# Foundations of Data Science & Machine Learning

Summary — Week 02
Devansh Singh Rathore
111701011

B.Tech. in Computer Science & Engineering
Indian Institute of Technology Palakkad

March 14, 2021

**Abstract**

We study about Perceptron Learning Algorithm and its proof of upper bound of its computational complexity. Further we discuss non-linear separators, embedding in higher dimensional space using Kernel trick.

## 1 Perceptron learning algorithm

→ **Perceptron** is a supervised machine learning algorithm used for solving binary classification problems.

**NOTE: We assume b = 0. Equivalently there exists a separating hyperplane passing through origin.**
$x = (x_1, x_2, ...x_n) \in \mathbb{R}^n$, $b \in \mathbb{R}$
$x' = (x_1, x_2, ...x_n, 1) \in \mathbb{R}^{(n+1)}$
since $a = (a_1, a_2, ...a_n) \in \mathbb{R}^n$, assume $a' = (a_1, a_2, ...a_n, -b) \in \mathbb{R}^{(n+1)}$
For the good points we needed, $\langle x, a\rangle \geq b + \delta/2$
Now, $\langle x', a'\rangle = \langle x, a\rangle - b \geq \delta/2$
Similarly, for bad points we needed, $\langle x, a\rangle \leq b - \delta/2$ Now, $\langle x', a'\rangle = \langle x, a\rangle - b \leq -\delta/2$

**Input:** Two finite sets $G, B \subset \mathbb{R}^n$ which are linearly separable by a hyperplane passing through the origin.

**Output:** $a \in \mathbb{R}^n$ such that $\forall x \in G$, $\langle x, a\rangle > 0$ and $\forall x \in B$, $\langle x, a\rangle < 0$.

**Algorithm:** initially let's keep a = 0
Repeat until there is no update in a -
  for each $x \in G$ :
    if($\langle x, a\rangle \leq 0$) :
      then a = a + x
  for each $x \in B$ :
    if($\langle x, a\rangle \geq 0$) :
      then a = a - x

**Claim:** If G and B are linearly separable then the Perceptron algorithm terminates after $O(R^2/\delta^2)$ i.e. $O(1/RelativeMargin^2)$ updates, where

R $= max_{x \in G \cup B} ||x||$

$\delta(margin) = min_{x \in G, y \in B} ||x - y||$

Relative Margin $= \delta/R$

**Proof:** Since G and B are given as linearly separable. $\exists a^*$ s.t. $\forall x \in G$, $\langle x, a^* \rangle \geq \delta/2$ and $\forall x \in B$, $\langle x, a^* \rangle \leq -\delta/2$.

assume $||a^*|| = 1$, (w.l.o.g.)

$\rightarrow$ For $x \in G$, update occurs when $\langle x, a \rangle \leq 0$

$a+ = x$

$\langle a, a^* \rangle + = \langle x, a^* \rangle$

since $\langle x, a^* \rangle \geq \delta/2$, increase in $\langle a, a^* \rangle \geq \delta/2$

$||a||^2 = ||a + x||^2$, so $||a||^2$ increases by $2 \langle x, a \rangle + ||x||^2$ $(\langle x, a \rangle \leq 0)$

i.e. $||a||^2$ increases by $\leq R^2$

$\rightarrow$ For $x \in B$, update occurs when $\langle x, a \rangle \geq 0$

$a- = x$

$\langle a, a^* \rangle - = \langle x, a^* \rangle$

since $- \langle x, a^* \rangle \geq \delta/2$, increase in $\langle a, a^* \rangle \geq \delta/2$

$||a||^2 = ||a - x||^2$, so $||a||^2$ increases by $-2 \langle x, a \rangle + ||x||^2$ $(\langle x, a \rangle \geq 0)$

i.e. $||a||^2$ increases by $\leq R^2$

$\rightarrow$ After k updates,

$||a|| \geq \langle a, a^* \rangle \geq k.\delta/2$, so $||a|| \geq k.\delta/2$ —(i)

while on the other hand $||a||^2 \leq k.R^2$ i.e. $||a|| \leq \sqrt{k}.R$ —(ii)

