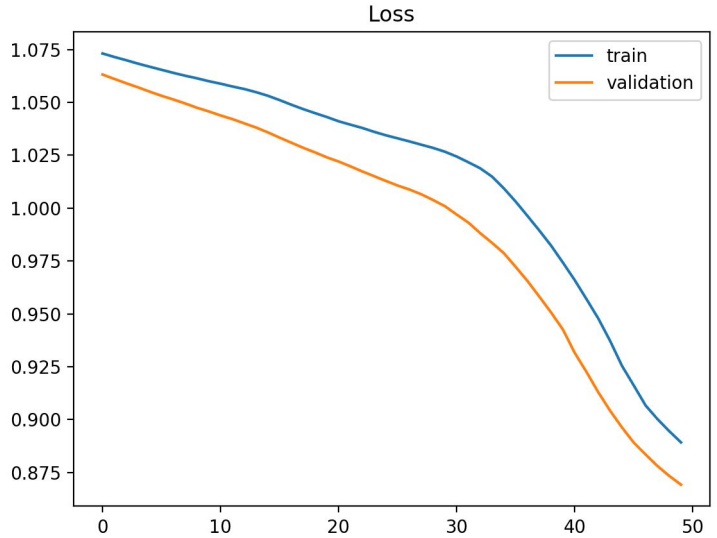
Curvas de aprendizaje

Curso de inteligencia artificial-Intermedio

Objetivo

Estudiar algunas herramientas que nos permita conocer si un modelo está cometiendo sobre-ajuste, sub-ajuste o se está ajustando bien a los datos.



Curvas de aprendizaje
Teoría

Introducción

La pregunta que motiva la construcción de las curvas de aprendizaje es

¿Cómo detectar el sobreajuste?

Primera idea

La manera más directa de detectar sobreajuste es comparar el rendimiento del modelo en el conjunto de entrenamiento con el conjunto de prueba o validación. Si el modelo tiene un **desempeño mucho mejor en el conjunto de entrenamiento** que en el conjunto de prueba, es probable que esté sobreajustando los datos de entrenamiento.

- **Sobreajuste**: Un desempeño excelente en el entrenamiento, pero pobre en el conjunto de prueba.
- **Modelo adecuado**: Un desempeño similar en ambos conjuntos, con valores razonables de error o precisión.

Primera idea

La manera más directa de detectar sobreajuste es comparar el rendimiento del modelo en el conjunto de entrenamiento con el conjunto de prueba o validación. Si el modelo tiene un **desempeño mucho mejor en el conjunto de entrenamiento** que en el conjunto de prueba, es probable que esté sobreajustando los datos de entrenamiento.

- **Sobreajuste**: Un desempeño excelente en el entrenamiento, pero pobre en el conjunto de prueba.
- **Modelo adecuado**: Un desempeño similar en ambos conjuntos, con valores razonables de error o precisión.

Ejemplo en métricas:

- Error en entrenamiento: 2%
- Error en prueba: 15%

Este tipo de discrepancia indica que el modelo ha aprendido demasiado bien los detalles (o incluso el ruido) de los datos de entrenamiento y no es capaz de generalizar a datos nuevos.

Segunda idea

La validación cruzada es una excelente forma de detectar sobreajuste. En este método, el conjunto de datos se divide en varios subconjuntos (o "folds"), y el modelo se entrena varias veces, probándose en un fold diferente cada vez. Si el desempeño varía mucho entre las diferentes particiones, podría ser un indicio de sobreajuste.

Segunda idea

La validación cruzada es una excelente forma de detectar sobreajuste. En este método, el conjunto de datos se divide en varios subconjuntos (o "folds"), y el modelo se entrena varias veces, probándose en un fold diferente cada vez. Si el desempeño varía mucho entre las diferentes particiones, podría ser un indicio de sobreajuste.

- **Sobreajuste**: El rendimiento en algunos folds es mucho peor que en otros (alta varianza).
- **Modelo bien ajustado**: El rendimiento es relativamente consistente entre los diferentes folds.

