HW 1 Report

Date: Jan-26-2025 15:06
Class: CSC-487 Deep Learning

Code Explanation

Briefly explain your solution and any design choices you made that weren't specified in the instructions.

In this homework assignment, I implemented two different types of classifiers for the tree dataset. This was using both scikit-learn and PyTorch. The data was pre-processed with PCA to reduce the amount of features.

I didn't really do any design changes, but for inspecting the data I added a few charts to see class distribution and spread of data in the dataset.

Clearly describe any external sources that were used (e.g. websites or AI tools) and how they were used.

I used both scikit-learn.org and pytorch.org documentation for their classifiers, modules, and a few other parameters that their functions use.

For setting up the PyTorch modules, I used this tutorial:
https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html

I had some assistance from Google Colab's Google Gemini autocomplete for the accuracy function since I was a bit confused about how to calculate correct predictions with the outputs and shape of the data.

Everything else was pretty similar to the labs that we have done, so it was pretty straightforward to complete.

Discussion

? What is the shape and data type of each provided matrix?

- The `train_features` matrix's shape is `(15707, 426)` and has a data type of `int16`. This means that there are `15,707` rows and `426` columns.
- The `train_labels` matrix shape is `(15707,)` and has a data type of `uint8`. This means there are `15,707` labels, corresponding to each training sample to match to a class.
- The `test_features` matrix shape is `(1554, 426)` and has a data type of `int16`. This means there are `1,554` rows and `426` columns.
- The `test_labels` matrix shape is `(1554,)` and has a data type of `uint8`. This means there are `1,554` labels, corresponding to each testing sample.

? What are the rows and columns of the matrices?

- Each row corresponds to a single tree species sample in the dataset.
- Each column represents a feature derived from hyperspectral imaging (as written in the research paper), which captures unique data from the samples.

? What are the ranges?

- Train set: `min = 0` and `max = 14,998`
- Test set: `min = 0` and `max = 6,908`

? How many classes are there and what are the classes (the answer is in the paper linked above).

There are 8 classes in the dataset - seven dominant tree species as well as dead standing trees.

How many examples are provided of each class in the train and test splits?

Training Split:

- Class 0: 2,519 examples
- Class 1: 821 examples
- Class 2: 1,575 examples
- Class 3: 3,980 examples
- Class 4: 2,640 examples
- Class 5: 88 examples
- Class 6: 852 examples
- Class 7: 3,232 examples

Test Split

- Class 0: 389 examples
- Class 1: 30 examples
- Class 2: 278 examples
- Class 3: 404 examples
- Class 4: 100 examples
- Class 5: 22 examples
- Class 6: 43 examples
- Class 7: 288 examples

Compare the two PyTorch models in terms of test performance and overfitting. Also compare your PyTorch results to your scikit-learn results.

The PyTorch **linear classifier** ended up with a `training accuracy of .8486` and a `test accuracy of .8281`. This indicates that there is minimal overfitting. In addition, the training stopped increasing in accuracy at around epoch 15 which means that the model likely converged early. The additional epochs were not necessary as it didn't significantly improve accuracy.

The PyTorch **Neural Network** classifier ended up with a `training accuracy of .9493` and a `testing accuracy of .8545`. Since the training accuracy is significantly higher than the testing accuracy, there is overfitting to the data. In addition, there is a rapid increase in accuracy in the first 5 epochs and slowly increased throughout the rest of the epochs.

Model	Train Accuracy	Test Accuracy	Overfitting	Description
Linear Classifier (PyTorch)	84.9%	82.8%	Minimal	Converged very quickly
Neural Network (PyTorch)	95%	85.4%	Medium	Higher test accuracy, but overfitted data.

When comparing the models between SKLearn and PyTorch, there were different winners.

Model	Test Accuracy
Linear Classifier (PyTorch)	82.8%
Linear Classifier (SKLearn)	83.3%
Neural Network (PyTorch)	85.4%
Neural Network (SKLearn)	83.3%

As you can see, the SKLearn linear classifier model did slightly better than the PyTorch model, while the PyTorch Neural Network doing a decent amount better than the SKLearn NN.

This is likely due to being able to have more control of the NN layers and dataloaders, etc.