From (i) and (ii), $k \leq 4 * R^2/\delta^2$

**Hence there cannot be more than $4 * R^2/\delta^2$ updates in Perceptron algorithm.**

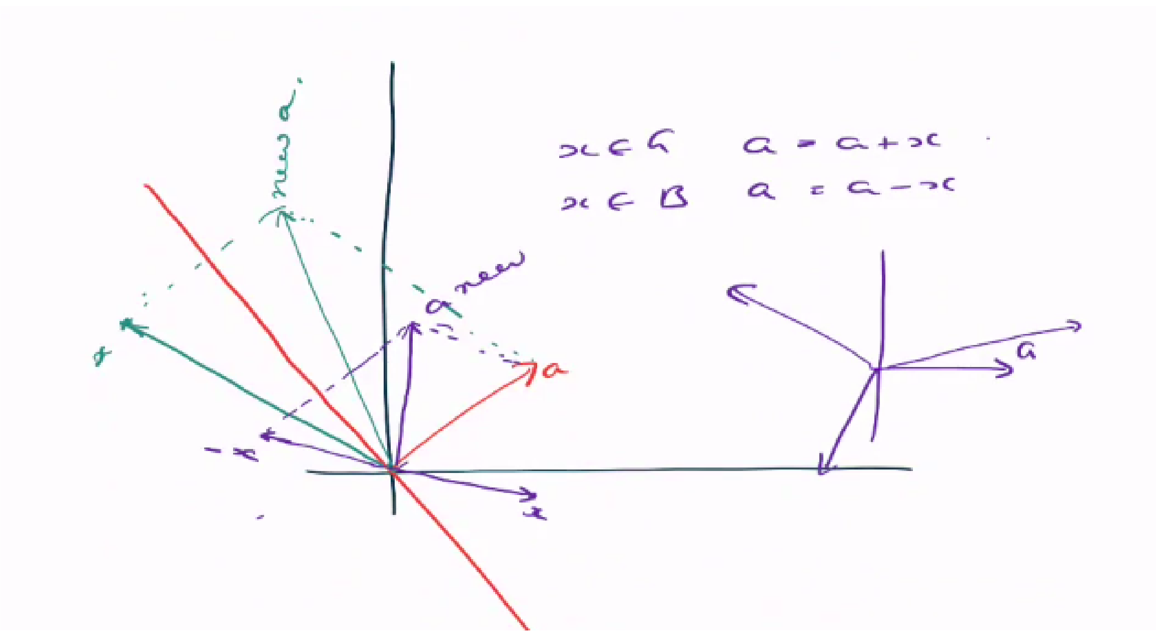


Fig 0.0 Working of Perceptron Algorithm

$\rightarrow$ **Support Vector Machine (SVM)** is supervised machine learning algorithm used for optimal linear classification. SVM tries to find the maximum margin between Good and Bad points and finds suitable separating hyperplane.

