Perceptron

The simplest neural network

Dr. Mauricio Toledo-Acosta

Diplomado Ciencia de Datos con Python

Table of Contents

Introducción

Perceptron

Modelo supervisado de clasificación binaria que busca encontrar una frontera de decisión que separe las clases de puntos. Esto lo hace usando una función umbral que depende de un vector de pesos, este vector lo va actualizando conforme recorre los puntos de entrenamiento.

El modelo lineal de clasificación

- El Algoritmo Perceptrón (Rosenblatt, 1961) jugó un rol muy importante en la historia del Machine Learning. Inicialmente, simulado en una computadora IBM 704 en Cornell en 1957. Para principios de 1960, se había diseñado un hardware de propósito específico, que implementaba directamente y de forma paralela el algoritmo de aprendizaje.

El modelo lineal de clasificación

- El Algoritmo Perceptrón (Rosenblatt, 1961) jugó un rol muy importante en la historia del Machine Learning. Inicialmente, simulado en una computadora IBM 704 en Cornell en 1957. Para principios de 1960, se había diseñado un hardware de propósito específico, que implementaba directamente y de forma paralela el algoritmo de aprendizaje.
- Fue criticado por Marvin Minsky, mostrando las deficiencias de las redes de perceptrón de una sola capa para resolver problemas no separables. Esto propició un vacío en la investigación de la computación neural que duró hasta mediados de los 80.

Table of Contents

$$y = \text{sgn}\left(\sum_{i=1}^{d} w_i x_i + w_0\right)$$

El algoritmo

Recorremos el conjunto de entrenamiento

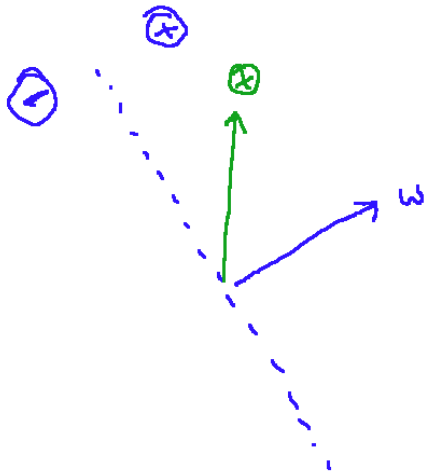
$$(x_1, y_1), (x_2, y_2) \cdots , (x_N, y_N).$$

El algoritmo

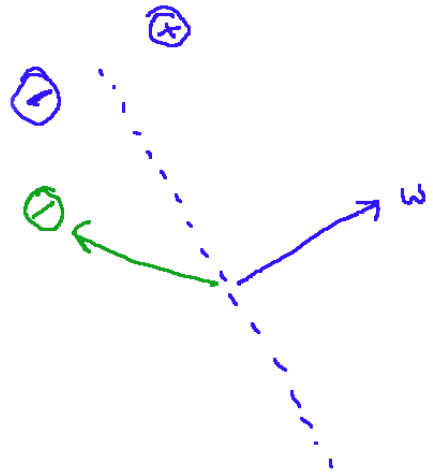
Recorremos el conjunto de entrenamiento

$$(x_1, y_1), (x_2, y_2) \cdots, (x_N, y_N).$$

a. Si el punto está bien clasificado:



(a) Si $y = +1$

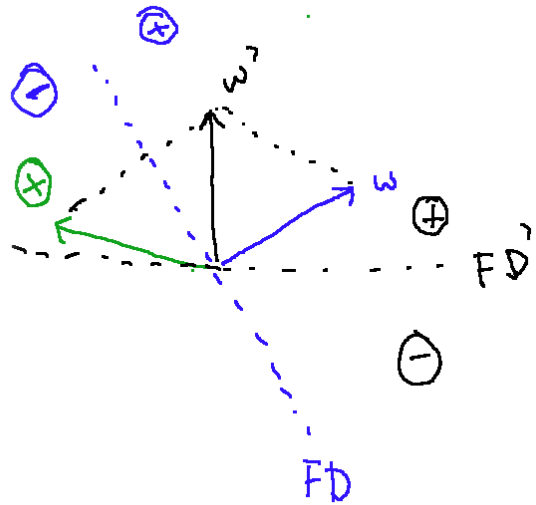


(b) Si $y = -1$

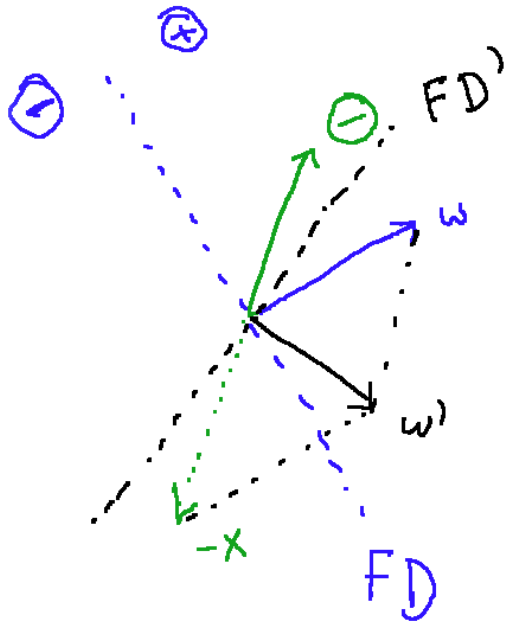
En ambos casos, $y_i \langle x_i, w \rangle > 0$

El algoritmo

- Si el punto está mal clasificado:



(a) Si $y = +1$

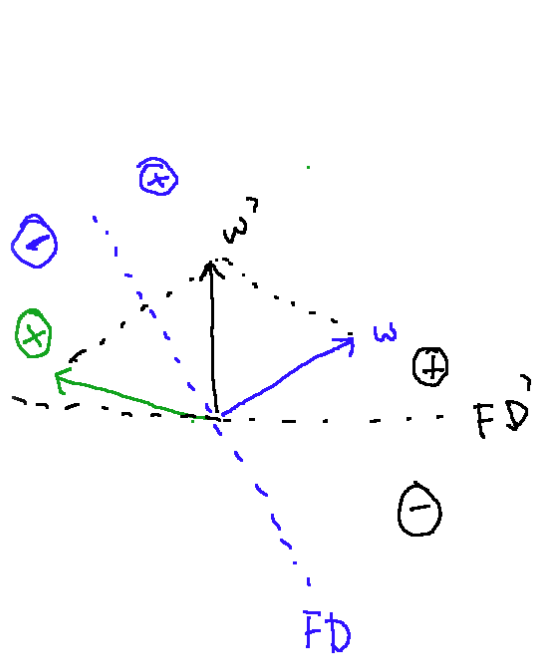


(b) Si $y = -1$

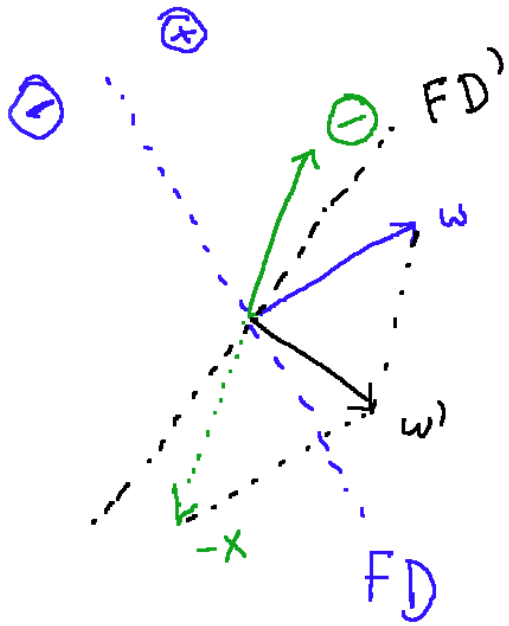
En ambos casos, $y_i \langle x_i, w \rangle < 0$

