Decision Tree and Random Forest implementation from scratch

https://github.com/shivaditya-meduri/ensembleLearning.git

March 10, 2022

Decision Tree

A decision tree is a binary tree that recursively segment the input space X, also called the feature space, along the coordinate directions to create a set of K hyper-rectangles R_k , $k \in 1, ...K$ as basic prediction units for fitting constant values γ_k within each of them. The decision tree model is formulated as follow:

$$f(x) = \sum_{k=1}^{k=K} \gamma_k \mathbb{1}_{x_k}$$

We implement the Decision Tree class in python using the concept of binary tree which will replicate the dividing of feature spaces to hyper-rectangles, recursive functions are applied to divide the feature space to smaller and purer hyper-rectangles to find the best columns and splits Depending on the task, the cost

Algorithm 1 Decision Tree Algorithm - buildTree(features, labels)

```
if $depth < maxDepth$ and $len(labels) > minSamplesSplit$ and $cost(labels) > threshold$ then
    $col, split \leftarrow findBestSplit(features, labels)$
    $root \leftarrow treeNode(col, split)$
    $leftNode \leftarrow buildTree(features[features.col <= split], labels[features.col <= split])$
    $root.leftInsert(leftNode)$
    $rightNode \leftarrow buildTree(features[features.col > split], labels[features.col > split])$
    $root.rightInsert(rightNode)$
    return root
else
    return treeNode(leaf=True, labels)
```

function for finding the best split changes. In this project implementation the gini Impurity is used to find the best split in case of decision tree classifier and the variance is used to find the best split for decision tree regressor. After the tree s trained, for a given set of test features we traverse through the tree and reach the leaf node, where in case of classifier, the majority class if outputted as the prediction and in case of regressor, the average is outputted as the prediction. Probabilities are predicted depending on the distribution of labels in the leaf node. In the above function for the decision tree, in this case, each region is represented by a leaf node and the label for that leaf node corresponds to the γ value for that region. This is how we implemented the decision tree algorithm, further parameters can be added to the if condition to hyper-parameterize the model even further for additional flexibility

Random Forest

Random Forest is an ensemble method made up of Decision Tree as the base estimator. We combine multiple weak learners, i.e. Decision Tree in this case to create a much more robust strong learner. We using bagging mechanism to sample data for training for each of the individual base estimators. The number of base estimators is a hyper-parameter which needs to be tuned for the best trade-off between performance and generalizability. After all the base estimators are trained, the majority voting of the estimators' prediction is chosen as the final prediction. Because we have multiple learners, the variance of the model is reduced. This algorithm will return a list of decision trees which can be used to make prediction for a test features set

Algorithm 2 Random Forest Algorithm - randomForest(features, labels, ntrees, **params)

```
$decisionTreesList \leftarrow list()$
$n \leftarrow 0$
**if** $n < ntrees$ **then**
    $featSample, labSample \leftarrow baggingSample(features, labels)$
    $decisionTreesList.append(decisionTree(featSample, labSample, **params))$
    $n \leftarrow n + 1$
**else**
    return decisionTreesList
```

and then the majority of the predicted classes is chosen as the final prediction in case of classification task and the average of the prediction of the base estimators is chosen as the final prediction for the regression task. Random Forest class takes as hyper-parameters, the same set of hyper-parameters of decision tree and uses it for each individual base estimator.

Testing the models implemented

The implemented classes of Decision Tree and Random Forest are tested for classification and regression tasks to asses how they are performing. The datasets tested include the Iris dataset for classification into type of flower, breast cancer dataset to classify between benign and malignant cases of cancer, and the boston housing dataset to predict the prices of houses. The datasets have been downloaded from UCI machine learning repository. It is observed that the Random Forest Classifier and Regressor outperform their Decision Tree counterparts.

Table 1: Model performance for different datasets

Model	Dataset	Score(Accuracy/RMSE)
Decision Tree Classifier	Iris	100%
Decision Tree Classifier	Breast Cancer	95%
Decision Tree Regressor	Boston Housing	53117.38
Random Forest Classifier	Iris	100%
Random Forest Classifier	Breast Cancer	97%
Random Forest Regressor	Boston Housing	52611.77

Further Experimentation

The models impelemented for this project are not optimized for computational efficiency and take a lot of time for training. In order to speed of computation, paralell processing can be employed, for example to train each of the individual base estimators in the Random Forest model, we can assign the tasks to multiple processors as they are independent processes. Also additional hyper-parameters can be added to the models for tuning for higher performance and flexibility

References

- GitHub Repo - https://github.com/shivaditya-meduri/ensembleLearning.git