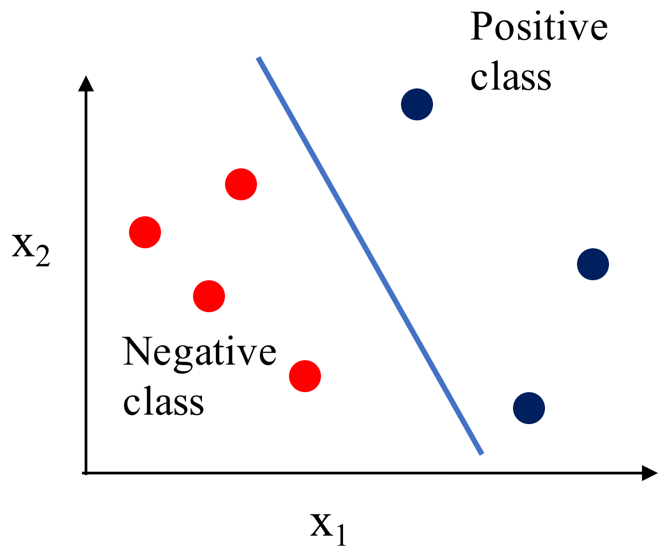
Machine Learning – BCSE209L
Perceptron

Dr. R. Bhargavi
Professor
SCOPE
VIT University

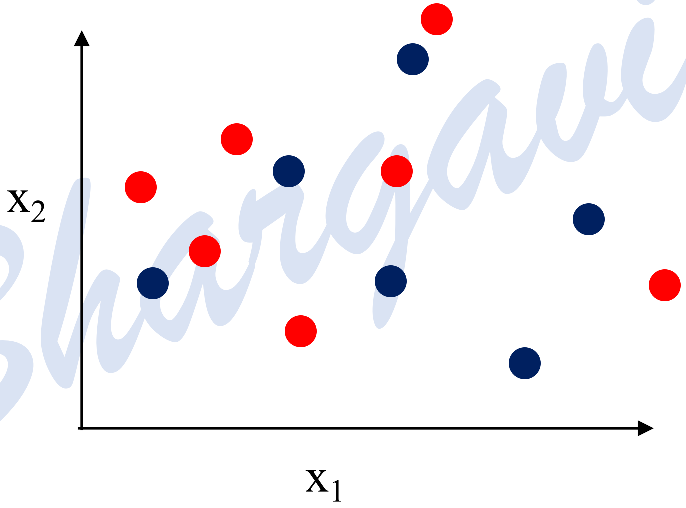
Linearly Separable Data

- Linearly separable data refers to a set of data points that can be perfectly separated into different classes using a linear decision boundary.
- Consider the binary classification problem of loan approval where the task is to develop a model that can predict whether a new applicant's request for loan can be approved or not based on existing credit amount (x_1), and salary (x_2).
- Let y be the loan approval status. Here $y \in \{0,1\}$ where 1 represents risky applicant and hence do not approve the loan (positive class) and, 0 represents safe applicant and hence approve loan (negative class).



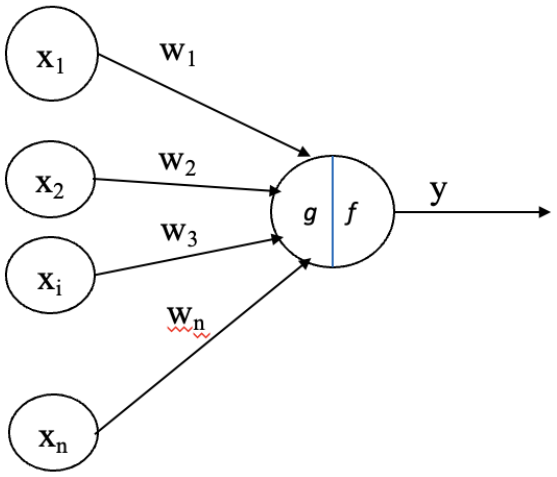
Linearly Non-Separable Data

- Linearly non-separable data refers to a set of data points that can not be perfectly separated into different classes using a linear decision boundary.



