Applied Deep Learning with Tensorflow and Pytorch Chapter 1

Introduction

- Python: programming language, Conda: package manager
- Colab: free GPU testing environment, allows you to write and execute Python in your browser
- Jupyter: service to keep easy track of your experiments

Installation

- **Python Installation:** sudo apt-get install python3.8 python3-pip
- **Miniconda Installation:** sh Miniconda3-latest-Linux-x86_64.sh -b; ∼ /miniconda3/bin/conda init; conda create –name d2l python=3.8 -y
- **D2L Notebooks Downloading:** mkdir d2l-en; cd d2l-en; curl https://www.d2l.ai/d2l-en.zip -o d2l-en.zip; unzip d2l-en.zip; rm d2l-en.zip;
- **D2L Notebooks Activation:** conda activate d2l
- **PyTorch Installation:** pip install torch torchvision
- **Tensorflow Installation:** pip install tensorflow tensorflow-probability
- **D2L Installation:** pip install -U d2l
- **Running Jupyter Notebook:** jupyter notebook
- **Conda Environment (e.g. D2L) Deactivation:** conda deactivate

Colab

- **Go to:** colab.research.google.com
- **Google Drive Linking:** from google.colab import drive drive.mount('/content/drive/')
- **Reaching your Gdrive:** !ls "/content/drive/My Drive/"
- **Upload python file " abc.py " to Gdrive and run with Colab:** !python3 "/content/drive/My Drive/Colab Notebooks/abc.py"
- **Run with Google Colab to Download " abc.py " from Google Drive:** from google.colab import files files.download('/content/drive/My Drive/Colab Notebooks/abc.py')

Overview of Deep Learning

- Deep learning can be extremely valuable if the data is high dimensional; not a single feature, but combination of features are very informative; and there is a large amount of training data.
- For tabular data, deep learning is rarely the correct model choice
- Possible use cases for deep learning are

Computer vision: image classification, object detection, image segmentation etc.
Natural language processing: machine translation, sentiment analysis, email classification etc.
Speech: recognition and generation

Tensorflow

- Open source software library for deep learning, released in 2015
- Originally developed by Google as a single infrastructure for machine learning

Installation
CPU: !pip install tensorflow
GPU: !pip install tensorflow-gpu
Import: import tensorflow as tf
Version: tf.__version__

Tensorflow 1.x

- You need to assemble a graph and run a session for computation
- Graph: a = tf.add(3, 5)
- Session: sess = tf.Session(); sess.run(a); sess.close();

Tensorflow 2.x

- Eager Execution: a = tf.add(3, 5)

Keras

- User-friendly, easy to extend high-level API for Tensorflow
- Keras is built-in to Tensorflow 2, accessible through tf.keras
- Model types: Sequential Model, Functional API, Models via Subclassing
- A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.
- The functional API can handle models with non-linear topology, shared layers, and even multiple inputs or outputs.
- **Data Installation:** keras.datasets.mnist.load_data()
- **Specification of training configuration:** model.compile()
- **Training the model** model.fit()
- **Evaluation of the model:** model.evaluate()
- Keras is a high-level API and it is not fully flexible. Non-standard layers or non-standard training procedure might not be possible in Keras: tf.GradientTape() allows you to tailor your training

Limitations of Tensorflow

- not very flexible for experimental projects due to decreasing popularity in academia
- hard to debug in complex projects
- the whole GPU is allocated for single model
- many packages for TF1 are deprecated in TF2

PyTorch

- PyTorch is a deep learning research platform, python-based scientific computing framework and a replacement for NumPy to use the power of GPUs
- developed by the Facebook artificial-intelligence research group

Installation

CPU: !pip install torch==1.8.0+cpu torchvision==0.9.0+cpu torchaudio==0.8.0 -f https://download.pytorch.org/whl/torch_stable.html
GPU: !pip install torch==1.8.0+cu111 torchvision==0.9.0+cu111 torchaudio==0.8.0 -f https://download.pytorch.org/whl/torch_stable.html
Import: import torch
Version: print(torch.__version__)

Three levels of abstraction:

- **Tensor:** Imperative ndarray; but runs on GPU
- **Variable:** wrapper around a PyTorch Tensor, and represents a node in a computational graph. If *x* is a variable:
  x.data gives its value
  x.grad is another Variable holding the gradient of x with respect to some scalar value
  x.grad-fn is a function object, which created the Variable
- **Module:** A neural network layer; may store learnable weights

Autograd

- Autograd is a PyTorch package for the differentiation for all operations on Tensors.
- **autograd.Variable** is the central class of the package. It wraps a Tensor and supports nearly all of the operations defined on it.
- With **.backward()** all the gradients computed automatically.

History of DL Tools