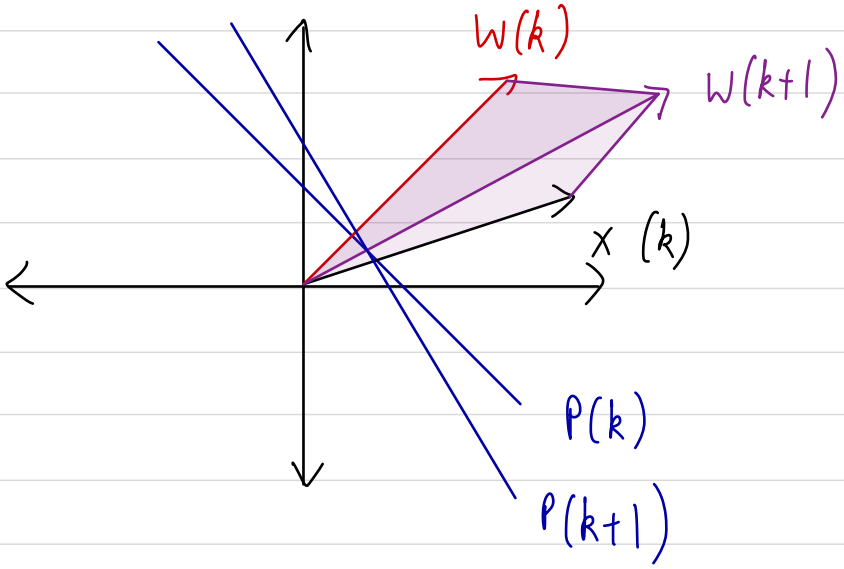
$$\Rightarrow \frac{1}{\|W\|}[x_1 \; x_2]\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \frac{w_0}{\|W\|}$$

Eg :



$$[x_1 \quad x_2]\begin{bmatrix} 0 \\ 1 \end{bmatrix} = 2$$

Let $W(k+1) = W(k) + X(k)$



Input → logistic regression → output

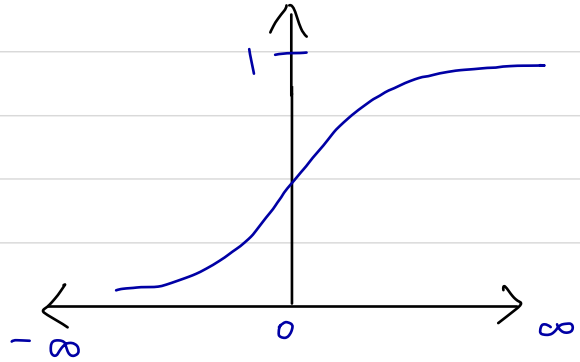
↓

sigmoid is used.

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$$

$x = -\infty \rightarrow 0$

$x = \infty \rightarrow 1$



Two class classifier

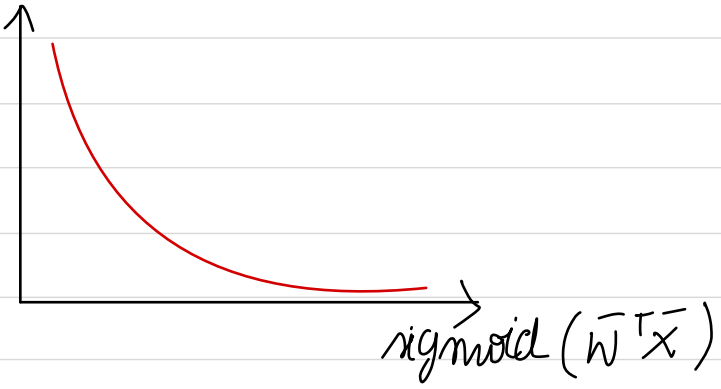
$\hat{y} = h(x) = 1$ if sigmoid $(\bar{w}^T \bar{x}) \geq 0.5$

