Assignment 3

CSE342 - Statistical Machine Learning

Ritika Thakur
2022408
CSAI

***Abstract*—Assignment 3**

I. Question

Question: Utilize the MNIST dataset available at https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz and focus on classes 0, 1, and 2. Apply dimensionality reduction using PCA. Train a decision tree with 3 terminal nodes on the reduced dataset. Employ bagging to create 5 datasets from the original dataset and determine overall accuracy using majority voting.

II. Methods

The following methods were employed to address the question:

- Principal Component Analysis (PCA) application to reduce the dimensions of the data
- Decision Tree Classifier used to make decision tree based on gini impurity for the three classes (CART Algorithm)
- Bagging used to create multiple decision trees and make decision based on majority voting.
- Reported class-wise and overall accuracy for both training and testing data.

III. PCA

PCA was utilized to reduce the dimensionality of the MNIST dataset. The dataset was separated into classes 0, 1, and 2, and PCA was applied to each class individually for both train data and test data.
The dataset was divided into train data, train data labels, test data, test data labels.

- The `mnistPCA` class performs PCA on the MNIST dataset, separating images by class (0, 1, and 2).
- Constructor initializes variables and converts class matrices to NumPy arrays.
- Data matrix `X` is created, centered, and PCA is applied to obtain eigenvalues and eigenvectors.
- `reduceDim` method reduces data dimensionality using specified eigenvalues.
- Instances of `mnistPCA` are created for training and test datasets, and PCA is applied.
- Reduced dimension data and labels for both datasets are printed.

IV. Decision Tree Classifier

A decision tree classifier was trained on the reduced dataset obtained from PCA. The maximum depth of the decision tree was set to 2 to limit overfitting.

- The `DecisionTreeClassifier` class implements a decision tree algorithm for classification.
- The constructor initializes the maximum depth of the tree.
- The `fit` method trains the decision tree model using the provided training data `X` and labels `y`.
- The `predict` method predicts the class labels for the input data `X`.
- The `_best_split` method finds the best split point for a given feature based on the Gini impurity criterion.
- The `_grow_tree` method recursively grows the decision tree up to the specified maximum depth.
- The `_predict` method traverses the decision tree to predict the class label for a single input.
- The `convert_tree_to_sklearn` function converts the decision tree to a string format similar to scikit-learn's decision tree representation.
- The `visualize_decision_tree` function visualizes the decision tree using Matplotlib.
- An instance of `DecisionTreeClassifier` is created, trained, and its performance on the training and test datasets is evaluated.
- The class names and feature names are initialized for visualization.
- The decision tree is visualized using the `visualize_decision_tree` function.

V. Bagging

Bagging was employed to create 5 datasets from the original dataset. For each dataset, a decision tree classifier was trained, and predictions were made using majority voting.

- The class names are initialized for classification.
- Bagging is performed by training multiple decision trees with bootstrapped samples.
- For each iteration of bagging (in this case, 5 decision trees are trained):
  1) Training data is sampled with replacement using the `resample` function.
  2) DecisionTreeClassifier instances are created and trained on the sampled data with a specified maximum depth.

3) The trained decision tree is visualized using the `visualize_decision_tree` function.
4) The trained decision tree is appended to the list of trees.

- After training all decision trees, the ensemble's performance is evaluated on the test dataset.
- For the training dataset:
  1) Predictions are made by each tree in the ensemble.
  2) Final predictions are determined by selecting the class with the most votes among all trees. If there is a tie or no clear majority, any class from the tied classes is chosen.
  3) Total accuracy and class-wise accuracy are calculated and printed for the training dataset.
- Similar steps are followed for the test dataset, and total accuracy and class-wise accuracy are calculated and printed.

VI. Important Assumptions

The following assumptions were made during the implementation:

- The MNIST dataset provided contains images of digits ranging from 0 to 9. Only classes 0, 1, and 2 were considered for this assignment.
- PCA was applied separately to each class to maintain class-specific information.
- Decision trees were limited to a maximum depth of 2 to prevent overfitting. However, increasing the depth to say 3 does not necessarily change a lot for our dataset in terms of accuracy.
- Although we are deciding the best split and threshold at each level, we are still generating the entire binary tree so that it is more analytical, especially for maximum depths ¿ 2.
- For bagging, 5 samples have been generated from the original dataset using resampling and `DecisionTreeClassifier` applied to these 5 samples. Increasing the number of trees to say 100-200 might show significant difference in accuracy but otherwise for our dataset we will only witness a minor difference.

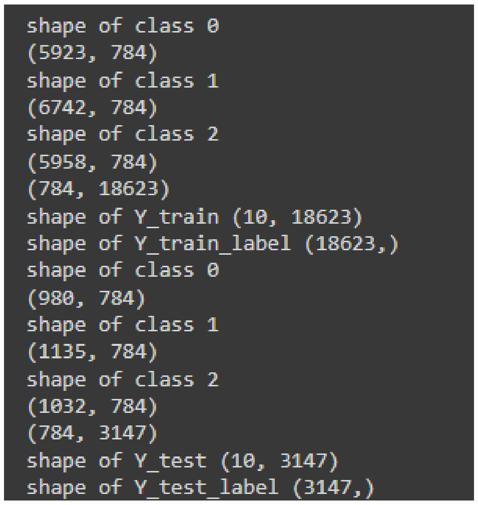
VII. Accuracy Analysis

The overall accuracy of the decision tree classifier and the bagging approach was analyzed on both the training and test datasets.

