ML Primer - Resumen

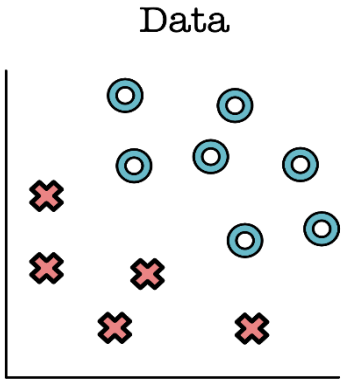
Alumno: Vargas Torres, Eros Aylthon

ML Primer

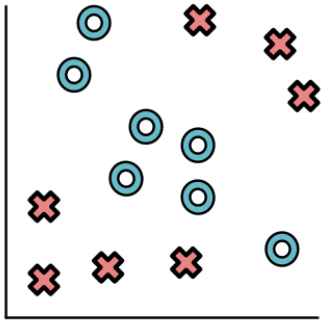
El aprendizaje automático (ML) es un área bien completa y desarrollada con diversos modelos de aprendizaje diferente, a continuación se mostrara los conceptos previos para el entendimiento del tema a abordar y ver la forma en cómo funciona una red neuronal.

Conjunto de datos

Partimos con un determinada cantidad de datos, en donde un dato estará representado por dos coordenadas (x_1, x_2) y además tendrá asignado una respectiva etiqueta 'y', que puede ser representada por una 'O' o 'X'.



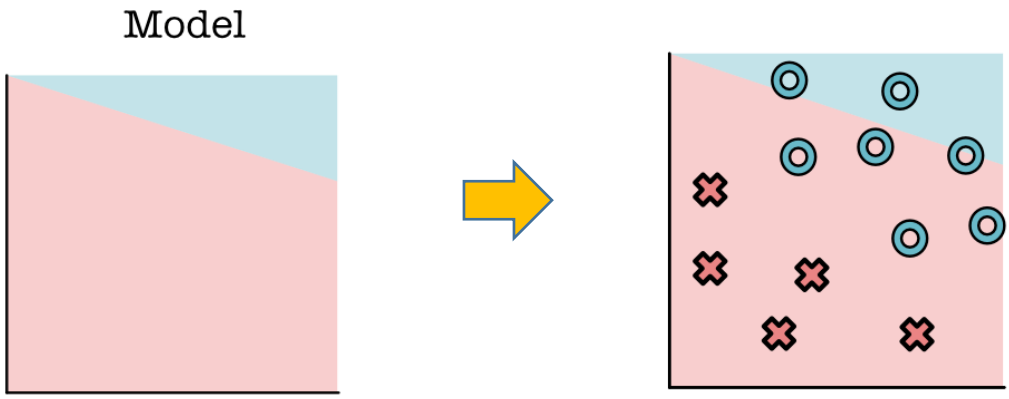
Algunas veces es posible encontrar conjunto de datos no tan simples, debido a que se dividen un poco más de lo común.



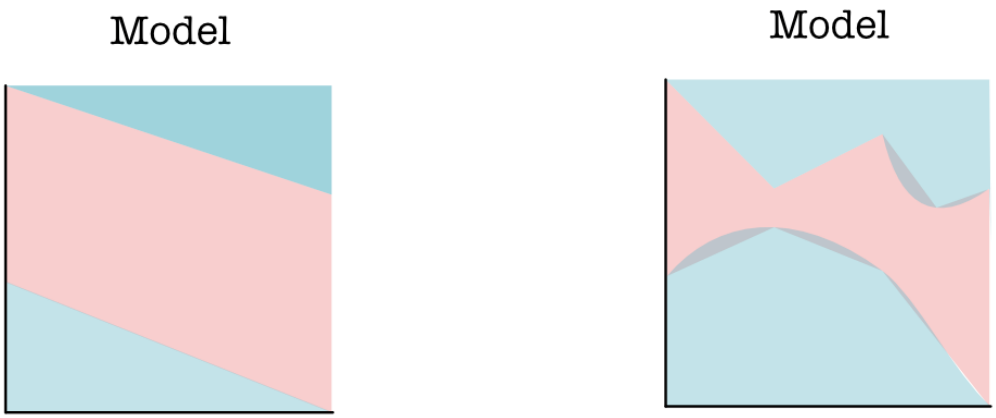
Modelo

Es necesario realizar un tipo de modelo que represente a los datos que tenemos. Un modelo se define como una función cuya finalidad es la de etiquetar a los puntos que tenemos como datos.

