Búsqueda de hiperparámetros

Aprendizaje Automático

Ingeniería de Robótica Software

Universidad Rey Juan Carlos

Introduction

¿Qué son los hiperparámetros?

- Los hiperparámetros son **valores** que **controlan** el comportamiento del **modelo** de Machine Learning, pero **no se aprenden directamente de los datos**.
- Se establecen **antes del entrenamiento** y afectan a cómo el **modelo aprende**.
- Ejemplos de hiperparámetros:
  - Tasa de aprendizaje (**learning rate**) en redes neuronales.
  - Número de árboles (**n_estimators**) en un Random Forest.
  - Profundidad máxima (**max_depth**) en un árbol de decisión.
  - Parámetro de regularización (**C**) en SVM.

Introduction

Diferencia entre parámetros y hiperparámetros

- **Parámetros**: Son **valores aprendidos** por el modelo durante el entrenamiento.
  - Ejemplo: Pesos y sesgos en una red neuronal.
- **Hiperparámetros**: Son **valores** externos **que no se aprenden**, pero afectan el proceso de aprendizaje.
  - Ejemplo: Número de capas en una red neuronal.

Introduction

¿Por qué son importantes los hiperparámetros?

- Los hiperparámetros tienen un **impacto directo en el rendimiento** del modelo.
- **Sobreajuste (overfitting):** Si el modelo es demasiado complejo (por ejemplo, demasiados árboles o capas), puede ajustarse demasiado a los datos de entrenamiento y no generalizar bien.
- **Subajuste (underfitting):** Si el modelo es demasiado simple (por ejemplo, pocos árboles o baja profundidad), no capturará los patrones importantes de los datos.
- Una buena configuración de hiperparámetros puede:
  - Mejorar la precisión del modelo.
  - Reducir el tiempo de entrenamiento.
  - Lograr un balance entre sobreajuste y subajuste.

Introduction

¿Qué es la búsqueda de hiperparámetros?

- Es el proceso de encontrar la **mejor combinación de hiperparámetros** para **optimizar** el **rendimiento** del modelo.
  - El objetivo es **maximizar la métrica de evaluación** (por ejemplo, accuracy, F1-score, AUC) en un conjunto de validación o mediante validación cruzada.

Introduction

Motivación para la búsqueda de hiperparámetros

- **Problema**: Elegir los hiperparámetros de manera manual (prueba y error) es ineficiente y poco sistemático.
- **Solución**: Usar métodos como Grid Search, Random Search o técnicas avanzadas para automatizar y optimizar este proceso.

Partición de los datos

¿Por qué dividir los datos?

- En Machine Learning, es fundamental **dividir los datos** en diferentes conjuntos para **evaluar el rendimiento del modelo** de **manera justa** y evitar el sobreajuste.
- El objetivo es **garantizar** que el **modelo generalice bien** en datos nuevos (no vistos durante el entrenamiento).

Partición de los datos

Tipos de particiones

- **Conjunto de entrenamiento (Training set):**
  - Se utiliza para **entrenar el modelo** (por ejemplo, pesos en una red neuronal).
  - Representa la mayor parte de los datos (60%-80% del total).
- **Conjunto de validación (Validation set):**
  - Se utiliza para evaluar el modelo durante el entrenamiento y ajustar los **hiperparámetros**.
  - Ayuda a seleccionar el mejor modelo o configuración sin usar el conjunto de prueba.
  - Representa típicamente el 10%-20% de los datos.
- **Conjunto de prueba (Test set):**
  - Se utiliza únicamente al final, para **evaluar el rendimiento del modelo final** en datos completamente nuevos.
  - Representa el 10%-20% de los datos.

Partición de los datos

¿Por qué necesitamos un conjunto de validación?

- Durante la **búsqueda de hiperparámetros**, necesitamos **evaluar diferentes configuraciones del modelo** (por ejemplo, diferentes tasas de aprendizaje o profundidades de árbol).
- Si usamos el **conjunto de entrenamiento** para esta evaluación, el modelo **podría sobreajustarse** a esos datos. Por eso, usamos un conjunto de validación separado.

Partición de los datos

Proceso de búsqueda de hiperparámetros con validación

1. Dividimos los datos en entrenamiento, validación y prueba.
2. Entrenamos el modelo con el conjunto de entrenamiento con una combinación de hiperparámetros.
3. Evaluamos el modelo con el conjunto de validación para cada combinación de hiperparámetros.
4. Seleccionamos la combinación que maximiza la métrica de evaluación (por ejemplo, accuracy, F1-score, etc.).
5. Una vez seleccionados los mejores hiperparámetros, entrenamos el modelo final con los datos de entrenamiento y validación combinados.
6. Evaluamos el modelo final en el conjunto de prueba.

Partición de los datos

Limitaciones de un conjunto de validación

- Si los **datos son pocos**, usar un conjunto de validación fijo puede **reducir** la cantidad de **datos** disponibles para el **entrenamiento**.
- En este caso se usa **validación cruzada** (*k-fold cross-validation*) en lugar de un conjunto de validación separado.

