Perceptron Learning Algorithm

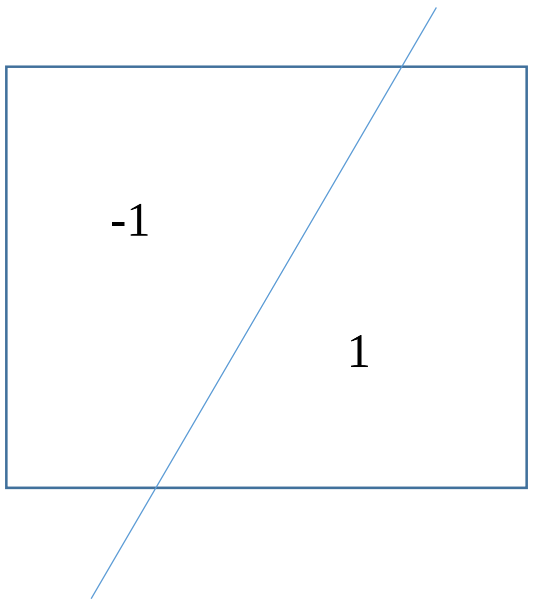
JrPhy

Introduction

- Perceptron Learning Algorithm(PLA) is the simplest idea for binary classifying, that is, all object are labeled 1 or -1, then find a methods to separate 1 and -1.
- Suppose the data can be separate by a line, so the goal is to finding the line.
- If the data can’t be separate by a line, then we will use another method.

Point and Line

- Suppose a plane is split by a line $ax+by+c = 0$.
- Point A(x_0, y_0) at the right of the line if $ax_0+by_0+c > 0$
- Point B(x_1, y_1) at the left of the line if $ax_1+by_1+c < 0$
- By this fact, we label the point 1 if it is at the right of the line, and -1 at the left or 1 at right -1 at left.
- So the dataset is
- $(x_0, y_0, 1)$
- $(x_1, y_1, -1)$



In matrix form

- Turn the eq. into matrix form, it's convenient to apply it on higher dimension.
- There are 3 coefficients in a line: a, b, c. collect them in a matrix ***w***, and the (x_i, y_i) in the other matrix ***x***.
- $\boldsymbol{w} = [a \;\; b \;\; c]$, $\boldsymbol{x} = [x_i \;\; y_i \;\; 1]$ → $\langle \boldsymbol{w}, \boldsymbol{x}^{\boldsymbol{T}} \rangle = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = ax_i + by_i + c$
- Note that 1 in ***x*** is not the label = *y*, it's the constant term.

In matrix form

- In higher dimension, the polynomial can be written as

$$w_n x_n + w_{n-1} x_{n-1} + \dots + w_1 x_1 + w_0 x_0 = \sum_{i=1}^{n} w_i x_i + w_0 x_0 = \sum_{i=0}^{n} w_i x_i = < w, x^T >$$

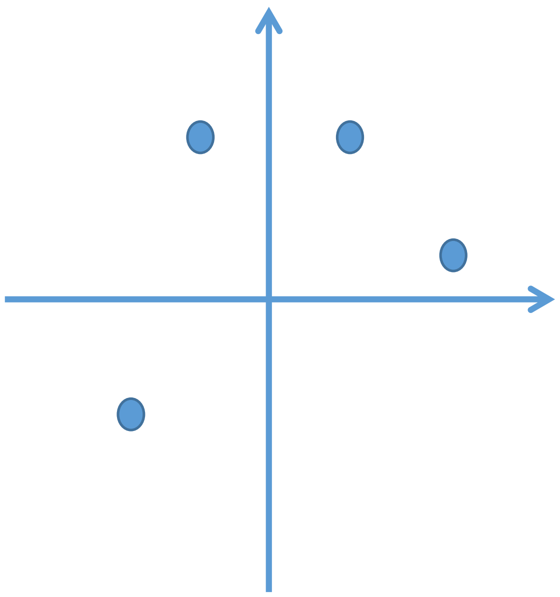
- Where $w_0 x_0$ is the threshold term, x_0 is a nonzero term, w_0 is initialized 0, and updated in each step.

