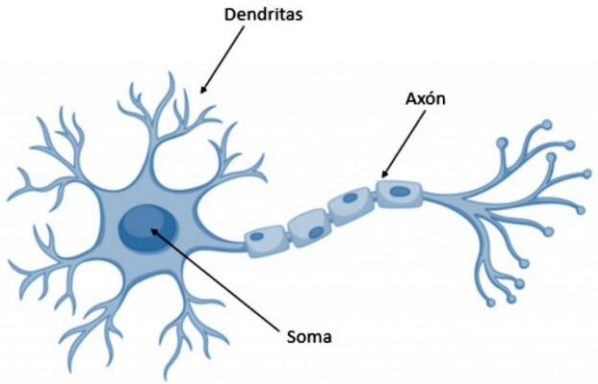
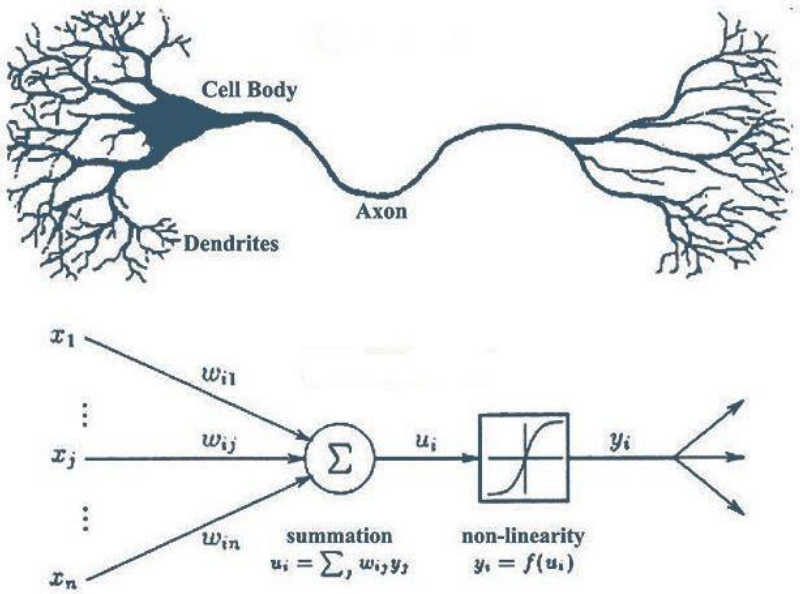
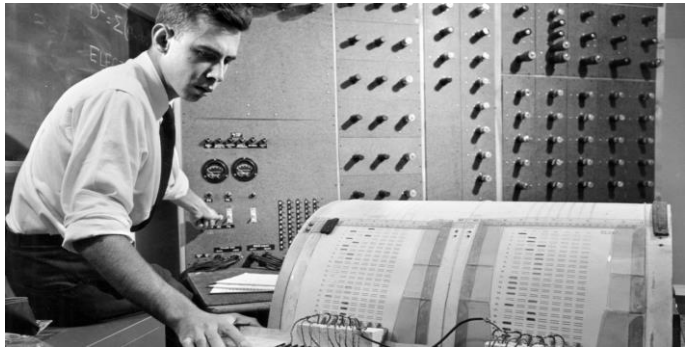
Sesión 4: Introducción a las Redes Neuronales.



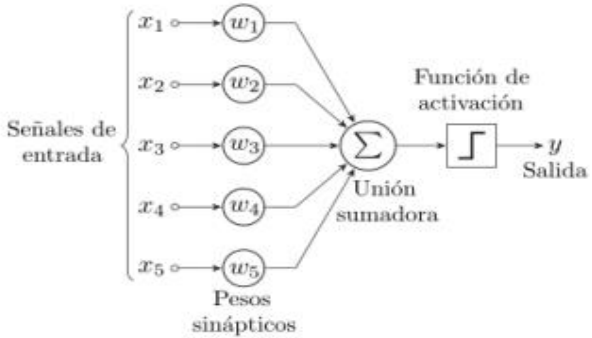


Historia



Las redes neuronales datan desde el año 1958 con Frank Rosenblatt quien es el creador del método del **Perceptrón** o también conocido como **Perceptrón Simple.**