Perceptron

- ***Classical perceptron*** model and learning algorithm introduced by Rosenblatt in 1958.
- Based on MCP(McCullouch Pitts) neuron model.
- Inputs have weights.
- Inputs can be real values.
- Perceptron learning: Supervised in nature and uses the error between the desired and predicted output to adjust the weight.
- Linear model
- Data must be linearly separable.



Example

- How a bank takes a decision to approve or reject loan of an applicant?
- Information available (**Input**): Applicant Name, Age, Gender, Salary, Years in job, Existing loan, etc.
- Decision to be taken (**Output**) : safe or risk applicant for loan approval
- Information available to solve the problem (**Dataset**): Previous loan applicants details and credit approval status
- Problem formulation: To come up with a formula/model that can determine worthiness of the applicant to approve or reject the loan.
- Unknown Target function f : X -> Y (Ideal credit approval formula)

Hypothesis Function

- Suppose $(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2)$, ----- $(\mathbf{x_n}, y_n)$ be the customer data.
- Here each $\mathbf{x_i}$ is a m-dimensional vector $(x_1, x_2, x_3, ---, x_m)^T$ representing m attributes/features of a customer.
- y_i is the loan approval status. $y_i \in \{+1, -1\}$
- Approve the loan if $\sum_{i=1}^{m} w_i x_i \geqslant Threshold$, Deny otherwise.
- Value of w_i depends on the importance of the attribute.
- Hypothesis function h ∈ H is

$$h(x) = sign\left(\sum_{i=1}^{m} w_i x_i - Threshold\right)$$

Hypothesis Function (cont…)

- Different values of **w** (i.e w_1, w_2, ---, w_n) and Threshold results in different hypotheses.
- Rewriting h(x) by replacing –Threshold with w_0 and introducing an artificial or dummy variable $x_0 = 1$, we get

$$h(x) = sign(\sum_{i=0}^{m} w_i x_i) \text{ or } sign(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

Mathematical Model

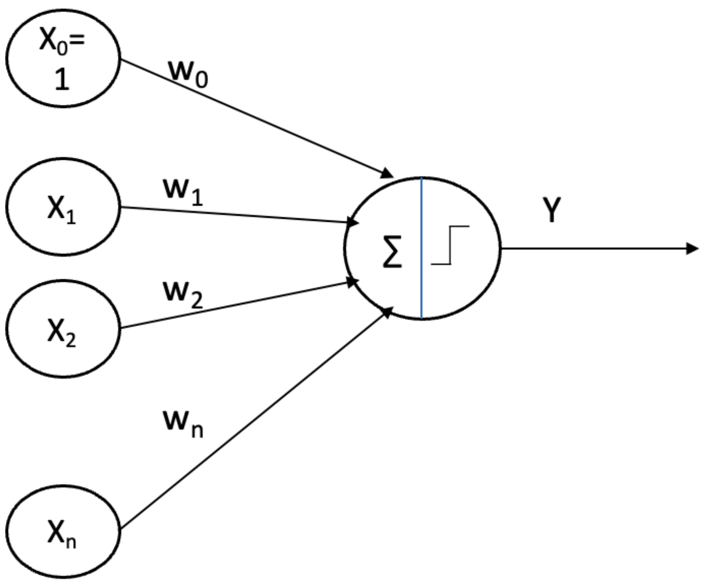
- $g(\boldsymbol{x}) = \sum_{i=1}^{n} w_i x_i$ (Here n is the number of features)
- $y = f(g(x)) = 1 \; if \; g(x) \geq \theta$
  $\qquad\qquad\; = 0 \; if \; g(x) < \theta$