0 or -1, if sigmoid $(\bar{w}^T \bar{x}) < 0.5$

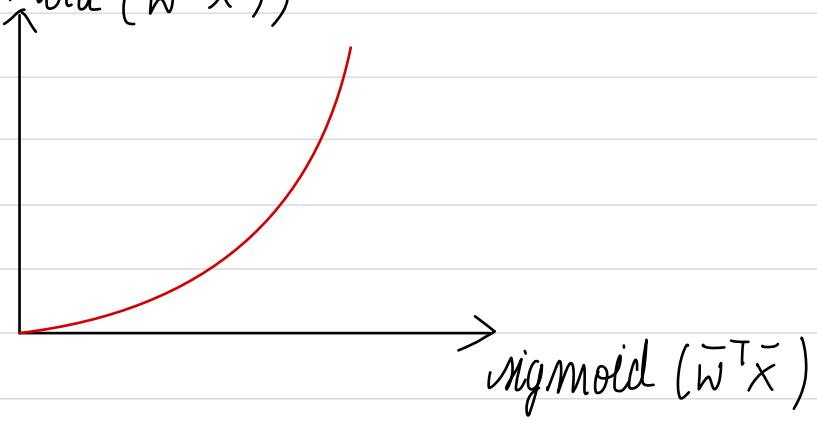
How to train this:

Least square method: Get cost function to train the function with sigmoid $(\bar{w}^T \bar{x})$

For class 1, i.e label $y = 1$, consider cost = $-\log(\text{sigmoid}(\bar{w}^T \bar{x}))$



For class 2 i.e label $y = 0$ or -1, consider cost: $-\log(1 - \text{sigmoid}(\bar{w}^T \bar{x}))$



∴ Total cost / loss function:

$$L(w) = -y \log(p) - (1-y) \log(1-p)$$

where $p = \text{sigmoid}(\bar{w}^T \bar{x})$

This loss is called Binary cross entropy or BCE loss.

Cross entropy:

Estimated distribution		True distribution
$\text{sigmoid}(\bar{w}^T\bar{x})$	$\Rightarrow$	1
$1 - \text{sigmoid}(\bar{w}^T\bar{x})$	$\Rightarrow$	0

If X follows the distribution $p_X(X)$, then

$$\text{cost function} = \min_w E(L(w))$$

$$= \min_w -\sum_{i=1}^{N} p_X(X_i)\left(Y_i \log(\text{sigmoid}(\bar{w}^T\bar{x}_i)) + (1-Y_i)\log(\text{sigmoid}(1-\bar{w}^T\bar{x}_i))\right)$$

This is called logistic regression.

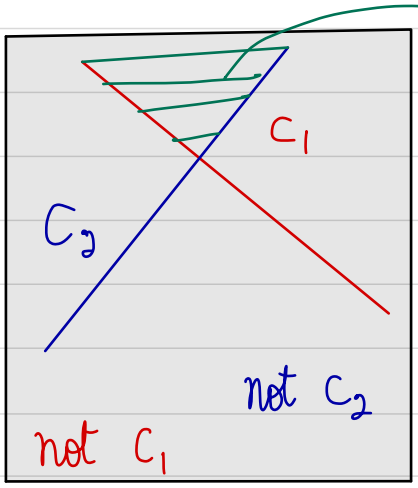
Multiclass problem

Goal is to create multiclass classifier that can classify k classes.

Earlier approaches:

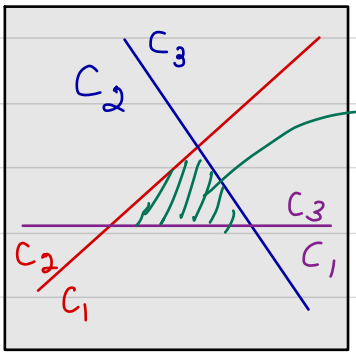
1) One v/s rest: We design k classifiers.
   1st classifier: C_1 and not C_1
   2nd classifier: C_2 and not C_2
   ⋮
   k^{th} classifier: C_k and not C_k

i.e we are using k 2-class linear classifiers.



region of ambiguity (causes problem)

2> one v/s one: $\frac{k(k-1)}{2}$ 2 class classifiers are needed.



region of ambiguity (causes problem)

These approaches can not generalize the linear discriminant function for multi class classification

To solve this: define the multiclass classification using linear discriminant function.

$$L_s(X, W) = W_s^T X + W_{os}, \quad s = 1, 2, 3, \ldots, k$$

X (feature vector) $\in C_j$

If $L_j(X_i, \bar{W}) \geq L_s(X_i, W)$

We compute $W_s^T X + W_{os}$ for $s = 1, 2, 3, \ldots, k$

if $L_2 >$ rest then $X_i \in C_2$

If $X_i \in C_j$ then Y_i would be a $k \times 1$ vector with j^{th} component = one and rest being zeroes.

$$\text{Error} = \left\| \begin{bmatrix} W_1^T X_i + W_{01} \\ W_2^T X_i + W_{02} \\ \vdots \\ W_K^T X_i + W_{0k} \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \right\|^2$$

$$g_j(x) = \frac{\exp(W_j^T x + W_{0j})}{\sum_{i=1}^{k} \exp(W_i^T x + W_{0i})} \rightarrow \text{softmax}$$

Range (0, 1)

$$q_j(x) = g_j(x)$$

Softmax function normalizes the value at the output of neural network.

