J48 Improvement and Analysis

DV2542 Project Report

Chai Yi,9008308232(chyi13)
Blekinge Institute of Technology
chyi13@student.bth.se

***Abstract*—This paper is the project report for Machine Learning course DV2542. According to my reading and research on articles about the Forest Covertype data set, C5.0 produces the highest classification accuracy among all algorithms and is the state-of-the-art approach for Covertype data set. This paper describes a J48 entropy gain ratio adjustment method and a J48 bagging method to reach a similar accuracy as C5.0. Then statistical test is conducted to compare the accuracies of these two methods with the performance of C5.0. The conclusion is the method could improve J48 to achieve a equal result as C5.0 on Cover type dataset.**

Keywords—C5.0; J48; entropy gain ratio; Cover type; optimization;

I. INTRODUCTION

The Forest Covertype data from UCI repository describes the actual forest covert type data of four wildness-areas located in Colorado, USA. There are some interesting characteristics of the dataset: The dataset has a large number of instances. It contains 581,012 instances and no missing value. The data set has a large number of attributes and the 54 attributes of this dataset actually represent 12 features. The dataset has both numeric and nominal attributes that first 7 attributes are numeric and rest 46 attributes are binary which only have 0 and 1. The classification task is to predict the Forest cover type which contains 7 classes[1].

According to Rajni Jain and Sonajhania Minz [2], they split cover type dataset into two equal parts randomly. After that, they compared some different algorithms on the split datasets, they proposed that back propagation having an accuracy 70.0%, J48 having an accuracy of 82.3% and the best performance is C5.0, whose accuracy is 83.7%. So I choose C5.0's result as the state-of-the-art performance on forest cover type dataset.

The report is structured as follows: Section II describes how data is sampled and experimental design details. Section III describes data analysis and two optimization methods. Section IV describes the methods implementation. Section V is the experiment result and the comparison of my approach with WEKA J48 and C5.0. Section VI is the conclusion of experiment. Section VII is the course reflection.

II. EXPERIMENTAL DESIGN OF DATASET

Known from C5.0 website in [3], the test set accuracy of C5.0 which uses 10-trial boosting is 96.6%. But the training set is half of total dataset which has 290,506 instances and the other half is used for testing. It is impossible for me to train or test with such large dataset. So I choose a sample approach which is used in [2].

The forest cover type dataset were arranged without balancing class frequencies in [2]. The dataset was randomly split into two equal parts (Using SPSS software): training set and testing set. And 10 random sample for both training and test data that each set has approximately 29000 instances and the distribution of each dataset was nearly the same as the original data set. And each training set is associated with a test set.

Since SPSS software is not available, I sampled and split the dataset by Resample filter of weka. The approach is:

1) Convert covetype.data and covtype.name to ARFF file which could be used by WEKA.

A ARFF file has two parts : Header information which contains the names of relation and the attributes names and types, Data information contains all data instances and are separated by comma. The following pic shows the result ARFF file.

```
@relation  'cover type'
@attribute 'Elevation'          NUMERIC
@attribute 'Aspect'             NUMERIC
@attribute 'Slope'              NUMERIC
....
@attribute 'Cover_Type'
                  {'1','2','3','4','5','6','7'}
@data
2596,51,3,258,0,510,221,232,148,6279,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,5,
2590,56,2,212,6,390,220,235,151,6225,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,5
....
```

- Randomize all instances in dataset. The original data in cover type dataset is not randomized.

```
Random  rand = new Random((int) System.currentTimeMillis());
data.randomize(rand);
```

- Split sample into two equal part: training data and test data.
- Apply supervised resample class. The class distribution of sample datasets should be the same as the original dataset. The resample method is executed 10 times to get 10 different training and test data sets.

```
weka.filters.supervised.instance.Resample sr = new weka.filters.supervised.instance.Resample();

sr.setNoReplacement(true);

sr.setSampleSizePercent(10); // 10% 29000

sr.setRandomSeed((int) System.currentTimeMillis());

sr.setInputFormat(largeTrain); // set sample input

    resampleData[i * 2] = Filter.useFilter(largeTrain, sr);
```