Tercera idea

Un indicio indirecto de sobreajuste puede surgir si el modelo está funcionando demasiado bien en un conjunto de entrenamiento que contiene **ruido** o datos irrelevantes. Si el modelo está memorizando anomalías o detalles no representativos, es probable que esté sobreajustando.

Una prueba común es **añadir ruido** artificial a los datos de entrenamiento. Si el rendimiento del modelo cae drásticamente, es posible que esté sobreajustando detalles irrelevantes.

Cuarta idea

A veces, probar con una versión más simple del modelo (por ejemplo, menos profundidad en un árbol de decisión, menos neuronas en una red neuronal) y ver si el rendimiento se mantiene o mejora puede ayudar a identificar sobreajuste.

- **Si el modelo más simple tiene un rendimiento similar o incluso mejor en el conjunto de prueba**, el modelo original probablemente estaba sobreajustando.

Técnicas avanzadas

- **Regularización**

  Idea esencial de la regularización: Penalizar parámetros grandes del modelo.

- **Uso de técnicas como Dropout (en redes neuronales)**

  Idea esencial de la regularización: Desactivar neuronas.

Otras alternativas avanzadas

Un indicio indirecto de sobreajuste puede surgir si el modelo está funcionando demasiado bien en un conjunto de entrenamiento que contiene **ruido** o datos irrelevantes. Si el modelo está memorizando anomalías o detalles no representativos, es probable que esté sobreajustando.

Una prueba común es **añadir ruido** artificial a los datos de entrenamiento. Si el rendimiento del modelo cae drásticamente, es posible que esté sobreajustando detalles irrelevantes.

Curvas de aprendizaje

1. ¿Qué son las curvas de aprendizaje?

Las **curvas de aprendizaje** son gráficos que muestran cómo cambia el rendimiento de un modelo de machine learning a medida que varía la cantidad de datos de entrenamiento o los tiempos de entrenamiento. Son una herramienta crucial para diagnosticar problemas de **subajuste (underfitting)** o **sobreajuste (overfitting)** y para determinar si añadir más datos o ajustar los hiperparámetros mejoraría el rendimiento del modelo.

Generalmente, una curva de aprendizaje traza:

- **Error de entrenamiento** (i.e. una métrica) vs. **número de ejemplos de entrenamiento**.
- **Error de validación** (i.e. una métrica) vs. **número de ejemplos de entrenamiento**.

Curvas de aprendizaje

2. Construcción de las curvas de aprendizaje

a) Componentes principales:

1. **Error de entrenamiento**: El error que comete el modelo al predecir los mismos datos que ha usado para entrenar. Inicialmente, el error es bajo, pero aumenta con la complejidad del modelo si hay sobreajuste.
2. **Error de validación**: El error que comete el modelo en datos que no ha visto durante el entrenamiento (conjunto de validación). Este error es clave para medir la capacidad de generalización del modelo.