- For a single decision tree classification, we have an overall accuracy of around 90.7 % for train dataset and around 81.6% for test dataset. This is for maximum depth of 2. For maximum depth = 10, we have overall accuracy of train data as 100% while for test data it drops down to 74% due to overfitting.
- For bagging, we have an overall accuracy of around 90.7% for train dataset and around 83.2% for test dataset. This is for 5 samples and maximum depth = 2. For 100 samples, we have overall accuracy of train data as 91% and for test data 83%. The accuracy does not change much due to our sample data not being randomized enough. Say if we added more noise to the dataset then our accuracy would have significantly increased with an increase in the number of samples. For 100 samples and maximum depth = 100, we have overall accuracy of train data as 100% since we are more extensively learning the decision tree for 100 random samples and each sample upto a depth of 100, and for test data the overall accuracy drops to 78.4% due to overfitting.
- An interesting observation is that although increasing the depth results in overfitting, yet using a sample of 100 trees over a single tree reduces the drop in accuracy due to overfitting.

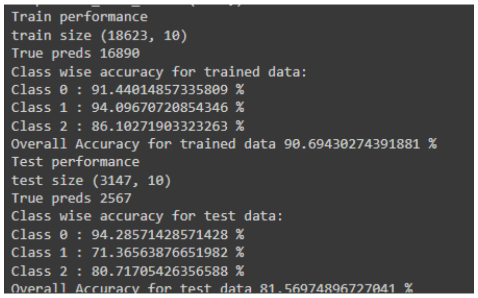
VIII. Results

The results of the accuracy analysis are presented below.



```
shape of class 0
(5923, 784)
shape of class 1
(6742, 784)
shape of class 2
(5958, 784)
(784, 18623)
shape of Y_train (10, 18623)
shape of Y_train_label (18623,)
shape of class 0
(980, 784)
shape of class 1
(1135, 784)
shape of class 2
(1032, 784)
(784, 3147)
shape of Y_test (10, 3147)
shape of Y_test_label (3147,)
```

Fig. 1. Result of applying PCA to Class 0, Class 1, Class 2



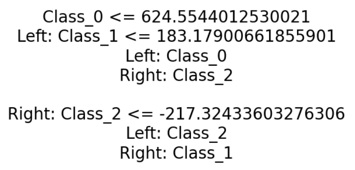
```
Train performance
train size (18623, 10)
True preds 16890
Class wise accuracy for trained data:
Class 0 : 91.44014857335809 %
Class 1 : 94.09670720854346 %
Class 2 : 86.10271903323263 %
Overall Accuracy for trained data 90.69430274391881 %
Test performance
test size (3147, 10)
True preds 2567
Class wise accuracy for test data:
Class 0 : 94.28571428571428 %
Class 1 : 71.36563876651982 %
Class 2 : 80.71705426356588 %
Overall Accuracy for test data 81.56974896727041 %
```

Fig. 2. Class-wise and Overall Accuracy of Single Decision Tree on Class 0, Class 1 and Class 2

IX. Observations and Conclusions

A. Observations

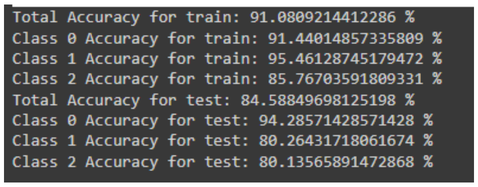
- Decision Tree Classifier is an efficient method for binary classification but can fall short for multi-class classification.



```
Class_0 <= 624.5544012530021
Left: Class_1 <= 183.17900661855901
Left: Class_0
Right: Class_2

Right: Class_2 <= -217.32433603276306
Left: Class_2
Right: Class_1
```

Fig. 3. Visualisation of the decision tree



```
Total Accuracy for train: 91.0809214412286 %
Class 0 Accuracy for train: 91.44014857335809 %
Class 1 Accuracy for train: 95.46128745179472 %
Class 2 Accuracy for train: 85.76703591809331 %
Total Accuracy for test: 84.58849698125198 %
Class 0 Accuracy for test: 94.28571428571428 %
Class 1 Accuracy for test: 80.26431718061674 %
Class 2 Accuracy for test: 80.13565891472868 %
```

Fig. 4. Class-wise and Overall Accuracy of Bagging on Class 0, Class 1 and Class 2 for 5 random samples

- Bagging is a powerful ensemble method that can improve the accuracy of decision trees by reducing overfitting.
- Increasing the depth of the decision tree can lead to overfitting, especially for small datasets like ours with only 3 classes.
- Bagging with multiple decision trees can help reduce the impact of overfitting and improve overall accuracy. However, increasing the maximum depth of the decision trees can lead to overfitting in the ensemble as well.
- An interesting observation is that bagging will reduce the impact of overfitting when compared to a single decision tree classifier even when the maximum depth is increased.

B. *Conclusions*

- Decision Tree Classifier and Bagging are effective methods for classification tasks.
- PCA can be used to reduce the dimensionality of the dataset and improve the efficiency of the classification algorithms.
- The choice of hyperparameters such as the maximum depth of the decision tree and the number of samples in bagging can significantly impact the performance of the classifier.
- Bagging can help reduce overfitting and improve the overall accuracy of the classifier, especially when using multiple decision trees with different samples.

X. Alternative Approaches and Future Scope

- Alternative approaches such as Random Forest can be explored to further improve the classification accuracy.
- The dataset can be augmented with additional samples to increase the diversity of the training data and improve the generalization of the classifier.
- We can also perform PCA while selecting random samples to increase noise and train the model better to improve accuracy.

XI. References

- https://towardsdatascience.com/decision-tree-from-scratch-in-python-46e99dfea775
- https://www.analyticsvidhya.com/blog/2020/10/all-about-decision-tree-from-scratch-with-python-implementation/