Práctica 0: Tutorial de uso de Google Colab y Python

Índice

Esta práctica consiste en un tutorial de Google Colab utilizando el extendido lenguaje de programación interpretado Python. Asociado a este tutorial básico, se adjunta un notebook llamado `p0_introduccion.ipynb`. Este archivo incluye una serie de ejemplos de las funcionalidades descritas en este tutorial, y que se irán utilizando durante el curso. Cada ejemplo incluye un ejercicio. Aunque no es obligatoria la entrega de esta práctica inicial, si se recomienda realizar los ejercicios en la medida de lo posible para entender mejor las herramientas descritas.

1. ¿Qué es Google Colab?

Google Colab (abreviatura de Google Colaboratory) es una plataforma gratuita de Google que permite escribir y ejecutar código Python en un entorno basado en Jupyter Notebooks. A diferencia de un entorno local, todo el procesamiento se realiza en la nube, lo que significa que no necesitas instalar nada en tu ordenador. Además, Colab permite compartir fácilmente los cuadernos con otros usuarios, facilitando el trabajo colaborativo, la docencia y la investigación. Google Colab es un entorno de cuadernos (notebooks) en la nube que permite escribir y ejecutar código Python directamente desde el navegador, con acceso gratuito a recursos de computación (incluyendo GPUs). Está basado en Jupyter Notebook.

2. Primeros pasos

Antes de comenzar a programar, es importante familiarizarse con la estructura del entorno de Google Colab y su funcionamiento básico. Esta sección te guiará a través del proceso de apertura de un cuaderno, su estructura principal y las operaciones básicas para trabajar con él.

2.1. Acceder a Google Colab

Para comenzar:

1. Ve a `https://colab.research.google.com/`
2. Inicia sesión con tu cuenta de Google.

3. Selecciona “Nuevo cuaderno” o abre uno existente desde tu Google Drive.

2.2. Estructura de un cuaderno

Un cuaderno contiene dos tipos principales de celdas:

- **Celdas de código**: donde escribes y ejecutas código Python.
- **Celdas de texto (Markdown)**: donde puedes escribir explicaciones, ecuaciones en LaTeX, listas, etc.

3. Uso básico de celdas

Los cuadernos en Colab se organizan en bloques llamados celdas. Puedes alternar entre celdas de código y celdas de texto para combinar explicaciones, fórmulas y código en un solo documento interactivo. Esta forma de trabajo facilita el aprendizaje y la documentación del proceso de resolución de problemas o experimentos computacionales.

3.1. Código Python

Puedes escribir cualquier código Python en una celda. Ejemplo:

```
import numpy as np
print(np.pi)
```

3.2. Texto en Markdown

Puedes escribir explicaciones con formato, por ejemplo:

- # Título
- **Negrita**
- *Cursiva*
- Listas numeradas o con viñetas

3.3. Fórmulas matemáticas con LaTeX

Puedes escribir expresiones matemáticas con LaTeX dentro de celdas de texto usando:

- En línea: $E = mc^2$
- En bloque:

$$x(t) = A\cos(\omega t) + B\sin(\omega t)$$

3.4. Guardar y compartir

Google Colab está integrado con Google Drive, lo que permite guardar automáticamente tus avances. También puedes descargar tus cuadernos o compartirlos con otras personas mediante un enlace. Esto lo convierte en una herramienta ideal para el trabajo colaborativo o la entrega de tareas.

- Los cuadernos se guardan automáticamente en tu Google Drive.
- Puedes exportarlos como PDF o .ipynb desde `Archivo >Descargar`.
- Puedes compartirlos como cualquier archivo de Google Drive.

4. Gráficas con `matplotlib`

Una parte fundamental de la simulación y el análisis de sistemas dinámicos es la representación gráfica de los resultados. En Python, la biblioteca `matplotlib` permite crear gráficos de forma sencilla y potente. Puedes representar datos temporales, relaciones entre variables o incluso diagramas de fase.

Ejemplo de generación de gráficas:

```
import matplotlib.pyplot as plt
import numpy as np

t = np.linspace(0, 10, 200)
x = np.sin(t)

plt.plot(t, x)
```

```
plt.xlabel('Tiempo')
plt.ylabel('Amplitud')
plt.title('Señal senoidal')
plt.grid(True)
plt.show()
```

5. Interactividad con `ipywidgets`

La interactividad permite explorar el comportamiento de un modelo dinámico modificando parámetros sin tener que cambiar manualmente el código. Esto es especialmente útil en contextos docentes o exploratorios, y puede lograrse en Colab con la ayuda de la biblioteca `ipywidgets`, que ofrece sliders, menús desplegables y otros controles gráficos.