Rewriting

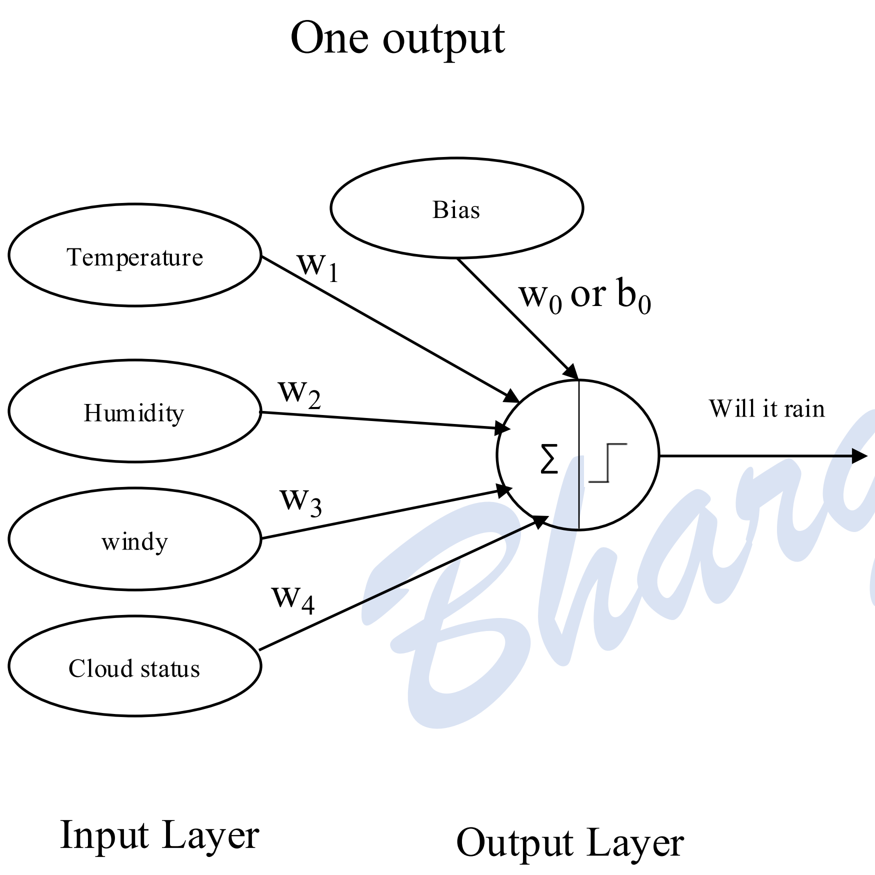
- $y = f(g(x)) = 1 \; if \; g(x) - \theta \geq 0$
  $\qquad\qquad\; = 0 \; if \; g(x) - \theta < 0$

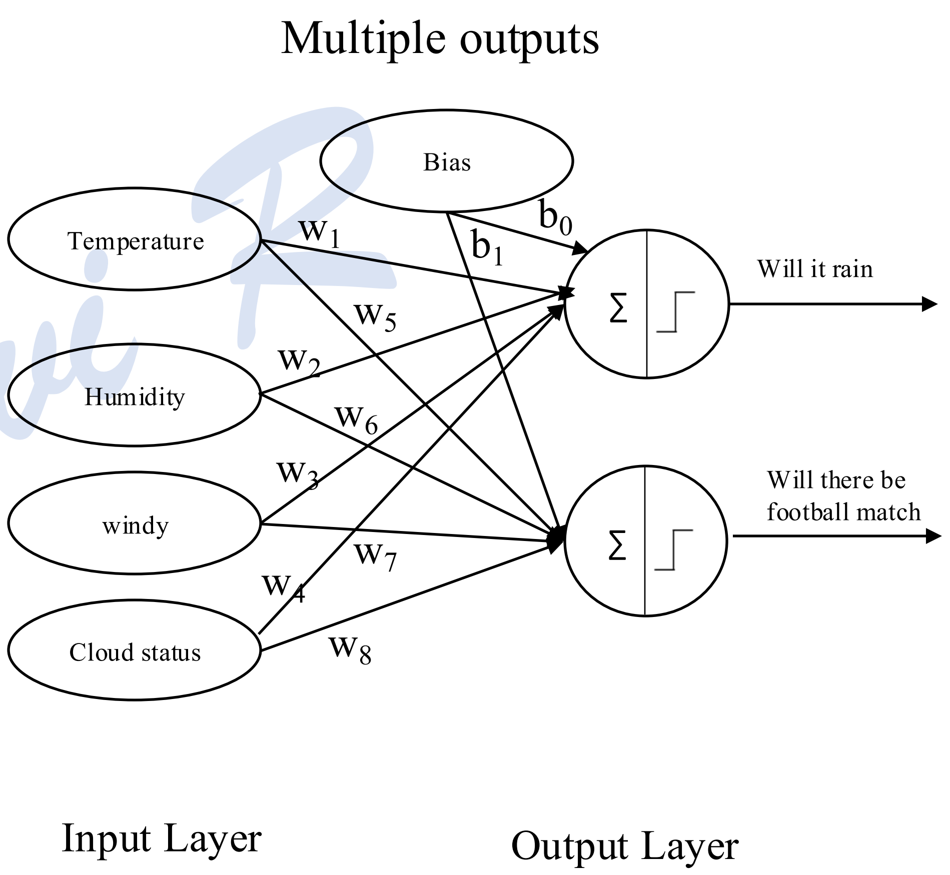
Replacing $-\theta$ with w_0 (called as Bias) and introducing a dummy input x_0

