Linear Classifier: Perceptron

Compiled by Karthikeyan S CED16I015
Guided by
Dr Umarani Jayaraman

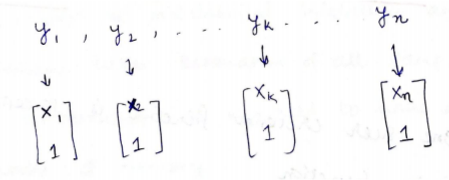
Department of Computer Science and Engineering
Indian Institute of Information Technology Design and Manufacturing
Kancheepuram

April 20, 2022

Introduction

- If the probability density function is not known then we can not estimate or have any parametric form of the probability density function.
- In such cases, we try to estimate the weight vector W and w_0 which separate the two classes if it linearly separable.
- Here W gives the orientation of the line while w_o gives position of the line which separate the two classes.
- With this assumption we try to design the linear classifiers.
- One of the linear classifier that we discuss in this is the **perceptron** and its convergence proof.

Introduction



$$X_{\mathrm{i}} = (x_1, x_2, ..., x_{\mathrm{d}})$$

$$\boxed{\mathrm{y} = \mathrm{d{+}1\ components} \approx \hat{d}}$$

$$\textit{If } a^{\mathrm{t}} y_{\mathrm{i}} > 0 \Rightarrow y_i \in \omega_1$$

$$a^{\mathrm{t}} y_{\mathrm{i}} < 0 \Rightarrow y_i \in \omega_2$$

Uniform criterion function

- For all the samples $a^t y_i > 0$ the weight vector 'a' is correctly classified.
- Otherwise, it is mis-classified and then we should update the weight vector from $a(k)$ to $a(k+1)$.
- We take some criterion function $J(a)$.
- $J(a)$ is minimised if 'a' is a solution vector/solution region.
- One such criterion function is perceptron criterion function

$$J_p(a) = \Sigma(-a^t y) \ \forall \text{ y mis-classified}$$

Perceptron algorithm

The perceptron algorithm is :

$$\boxed{\begin{array}{c} \text{a}(0) = \text{Initial weight vector; arbitrary} \\ \text{a}(\text{k}+1) = \text{a}(\text{k}) + \eta(k)\ \Sigma y\ \forall\ \text{y mis-classified} \end{array}}$$

- $J_{\mathrm{p}}(a)$ can have minimum value which is zero.
- It has a global minimum and that can be obtained using iterative procedure, whenever 'a' is in solution region/solution vector.

Issues in Perceptron algorithm

- We can find that there is a problem in this procedure.
- **The problem is in terms of memory requirement for execution of this algorithm**.
- In real situation, we may have 1000s of such samples which will be mis-classified initially.
- And the algorithm takes summation of all samples which are mis-classified; so we need to have large amount of memory.
- **The solution is instead of considering all the samples together, we can consider sample by sample.**
- As a result, we can have a sequential version of perceprtion algorithm.

Sequential Version of Perceptron algorithm

In y_1, y_2,..., y_k,....,y_n $\Rightarrow$ If y_k is mis-classified, then:

$$\boxed{\begin{array}{c} a(0) = \text{arbitrary} \\ a(k+1) = a(k) + \eta(k)y_k \end{array}}$$

- Memory requirement is much less as compared to previous algorithm.

Sequential Version of Perceptron algorithm

One of the variant of perceptron that is easier to analyse:

- We shall consider **the samples in a sequence** and shall modify the weight vector whenever it is mis-classified a **single sample**.
- $\eta(k)$ - constant $\Rightarrow$ Fixed increment case.
- $\eta(k) = 1$ with no loss in generality.
- Accordingly, the modified perceptron algorithm is as follows

$$\boxed{\begin{array}{c} \mathsf{a(0)} = \text{arbitrary} \\ \mathsf{a(k+1)} = \mathsf{a(k)} + 1.\mathsf{y_k} \end{array}}$$