During training cross entropy loss is minimised

Softmax generates k probabilities for k class classification

$$\text{Estimated probabilities} = \begin{bmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_j(x) \end{bmatrix}$$

By softmax function,

if $g_2(x) > g_1(x)$ & $g_2(x) > g_3(x) \ldots g_k(x)$

Then classifier says $x \in$ class 2.
But true class maybe class 3.

Cross entropy loss:

→ Loss = $y_{i1} \log(g_1(x_i)) + y_{i2} \log(g_2(x_i)) + \ldots + y_{ik} \log(g_k(x_i))$

→ If $x_i \in C_j$ the loss can be written as
Loss = $y_{ij} \log(g_j(x_i)) = \log(g_j(x_i))$ //

→ For all the samples, samples follows a particular distribution, cross entropy is Expectation i.e $E_{P_{x_i}(x)}(\log g_j(x_i))$

Training ⇒ $\min_{w} E_{P_{x_i}(x)}(\log(g_j(x_i)))$

We have to use optimization algorithm to minimize the loss function

Iterative algorithm such as Back propagation is used i.e

$$W(k+1) = W(k) - \eta \frac{\partial Error}{\partial W(k)}$$

We need to find Partials.

Back propagation algorithm is used to find partials and update weights

Assignment:

1) Write a note on Liquid Neural Networks. i.e difference between Liquid NN & static (normal) NN's

Project: Application of LNN.

Neural networks:

Motivation: (x_1, x_2, y)

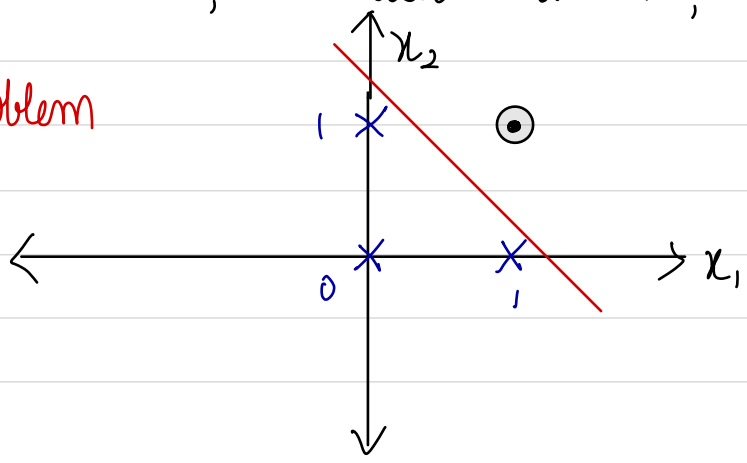
Data 1: $\{(0,0,0), (0,1,0), (1,0,0), (1,1,1)\}$

Data 2: $\{(0,0,0), (0,1,1), (1,0,1), (1,1,1)\}$

Data 3: $\{(0,0,0), (0,1,1), (1,0,1), (1,1,0\}$

Data 1 is AND, Data 2 is OR, Data 3 is XOR

AND problem



$f(x) = W^T X + W_0$

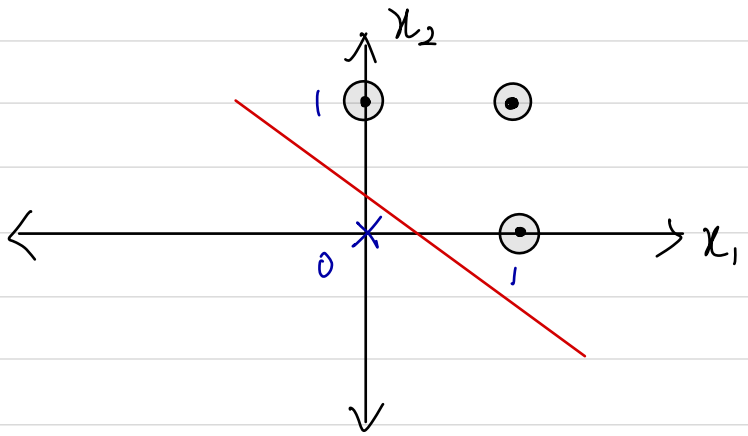
Linear classifier : $h(x) = 1$ if $W^T X + W_0 > 0$