El algoritmo

- Si el punto está mal clasificado:



(a) Si $y = +1$



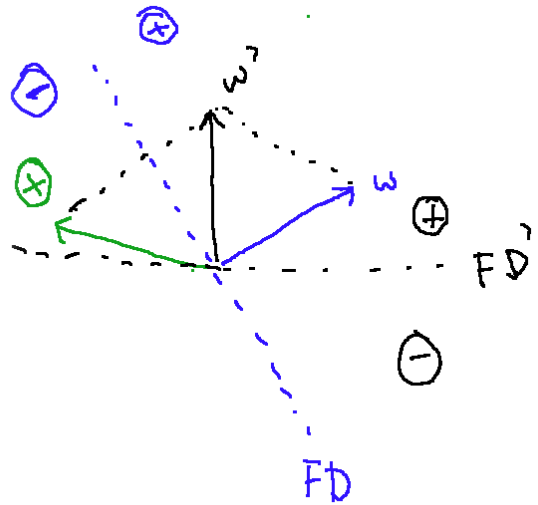
(b) Si $y = -1$

En ambos casos, $y_i \langle x_i, w \rangle < 0$

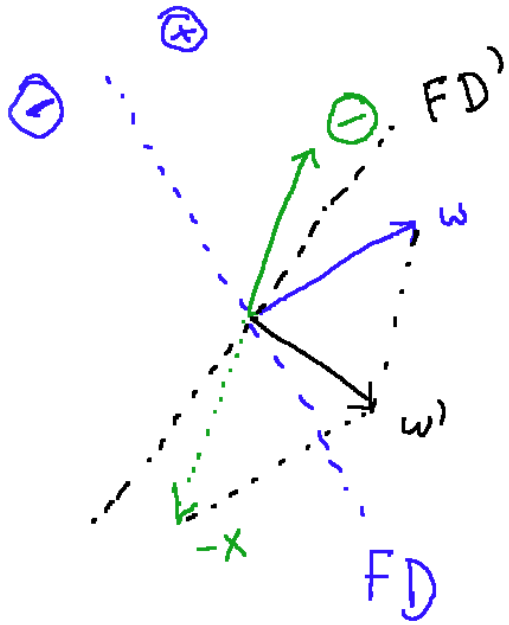
Actualizamos el vector w con $w + yx$

El algoritmo

- b Si el punto está mal clasificado:



(a) Si $y = +1$



(b) Si $y = -1$

En ambos casos, $y_i \langle x_i, w \rangle < 0$

Actualizamos el vector w con $w + yx$

Repetimos para el conjunto de entrenamiento y realizamos varias épocas.

El algoritmo

El algoritmo se resume así, para cada época:

Entrada: *Datos etiquetados de entrenamiento* $\mathbf{X}$ *en coordenadas homogéneas*

Salida: *Vector de pesos* $\mathbf{w}$ *que define al clasificador*

$\mathbf{w} = \mathbf{0}$ *#Otras inicializaciones son posibles*
$converge = \textbf{\textit{Falso}}$
mientras $converge == \textbf{\textit{Falso}}$:
    $converge = \textbf{\textit{Verdadero}}$
    para i **en** $|\mathbf{X}|$:
        si $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \leq 0$ **entonces**: #xi mal clasificado
            $\boldsymbol{w} = \boldsymbol{w} + y_i \eta \mathbf{x}_i$ #$0 < \eta \leq 1$ es la tasa de aprendizaje
            $converge = \textbf{\textit{Falso}}$
        fin
    fin
fin

El algoritmo

El algoritmo se resume así, para cada época:

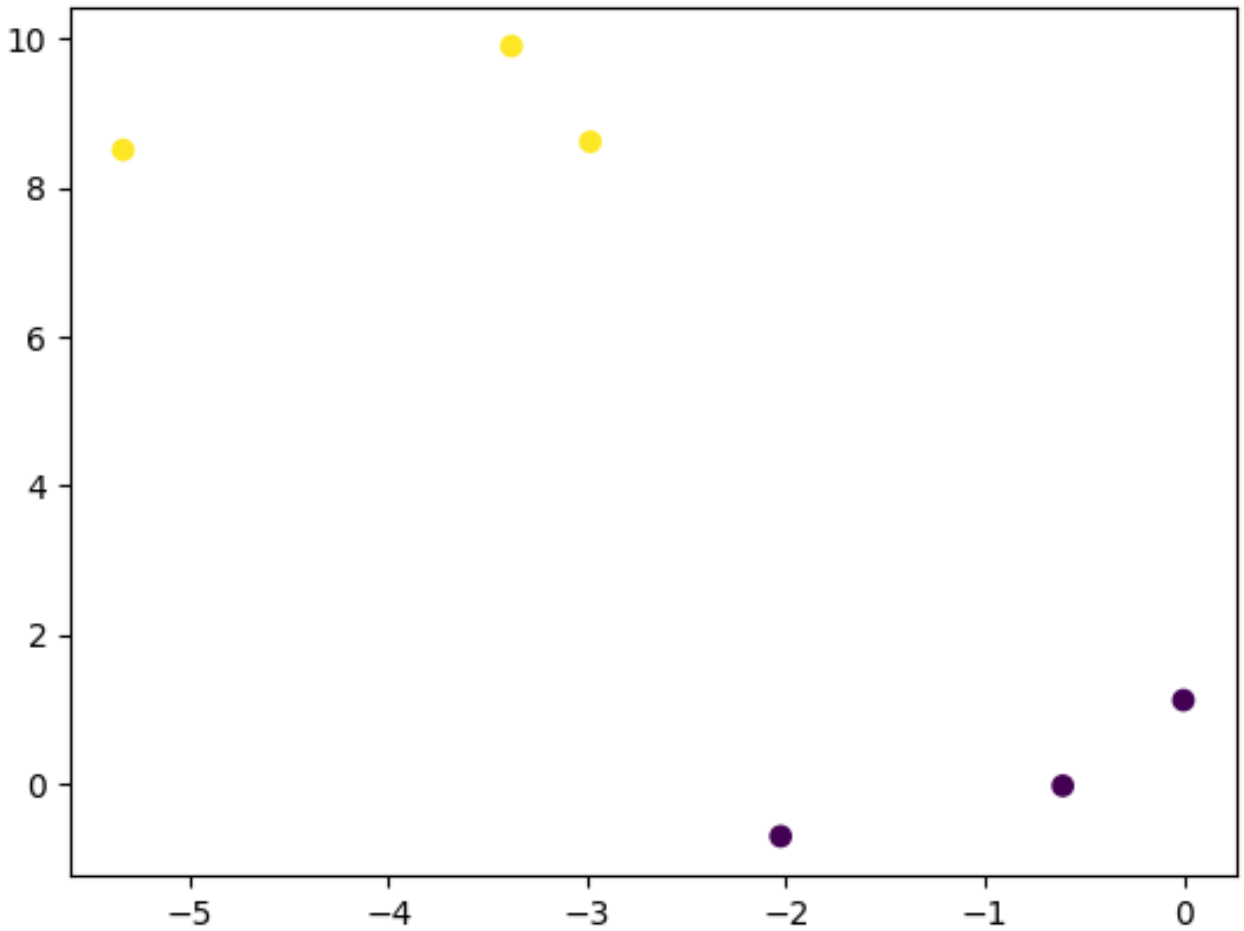
Entrada: *Datos etiquetados de entrenamiento* $\mathbf{X}$ *en coordenadas homogéneas*