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

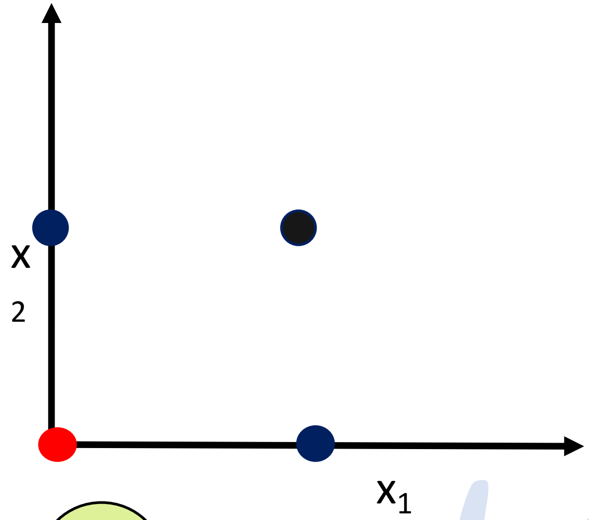


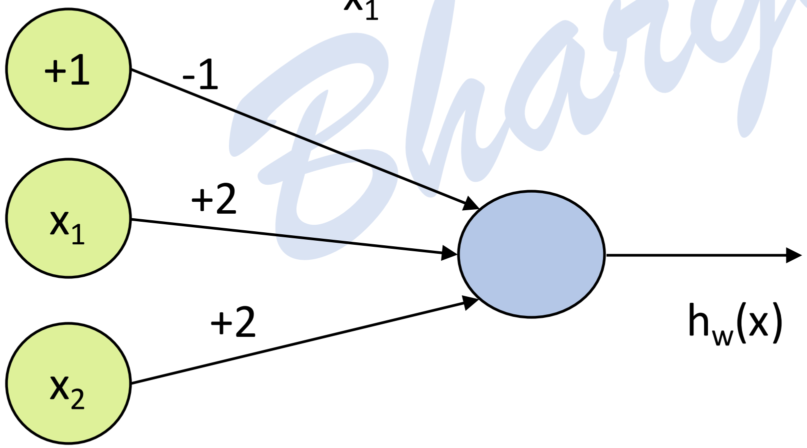
Single Layer Neural Network Architecture





