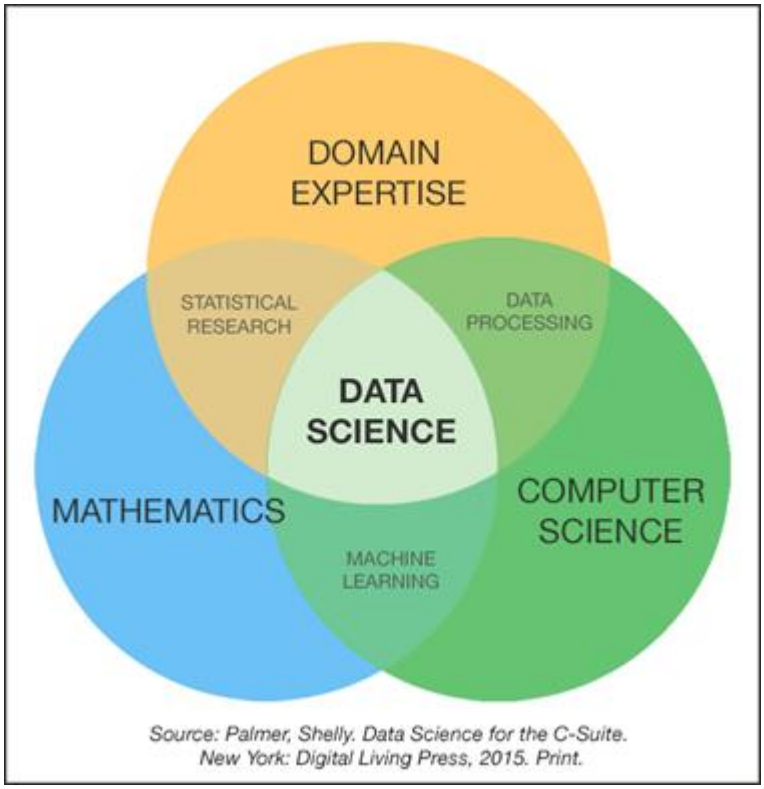
CTIC UNI
CENTRO DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIONES
UNIVERSIDAD NACIONAL DE INGENIERÍA

CIENCIA DE DATOS:

APRENDE LOS FUNDAMENTOS DE MANERA PRÁCTICA



SESION 06

APRENDIZAJE NO SUPERVISADO DEEP LEARNING

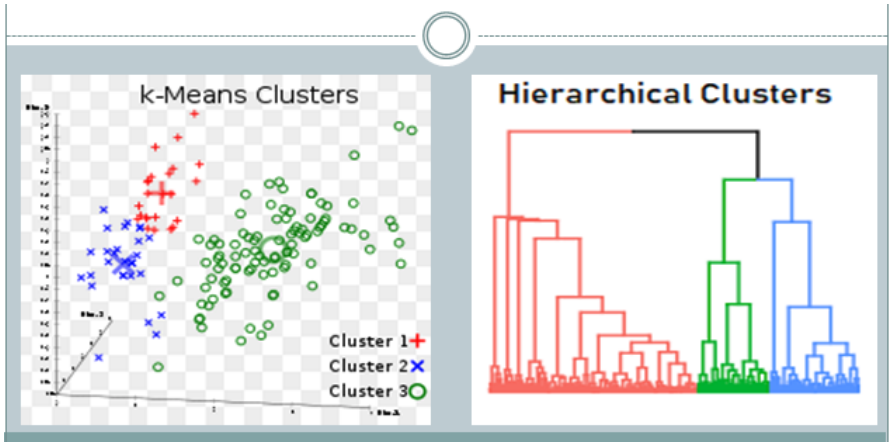
Juan Antonio Chipoco Vidal
jchipoco@gmail.com

ÍNDICE

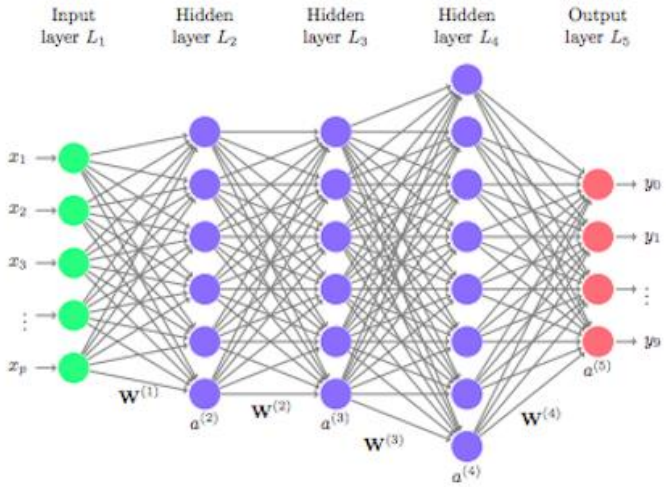
Objetivo

El objetivo de esta sesion es dar a conocer nociones y algoritmos basicos de aprendizaje no supervisado y aprendizaje profundo. Cada uno de estos temas merece un curso completo por lo que debe tomarse esta sesion solamente como una pequeña introduccion a ambas areas.

Revisaremos como funcionan el clustering jerarquico y el k-means en el caso de aprendizaje no supervisado.



Tambien revisaremos algunos conceptos y algoritmos de Deep Learning.



Aprendizaje no supervisado

En algunos problemas de reconocimiento de patrones, los datos de entrenamiento consisten en un conjunto de vectores de entrada x sin ningún valor objetivo correspondiente. El objetivo de tales problemas de aprendizaje no supervisados puede ser descubrir grupos de ejemplos similares dentro de los datos, lo que se denomina agrupación, o determinar cómo se distribuyen los datos en el espacio, lo que se conoce como estimación de densidad. Para plantearlo en términos más simples, para un espacio de n muestras de x1 a xn, no se proporcionan etiquetas de clase para cada muestra, por lo que se conoce como aprendizaje sin profesor.

Problemas con el aprendizaje no supervisado:

El aprendizaje no supervisado es más difícil de realizar en comparación con el aprendizaje supervisado.

¿Cómo sabemos si los resultados son significativos si no hay etiquetas de respuesta disponibles?

Se requiere que un experto revise los resultados (evaluación externa)

Definir una función objetivo sobre la agrupación (evaluación interna)

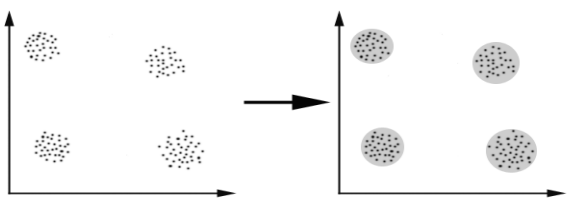
¿Por qué se necesita el aprendizaje no supervisado a pesar de estos problemas?

Etiquetar grandes conjuntos de datos es muy costoso y, por lo tanto, solo podemos etiquetar algunos ejemplos manualmente.
Ejemplo: reconocimiento de voz

Puede haber casos en los que no sepamos en cuántas clases se dividen los datos.
Ejemplo: minería de datos

Es posible que deseemos utilizar el agrupamiento para obtener una idea de la estructura de los datos antes de diseñar un clasificador.