Salida: *Vector de pesos* $\mathbf{w}$ *que define al clasificador*

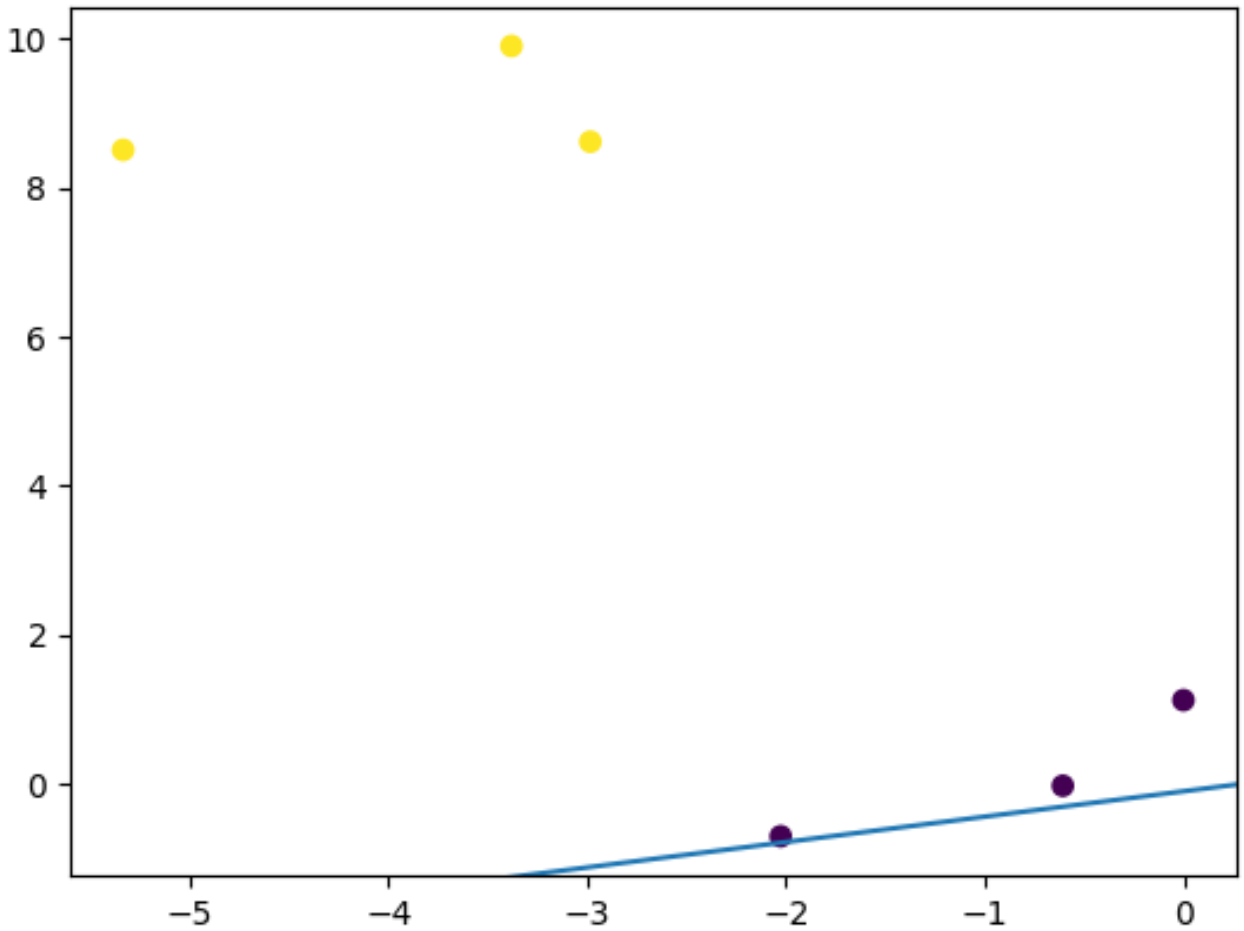
$\mathbf{w} = \mathbf{0}$ *#Otras inicializaciones son posibles*
$converge = \textbf{\textit{Falso}}$
mientras $converge == \textbf{\textit{Falso}}$:
    $converge = \textbf{\textit{Verdadero}}$
    para i **en** $|\mathbf{X}|$:
        si $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \leq 0$ **entonces**: #xi mal clasificado
            $\boldsymbol{w} = \boldsymbol{w} + y_i \eta \mathbf{x}_i$ #$0 < \eta \leq 1$ es la tasa de aprendizaje
            $converge = \textbf{\textit{Falso}}$
        fin
    fin
fin

