Assignment 06
Neural Networks
EECE-6822 Machine Learning

Cris Ababei
Electrical and Computer Engr., Marquette University

1. Objective

The objective of this activity include: (1) to run several code examples that work with neural networks implemented in Tensorflow, in Pytorch, or directly in Pyton from scratch and (2) investigate train/test split of datasets for NNs.

2. Prerequisite Readings

Murphy
- 13-13.4.3 on neural networks

Abu-Mostafa
- 1.1.2 on the perceptron

Geron
- Ch.10: intro to neural networks

Raschka
- Ch.11 and Ch.12: MLP from scratch in Python and in Pytorch

3. Code Examples

Example 1: Neural Networks with Keras TensorFlow

This is the example code from Ch.10 from Aurelian Geron's book.
[*B3-Geron] Aurelien Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly, 2022.
Open in your google colab and go through all the code; observe all details. Of course, you should read first the chapter itself from the book, before going through the code.
https://github.com/ageron/handson-ml3/blob/main/10_neural_nets_with_keras.ipynb
The example starts first with the Perceptron, in SciKit-Learn, applied to the Iris Flowers Dataset, using only two input features (of the four) and one class only, Iris Setosa.

It then shows the use of: from sklearn.neural_network import MLPRegressor, to implement a MLP regressor for the California Housing dataset.

Then, it uses: from sklearn.neural_network import MLPClassifier, to use MLP for classification on the Iris Flowers Dataset.

Then, and this is what we wanted to see, it shows how to implement MLPs with Keras for classification on the Fashion MNIST dataset. The training set contains 60,000 grayscale images, each 28x28 pixels. The labels are the class IDs (represented as uint8), from 0 to 9. Th model is created easily using the Sequential API:

```
model = tf.keras.Sequential()

model.add(tf.keras.layers.InputLayer(input_shape=[28, 28]))

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(300, activation="relu"))

model.add(tf.keras.layers.Dense(100, activation="relu"))

model.add(tf.keras.layers.Dense(10, activation="softmax"))
```

```
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])
```

Please observe the use of model.summary().

Then, the example shows how to build a **Regression MLP** Using the Sequential API for the California housing dataset. You could stop here with this example. However, the rest of the example covers some more advanced or useful to know topics including: Building Complex Models Using the Functional API, Using the Subclassing API to Build Dynamic Models, Saving and Restoring a Model, Using Callbacks, Using TensorBoard for Visualization, and Fine-Tuning Neural Network Hyperparameters.

Example 2: NNs from Scratch in Python and in Pytorch

[A] Multilayer Artificial NN from Scratch

This code example is from ch.11 from (uses the MNIST digits 0-9 dataset):
[*B3-Raschka] Sebastian Raschka, Yuxi Liu, and Vahid Mirjalili, Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python, Packt Publishing, 2022.
Again, the source code of this book is in this GitHub repository: https://github.com/rasbt/machine-learning-book
Follow the details presented in Assignment 2 on how to run code from Raschka book. We will look at the notebook: https://github.com/rasbt/machine-learning-book/blob/main/ch11/ch11.ipynb
However, to execute the code, I found the easiest to actually run directly the Python code corresponding to this notebook in Anaconda Spyder:
EECE6822\Book_Code_Raschka\machine-learning-book-main\ch11\ch11.py

At this time, you should run the notebook or run directly the Python code. You should also first read the chapter from the book itself, before running the code. Essentially, in this example, we see the development in Python of a multilayer NN:

- Gaining a conceptual understanding of multilayer NNs
- Implementing the fundamental backpropagation algorithm for NN training from scratch
- Training a basic multilayer NN for image classification

I find this example particularly helpful for understanding the Backpropagation algorithm and the math behind it.
NOTE: You may need to change the following line if you get an error while using fetch_openml:

```
X, y = fetch_openml('mnist_784', version=1, return_X_y=True, parser="liac-arff")
```

[B] Building an NN model in PyTorch

This code example is in ch.12 (part 2) from:
[*B3-Raschka] Sebastian Raschka, Yuxi Liu, and Vahid Mirjalili, Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python, Packt Publishing, 2022.
In this example we explore the following topics:

- How PyTorch improves training performance
- Working with PyTorch's Dataset and DataLoader to build input pipelines and enable efficient
- model training
- Working with PyTorch to write optimized machine learning code
- **Using the torch.nn module to implement common deep learning architectures conveniently**
- Choosing activation functions for artificial NNs

At this time, run the code as a Jupyter notebook or directly as Python code in Anaconda Spyder.

https://github.com/rasbt/machine-learning-book/blob/main/ch12/ch12_part2.ipynb

EECE6822\Book_Code_Raschka\machine-learning-book-main\ch12\ch12-part2.py

NOTE: You may need to add to the code also this import:

```
from IPython.display import Image as IPythonImage
```

Also change this line to look like this:

```
model_new = torch.load(path, weights_only=False)
```

Example 3: MLPs in TF2 and Pytorch

These are examples suggested by K. Murphy on his github repository for the textbook:
https://github.com/probml/pyprobml/tree/master/notebooks/book1/13

[A] MNIST image classification with MLP - TF2

It illustrates the use of MLP on MNIST for digits 0-9 classification using **TensorFlow (TF) 2.0**.
The actual code that you should open in your google colab is this:
https://colab.research.google.com/github/probml/pyprobml/blob/master/notebooks/book1/13/mlp_mnist_tf.ipynb
The example shows how easy is to construct a NN model. Train, and test it in Keras TF2:

```
model = keras.Sequential(
  [
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax),
  ]
)
model.summary()
model.compile(optimizer="adam",
  loss="sparse_categorical_crossentropy", metrics=["accuracy"])
```

The model takes images 28x28 pixels of hand written digits 0-9 and implements a classifier using softmax activation function in the output layer.