Curvas de aprendizaje

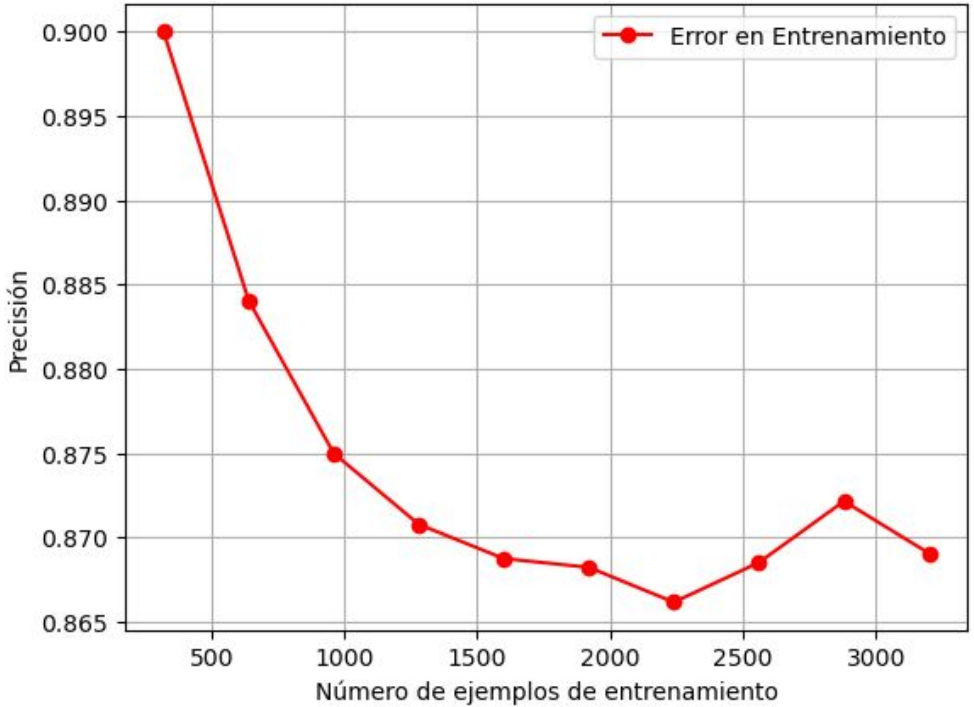
2. Construcción de las curvas de aprendizaje

b) Pasos para construir las curvas de aprendizaje:

1. **Dividir los datos** en un conjunto de entrenamiento y uno de validación (80%-20%, por ejemplo).
2. **Entrenar el modelo** varias veces, aumentando progresivamente la cantidad de datos de entrenamiento en cada iteración.
3. **Registrar el error** (o exactitud) tanto en los datos de entrenamiento como en los de validación para cada conjunto creciente.
4. **Graficar los resultados**:
    - En el eje X: La cantidad de datos de entrenamiento.
    - En el eje Y: El error o la precisión.