El vector w es una combinación lineal de los vectores x_i

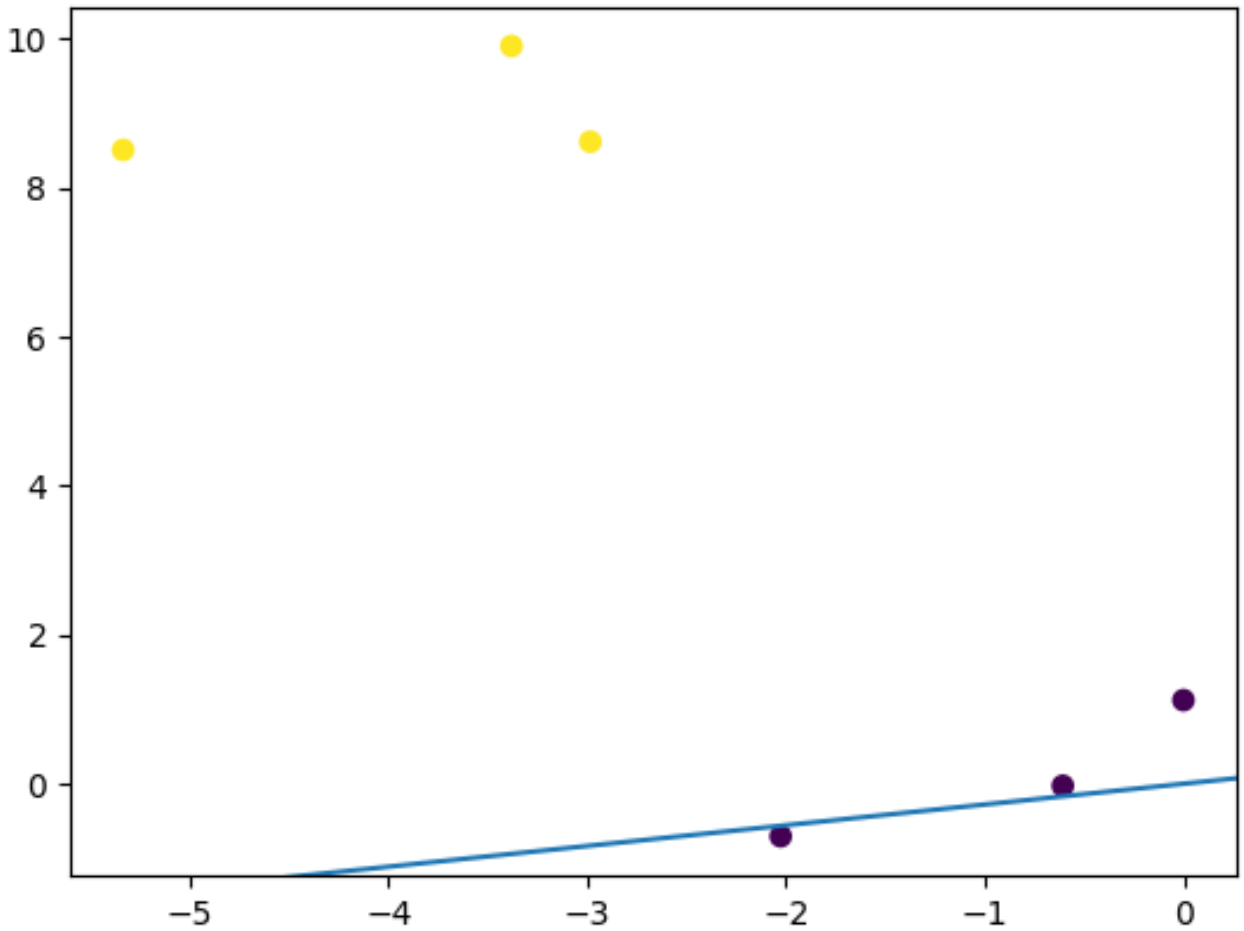
El algoritmo: un ejemplo



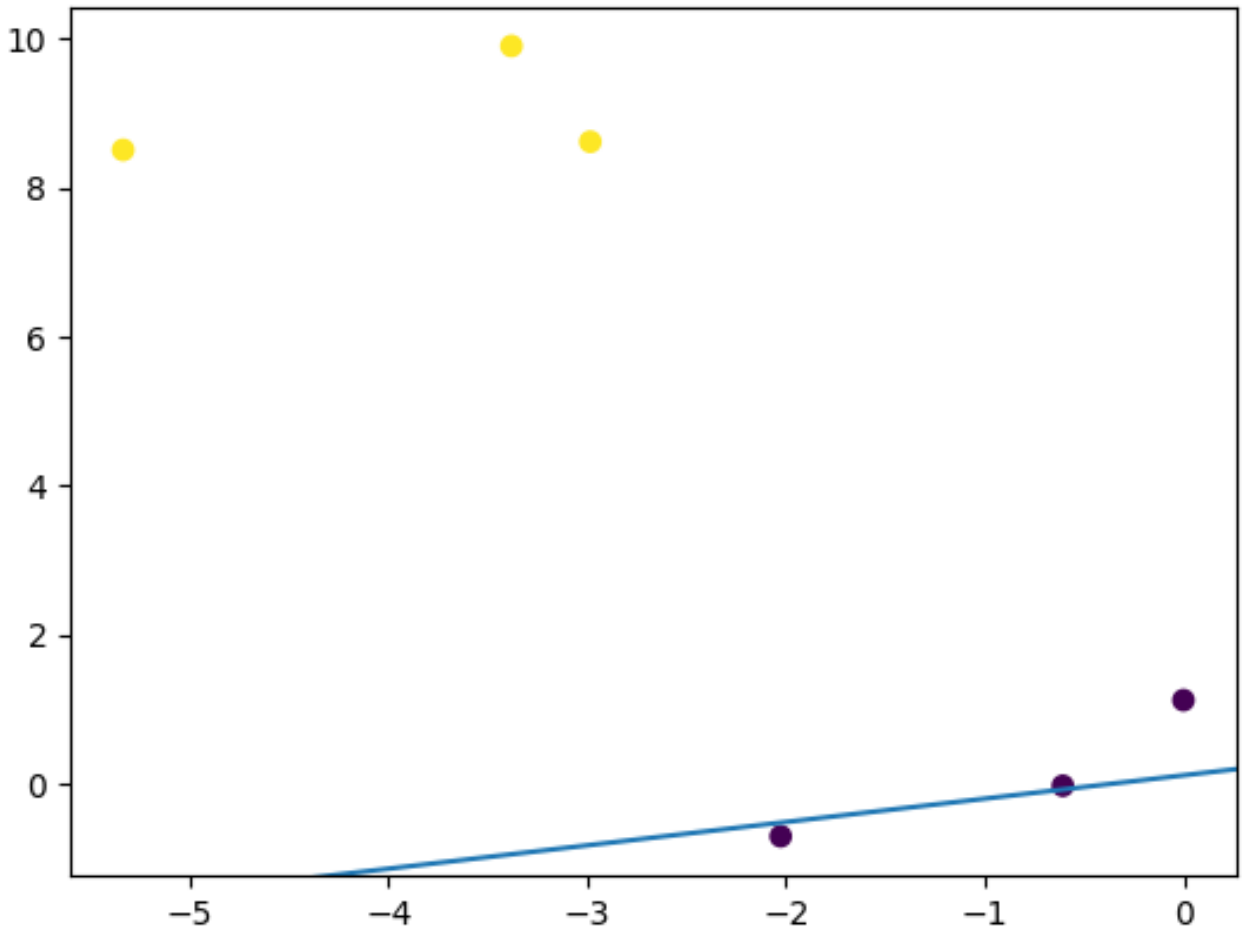
El algoritmo: un ejemplo



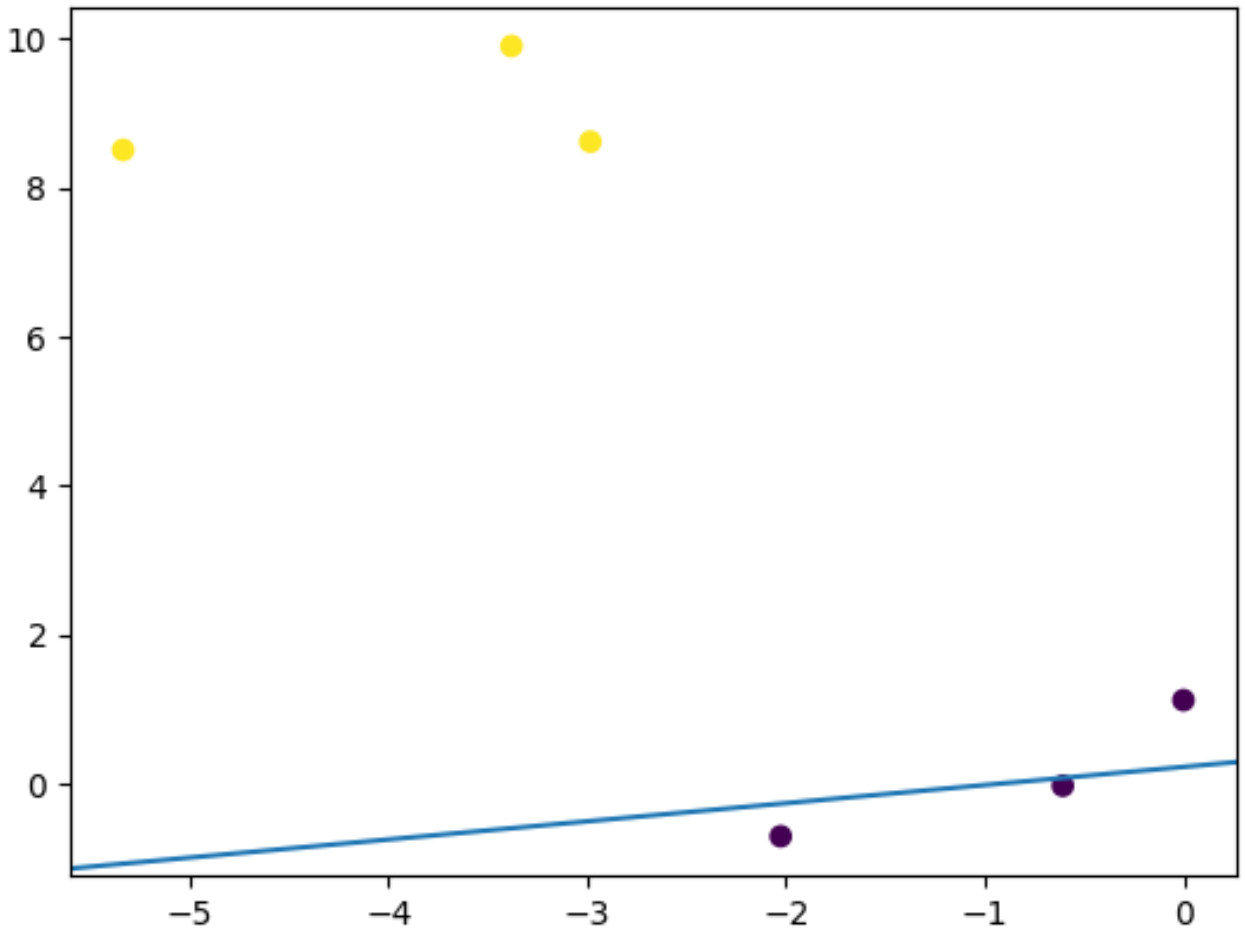
El algoritmo: un ejemplo



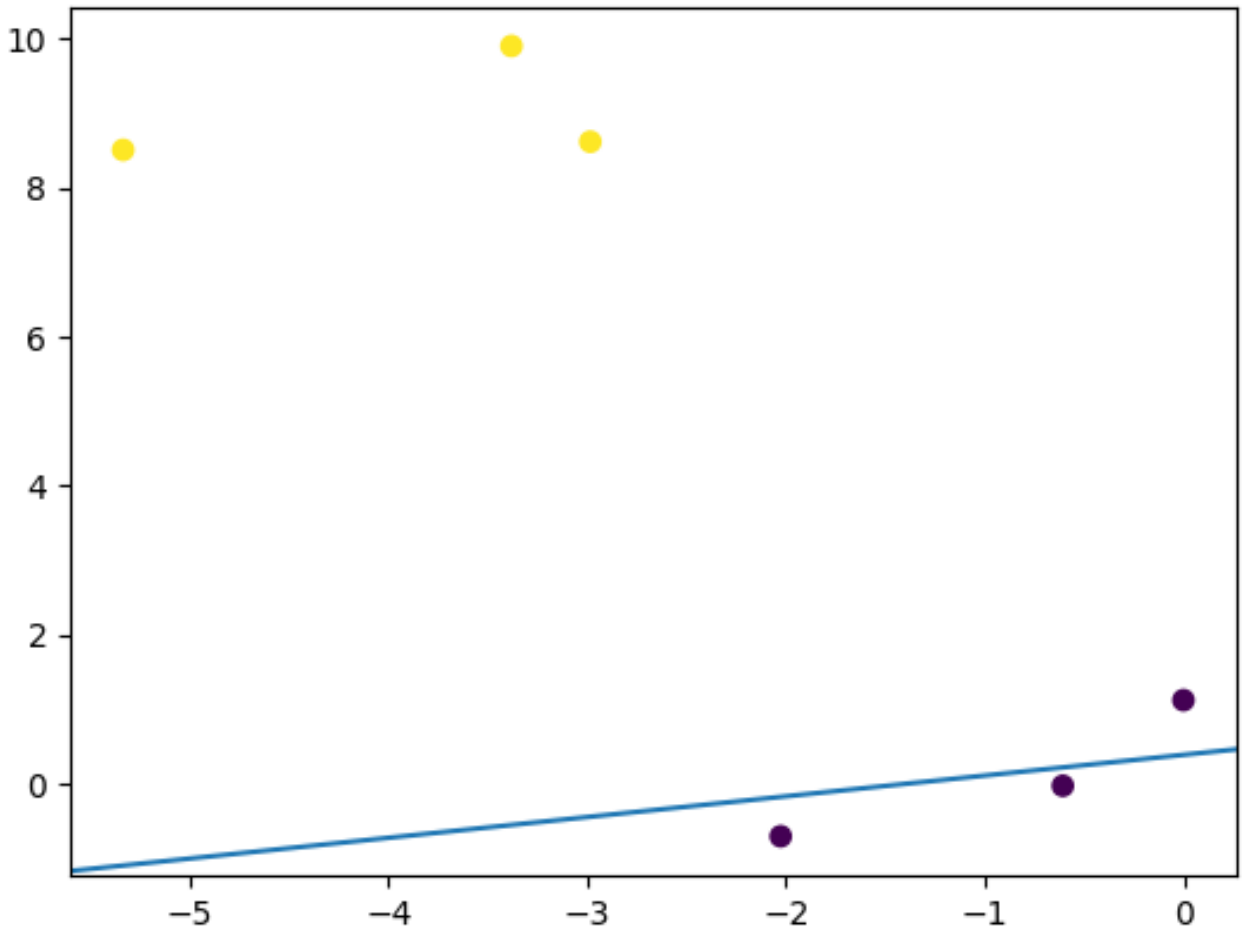
El algoritmo: un ejemplo



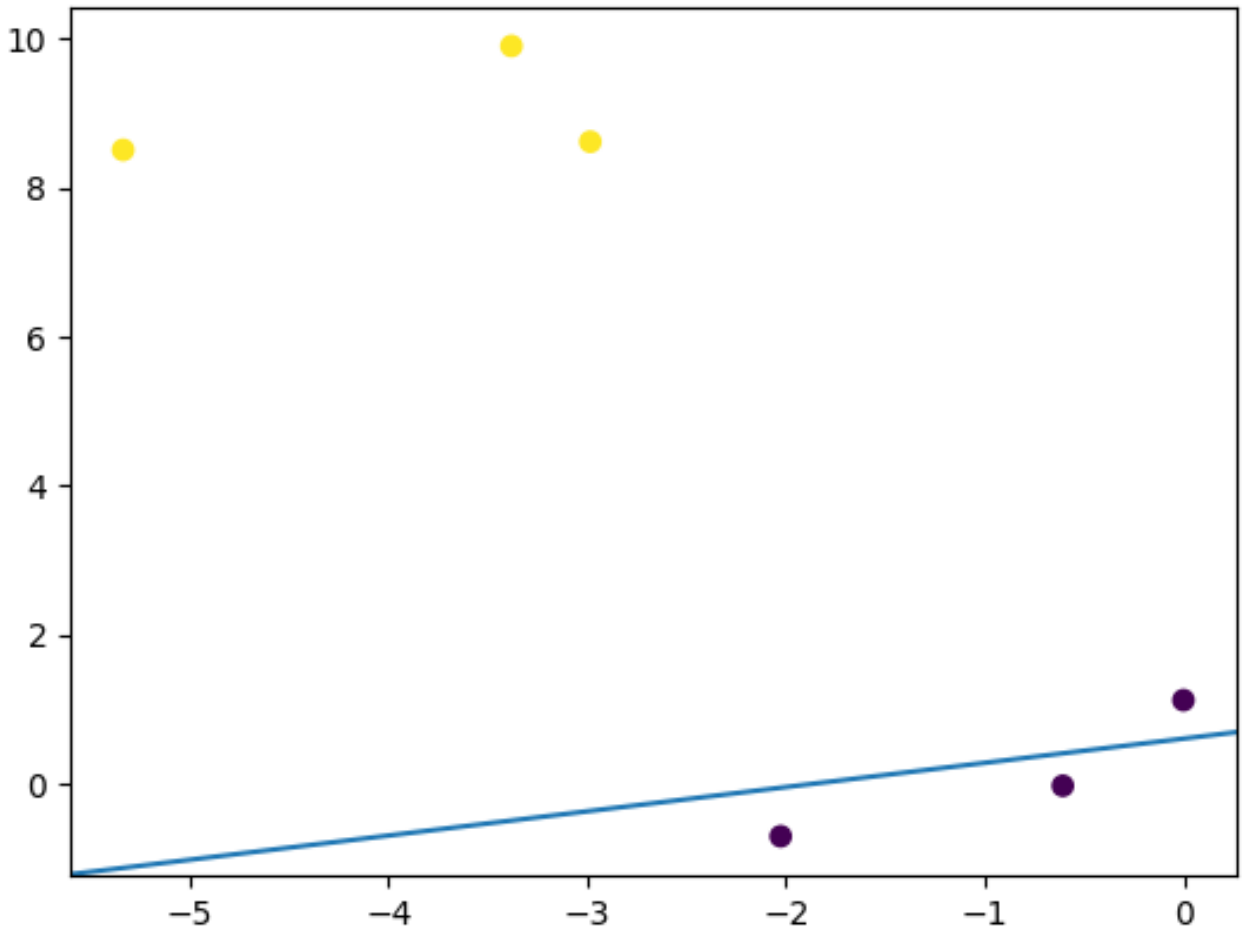