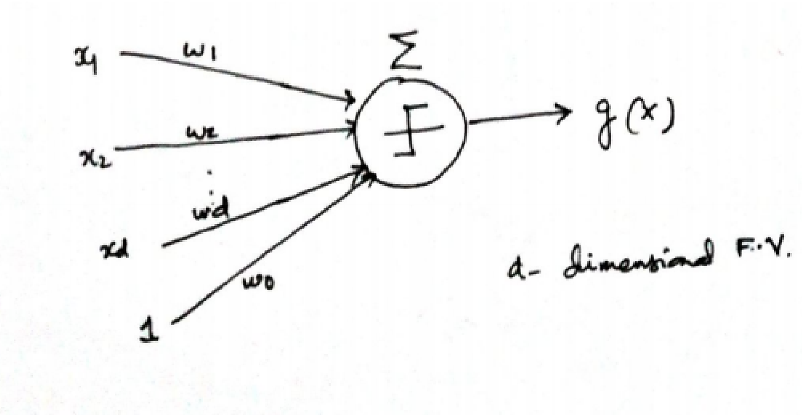
Perceptron algorithm: Sequential Version

ALGORITHM - Fixed-Increment Single-Sample Perceptron

- Initialize a, k $\leftarrow$ 0
- **do** k $\leftarrow$ (k+1) mod n
- If y_k is misclassified by 'a' then a $\leftarrow$ a + y_k
- **Until** all samples are correctly classified
- return a

End ALGORITHM

Two category case



Example: Perceptron learning algorithm

Pattern no	1	2	Class	
x_1	0.5	3.0	x, 1	
x_2	1	3.0	x, 1	
x_3	0.5	2.5	x, 1	ω_1
x_4	1	2.5	x, 1	
x_5	1.5	2.5	x, 1	
x_6	4.5	1	0, 2	
x_7	5	1	0, 2	ω_2
x_8	4.5	0.5	0, 2	
x_9	5.5	0.5	0, 2	

Example: Perceptron learning algorithm

Pattern no	1	2	3
x_1	-0.5	-3.0	-1
x_2	-1	-3.0	-1
x_3	-0.5	-2.5	-1
x_4	-1	-2.5	-1
x_5	-1.5	-2.5	-1
x_6	4.5	1	1
x_7	5	1	1
x_8	4.5	0.5	1
x_9	5.5	0.5	1

ω_2: augment the vector and negate it $\begin{bmatrix} -x_1 \\ -x_2 \\ -1 \end{bmatrix}$

ω_1: augment the vector ~~and negate it~~

Example: Perceptron learning algorithm

$$w_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \text{ and } x_1 = \begin{pmatrix} -0.5 \\ -3.0 \\ -1 \end{pmatrix}$$

here ${w_1}^t x_1 = 0$ so $w_2 = w_1 + x_1$ represented by

$\boxed{a(k+1) = a(k) + \eta(k) y_k \Sigma y}$ for all y misclassified

$$w_2 = w_1 + x_1$$

$$= \begin{pmatrix} -0.5 \\ -3.0 \\ -1 \end{pmatrix}$$

Example: Perceptron learning algorithm

next we consider the patten x_2 : ${w_2}^t x_2$

$$\begin{pmatrix} -0.5 & -3.0 & -1 \end{pmatrix} \begin{pmatrix} -1 \\ -3 \\ -1 \end{pmatrix} = 10.5 > 0$$

x_3, x_4 and x_5 are also properly classified

$$\begin{pmatrix} -0.5 & -3.0 & -1 \end{pmatrix} \begin{pmatrix} -0.5 \\ -2.5 \\ -1 \end{pmatrix} = 8.75 > 0$$

$$\begin{pmatrix} -0.5 & -3.0 & -1 \end{pmatrix} \begin{pmatrix} -1 \\ -2.5 \\ -1 \end{pmatrix} = 9 > 0$$

$$\begin{pmatrix} -0.5 & -3.0 & -1 \end{pmatrix} \begin{pmatrix} -1.5 \\ -2.5 \\ -1 \end{pmatrix} = 9.25 > 0$$

Example: Perceptron learning algorithm

$$\begin{pmatrix} -0.5 & -3.0 & -1 \end{pmatrix} \begin{pmatrix} 4.5 \\ 1 \\ 1 \end{pmatrix} = \text{-6.25} < 0$$

so update weight vector

$w_3 = w{=}_2 + x_6$

$$= \begin{pmatrix} -0.5 \\ -3 \\ -1 \end{pmatrix} + \begin{pmatrix} 4.5 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ -2 \\ 0 \end{pmatrix}$$

note that w_3 classifies patterns x_7, x_8, x_9 and in the next iteration x_1,x_2,x_3 and x_4 correctly.