El clustering puede considerarse el problema de aprendizaje no supervisado más importante; entonces, como cualquier otro problema de este tipo, se trata de encontrar una estructura en una colección de datos no etiquetados. Una definición vaga de clustering podría ser "el proceso de organizar objetos en grupos cuyos miembros son similares de alguna manera". Por lo tanto, un clúster es una colección de objetos que son "similares" entre ellos y son "diferentes" a los objetos pertenecientes a otros clústeres.



K-means

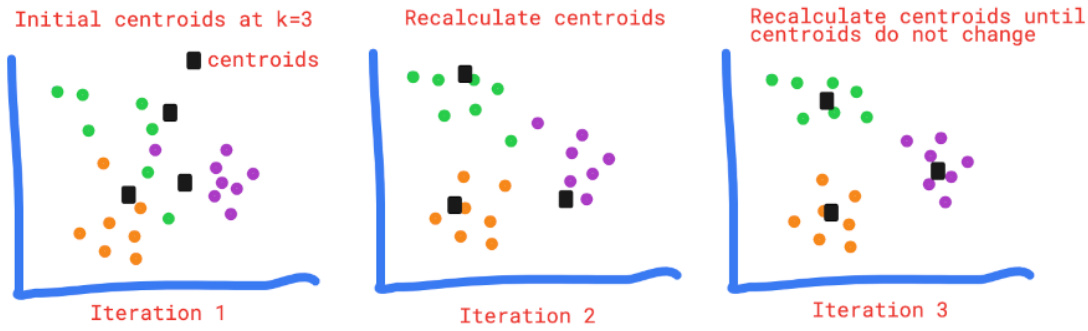
La agrupación en clústeres de k-means es un método de agrupación no supervisado, iterativo y basado en prototipos en el que todos los puntos de datos se dividen en un número k de clusters, cada uno de los cuales está representado por sus centroides (prototipo). El centroide de un cluster suele ser una media de todos los puntos de datos de ese conglomerado. k-means es un algoritmo de agrupación en particiones y funciona bien con clusters de forma esférica.

Los puntos de datos en un grupo están más cerca de los centroides de ese grupo. Hay una gran similitud entre los puntos de datos en un grupo (alta similitud intraclase o dentro del grupo) y los puntos de datos de un grupo son diferentes a los puntos de datos de otro grupo (baja similitud entre clases o entre grupos).

La similitud y la diferencia se calculan utilizando la distancia del cluster entre los puntos de datos. A menudo se calcula utilizando la distancia euclidiana o de correlación (por ejemplo, distancias de correlación de Pearson, Spearman, Kendall)

Pasos involucrados en el algoritmo de agrupamiento k-means:

1 Elija el número k de grupos y determine sus centroides.
2 Asigne cada punto de datos a su centroide más cercano usando metricas de distancia.
3 Vuelva a calcular los nuevos centroides y nuevamente asigne cada punto de datos a su centroide más cercano.
4 Repita los pasos 3 y 4 hasta que los centroides no cambien o no cambie la función de criterio (J)



Supongamos que tenemos (x_1, x_2, …, x_n) muestras en n dimensiones, entonces el agrupamiento de k-means divide las n muestras en k agrupamientos de manera que la suma total de cuadrados dentro del agrupamiento se minimiza,

$$J = \sum_{j=1}^{k} \sum_{i \in C_j} \| x_i - c_j \|^2$$

J = criterion function
k = number of clusters
x_i = ith sample
c_j = centroid of jth cluster C_j

K-means

El agrupamiento de k-means encuentra el número óptimo de agrupamientos (k) mientras minimiza la función de criterio de agrupamiento (J). Cada grupo k contiene al menos un punto de datos. El método k-means produce una estructura de clúster plana, es decir, no hay una estructura anidada como en el agrupamiento jerárquico.

Precisión del cluster de k-means:

Como k-means es un método de cluster (no de clasificación), no se debe evaluar la precisión. Esto se debe a que no entrenamos el modelo con datos de etiquetas de clase y, por lo tanto, habrá incoherencias entre las etiquetas de clase verdaderas y las etiquetas de clase predichas. Puede comparar el diagrama de dispersión del conjunto de datos original y el diagrama de dispersión después de la agrupación en clusters de k- means para evaluar el rendimiento.

k-means limitación del cluster:

1 En el agrupamiento de k-means, es esencial proporcionar el número de grupos que se formarán a partir de los datos. Si el conjunto de datos está bien separado, sera fácil identificar la cantidad óptima de grupos mediante el método elbow. Pero, si los datos no están bien separados, sera difícil encontrar el número óptimo de grupos.

2 El agrupamiento de k-means no es aplicable a los datos categóricos ya que su prototipo se basa en el centroide. Si tiene datos categóricos, es mejor usar el método de agrupación k-medoids (Partition Around Medoids - PAM). En k-medoids, el prototipo es medoid (punto de datos más representativo para un clúster).

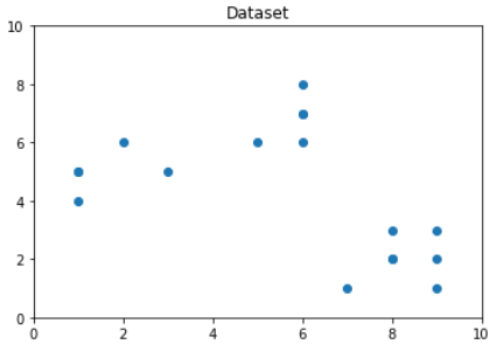
3 El agrupamiento de k-means es sensible a los valores atípicos. El agrupamiento de DBSCAN es un método alternativo para agrupar conjuntos de datos con ruido/valores atípicos.

4 La agrupación en clusters de k-means funciona bien con clusters de forma esférica de tamaños similares. Si hay conglomerados de formas arbitrarias, es posible que k-means no funcione bien. Debe considerar la agrupación en clústeres de DBSCAN para clusters de forma arbitraria.

5 Las k-means pueden producir grupos vacíos (sin puntos asignados al grupo) según el método de inicialización y el valor de k. En este caso, debe intentar agrupar con diferentes valores de k.

K-means: Elbow Method

Un paso fundamental para cualquier algoritmo no supervisado es determinar el número óptimo de clusters en los que se pueden agrupar los datos. El método del codo es uno de los métodos más populares para determinar este valor óptimo de k.



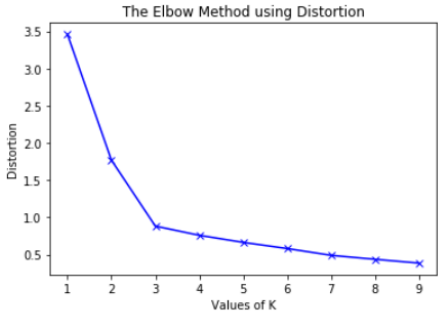
A partir de la visualización anterior, podemos ver que la cantidad óptima de clústeres debe ser alrededor de 3. Pero la visualización de los datos por sí sola no siempre puede dar la respuesta correcta.