El algoritmo: un ejemplo



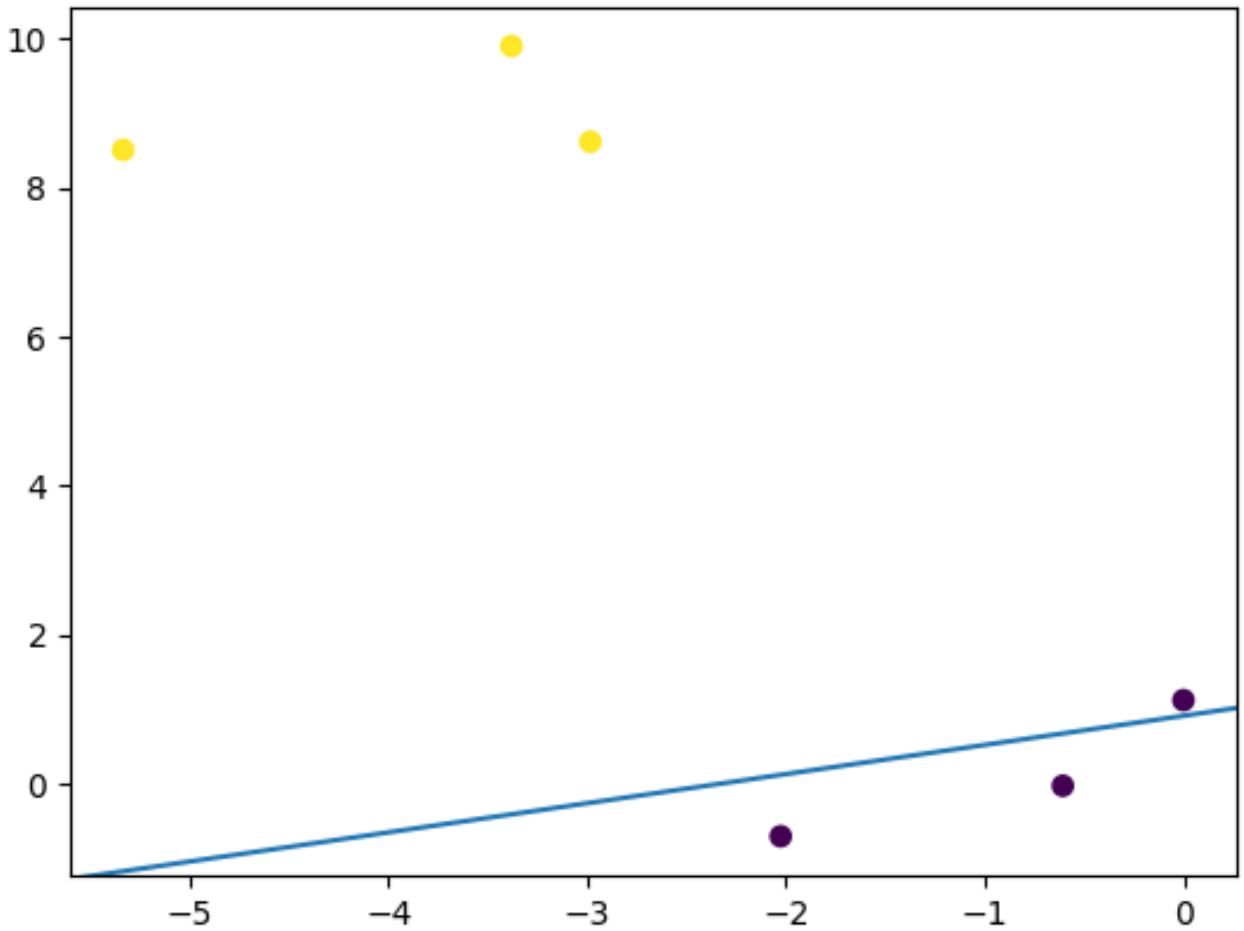
El algoritmo: un ejemplo



El algoritmo: un ejemplo



El algoritmo: un ejemplo



El algoritmo: un ejemplo

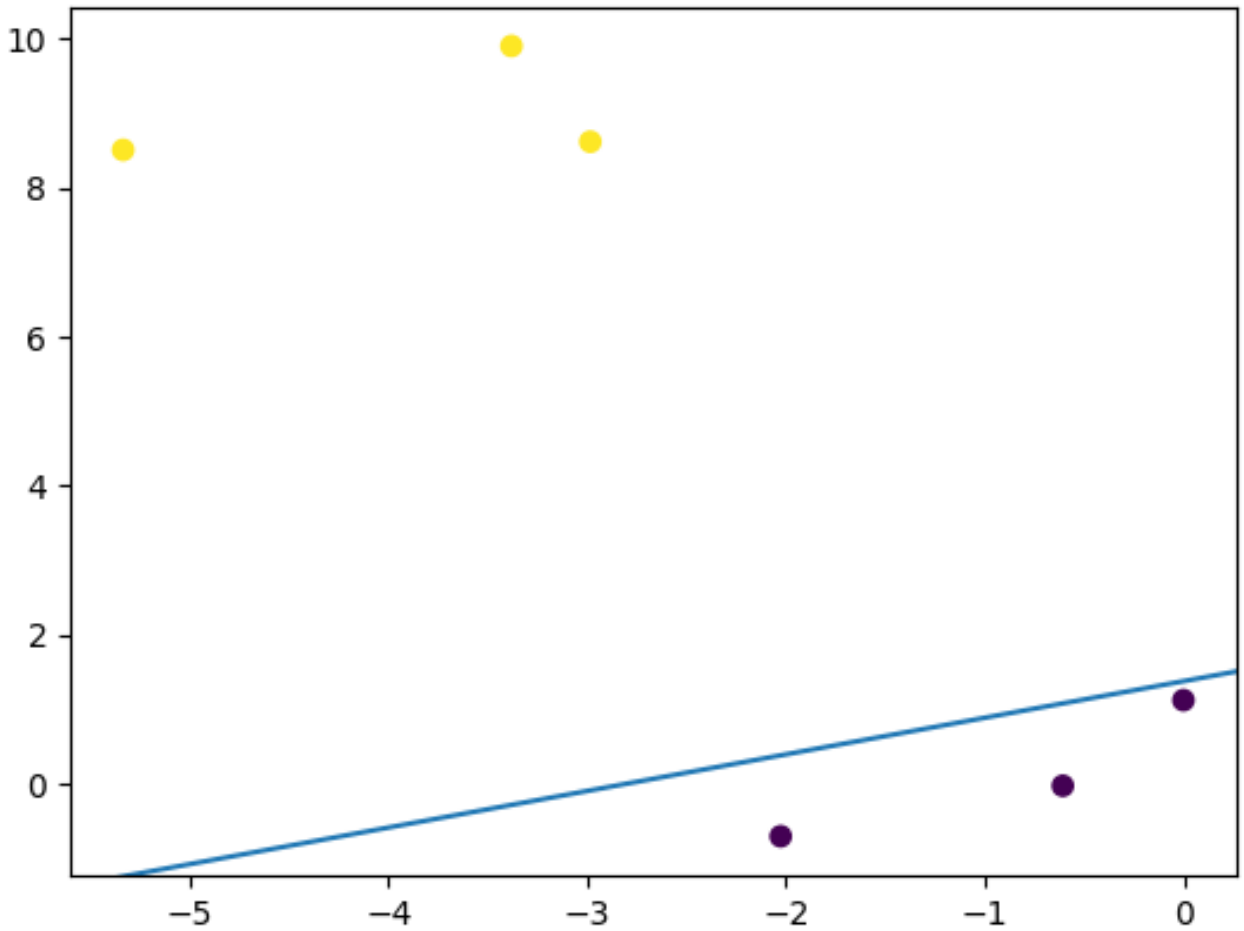


Table of Contents

El método dual

Observar que, al final del entrenamiento:

$$w = \alpha_1 y_1 x_1 + \alpha_2 y_2 x_2 + \cdots \alpha_N y_N x_N,$$

donde los α_i tienen que ver con el número de veces que el punto x_i fue mal clasificado.

Entonces, $w = \sum_{i=1}^{N} \alpha_i y_i x_i$ y por lo tanto

$$y_j = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \langle x_j, x_i \rangle\right)$$

Algoritmo del perceptron dual

$\boldsymbol{\alpha} \coloneqq (\alpha_1, \alpha_2, \cdots, \alpha_N) = 0$

$converge = \textbf{\textit{Falso}}$

mientras $converge == \textbf{\textit{Falso}}$:

  $converge = \textbf{\textit{Verdadero}}$

  para i **en** $|\mathbf{X}|$:

    toma $(\mathbf{x}_i, y_i)$ de los datos

    si $y_i \sum_{j=1}^{|\mathbf{X}|} \alpha_j y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \leq 0$ **entonces:** #xi mal clasificado

