CS-671: Deep Learning and its Applications

Lecture: 02 (a)

Basics of Machine Learning: Linear Classification (I)

Aditya Nigam, Assistant Professor
School of Computing and Electrical Engineering (SCEE)
Indian Institute of Technology, Mandi
http://faculty.iitmandi.ac.in/ãditya/ aditya@iitmandi.ac.in

Presentation for CS-671@IIT Mandi (21 February, 2019)
*Reference Book: Pattern Classification
David G. Stork, Peter E. Hart, and Richard O. Duda

February - May, 2019

Perceptron Algorithm: It is a discriminative technique in which
(*) we try to learn the discriminant fn directly. we donot attempt to learn the parameter distribution instead we will learn the boundary explicity. $\Longrightarrow$ Our Classifier will be using discriminative features to learn discriminant fn. itself.

(*) In other non-discriminant techniques (such as Bayes D. fn) we learn the data distribution parameter and boundary was the result of that.

(*) Now we learn the boundary b/w 2 classes directly and hence we need not to assume that data is following certain distribution. Rather we assume data is linearly seperable or not.

Initially we assume linear discriminant fn. and perform LDA, as non-linear fn's are very difficult to learn and require more complex parameterization.

$$f(x) = w^T x + w_0 = 0$$

w → weight vector

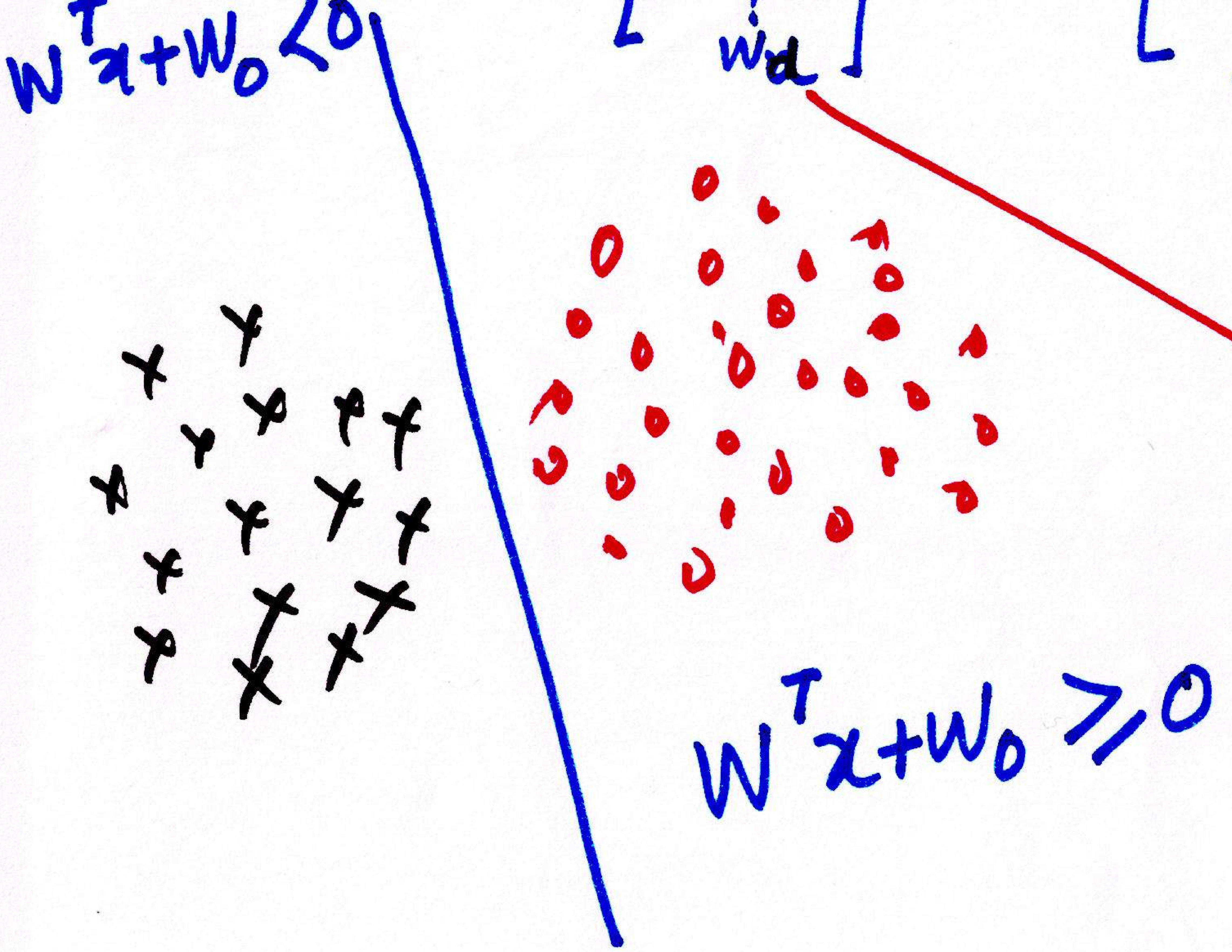
x → data point [Column vector]

w_0 → bias

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_d \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_d \end{bmatrix}$$

$$\Downarrow$$

$$\boxed{w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots w_0 = 0}$$



$$a = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} \qquad z = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_d \end{bmatrix}$$

$$f(x) = \bar{a}^T z$$ [Augumented Representation]