$h(x) = 0$, otherwise

Or $h(x) = 1$ if sigmoid $(W^T X + W_0) > 0.5$

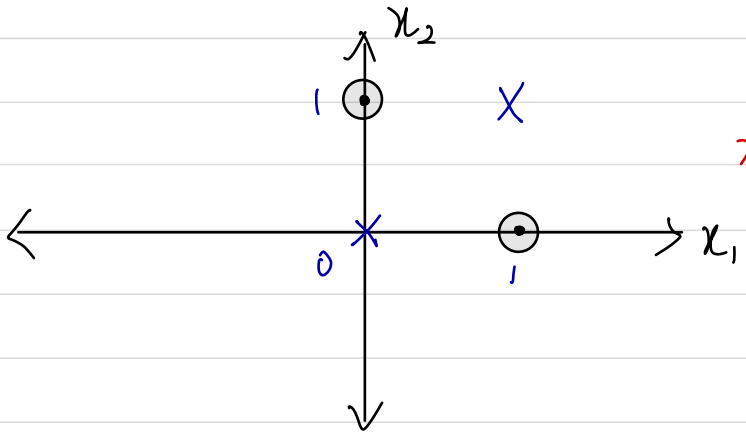
$h(x) = 0$, otherwise

Data 2 :



OR problem

Data 3 :



XOR problem

This can be sperated by non linear classifiers such as: 1> SVM
2> Random forest
3> Neural networks

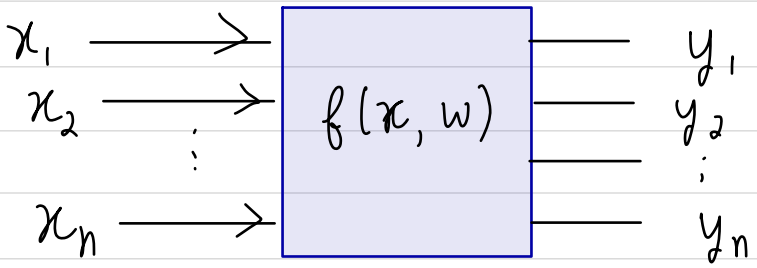
Data 3 can be sperated linearly by projecting it to higher dimension.

To solve XOR problem, classifiers from AND and OR problem can be combined.

We can generate non linear function using least square method.

Simplest 1D non linear function: $a_0 + a_1 x + a_2 x^2$

Neural network system:



$f(x, w) = \mathbb{R}^n \rightarrow \mathbb{R}^m$

Non linear function can be used to classify the linearly non sperable classes using neural networks.

Tasks that can be performed using neural networks:

1) classification : 2 class or multiclass
$f(x, w) : \mathbb{R}^n \rightarrow (0, 1)$
2) Regression : $f(x, w) : \mathbb{R}^n \rightarrow \mathbb{R}^m$
3) Unsupervised learning
4) Compression

