Introducción a PyTorch

José Miguel Buenaposada Biencinto
Iván Ramírez Díaz

Aprendizaje Automático II - Grado en Inteligencia Artificial

November 25, 2024

URJC

Índice

URJC

Índice

URJC

Claves del auge del Deep Learning

1. **Datos**:
   - Disponibilidad masiva de datos etiquetados (y no etiquetados).
2. **Hardware paralelizable**:
   - GPUs y TPUs altamente optimizadas para cálculos intensivos.
   - Uso eficiente de recursos gracias a librerías como CUDA.
3. **Algoritmos**:
   - Algoritmos que permiten optimizar funciones altamente complejas como las redes neuronales profundas.

URJC

Índice

URJC

¿Qué es PyTorch?

- **PyTorch** es una biblioteca de código abierto para el aprendizaje profundo, desarrollada por Facebook AI Research (FAIR).
- Lanzado en **enero de 2017**, PyTorch se ha destacado por su facilidad de uso y flexibilidad.
- Es muy popular en la investigación académica gracias a sus **grafos dinámicos**, que permiten la construcción de modelos adaptativos y comprensibles.

URJC

Comparación con otros frameworks de aprendizaje profundo

- **TensorFlow (lanzado en 2015)**:
  - Inicialmente conocido por sus **grafos estáticos**, lo que complicaba la depuración.
  - En **2019**, la versión 2.0 mejoró la usabilidad, acercándose al enfoque de PyTorch.
- **Keras (lanzado en 2015)**:
  - Biblioteca de alto nivel que utilizaba **TensorFlow como backend**.
  - Simplificó la creación de modelos, pero PyTorch ofrecía **más control** desde el inicio.
- **PyTorch (lanzado en 2017)**:
  - Popularidad creciente desde **2018**, ganando terreno en la comunidad académica.
  - La **versión 1.0 en 2018** integró capacidades para producción.

URJC

Ventajas de PyTorch

- **Simplicidad y legibilidad**: La sintaxis de PyTorch es intuitiva y fácil de aprender para los programadores familiarizados con Python.
- **Soporte para grafos dinámicos**: Introducido desde su lanzamiento en 2017, ideal para investigación y prototipado rápido.
- **Comunidad y ecosistema**:
  - En **2020**, PyTorch superó a TensorFlow en citas académicas.
  - Fuerte soporte de **Meta (Facebook)** y una activa comunidad de desarrolladores.

URJC

Índice

URJC

Pros y contras de Google Colab

- **Ventajas**:
  - **Acceso a GPU y TPU gratuitas**: Acelera el entrenamiento de modelos de aprendizaje profundo.
  - **Entorno preconfigurado**: Incluye PyTorch y otras bibliotecas de forma predeterminada.
  - **Basado en la nube**: Accesible desde cualquier lugar sin necesidad de hardware potente.
- **Desventaja principal**:
  - **Limitaciones en la depuración**: La depuración de código puede ser menos eficiente debido a la ejecución en la nube y a la falta de integración directa con entornos locales.

URJC

Índice

URJC

Tensores en Pytorch

- Un **tensor** puede ser entendido matemáticamente como una matriz multidimensional.
- Es la unidad fundamental para almacenar datos en PyTorch.
- Los tensores en PyTorch son similares a los arrays de **NumPy**, pero con la ventaja de que pueden ser **procesados en GPU**.
- PyTorch proporciona una función `torch.tensor` para crear y manipular tensores.

URJC

Operaciones con tensores

- PyTorch permite realizar una amplia variedad de operaciones matemáticas con tensores:
  - **Operaciones aritméticas básicas**.
  - **Operaciones de álgebra lineal**: como producto punto a punto, transposición, inversión de matrices, svd, etc.
  - **Redimensionado de tensores**: usando funciones como `reshape` y `view`.
  - **Operaciones de agregación**: como `mean`, `sum`, `max`, etc.
  - **Y muchas más**...
- Estas operaciones están altamente optimizadas para ejecutarse de manera eficiente en **GPU**.