If data is linearly separable then in 2D space a line 3
Can be seen as a linear fn $f(x) = w_1 x_1 + w_2 x_2 + w_0$ that Can seperate two classes, and we are looking to learn such functions and its parameters. Hence basically we are looking to learn : $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ and $[w_0]$ weights/coefficient of the straight line.

vector → $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

scalar → $[w_0]$

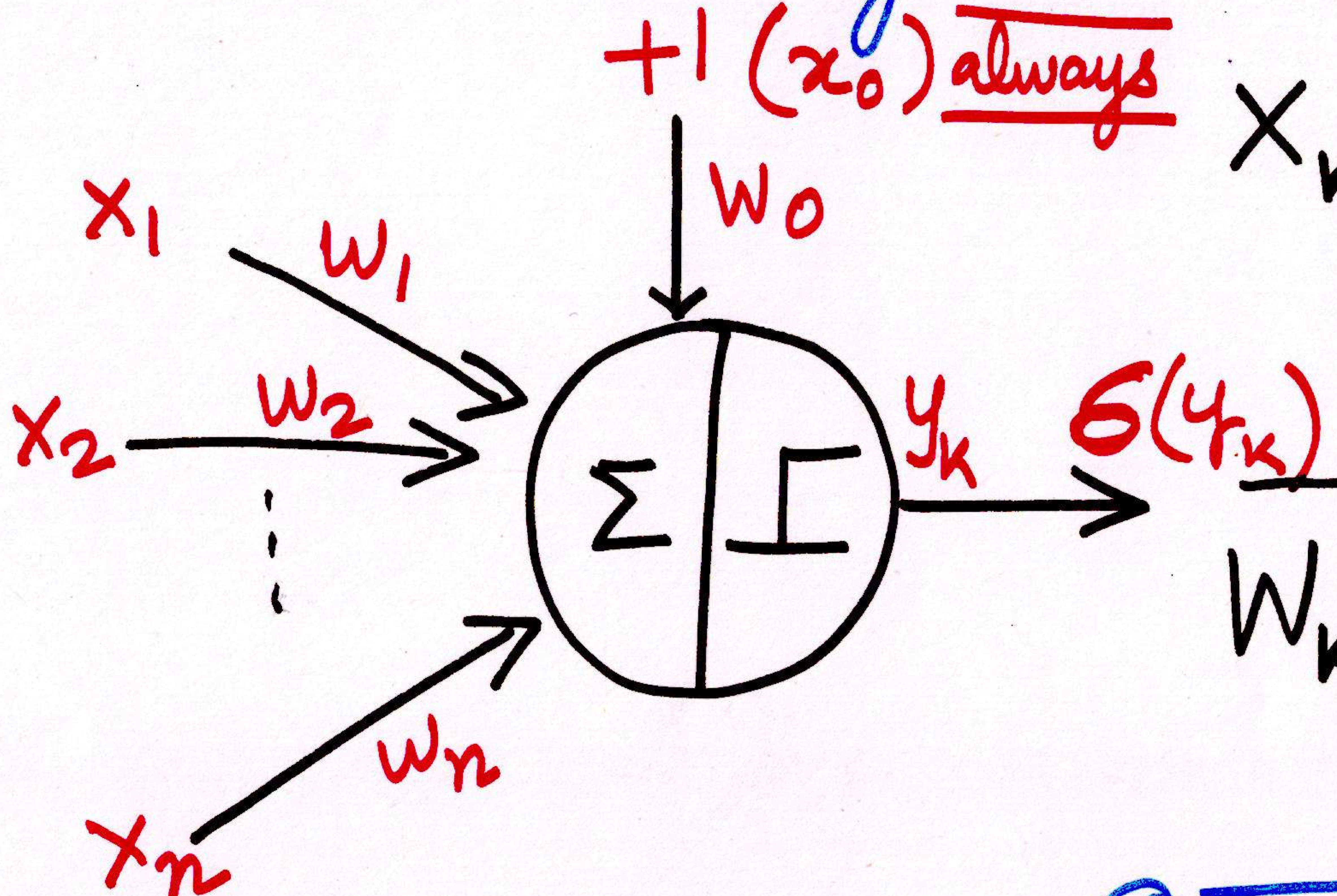
$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

$$\underline{y = mx + c}$$

$$x_2 = \underbrace{-\frac{w_1}{w_2}}_{m\,[\text{slope}]} x_1 \underbrace{- \frac{w_0}{w_2}}_{c\,[\text{Intercept}]}$$

TLN (Threshold logic neuron):

Linear neuron whose O/P is directed into a unit step or signum fn. It is adaptive as its weights are changing according to some fixed rule. **Widrow** called it as adaptive linear element (Adaline) ← Typically they use perceptron or LMS algorithm.



$$X_k = (x_0, x_1^k, x_2^k, --- x_n^k)^T, \quad X_k \in \mathbb{R}^{n+1}$$

$$W_k = (w_0^k, w_1^k, w_2^k --- w_n^k)^T, \quad W_k \in \mathbb{R}^{n+1}$$

k ≡ iteration number as these values are time dependent.

