Práctica 3
Técnicas de validación

TÉCNICAS ESTADÍSTICAS PARA EL APRENDIZAJE II

Máster Universitario en Estadística Computacional
y Ciencia de Datos para la Toma de Decisiones

Técnicas de validación

Los repositorios **UCI Machine Learning** y **Kaggle** albergan una colección de bases de datos ampliamente utilizadas en Machine Learning.

En esta práctica trabajaremos con la **base de datos *Stroke*** disponible en Kaggle**.**

Según la Organización Mundial de la Salud (OMS), el ictus es la segunda causa de muerte en el mundo, responsable de aproximadamente el 11% del total de fallecimientos. El banco de datos *Stroke* se utiliza para predecir si es probable que un paciente sufra un ictus en función de los parámetros de entrada como el sexo, la edad, diversas enfermedades y estatus de fumador. Cada fila de los datos proporciona información relevante sobre el paciente.

Descripción de la base de datos *Stroke*

Variable respuesta: stroke
Tipo de variable respuesta: categórica
Tipo de problema que se quiere resolver: clasificación
Valores perdidos: sí (en la variable bmi)
Número de registros: 5110
Número de variables: 12

Descripción de la base de datos *Stroke*

Variables de la base de datos:

id: identificador único del sujeto
gender: sexo ("Male" / "Female" / "Other")
age: edad del paciente (en años)
hypertension: el paciente tiene hipertensión ("Yes" / "No")
heart_disease: el paciente tiene una enfermedad del corazón ("Yes" / "No")
ever_married: el paciente se ha casado alguna vez ("Yes" / "No")
work_type: tipo de trabajo ("children" / "Govt_jov" / "Never_worked" /"Private" / "Self-employed")
Residence_type: tipo de residencia ("Rural" / "Urban")
avg_glucose_level: media de glucosa en sangre
bmi: índice de masa corporal
smoking_status: nivel de fumador ("formerly smoked" / "never smoked" / "smokes" / "Unknown")
stroke: el paciente sufre un ictus ("Yes"/ "No")

Técnicas de validación

En una base de datos como esta debemos prestar especial atención al **preprocesamiento de datos**.

Tenemos **datos faltantes** y hay **variables categóricas** que debemos tratar (todos los datos deben ser númericos).

Además de todo lo anterior, como siempre, tendremos que **dividir en train/test**, **normalizar** las variables numéricas …

Recordemos que la elección de la **función de activación de la última capa** y de la **función de pérdida** viene determinada en gran medida por el tipo de problema que se desea abordar:

Tipo de problema	Función de activación de la última capa	Función de pérdida
Clasificación múltiple con one-hot	softmax	categorical_crossentropy
Regresión		mse
Clasificación binaria	sigmoid	binary_crossentropy

Técnicas de validación

Una buena práctica es **crear nuestro modelo en una función** para poder tener ordenado el código correctamente.

En esta práctica vamos a crear nuestro modelo en una función y tenerlo disponible en cualquier momento que lo necesitemos.

En esta práctica también vamos a ver algunas formas que podemos utilizar para **evaluar el rendimiento del modelo**:

- División automática de datos de validación
- División manual de datos de validación
- Validación cruzada

División automática de datos de validación

Keras puede separar los datos en entrenamiento/validación y evaluar el rendimiento del modelo en cada época. Podemos hacer esto configurando el argumento ***validation_split*** en la función *fit()* en un porcentaje del tamaño de su conjunto de datos de entrenamiento. Con *validation_split* **se dividen los datos por orden**, tomando primero los datos de entrenamiento y después los datos de validación.

Un valor razonable podría ser 0.2 o 0.33 para el 20% o el 33% de los datos de entrenamiento para validación.

Durante el entrenamiento, podemos ver que la salida detallada en cada época muestra la pérdida y *accuracy* tanto en el conjunto de datos de entrenamiento como en el conjunto de datos de validación.

División manual de datos de validación

Keras también permite especificar manualmente el conjunto de datos que se utilizará para la validación durante el entrenamiento.

El conjunto de datos de validación se puede especificar en la función *fit()* mediante el argumento ***validation_data*** que toma una lista de los conjuntos de datos de entrada y salida.

Como antes, el entrenamiento proporciona una salida detallada que incluye la pérdida y *accuracy* del modelo en los conjuntos de datos de entrenamiento y validación para cada época.

Validación cruzada

El **estándar para la evaluación de modelos de aprendizaje automático** es la validación cruzada, aunque esta técnica a menudo no se usa para evaluar modelos de aprendizaje profundo debido a su gasto computacional.

Por ejemplo, la validación cruzada se usa a menudo con k=5 o k=10, por lo que se deben construir y evaluar 5 o 10 modelos, lo que aumenta considerablemente el tiempo de evaluación de un modelo.

Como sabemos, este método de validación consiste en dividir los datos en k particiones de igual tamaño. Para cada partición i, se entrena un modelo en las k-1 particiones restantes, utilizando la partición i como conjunto de validación. La puntuación final es la media de las k puntuaciones obtenidas.

Cuando el **problema** es lo suficientemente **pequeño o** si se tienen **suficientes recursos** informáticos, la validación cruzada brinda una estimación menos sesgada del rendimiento del modelo.

Ejercicio 1.

Utiliza otras configuraciones y verifica si la red se comporta mejor.

Por ejemplo, puedes probar a cambiar el número de neuronas de la red, añadir otra capa oculta, cambiar el optimizador, probar con un número de épocas diferente…

Evalúa tu nuevo modelo en el conjunto de test y guarda el modelo con la función *save()* de Keras.