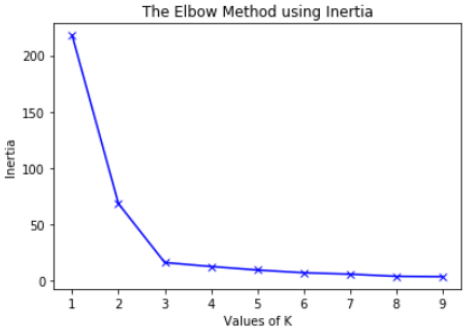
Definamos lo siguiente:

Distorsión: se calcula como el promedio de las distancias al cuadrado desde los centros de los clústeres respectivos. Por lo general, se utiliza la métrica de distancia euclidiana.

Inercia: Es la suma de las distancias al cuadrado de las muestras a su centro del cluster más cercano.

Iteramos los valores de k de 1 a 9 y calculamos los valores de distorsiones para cada valor de k y calculamos la distorsión y la inercia para cada valor de k en el rango dado.





Para determinar el número óptimo de grupos, tenemos que seleccionar el valor de k en el "codo", es decir, el punto después del cual la distorsión/inercia comienza a disminuir de forma lineal. Por lo tanto, para los datos dados, concluimos que el número óptimo de conglomerados para los datos es 3.

K-means: Silhouette Algorithm

Uno de los pasos fundamentales de un algoritmo de aprendizaje no supervisado es determinar el número de clusters en los que se pueden dividir los datos. El algoritmo de silueta es uno de los muchos algoritmos para determinar el número óptimo de grupos para una técnica de aprendizaje no supervisada.

En el algoritmo de Silhouette, asumimos que los datos ya se han agrupado en k clusters mediante una técnica de agrupamiento (típicamente técnica de agrupamiento de K-Means). Luego, para cada punto de datos, definimos lo siguiente:

C(i) -El grupo asignado al i-ésimo punto de datos

|C(yo)| – El número de puntos de datos en el grupo asignado al i-ésimo punto de datos

a(i) – Da una medida de qué tan bien asignado está el i-ésimo punto de datos a su grupo

$$a(i) = \frac{1}{|C(i)|-1} \sum_{C(i), i \neq j} d(i,j)$$

b(i) – Se define como la disimilitud promedio con el grupo más cercano que no es su grupo

$$b(i) = min_{i \neq j} \left(\frac{1}{|C(j)|} \sum_{j \epsilon C(j)} d(i,j) \right)$$

El coeficiente de silueta s(i) viene dado por:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

Determinamos la silueta promedio para cada valor de k y para el valor de k que tiene el valor máximo de s(i) se considera el número óptimo de grupos para el algoritmo de aprendizaje no supervisado.

Consideremos los siguientes datos:

S.No	X1	X2
1.	-7.36	6.37
2.	3.08	-6.78
3.	5.03	-8.31
4.	-1.93	-0.92
5.	-8.86	6.60

K-means: Silhouette Algorithm

Ahora iteramos los valores de k de 2 a 5. Suponemos que no existen datos prácticos para los cuales todos los puntos de datos se puedan agrupar de manera óptima en 1 grupo.

Construimos las siguientes tablas para cada valor de k:

k = 2

S.No	a(i)	b(i)	s(i)
1.	5.31	14.1	0.62
2.	2.47	13.15	0.81
3.	2.47	14.97	0.84
4.	9.66	8.93	-0.076
5.	5.88	19.16	0.69

Average value of s(i) = 0.58

k = 3

S.No	a(i)	b(i)	s(i)
1.	1.52	9.09	0.83
2.	2.47	7.71	0.68
3.	2.47	10.15	0.76
4.	0	7.71	1
5.	1.52	17.93	0.92

Average value of s(i) = 0.84

k = 4

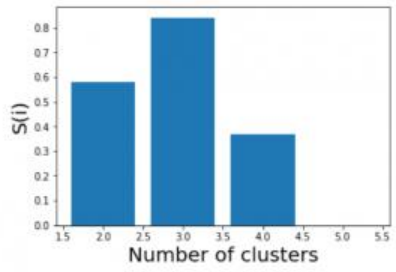
S.No	a(i)	b(i)	s(i)
1.	1.52	9.09	0.83
2.	infinite	2.47	0
3.	infinite	2.47	0
4.	infinite	7.71	0
5.	1.52	10.23	0.85

Average value of s(i) = 0.37

k = 5

S.No	a(i)	b(i)	s(i)
1.	infinite	1.52	0
2.	infinite	2.47	0
3.	infinite	2.47	0
4.	infinite	7.71	0
5.	infinite	1.52	0

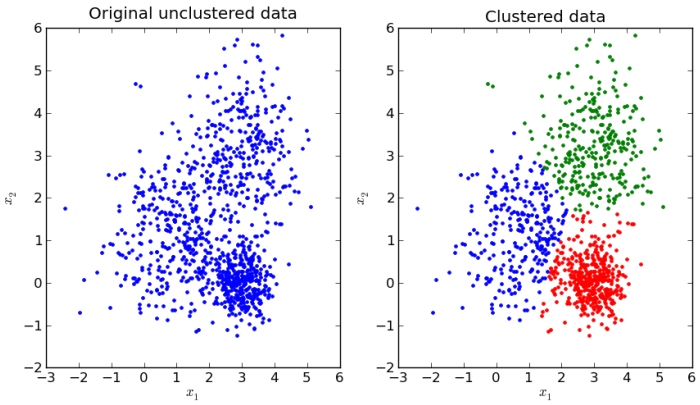
Average value of s(i) = 0



Vemos que el valor más alto de s(i) existe para k = 3. Por lo tanto, concluimos que el número óptimo de conglomerados para los datos dados es 3.

Hierarchical clustering

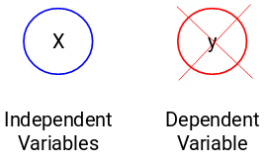
El clustering es básicamente una técnica que agrupa puntos de datos similares de modo que los puntos en el mismo grupo son más similares entre sí que los puntos en los otros grupos. El grupo de puntos de datos similares se llama Cluster.



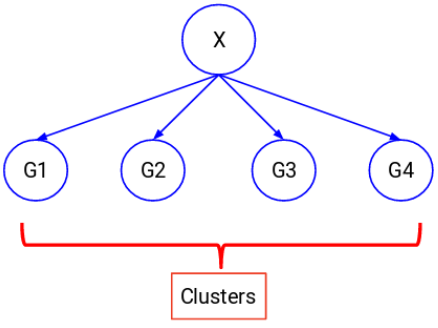
Diferencias entre los modelos de agrupamiento y clasificación/regresión:

En los modelos de clasificación y regresión, se nos proporciona un conjunto de datos (D) que contiene puntos de datos (x_i) y etiquetas de clase (y_i). Donde, los y_i pertenecen a {0,1} o {0,1,2,…,n) para modelos de Clasificación y los y_i pertenecen a valores reales para modelos de regresión.

Cuando se trata de agrupamiento, se nos proporciona un conjunto de datos que contiene solo puntos de datos (x_i). Aquí no contamos con las etiquetas de clase (y_i).



Intentamos dividir los datos completos en un conjunto de grupos en estos casos. Estos grupos se conocen como clústeres y el proceso de creación de estos clústeres se conoce como clustering.