En el modelo 2D, podremos realizar un modelo por su respectivo límite de decisión. Además podríamos realizar una superposición de los datos simples que teníamos con el modelo, de esta manera poder observar que tan bien se ajusta el modelo, con los datos simples mencionados anteriormente.

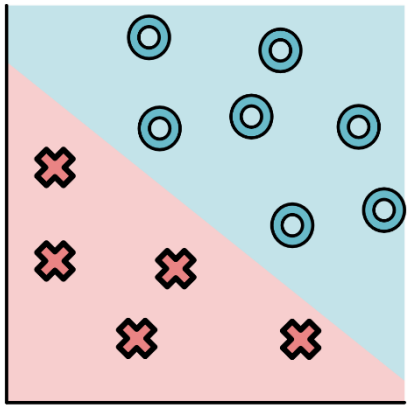


Podremos encontrar diversos modelos, que se realizan en base al posicionamiento de los datos de entrenamiento que estamos analizando.



La clase de modelo identifica la forma más general de aquellos modelos que quiera explorar. Debido a que nosotros no conocemos la distribución del conjunto de datos, entonces proporcionaremos una clase de funciones para que nuestro sistema pueda explorarlos, con el fin de encontrar el mejor modelo de dicha clase, a este último procedimiento se le denomina aprendiza automático.

La primera clase de modelo que tomamos en cuenta son los modelos lineales (modelos que separan el espacio, mediante una línea recta). El modelo presentado inicialmente es posible verlo mejor intuitivamente de esta manera:

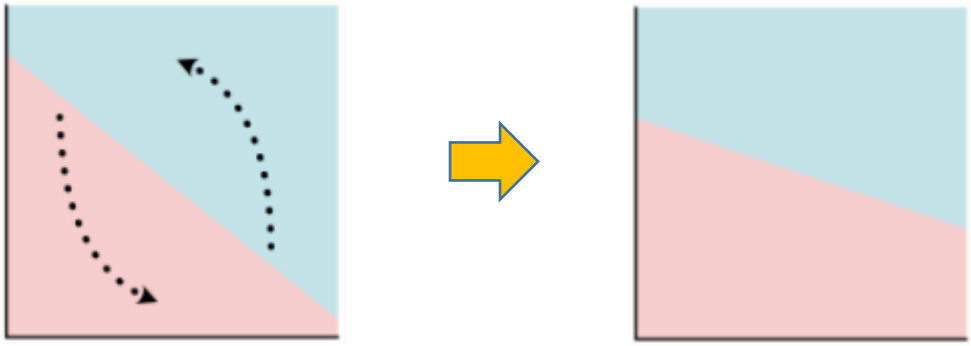


Parámetros

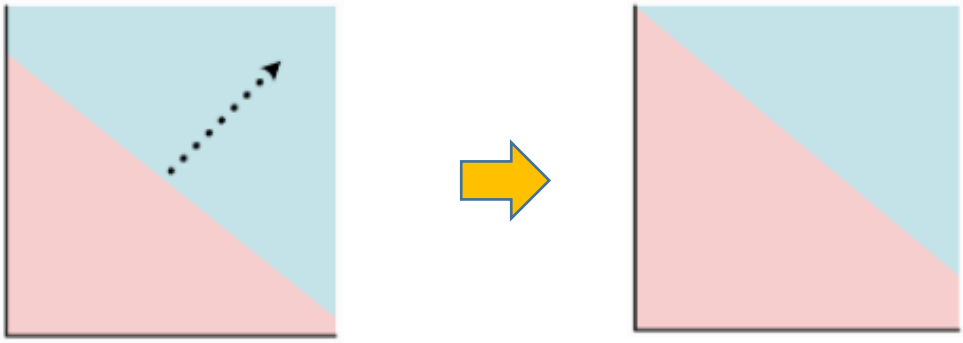
Cuando determinamos nuestra clase modelo, será necesario moverse entre los modelos de dicha clase.

En los modelos lineales, podremos realizar los siguientes movimientos:

- Rotando la pendiente:



- Cambiando el corte del separador:



Los parámetros se definen como un conjunto de valores numéricos que determinan las decisiones de algún modelo, dichos valores son decisivos para acopiar la forma de actuar del modelo y además son de suma importancia para realizar predicciones sobre un conjunto de datos analizados.

