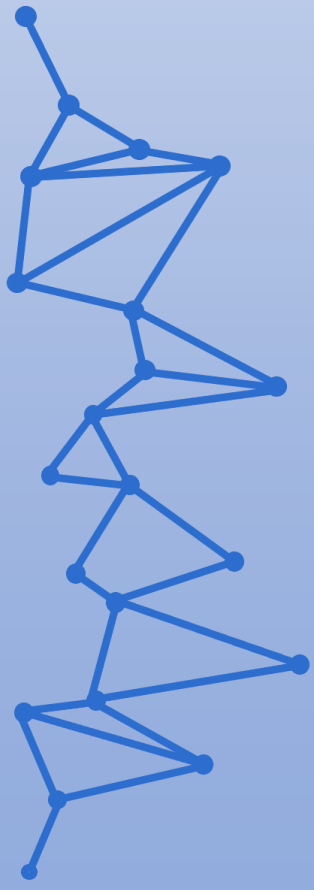
Curso de Especialización de
Inteligencia Artificial y Big Data (IABD)



Programación de Inteligencia Artificial

UD06. Ajustes de un modelo de aprendizaje automático.
Resumen.

JUAN ANTONIO GARCIA MUELAS

Para dividir el dataset entre los datos de entrenamiento (train) y datos para la evaluación (test), **tenemos en Keras la función `train_test_split`,** a la que se pasan la **porción de datos** que queremos **reservar para el test y el grado de aleatoriedad** en el reparto entre ambos conjuntos:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

En este caso, se ha dedicado el 75% de los datos para entrenamiento y el 25% para el test.

El **sobre entrenamiento** u "**`overfitting`**" sucede **cuando se cuelan datos anómalos** que el algoritmo trata de adaptar, **y el modelo se confunde**.

El término "**`overfitting`**" también se traduce como "**sobre ajuste**", en su forma más literal. Al fin y al cabo, consiste en tratar de ajustar la superficie o hiperplano solución a la distribución de los datos, aunque éstos no siempre estén donde deberían estar.

Una técnica que nos permite monitorizar el posible overfitting en un entrenamiento con Kerase es reservar datos de validación del dataset y utilizarlos en fit, como parámetro "validation_data".

Para aumentar el desempeño del modelo y **evitar el sobre entrenamiento**, hay varias soluciones, que forman parte del campo de técnicas denominado "**Regularización**", como:

- ✓ **Contar con más datos:** cuantos más casos pasen por el entrenamiento, más generalista conseguiremos que sea el modelo, y las anomalías tendrán poco impacto.
- ✓ **Reducir el tamaño de la red:** el sobre entrenamiento parece está relacionado con la cantidad de información que es capaz de "almacenar" el modelo a través de los pesos. Reduciendo el número de parámetros del modelo significa **reducir capas o neuronas**. Como criterio general, la idea sería partir de un modelo lo más pequeño posible, e ir ajustando los parámetros en base a la evaluación que vamos haciendo de éste.
- ✓ **Regularización de los pesos:** esta técnica consiste en **limitar el valor** que pueden alcanzar los **pesos**, **como una forma de limitar la capacidad de memoria** del modelo.
- ✓ **Descarte (dropout):** si se hacen nulos, de forma arbitraria, ciertos pesos por capa, durante el entrenamiento, mejora bastante la capacidad de generalización del modelo. **Es la técnica más utiliza y efectiva** tras el decaimiento de los pesos. Consiste en deshacerse de algunas salidas al azar. EL valor (**`dropout rate`**) en Keras **suele estar entre 0.2 y 0.5**.

Dentro de su estrategia de mejora y evolución de Tensorflow, Google lanzó un **kit de herramientas de visualización** con una interfaz muy sencilla, denominado **TensorBoard.** Las principales **funcionalidades** disponibles son:

- ✓ **Monitorizar** de forma visual las **métricas** de **`"Loss"`** **y** la **precisión**.
- ✓ Contar con una **representación de tipo grafo** del modelo.
- ✓ **Gráficas de histogramas** de coeficientes del modelo.
- ✓ Hacer **seguimiento del histórico** de proyecto, **creando perfiles**.

Si **iniciamos TensorBoard antes del entrenamiento**, lo **podremos monitorizar mientras** éste va transcurriendo.

La **normalización** se suele aplicar **centrando los datos en 0**, restando la media a los datos y dividiéndolos por la desviación estándar.

La **normalización por lotes** propone que se aplique esta normalización como una **capa de transformación** de los datos que salen de una capa y van a entrar en otra. Para aplicarla, existe

la capa **`BatchNormalization`** en Keras, que **suele ir detrás de una capa convolucional o neuronal**.

Su principal efecto es que **beneficia al entrenamiento en redes con muchas capas**, ayudando a propagar el gradiente.

Una herramienta disponible en Tensorboard para esta evaluación de los hiperparámetros, es **`Hparams`**, que **presenta de forma gráfica las relaciones entre** estos **parámetros** en las distintas pruebas realizadas.