Assignment 4 General Report
CS 412 FALL 2017
Yayi Ning

Notes for grader: used Binary-Tree in this mp

Introduction

This project implemented multi-value-attributes and multi-class decision tree. Each tree node contains the attribute and the attribute value (one value per split). The criteria used for choosing best next splitting point is Gini-Index.
Ensemble method implemented for better performance is RandomForest.

Implementation

Tree Structure:

Each node stores: [Attribute and one of its specific valve]
[Decision (give prediction)]
[Left child node]
[Right child node]

New data classification:

If node.Decision has value, then return the prediction. Otherwise, if current data attribute's value match with current node, sent to LeftChild. If not, sent to RightChild.

Tree Construction:

Build tree nodes recursively.
Split each attribute with its possible value. For example ([attr_1 = 1], [attr_1 = 2], [attr_1 = 3], [attr_1 = 4]) are considered as 4 separated possible splitting pints.
At each split, calculate the gini-index in for what left in attribute array.
Then partition the data into two list.
Build left child and right child the same way recursively.

Ramdom Forest:

To form N trees, for each tree:
Randomly choice [alpha]*original data length number of data from training data. Then construct a tree using bagged data.
Use majority vote policy to form final decision from N predictions.

Performance

Data: balance.scale

Training Data	balance.scale.train	balance.scale.train
Testing Data	**balance.scale.train**	**balance.scale.train**
Algorithm	Decision Tree	Random Forest
Accuracy	**0.1**	**0.8975**

Class	1	2	3	1	2	3
Specificity	1.0	1.0	1.0	1.0	0.9159	0.892
Recall	1.0	1.0	1.0	0.0	0.957	0.9679
Precision	1.0	1.0	1.0	0.0	0.9082	0.8873
F-1 Score	1.0	1.0	1.0	0.0	0.9319	0.9258
F-beta Score (beta = 0.5)	1.0	1.0	1.0	0.0	0.9175	0.9023
F-beta Score (beta = 2)	1.0	1.0	1.0	0.0	0.9468	0.9506

Training Data	balance.scale.train	balance.scale.train
Testing Data	**balance.scale.test**	**balance.scale.test**
Algorithm	Decision Tree	Random Forest
Accuracy	**0.7155**	**0.8355**

Class	1	2	3	1	2	3
Specificity	0.8965	0.8211	0.8306	1.0	0.7967	0.9032
Recall	0.0	0.8431	0.7425	0.0	0.9706	0.8812
Precision	0.0	0.7962	0.7812	0.0	0.7984	0.8812
F-1 Score	0.0	0.8190	0.7614	0.0	0.8761	0.8812
F-beta Score (beta = 0.5)	0.0	0.8052	0.7731	0.0	0.8278	0.8812
F-beta Score (beta = 2)	0.0	0.8333	0.75	0.0	0.9305	0.8812

Parameter analysis and conclusion for balance.scale:

Decision tree for balance.scale.train gives 1.0 accuracy since the tree is built according to the balance.scale.train. Tree is not pruned. When test using balance.scale.test, the accuracy decreased to 0.715. Later used ensemble method: random forest.

In random forest:

Number of tree built = 50

Lamda = 0.1 (Lamda is bagging size compare with whole data size)

Random forest have relatively low accuracy when using balance.scale.train for testing since it only using portion of the data, randomly. However it boost 10% more accuracy for unseen testing data (balance.scale.test).

Therefore, generally speaking, for unseen data, random forest ensemble method perform well for 50 trees with (Feature = 4 attribute*5 value each) data.

Data: nursery

Training Data	nursery.train	nursery.train
Testing Data	**nursery.train**	**nursery.train**
Algorithm	Decision Tree	Random Forest
Accuracy	**1.0**	**0.9996**

	Decision Tree					Random Forest				
Class	1	2	3	4	5	1	2	3	4	5
Specificity	1.0	1.0	1.0	1.0	1.0	0.9998	1.0	0.9996	1.0	1.0
Recall	1.0	1.0	1.0	1.0	1.0	0.9993	1.0	0.9996	1.0	1.0
Precision	1.0	1.0	1.0	1.0	1.0	0.9996	1.0	0.9992	1.0	1.0
F-1 Score	1.0	1.0	1.0	1.0	1.0	0.9994	1.0	0.9994	1.0	1.0
F-beta Score (beta = 0.5)	1.0	1.0	1.0	1.0	1.0	0.9996	1.0	0.9993	1.0	1.0
F-beta Score (beta = 2)	1.0	1.0	1.0	1.0	1.0	0.9993	1.0	0.9995	1.0	1.0

Training Data	nursery.train	nursery.train
Testing Data	**nursery.test**	**nursery.test**
Algorithm	Decision Tree	Random Forest
Accuracy	**0.9918**	**0.9879**

Decision Tree	Random Forest

Parameter analysis and conclusion for nursery:

Class	1	2	3	4	5	1	2	3	4	5
Specificity	0.9942	1.0	0.9943	1.0	0.9996	0.9917	0.9996	0.9913	1.0	0.9998
Recall	0.9881	0.9538	0.9902	1.0	0.0	0.9806	0.9385	0.987	1.0	0.0
Precision	0.9881	1.0	0.9877	1.0	0.0	0.983	0.9839	0.9812	1.0	0.0
F-1 Score	0.9881	0.9764	0.9889	1.0	0.0	0.9818	0.9606	0.9841	1.0	0.0
F-beta Score (beta = 0.5)	0.9881	0.9904	0.9882	1.0	0.0	0.9825	0.9744	0.9824	1.0	0.0
F-beta Score (beta = 2)	0.9881	0.9627	0.9897	1.0	0.0	0.9811	0.9472	0.9858	1.0	0.0

Decision tree for nursery.train is 1.0 accuracy since the tree is built according to the nursery.train itself and trees are not pruned. But when test using nursery.test, the acc decreased a little bit. Later used ensemble method: random forest.
In random forest:

Number of tree build = 10
Lamda = 0.6 (Lamda is bagging size compare with whole data size)

Random forest have relatively low accuracy when using balance.scale.train for testing since it only using portion of the data randomly. Random forest did not boost any acc but in fact decrease some accuracy (tiny) on this data, since the original decision tree give good enough accuracy. The randomness and bagging will only add unstableness here. Therefore, decision tree algorithm data is good enough for unseen data (nursery.test with (Feature = 8 attribute*3 value each).