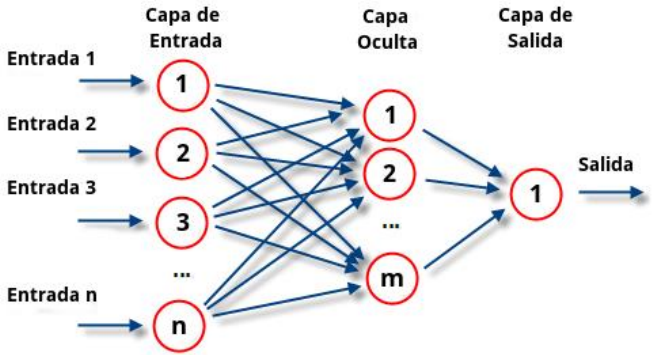
Se trata del modelo más sencillo de redes neuronales artificiales, ya que consta de una sola capa de neuronas con una única salida **y**



Multi-Layer Perceptron

Multi-Layer Perceptron: Es una red de perceptrones que consta de diferentes capas donde cada una de ellas tiene un nodo como bias (sesgo).

Proceso:
Se alimentan los datos de input en la capa de entrada y se toma el output de la capa de salida. Podemos **aumentar el número de Layers en la capa oculta tanto como queramos**, para hacer que el modelo sea más complejo de acuerdo con nuestra tarea.



¿Qué es una función de activación?

Tenemos diversas funciones de activación disponibles en Keras:

- Sigmoid function
- Softmax function
- Tanh tangent function
- ReLu function

Función de activación "Softmax"

La función Softmax transforma las salidas a una representación en forma de probabilidades, de tal manera que el sumatorio de todas las probabilidades de las salidas de 1.

Características:

- Se utiliza cuando queremos tener una representación en forma de probabilidades
- Esta acotada entre 0 y 1
- Muy diferenciable
- Se utiliza para normalizar tipo multiclase
- Buen rendimiento en las últimas capas

Función de activación "Sigmoide"

La función de activación sigmoide transforma los valores introducidos a una escala (0,1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a 0.

Características:

- Satura y mata el gradiente
- Lenta convergencia
- No esta centrada en el cero
- Esta acotada entre 0 y 1
- Buen rendimiento en la última capa

Función de activación "ReLU – Rectified Lineal Unit"

La función ReLU transforma los valores introducidos anulando los valores negativos y dejando los positivos tal y como entran.

Características:

- Activación Sparse – solo se activa si son positivos
- No está acotada
- Se pueden morir demasiadas neuronas
- Se comportan bien con imágenes
- Buen desempeño en redes convolucionales

Función de activación "Tangente hiperbólica"

La función tangente hiperbólica transforma los valores introducidos a una escala (-1, 1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a -1.

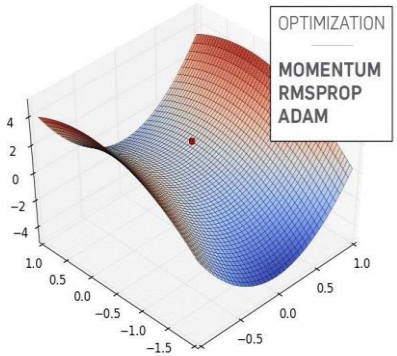
Características:

- Muy similar a la sigmoide
- Satura y mata el gradiente
- Lenta convergencia
- Centrada en 0
- Esta acotada entre -1 y 1
- Se utiliza para decidir entre una opción y la contraria
- Buen desempeño en redes recurrentes

Optimizadores

Momentum: Obtiene las gradientes del vector de momento, y actualiza los pesos simplemente añadiendolo a la ecuación. En otras palabras, este optimizador es para acelerar y no para hacer más rápido el cálculo. El parámetro **beta es un símil de la fricción en la aceleración**.

