STA 4364 HW 4

due December 7 by 11:59PM on Webcourses

Submission Format: All problems must be submitted in either an R Markdown file or a Jupyter Notebook files. You can submit **at most one file for each problem, except for Problem 3 where you can submit up to 2 files, one for each network.** Due to the long runtime of neural network training, it is suggested that all neural network training codes are recorded in separate files. Any other format will result in a 4-point deduction from the final score.

Problem 1: (10 points) In this problem you will compare the performance of a variety of classifiers that you have learned about throughout the course. The data is in the file `magic04.data` and the column names are in the file `magic04.names`. The last column is a categorical response with values `g` or `h`, and the rest of the columns are numerical features. You can read more about the dataset here.

(a) Load the data (can use the pandas function `read_table` with the arguments `sep=','` and `header=None`). Split the data into a training and test set. Scale and center the columns using the mean and standard deviation of each column from the *training set* (make sure you use the same scaling on the test set that is used on the training set).

(b) Learn the following models to classify the training data:

- **Logistic Regression**: Can import `LogisticRegression` from `sklearn.linear_model`.
- **LDA**: Can import `LinearDiscriminantAnalysis` from `sklearn.discriminant_analysis`.
- **KNN Classifier**: Need to choose the number of neighbors k.
- **Linear SVM**: Need to choose the margin penalty C as a hyperparameter.
- **Gaussian SVM**: Need to choose the margin penalty C and the radius width γ.
- **Random Forest**: Need to choose the number of trees.
- **Gradient Boosted Decision Tree**: Need to choose the number of trees and learning rate. If you want, you can also experiment with randomly selecting rows and columns when growing each tree.

To tune hyperparameters for each model, you can either use cross-validation or hand-tune by examining the model performance for reasonable values of the hyper-parameters.

(c) Apply your models to the test set. Report the accuracy, visualize an ROC curve, and report the AUC for each model. For Logistic Regression, Random Forests, and Gradient Boosted Decision Trees, report the most meaningful predictors.

Problem 2: (10 points) In this problem, you will compare the performance of several different regression models you have learned throughout the course. The data is in the file `qsar_aquatic_toxicity.csv`. All columns are numerical features. The last column in the response variable, and the other columns are predictors. You can read more about the dataset here.

(a) Load the data (can use the pandas function `read_csv` with the arguments `sep=';'` and `header=None`). Split the data into a training and test set. Scale and center the columns using the mean and standard deviation of each column from the *training set* (make sure you use the same scaling on the test set that is used on the training set).

(b) Learn the following models to classify the training data:

- **Linear Regression**: Can import `LinearRegression` from `sklearn.linear_model`.
- **KNN Regression**: Need to tune the number of neighbors k.
- **Random Forest**: Need to choose the number of trees.

- **Gradient Boosted Decision Tree**: Need to choose the number of trees and learning rate. If you want, you can also experiment with randomly selecting rows and columns when growing each tree.

To tune hyperparameters for each model, you can either use cross-validation or hand-tune by examining the model performance for reasonable values of the hyper-parameters.

(c) Apply your models to the test set. Report the MSE on the test data along with the R^2 value for each model. For Linear Regression, Random Forests, and Gradient Boosted Decision Trees, report the most meaningful predictors.

Problem 3: (20 points) In this problem, you will learn a classifier network for the CIFAR-10 dataset. The dataset has ten classes of different animals and vehicles. Each observation is a 32×32 pixel image. There are 50000 training images and 10000 test images distributed evenly across the 10 classes. More information can be found here.

You can use the MNIST classifier code provided in class a guideline for your code. To download the data, use the `torchvision.datasets.CIFAR10` (has the same arguments as `torchvision.datasets.MNIST` that we used in class). For the optimizer, you can use Adam with a learning rate of 0.0001. You will analyze two different networks:

- A ConvNet with the following structure:
  1. 3×3 convolution with stride 1, padding 1, and 3 in-channels (one for each RGB image channel) and 32 out channels.
  2. Batch Norm with 32 channels.
  3. ReLU Activation function.
  4. 4×4 convolution with stride 2, padding 1, 32 in-channels and 64 out channels
  5. Batch Norm with 64 channels.
  6. ReLU Activation function.
  7. 4×4 convolution with stride 2, padding 1, 64 in-channels and 128 out channels
  8. Batch Norm with 128 channels.
  9. ReLU Activation function.
  10. 4×4 convolution with stride 2, padding 1, 128 in-channels and 256 out channels
  11. Batch Norm with 256 channels.
  12. ReLU Activation function.
  13. 4×4 convolution with stride 1, padding 0, 256 in-channels and 10 out channels (one out channel for each CIFAR-10 class)

  Important Note: You will need to use `.squeeze()` on your final output! Do this before return a value in the `forward` function.

- A Wide ResNet classifier provided in the file `wide_resnet.py`.

Make two plots to for each network training process (4 plots total):

- A loss plot showing cross-entropy loss for the training and validation data for each training epoch.
- An accuracy plot showing the proportion of correct predictions for the training and validation data for each epoch.

Compare the performance of the two networks on the training and testing data.