Supervised and experiential learning

Practical Work 2

Valeriu Vicol

Barcelona, May 19, 2022

Contents

1 Introduction

The purpose of this practical work is to gain knowledge regarding implementation, usage and analysis of two combination of multiple classifiers Random forest and Decision forest . In the chapter bellow will be described implementation steps and analysis of results on 3 datasets of different size.

1.1 CART

Classification and Regression Trees or CART is the foundation of the next describe algorithms as each of them contains many CART trees with different configuration. Basically cart tree represent a binary tree where each node split the dataset by a feature until we get leaf node of one single category.

1.2 Decision Forest

Firstly described in The random subspace method for constructing decision forests [2], Ho established that forests of trees splitting with oblique hyperplanes can gain accuracy as they grow without suffering from overtraining, as long as the forests are randomly restricted to be sensitive to only selected feature dimensions. A subsequent work along the same lines concluded that other splitting methods behave similarly, as long as they are randomly forced to be insensitive to some feature dimensions.[3]

1.3 Random Forest

Described in Random forests [1] are similar to Decision Forest but with slightly different approach where each classification tree, is acting in different randomly selected subspace of features comparing to Decision Forest.

2 Implementation

Implementation of the algorithms were done in python, with usage of libraries like sklearn(for data processing), pandas and numpy.

2.1 CART

As this is the core structure of next 2 classifiers, his implementation will be describe first, mainly the method *fit* will be describe as it contains the main logic of building the tree. In Algorithm 9 we can see how the tree is build, first thing which worth to mention in this implementation was used 2 stopping criteria for tree to stop growing first is `max_depth` which doesn't allow tree to over certain limit like this we avoid cases were we have overfitted tree and second `min_sample` represent minimum samples per node here is a similar case as previous we don't allow the tree to grow and create example for very specific case. This algorithm is ran recursively until the stop criteria is met. Also in Algorithm 9 we can see how the prediction is happening, this algorithm is recursive and it goes in depth of tree until it find a leaf node which will be able to return the label for the current instance .

2.1.1 Structure

For implementation of algorithm were used object oriented approach as it was found most appropriate for that kind of algorithms as it allows us to extend functionality and to maintain easily project. For implementation of CART (class `ClassificationTree`) next classes were used:

- `Node`: represent the contract which have to be implemented by 2 types of notes, `LeafNode` and `DecisionNode` which are the core components of the tree, containing all required data for split data processing functions
- `SplitMetadata`: is the util class which are containing functionality regarding how the node will be split it has 2 implementation `NumericalSplitMetadata` and `CategoricalSplitMetadata`
- `TreeUtils`: which contains a set of utils methods of how the tree will be build

Input:
$\vec{x}$ which represent the entries
$\vec{y}$ which are the labels

while $min\_samples \leq samples$ ***and*** $deph \leq max\_depth$ **do**
  $best\_split \leftarrow$ split the dataset based on the feature with highest info_gain
  if $best\_split.info > 0$ **then**
    $left\_subset, right\_subset \leftarrow$ split dataset on $best\_split$ threshold
    $left\_node \leftarrow$ call the same function but with $left\_subset$
    $right\_node \leftarrow$ call the same function but with $right\_subset$
    return `DecisionNode(left_node,right_node)`
  end
  else
    return `LeafNode` with threshold value of $best\_split$
  end
end

Algorithm 1: Building CART

Input :
$\vec{x}$: entity which have to be predicted,
`node`: which might predict y
Output:
$\hat{y}$: of similar size to $\vec{x}$

if $current\_node.is\_leaf$ **then**
  return $current\_node.value$
end
if $feature\_value \leq current\_node.threshold$ **then**
  return call this function but as root node we set $current\_node.left$
end
else
  return call this function but as root node we set $current\_node.right$
end