Hierarchical clustering

Hay ciertos desafíos con K-means. Siempre trata de hacer racimos del mismo tamaño. Además, tenemos que decidir el número de grupos al comienzo del algoritmo. Idealmente, no sabríamos cuántos grupos deberíamos tener al comienzo del algoritmo y, por lo tanto, es un desafío con K-means.

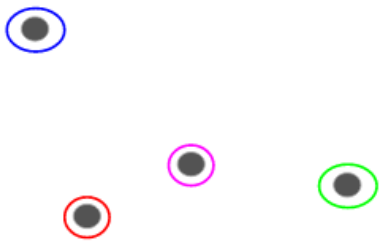
En minería de datos y estadísticas, el agrupamiento jerárquico (también llamado análisis de conglomerados jerárquicos o HCA) es un método de análisis de conglomerados que busca construir una jerarquía de conglomerados.

Hay principalmente dos tipos de agrupamiento jerárquico:

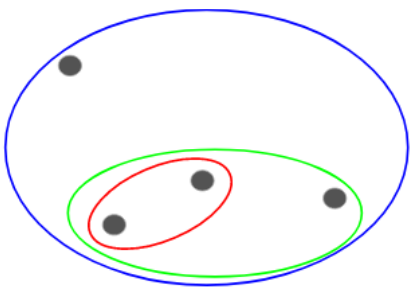
Agrupación jerárquica aglomerativa
Agrupación jerárquica divisiva

Agrupación jerárquica aglomerativa

Asignamos cada punto a un grupo individual en esta técnica. Supongamos que hay 4 puntos de datos. Asignaremos cada uno de estos puntos a un grupo y, por lo tanto, tendremos 4 grupos al principio:



Luego, en cada iteración, fusionamos el par de clústeres más cercano y repetimos este paso hasta que solo quede un único clúster:

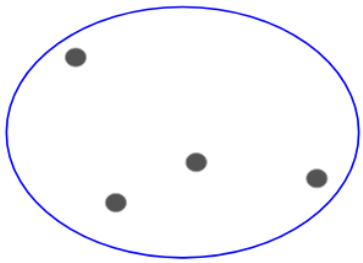


Hierarchical clustering

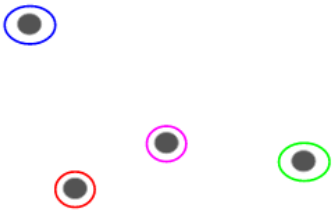
Agrupación jerárquica divisiva

El agrupamiento jerárquico divisivo funciona de manera opuesta. En lugar de comenzar con n conglomerados (en el caso de n observaciones), comenzamos con un solo conglomerado y asignamos todos los puntos a ese conglomerado.

No importa si tenemos 10 o 1000 puntos de datos. Todos estos puntos pertenecerán al mismo clúster al principio:



Ahora, en cada iteración, dividimos el punto más lejano del grupo y repetimos este proceso hasta que cada grupo contenga un solo punto:



Fusionamos los puntos o grupos más similares en el agrupamiento jerárquico. Pero ¿cómo decidimos qué puntos son similares y cuáles no?

Unaforma de calcular la similitud es tomar la distancia entre los centroides de estos grupos. Los puntos que tienen la menor distancia se denominan puntos similares y podemos fusionarlos. También podemos referirnos a esto como un algoritmo basado en la distancia (ya que estamos calculando las distancias entre los grupos).

En el agrupamiento jerárquico, tenemos un concepto llamado matriz de proximidad. Esto almacena las distancias entre cada punto. Tomemos un ejemplo para comprender esta matriz, así como los pasos para realizar el agrupamiento jerárquico.

Hierarchical clustering

Matriz de proximidad

Supongamos que una maestra quiere dividir a sus alumnos en diferentes grupos. Tiene las calificaciones obtenidas por cada estudiante en un examen y, en función de estas calificaciones, quiere segmentarlos por grupos. No hay un objetivo fijo aquí en cuanto a cuántos grupos tener. Dado que la profesora no sabe qué tipo de alumnos debe asignar a qué grupo, no se puede resolver como un problema de aprendizaje supervisado. Entonces, intentaremos aplicar la agrupación jerárquica aquí y segmentar a los estudiantes en diferentes grupos.

Student_ID	Marks
1	10
2	7
3	28
4	20
5	35

Primero, crearemos una matriz de proximidad que nos dirá la distancia entre cada uno de estos puntos. Como estamos calculando la distancia de cada punto desde cada uno de los otros puntos, obtendremos una matriz cuadrada de forma n x n (donde n es el número de observaciones).

Generemos la matriz de proximidad de 5 x 5 para nuestro ejemplo:

ID	**1**	**2**	**3**	**4**	**5**
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Hierarchical clustering

Pasos:

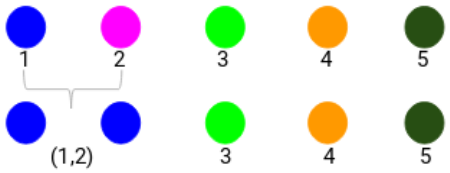
Primero, asignamos todos los puntos a un clúster individual:

Los diferentes colores aquí representan diferentes grupos. Vemos que tenemos 5 grupos diferentes para los 5 puntos en nuestros datos.

A continuación, veremos la distancia más pequeña en la matriz de proximidad y fusionaremos los puntos con la distancia más pequeña. Luego actualizamos la matriz de proximidad:

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Aquí, la distancia más pequeña es 3 y, por lo tanto, fusionaremos los puntos 1 y 2:



Veamos los clústeres actualizados y, en consecuencia, actualicemos la matriz de proximidad:

Student_ID	Marks
(1,2)	10
3	28
4	20
5	35

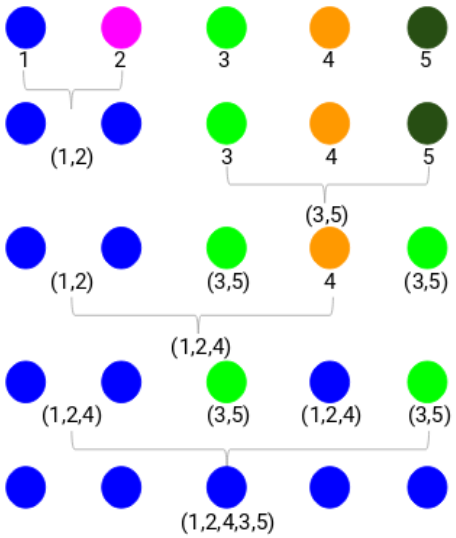
Observemos que hemos tomado el máximo de las dos marcas (7, 10) para reemplazar las marcas de este grupo. En lugar del máximo, también podemos tomar el valor mínimo o los valores medios.

Hierarchical clustering

Ahora, calcularemos nuevamente la matriz de proximidad para estos clústeres:

ID	(1,2)	3	4	5
(1,2)	0	18	10	25
3	18	0	8	7
4	10	8	0	15
5	25	7	15	0

Recursivamente veremos la distancia mínima en la matriz de proximidad y luego fusionaremos el par de grupos más cercano. Obtendremos los clústeres fusionados como se muestra a continuación después de repetir estos pasos:



Empezamos con 5 clusters y finalmente tenemos un solo cluster. Así es como funciona el agrupamiento jerárquico aglomerativo. Pero: ¿cómo decidimos la cantidad de clusters?