[B] CIFAR image classification with MLP - PyTorch

This is another example suggested by K. Murphy on his github repository for the textbook:
https://github.com/probml/pyprobml/tree/master/notebooks/book1/13
It illustrates the use of MLP on CIFAR dataset for image classification **in PyTorch**.
This example illustrates how to fit an MLP to a two-class version of CIFAR. It is adapted from the code from Chap. 7 of the Deep Learning With PyTorch book:
https://github.com/deep-learning-with-pytorch/dlwpt-code/tree/master/p1ch7
The actual code that you should open in your google colab is this:
https://colab.research.google.com/github/probml/pyprobml/blob/master/notebooks/book1/13/mlp_cifar_pytorch.ipynb
The dataset CIFAR has 50000 images 32x32 pixels, each with 3 color channnels. We standardize the features by computing the mean and std of each channel, averaging across all pixels and all images to help the optimization.
It has 10 classes:

```
class_names = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse",
"ship", "truck"]
```

This example works with only two classes actually, airplane and bird. It shows how to build a shallow fully connected network:

```
img, label = cifar10[0]
img = img.view(-1)
ninputs = len(img)
nhidden = 512
nclasses = 2
torch.manual_seed(0)
model = nn.Sequential(nn.Linear(ninputs, nhidden), nn.Tanh(),
  nn.Linear(nhidden, nclasses), nn.LogSoftmax(dim=1))
print(model)
```

Example 4: mlm tutorials and code examples on NNs

In this part, we will look at several tutorials from **machinelearningmastery (mlm)**. Read the following tutorials and run the code where applicable:

--Develop a Neural Network for Cancer Survival Dataset

https://machinelearningmastery.com/neural-network-for-cancer-survival-dataset/

--TensorFlow 2 Tutorial: Get Started in Deep Learning with tf.keras

https://machinelearningmastery.com/tensorflow-tutorial-deep-learning-with-tf-keras/

--PyTorch Tutorial: How to Develop Deep Learning Models with Python

https://machinelearningmastery.com/pytorch-tutorial-develop-deep-learning-models/

--How to Improve Performance With Transfer Learning for Deep Learning Neural Networks

https://machinelearningmastery.com/how-to-improve-performance-with-transfer-learning-for-deep-learning-neural-networks/

4. Assignment

The objective is to investigate how train/test split of a dataset impact the performance of a NN model.

You will reuse code from a given example, included within .zip archive for this lecture. Specifically, this is the Jupyter Notebook tha you should run and study:
TF_MNIST_Classification.ipynb
In this example, we use a dense NN to do digit Classification. We work with MNIST handwritten digits dataset (https://www.tensorflow.org/datasets/catalog/mnist). The MNIST database of handwritten digits includes a training set of 60,000 28x28 grayscale images of the 10 digits along a test set of 10,000 images. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

The example splits the MNIST dataset of 70,000 images into 60,000 for training, 0 for validation, and 10,000 for testing. This is roughly an 85%-0%-15% split for training, validation and testing.

Part 1: You must modify and redo this example four different times (each time, run the same experiment 5 times and calculate averages and std that you will report) to investigate the following splits:
60-0-40%
70-0-30%

80-0-20%
90-0-10%
In all four cases, during the "Train the model" section record the values of "Training Data Accuracy:". Also, save each of the plots loss vs. epoch in this section (one only for each of the five runs of the same experiment). You will have to include them in your report – all plots within the same figure, for ease of comparison!
From the "Testing the trained model" section, record the values of "Testing Data Accuracy:".
Then, create two additional separate plots:

- Training Data accuracy Vs. Splits
- Testing Data Accuracy Vs. Splits

Report the results and provide a discussion.

Part 2: In this part, you must keep the dataset split that provided the best results in Part 1. Then, change the activation function of the hidden layer to investigate: **tanh** and **ReLU** (currently in the code) activation functions. Train the model for each case and record testing data accuracy. Report the results, compare them, and provide a discussion.

5. Deliverables

You must write (typed) a report and upload it as a PDF file on D2L. The report should be named "**hw6_report_LastName.pdf**". You should also create a .zip archive with all your code and implementations of all parts of the assignment. Upload also this archive .zip file with the name "**hw6_implementation_code_LastName.zip**" to D2L. Hence, your D2L should contain two items: the report and the .zip file. **Do not include the report inside the .zip and upload only the .zip. They should be two separate items!**

The report should include the following sections and subsections. Make sure section titles are in bold font and pages are numbered.

1) **Title + course info + your name**
2) **Summary.** Describe in one paragraph what the objective of the assignment is.
3) **Neural Networks.** Write a short paragraph to describe what a neural network is.
4) **Description of Experiments and Discussion.** Describe the experiments you did. All tables and figures should be numbered and should have captions. All plots in all figures should have axes labels and titles. Present a meaningful discussion with the interpretation of the results you obtained. Explain if you expected the results or not; discuss the intuition behind it.
5) **Conclusion.** Present your conclusions; highlight what are your main takeaways that you learned from this assignment. Describe what issues you encountered and how you solved them.
6) **References.** Include all references that you used, as a numbered list. Cite them in the report itself; do not just list them! If your report has References that are not cited in the report, points will be deducted!
7) If your report has References that are not cited in the report, points will be deducted!