Non linear functions are built up through

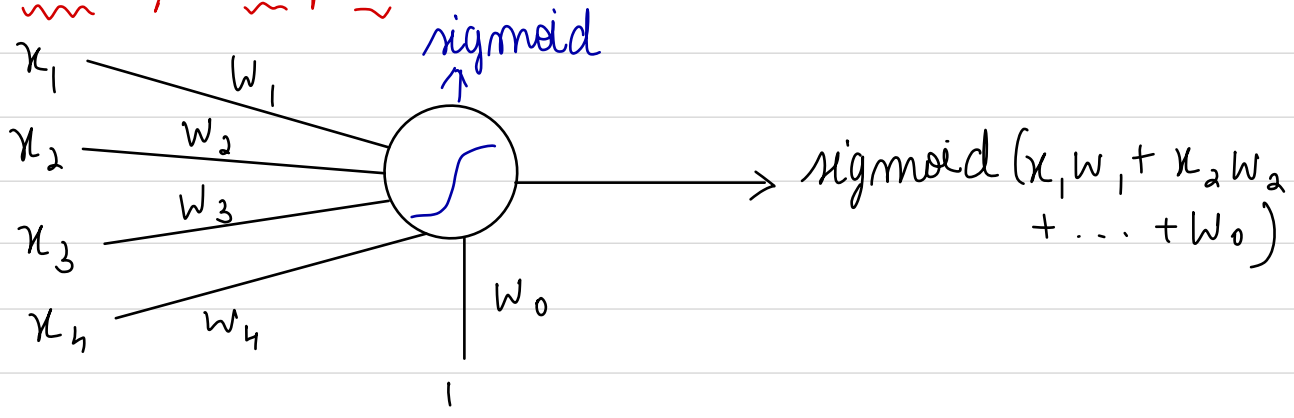
composition of summation & sigmoid

Neural networks are feed forward networks.

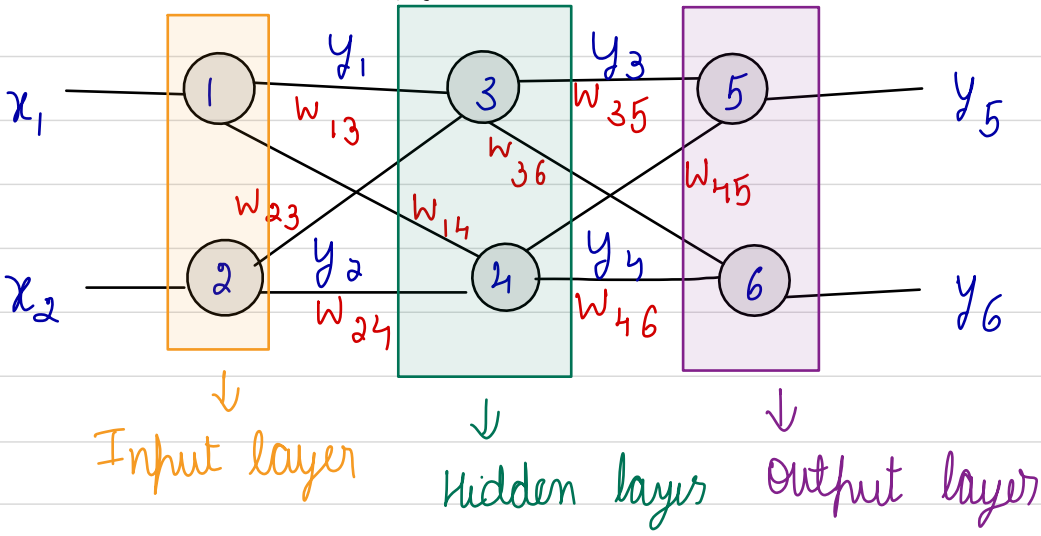
Description of Neural networks

1> A large network of interconnected units.
2> Each unit has simple input - output mapping
3> Each interconnection has numerical weight attached to it.
4> outputs of units dpend on outputs and connection weights of units connected to it.
5> Knowledge resides in the weights
6> Problem solving ability is often through learning.

Neuron / Perceptron



Consider a neural network:

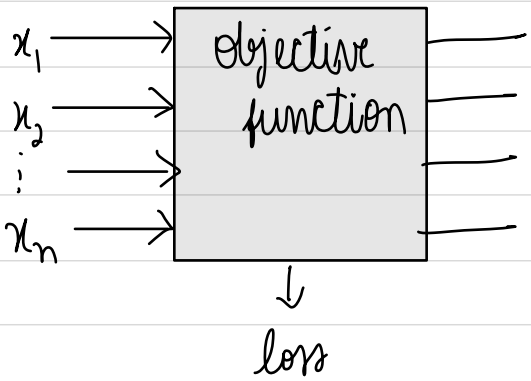


This a fed forward network

→ Artificial neural networks provide a good parameterized class of non linear functions to learn non linear classification.

→ Non linear functions are built up through composition of summation & sigmoids, which can be used both for classification and regression.

→ How to train a Neural Networks ?



objective

↓

Error function

We cannot use Least square method as we don't known the basis function.