Algorithm 2: Predicting using cart CART

2.2 Random Forest

Second implemented algorithm is the random forest, it represent just a collection CART with different sub-spaces of features, in Algorithm 4, are described steps of building a random forest, some technical steps were omitted to keep the visualisation of algorithm simplified. Worth to mention that in building algorithm also Random forest is saving the indexes of features which were used for building each tree in forest, then when need a prediction (Algorithm 5) for each tree in T we extract the indexes and then require prediction of input based on the extracted indexes

Input :
`pd`: represent the dataframe
F: number of features to use
NT: number of trees

for $i \leftarrow 1$ *to* ***NT*** **do**
  $sample\_data \leftarrow$ data-frame with F features
  $T_i \leftarrow$ build a tree with $sampled\_data$
end

Algorithm 3: Building random forest

Input :
$\vec{x}$: instances to predict
Output:
$\hat{y}$: predicted labels

for $i \leftarrow 1$ *to* ***NT*** **do**
  $vote \leftarrow T_i.predict(..)$
end
return $modified\_plurality(votes)$ collected votes of T

Algorithm 4: Predicting using cart Random Forest

2.2.1 Structure

Random classifier implements interface `BaseClassifier` which represent the contract that should be implemented be all classifiers in this project.For building this algorithm next components were used with different responsibilities :

- `ClassificationTree`: represent the implementation of CART, the number of trees and how many features each should have is defined in the constructor
- `modified_plurality(...)`: function which get the which will decide final response by modified plurality voting of tree

As in current implementation of CART each tree should return a result, and for experiments we have even number of trees in same cases, a voting mechanism was implemented to pick the results, modified plurality was chosen as it is the best fit for current case because with modified plurality we can handle ties and even number of trees.

3 Decision Forest

Last implementation is decision forest, we can observe in Algorithm in 5 how the forest is build, here is a similar thing as in Random Forest some operations were omitted from algorithm definition because they are too tehnical, like how we are saving the indexes of feature space. However we can see in Algorithm 5 that prediction of Decision Forest and random forest, as from begin code designed in such way that it can reused.

Input :
`pd`: represent the dataframe
F: number of features to use
NT: number of trees

sample_data ← data-frame with F features
for $i \leftarrow 1$ *to* ***NT*** **do**
    sub_sample_data ← shuffled random number of entities from *sample_data*
    T_i ← build a tree with *sub_sample_data*
end

Algorithm 5: Building random forest

Input :
$\vec{x}$: instances to predict
Output:
$\hat{y}$: predicted labels

for $i \leftarrow 1$ *to* ***NT*** **do**
    vote ← $T_i.predict(..)$
end
return $modified\_plurality(votes)$ collected votes of T

Algorithm 6: Predicting using cart Decision Forest

4 Datasets

For testing the algorithm were picked 3 different datasets with different size, algorithms have categorical and numerical attributes as for first dataset was picked wine because it is relatively small, and it has only numerical attributes, similar thing is with second choice as it also mainly contains the numerical attributes and one categorical. And the third dataset is seismic bumps which contains plenty of numerical attributes and categorical. As the algorithms are tree based they do not require any normalization, also we don't have any missing data in the datasets, so there were not applied any pre-processing operations on the datasets. Second thing which have to be mentioned is that hypera-maeter NT number of trees was limited to 50 instead of 100 as it takes a lot of time big datasets.

4.1 Wine

Wine dataset is the smallest dataset, it consists 179 numerical rows, in Table 1 we can observe results of the classifiers, from the results we can observe that at the beginning both algorithms have same accuracy, when numer of trees is 1 or 10 but as it grows to 25 the Random Forest return way better results, about 98% with only 3 trees, which is a good results.