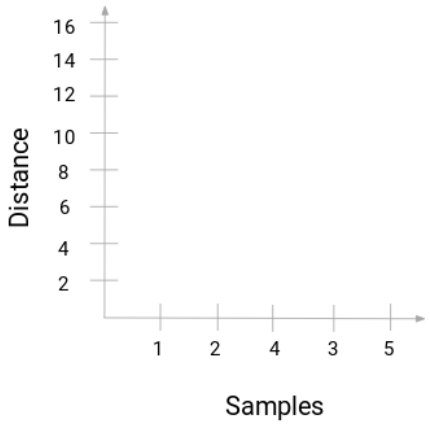
¿Cómo debemos elegir el número de clústeres en el agrupamiento jerárquico?

Para obtener el número de grupos para el agrupamiento jerárquico, utilizamos un concepto llamado Dendograma. Un dendrograma es un diagrama en forma de árbol que registra las secuencias de fusiones o divisiones.

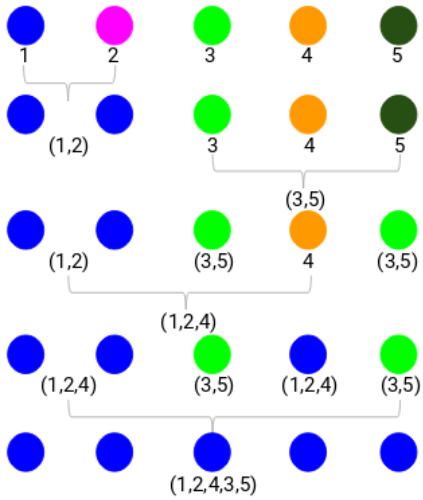
Volvamos a nuestro ejemplo de maestro-alumno. Cada vez que fusionamos dos grupos, un dendrograma registrará la distancia entre estos grupos y la representará en forma de gráfico.

Hierarchical clustering

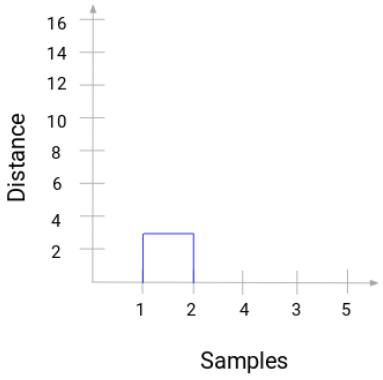
Veamos cómo se ve un dendograma:



Tenemos las muestras del conjunto de datos en el eje x y la distancia en el eje y. Siempre que se fusionen dos clústeres, los uniremos en este dendrograma y la altura de la unión será la distancia entre estos puntos. Construyamos el dendograma para nuestro ejemplo:

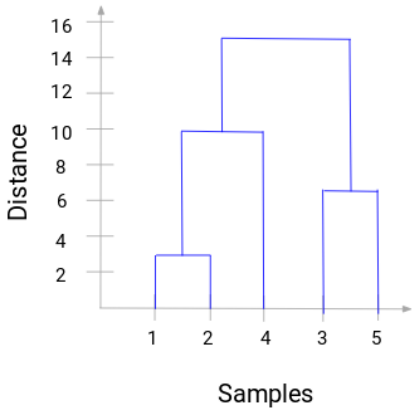


Comenzamos fusionando las muestras 1 y 2 y la distancia entre estas dos muestras fue 3:



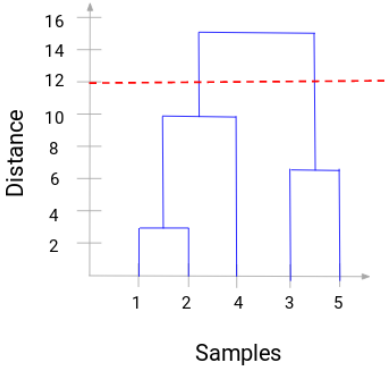
Hierarchical clustering

Aquí podemos ver que hemos fusionado las muestras 1 y 2. La línea vertical representa la distancia entre estas muestras. De manera similar, trazamos todos los pasos en los que fusionamos los grupos y, finalmente, obtenemos un dendrograma como este:



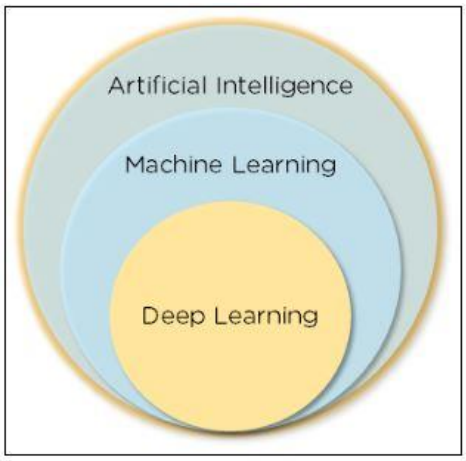
Podemos visualizar claramente los pasos de la agrupación jerárquica. Cuanto mayor sea la distancia de las líneas verticales en el dendrograma, mayor será la distancia entre esos grupos.

Ahora, podemos establecer una distancia de umbral y dibujar una línea horizontal (generalmente, tratamos de establecer el umbral de tal manera que corte la línea vertical más alta). Establezcamos este umbral en 12 y dibujemos una línea horizontal:



El número de conglomerados será el número de líneas verticales que se cruzan con la línea dibujada con el umbral. En el ejemplo anterior, dado que la línea roja se cruza con 2 líneas verticales, tendremos 2 grupos. Un conglomerado tendrá una muestra (1,2,4) y el otro tendrá una muestra (3,5).

Deep Learning



La inteligencia artificial es la capacidad de una máquina para imitar el comportamiento humano inteligente. El aprendizaje automático permite que un sistema aprenda y mejore a partir de la experiencia automáticamente. El aprendizaje profundo es una aplicación de aprendizaje automático que utiliza algoritmos complejos y redes neuronales profundas para entrenar un modelo.

Deep Learning es una subdivisión del aprendizaje automático que imita el funcionamiento de un cerebro humano con la ayuda de redes neuronales artificiales. Es útil en el procesamiento de Big Data y puede crear patrones importantes que brindan información valiosa sobre la toma de decisiones importantes. El etiquetado manual de datos no supervisados requiere mucho tiempo y es costoso. Los tutoriales de DeepLearning ayudan a superar esto con la ayuda de algoritmos altamente sofisticados que brindan información esencial al analizar y acumular los datos.

Deep Learning aprovecha las diferentes capas de redes neuronales que permiten aprender, desaprender y volver a aprender. Cubre todas las habilidades y algoritmos esenciales de CNN a RNN y, por lo tanto, proporciona soluciones más inteligentes.

Importancia del aprendizaje profundo:

El aprendizaje automático funciona solo con conjuntos de datos estructurados y semiestructurados, mientras que el aprendizaje profundo funciona tanto con datos estructurados como no estructurados.

Los algoritmos de aprendizaje profundo pueden realizar operaciones complejas de manera eficiente, mientras que los algoritmos de aprendizaje automático no.

Los algoritmos de aprendizaje automático utilizan datos de muestra etiquetados para extraer patrones, mientras que el aprendizaje profundo acepta grandes volúmenes de datos como entrada y analiza los datos de entrada para extraer características de un objeto.