→ So we use steepest descent method: It is an iterative method. At each iteration we update parameters.

$$W(k+1) = W(k) - \lambda \nabla_{W} (\text{error function})$$

At k^{th} iteration:

$$W_{ij}(k+1) = W_{ij}(k) - \lambda \frac{\partial \text{Error}}{\partial W_{ij}}$$

↓ Stepsize

Back - Propagation:

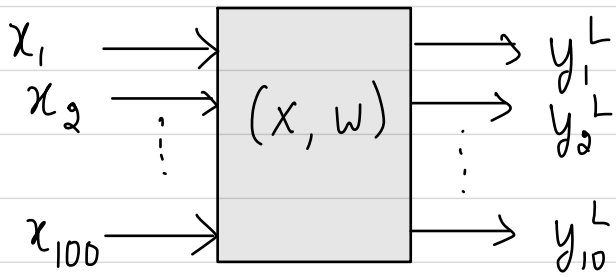
1> First compute partials at the output
2> Then move backwards towards input.

- consider a multilayer feed foward networks

→ L - Number of layers
→ n_l - number of nodes in layer 'l' where $l = 1, 2, 3, \ldots L$
→ y_i^l - output of i^{th} node in layer 'l'
→ w_{ij}^l - weight of connection from node 'i' of layer 'l' to the j^{th} node of layer 'l+1'
→ η_i^l - net output of node 'i' of layer 'l'

NOTE: $y_i^l = \text{activation}(\eta_i^l)$

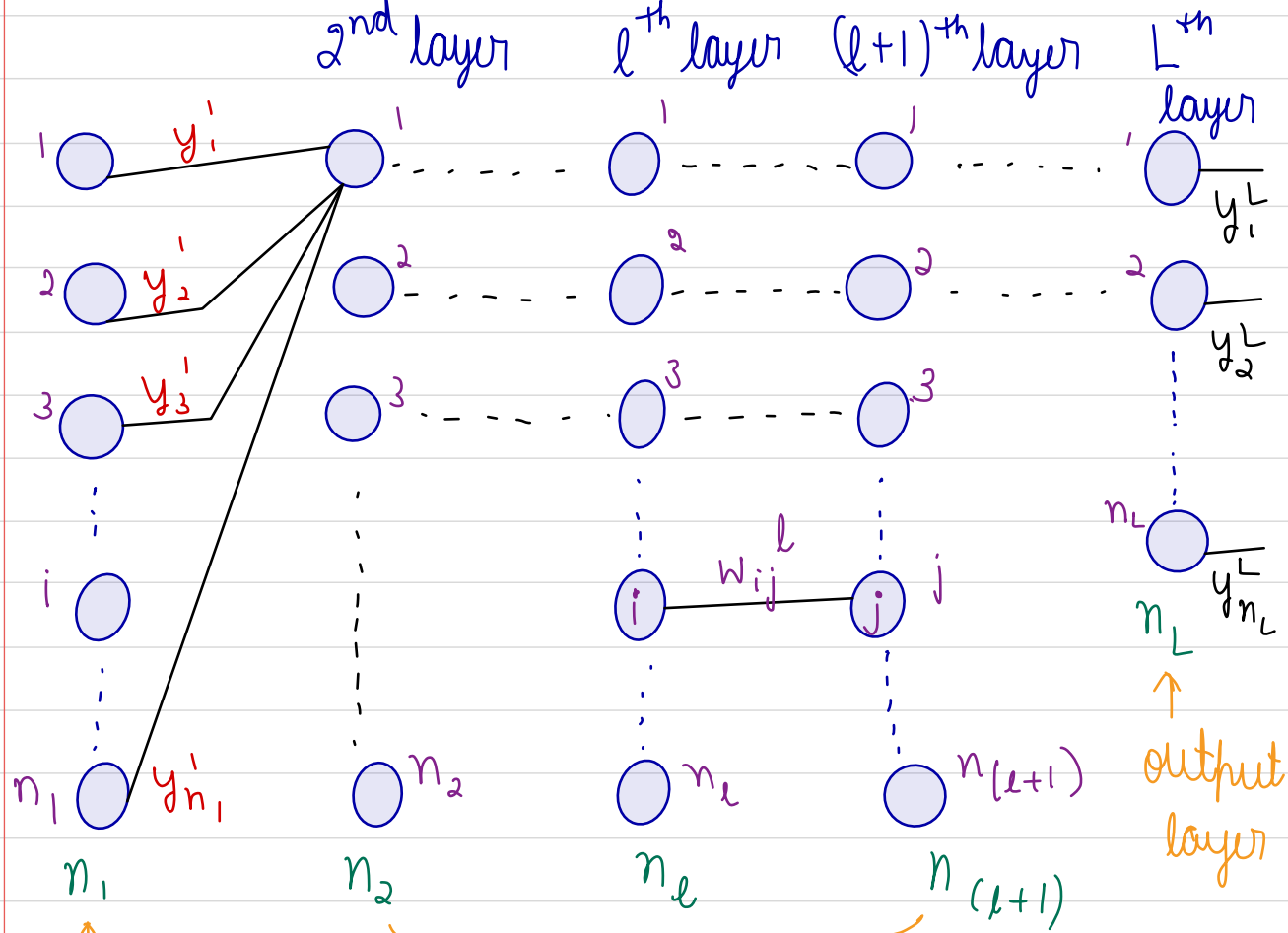
Eg: let $n_1 = 100$ and $n_L = 10$ (output)