Example: Perceptron learning algorithm

$$w_3{}^t x_7 = \begin{pmatrix} 4 & -2 & 0 \end{pmatrix} \begin{pmatrix} 5 \\ 1 \\ 1 \end{pmatrix} = 18$$

$$w_3{}^t x_8 = \begin{pmatrix} 4 & -2 & 0 \end{pmatrix} \begin{pmatrix} 4.5 \\ 0.5 \\ 1 \end{pmatrix} = 17$$

$$w_3{}^t x_9 = \begin{pmatrix} 4 & -2 & 0 \end{pmatrix} \begin{pmatrix} 5.5 \\ 0.5 \\ 1 \end{pmatrix} = 21$$

$$w_3{}^t x_1 = \begin{pmatrix} 4 & -2 & 0 \end{pmatrix} \begin{pmatrix} -0.5 \\ -3.0 \\ -1 \end{pmatrix} = 4$$

$$w_3{}^t x_2 = \begin{pmatrix} 4 & -2 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ -3 \\ -1 \end{pmatrix} = 2$$

Example: Perceptron learning algorithm

$$w_3{}^t x_3 = \begin{pmatrix} 4 & -2 & 0 \end{pmatrix} \begin{pmatrix} -0.5 \\ -2.5 \\ -1 \end{pmatrix} = 3$$

$$w_3{}^t x_4 = \begin{pmatrix} 4 & -2 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ -2.5 \\ -1 \end{pmatrix} = 1$$

However x_5 is misclassified by w_3, note that $w_3{}^t x_5$ is -1

$$w_3{}^t x_2 = \begin{pmatrix} 4 & -2 & 0 \end{pmatrix} \begin{pmatrix} -1.5 \\ -2.5 \\ -1 \end{pmatrix} = \text{-1} < 0$$

So, update weight vector $w_4 = w_3 + x_5$

$$w_4 = \begin{pmatrix} 4 \\ -2 \\ 0 \end{pmatrix} + \begin{pmatrix} -1.5 \\ -2.5 \\ -1 \end{pmatrix} = \begin{pmatrix} 2.5 \\ -4.5 \\ -1 \end{pmatrix}$$

Example: Perceptron learning algorithm

w_4 classifies patterns x_6, x_7, x_8, x_9, x_1, x_2, x_3, x_4 and x_5 correctly

$$w_4{}^t x_6 = \begin{pmatrix} 2.5 & -4.5 & -1 \end{pmatrix} \begin{pmatrix} 4.5 \\ 1 \\ 1 \end{pmatrix} = 5.75$$

$$w_4{}^t x_7 = \begin{pmatrix} 2.5 & -4.5 & -1 \end{pmatrix} \begin{pmatrix} 5 \\ 1 \\ 1 \end{pmatrix} = 7$$

$$w_4{}^t x_8 = \begin{pmatrix} 2.5 & -4.5 & -1 \end{pmatrix} \begin{pmatrix} 4.5 \\ 0.5 \\ 1 \end{pmatrix} = 8$$

$$w_4{}^t x_9 = \begin{pmatrix} 2.5 & -4.5 & -1 \end{pmatrix} \begin{pmatrix} 5.5 \\ 0.5 \\ 1 \end{pmatrix} = 10.5$$

Example: Perceptron learning algorithm

$$w_4{}^t x_1 = \begin{pmatrix} 2.5 & -4.5 & -1 \end{pmatrix} \begin{pmatrix} -0.5 \\ -3.0 \\ -1 \end{pmatrix} = 13.25$$

$$w_4{}^t x_2 = \begin{pmatrix} 2.5 & -4.5 & -1 \end{pmatrix} \begin{pmatrix} -1 \\ -3 \\ -1 \end{pmatrix} = 11.5$$

$$w_4{}^t x_3 = \begin{pmatrix} 2.5 & -4.5 & -1 \end{pmatrix} \begin{pmatrix} -0.5 \\ -2.5 \\ -1 \end{pmatrix} = 11$$

$$w_4{}^t x_4 = \begin{pmatrix} 2.5 & -4.5 & -1 \end{pmatrix} \begin{pmatrix} -1 \\ -2.5 \\ -1 \end{pmatrix} = 9.75$$

$$w_4{}^t x_5 = \begin{pmatrix} 2.5 & -4.5 & -1 \end{pmatrix} \begin{pmatrix} -1.5 \\ -2.5 \\ -1 \end{pmatrix} = 8.5$$