      $\alpha_i = \alpha_i + 1$

      $converge = \textbf{\textit{Falso}}$

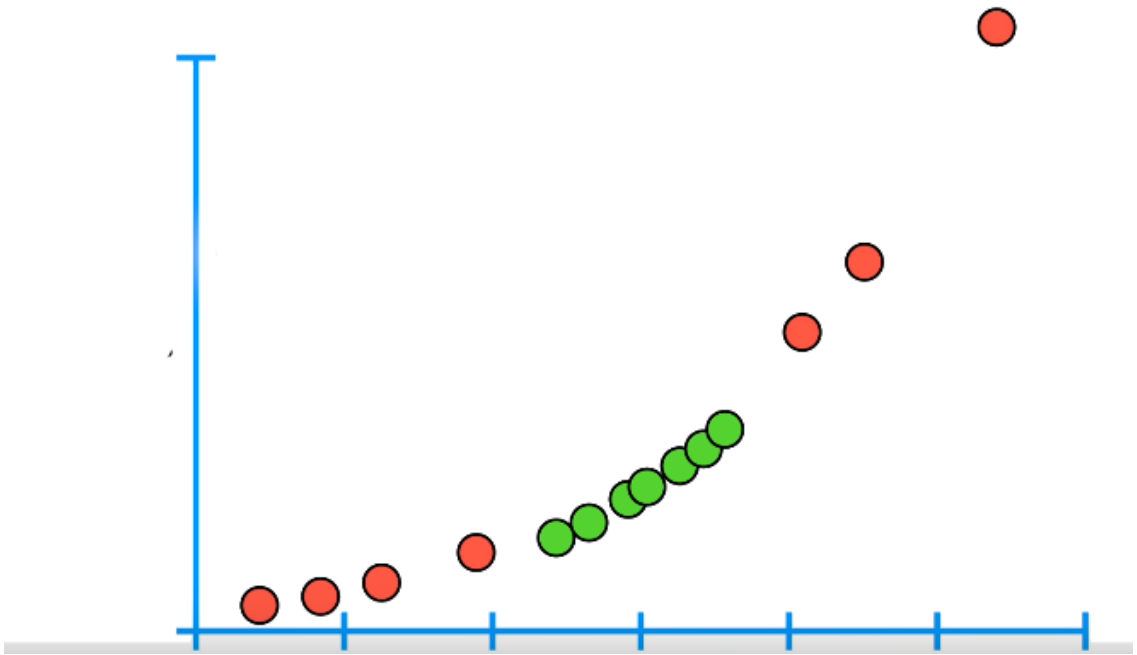
    fin

Table of Contents

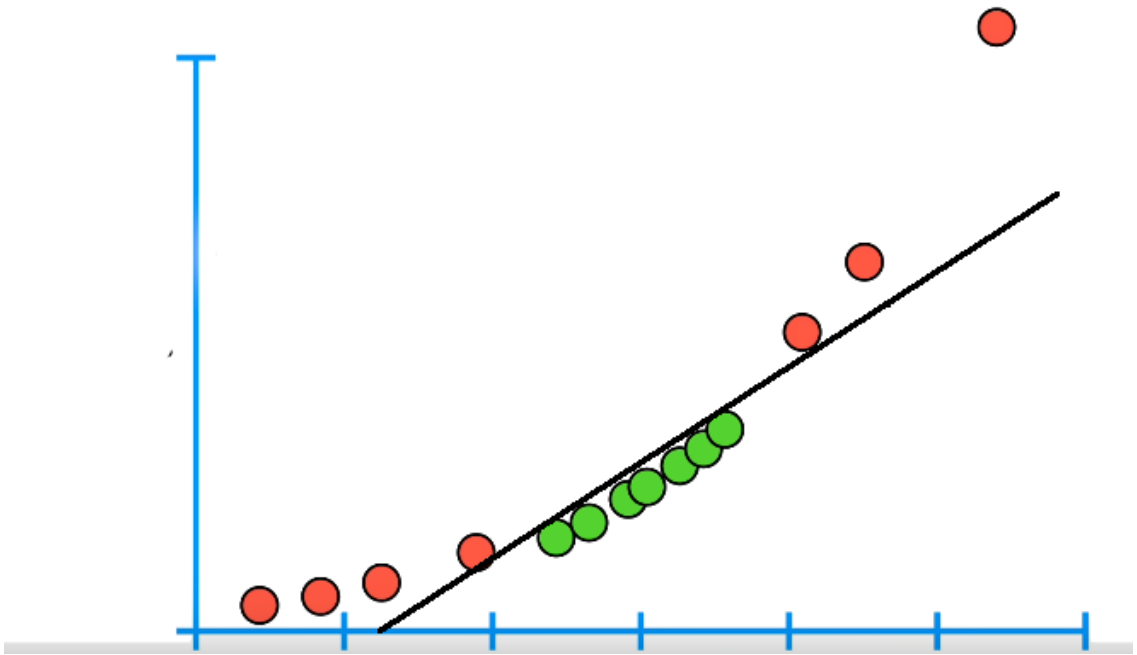
El truco del Kernel

Supongamos que queremos determinar qué dosis de un medicamento es la adecuada para tratar alguna condición. Coloreamos en verde las dosis que lograron una mejoría y en rojo las que no. ¿Cómo logramos una separación lineal de los datos?

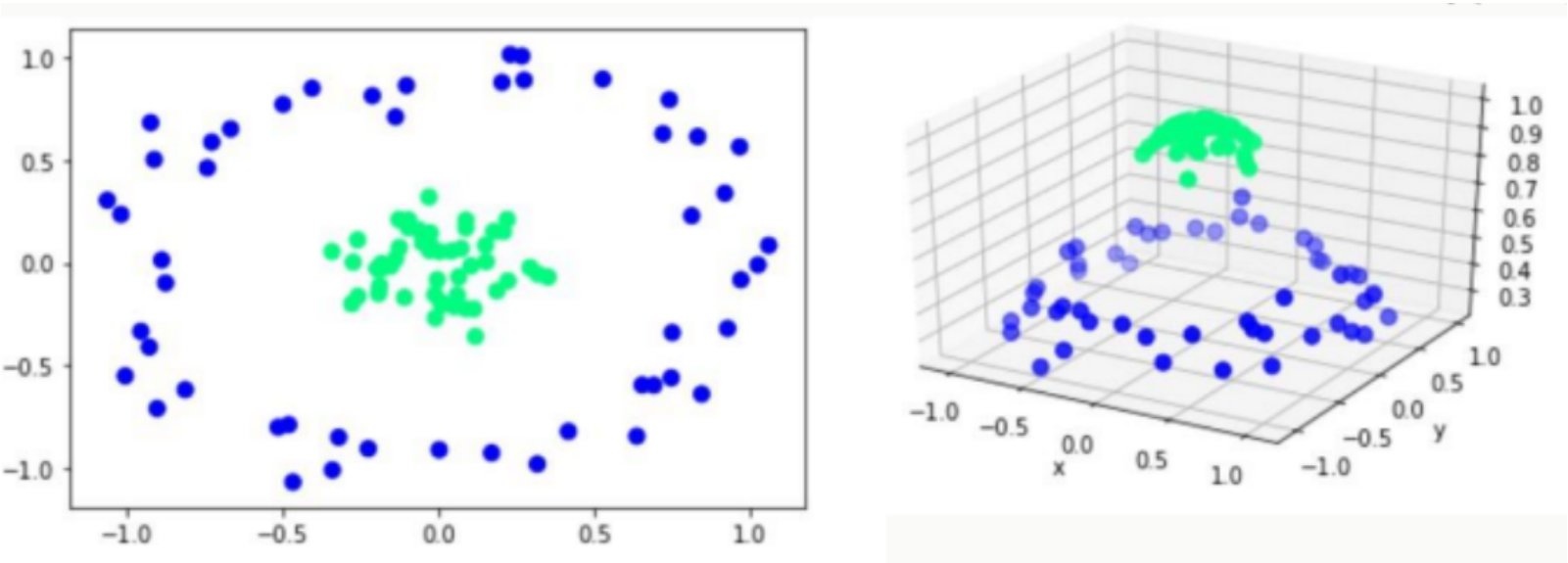
El truco del Kernel



El truco del Kernel



Ejemplo $z = x^2 + y^2$



Tipos de Kernel

Los cuatro principales kernels son:

Lineal	$\kappa(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$
Polinomial	$\kappa(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + r)^p,\ r \geq 0$
Gaussiano (Radial Basis Function)	$\kappa(\mathbf{x}, \mathbf{y}) = \mathrm{e}^{-\gamma \|\mathbf{x}-\mathbf{y}\|^2}$
Sigmoide	$\kappa(\mathbf{x}, \mathbf{y}) = \tanh\left(\langle \mathbf{x}, \mathbf{y} \rangle + r\right)$

Algoritmo Perceptron con Kernel

$\boldsymbol{\alpha} \coloneqq (\alpha_1, \alpha_2, \cdots, \alpha_N) = 0$; $\kappa(\mathbf{x}_i, \mathbf{x}_j)$: *entrada*

$converge = \textbf{\textit{Falso}}$

mientras $converge == \textbf{\textit{Falso}}$:

  $converge = \textbf{\textit{Verdadero}}$

  para i **en** $|\mathbf{X}|$:

    toma $(\mathbf{x}_i, y_i)$ de los datos

    si $y_i \sum_{j=1}^{|\mathbf{X}|} \alpha_j y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \leq 0$ **entonces**: #xi mal clasificado

      $\alpha_i = \alpha_i + 1$

      $converge = \textbf{\textit{Falso}}$