III. DATA ANALYSIS AND OPTIMIZATION METHODS

A. Covtype dataset analysis and Information Entropy

Seen from the Covertype dataset, first 10 attributes (Elevation, Aspect, Slope, Horizontal_Distance_To_Hydrology, Vertical_Distance_To_Hydrology, Horizontal_Distance_To_Roadways, Hillshade_9am, Hillshade_Noon, Hillshade_3pm, Horizontal_Distance_To_Fire_Points) are numeric attributes.

And I conducted a statistical analysis on these attributes in one sample dataset which is created in II. The small dataset has 29050 instances and it could reflect the entire dataset distribution.

TABLE I. TEN NUMERIC ATTRIBUTES ANALYSIS

Attribute	Mean	StdDev
Elevation	2961.92	279.224
Aspect	154.983	111.509
Slope	14.092	7.49
Horizontal_Distance_To_Hydrology	270.265	212.818
Vertical_Distance_To_Hydrology	46.131	57.937
Horizontal_Distance_To_Roadways	2336.906	1550.457
Hillshade_9am,	212.371	26.832
Hillshade_Noon	223.222	16.619
Hillshade_3pm	142.162	38.327
Horizontal_Distance_To_Fire_Points	1990.942	1329.231

As seen from the result table, the Standard Deviation ranges from 7.49 to 1550.457 which means it has a large difference. So I considered the Standard Deviation difference as my research point.

1) Standard Deviation

Standard deviation reflects the class distribution of the dataset. If an attribute has a low standard deviation, it indicates the data values tend to be very close to its mean value. the distribution is simpler. If an attribute has a large standard deviation, it indicates the data values spread out over a large range of values and it is highly randomized [4].

2) Information Entropy

Information entropy describes the expected value of the information in an attribute. It is a measure of the randomness of the values of an attribute [5]. And In our forest cover type scenario, we could say that with a larger standard deviation, the information entropy will be larger as well.

B. Entropy Gain Ratio Optimization Method (MJ48)

J48 is the java implementation of C4.5 in Weka. It builds decision trees from training dataset like ID3 but uses information gain ratio. J48 builds decision tree by choosing an attribute with the largest information entropy gain ratio as current split node [6]. Therefore J48 is considered as a local optima algorithm.

And from my last version of report, I used boosting to improve the result. Boosting describes a reweighted technique: It increases weight in a wrong classified instance and the reweighted instances could be used in the next boosting loop. So based on this idea, I introduced a balancing coefficient into entropy gain equations to increase or decrease an attribute information entropy.

The brief process is as below: If a numeric attribute has a large standard deviation, the information entropy will be multiplied by a larger balancing coefficient α, its split information will multiply by a smaller coefficient β, α and β are determined by standard deviation. And on the other side, a attribute with a small standard deviation will multiply by a smaller α and a larger β.

The following equation is the original entropy calculation (1) and modified version (2):

$$\text{info}(D) = \sum_{i=1}^{y} p_i \log_2(p_i) \tag{1}$$

$$\text{info}(D) = \sum_{i=1}^{y} \alpha * p_i \log_2(p_i) \tag{2}$$

Split information gain in (3) also need to be changed by β (4).

$$\text{splitInfo}_A(D) = -\sum_{i=1}^{y} p_{D_i} \log_2 p_{D_i} \tag{3}$$

$$\text{splitInfo}_A(D) = -\sum_{i=1}^{y} \beta * p_{D_i} \log_2 p_{D_i} \tag{4}$$

For Large standard deviation (StdDev>200) α, β is calculated in (5), for small standard deviation (StdDev<200) is in (6), StdDev means the standard deviation of current attribute values.

$$\alpha = 1 + \log_{10}(StdDev)/2.2 \quad \beta = 1 + \log10(StdDev)/10 \tag{5}$$

$$\alpha = 1 + \log_{10}(StdDev)/10 \quad \beta = 1 + \log_{10}(StdDev)/2.2 \tag{6}$$

The reason why to choose 200 as the cutoff point is: according to Table I, 200 is large enough for a small standard deviation which could divide attributes into two parts, and 200 is the best value according to my tests.

