Las capas completamente conectadas se densifican utilizando la **clase Dense**.

Podemos especificar el número de neuronas en la capa como el primer argumento y especificar la función de activación usando el argumento de **activación**.

Usaremos la **función de activación del rectificador** (relu) en las dos primeras capas y la **función de activación sigmoide** en la capa de salida.

Antes las funciones de activación sigmoide y tanh eran preferidas para todas las capas. Hoy en día, se observa un mejor rendimiento utilizando la función de activación del rectificador.

Utilizamos una función de activación sigmoide en la capa de salida para asegurarnos de que nuestra salida de red está entre 0 y 1 y es fácil de mapear a cualquier probabilidad de clase 1 o encajar en una clase dura de cualquiera de las dos clases con un umbral por defecto de 0,5.

Unimos todo sumando cada capa.

* Como se definenumero de capas?
* Como se define cuantas neuronas internas por capa?

La primera capa oculta tiene 12 neuronas y espera 8 variables de entrada (p. ej. entrada dim=8).

La segunda capa oculta tiene 8 neuronas

Finalmente la capa de salida que tiene 1 neurona para predecir la clase (inicio de diabetes o no).

from keras.models import Sequential

from keras.layers import Dense

from sklearn.datasets import make\_blobs

from sklearn.preprocessing import MinMaxScaler

X.shape (100,2)

y.shape(100,) 0 y 1

|  |  |
| --- | --- |
|  | # Crea el modelo  model = Sequential()  #capa 1 12 neuronas  #capa 2 8 neuronas  #capa 3 1 neurona salida  model.add(Dense(12, input\_dim=8, activation='relu'))  model.add(Dense(8, activation='relu'))  model.add(Dense(1, activation='sigmoid')) |

model.compile(loss='binary\_crossentropy', optimizer='adam')

model.fit(X, y, epochs=500, verbose=0)

# model.fit(X, Y, epochs=150, batch\_size=10)

Predecir

# new instances where we do not know the answer

Xnew, \_ = make\_blobs(n\_samples=3, centers=2, n\_features=2, random\_state=1)

Xnew = scalar.transform(Xnew)

# make a prediction

ynew = model.predict(Xnew)

# show the inputs and predicted outputs

for i in range(len(Xnew)):

print("X=%s, Predicted=%s" % (Xnew[i], ynew[i]))

X=[0.89337759 0.65864154], Predicted=[0.00566614]

X=[0.29097707 0.12978982], Predicted=[0.9967221]

X=[0.78082614 0.75391697], Predicted=[0.00513867]