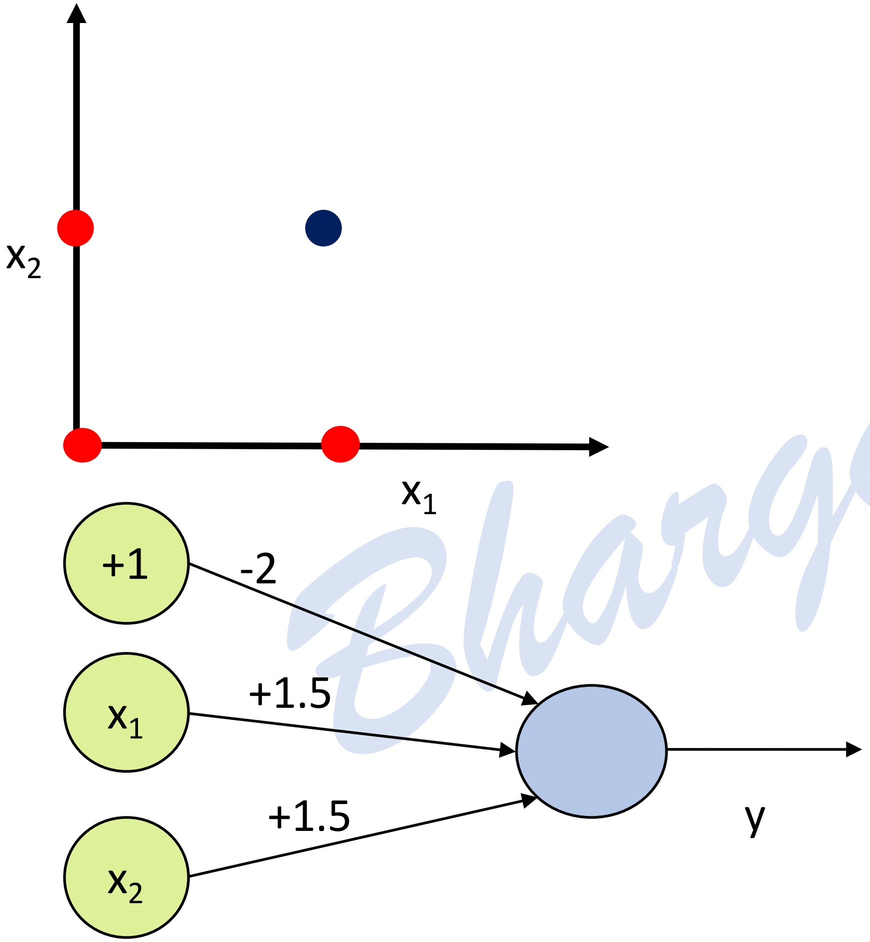
Example – Linear Function (OR)





x_1	x_2	y
0	0	f(-1) = 0
0	1	f(1) =1
1	0	f(1) = 1
1	1	f(3) = 1

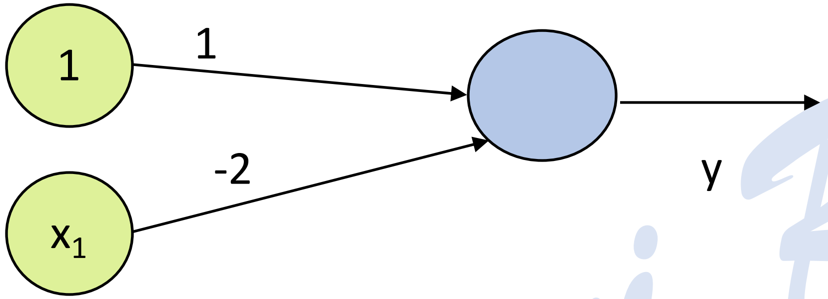
Example – Linear Function (AND)



x_1	x_2	y
0	0	f(-2) = 0
0	1	f(-0.5) = 0
1	0	f(-0.5) = 0
1	1	f(1) = 1

Examples -Linear Functions (cont..)

NOT



x_1	y
0	f(1) =1
1	f(-1) = 0

Perceptron Learning Algorithm

- Substitute each of the training data set $(\boldsymbol{x}_1, y_1)$, …. $(\boldsymbol{x}_n, y_n)$ in the hypothesis,

$$h(x) = sign(\sum_{i=0}^{m} w_i x_i) \text{ or } sign(\boldsymbol{w}^T \boldsymbol{x})$$

- Determine w_i which can classify all the training dataset correctly.