Deep Learning

El rendimiento de los algoritmos de aprendizaje automático disminuye a medida que aumenta la cantidad de datos; por lo que para mantener el rendimiento del modelo, necesitamos un aprendizaje profundo.

¿Qué son las Redes Neuronales?

Ahora que sabemos qué es exactamente el aprendizaje profundo, su aplicación e importancia, a continuación, veamos las redes neuronales y sus operaciones. Una red neuronal es un sistema modelado en el cerebro humano, que consta de una capa de entrada, varias capas ocultas y una capa de salida. Los datos se alimentan como entrada a las neuronas. La información se transfiere a la siguiente capa utilizando pesos y sesgos apropiados. La salida es el valor final predicho por la neurona artificial.

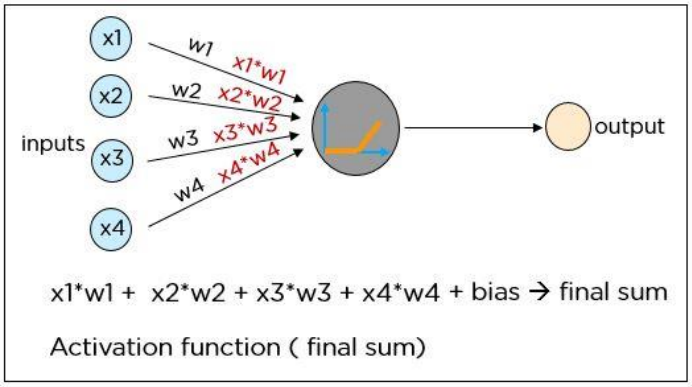
Cada neurona en una red neuronal realiza las siguientes operaciones:

Se encuentra el producto de cada entrada y el peso del canal por el que se pasa.

Se calcula la suma de los productos ponderados, que se denomina suma ponderada

Se suma un valor de sesgo de la neurona a la suma ponderada

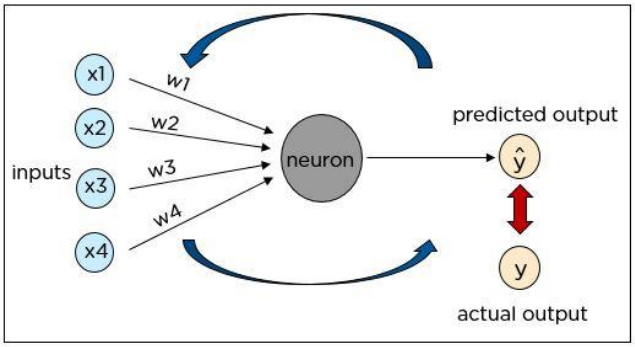
La suma final se somete a una función particular conocida como función de activación.



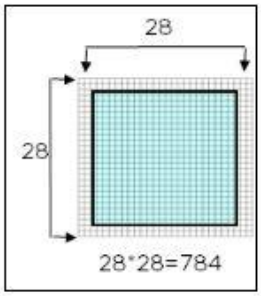
Deep Learning

Función de costo:

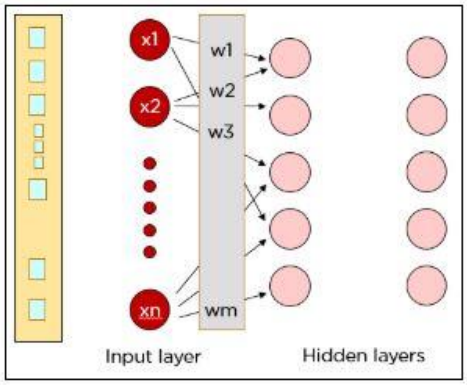
La función de costo es uno de los componentes significativos de una red neuronal. El valor del costo es la diferencia entre la salida prevista de las redes neuronales y la salida real de un conjunto de datos de entrenamiento etiquetados. El valor de menor costo se obtiene al hacer ajustes a los pesos y sesgos de forma iterativa a lo largo del proceso de entrenamiento.



Veamos el proceso completo paso a paso:



Cada píxel se alimenta como entrada a las neuronas en la primera capa. Las capas ocultas mejoran la precisión de la salida. Los datos se transmiten de una capa a otra sobre los canales de sobrepeso. Cada neurona de una capa se pondera con respecto a cada una de las neuronas de la siguiente capa.

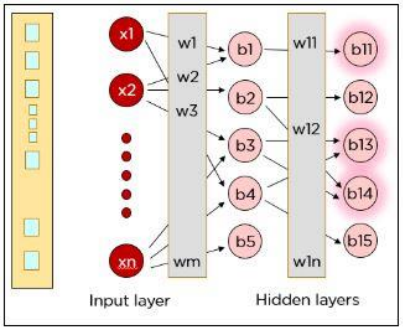


Cada neurona en la primera capa oculta toma un subconjunto de las entradas y las procesa. Todas las entradas se multiplican por sus respectivos pesos y se agrega un sesgo. La salida de la suma ponderada se aplica a una función de activación. Los resultados de la función de activación determinan qué neuronas se activarán en la siguiente capa.

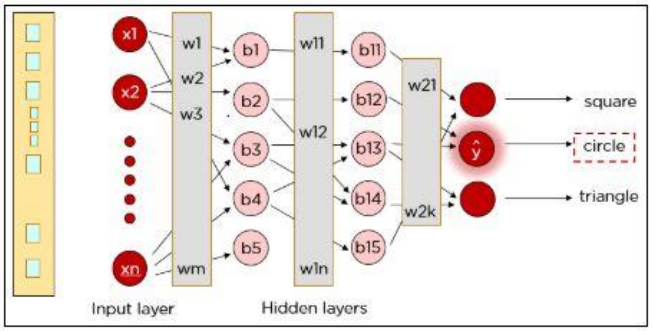
Paso 1: $x_1*w_1 + x_2*w_2 + b_1$

Paso 2: $\Phi(x_1* w_1 + x_2* w_2 + b_1)$

donde Φ es una función de activación



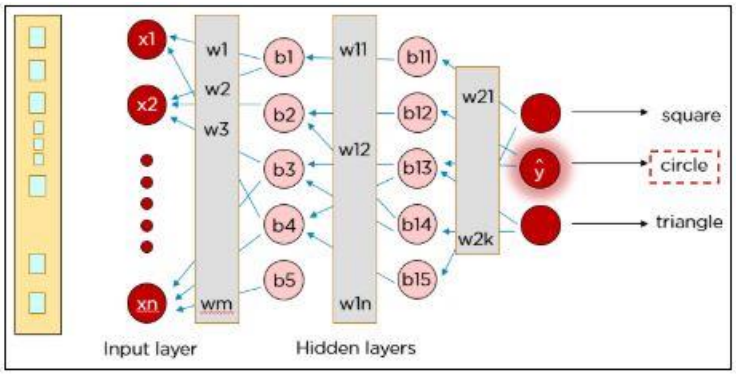
Los pasos anteriores se realizan nuevamente para garantizar que la información llegue a la capa de salida, después de lo cual se activa una sola neurona en la capa de salida según el valor de la función de activación.



