Classifiers and Regressors Report

Nayanika Ghosh (4191 4976) | Disha Nayar (6199 1035)

Our project submission consists of the following files :

1. Main.py – Implements the classifiers and regressors for the given dataset
2. HumanVsCom.py - Implements the human game play of Tic tac toe
3. LRWeights, knnWeights, MLPWeights, – contains the weights of the linear regression, KNN and MLP trained model respectively.
4. datasets-part1 – folder containing the three dataset files.

Explanation for Implementation

The two python files consist the logic for implementation. Below we describe what each of the files contain and how we have implemented the models.

Main.py file consists of two classes. These classes define classifier models and regressor models respectively

1. **Classifiers** – The class consists of 3 classifier model, namely Linear SVM, k-nearest neighbor(KNN) and Multi-layer perceptron(MLP). For each of them, the model is trained using k-fold validation, with k = 10. The final accuracy reported is the average of the accuracy observed in each of the k folds. Further, the confusion matrix is also averaged over all the k folds and normalized at the end. Specific details about each classifier is as below:
    1. Linear SVM - We have used scikit-learn's SVC function with kernel = 'linear' to train a linear SVM.
    2. K-nearest neighbors - We have used scikit-learn's kNeighborsClassifier to train a KNN model with number of neighbors set to 9.
    3. MLP - We have used scikit-learn's MLPClassifier to train the classifier. The optimizer chosen is Adam's optimizer. The model is designed to have 3 hidden layers with 200, 100 and 40 units, respectively. Further, we have used RELU activation function for the hidden layers.

2. **Regressors** – The class consists of 3 regressor models, namely Linear regressor, k-nearest neighbor(KNN) and Multi-layer perceptron(MLP). As with Classifiers, the models are trained using k-fold validation, with k = 10. The final accuracy reported is average of the accuracy observed in each of the k folds. Specific details about each regressor is as below:
    a. Linear Regression - Linear regressor is modelled as y = w.x + bias, where w is the weight vector. Since for the given dataset - "Intermediate boards optimal play (multi label)" we need to find optimal moves among 9 positions, we create 9 separate models for predicting regression for each position and consider the one with maximum predicted value as the optimal move. This is done by setting the y value of position with maximum prediction to 1 and rest as 0. The weight vector for each of the 9 model is calculated using normal equation.

b. K-nearest neighbors - We have used scikit-learn's kNeighborsRegressor to train the regressor with the number of neighbors set to 9. Unlike in linear regression, we consider all the positions as optimal if their y values are greater than the threshold(in our case threshold is 0.5).
c. MLP - We have used scikit-learn's MLPRegressor to train the regressor. We have used same network configuration here as in MLPClassifier. We also get the y optimal in a similar way as with the KNN regressor.

HumanVSCom.py consists of a single class that simulates the Tic tac toe game between humans(us) and the computer. In our implementation, the user is given the option to choose among the three regressors they want to play against. Until either one wins or draws, the game continues entailing input from human and model alternatively. In each turn, we pass the current state of the tic tac toe board to the model and update the board with the predicted move.

Evaluation Results

Classifier and Regressors Output:
Figure 1 shows the Accuracy and the confusion matrix for the three classifiers on tictac_final dataset.
Figure 2 shows the Accuracy and the confusion matrix for the tree classifiers on tictac_single dataset.
Figure 3 shows the accuracy for the three regressors on tictac_multi dataset.

```
Output for Classifiers

Accuracy of Linear SVM on  Final boards classification dataset  is  0.983311403508772
Confusion Matrix of Linear SVM  on  Final boards classification dataset  is
 [[0.95 0.05]
 [0.   1.  ]]


Accuracy of k-Nearest Classifier on  Final boards classification dataset  is  0.9916557017543861
Confusion Matrix of k-Nearest Classifier on  Final boards classification dataset  is
 [[0.98 0.02]
 [0.   1.  ]]


Accuracy of MLP Classifier on  Final boards classification dataset  is  0.9906140350877193
Confusion Matrix of MLP Classifier on  Final boards classification dataset  is
 [[0.97 0.03]
 [0.   1.  ]]
```