Initialize random values for all w_i

For each $(\boldsymbol{x}_i, y_i)$

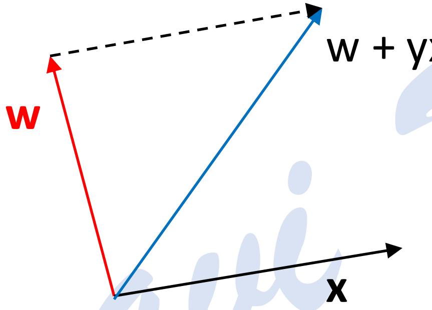
if $sign(\boldsymbol{W}^T \boldsymbol{x}_i) \mathrel{!=} y_i$ *then*

$\boxed{\boldsymbol{w_i} = \boldsymbol{w_i} + \Delta \boldsymbol{w}_i}$ → ***Learning rule***

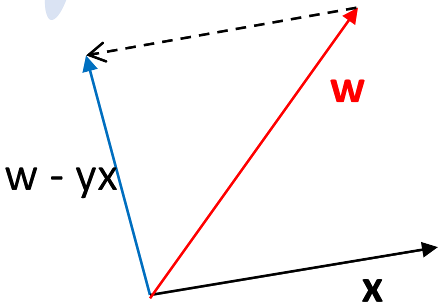
Where $\Delta w_i = \eta((y_i - h_w(\boldsymbol{x_i}))x_i)$

PLA (cont…)

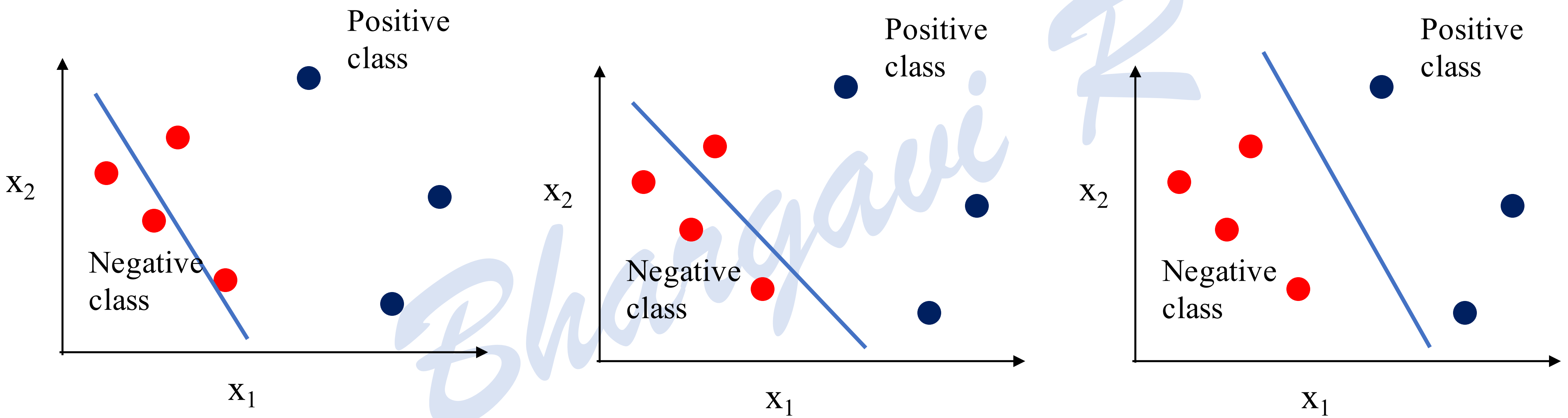
- Misclassification and weight updation when y = +1 & y_{pred} = -1



- Misclassification and weight updation when y = -1 y_{pred} = +1



Perceptron Learning (cont…)