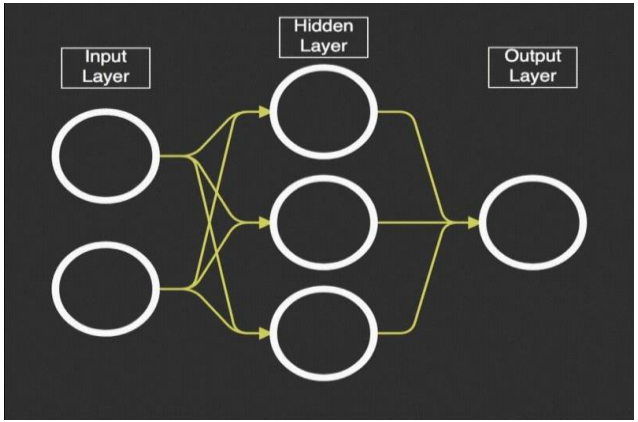
Adam: El optimizador de Adam es una combinación de los algoritmos Momentum (acelera la búsqueda del valor mínimo) y RMSDrop (impide las oscilaciones).



Back Propagation

La idea de backpropagation es ver junto con derivadas parciales qué tanto afectan las variables a los errores de los outputs.

Estas gradientes se propagan hacia todas las capas anteriores y se puede ver y corregir los pesos de la variable que más ha influido en el error.



El método emplea un ciclo propagación – adaptación de dos fases. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas siguientes de la red, hasta generar una salida.

La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.

https://www.youtube.com/watch?v=eNIqz_noix8&ab_channel=DotCSV (6:02-9:51)

Learning Rate

Es un hiperparámetro que controla cuánto cambiar el modelo en respuesta al error estimado cada vez que se actualizan los pesos del modelo (determina el tamaño del paso en cada iteración)

Elegir el learning rate es un desafío, ya que un valor demasiado pequeño puede resultar en un largo proceso de entrenamiento que podría atascarse, mientras que un valor demasiado grande puede resultar en aprender un conjunto de pesos subóptimo demasiado rápido o un proceso de entrenamiento inestable.

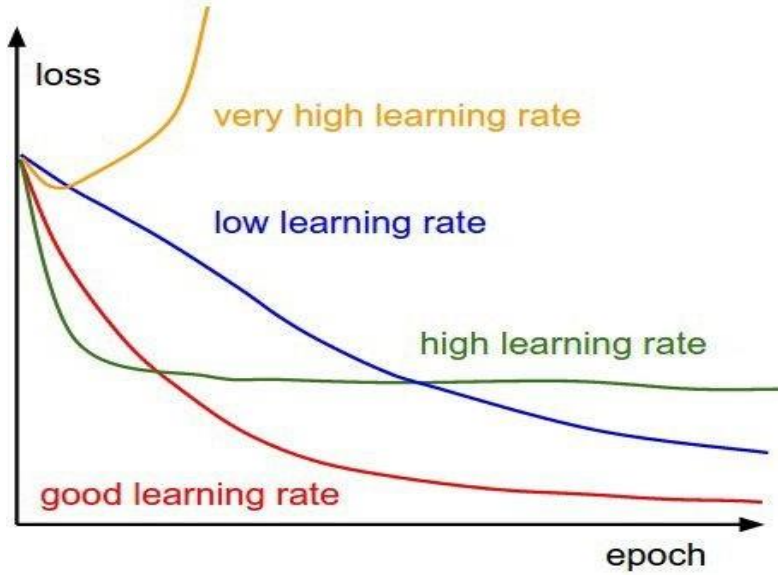
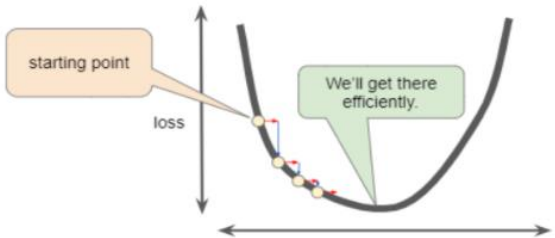


Figura 8. La tasa de aprendizaje es la correcta.