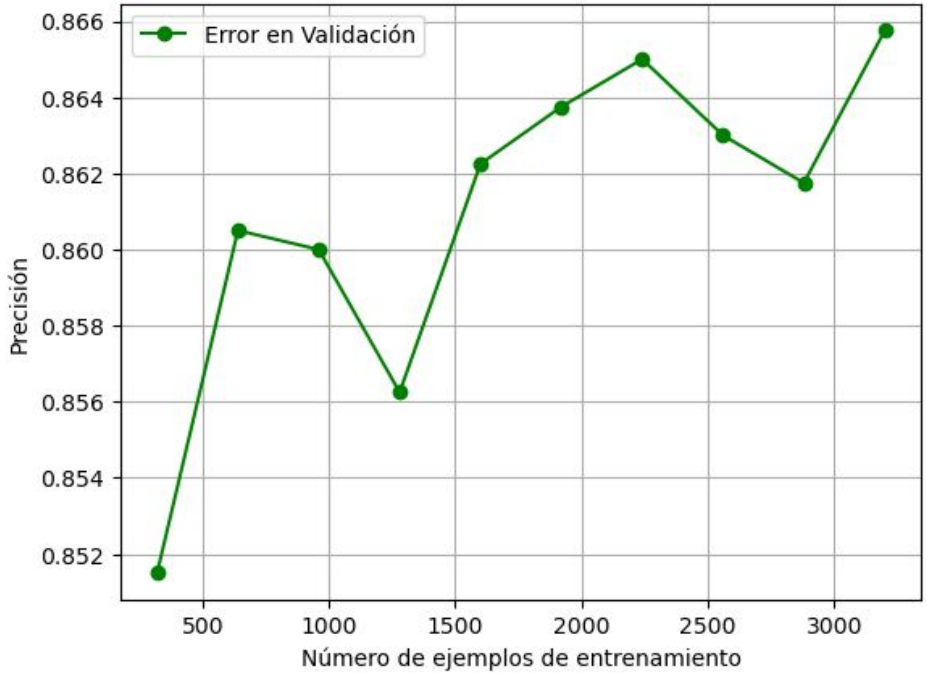
Curvas de aprendizaje

2. Construcción de las curvas de aprendizaje



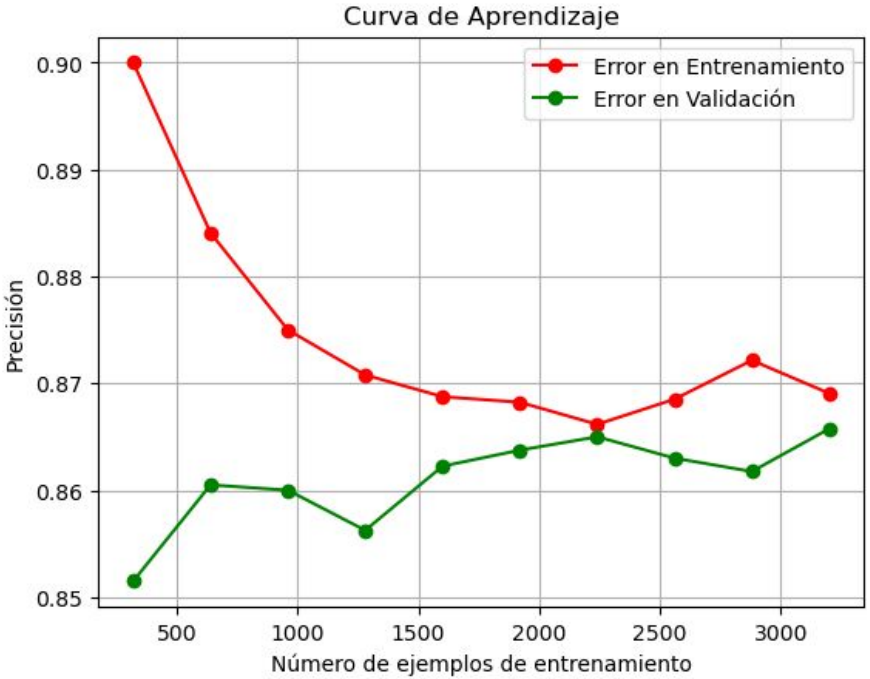
Curvas de aprendizaje

2. Construcción de las curvas de aprendizaje



Curvas de aprendizaje

2. Construcción de las curvas de aprendizaje



Curvas de aprendizaje

3. Interpretación de las curvas de aprendizaje

a) Curva de subajuste (underfitting)

- **Descripción**: Si tanto el error de entrenamiento como el error de validación son **altos** y no disminuyen significativamente a medida que se añade más data, el modelo es demasiado simple (p. ej., un modelo lineal para un problema no lineal).
- **Solución**:
  - Aumentar la **complejidad del modelo** (por ejemplo, usar un modelo no lineal).
  - Añadir **características más relevantes** al conjunto de datos.

Curvas de aprendizaje

3. Interpretación de las curvas de aprendizaje

a) Curva de subajuste (underfitting)

- **Descripción**: Si tanto el error de entrenamiento como el error de validación son **altos** y no disminuyen significativamente a medida que se añade más data, el modelo es demasiado simple (p. ej., un modelo lineal para un problema no lineal).
- **Solución**:
  - Aumentar la **complejidad del modelo** (por ejemplo, usar un modelo no lineal).
  - Añadir **características más relevantes** al conjunto de datos.

Curvas de aprendizaje

b) Curva de sobreajuste (overfitting)

