CS 59000 Application of Deep Learning

Homework 5: Machine Learning and Artificial Neural Networks

Due on **October 9**, 2024, 11:59pm

The goal of this homework assignment is to help you get familiar with machine learning models and artificial neural networks (ANN). For parts 1 – 2, please put the code into the same Jupyter Notebook “parts_1_2.ipynb”. Please keep all running outputs in all Jupyter Notebooks.

1. **XGBoost Model for Predicting Median Housing Price**. Download “end_to_end_project.ipynb” file from Brightspace, rename it to “parts_1_2.ipynb”, and upload it to Colab to implement Parts 1 and 2. At the end of the Jupyter Notebook, please add the XGBoost model with 100 trees and 42 for “random_state”. Show the results for calling “model_cross_val_scores” function. After training the XGBoost model using the prepared training dataset, please show the RMSE for predicting the prepared test dataset. Please feel free to use “AI Assistant” to help implement this part in Colab.

2. **Lazy Predict**. In the same Jupyter Notebook, i.e., “parts_1_2.ipynb”, add the code after part 1, i.e., at the end of the file, to apply Lazy Predict. Please refer to the following project: https://github.com/shankarpandala/lazypredict. Specifically, apply Lazy Predict regressor to prepared housing price dataset (i.e., prepared training and test datasets), and print results for all models. Please look into the outputs, and compare results from Lazy Predict with those results from individual model such as random forest and XGBoost.

3. **PyTorch for MNIST Dataset**. Please use PyTorch to implement the class example on MNIST dataset that was based on the Keras library. Please use the same neural network structure, loss function, and optimizer. Train the model with 5 epochs and 128 batch sizes. Show the accuracy for the test dataset. Please feel free to ask an AI tool to help generate the code.

4. **Avoid Overfitting in ANN**. In this part, two different technologies are applied to the class example “binary_classification_example.ipynb” in order to reduce the overfitting. The first technology is called dropout, which was developed by Geoff Hinton and his students at the University of Toronto. Dropout is one of the most effective and most commonly used regularization techniques for neural networks. Dropout, applied to a layer, consists of randomly dropping out (setting to zero) a number of output features of the layer during training. Specifically, add the following dropout layer right after each hidden layer (of two hidden layers):

   ```
   layers.Dropout(0.5)
   ```

   The second technology is called early stopping. That is, instead of finding the right epochs and then retrain the model, we can apply callbacks for the fit() function. Remove the section of “Refine the model”. In the section of “Fit the model with a validation set”, before training the model, add the following code for the list of callbacks:

```
callbacks_list = [
  keras.callbacks.EarlyStopping(
    monitor="val_accuracy",
    patience=2,
  ),
  keras.callbacks.ModelCheckpoint(
    filepath="checkpoint_path.keras",
    monitor="val_loss",
    save_best_only=True,
  )
]
```

There are two callbacks applied: (1) Based on the validation accuracy, if it is not improved for two epochs, training will be stopped. (2) Based on the validation loss, save the best model to file “checkpoint_path.keras”.

In the “model.fit” function, add the following:

```
callbacks=callbacks_list,
```

Additionally, at the beginning of the section of “Predict new data”, add the following code to load the best model:

```
model = keras.models.load_model("checkpoint_path.keras")
```

Please run the modified code to see if the training will be automatically early stopped and the performance of prediction can be (slightly) improved due to the dropout.

Please upload “parts_1_2.ipynb”, “part_3.ipynb”, and “part_4.ipynb” files to your GitHub homework repo under “hw5”.

Grading rubric:

- Part 1 – 25pts
- Part 2 – 25pts
- Part 3 – 25pts
- Part 4 – 25pts

Total: 100pts