Redes Neuronales en Keras

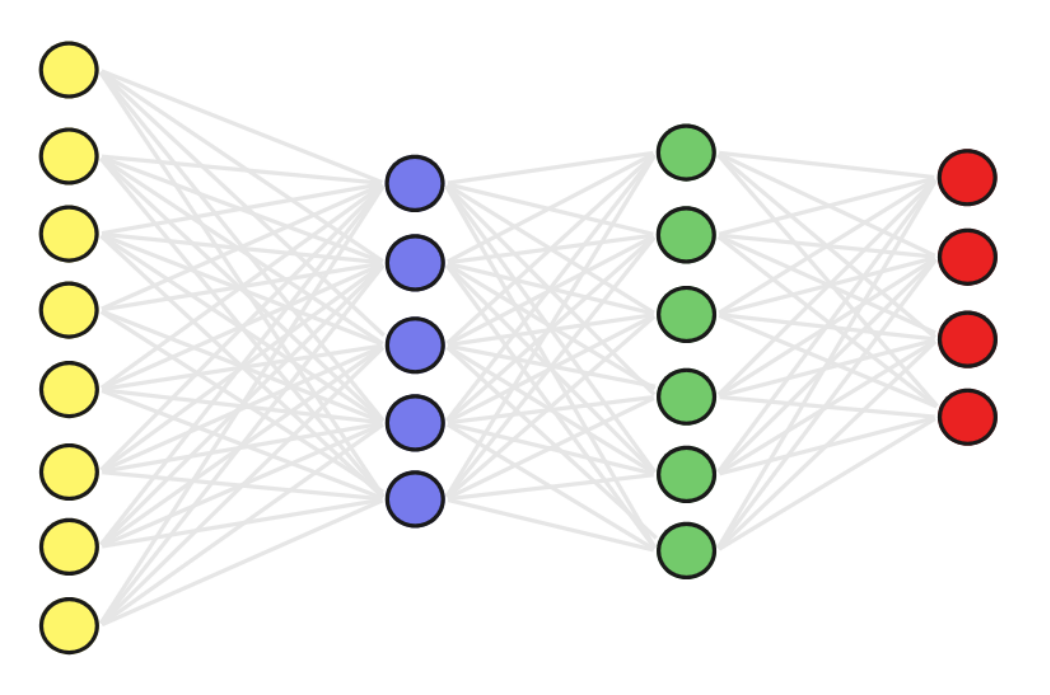
Alfonso Tobar Arancibia
Data Scientist
13-10-2020


github.com/Rcubes/clases-ml/Slides

Redes Neuronales Densas

El tipo de Redes Neuronales más básicas son las Redes Neuronales densas, también llamadas DNN. Las principales características de este tipo de Redes son que:

- Cada Neurona de entrada mapea un **feature** de los datos.
- Todas las Neuronas de una capa se encuentran unidas con todas las neuronas de la capa siguiente.
- Se debe utilizar una Neurona por cada valor de salida.



Cargando Keras desde Tensorflow

Keras corresponde a la API de alto nivel que permite crear las arquitecturas de Redes Neuronales más populares. En el caso de querer dedicarse a la investigación y a generar modelos 100% customizados, entonces convendrá trabajar con Tensorflow directamente.

La API de Tensorflow está cambiando rápidamente por lo que es bueno estar revisando la documentación en todo momento. A partir de Tensorflow 2.0, Keras se encuentra integrado. Esto porque en ocasiones anteriores el desarrollo de Keras estaba atrasado y generaba grandes problemas de compatibilidad.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

Lo primero que debe instanciarse es el tipo de modelo. En nuestro caso será `Sequential()`, y luego se irán agregando capas:

```
model = Sequential(name = 'Modelo')
model.add(Dense(name = 'hidden_1', units = 32, input_shape = (10,), activation = 'relu'))
model.add(Dense(name = 'hidden_2', units = 64, activation = 'relu'))
model.add(Dense(name = 'output', units = 1, activation = None))
```

Crear DNN en Keras

```
model = Sequential(name = 'Modelo')
```

- Esto corresponde a instanciar el Modelo. No importa que tipo de Red Neuronal se utilice siempré se utilizará `Sequential()`.

Pros: API Extremadamente sencilla.

Cons: Sólo permite capas de Neuronas que se aplican de manera secuencial.

```
model.add(Dense(name = 'hidden_1', units = 32, input_shape = (10,), activation = 'relu'))
```

Se utiliza el método `.add()` junto con la capa a utilizar. En este caso una capa densa, que creará uniones entre todas las neuronas de la capa anterior (en este caso la de entrada) y la actual (en este caso 'hidden_1').

- **name**: Corresponde a un nombre para referirse a cada elemento del modelo. *OJO: No usar espacios*.
- **units**: corresponde al número de neuronas de la capa.
- **input_shape**: Corresponde a un parámetro que se coloca en la primera capa definida en Keras. Este parámetro es una tupla que define cuántas features se utilizarán para el proceso de modelamiento. No es necesario definirlo para las siguientes capas ya que se infiere capa a capa.
- **activation**: Corresponde a la función de activación de la capa. Esta debe ser escogida cuidadosamente dependiendo de su ubicación.

Compilar DNN en Keras

```
model.compile(loss = 'binary_crossentropy', optimizer = 'sgd', metrics = ['accuracy'])
```

`.compile()` generará el empaquetamiento final con las características con las que se optimizará y medirá la performance del modelo.

- **loss**: Corresponde a la función de costo. El propósito de esta función es minimizarla lo más posible.
- **optimizer**: Corresponde al proceso de optimización con el que se encontrará los pesos óptimos de tal manera que la red neuronal aprenda.
- **metrics**: Corresponde a la métrica que se utilizará para encontrar el modelo óptimo.

Entrenar el modelo

```
model.fit(X_train, y_train, epochs = 20, validation_split = 0.1, shuffle = True)
```

- **epochs**: Corresponde al número de pasadas de entrenamiento que se utilizarán para entrenar.
- **validation_split**: Corresponde a un parámetro opcional que genera un split de validación para la búsqueda de hiperparámetros.
- **shuffle**: Permite revolver el dataset antes del split de validación.

Recomendaciones

Tipo de Problema	Función Activación	Función Costo
Binary classification	sigmoid	binary_crossentropy
Multi-class, single-label classification	softmax	categorical_crossentropy
Multi-class, multi-label classification	sigmoid	binary_crossentropy
Regression to arbitrary values	None	mse
Regression to values between 0 and 1	sigmoid	mse or binary_crossentropy

- Usar **relu** sólo en capas ocultas, nunca como capa de salida.
- La capa de entrada en Keras es implicita, por lo tanto no tiene asociada una función de activación.
- No usar función de activación es equivalente a usar una Función Identidad.
- Para problemas de regresión también se puede usar mae tanto como función de Costo como métrica de evaluación.