X_k: Pattern presented at k^{th} iteration and W_k is the neuronal weight vector.

The neuronal activation $y_k = X_k^T W_k \Longrightarrow$ Determins whether neuron fires -1 or not -0

Using this logic

Binary threshold neuronal signal fn.

$$\delta(y_k) = \begin{cases} 1 & y_k > 0 \\ 0 & y_k < 0 \end{cases}$$

($y_k = 0$; assumed as misclassification and weights are designed to avoid it.)

So, we have C_0 & C_1 (two classes) with $\mathcal{X}_0$ $\mathcal{X}_1$ = set of patterns of these classes. Hence Traing set $\Rightarrow \mathcal{X} = \mathcal{X}_0 \cup \mathcal{X}_1$

We are looking to compute weight vector W_s s.t

$\forall\ X_k \in \mathcal{X}_1$, $\delta(y_k) = 1$ and for all $X_k \in \mathcal{X}_0$, $\delta(y_k) = 0$

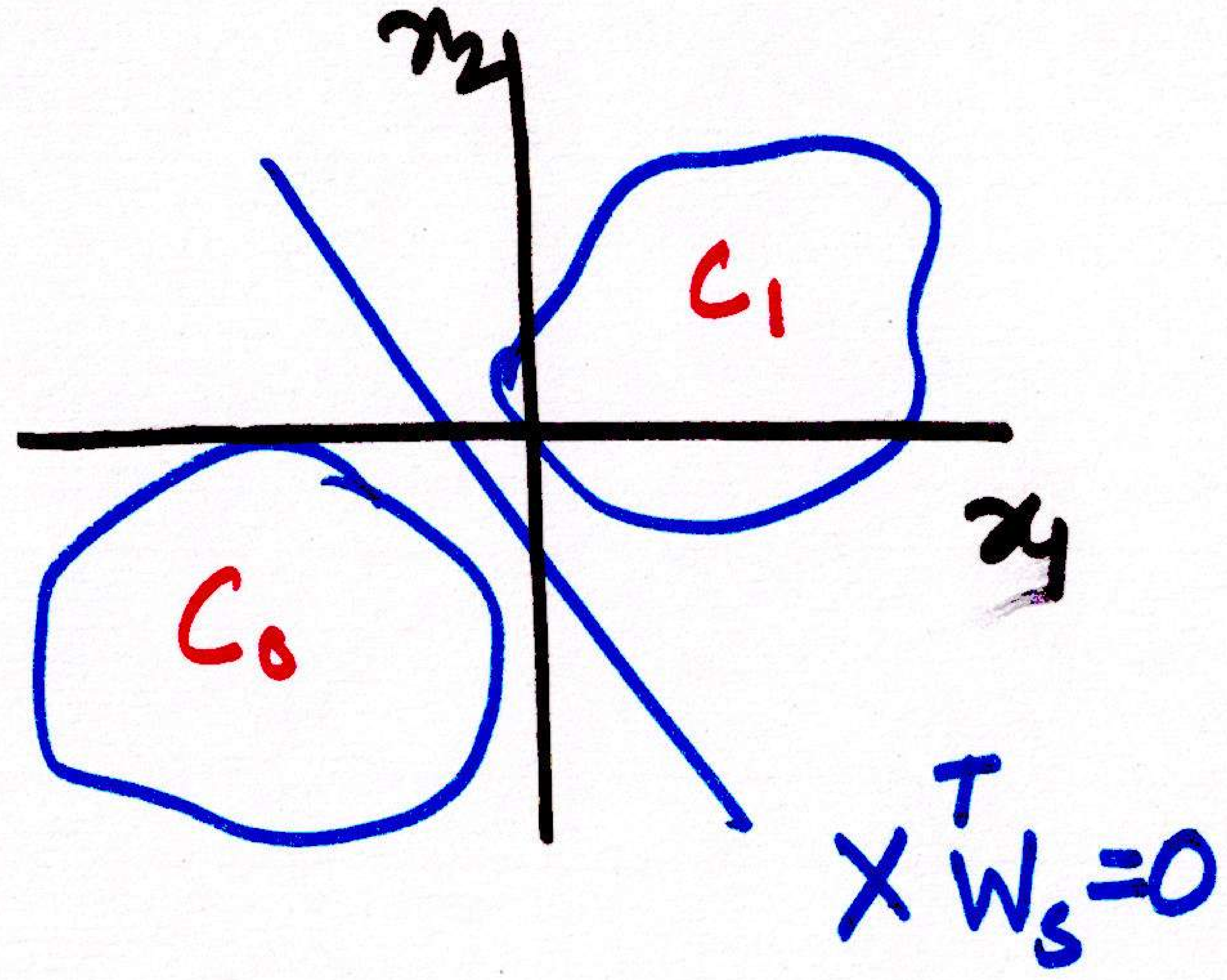
$\Downarrow$ Innerproduct is +ve

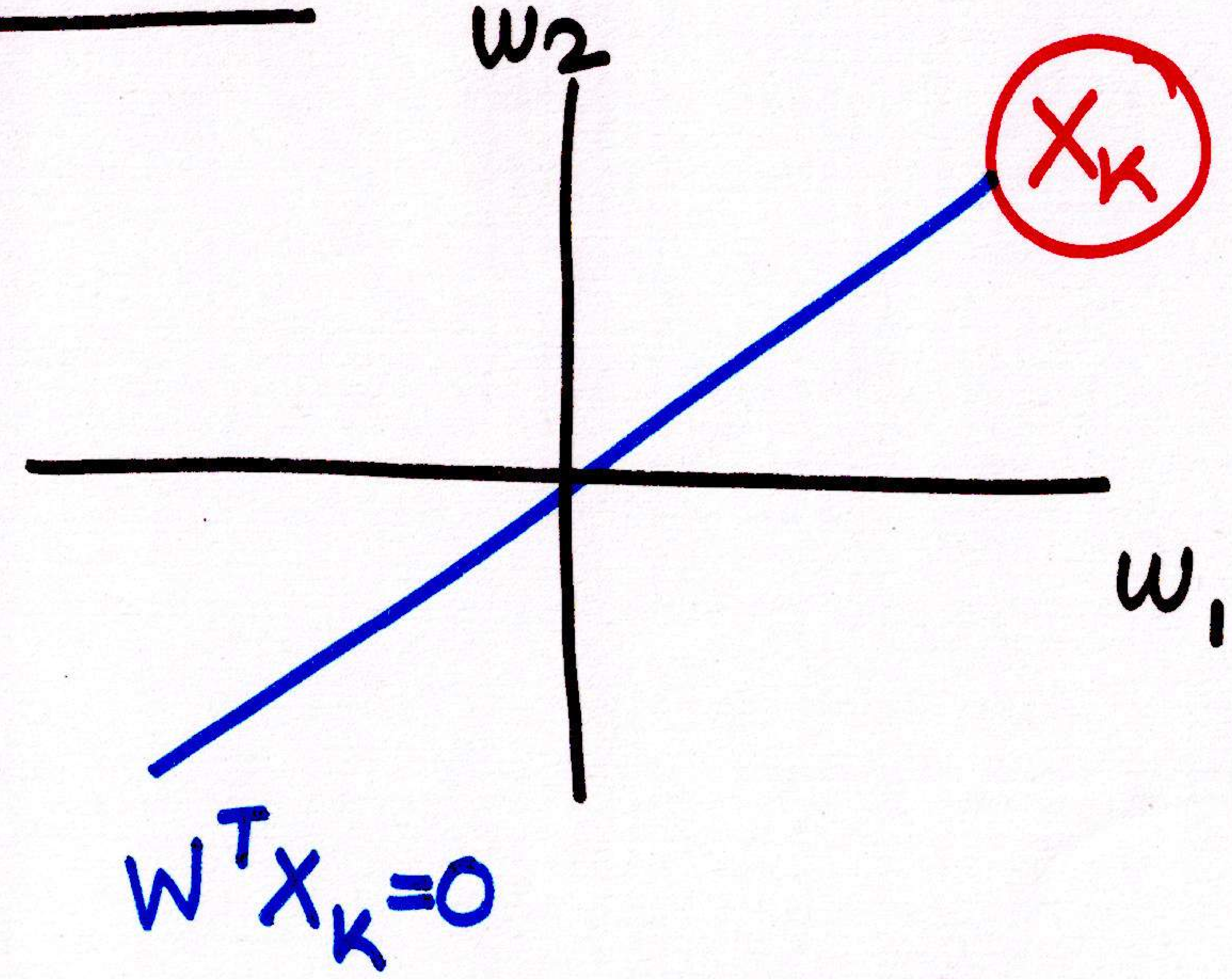
$$\boxed{X_k^T W_s > 0}$$

$\Downarrow$ Innerproduct is −ve

$$\boxed{X_k^T W_s < 0}$$

Pattern Space and Weight Space :





Pattern space

$X^T W_s$ divides patternspace into two parts, one in which inner product is +ve, other in which inner product is −ve. In classes are linearly seperable then C_0 & C_1 will be in 2 sides of this hyperplane.

Weight space

$W^T X_K$ divides weight space into two parts. This hyperplane is the locus of all points W such that $\boxed{W^T X_K = 0}$.

