AI for Physics
Assignment 1

Tongji University
Due: Sunday, October 19, 2025, 11:59pm

Policies

- Prepare a single Jupyter notebook (`.ipynb`) containing your complete report, all executable code, and any generated outputs (e.g., images, results).
- Name your file "`Homework-1-YourStudentNumber.ipynb`" and submit it to Canvas.
- Submissions after the deadline will not be accepted.
- **Policy on AI Tools:**
  - The submission of any code, text, or answers directly generated by AI tools (including but not limited to ChatGPT, Claude, Copilot, or similar systems) as one's own original work is **strictly prohibited**.
  - Using AI to complete the core, analytical, or interpretative parts of the assignment for you is **strictly prohibited**.
  - You are **allowed** to use AI as a search engine or coding assistant for tasks like explaining a general programming concept or error message; or looking up the syntax for a standard function (e.g., `sklearn.metrics.roc_curve`).
  - If you use an AI tool for any assistance, you must include a section titled "AI Use Declaration" at the end of your report. In this section, you must state which tool you used and describe specifically how you used it (e.g., "I used ChatGPT to debug a dimension mismatch error in my TensorFlow model" or "I used Claude to get an example of how to use `xgboost.plot_importance`"). **Failure to declare the use of AI will be treated as an academic integrity violation.**

1 Neural networks vs. boosted decision trees [10 Points]

In this problem, you will compare the performance of neural networks and boosted decision trees for binary classification on a given dataset (see `Homework-1-Data.zip`).

This dataset is taken from the experiment searching for neutrino oscillations, used to distinguish electron neutrinos (signal) from muon neutrinos (background). The dataset contains 130,065 samples with 50 features. In the `Homework-1-Data.txt` file, the first line is the the number of signal events followed by the number of background events. The signal events come first, followed by the background events. Each line, after the first line, has 50 real-number variables (i.e. features) for one event. You should randomly split the dataset into training (80%) and testing (20%) subsets.

Problem A [2 points]: Using the dataset and XGBoost, train a boosted decision tree on the training dataset. Use the Scikit-learn API `xgboost.XGBClassifier`. For an initial choice of hyperparameters use 100 trees (`n_estimators`), maximum tree depth (`max_depth`) of 10, learning rate (`learning_rate`) of 0.1, `colsample_bytree` of 0.8, and `subsample` of 0.8.

Plot the receiver operating characteristic (ROC) curve using the testing dataset. What area under the curve (AUC) and accuracy do you achieve?

Problem B [2 points]: Plot the F-score for all the 10 "most important" features using `xgboost.plot_importance`. Which feature is the most important?

Plot this feature using the testing dataset in a 1D histogram separately for signal and background. For the histogram binning, use 100 bins from the minimum value of this feature to the maximum value of this feature in the testing dataset. What do you notice about this feature?

Problem C [2 points]: Using the dataset and the Keras Model API (or PyTorch), train a neural network with 3 hidden layers each with 128 units (i.e. neurons) and `tanh` activation function. The final layer should have sigmoid activation. Use the binary cross-entropy loss function, the SGD optimizer with a learning rate of 0.01 (which is the default), and a batch size of 128. Train the model for 50 epochs.

Plot the receiver operating characteristic (ROC) curve using the testing dataset. What AUC and accuracy do you achieve?

Problem D [1 points]: Swap out the `tanh` for `ReLU` activation function, while keeping everything else the same. Does the network train effectively? Why or why not?

Problem E [1 points]: Now, we will make two minor changes to the network with `ReLU` activations: preprocessing and the optimizer.

For the feature preprocessing use `sklearn.preprocessing.StandardScaler` to standardize the input features. Note you should fit the standard scaler to the training data *only*, and apply it to both the training and testing data. For the optimizer, use Adam with a learning rate of 0.001 (which is the default) instead of SGD. Train the model for 50 epochs.

Plot the receiver operating characteristic (ROC) curve using the testing dataset. What AUC and accuracy do you achieve now? Is it comparable to the BDT?

Problem F [2 points]: What else can you do to improve the AUC? Show the highest AUC you achieved and the specific method you used.