Puedes usar controles interactivos como sliders:

```
from ipywidgets import interact
import matplotlib.pyplot as plt
import numpy as np

def senal(A=1.0):
    t = np.linspace(0, 10, 200)
    y = A * np.sin(t)
    plt.plot(t, y)
    plt.grid(True)
    plt.show()

interact(senal, A=(0.1, 5.0, 0.1))
```

6. Uso de GPU en Google Colab

Google Colab permite el uso gratuito de GPU y TPU para tareas de alto rendimiento como entrenamiento de modelos de aprendizaje automático.

6.1. Cómo activar una GPU

1. Abre tu cuaderno en Google Colab.

2. Ve al menú `Entorno de ejecución >Cambiar tipo de entorno de ejecución`.

3. En "Acelerador por hardware", selecciona `GPU` (o `TPU` si lo deseas).

4. Haz clic en "Guardar".

6.2. Verificar si se está usando GPU

Puedes ejecutar el siguiente código para comprobar si tienes acceso a una GPU:

```
import tensorflow as tf
print("GPU disponible:", tf.config.list_physical_devices('GPU'))
```

También puedes usar PyTorch:

```
import torch
print("¿GPU disponible?", torch.cuda.is_available())
print("Nombre de la GPU:", torch.cuda.get_device_name(0) if torch.cuda.is_availabl
```

¿Por qué usar una GPU?

Las unidades de procesamiento gráfico (GPU) están diseñadas para realizar muchas operaciones en paralelo, lo que las hace especialmente útiles para tareas que implican procesamiento masivo de datos o cálculos matriciales. Esto incluye, entre otras:

- Simulación numérica de sistemas dinámicos complejos.
- Entrenamiento y evaluación de modelos de aprendizaje profundo.
- Procesamiento de imágenes o grandes volúmenes de datos.

Diferencias entre CPU, GPU y TPU

- **CPU (Unidad Central de Procesamiento)**: ideal para tareas secuenciales y de propósito general. Tiene pocos núcleos, pero potentes.
- **GPU (Unidad de Procesamiento Gráfico)**: diseñada para tareas en paralelo. Tiene cientos o miles de núcleos pequeños y eficientes.

- **TPU (Unidad de Procesamiento Tensorial)**: diseñada por Google específicamente para acelerar cargas de trabajo de aprendizaje automático, especialmente con TensorFlow.

Recomendaciones de uso

- Usa GPU solo si el cálculo lo justifica. Para scripts simples o ligeros, una CPU es suficiente.
- Evita ejecutar muchas tareas simultáneas en GPU; podrías alcanzar el límite de uso gratuito.
- Aprovecha bibliotecas optimizadas como TensorFlow, PyTorch o CuPy para utilizar eficientemente la GPU.

7. Librerías para Aprendizaje Automático e Inteligencia Artificial

En Google Colab es posible aprovechar librerías muy potentes para aprendizaje automático y modelado inteligente de sistemas. Algunas de las más relevantes son:

7.1. Scikit-learn

Scikit-learn es una librería en Python ampliamente utilizada para el aprendizaje automático clásico. Ofrece algoritmos de clasificación, regresión, clustering, reducción de dimensionalidad y validación de modelos.

```
from sklearn.linear_model import LinearRegression
from sklearn.datasets import load_boston

# Cargar datos
X, y = load_boston(return_X_y=True)

# Definir modelo
modelo = LinearRegression()
modelo.fit(X, y)
```

```
print("Coeficientes:", modelo.coef_)
```

7.2. TensorFlow

TensorFlow es un framework de código abierto desarrollado por Google para construir y entrenar redes neuronales profundas. Es altamente escalable y se integra con GPU/TPU en Colab.

```
import tensorflow as tf

# Definir un modelo secuencial simple
modelo = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1)
])

modelo.compile(optimizer='adam', loss='mse')
```

7.3. PyTorch

PyTorch es otro framework muy popular, desarrollado por Facebook, enfocado en la flexibilidad y facilidad de uso para investigación y prototipado rápido.

```
import torch
import torch.nn as nn

# Definir un modelo simple
class Modelo(nn.Module):
    def __init__(self):
        super(Modelo, self).__init__()
        self.fc1 = nn.Linear(10, 50)
        self.fc2 = nn.Linear(50, 1)
    def forward(self, x):
        x = torch.relu(self.fc1(x))
        return self.fc2(x)

modelo = Modelo()
```

7.4. Keras

Keras es una API de alto nivel para construir y entrenar modelos de aprendizaje profundo de manera sencilla, integrándose dentro de TensorFlow.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

modelo = Sequential([
    Dense(64, activation='relu'),
    Dense(1)
])
```

Estas librerías permiten construir desde modelos sencillos hasta arquitecturas complejas de aprendizaje profundo, facilitando la simulación de sistemas inteligentes en Colab.