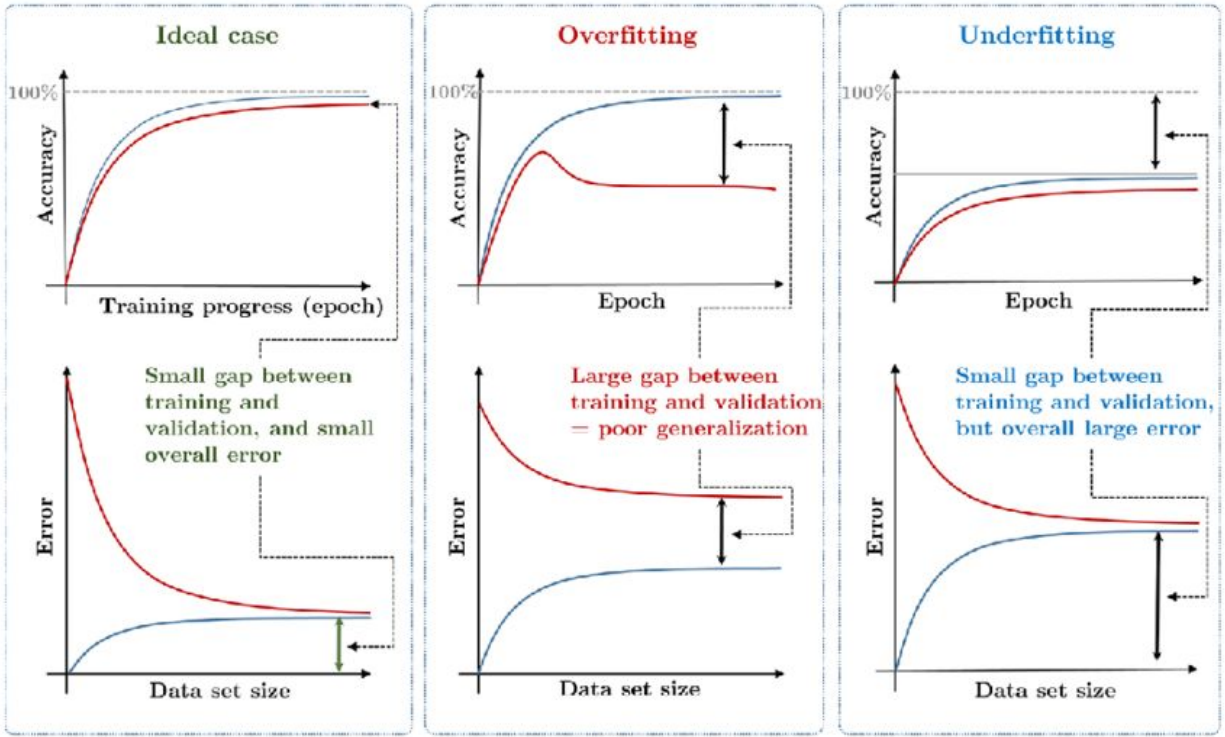
- **Descripción**: Si el error de entrenamiento es muy bajo, pero el error de validación es mucho más alto, el modelo ha memorizado los datos de entrenamiento y no puede generalizar bien a datos nuevos.
- **Solución**:
  - Utilizar **regularización** (L1, L2 o Dropout en redes neuronales).
  - **Reducir la complejidad** del modelo (menos profundidad en árboles de decisión, redes neuronales más pequeñas).
  - Añadir **más datos de entrenamiento**.

Curvas de aprendizaje

c) Curva ideal

- **Descripción**: En una curva ideal, el error de entrenamiento disminuye progresivamente, y el error de validación converge hacia un valor cercano al error de entrenamiento. Esto indica que el modelo está generalizando correctamente.
- **Indicadores de una curva ideal**:
  - Pequeña diferencia entre el error de entrenamiento y el error de validación.
  - El error se estabiliza a medida que aumentan los datos de entrenamiento.

Curvas de aprendizaje



Curvas de aprendizaje

4. Usos de las curvas de aprendizaje

Las curvas de aprendizaje pueden ayudarte a responder preguntas como:

- **¿El modelo es lo suficientemente complejo?**: Si hay subajuste, el modelo puede necesitar mayor complejidad.
- **¿El modelo está sobreajustando?**: Si hay sobreajuste, es necesario simplificarlo o usar más datos.
- **¿Vale la pena recolectar más datos?**: Si las curvas de entrenamiento y validación convergen, añadir más datos probablemente no mejorará el modelo.

Implementación en Python

Nuevas librerías

```
import numpy as np import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve from
sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification from
sklearn.model_selection import train_test_split
```

Importaciones:

- `learning_curve` para implementar las curvas de aprendizaje en Python.