El modelo lineal se representa matemáticamente de la siguiente manera:

$$m(x_1, x_2; w_1, w_2, b) = x_1.w_1 + x_2.w_2 + b$$

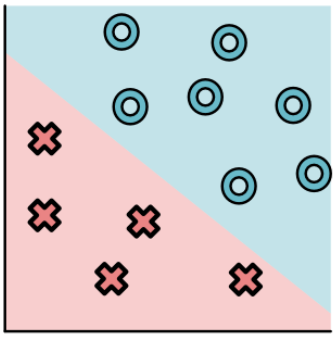
Donde:

- w_1, w_2, b = parámetros
- x_1, x_2 = Puntos de entrada

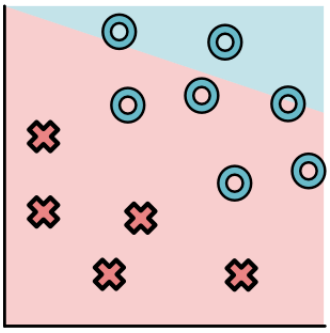
Pérdida

Es posible clasificar los modelos como buenos o malos, a raíz de nuestros datos que tenemos:

- Modelo bueno, que es el que no comete ningún error.



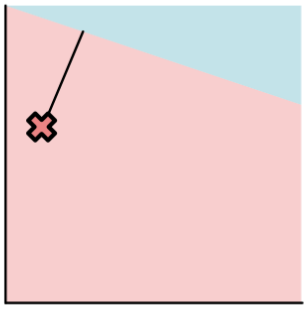
- Modelo malo, que es el que comete diversos errores.



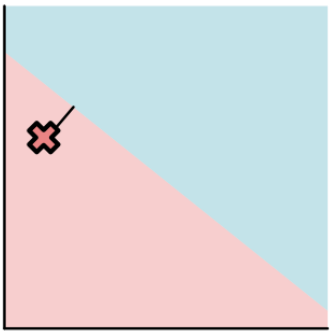
Haremos uso de una función de perdida que medirá que tan mal está el modelo, y como podremos intuir, el modelo que hace que la función de perdida sea la más pequeña posible, quiere decir que dicho modelo es bueno; dicha función estará basada en la dirección y distancia de los puntos que representa a los datos hasta la línea que representa el límite de decisión.

Para que sea más simple, analizaremos un solo punto en múltiples modelos, en donde podrá clasificarse la siguiente manera:

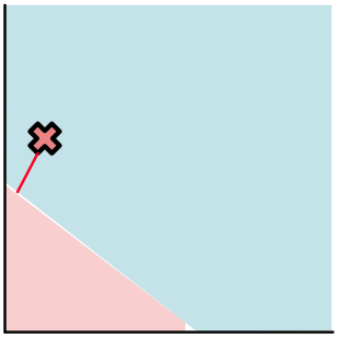
- Bueno: El punto está en el lado correcto y muy lejos de la línea.



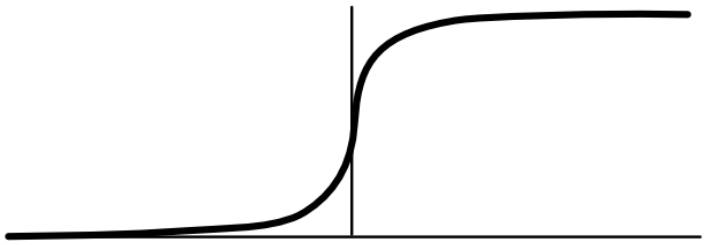
- Preocupante: El punto está en el lado correcto pero muy cerca de la línea.



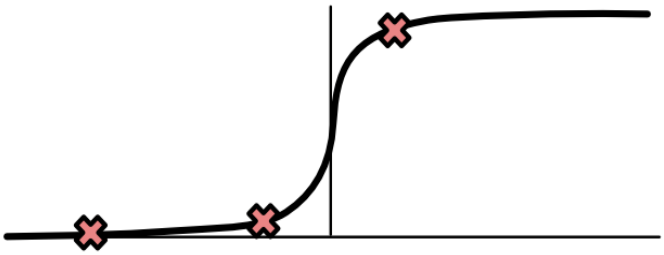
- Malo: El punto está en el lado incorrecto.



La función con la que se centra para trabajar, es la función sigmoidea, en donde podemos observar que para entradas de fuertes negativos, va hacia cero, pero para fuertes positivas, va hacia 1, en la parte del medio podemos visualizar una curva suave en forma de S.



A continuación vemos un ejemplo, en donde vemos que han aterrizado 3 puntos en determinadas posiciones de la función, si deseamos determinar la pérdida total de algún modelo, entonces dicho valor de perdida será igual al producto de cada perdida de dichos puntos.