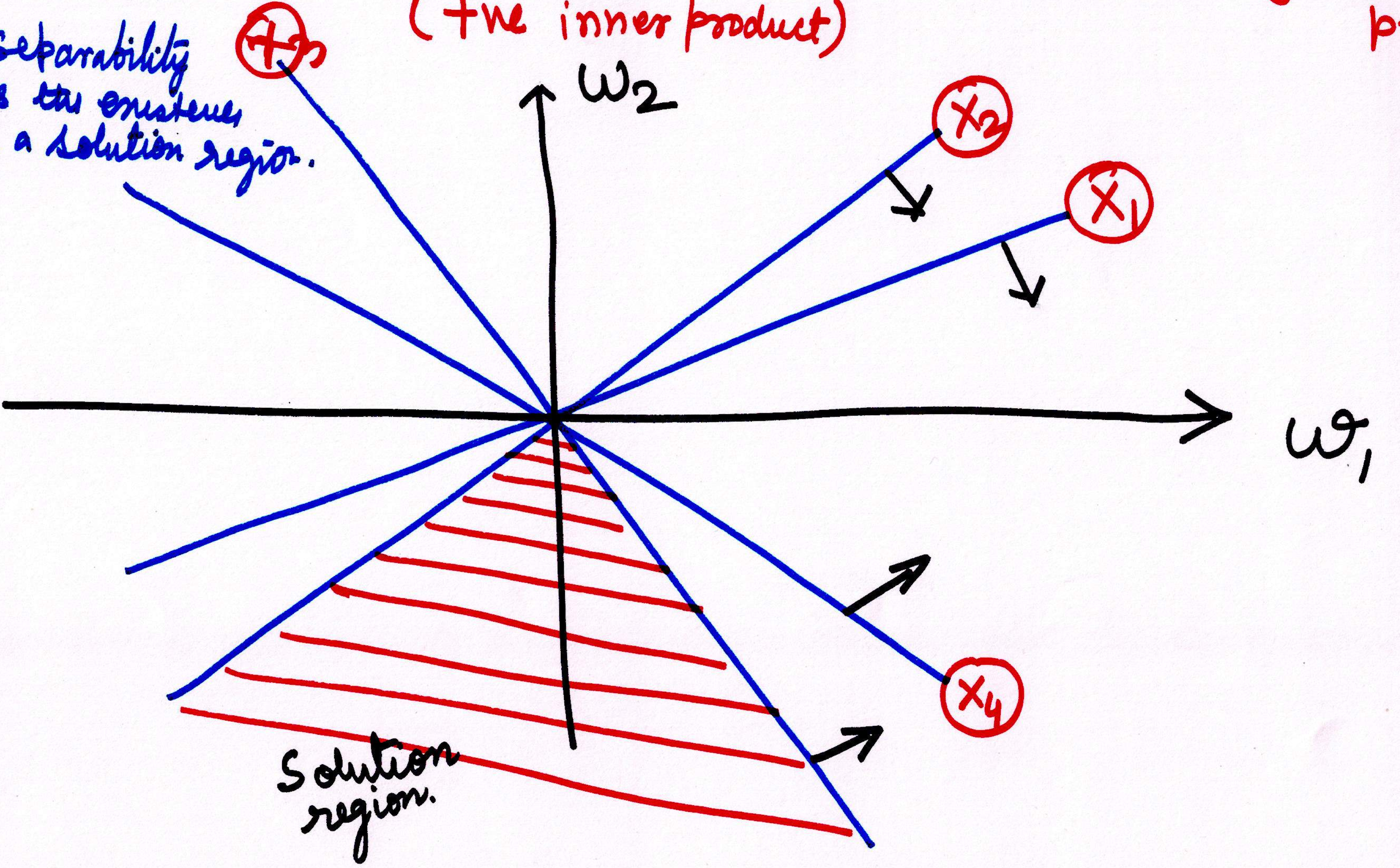
for

For each X_K in pattern space there will be a corresponding h-plane in weight space, while for every point in weight space there is a corresponding h-plane in pattern space.

Consider 4 patterns divided into two pattern sets $X_1 = \{x_1, x_2\}$ and $X_0 = \{x_3, x_4\}$. Using TLN classifier with neuronal signal 1 for X_1 and neuronal signal 0 for X_0.

(+ve inner product) (-ve inner product.)

Linear separability guarantees the existence of such a solution region.



In order to design an automated weight update procedure that can search out a solution weight vector, starting with an arbitrary initial weight vector we have to -:

* Consider each pattern in turn to assess the correctness of the present classification.

