# Laboratorio: Preparación del entorno

**Aprendizaje**
**Profundo**

Grado en Ingeniería y Ciencia de datos (Universidad de Oviedo)

Pablo González, Pablo Pérez
{gonzalezgpablo, pabloperez}@uniovi.es
Centro de Inteligencia Artificial, Gijón

# Introducción

Durante estas prácticas trabajaremos fundamentalmente usando **Python** y el framework de aprendizaje profundo **PyTorch**.

## Utilización de GPUs

PyTorch permite la ejecución de operaciones con tensores tanto en CPU como en GPU. En este curso aprenderemos a trabajar con este framework, tanto en local (generalmente con CPU), como en servidores especializados (GPU) y también con **Google Colab**, donde Google pone a nuestra disposición recursos gratuitos a través de **Notebooks de Jupyter**.

# Entorno virtuales

Los entornos virtuales son herramientas muy útiles para aislar las librerías que utilizaremos en nuestros experimentos en un **entorno virtual**. Las opciones más utilizadas son los gestores de paquetes **conda** y **pip**.

Para la instalación de **conda**, tenemos dos opciones principales:

- **Anaconda**. Incluye Python, conda, una amplia gama de bibliotecas científicas y herramientas útiles para análisis de datos y ciencia de datos.
- **Miniconda**. Es una versión más ligera y básica de Anaconda que incluye solo Python y conda, sin paquetes adicionales preinstalados.

## Instalación

Tanto **Anaconda** como **Miniconda** se pueden instalar desde el siguiente enlace.

# Instalación de conda

Para instalar conda necesitamos descargar el paquete correspondiente e instalarlo. Podemos hacerlo con las siguientes instrucciones:

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh
```

Una vez instalado y de reiniciar la terminal deberíamos de ser capaces de ejecutar el comando *conda*.

```
(base) nvidia@dl-gpu:~$ conda
usage: conda [-h] [-V] command ...

conda is a tool for managing and deploying applications, environments and packag
es.

Options:

positional arguments:
  command
    clean             Remove unused packages and caches.
    compare           Compare packages between conda environments.
    config            Modify configuration values in .condarc. This is modeled
                      after the git config command. Writes to the user
                      .condarc file (/home/nvidia/.condarc) by default. Use
                      the --show-sources flag to display all identified
                      configuration locations on your computer.
```

# Creación de un entorno virtual

## Entorno virtual base

Conda por defecto crea un entorno virtual llamado **base**. Es una buena práctica no instalar dependencias en este entorno. En su lugar crearemos un entorno virtual nuevo con los paquetes que nos interesen.

Para crear un entorno procederemos de la siguiente manera. En este caso la versión de python elegida será la 3.9:

```
conda create --name mienv python=3.9
```

Para activar este entorno:

```
conda activate mienv
```

Para desactivar el entorno:

```
conda deactivate
```

# Instalación de paquetes

La instalación de paquetes puede hacerse a través de *conda install* o utilizando el gestor de paquetes *pip*. Por ejemplo, vamos a instalar pytorch:

```
pip install torch torchvision torchaudio
```

Podemos probar ahora que se ha instalado correctamente de la siguiente manera:

```
(mienv) nvidia@dl-gpu:~$ python
Python 3.9.17 (main, Jul  5 2023, 20:41:20)
[GCC 11.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.cuda.is_available()
True
>>>
```

## Versiones de los paquetes

Es recomendable a la hora de instalar paquetes indicar la versión, por ejemplo *pip install torch==2.0.1*

# Otros comandos interesantes

Otros comandos interesantes en **conda** son los siguientes:

- `conda list`. Lista los paquetes instalados en el entorno activo.
- `conda env remove --name mienv`. Borra completamente el entorno con el nombre indicado.
- `conda env list`. Lista los entornos de conda que tenemos disponibles.

También es interesante conocer algún comando extra en **pip**:

- `pip uninstall paquete`. Desinstala el paquete indicado.
- `pip freeze`. Devuelve la lista de paquetes instalados por pip.

## Otros comandos

Puedes encontrar una lista de comandos completa para **conda** aquí y una para **pip** aquí.

# Replicación de un entorno

Muchas veces es interesante ser capaces de **salvar un entorno de conda** para poder **restaurarlo en otra máquina**, o que otros científicos de datos puedan replicarlo exactamente para poder ejecutar nuestro código sin esfuerzo. Para ello:

```
conda env export > environment.yml
```

Si observas el fichero environment.yml creado, verás todas las dependencias instaladas por conda inicialmente, más las instaladas con pip.

Para recrear este entorno:

```
conda env create -f environment.yml
```

# Google Colab

Qué es Google Colab:

1. Google Colab es un servicio de Google basado en la nube.
2. Permite ejecutar **notebooks** de Jupyter de forma interactiva.
3. Proporciona recursos computacionales como **CPU**, **GPU** y **TPU**.
4. Colab integra almacenamiento en Google Drive y también Github.
5. Es una herramienta excelente para aprender y desarrollar proyectos de aprendizaje automático.