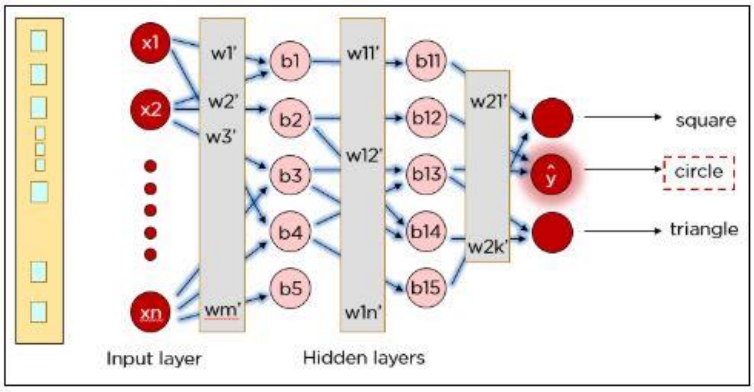
Como se puede ver, nuestra entrada real fue un cuadrado, pero la red neuronal predijo la salida como un círculo. ¿Qué salió mal?

La red neuronal debe entrenarse hasta que la salida pronosticada sea correcta y la salida pronosticada se compare con la salida real mediante el cálculo de la función de costo.

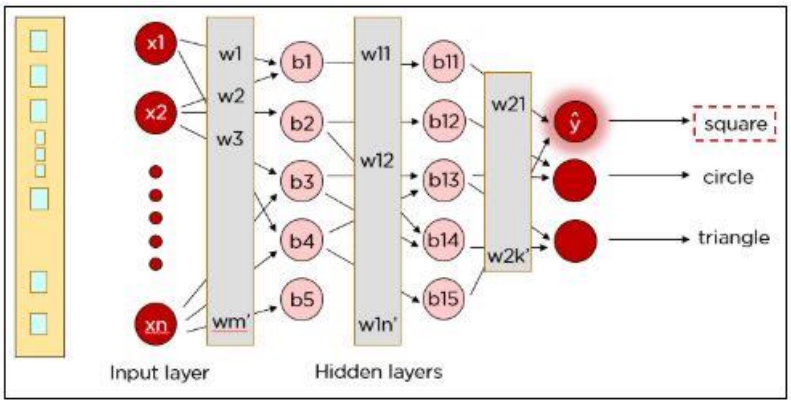
La función de costo se calcula usando la fórmula donde y es el valor real e ŷ es el valor pronosticado. La función de costo determina el error en la predicción y lo informa a la red neuronal. Esto se llama **backpropagation**:



Los pesos se ajustan para reducir el error. La red está entrenada con los nuevos pesos:



Una vez más, se determina el costo y se continúa con el procedimiento de **backpropagation** hasta que el costo no se puede reducir más:



Deep Learning

Las funciones de activación son una parte crítica del diseño de una red neuronal.

La elección de la función de activación en la capa oculta controlará qué tan bien aprende el modelo de red el conjunto de datos de entrenamiento. La elección de la función de activación en la capa de salida definirá el tipo de predicciones que puede hacer el modelo.

Como tal, se debe hacer una elección cuidadosa de la función de activación para cada proyecto de red neuronal de aprendizaje profundo.

Una función de activación en una red neuronal define cómo la suma ponderada de la entrada se transforma en una salida de un nodo o nodos en una capa de la red.

A veces, la función de activación se denomina "función de transferencia". Muchas funciones de activación no son lineales y pueden denominarse como la "no linealidad" en la capa o el diseño de la red.

La elección de la función de activación tiene un gran impacto en la capacidad y el rendimiento de la red neuronal, y se pueden usar diferentes funciones de activación en diferentes partes del modelo.

Una red puede tener tres tipos de capas: capas de entrada que toman la entrada sin procesar del dominio, capas ocultas que toman la entrada de otra capa y pasan la salida a otra capa y capas de salida que hacen una predicción.

Todas las capas ocultas suelen utilizar la misma función de activación. La capa de salida generalmente usará una función de activación diferente de las capas ocultas y depende del tipo de predicción requerida por el modelo.

Las funciones de activación también suelen ser diferenciables, lo que significa que la derivada de primer orden se puede calcular para un valor de entrada dado. Esto es necesario dado que las redes neuronales normalmente se entrenan utilizando el algoritmo de *backpropagation* del error lo cual requiere la derivada del error de predicción para actualizar los pesos del modelo.

Hay muchos tipos diferentes de funciones de activación que se usan en las redes neuronales, aunque quizás solo se use una pequeña cantidad de funciones en la práctica para las capas ocultas y de salida.

Por lo general, se utiliza una función de activación no lineal diferenciable en las capas ocultas de una red neuronal. Esto permite que el modelo aprenda funciones más complejas que una red entrenada usando una función de activación lineal.

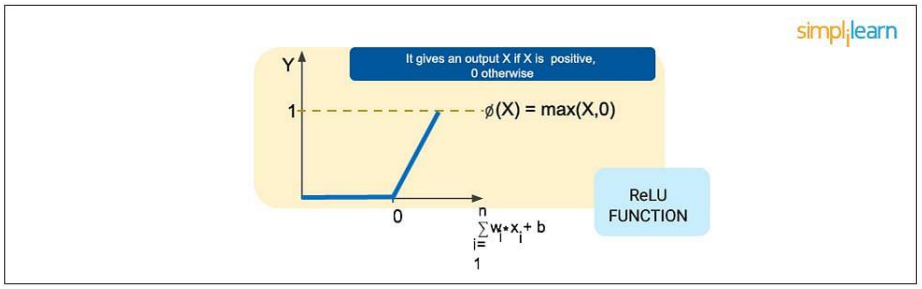
Echemos un vistazo a las funciones de activación utilizadas para **las *capas ocultas***

Función ReLU (unidad lineal rectificada):

La función de activación lineal rectificada, o función de activación ReLU, es quizás la función más común utilizada para capas ocultas.

Es común porque es simple de implementar y efectiva para superar las limitaciones de otras funciones de activación populares anteriormente, como la sigmoide y . Específicamente, es menos susceptible a gradientes que se desvanecen que evitan que se entrenen modelos profundos, aunque puede sufrir otros problemas como unidades saturadas o "muertas".

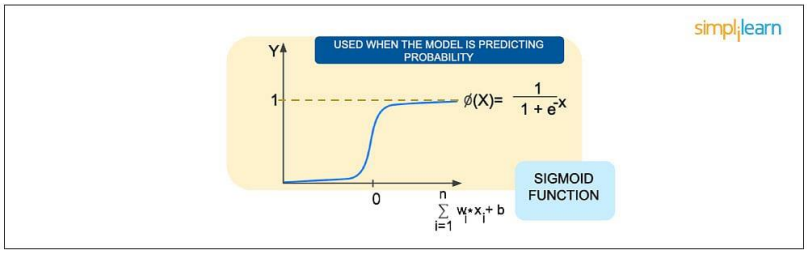
La función ReLU (unidad lineal rectificada) devuelve 1 si x es superior a 1 y si x es menor que 0, devolvera 0.