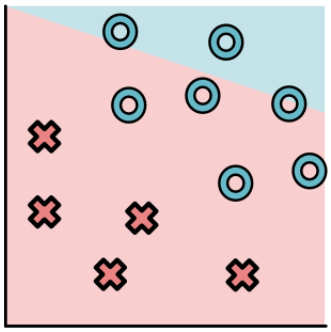
Parámetros de ajustes

Nos permitirá identificar un buen modelo a raíz de la función de perdida. Debido a su dificultad para trabajar con una gran cantidad de parámetros, será necesario hacer uso de la librería "MiniTorch", ya que con ello, mediante una codificación minuciosa, podremos estructurar un marco del ajuste de los parámetros para la clasificación supervisada, de una forma eficaz y automática.

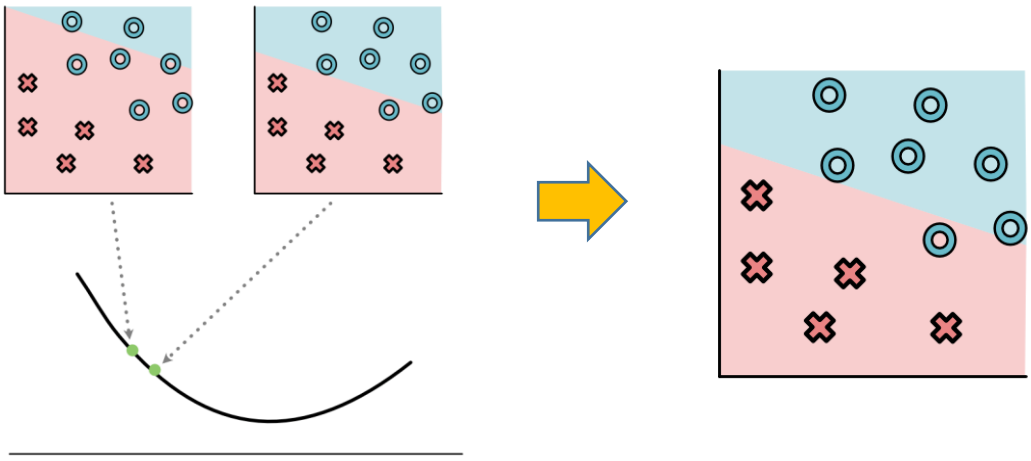
La forma de ajuste que se centra dicha biblioteca, es la de "descenso de gradiente", que trabaja de la siguiente manera:

- Determina la función de perdida (L) , para los datos que tenemos con los parámetros
- Observaremos el pequeño cambio de la perdida, en cada uno de los parámetros
- Actualiza dichos parámetros, con un leve cambio de dirección con la que reduzca un poco más la perdida.

Si analizamos el modelo malo, mostrado anteriormente, sabemos que dicho modelo tiene una gran pérdida, es por ello que debemos considerar en cambiar el parámetro de "rotar la pendiente" y "cambiar el corte del separador", y así encontrar un mejor modelo.



Pero vemos que si nos centramos únicamente en el parámetro que controla el corte separador, y la bajamos un poco, podremos observar que la perdida disminuirá considerablemente, y de esta manera conseguir un mejor modelo.



Vemos que para un pequeño problema, solo era moverlo un poco y observarlo, pero como hemos dicho anteriormente, habrá problemas que contengan una gran cantidad de parámetros, y esto nos tomara muchísimo tiempo.

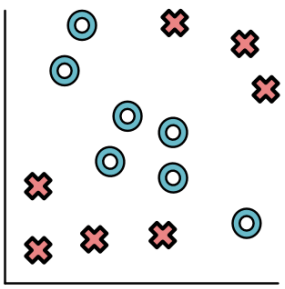
De manera más eficiente, podríamos usar el cálculo y conseguir la derivada de la función de perdida con respecto al determinado parámetro, lo cual nos dirá como debemos cambiar el parámetro de tal manera que se adapte a cualquier perdida, por ello debemos tomar a dicha derivada de manera correcta, incluso sería mejor si tomásemos la gradiente (conjunto de derivadas) de manera eficiente, ya que nos indicaría hacia qué dirección deberían moverse todos.

Redes neuronales

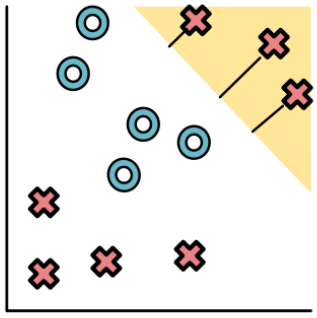
Hasta ahora hemos visto que la clase de modelo lineal nos posibilita a encontrar buenos ajustes a los datos, sin embargo para modelos no lineales, además que no se puede separar linealmente, aún sigue siendo un problema.