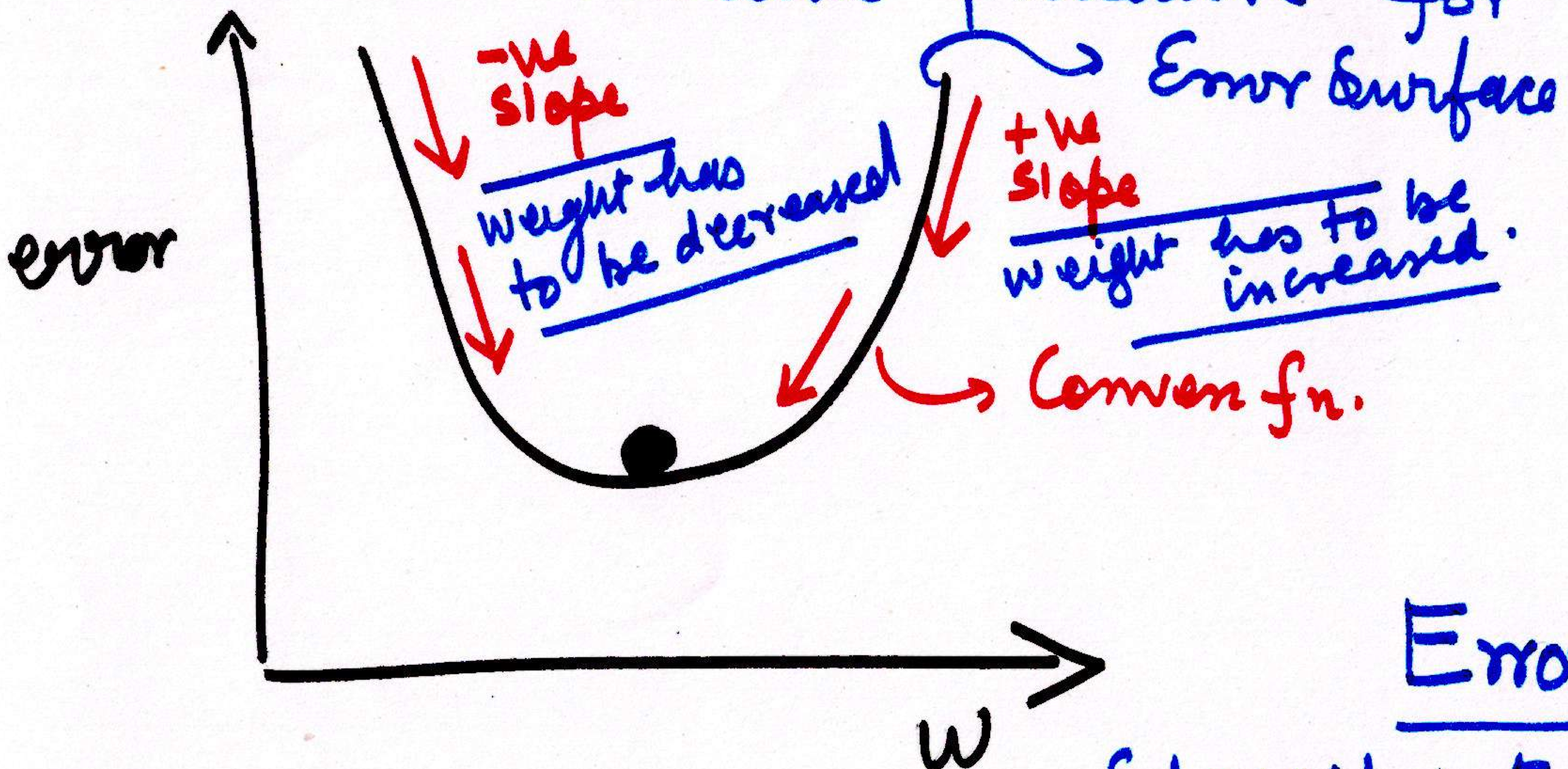
* Subsequently adjust the weight vector to eliminate any classification error.

* As set of all solution vector forms a convex cone, the weight update procedure should terminate as soon as it penetrates the boundary of this cone.

Perceptron Learning: Our assumption is that data is of 2-classes and it is linearily seperable.

- Starting with any initial (random) values of our weight vector $\bar{w}$ & $\bar{w}_0$ (z).
- Estimate the error wrt this h-plane.
- Comeup with an update Δw, that can reduce this error.

This is an iterative procedure for error minimization.



It looks for the direction that minimizes error.

$$\bar{w}(K+1) = \bar{w}(K) + \Delta\bar{w}(K)$$

$$\left\{ \Delta\bar{w} \propto -\frac{\partial \varepsilon}{\partial w} \right\}$$

Error → Misclassification

Classification is happening as

$$\text{Classification} \begin{cases} \text{if } \bar{w}\bar{x} + w_0 \geq 0 \text{ then } \bar{x}_n \in C_1 \\ \text{if } \bar{w}\bar{x} + w_0 < 0 \text{ then } \bar{x}_n \in C_2 \end{cases}$$

D_M : Have a set of misclassified Training examples.

D_1 and D_2 : Data of (+ve) and (−ve) class

y_n : Class labels associated with x_n , $y_n \in \{+1, -1\}$

Augmented Vectors : $\bar{a} = \begin{bmatrix} w_0 \\ w_1 \\ | \\ w_d \end{bmatrix}$ and $z = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$

Class Assignment :

$y_n = +1 \rightarrow$ if $w^T x_n + w_0 \geq 0 \longrightarrow \bar{a}^T z_n \geq 0$

$y_n = -1 \rightarrow$ if $w^T x_n + w_0 < 0 \longrightarrow \bar{a}^T z_n < 0$

$\Longrightarrow y_n * (a^T z_n) > 0$ → TRUE for Correct classification → FALSE for Misclassification.

