Assignment 5: Fully Connected ANN for Tabulated Data

40 Points

Deliverable: Notebook (.ipynb file) with all required code and answers to questions/discussions provided in Markdown cells. Note that Assignment 6 will continue this analysis.

Overview: The goal of this exercise is to gain experience building and training fully connected neural networks along with building some intuition on how to undertake the neural network training and validation processes. You will work with the Covertype dataset, which is available on the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Covertype. It has also been provided as part of the class data. This dataset consists of 581,012 records. The goal is to differentiate 7 forest types using landscape characteristics. The forest types are differentiated in the Cover_Type column using numeric codes as follows:

1 = Spruce/Fir, 2 = Lodgepole Pine, 3 = Ponderosa Pine, 4 = Cottonwood/Willow, 5 = Aspen, 6 = Douglas-fir, 7 = Krummholz

You will use the following predictor variables. Note that there are other predictor variables available which will not be used here.

Elevation, Aspect, Slope, Horizontal_Distance_To_Hydrology, Vertical_Distance_To_Hydrology, Horizontal_Distance_To_Roadways, Hillshade_9am, Hillshade_Noon, Hillshade_3pm

Task 1: Pre-Processing (5 Points)

1. Split the available data into separate training, validation, and testing partitions. Use 60% of the samples to train the model, 20% for validation at the end of each training epoch, and 20% for testing the final model.
2. Normalize all predictor variables to *z*-scores using means and standard deviations derived from the training data only.
3. Define a DataSet subclass that delivers the data to the DataLoader in the correct format. Make sure that the shapes and data types of the tensors are correct.

Task 2: Data Exploration (10 Points)

1. Use Pandas to find the count of samples in each class/forest type in each of the three data partitions.
2. Create grouped boxplots to compare the distribution of each predictor variable between the seven forest types.
3. Write up your findings. Are there issues with data imbalance between the classes? What predictor variables seem to be most important for differentiating the classes?

Task 3: Build a Fully Connected Neural Network Architecture (5 Points)

1. Subclass nn.Module to define your neural network architecture. The architecture should have the following structure:

   Fully Connected Layer (In = Number of Predictor Variables, Out = 256) → 1D Batch Normalization → ReLU → Fully Connected Layer (In = 256, Out = 256) → 1D Batch Normalization → ReLU → Fully Connected Layer (In = 256, Out = Number of Differentiated Classes)

Task 4: Train Model (10 Points)

1. Define a DataLoader for the training data that shuffles the samples and provides the data in batches of 256 samples. Drop the last mini-batch and shuffle the data.
2. Define a DataLoader for the validation data that provides the data in mini-batches of 256 samples. Drop the last mini-batch.
3. Define a Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.001.
4. Define the loss metric as Cross Entropy Loss. Note that this expects raw logits as opposed to probabilities, so there is no need to pass the results through a softmax activation beforehand.
5. Use TorchMetrics to calculate overall accuracy during the training process for the training and validation samples separately.
6. Create a training loop to train the model for 50 epochs. Log the epoch number, loss, and overall accuracy for both the training and validation data.
7. Save the model weights/parameters at the epoch that provides the lowest loss for the validation data.

Task 5: Model Assessment (10 Points)

1. Create a plot that shows both the training and validation loss across all 50 epochs.
2. Define a DataLoader for the testing data that provides the data in mini-batches of 256 samples. Drop the last mini-batch.
3. Predict the testing data and use the TorchMetrics package to obtain the following aggregated assessment metrics: overall accuracy and class-aggregated macro-averaged precision, recall, and F1-score.
4. Predict the testing data and use the TorchMetrics package to obtain the following class-level metrics: confusion matrix, recall, precision, and F1-score.
5. Discuss the model performance. Is there evidence of overfitting? How well is the model performing? What classes are most confused?