I use $\log_{10}$ to narrow the possible range of standard deviation. Because standard deviation could range from 1 to 2000, but α and β should be in a range between 1 to 3, since larger or smaller coefficient α, β will receive a lower accuracy.

C. MJ48 Bagging Method

Another method to improve J48 is bagging.

Bagging is based on bootstrap resampling. But I have created resample data sets in section II, so I use voting to generate the final cllasifier. The procedure is as follows:

For each MJ48 classifier{

Use training dataset to train the classifier.

Use pair test dataset to test the classifier and get the accuracy

}

The new instance will be classified by every MJ48 classifier. And the instance will be classified as the class which gets most votes.

IV. Implementation

A. MJ48

1) C45Spilt.java

Function void buildClassifier(Instances trainInstances) is used to process every attributes and to choose the best attribute node. In this function, I can get the standard deviation of every numeric attribute and calculate α and β.

```
if (trainInstances.attribute(m_attIndex).isNominal()) {
....
}else{
....
Double stdDev = trainInstances.attributeStats(m_attIndex).numericStats.stdDev;
if (stdDev > 200)
        alpha = Math.log10(stdDev)/2.2+1;
        beta = Math.log10(stdDev)/10+1;
Else
        alpha = Math.log10(stdDev)/10+1;
        beta = Math.log10(stdDev)/2.2+1;
}
```

2) InfoGainSplitCrit.java

Function double splitCritValue(Distribution bags, double totalNoInst, double oldEnt, double alpha) calculates information gain, the parameter alpha represents the 'α' in the equation (2) and the return value of this function will be multiplied by alpha.

```
return alpha*numerator / bags.total();
```

3) GainRatioSplitCrit.java

Function double splitCritValue1(Distribution bags, double totalnoInst, double numerator,double beta), this function calculates attribute information gain ratio, the parameter beta represents the 'β'. The final return value multiply by 1/beta according to equation (4).

```
return numerator / (denumerator*beta);
```

B. Bagging MJ48

I use 5 MJ48 classifiers as basic voting classifiers.

- Void buildClassifier() creates 5 MJ48 classifiers and stores 5 corresponding test dataset accuracies in a double accuracy[5].

```
public void buildClassifier(Instances data) throws Exception {
        double total = 0.0;
        for (int i = 0; i<5; i++){
                mj48[i] = new j48.J48();
                mj48[i].buildClassifier(result[i*2]);
                Evaluation tEval = new Evaluation(result[i*2]);
                tEval.evaluateModel(mj48[i],result[i*2+1]);
                accuracy[i] = (1-tEval.errorRate());
        }
}
```

- Double classifyInstance() returns the majority class id.

```
public double classifyInstance(Instance inst) throws Exception{
        double classId = 0;
        double[] vote = new double[inst.numClasses()];
        for (int i = 0; i<5; i++){
                vote[(int)mj48[i].classifyInstance(inst)]+= accuracy[i];
        }
        double max = -100;
        for (int i = 0; i<inst.numClasses(); i++)
                if (vote[i]>max){
                        max =  vote[i];
                        classId = i;}
        return classId;
}
```

V. Experimental Result

A. MJ48

Every training dataset and test dataset has 29050 instances. I chose 5 resample data which had 5 training data and 5 test data. Thus I did 5 different test both in MJ48 and Weka's J48. The result is in Table II.

TABLE II. MJ48 RESULT

Data No	**MJ48**			**J48**		
	accuracy	**Leave num**	**Node num**	**accuracy**	**Leave num**	**Node num**
1	83.676%	2149	4297	82.044%	1824	3647
2	84.144%	2109	4217	81.173%	1938	3875
3	83.883%	2119	4237	81.172%	1872	3743
4	83.408%	2106	4211	81.750%	1854	3707
5	83.391%	2139	4277	81.644%	1802	3603

1) Compare with the-state-of-art performance(C5.0)

The accuracies of five runs of MJ48 are 83.676%, 84.144%, 83.883%, 83.408%, 83.391%.