input space — hypothesis $f(x, w)$ — Latent space $\mathbb{R}^{100} \to \mathbb{R}^{10}$ — vector value function

Eg for vector valued function is gradient function.

Fed forward Neural network



→ y_i^1 are output at input layer

→ From layer 2, compute their output successively through each layer

$$f(x^i, w): \mathbb{R}^{n_1} \longrightarrow \mathbb{R}^{n_L}$$

i.e $x^i \in \mathbb{R}^{n_1}$ & $d^i \in \mathbb{R}^{n_L}$ where d^i is the i^{th} target vector

Here d^i can be a target vector of :
- i> regression
- ii> two class classification
- iii> Multi class classification

→ For multiclass classification elements of target vectors are binary values.

→ For regression, elements of target vector are real values.

Error: (MSE):

$$\left\| \begin{bmatrix} y_1^L \\ y_2^L \\ \vdots \\ y_{n_L}^L \end{bmatrix} - \begin{bmatrix} d_1^i \\ d_2^i \\ \vdots \\ d_{n_L}^i \end{bmatrix} \right\|^2$$

Squared error loss: $\gamma^L(w, x)$

For the i^{th} sample

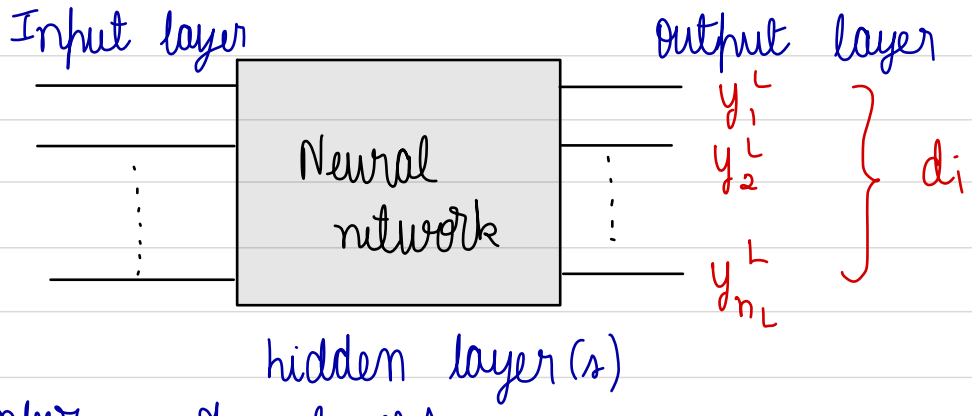
$\| \gamma_i^L(w, x^i) - d^i \|^2$ → Error for one sample

Total error minimization:

$$\min_{W} \frac{1}{2N} \sum_{i=1}^{N} \| Y_i^L(W, x^i) - d^i \|^2$$

Usually no. of output coordinates $\leq$ no of input coordinates

Consider neural network as a black box as shown:



L = number of layers

$y_1^L(x^i, W)$, $y_2^L(x^i, W)$ ..... $y_{n_L}^L(x^i, W)$

Error for one sample, $J_i(W) = \| Y_{n_L}^L - d^i \|^2$

For all samples: $\min_{W} \frac{1}{2N} \sum_{i=1}^{N} \| Y_{n_L}^L - d^i \|^2$

Updation: $W(k+1) = W(k) - \lambda \nabla J(W, k)$

$$W_{ij}^{l}(k+1) = W_{ij}^{l}(k) - \lambda \sum_{s=1}^{N} \frac{\partial J_s(W(k))}{\partial W_{ij}^{l}}$$

