CPSC 585 - Artificial Neural Networks - Spring 2020

Project 2, due April 22

Last updated Tuesday, April 14, 5:20 pm PDT

Platforms

This project has the same platform requirements as Project 1, but access to a GPU or TPU will speed up training significantly. If you do not have access to a physical machine with a GPU, there are cloud services that offer free access to GPU or TPU-enabled versions of Jupyter, including Google Colaboratory (GPU, TPU), Kaggle Notebooks (GPU, TPU), and Gradient Community Notebooks.

Libraries

You will need Keras and TensorFlow. While Keras is capable of using multiple backends, the current recommendation from the Keras team is to use the `tf.keras` module built into TensorFlow.

You may also wish to use other Python libraries such as scikit-learn and pandas.

Warmup

Examples in the Nielsen book use the MNIST dataset of handwritten digits, which is also included with Keras. Since the dataset is reasonably small and easily available, you may wish to use this dataset while setting up and getting familiar with Keras and your notebook environment.

Keras includes several examples using MNIST, in particular `mnist_mlp.py`. Try running this code (but don't forget to switch to `tf.keras` instead) and verifying that you can match the accuracy of 98.40% claimed in the comments.

Compare the network architecture in `mnist_mlp.py` with the architecture that Nielsen describes in the section on dropout in Chapter 3 of the Nielsen book

Dataset

The Extended MNIST or EMNIST dataset expands on the original MNIST, adding handwritten letters as well as additional samples of handwritten digits. There are several "splits" of the data by various characteristics. We will be using the “EMNIST Letters” dataset, which contains data split into 26 classes, one for each letter in the English alphabet.

Extracting and loading the dataset

Download the Matlab format dataset and extract `matlab/emnist-letters.mat`. This file can be opened by `scipy.io.loadmat()`, but you will need some guidance in figuring out how to navigate the structure. See this answer on StackOverflow for details on retrieving the training, validation, and test sets.

Dataset classes

The EMNIST Letters dataset folds together both upper- and lowercase letters into a single class. The `data['mapping']` field maps from class numbers to ASCII codes of the corresponding letters. For example, class 1 maps to ASCII codes 65 and 97 (`'A'` and `'a'`). This may affect your network design.

An MLP for EMNIST Letters

Begin by applying the network architecture from `mnist_mlp.py` to the EMNIST Letters data, modifying the number of classes. What accuracy do you achieve? How does this compare with the accuracy for MNIST?

Keeping the same number of `Dense` layers in the network (i.e. a standard MLP with one hidden layer), modify the architecture to improve the accuracy. You will need to decide on an appropriate number of neurons in the hidden layer. Keep in mind that:

- There are 26 classes rather than 10, so you will likely need a larger hidden layer than the network for recognizing digits.
- In addition to having more classes, the EMNIST Letters data mixes upper- and lowercase letters within each class, so even with enough neurons in the hidden layer, your accuracy is likely to be lower. See the details in the EMNIST paper for the kind of performance you might reasonably expect.

Once you have settled on the size of the hidden layer, use the techniques you learned in Chapter 3 of the Nielsen book to obtain the highest accuracy you can on the validation set.

When finished, evaluate your results on the test set. Compare the performance on the test set of your original and final networks for EMNIST Letters.

Submission

Submit your project by uploading your `.ipynb` file and any other relevant artifacts to the `project2/` subdirectory of the folder that will be shared with you on Dropbox. Do **not** submit copies of the dataset. You may also share a link to a version of your notebook hosted in the cloud, but you must download and submit the `.ipynb` file as well. A submission sheet will be provided in the same Dropbox folder.

You may work alone, or make a single submission for a team of 2-3 students. If you work in a team, only one submission is required, but for safety consider uploading copies to each team member's submission folder. (Make certain, however, that the copies are the same in each case; the instructor will not attempt to ascertain which is the “real” submission.)

To finalize your submission, upload your files and fill out the sheet with the requested information by the end of class on the due date. Do not respond to the questions in the submission sheet using the Dropbox commenting feature; modify the document itself. Failure to follow any of these submission instructions exactly will incur a **10%** penalty on the project grade for all team members.