Figure 1

```
Accuracy of Linear SVM on  Intermediate boards optimal play (single label)  is  0.3654359057903556
Confusion Matrix of Linear SVM  on  Intermediate boards optimal play (single label)  is
 [[1.   0.   0.   0.   0.   0.   0.   0.   0.  ]
 [0.51 0.14 0.16 0.   0.18 0.   0.   0.   0.  ]
 [0.55 0.   0.29 0.   0.17 0.   0.   0.   0.  ]
 [0.58 0.07 0.16 0.03 0.15 0.   0.   0.   0.  ]
 [0.6  0.   0.   0.   0.4  0.   0.   0.   0.  ]
 [0.56 0.11 0.23 0.01 0.09 0.   0.   0.   0.  ]
 [0.59 0.08 0.2  0.02 0.11 0.   0.   0.   0.  ]
 [0.62 0.17 0.14 0.   0.07 0.   0.   0.   0.  ]
 [0.59 0.08 0.27 0.   0.05 0.   0.   0.   0.  ]]


Accuracy of k-Nearest Classifier on  Intermediate boards optimal play (single label)  is  0.770566002606591
Confusion Matrix of k-Nearest Classifier on  Intermediate boards optimal play (single label)  is
 [[0.87 0.02 0.04 0.01 0.03 0.01 0.02 0.   0.01]
 [0.05 0.71 0.06 0.03 0.07 0.01 0.03 0.   0.03]
 [0.07 0.04 0.79 0.02 0.04 0.01 0.01 0.   0.02]
 [0.08 0.05 0.04 0.68 0.07 0.01 0.03 0.01 0.04]
 [0.09 0.03 0.03 0.02 0.82 0.01 0.01 0.   0.  ]
 [0.07 0.08 0.02 0.03 0.01 0.71 0.03 0.01 0.03]
 [0.12 0.06 0.05 0.02 0.02 0.   0.71 0.01 0.01]
 [0.12 0.07 0.03 0.04 0.03 0.03 0.03 0.64 0.01]
 [0.11 0.05 0.04 0.03 0.04 0.01 0.01 0.01 0.7 ]]

Accuracy of MLP Classifier on  Intermediate boards optimal play (single label)  is  0.9558850772668034
Confusion Matrix of MLP  on  Intermediate boards optimal play (single label)  is
 [[0.97 0.01 0.01 0.   0.01 0.   0.   0.   0.  ]
 [0.01 0.94 0.01 0.   0.01 0.   0.   0.   0.01]
 [0.01 0.   0.95 0.01 0.02 0.   0.01 0.   0.  ]
 [0.02 0.01 0.   0.95 0.01 0.   0.   0.   0.01]
 [0.01 0.01 0.01 0.01 0.96 0.   0.   0.   0.  ]
 [0.01 0.01 0.   0.01 0.   0.95 0.   0.01 0.  ]
 [0.02 0.01 0.   0.01 0.   0.   0.95 0.01 0.  ]
 [0.01 0.02 0.   0.01 0.   0.01 0.   0.94 0.  ]
 [0.02 0.01 0.01 0.   0.   0.   0.   0.   0.95]]
```

Figure 2

```
Output for Regresors

Accuracy of Linear Regressor on Intermediate boards optimal play (multi label): is  0.7363251980802249

Accuracy of k-Nearest Regressor on Intermediate boards optimal play (multi label): is  0.9088520397608585

Accuracy of MLP Regressor on Intermediate boards optimal play (multi label): is  0.9739149806574421
```

Figure 3

Method that worked best:
We notice that for classifiers, MLP overall performs better that KNN and Linear SVM. Linear SVM gives a good accuracy (0.98) on the final dataset but performs poorly on the single dataset (0.36). This is because the data in the single dataset is clearly not linearly separable. Also, KNN does not do that well on the single dataset compared to MLP is because, since we have 9 output classes, we believe that the data might be imbalanced while training.

For Regressors, we notice that MLP performs the best, but even KNN gives a good accuracy of 90%. Linear regression fails to give good performance here as the data is non-linear.

Classifiers on 1/10th data:
Further, we trained our models on 1/10th of the data. Below we report the accuracies for each of the model for the same
For "Final boards classification dataset"

Linear SVM = 98%
KNN - 73%
MLP - 85%

For Intermediate boards optimal play (single label)

Linear SVM = 38.4%
KNN - 55.19%
MLP - 95.6%

We notice that Linear SVM performs similar on 1/10th of the data but for MLP and KNN the accuracy decreases significantly.

Scalability of models:
As noticeable from the results on 1/10th of the data, some models scale better to larger dataset. This happens because, as we increase the size of the dataset, the function capacity of the model increases. Therefore, chances to converge to an optimal function increase. Hence, these methods scale better to large datasets.

Human vs Computer evaluation:
We noticed that it was quite hard to beat MLP and KNN regressors. It was rare that we won against these two regressors. However, with the linear regressor, it is quite east to win. Mostly, if we skip an optimal move, linear regression model tends to win against the human(us). We have walked through a game play against MLP and Linear regressor in our video.

Instructions to Run

1. Prerequisites software : python, NumPy and scikit-learn needs to be installed.
2. Unzip the Project.zip
3. To run Main.py use command : python Main.py
   This should output the statistical accuracy and confusion matrices for classifiers and accuracies for regressors.
4. To run HumanVsCom.py use command : python HumanVsCom.py
   Enter the cell number 1-9 (numbered column wise) to play as 'X'