# 2 Non-linear Separators using the Kernel Trick

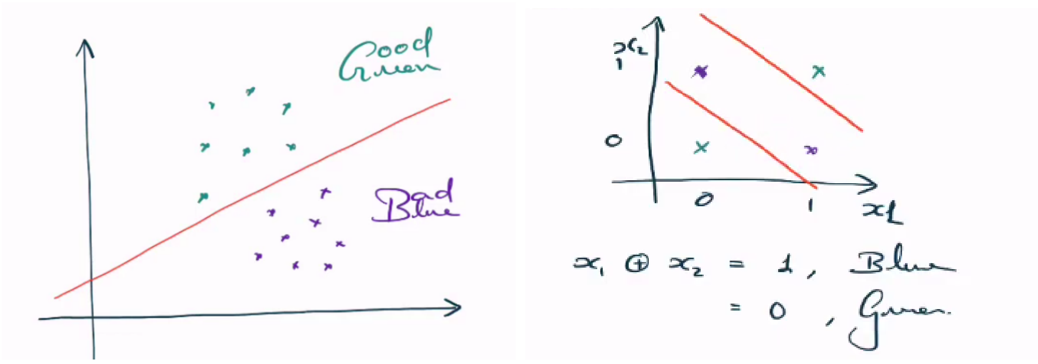


Fig 0.1 Linear Vs Non Linear Classification problem

→ We can have multiple linear classifier chunks for classifying points which cannot be classified using a single linear classifier hyperplane.

→ The base case of non-linear classification problem (Fig 0.1) is the XOR plot.

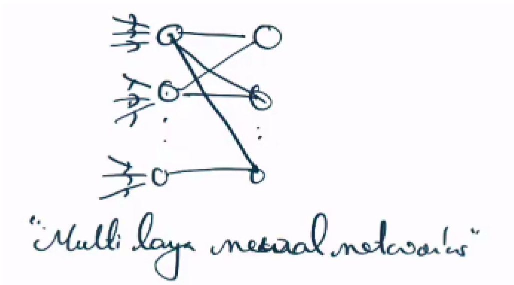


Fig 0.2 Multilayer Neural Network

## 2.1 Embedding in a higher dimensional space

→ Let's take an example where the Good points are within a circular boundary in a 2D plane and the Bad points lie outside the circular margin of radius R.

Now, the **Embedding function** is defined as: $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^N$.

Here in this example, $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$

$\phi((x_1, x_2)) = (x_1, x_2, x_1^2 + x_2^2)$

→ Now in 3D space, the Good points lie below the plane $x3(= x_1^2 + x_2^2) = R^2$, while the Bad points lie above this plane. Hence we can say that the $\phi$ images of G and B are classified using **linear separator in** $\mathbb{R}^3$.

→ In general terms,

$\phi : \mathbb{R} \rightarrow \mathbb{R}^{N=d+1}$

$\phi(x) = (1, x^1, x^2, ...x^d) \in \mathbb{R}^{d+1}$

Separating hyperplane in $\mathbb{R}^{d+1}$ looks like $(a_0, a_1, ...a_d)$

And for classification, we need to deal with polynomial equations like

$a_0 + a_1x^1 + a_2x^2 + ...a_dx^d >_{<} 0$

→ So you have at your disposal all polynomials of degree $\leq d$ as potential separators.

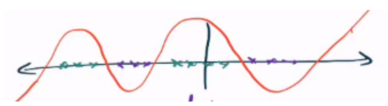

Fig 0.3 Polynomial Classifier

$\rightarrow \phi : \mathbb{R}^2 \rightarrow \mathbb{R}^N$

for d = 3, $\phi((x_1, x_2)) = (1, x_1, x_2, x_1^2, x_2^2, x_1x_2, x_1^3, x_2^3, x_1^2x_2, x_1x_2^2)$, so N = 10

$\rightarrow \phi : \mathbb{R}^3 \rightarrow \mathbb{R}^N$

for d = 2, n = 3, $\phi((x_1, x_2, x_3)) = (1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3)$, so N = 10

$\rightarrow$ In general, N(n, d) = number of terms with degree $\leq$ d in $(1 + x_1^1 + x_1^2 + ...x_1^d)(1 + x_2^1 + x_2^2 + ...x_2^d)...(1 + x_n^1 + x_n^2 + ...x_n^d)$

$= C_d^{n+d}$

= No. of monic monomials in n variables with degree $\leq$ d.