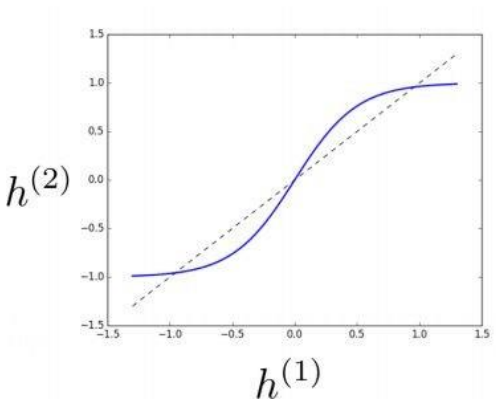


Vanishing and Exploding Gradient

Vanishing

Las gradientes resultan ser pequeñas en cierto punto del entrenamiento.

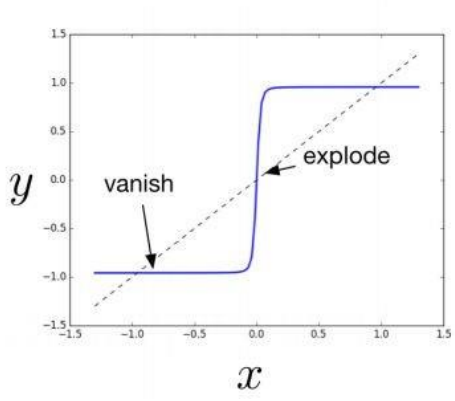
Problema: Los pesos pueden llegar a variar tan poco que llega quedarse en un estado neutral.



Exploding

Las gradientes resultan ser muy grandes y el algoritmo diverge.

Problema: Los pesos varían de tal manera que resulta que el modelo sea inestable durante el training.

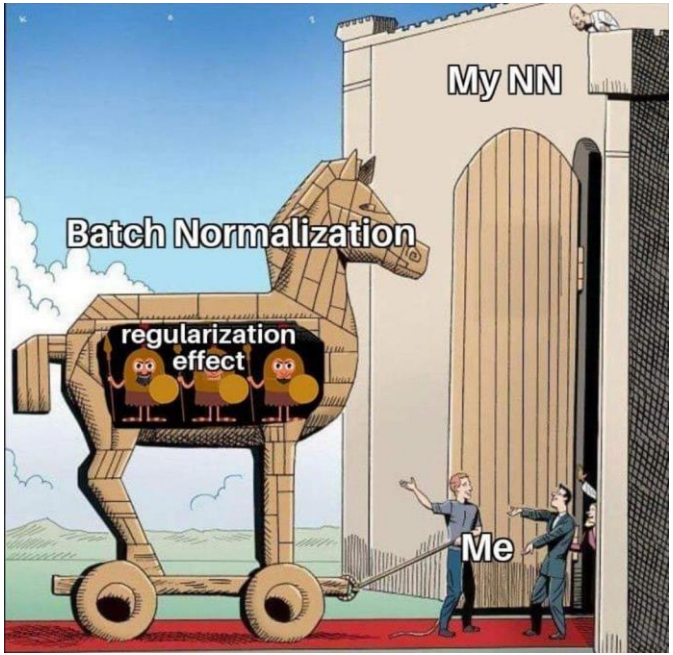


Batch Normalization

Dado los problemas de vanishing y exploits se recomienda la utilización del método Batch Normalization en el Deep Neural Network.

Esto Keras te lo permite fácilmente. **BN se ha convertido en la regla más usada dentro de Deep Learning**, tanto así que se sobreentiende que se usa en cada arquitectura.