We have to modify our weights such that after each iteration D_M reduces and we stops when it is empty.

$\Longrightarrow$ In order to show convergence one has to assume that linear sep. and weights are changed and error got reduced (atleast NOT increase) in any step.

Perceptron Cost fn: $$J_p = -\sum_{x_m \in D_m} y_n (a^T z_n)$$ (total error) "

$x_m \in D_m$ (set of mis-classified samples)

At any K^{th} iteration: $$\Delta \bar{a}(K) \propto -\frac{\partial J_p}{\partial w}$$

$$\therefore \Delta \bar{a}(K) = -\eta \frac{\partial J_p}{\partial a(K)} = -\eta \frac{\partial}{\partial a(K)} \left(-\sum_{x_m \in D_m(K)} y_m (\bar{a}(K))^T z_m \right)$$

h plane parameter updation

Learning Rate

$$= \eta \sum_{x_m \in D_m(K)} y_n \bar{z}_n$$

Update Rule: $$a(K+1) = a(K) + \Delta \bar{a}(K)$$

$$\therefore a(K+1) = a(K) + \eta \sum_{x_m \in D_m(K)} y_n z_n$$

Learning rate

label

Augmented misclassified example

Set of mis-classified examples at K^{th} iteration.

ALGO: ① Initilize augumented weight vector $a[0]$ for 0^{th} iteration.

② At every (say k^{th}) iteration, Maintain a mis-classification set $D_m(K)$ for $\bar{a}(K)$ (h-plane).

③ Update the h-plane parameter as:

$$a(K+1) = a(K) + \eta \sum_{x_n \in D_m(K)} y_n z_n$$

[Batched Perceptron]

h-plane updation only after full batch.

④ Repeat step 2 till $D_m(K)$ is empty.

The perceptron learning requires only a single neuron and its Convergence proof states - that it will take only finite number of steps to Converge provided the data is linearly seperable. [PROOF in BOOK].

Single sample perceptron : It donot consider all misclassified examples, instead consider the current mis-classified example and update the h-plane.

Instantaneous error $\longrightarrow$ $$J_p = -y_n(\bar{a}^T z_n)$$ error wrt to a particular example.