Función Sigmoide:

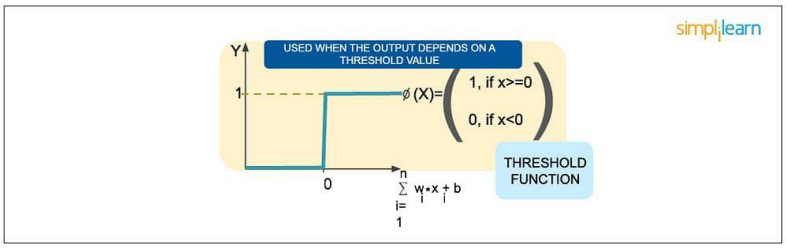
La función de activación sigmoide también se denomina función logística.
Es la misma función utilizada en el algoritmo de clasificación de regresión logística.

La función toma cualquier valor real como entrada y genera valores en el rango de 0 a 1. Cuanto mayor sea la entrada (más positiva), más cerca estará el valor de salida de 1.0, mientras que cuanto más pequeña sea la entrada (más negativa), más cerca estará la función de salida de 0.0.



Función de umbral:

La función de umbral se usa cuando no quiere preocuparse por la incertidumbre en el medio.

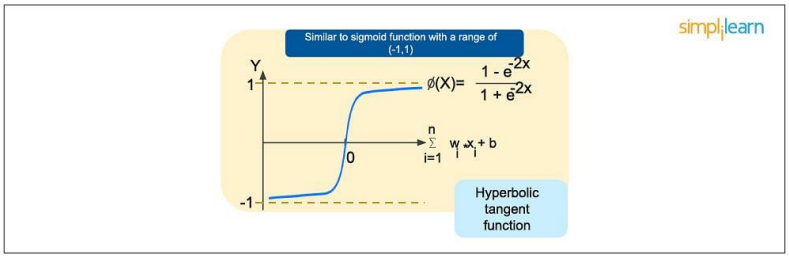


Función tangente hiperbólica

La función de activación de la tangente hiperbólica también se conoce simplemente como la función Tanh (también "tanh" y "TanH").

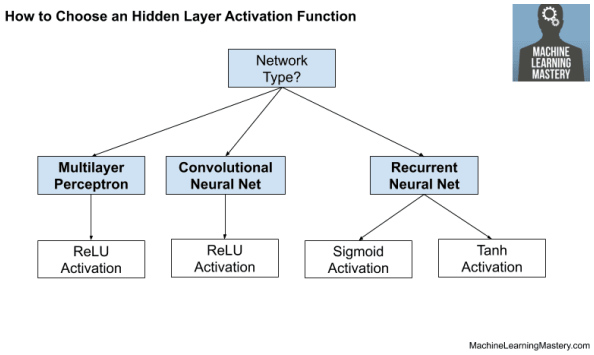
Es muy similar a la función de activación sigmoide e incluso tiene la misma forma de S.

La función toma cualquier valor real como entrada y genera valores en el rango de -1 a 1. Cuanto mayor sea la entrada (más positiva), más cerca estará el valor de salida de 1,0, mientras que cuanto más pequeña sea la entrada (más negativa), más cerca la salida será a -1.0.



Si no estamos seguros de qué función de activación usar para nuestra red, se deben probar algunas y comparar los resultados.

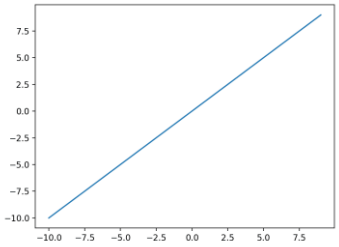
La siguiente figura resume cómo elegir una función de activación para las capas ocultas de su modelo de red neuronal.



En el caso de las capas de salida hay 3 funciones de activacion que se deben contemplar.

Las funcion de activacion sigmoide que ya describimos en la sesion anterior.

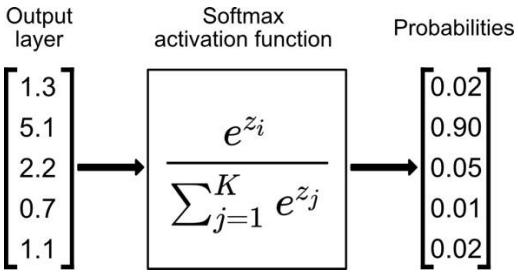
La función de activación lineal también se denomina "identidad" (multiplicada por 1,0) o "sin activación". Esto se debe a que la función de activación lineal no cambia la suma ponderada de la entrada de ninguna manera y, en cambio, devuelve el valor directamente.



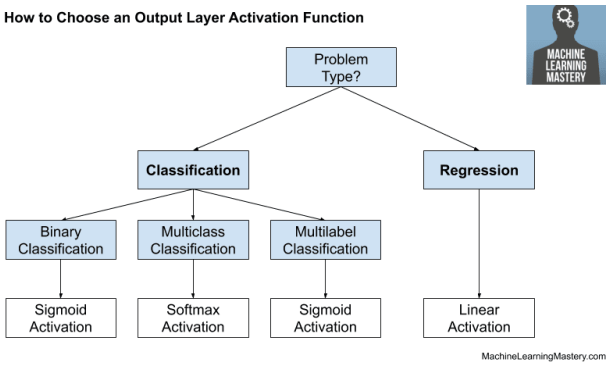
La función softmax genera un vector de valores que suman 1,0 que se pueden interpretar como probabilidades de pertenencia a una clase.

Está relacionado con la función argmax que genera un 0 para todas las opciones y un 1 para la opción elegida. Softmax es una versión "más suave" de argmax que permite una salida similar a la probabilidad de una función en la que el ganador se lo lleva todo.

Como tal, la entrada a la función es un vector de valores reales y la salida es un vector de la misma longitud con valores que suman 1.0 como probabilidades.



La siguiente figura resume cómo elegir una función de activación para la capa de salida de su modelo de red neuronal.



Deep Learning

Types of Algorithms used in Deep Learning

Los algoritmos de aprendizaje profundo funcionan con casi cualquier tipo de datos y requieren grandes cantidades de potencia informática e información para resolver problemas complicados.

1. Convolutional Neural Networks (CNNs)
2. Long Short Term Memory Networks (LSTMs)
3. Recurrent Neural Networks (RNNs)
4. Generative Adversarial Networks (GANs)
5. Radial Basis Function Networks (RBFNs)
6. Multilayer Perceptrons (MLPs)
7. Self Organizing Maps (SOMs)
8. Deep Belief Networks (DBNs)
9. Restricted Boltzmann Machines(RBMs)
10. Autoencoders

Deep Learning

Deep Learning Platforms:

Keras
Keras es un marco de Python para el aprendizaje profundo. Su USP es la reutilización de código para CPU y GPU.

TensorFlow
TensorFlow es una biblioteca de aprendizaje profundo de código abierto desarrollada por Google. Está desarrollado en C++ y tiene su implementación en Python. Keras ahora se puede ejecutar sobre TensorFlow.

DL4J
Deep Learning for Java (DL4J) es la primera biblioteca de aprendizaje profundo escrita para Java y Scala. Está integrado con Hadoop y Apache Spark.

TensorFlow de Google es actualmente la biblioteca de aprendizaje más popular del mundo. Se basa en el concepto de tensores, que se pueden considerer son vectores o matrices de n dimensiones.

A continuación se muestra un ejemplo de tensores que tienen 1D, 2D y multidimensionalidad.