C5.0 is the-state-of-art performance algorithm in this dataset. The accuracy is 83.7% with 29000 instances in [2].

I conducted One Sample T test with 83.7% in C5.0. For significance level 0.05, t = 0.0028, p-value = 0.9979, and 95% of confidence interval is from 83.30193 to 84.09887. According to t-value table, df = 4 significance level = 0.05, t = 2.7764. t = 0.0028 < t = 2.7764 p-value = 0.9979 > 0.05, So I accepted null hypothesis: there is no significant difference between MJ48 result and C5.0 result.

2) Compare with J48

The J48 accuracies are 82.044%, 81.173%, 81.172%, 81.750%, 81.644%.

I conducted paired T test with MJ48 and J48. For significance level 0.05, t = 7.4832, p-value = 0.001745, and the 95% of difference confidence interval is from 1.343585 to 2.944015. P-value = 0.001745 < 0.05, So I rejected null hypothesis: MJ48 is different from J48 and MJ48 is better.

From Table II, we could know that MJ48 has a larger number of nodes than J48. It indicates MJ48 creates more split nodes and decision branches.

B. Bagging MJ48

TABLE III. BAGGING RESULT

Data No.	Accuracy
1	89.039%
2	89.243%
3	89.328%
4	88.929%
5	89.102%

In Table III, I got 5 runs of bagging MJ48 result accuracies (same training and test dataset as MJ48 in section A): 89.039%, 89.243%, 89.328%, 88.929% and 89.102%.

And I conducted paired T test with MJ48 and bagging MJ48. For significance level 0.05, t = -54.0766, p-value = 7e-07 < 0.05. So I rejected null hypothesis : bagging MJ48 is significant different from MJ48 and it has a better performance.

VI. CONCLUSION

From the experiment result in V, I found that MJ48 improves Weka's J48 performance. Under same criteria, MJ48 produces the same performance as C5.0 in [2]. On the other hand bagging algorithm improves the performance of MJ48 significantly on forest cover type dataset.

VII. REFLECTIONS

Before taking this machine learning course, I took machine learning course in Cousera. I find it is totally different in DV2542. In cousera machine learning is about learning several algorithms and implementing them with octave. After learning DV2542, I find the previous course is a kind of machine learning introduction. Because DV2542 is more concentrated in searching ML articles, implementing algorithms which I found and conducting statistical testing on algorithms. It is about real approaches and real problems.

In DV2542, I learned basic machine learning concept of supervised learning and unsupervised learning and some classical algorithms: ID3 and C4.5. In terms of statistical test I also learned t-test which is used to compare two different algorithm on same dataset and Mann-Whitney U test is to compare on multiple datasets. So in project development, J48, t-test, and related articles analyses are three important parts which I learned from this course.

REFERENCES

[1] Jock A. Blackard,Dr. Denis J. Dean, Covertype Data Set,1998,http://archive.ics.uci.edu/ml/datasets/Covertype

[2] Rajni Jain,Sonajhania Minz,"Drawing conclusion from forest cover type data-the hybridized rough set model".Journal of the indian society of agricultural statistics,62(1),pp.75-84.2008

[3] Rulequest Research,Data Mining Tools See5 and C5.0,2013,http://rulequest.com/see5-comparison.html.

[4] Jan,Is Standard Deviation the same as Entropy ,2014, http://math.stackexchange.com/questions/651077/is-standard-deviation-the-same-as-entropy

[5] Wikipedia, Entropy(Information theory), 201401,http://en.wikipedia.org/wiki/Entropy_(information_theory)

[6] Wikipedia, C4.5 algorithm , 2013, http://en.wikipedia.org/wiki/C4.5_algorithm