$$<\boldsymbol{w}, \boldsymbol{x^T}> > 0 \rightarrow +1 \ ; \quad <\boldsymbol{w}, \boldsymbol{x^T}> < 0 \rightarrow -1$$

- $<\boldsymbol{w}, \boldsymbol{x^T}> == y \rightarrow$ continue, $<\boldsymbol{w}, \boldsymbol{x^T}> !=$ label $\rightarrow w^{t+1} = w^t + x_i y_i$
- In the following example, w[0], w[1], w[2] are the coefficient of constant, *x* and *y* respectively

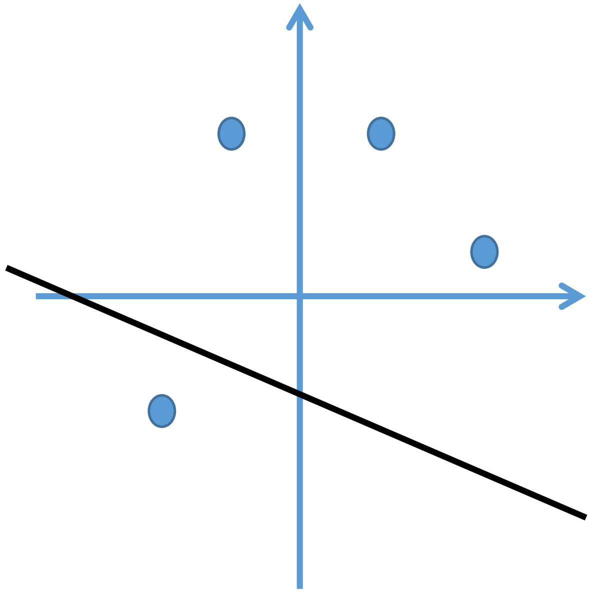
How PLA works

- Suppose there are 4 points in 2D-plane
- (0.3, 0.7, -1),
- (-0.4, -0.6, 1),
- (0.9, 0.2, -1),
- (-0.3, 0.7, 1)
- The last is label
- Start from the (0, 0)
- Here we set constant term as 1



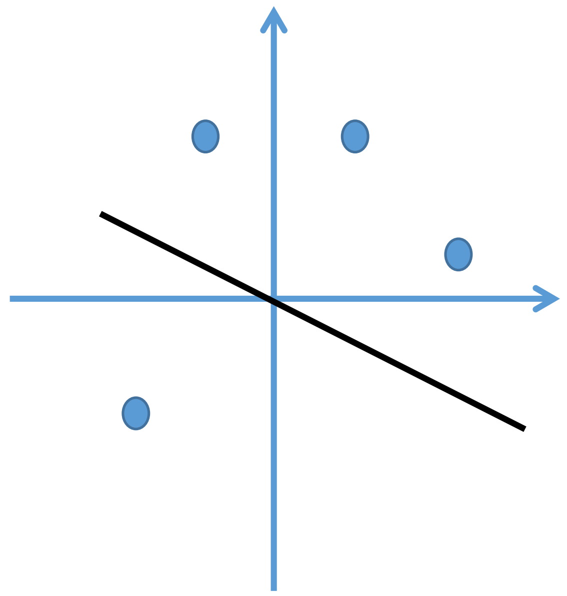
How PLA works

- 1^{st} step we get 0 because ***w*** is a 0 vector
- **(0.3, 0.7, -1)**,
- (-0.4, -0.6, 1),
- (0.9, 0.2, -1),
- (-0.3, 0.7, 1)
- After updating ***w***, we get
- w[0] = -1, w[1] = -0.3, w[2] = -0.7
- line: $-1-0.3x-0.7y = 0$