Perceptron – Numerical Example

- Consider a hypothetical Car door alarm scenario. The alarm is raised when one of the doors is opened and, when all the doors are closed then no alarm is raised. Let's see how a perceptron model learns this scenario. We will use alarm signalling, and door open status with 1 and the door closed status with 0. No alarm is indicated by -1. Here the inputs are the doors status, and the output is the alarm status. Also, assume that the car has just two doors.
- The alarm status $y \in \{-1, 1\}$: 1 – Alarm raised

  : -1 – No alarm
- The door status $x \in \{0, 1\}$: 1 - Open

  : 0 - Closed

Example (cont…)

Door1 (x_1)	Door2 (x_2)	Alarm (y)
0	0	-1
0	1	1
1	0	1
1	1	1

- Let the Learning rate (η) = 1
- Initialize w values i.e $\boldsymbol{w} = [w_0, w_1, w_2]^T$ to $[0,0,0]^T$
- Now, Compute $y_{pred} = sign(\sum_{i=0}^{m} w_i x_i)$ for each input.
- $y_{pred} = \begin{cases} +1 & if \sum_{i=0}^{m} w_i x_i \geq 0 \\ -1 & if \sum_{i=0}^{m} w_i x_i < 0 \end{cases}$
- If (y_{pred} != y) then update **w** as $\boldsymbol{w}_{new} = \boldsymbol{w}_{previous} + \Delta \boldsymbol{w}_i$
- Where the update factor $\Delta w_i = \eta(y_i - y_{pred})x_i$

Example (cont…)

- $\Delta \boldsymbol{w}_i = \eta(y_i - y_{pred})\mathbf{x}_i$
- Epoch 1 – Initial weights $\mathbf{w} = [w_0, w_1, w_2]^T$ to $[0,0,0]^T$

x_0	x_1	x_2	$\boldsymbol{w}\,\boldsymbol{x}_i$	y_{pred}	y	Δw_0	Δw_1	Δw_2	w_0	w_1	w_2
1	0	0	0	1	-1	-2	0	0	-2	0	0
1	0	1	-2	-1	1	2	0	2	0	0	2
1	1	0	0	1	1	0	0	0	0	0	2
1	1	1	2	1	1	0	0	0	0	0	2

Example (cont…)

- Epoch 2 - $\mathbf{w} = [w_0, w_1, w_2]^T$ to $[0,0,2]^T$

x_0	x_1	x_2	$\boldsymbol{w}\boldsymbol{x_i}$	y_{pred}	y	Δw_0	Δw_1	Δw_2	w_0	w_1	w_2
1	0	0	0	1	-1	-2	0	0	-2	0	2
1	0	1	0	1	1	0	0	0	-2	0	2
1	1	0	-2	-1	1	2	2	0	0	2	2
1	1	1	4	1	1	0	0	0	0	2	2

Example (cont…)

- Epoch 3 - $\mathbf{w} = [w_0, w_1, w_2]^T$ to $[0,2,2]^T$

x_0	x_1	x_2	$\boldsymbol{w}\boldsymbol{x_i}$	y_{pred}	y	Δw_0	Δw_1	Δw_2	w_0	w_1	w_2
1	0	0	0	1	-1	-2	0	0	-2	2	2
1	0	1	0	1	1	0	0	0	-2	2	2
1	1	0	0	1	1	0	0	0	-2	2	2
1	1	1	2	1	1	0	0	0	-2	2	2

Example (cont…)

- Epoch 4 - $\mathbf{w} = [w_0, w_1, w_2]^T$ to $[-2,2,2]^T$

x_0	x_1	x_2	$\boldsymbol{w x_i}$	y_{pred}	y	Δw_0	Δw_1	Δw_2	w_0	w_1	w_2
1	0	0	-2	-1	-1	0	0	0	-2	2	2
1	0	1	0	1	1	0	0	0	-2	2	2
1	1	0	0	1	1	0	0	0	-2	2	2
1	1	1	2	1	1	0	0	0	-2	2	2