Example: Perceptron learning algorithm

- So w_4 (or) a_4 is the desired vector 'a'
- In other words $2.5x_1$ $-4.5x_2$ $-1 = 0$ is the equation of the decision boundary.
- Equivalently, the line separating the two classes is $5x_1$ $-9x_2$$-2 = 0$
- w_1=5,w_2=-9,w_0=-2

Recap: Convergence of Perceptron Algorithm

Perceptron Criterion:
{X} -> {y}

$$\begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_d \end{bmatrix} \text{->} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_d \\ 1 \end{bmatrix}$$

If $a^t y > 0$ then $y \in \omega_1$
If $a^t y < 0$ then $y \in \omega_2$

Recap:Uniform Criterion Function

- For all the samples $a^t y > 0$ the weight vector a is correctly classified.
- otherwise it is misclassified.
- Then we should update the weight vector a(k) to a(k+1) we are interested to find the weight vector 'a'.
- $J(a)$ has to be minimum.

a(0) - arbitrary
a(k+1) = a(k) - $\eta(k)\nabla J(a(k))$

Criterion:
$\mathrm{J_p(a)} = \Sigma(-a^t y)\ \forall y - misclassified$
$a(0) - arbitrary$
$a(k+1) = a(k) + \eta(k)\Sigma y\ \forall\ y - misclassified$

Recap: Sequential Version of Perceptron Algorithm

y^1, y^2, y^3,...,y^k,...,y^n -> k^{th} sample misclassified

a(0) - arbitrary
a(k+1) = a(k) + ηy^k

Perceptron Algorithm: Convergence Proof

- To demonstrate that the above sequential algorithm converge lets consider the two dimensional case:

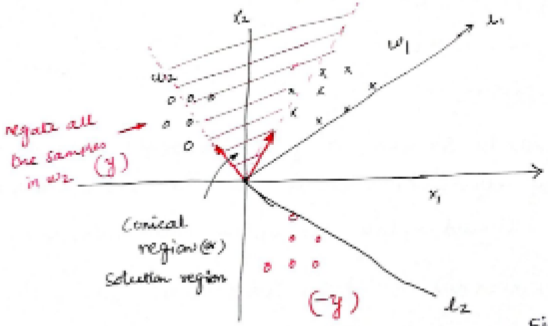


Fig. 1

Perceptron Algorithm: Convergence Proof

- Weight vector 'a' is orthogonal to the decision surface.
- In 2-D it is nothing but a line.
- What are straight lines which actually separates these two classes?
- We could have some limiting cases, two lines l_1 and l_2.
- Any line that lies in between these two limiting lines l_1 and l_2 which properly separates these two classes without error.

Perceptron Algorithm: Convergence Proof

- Now the weight vectors are orthogonal to the decision boundary.
- Any weight vector 'a' lies within the conical region is solving our purpose.
- The conical region is the solution region.
- Our weight vectors should lie within this solution region.
- When the algorithm converges the weight vectors should lie within our solution region.

Perceptron Convergence Proof: Algorithm Illustration

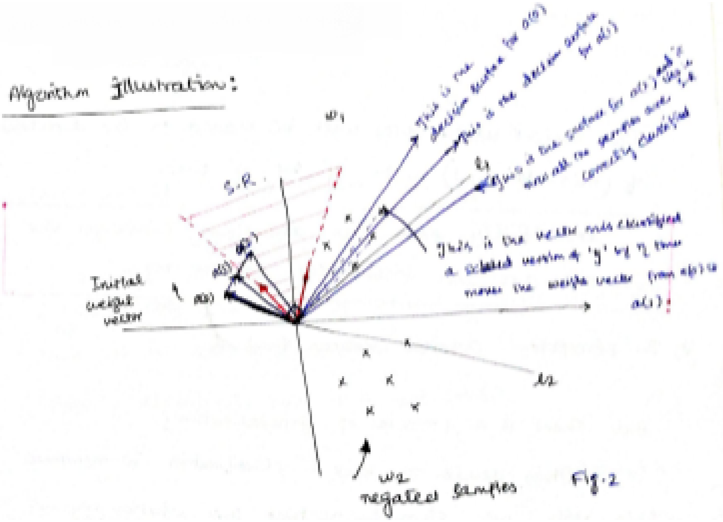


Fig.2

Perceptron Convergence Proof: Algorithm Illustration

- The initial weight vector $a(0)$ misclassifies the 3 samples in ω_1.
- The decision surface corresponding to the weight vectors $a(0)$ which is drawn in blue line.
- According to the algorithm:

$$a(k) = a(k-1) + \eta \Sigma y \forall y - misclassified$$

$$a(k) = a(k-1) + \eta y$$

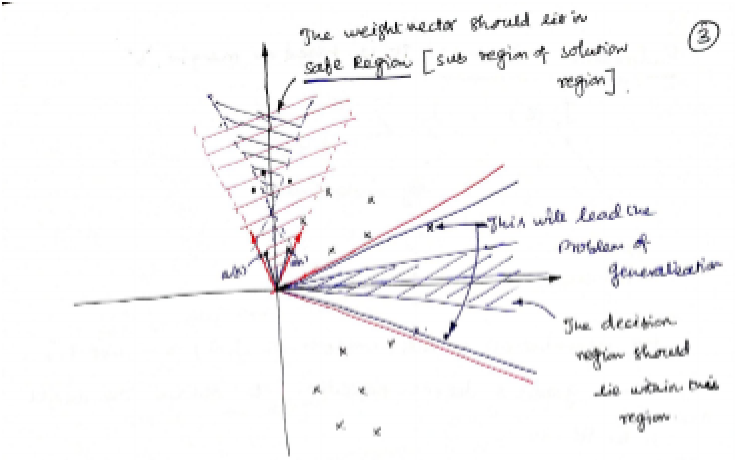
- This vector $'y'$ is scaled by a factor η in the direction of 'y' and added with the previous weight vector $a(k-1)$

Perceptron Convergence Proof: Algorithm Illustration

- The weight vector $a(0)$ will be moved in the direction of misclassified vector $'y'$ by η times.
- And finally when the algorithm converges the weight vectors lie within the solution region.
- This is ensured by the perceptron criterion.

But there is a problem of generalization.

- This leads to risk in classification.
- To minimise this risk, we should restrict the solution region some where as the safe region (sub space of solution region).
- That means we should ensure the weight vector 'a' should lie in safe region (refer Fig. 3).


③
The weight vector should lie in
safe Region [sub region of solution region].
This will lead the problem of generalization
The decision region should lie within this region

Perceptron Convergence Proof: Algorithm Illustration

- In order to ensure the weight vector 'a' should lie in safe region, It should be $>$ some margin b
- This can be ensured by the rule $a^t y > b$, for some positive constant b.
- We would say now, any y which satisfies $a^t y > b$ then it is safely classified.
- If it is > 0 then it is properly classified but it is not in the safe region.
- With this, we can ensure that the weight vector should lie on the safe region.
- The perceptron criterion is not only the criteria function to design a linear classifier.
- One of the criteria function can be defined based on the margin (b); It is called as relaxation criterion.

Relaxation Criterion

- It is based on margin b

$$J_r(a) = \frac{1}{2}\Sigma\frac{(a^t y - b)^2}{||y||^2}\forall y - misclassified$$

- For minimization of this criteria function $J_r(a)$ we use the same gradient descent procedure to obtain the weight vector 'a'.

$$\triangledown J_r(a) = \Sigma\frac{(a^t y - b)^2}{||y||^2}$$

$$= \Sigma\frac{(a^t y - b)}{||y||^2}.y\forall y - misclassified$$

$\boxed{\text{a(0)} = \text{arbitrary}}$ $\boxed{\text{a(k+1)} = \text{a(k)} + \eta\Sigma\frac{b-a^t y}{||y||^2}.y\forall y - misclassified}$

