Introduction

Random Forest is a machine learning algorithm used for classification and regression problems. The algorithm is an ensemble method that trains multiple decision trees and aggregates their predictions to produce a more accurate prediction.

How we Implement Random Forest Classifier involves the following steps:

1 - Preprocessing of the data

Clean and transform the data to get it ready for modeling. Remove semicolon(;) and ID from the data. Make each number into one column with its dimension name. Find unique classes with given numbers. For example: 10 instances from chair, camera,…

2- Splitting the data into training and testing sets

To evaluate the performance of the model. We did this for both whole data and training images per class.

3 - Training the model

Train the Random Forest Classifier on the training data using the scikit-learn library. We train model with 410 estimators without any parameter. The accuracy of our model was 100% on training set and 51% on test set. At the same time, we train our model with images per class with 10 features. The accuracy on test set was 27%. We can see that our model suffers from lack of data.

4 - Setting the hyperparameters

The key hyperparameters in a Random Forest Classifier include the number of trees in the forest, the maximum depth of each tree, and the minimum number of samples required to split an internal node.
1- max_depth: defined as the longest path between the root node and the leaf node.

2- min_sample_split: Parameter that tells the decision tree in a random forest the minimum required number of observations in any given node to split it.

3- max_leaf_nodes: This hyperparameter sets a condition on the splitting of the nodes in the tree and hence restricts the growth of the tree.

4 - min_samples_leaf: This Random Forest hyperparameter specifies the minimum number of samples that should be present in the leaf node after splitting a node.

5. n_estimators: Number of trees in the forest.

6. max_sample: The max_samples hyperparameter determines what fraction of the original dataset is given to any individual tree.

7. max_features: This resembles the number of maximum features provided to each tree in a random forest.

8. bootstrap: Method for sampling data points (with or without replacement).

9. criterion: The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.

5 - The best hyperparameters for a Random Forest Classifier

It will vary depending on the specific dataset being used. It is recommended to use a grid search or a random search with cross-validation to find the optimal hyperparameters. However, we used both to find the best hyperparameter. However, We only used it on 10 features per unique classes. It takes a long time but it gives us best hyper-parameter. Again we test our sub-data (images per classes) the accuracy was increased from 27% to 32%. It was a satisfying result. We implement these parameters to our main model with the whole data and the accuracy was increased from 51% to 52%.

conclusion

Random Forest Classifier is a powerful algorithm that can be used for both binary and multi-class classification problems. It is relatively simple to implement and provides good results for many datasets. But, in this case we could not find any better solution to increase the accuracy of our model.

Fariborz Bagherzadeh, Erfan Salehi