URJC

Índice

URJC

Autograd: El motor de diferenciación automática

- **Autograd** es el motor de diferenciación automática de PyTorch, utilizado para calcular los gradientes de las funciones en redes neuronales.
- Permite la optimización de modelos al calcular automáticamente las derivadas de las funciones de pérdida, siempre y cuanto todas las operaciones se hayan definido con funciones de Pytorch sobre tensores.
- Para ello, utiliza **backpropagation**.

URJC

Índice

URJC

Datasets en PyTorch

- En PyTorch, un `Dataset` es una colección de datos que se puede utilizar para entrenamiento y test de modelos.
- `Dataset` es una clase abstracta que se utiliza para representar los datos en un formato adecuado para el entrenamiento.
- En un `Dataset`, se almacenan los datos (por ejemplo, imágenes o texto) y las etiquetas correspondientes (por ejemplo, las clases en un problema de clasificación).
- Un `Dataset` necesita sobrescribir dos métodos clave:
  - `__len__()` : Devuelve el número total de ejemplos en el conjunto de datos.
  - `__getitem__()` : Devuelve un ejemplo específico a partir de un índice dado.
- Estos métodos permiten que PyTorch acceda a los datos y etiquetas de manera eficiente durante el entrenamiento.

URJC

Dataloaders en PyTorch

- El `DataLoader` es una herramienta que se utiliza para cargar los datos de un `Dataset` de manera eficiente y en mini-lotes (batches) durante el entrenamiento del modelo.
- El `DataLoader` proporciona funcionalidades clave como:
  - **Batching**: Divide los datos en mini-lotes (batches), lo que mejora la eficiencia del entrenamiento.
  - **Shuffling**: Mezcla aleatoria de los datos antes de cada época. Se puede, además, definir métodos de muestreo (samplers).
  - **Paralelización**: Utiliza múltiples procesos para cargar los datos de forma concurrente, lo que mejora la velocidad de carga, especialmente para grandes conjuntos de datos.
- El `DataLoader` toma un `Dataset` y devuelve iteradores que proporcionan lotes de datos durante el entrenamiento.
- Es muy útil cuando se trabaja con grandes volúmenes de datos que no caben completamente en la memoria.

URJC

Relación entre Datasets y Dataloaders

- Un `Dataset` contiene los datos y etiquetas, mientras que el `DataLoader` se encarga de cargar esos datos en lotes de manera eficiente.
- El `Dataset` puede ser considerado como la "fuente de datos", mientras que el `DataLoader` es el "gestor de acceso" que organiza cómo esos datos son alimentados al modelo.
- Usar `DataLoader` mejora el rendimiento, permite mayor control sobre el proceso de carga de datos, y facilita el manejo de grandes volúmenes de datos durante el entrenamiento de modelos.

URJC

Índice

URJC

Arquitecturas

- `torch.nn` es un módulo de PyTorch utilizado para construir redes neuronales.
- Facilita la creación de redes neuronales mediante clases de alto nivel que representan las capas de la red.
- Algunas de las principales clases de `torch.nn` incluyen:
  - `nn.Linear`: Capa completamente conectada (fully connected).
  - `nn.Conv2d`: Capa convolucional 2D.
  - `nn.ReLU`: Función de activación ReLU.
  - `nn.Softmax`: Función de activación Softmax.
- `nn.Module` es la clase base para todos los modelos y contiene las funciones para definir las capas y la propagación hacia adelante (`forward()`).

URJC

Índice

URJC

Optimizadores en PyTorch

- Los optimizadores en PyTorch se encuentran en el módulo `torch.optim`.
- Los optimizadores reciben como entrada:
  - **Parámetros del modelo** (los pesos y sesgos de la red neuronal o el modelo que estamos entrenando).
  - **Tasa de aprendizaje** (`lr`), que controla el tamaño del paso en cada actualización de parámetros.
  - Opcionalmente, algunos optimizadores pueden recibir otros hiperparámetros como `momentum`, `beta1`, `beta2`, etc.
- **Ejemplo**:
  - Al crear un modelo en PyTorch, los optimizadores se instancian utilizando el módulo `torch.optim`. Por ejemplo:
    - `optimizer = torch.optim.Adam(model.parameters(), lr=0.001)`

URJC