Sequential version of Relaxation Criterion

$$a(0) = arbitrary$$

$$a(k+1) = a(k) + \eta \frac{b - a^{t}(k) y^{k}}{||y^{k}||^{2}} . y^{k}$$

- Here, the samples are considered one after another.
- The moment, when we find the vector 'y' is misclassified, we should update the weight vector.
- It can be noted that whether we use perceptron criteria or relaxation criteria, in both cases, the convergence is guaranteed if the classes are linearly separable.
- Otherwise, the algorithm can never converge.
- We can make use of these algorithms only if we know for sure the classes are linearly separable.
- However, if we are not sure (or) do not know if the classes are linearly separable or not, still we can design linear classifier with **minimum error.**

II. Minimum Squared Error - For Non Separable Case

- The criterion function thus so far, have focused their attention on the mis-classified samples.
- Now, we shall consider a criterion function that involves all of the samples.
- Previously, the decision rule was $a^t y > 0$.
- Now, we shall try to make $a^t y > b$.
- The decision surface is $a^t y = b$, where b is some positive constant.
- We should get a solution to this equation $a^t y = b$.
- The solution of this equation can be obtained by this minimum squared error procedure to be more generalization:

$$a^t y_i = b_i \text{ : for every sample } y_i$$

- We can have different margins for generalization.

Minimum Squared Error - For Non Separable Case

- For every i^{th} sample, we have such an equation.
- So for 'n' number of samples, 'n' number of equations.
- So, we have 'n' number of simultaneous equations and solve this number of simultaneous equations.
- This can be simplified by introducing matrix.

Minimum Squared Error - For Non Separable Case

- In matrix form:

$$\begin{bmatrix} Y_{10} & Y_{11} & Y_{12} & \dots & Y_{1d} \\ Y_{20} & Y_{21} & Y_{22} & \dots & Y_{2d} \\ . & & & & \\ . & & & & \\ . & & & & \\ Y_{n0} & Y_{n1} & Y_{n2} & \dots & Y_{nd} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ . \\ . \\ a_d \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ . \\ b_n \end{bmatrix}$$

- In compact form:
- $Ya = b$
- Find the weight vector 'a' satisfying the above matrix
- $a = Y^{-1} b$

Minimum Squared Error - For Non Separable Case

- But, the problem is this 'y' is not a square matrix; it is a rectangular matrix.
- No. of rows = no. of samples
- No. of columns = d+1 (or) $\hat{d}$; usually with more rows that columns.
- In this case, the vector 'a' is over determined.
- So; we cant get an exact solution for this vector 'a'.
- To get the solution for this vector 'a' we can define an error vector:

$$e = Ya - b$$

Our aim is to get a solution for 'a' that minimises this error:

- Y is training sample and 'b' is margin; so both 'Y' and 'b' is known
- 'a' is unknown: try to get solution for 'a' which will minimize this error.

Sum of Squared Error Criterion

- Let's define a criterion function (*i.e*) Sum of Squared Error criterion
- $J_s(a) = ||Ya - b||^2$
- which is nothing but
- $J_s(a) = \Sigma(a^t y_i - b_i)^2$
- This can be solved by gradient descent approach; we can start initial weight vector 'a' and go on updating it.
- $\nabla J_s(a) = \Sigma 2(a^t y_i - b_i).y_i$
- $\nabla J_s(a) = 2Y^t(Ya - b) = 0$

Closed form solution

- $\nabla J_s(a) = 2Y^t(Ya - b) = 0$
- $2Y^t(Ya - b) = 0$
- $2Y^tYa - 2Y^tb = 0$
- $Y^tYa = Y^tb$
- $a = (Y^tY)^{-1}Y^tb$
- where Y is a rectangular matrix of dimension nXd, but Y^tY will be a square matrix of dXd and quite often this matrix is non singular.
- $a = Y^+b$ where Y^+ is the $(Y^tY)^{-1}Y^t$ **pseudo inverse** of Y.

Closed form solution

Note:

- If Y is square and non singular, the pseudo inverse coincides with the regular inverse.
- $Y^+ Y = I$
- But, $YY^+ \neq I$
- However, MSE solution always exists and that $a = Y^+ b$ is an MSE solution to $Ya = b$.
- The MSE solution depends on the margin vector 'b'
- Different choices for 'b' give the solution different properties.

Problem of Generalization

- Generalization is a term used to describe a models ability to react to new data.
- That is, after being trained on a training set, a model can digest new data and make accurate predictions.

THANK YOU