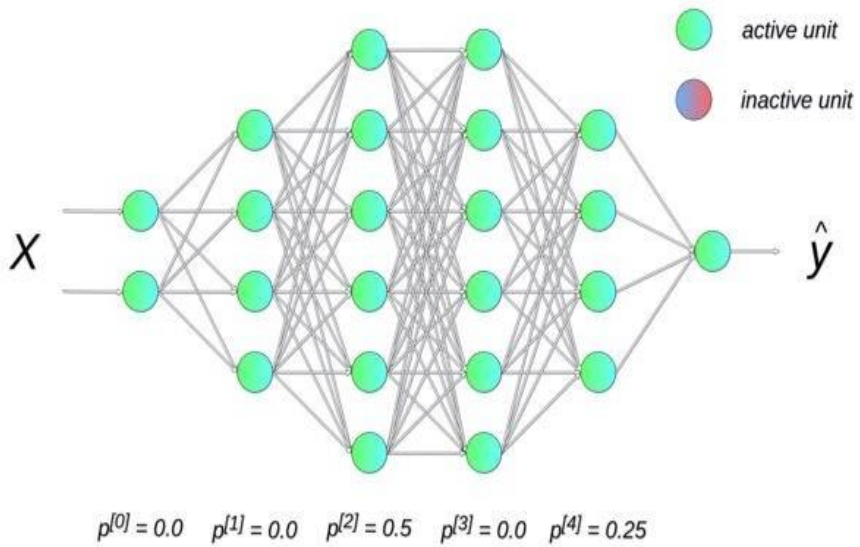
Batch Normalization ayuda a corregir el entrenamiento lento o inestable.



DropOut

Es una de las técnicas que **más funcionan dentro de una red neuronal** y consiste en dropear nodos temporalmente en cada iteración al momento de entrenar. Estos nodos se dropean con una probabilidad p, que usualmente es en 50%.

Pero cuidado, el que se dropeen nodos temporalmente hará también que los pesos se ajusten doblemente así que tenemos dos opciones que funcionan casi igual: dividir entre (1-p) en cada corrección e iteración o multiplicar por (1-p) cada weigth que derive del entrenamiento.



Nota: Es aplicado tanto a los inputs como a los hidden layers, pero no a outputs.

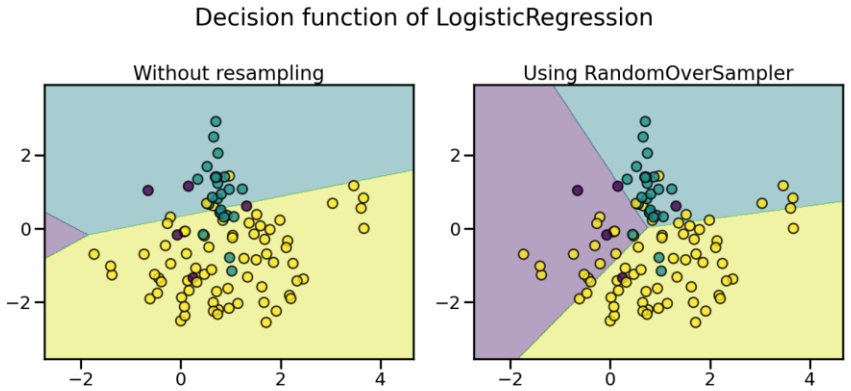
Técnicas de Sobremuestreo

Consiste en aumentar el número de datos que contiene nuestra clase minoritaria (equilibrar la data)

1: Random Over-Sampler (ROS)

- Lo que hace es duplicar entradas de la clase minoritaria.