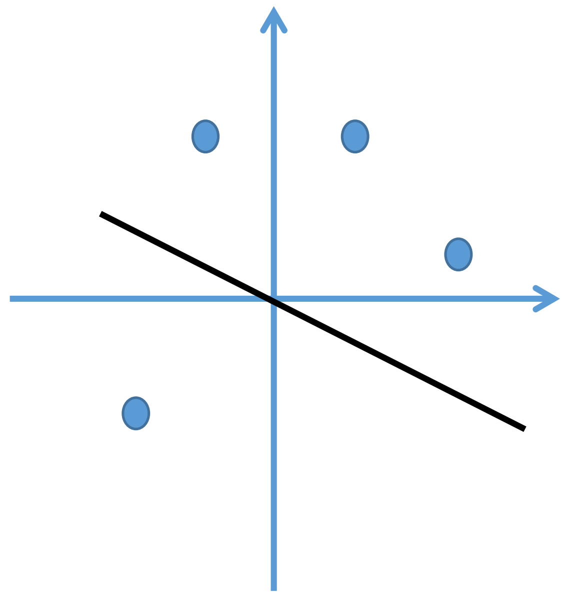
How PLA works

- 2^{nd} step
- (0.3, 0.7, -1),
- **(-0.4, -0.6, 1),**
- (0.9, 0.2, -1),
- (-0.3, 0.7, 1)
- After updating ***w***, we get
- w[0] = 0, w[1] = -0.7, w[2] = -1.3
- line: $-0.7x-1.3y = 0$



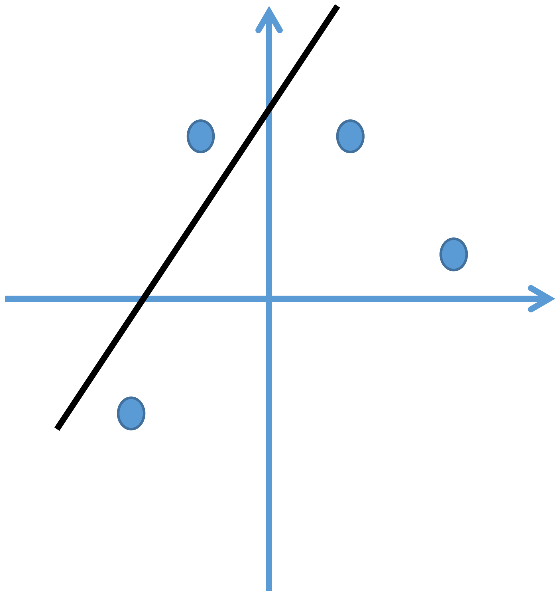
How PLA works

- 3rd step the sign of inner product is the same as *y*, so no updating
- (0.3, 0.7, -1),
- (-0.4, -0.6, 1),
- **(0.9, 0.2, -1),**
- (-0.3, 0.7, 1)



How PLA works

- 4^{th} step
- (0.3, 0.7, -1),
- (-0.4, -0.6, 1),
- (0.9, 0.2, -1),
- **(-0.3, 0.7, 1)**
- After updating ***w***, we get
- w[0] = 1, w[1] = -1, w[2] = 0.6
- line: $1-x+0.6y = 0$
- The final result is w[0] = 0, w[1] = -1.9, w[2] = -0.8



How PLA works

- The value of w in each step as following

w[0]	w[1]	w[2]
0.000000	0.000000	0.000000
0.000000	0.000000	0.000000
1.000000	-0.400000	-0.600000
0.000000	-1.300000	-0.800000
1.000000	-1.600000	-0.100000
0.000000	-1.900000	-0.800000
0.000000	-1.900000	-0.800000
0.000000	-1.900000	-0.800000
0.000000	-1.900000	-0.800000
0.000000	-1.900000	-0.800000
0.000000	-1.900000	-0.800000
0.000000	-1.900000	-0.800000

How PLA works

- It will stop until all classifications are correct. So if your data is **not linear separable**, the program will not stop.
- If the data is linear separable, then after many loops, i,e, it makes no mistake, then PLA stops.
- The same data with another order, you may get different lines, but the line still separate all the data.