Las redes neuronales, pueden ser usadas para precisar una más grande cantidad de separadores, en el que divide la clasificación en 2 o más etapas, donde podemos ver que en cada etapa se encarga de remodelar los datos que tenemos en nuevos puntos, y en la etapa final es como un clasificador lineal sobre el punto transformado.

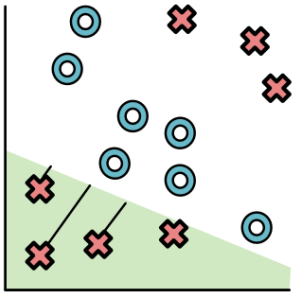
Si analizamos para datos que se dividen en múltiples segmentos, como el siguiente modelo:



Con una red neuronal podremos generar un primeramente un separador (de color amarillo) con la finalidad de separar los puntos rojos en la parte superior derecha (respecto al lector).

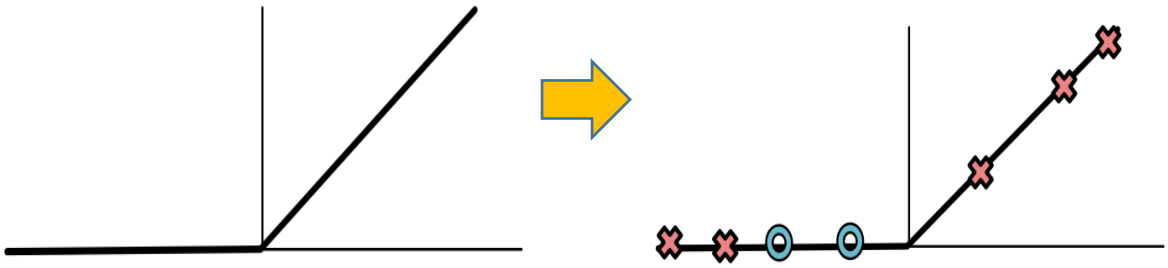


Ahora podremos producir un separador de color verde para los puntos rojos que están en la parte inferior izquierda (respecto al lector).

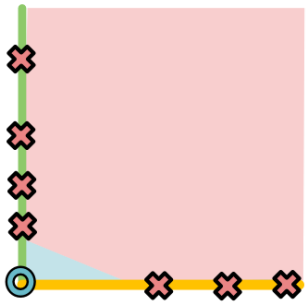


La red neuronal posibilita la transformación de puntos en función de la distancia desde los separadores. Es posible trabajar con cualquier función para realizar la transformación, de forma ideal, dicha función haría que los puntos en la zona amarilla y verdes, fueran altos y los otros puntos bajos, y esto último se puede realizar mediante la función ReLU("eliminar valores por debajo de cero").

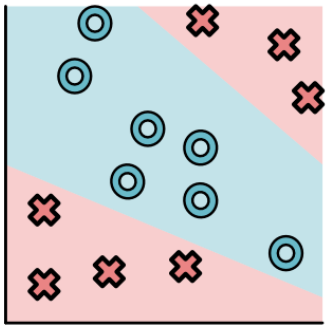
La función ReLU genera estos valores para el separador amarillo:



Los X que estaban en la parte superior se ubican ahora dentro la función, en la parte positiva, mientras que los 'X' y 'O' que estaban en su parte inferior, se ubican en 0. Entonces x_1 corresponderá a los puntos de la zona amarilla, mientras que x_2 corresponderá a los puntos de la zona verde, como vemos a continuación:



Si regresamos al modelo original, podremos ver que con este proceso pudimos dos separadores para los datos que tenemos.



Podemos expresar matemáticamente los datos transformados (denotados como h_1, h_2) obtenidos a raíz de la aplicación de separadores con distintos parámetros a nuestros datos que tuvimos inicialmente:

$$h_1 = ReLU(x_1.w_1^0 + x_2.w_2^0 + b^0)$$

$$h_2 = ReLU(x_1.w_1^0 + x_2.w_2^1 + b^1)$$

$$m(x_1, x_2) = h_1.w_1 + h_2.w_2 + b$$

Donde:

- $w_1, w_2, w_1^0, w_2^0, w_1^1, w_2^1, b, b^0, b^1 =$ parámetros

Finalmente agregar que es posible flexibilizar modelos a medida de que se aumente los parámetros a ajustar.

Esta red neuronal será el punto de partida para los primeros modelos, y si quisiéramos adaptarlo eficientemente será necesario crear una infraestructura de sistemas, y cuando esta última sea posible de tenerla, podremos ser capaces de adaptar la gran mayoría de modelos modernos de redes neuronales.