$\rightarrow$ **Perceptron Learning algorithm with embedding**

**Input:** $G = x_1, x_2, ...x_k \subseteq \mathbb{R}^n$

$B = x_{k+1}, x_{k+2}, ...x_l \subseteq \mathbb{R}^n$

$\phi : \mathbb{R}^n \rightarrow \mathbb{R}^N$

**Output:** $a \in \mathbb{R}^N$, $\forall x_j \in G, \langle a, \phi(x_j) > 0 \rangle$, $\forall x_j \in B, \langle a, \phi(x_j) < 0 \rangle$

**Algorithm:**

Initially $a = (0, 0, ...) \in \mathbb{R}^N$

Repeat until there is no update in a:

  for each $x_j \in G$ :

    if $\langle a, \phi(x_j) \rangle < 0, a = a + \phi(x_j)$

  for each $x_j \in B$ :

    if $\langle a, \phi(x_j) \rangle > 0, a = a - \phi(x_j)$

$\rightarrow$ This will terminate when $\phi_G = \{\phi(x) : x \in G\}$ & $\phi_B = \{\phi(x) : x \in B\}$ are linearly separable in $\mathbb{R}^N$.

$\rightarrow$ This algorithm is computationally expensive when N is very large. We can make it efficient using the **"Kernel trick"**.

## 2.2 Kernel Trick

$\rightarrow$ Observation 1: Most of the computations are inner products of the form $\langle a, \phi(x) \rangle$.

$\rightarrow$ Observation 2: a is a linear (integer) combination of $\phi(x_i), i \in G \cup B$ i.e.

$$a = \sum_{j=1}^{l} \alpha_j \phi(x_j)$$

where $\alpha_j$ = isBad($x_j$) * number of times $x_j$ passed the update condition

isBad($x_j$) = -1 if $x_j \in B$, else 1.

$\rightarrow$ From Observation 1 & 2 -

$\langle a, \phi(x_i) \rangle = \left\langle \sum_{j=1}^{l} \alpha_j \phi(x_j), \phi(x_i) \right\rangle$

  $= \sum_{j=1}^{l} \alpha_j \langle \phi(x_j), \phi(x_i) \rangle$, where $x_i, x_j \in G \cup B$

$\rightarrow$ So the problem of computational efficiency boils down to checking whether we can compute $\langle \phi(x_i), \phi(x_j) \rangle$ efficiently i.e. with complexity/time independent of N.

if $\phi : \mathbb{R} \rightarrow \mathbb{R}^{d+1}$

$\phi(x) = (1, x, x^2, ...x^d)$

$\phi(y) = (1, y, y^2, ...y^d)$

$\langle \phi(x), \phi(y) \rangle = 1 + xy + x^2y^2 + ... + x^dy^d$

  $= (1 - (xy)^{d+1})/(1 - (xy))$, using G.P.

$\rightarrow$ **Perceptron Learning algorithm with Kernel Trick**

**Input:** $G = x_1, x_2, ...x_k \subseteq \mathbb{R}^n$

$B = x_{k+1}, x_{k+2}, ...x_l \subseteq \mathbb{R}^n$
$K : \mathbb{R}^n x \mathbb{R}^n \rightarrow \mathbb{R}$, $K(x, y) = \langle \phi(x), \phi(y) \rangle$
**Output:** $(\alpha_1, \alpha_2, ...\alpha_l) \in \mathbb{Z}^l, \forall x_i \in G, \sum_{j=1}^{l} \alpha_j K(x_j, x_i) > 0, \forall x_i \in B, \sum_{j=1}^{l} \alpha_j K(x_j, x_i) < 0$
**Algorithm:**
Initially $\alpha = (0, 0, ...) \in \mathbb{R}^N$
Repeat until there is no update in $\alpha$:
    for each $x_j \in G$ :
        if $\sum_{i=1}^{l} \alpha_j K(x_j, x_i) \leq 0, \alpha_j + = 1$
    for each $x_j \in B$ :
        if $\sum_{i=1}^{l} \alpha_j K(x_j, x_i) \geq 0, \alpha_j - = 1$