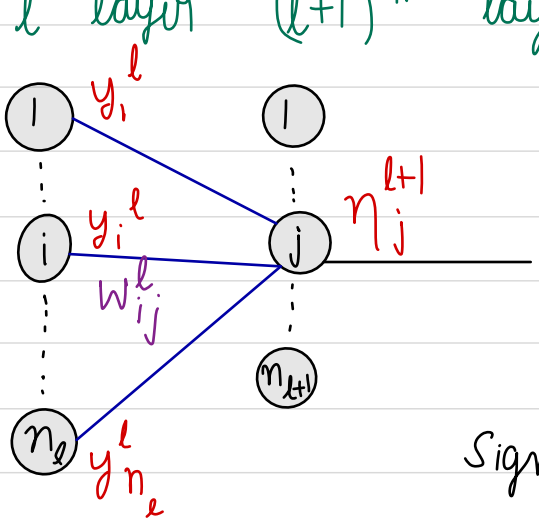
→ Individual weight

$$J = \frac{1}{2} \sum_{j=1}^{n_L} (y_j^L - d_j)^2$$

$\frac{\partial J}{\partial w_{ij}}$ → To find this backpropagation is used.

In a general layered network, the weight w_{ij}^{l} can affect the final output "J" through its effect of η_j^{l+1}, η_j^{l+1} is net input of j^{th} node of $(l+1)^{th}$ layer.

i.e



$$\text{Sigmoid}(\eta_j^{l+1}) = y_j^{l+1}$$

So $$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial \eta_j^{l+1}} \cdot \frac{\partial \eta_j^{l+1}}{\partial w_{ij}^l}$$

where η_{ij}^{l+1} is the net input to sigmoid.

$$\frac{\partial \eta_j^{l+1}}{\partial w_{ij}^l} = \frac{\partial}{\partial w_{ij}} \left(\sum_{s=1}^{n_l} (w_{sj}^l \, y_s^l) \right) = y_i^l$$

$$\therefore \frac{\partial J}{\partial w_{ij}^l} = \frac{\partial J}{\partial \eta_j^{l+1}} \cdot y_i^l$$

Define $$\frac{\partial J}{\partial \eta_j^{l+1}} = \delta_j^{l+1}$$

$$\therefore \frac{\partial J}{\partial w_{ij}^l} = \delta_j^{l+1} \cdot y_i^l$$

By definition $\delta_j^{\,l} = \frac{\partial J}{\partial \eta_j^{\,l}} = \sum_{s=1}^{n_{l+1}} \left(\frac{\partial J}{\partial \eta_s^{\,l+1}} \cdot \frac{\partial \eta_s^{\,l+1}}{\partial \eta_{ij}^{\,l}} \right)$

$$\delta_j^{\,l} = \frac{\partial J}{\partial \eta_j^{\,l}} = \sum_{s=1}^{n_{l+1}} \left(\frac{\partial J}{\partial \eta_s^{\,l+1}} \; \frac{\partial \eta_s^{\,l+1}}{\partial y_i^{\,l}} \; \frac{\partial y_j^{\,l}}{\partial \eta_{ij}} \right)$$

$$\therefore \delta_j^{\,l} = \sum_{s=1}^{n_{l+1}} \left(\delta_s^{\,l+1} \; w_{js}^{\,l} \; f'(\eta_j^{\,l}) \right) \quad \text{where } f' = \frac{\partial y_j^{\,l}}{\partial \eta_{ij}}$$

$$\delta_j^{\,l+1} = \sum_{s=1}^{n_{l+2}} \left(\delta_s^{\,l+2} \; w_{js}^{\,l+1} \; f'(\eta_j^{\,l+1}) \right)$$

$\delta_j^{\,L} \;:\; j = 1, 2, \ldots n_L$

$\delta_j^{\,L-1} \;:\; j = 1, 2, \ldots n_{L-1}$

$\vdots$

$\delta_j^{\,1} \;:\; j = 1, 2, \ldots, n_1$

We have $J = \frac{1}{2} \sum_{j=1}^{n_L} (y_j^{\,L} - d_j)^2$

$$\therefore \delta_j^{\,L} = \frac{\partial J}{\partial \eta_j^{\,L}} = \frac{\partial J}{\partial y_j^{\,L}} \; \frac{\partial y_j^{\,L}}{\partial \eta_j^{\,L}}$$

$$= (y_L^{\,L} - d_j)^j \; f'(\eta_j^{\,L})$$