NT	F	accuracy
1	3	81.36
1	4	76.27
1	9	84.75
1	10	96.61
10	3	83.05
10	4	84.75
10	9	91.53
10	10	86.44
25	3	69.49
25	4	71.19
25	9	94.92
25	10	93.22
50	3	81.36
50	4	91.53
50	9	94.92
50	10	91.53

(a) Decision Forest

NT	F	accuracy
1	1	47.46
1	3	83.05
1	4	93.22
1	3	77.97
10	1	79.66
10	3	98.31
10	4	98.31
10	3	98.31
25	1	71.19
25	3	98.31
25	4	98.31
25	3	98.31
50	1	84.75
50	3	100.0
50	4	98.31
50	3	98.31

(b) Random

Table 1: Metrics of the rise classifier on the test wine set

On this dataset both algorithms works well as expected because datased is relatively small and even if one features we can achieve some relative good results. However for this dataset most of tree define field Alchool in dataset as the most important one, feature importance ca be found in folder `/out/wine.out`

4.2 Breast cancer

Second dataset is the Breast cancer, we can observe here similar picture as in wine, results are similar still random forest has better results than decision forest, even though at the begining the decision forest has better results. We can clearly say that algorithms are working well with small-mid size datasets . Regarding features importance is hard to analyze as they are picked mainly random and represent only numerical attributes

NT	F	accuracy
1	2	90.04
1	3	92.21
1	6	94.37
1	4	96.1
10	2	94.37
10	3	93.51
10	6	92.21
10	4	93.94
25	2	89.18
25	3	93.07
25	6	92.64
25	4	95.24
50	2	91.34
50	3	91.77
50	6	93.94
50	4	93.51

(a) Decision Forest

NT	F	accuracy
1	1	89.18
1	3	94.37
1	4	90.04
1	3	94.37
10	1	94.81
10	3	94.81
10	4	96.97
10	3	95.67
25	1	97.84
25	3	98.27
25	4	97.84
25	3	97.84
50	1	97.4
50	3	98.27
50	4	96.54
50	3	98.27

(b) Random Forest

Table 2: Metrics of classifiers on the test breast-cancer set

4.3 Obesity

Obestiy is the last dataset on which were tested the algorithms also it is the biggest one one with 2110 entries which. In Table 3 we can see the results and here the picture is already completly different from the previous datasets as we can see, with small amounts of tree we cannot achive as good results as we did in the previous sets for example in random forest with just one tree we can achieve maximum 52% accuracy. Howeever if we look at the evolution of metrics in the table we can clearly say that for this dataset decision forest has better results, one of the reasons is the number of features, as we can see in some cases decision forest has 16 features comparing to 4 in random forest but the results are similar with only 3% difference, which means if we would increase the number of features we would get similar results to decision forest or even better. In most of cases for this dataset numerical features gain highest importance.

NT	F	accuracy
1	4	51.22
1	5	47.49
1	12	86.23
1	16	86.51
10	4	40.03
10	5	52.37
10	12	85.65
10	16	86.51
25	4	70.3
25	5	70.44
25	12	68.15
25	16	86.66
50	4	57.82
50	5	80.63
50	12	86.08
50	16	86.08

(a) Decision Forest

NT	F	accuracy
1	1	22.67
1	3	45.05
1	5	51.22
1	4	52.8
10	1	22.96
10	3	66.71
10	5	67.0
10	4	75.32
25	1	43.19
25	3	73.6
25	5	75.32
25	4	77.76
50	1	32.71
50	3	73.31
50	5	84.65
50	4	83.5

(b) Random Forest

Table 3: Metrics of the rise classifier on the test obesity set

5 Conclusion

During this practical assignment was gained knowledge regarding the implementation of CART and classifiers like Decision Forest and Random Forest. As we can saw in previous section we can achieve great results using this 2 algortithms, they have one major problem is the speed for big datasets and require higher number of trees and features in order to generalize better, so it is a point of further investigation how can this part can improved , also for further improvements will be good to implement pruning mechanism for CART and try to test the algorithms again and check how they behave. Unfortunately due to the lack of time these topics didn't been investigated in depth and the report have some gaps.

References

[1] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

[3] Wikipedia contributors. Random forest — Wikipedia, the free encyclopedia, 2022. [Online; accessed 17-May-2022].