- Random Over-Sampler implica duplicar aleatoriamente ejemplos de la clase minoritaria y agregarlos al conjunto de datos de entrenamiento.



https://www.youtube.com/watch?v=Ii1_NXobwKI&t=409s&ab_channel=SacaTuCarrera

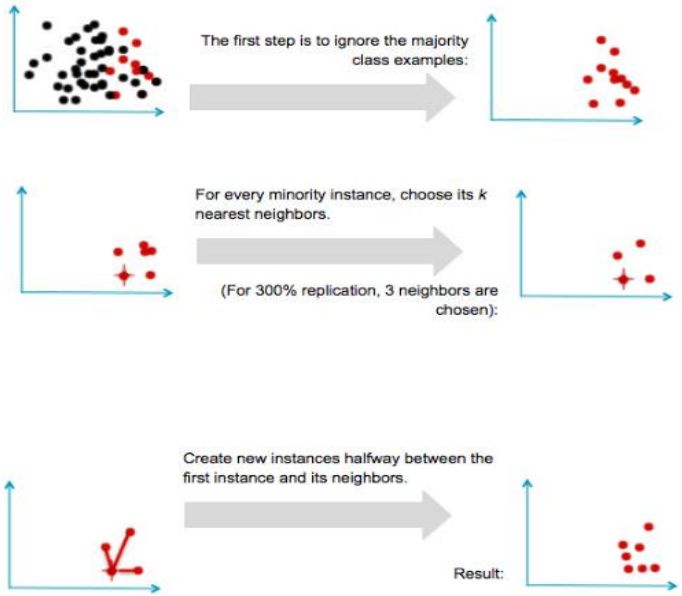
Técnicas de Sobremuestreo

Consiste en aumentar el número de datos que contiene nuestra clase minoritaria (equilibrar la data)

2: SMOTE

- Funciona similar, pero en este caso lo que hace es crear muestras sintéticas, lo que hace es coger las características de 2 datos y calcular la media ponderada de esas características. Generándonos un nuevo dato.

- Diferencia con el Random Over-Sampler……..



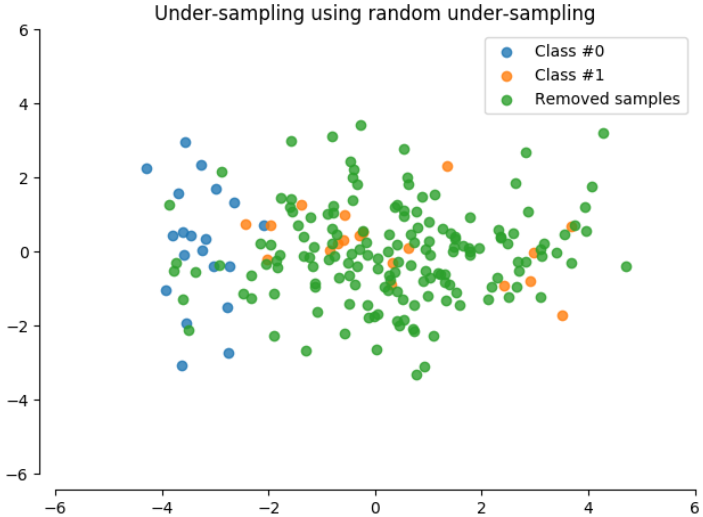
Técnicas de Submuestreo

Consiste en disminuir el número de datos que contiene nuestra clase mayoritaria (equilibrar la data)

1: Random Under-Sampler (RUS)

- Lo que hace es quitar muestras de manera aleatoria

- Submuestrea la(s) clase(s) mayoritaria(s) seleccionando muestras al azar con o sin reemplazo.



Técnicas de Submuestreo

Consiste en disminuir el número de datos que contiene nuestra clase mayoritaria (equilibrar la data)

2: Pérdidas Cercanas (Near miss)

- Near-miss es un algoritmo que puede ayudar a equilibrar un conjunto de datos desequilibrado.
- Cuando dos puntos que pertenecen a clases diferentes están muy cerca uno del otro en la distribución, este algoritmo elimina el punto de datos de la clase más grande tratando de equilibrar la distribución.