Data: led

Training Data	led.train	led.train
Testing Data	**led.train**	**led.train**
Algorithm	Decision Tree	Random Forest
Accuracy	**0.8596**	**0.8586**

Class	1	2	1	2
Specificity	0.8958	0.7774	0.8937	0.779
Recall	0.7774	0.8958	0.779	0.8937
Precision	0.7666	0.9014	0.7634	0.9018
F-1 Score	0.772	0.8986	0.7711	0.8977
F-beta Score (beta = 0.5)	0.7688	0.9003	0.7665	0.9002
F-beta Score (beta = 2)	0.7752	0.8969	0.7758	0.8953

Training Data	led.train	led.train
Testing Data	**led.test**	**led.test**
Algorithm	Decision Tree	Random Forest
Accuracy	**0.8554**	**0.8563**

Class	1	2	1	2
Specificity	0.8889	0.7806	0.8902	0.7806
Recall	0.7806	0.8889	0.7806	0.8902
Precision	0.759	0.9004	0.7611	0.9005
F-1 Score	0.7697	0.8946	0.7707	0.8953
F-beta Score (beta = 0.5)	0.7632	0.8981	0.7649	0.8984
F-beta Score (beta = 2)	0.7762	0.8912	0.7766	0.8922

Parameter analysis and conclusion for led:

Decision tree for led is 0.8596 accurate rate. This acc is not 1.0. I believe the reason behind it is that the class number is only 2 and when there is no enough attribute, the decision tree then, just use majority vote rule. Since we only have [attribute = 7*2], ensemble method, random forest does not boost much accuracy like the other data.

In random forest:

Number of tree build = 50

Lamda = 0.5 (Lamda is bagging size compare with whole data size)

Random forest have relatively low accuracy as expected. I believe this time since the attributes are very limited (7*2 possible splits). Therefore both decision tree and random forest have around 0.85 for accuracy rate and does not change much no matter how to adjust the parameters in random forest. One conclusion can be draw from this data set: too little attributes and their possible value will limit the accuracy for decision tree and random forest classifier (compare to other data sets).

Data: synthetic.social

Training Data	synthetic.social.train	synthetic.social.train
Testing Data	**synthetic.social.train**	**synthetic.social.train**
Algorithm	Decision Tree	Random Forest
Accuracy	**0.1**	**0.9207**

Class	1	2	3	4	1	2	3	4
Specificity	1.0	1.0	1.0	1.0	0.9837	0.9737	0.9704	0.9663
Recall	1.0	1.0	1.0	1.0	0.8948	0.8954	0.9388	0.953
Precision	1.0	1.0	1.0	1.0	0.9465	0.9197	0.9161	0.9033
F-1 Score	1.0	1.0	1.0	1.0	0.9199	0.9074	0.9273	0.9275
F-beta Score (beta = 0.5)	1.0	1.0	1.0	1.0	0.9357	0.9147	0.9206	0.9128
F-beta Score (beta = 2)	1.0	1.0	1.0	1.0	0.9047	0.9001	0.9342	0.9426

Training Data	synthetic.social.train	synthetic.social.train
Testing Data	**synthetic.social.test**	**synthetic.social.test**
Algorithm	Decision Tree	Random Forest
Accuracy	**0.4409**	**0.678**

Class	1	2	3	4	1	2	3	4
Specificity	0.7284	0.8339	0.8548	0.8376	0.8975	0.9086	0.8724	0.8926
Recall	0.4851	0.4531	0.5345	0.338	0.6306	0.6082	0.7543	0.7255
Precision	0.3652	0.4387	0.496	0.4979	0.6926	0.6835	0.641	0.6981
F-1 Score	0.4167	0.4458	0.5145	0.4027	0.6602	0.6436	0.6931	0.7115
F-beta Score (beta = 0.5)	0.3842	0.4415	0.5032	0.4549	0.6793	0.667	0.6609	0.7034
F-beta Score (beta = 2)	0.4552	0.4501	0.5263	0.3612	0.6421	0.6219	0.7286	0.7198

Parameter analysis and conclusion for synthetic.social:

When using synthetic.scocial.train as testing data Decision tree give 1.0 accurate rate. However when using synthetic.scocial.test as testing data, the accuracy drop to only 0.44. After use ensemble method, random forest, the accuracy increased about 20%.

In random forest:

Number of tree build = 100

Lamda = 0.5 (Lamda is bagging size compare with whole data size)

ALPHA = 0.2 (only for synthetic.scocial data to randomly choice 20% attributes as possible split point)

Random forest have boost accuracy by setting less attribute. Compare with random forest, decision tree used up almost all attribute. In fact, before parameter ALPHA was added to prune attribute, even random forest did not boost the accuracy much. Therefore, I believe that when there are too much attributes, the dependency between attributes will have negative effect on model. Therefore, when encounter large amount of attribute (number of attribute * each possible value > 50). Pre-prune on attribute is very essential.