Código k-fold cross validation

```
def plot_learning_curve(estimator, X, y):
```

- **`def`**: Es la palabra clave en Python para definir una función.
- **`plot_learning_curve`**: Es el nombre de la función que estamos definiendo.
- **`estimator`**: Este es el modelo que se va a evaluar (puede ser un clasificador, como un `RandomForestClassifier`, un regresor, etc.).
- **`X`**: Las características (features) o variables independientes. Es el conjunto de datos con el cual entrenas el modelo.
- **`y`**: Las etiquetas o variables dependientes que se quieren predecir. Es el conjunto de respuestas o etiquetas asociadas a los datos de entrenamiento.

Código k-fold cross validation

```
train_sizes, train_scores, val_scores = learning_curve(estimator, X, y, cv=5,
scoring='accuracy', train_sizes=np.linspace(0.1, 1.0, 10))
```

Vamos a desglosar esta línea en partes:

a) **`train_sizes, train_scores, val_scores`**:

- **`train_sizes`**: Son los diferentes tamaños de subconjuntos de datos de entrenamiento que se van a usar para generar la curva de aprendizaje. Representan una fracción del total de datos. Por ejemplo, si tienes 1000 ejemplos y `train_sizes` contiene 100, 200, 300, ..., el modelo se entrenará usando estas cantidades de datos progresivamente.
- **`train_scores`**: Son los resultados (en este caso, las precisiones o accuracies) del modelo cuando se entrena con los diferentes tamaños de datos en el conjunto de **entrenamiento**.
- **`val_scores`**: Son los resultados (precisiones) del modelo evaluado en el conjunto de **validación** (datos no vistos) después de entrenarse con diferentes tamaños de datos de entrenamiento.

Código k-fold cross validation

```
train_sizes, train_scores, val_scores = learning_curve(estimator, X, y, cv=5,
scoring='accuracy', train_sizes=np.linspace(0.1, 1.0, 10))
```

b) `learning_curve`:

La función `learning_curve` de Scikit-learn es la que realmente calcula los datos necesarios para crear la curva de aprendizaje. Toma como entrada el modelo (estimador), los datos y las etiquetas, y devuelve los tamaños de entrenamiento, los resultados en entrenamiento y los resultados en validación.

Vamos a ver los parámetros que le estamos pasando a esta función:

- **`estimator`**: Es el modelo que estamos evaluando. Puede ser cualquier modelo de Scikit-learn, como un `RandomForestClassifier`, `LogisticRegression`, etc.
- **`X`** y **`y`**: Son los datos de entrenamiento y las etiquetas que definimos antes.
- **`cv=5`**: Es el número de particiones o "folds" en la validación cruzada. En este caso, se está usando validación cruzada con 5 particiones.

Código k-fold cross validation

```
train_sizes, train_scores, val_scores = learning_curve(estimator, X, y, cv=5,
scoring='accuracy', train_sizes=np.linspace(0.1, 1.0, 10))
```

b) `learning_curve`:

- `scoring='accuracy'`: Esta es la métrica que estamos utilizando para evaluar el rendimiento del modelo. Aquí se está utilizando la **precisión (accuracy)**, que es adecuada para problemas de clasificación. Si estuviéramos trabajando con un problema de regresión, podríamos cambiarla a métricas como **MSE** (error cuadrático medio) o **R^2**.
- `train_sizes=np.linspace(0.1, 1.0, 10)`: Este parámetro especifica los tamaños de los subconjuntos de datos de entrenamiento. `np.linspace(0.1, 1.0, 10)` genera 10 puntos uniformemente distribuidos entre el 10% y el 100% del total de datos.
    - `np.linspace(0.1, 1.0, 10)` devuelve una lista de 10 valores que representan fracciones del tamaño total de los datos de entrenamiento. Por ejemplo, si tienes 1000 ejemplos, esto generaría subconjuntos con tamaños como 100, 200, 300, ..., hasta llegar al 100% de los datos de entrenamiento.