Partición de los datos

¿Qué es la validación cruzada (cross-validation)?

- Es una **técnica** que permite **evaluar el modelo de manera más robusta** cuando los **datos** son **limitados**.
- En lugar de usar un conjunto de validación fijo, **se divide el conjunto de entrenamiento en múltiples particiones** (o "folds") y se entrena y valida el modelo varias veces.

Partición de los datos

Proceso de k-fold cross-validation

1. Dividimos los datos en entrenamiento y prueba.
2. Entrenamos el modelo con el conjunto de entrenamiento con una combinación de hiperparámetros:
   - Para cada combinación de hiperparámetros, entrenamos el modelo k veces, utilizando k-1 folds para entrenamiento y 1 fold para validación.
   - Calculamos la métrica de evaluación (por ejemplo, accuracy, F1-score) en cada fold.
   - Promediamos las métricas obtenidas en los k folds para obtener una evaluación final de esa combinación de hiperparámetros.
3. Seleccionamos la combinación de hiperparámetros que maximiza la métrica promedio.
4. Una vez seleccionados los mejores hiperparámetros, entrenamos el modelo final con los datos de entrenamiento completo.
5. Evaluamos el modelo final en el conjunto de prueba.

Métodos de búsqueda de hiperparámetros

- La **búsqueda de hiperparámetros** es el proceso de **encontrar la mejor combinación de configuraciones** que **optimicen el rendimiento de un modelo** de Machine Learning.
- Existen **varios métodos** para realizar esta búsqueda, cada uno con sus ventajas y desventajas.
  - Búsqueda manual
  - Grid Search
  - Random Search
  - Búsqueda Bayesiana
  - Otros métodos avanzados

Métodos de búsqueda de hiperparámetros

Búsqueda manual

- Consiste en **ajustar los hiperparámetros manualmente**, **probando diferentes combinaciones** basadas en la **intuición, experiencia o conocimiento** del problema.
- Por ejemplo, probar diferentes tasas de aprendizaje o profundidades de árbol y observar cómo afecta al rendimiento del modelo.
  - Ineficiente y poco sistemático.
  - Difícil de aplicar cuando hay muchos hiperparámetros o combinaciones posibles.
  - Depende de la experiencia del usuario, lo que puede llevar a resultados subóptimo.

Métodos de búsqueda de hiperparámetros

Grid Search

- **Busca en un espacio definido de combinaciones** de hiperparámetros.
- Se define un rango o conjunto de valores posibles para cada hiperparámetro, y el método **prueba todas las combinaciones posibles**.
- Por ejemplo, si tienes dos hiperparámetros con 3 valores cada uno, Grid Search probará las 9 combinaciones posibles.
- **Ventajas:**
  - Garantiza encontrar la mejor combinación dentro del espacio definido.
  - Fácil de implementar y paralelizar.
  - Ideal para espacios de búsqueda pequeños y bien definidos.
- **Desventajas:**
  - Computacionalmente costoso, especialmente cuando hay muchos hiperparámetros o valores posibles (combinación explosiva).
  - No es eficiente si algunos hiperparámetros tienen menor impacto en el rendimiento.

Métodos de búsqueda de hiperparámetros

Random Search

- **Selecciona combinaciones a**leatorias de hiperparámetros dentro de un rango definido.
- En lugar de probar todas las combinaciones posibles (como en Grid Search), selecciona un **número fijo de combinaciones aleatorias**.
- **Ventajas:**
  - Más eficiente que Grid Search, especialmente cuando algunos hiperparámetros tienen menor impacto en el rendimiento.
  - Puede explorar un espacio de búsqueda más amplio en menos tiempo.
  - Reduce el costo computacional al limitar el número de combinaciones probadas.
- **Desventajas:**
  - No garantiza encontrar la mejor combinación de hiperparámetros.
  - Puede requerir muchas iteraciones para obtener buenos resultados si el espacio de búsqueda es muy grande.

Métodos de búsqueda de hiperparámetros

Búsqueda Bayesiana

- Utiliza **modelos probabilísticos** para explorar el espacio de búsqueda de manera más inteligente.
- En lugar de probar combinaciones al azar o exhaustivamente, la búsqueda bayesiana **utiliza los resultados de combinaciones anteriores para predecir qué combinaciones podrían ser más prometedoras**.
- Ejemplo: Algoritmos como Tree-structured Parzen Estimators (TPE) o Gaussian Processes.
  - Más eficiente que Grid Search y Random Search, ya que se enfoca en las regiones del espacio de búsqueda con mayor probabilidad de contener la mejor combinación.
  - Reduce el número de combinaciones necesarias para encontrar buenos resultados.

Métodos de búsqueda de hiperparámetros

Métodos avanzados

- **Optimización evolutiva**: Inspirada en algoritmos genéticos, utiliza conceptos como mutación, selección y cruce para explorar el espacio de búsqueda.
- **Ventajas**: Puede encontrar combinaciones óptimas en espacios de búsqueda complejos.
- **Desventajas**: Computacionalmente costoso y más difícil de implementar.