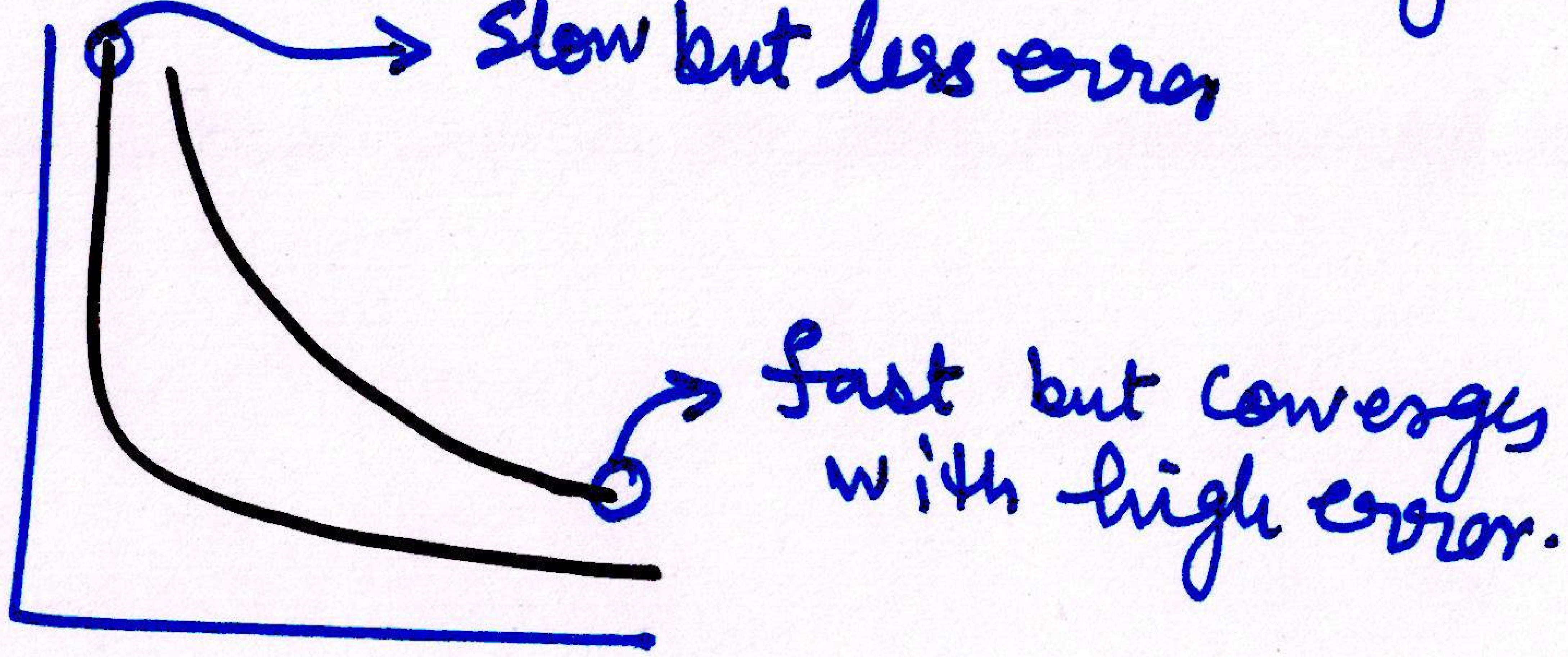
1- Initialze $a(0)$

2- At any K^{th} iteration : $a(K+1) = a(K) - \eta \frac{\partial J_p}{\partial a}$

$$a(K+1) = a(K) + \eta\, y_n z_n$$

3- Repeat step 2 untill all x_n's are correctly classified.

Batch update $\Longrightarrow$ slow convergence | Single $\Longrightarrow$ fast but may oscillate



We have seen fixed η.
One can start with high η and then reduce η as going on and fix it after few sets.

Convergence Proof for Single-Sample Correction :

fixed increment ($\eta = 1$)

Assuming we have ③ samples, y_1, y_2, y_3 and are presented Cyclically :

→ Mis-classified

$$y_1,\ y_2,\ y_3,\ y_1,\ y_2,\ y_3,\ y_1,\ y_2,\ y_3\ ---$$

$$y_1 \to y^1,\quad y_3 \to y^2,\quad y_1 \to y^3,\quad y_2 \to y^4,\quad y_2 \to y^5$$

Hence fixed-increment rule for generating a sequence of weight vectors

① $a(1)$ arbitrary ② $a(k+1) = a(k) + y^k \quad k \geq 1$

Here $\boxed{a^t(k)\, y^k \leq 0}$ as mis-classified. (Update untill all Samples are Correctly classified)

Geometrically: ① $a(k)$ is misclassifies y^k; ② Hence $a(k)$ is not on the +ve Side of $[a^t y^k \geq 0]$ h-plane.

③ Adding y^k to $a(k)$ moves weight vector towards h-plane.

④ New inner product $a^t(k+1)\, y^k$ is larger than old inner product $\cancel{a^t(k+1)}$ $a^t(k)\, y^k$ by amount $\| y^k \|^2$

Correction in good direction.

It terminates only when samples are linearly seperable [we prove it indeed converge] 18

If traing samples are linearly seperable, then sequence of weight vectors will terminate at a solution vector.

Proof: Let $\hat{a} \Rightarrow$ solution vector

we try to show $\|a(k+1) - \hat{a}\| < \|a(k) - \hat{a}\|$ — (1)

(for sufficiently long solution vector)

(*) $\hat{a}^t y_i \geq 0, \forall i$ — (2) (*) $\alpha \rightarrow$ +ve scale factor (3)

$$\therefore \quad a(k+1) - \alpha\hat{a} = (a(k) - \alpha\hat{a}) + y^k \quad (4) \quad (\text{from previous})$$

$$\longrightarrow \|a(k+1) - \alpha\hat{a}\|^2 = \|a(k) - \alpha\hat{a}\|^2 + 2(a(k) - \alpha\hat{a})^t y^k + \|y^k\|^2 \quad (5)$$

Using (2)

$$\|a(k+1) - \alpha\hat{a}\|^2 \leq \|a(k) - \alpha\hat{a}\|^2 - 2\alpha\hat{a}^t y^k + \|y^k\|^2 \quad (6)$$

strictly +ve

if (α) is very high it will dominate $\|y^k\|^2$.

$$\left.\begin{aligned} \beta^2 &= \max_i \|y_i\|^2 \\ \gamma &= \min_i [\hat{a}^t y_i] > 0 \end{aligned}\right\} \quad (7)$$

$$\therefore \; \| a(k+1) - \alpha \hat{a} \|^2 \leqslant \| a(k) - \alpha \hat{a} \|^2 - 2\alpha\gamma + \beta^2 \quad (8)$$

Choose: $\left[\alpha = \dfrac{\beta^2}{\gamma} \right]$

We get

$$\| a(k+1) - \alpha \hat{a} \|^2 \leqslant \| a(k) - \alpha \hat{a} \|^2 - \beta^2 \quad (9)$$

Hence the squared distance from $a(k)$ to $\alpha \hat{a}$ is reduced by atleast β^2 at each Correction (10)

After K Corrections: Starting from $a(1)$

$$\| a(k+1) - \alpha \hat{a} \|^2 \leqslant \| a(1) - \alpha \hat{a} \|^2 - k\beta^2 \quad (11)$$

As above thing Cannot be (−ve) → Sequence of Correction must terminate (12) after no more than k_0 Corrections where

$$\boxed{k_0 = \frac{\| a(1) - \alpha \hat{a} \|^2}{\beta^2}} \quad (13)$$