COMP5318 - Machine Learning and Data Mining
assign2 report

Shen, Chun cshe6391@uni.sydney.edu.au - 460317940
Shahrasari, Delaram dsha6256@uni.sydney.edu.au - 470183205
Zhou, Chengcheng czho9311@uni.sydney.edu.au - 460090157

June 5, 2017

Abstract

In machine learning and data mining areas, different data sets inherent diverse data features, and different algorithms cater for distinctive applicable fields. It is difficult to find out an absolutely universal method to suit for any datasets. Hence, finding out an optimal algorithm for a classification problem in data mining seems to be very necessary and important.In this section, three classifier based on three classification algorithms, Decision Tree, Random Forest and Logistic Regression, were implemented to classify a large-scale dataset Forest Cover Type[5]. In order to increase execution efficiency and optimize classifier, three methods were used to preprocess the datasets and stratified k-fold cross-validation were executed to leverage imbalanced label distribution.The performances of this three models are different and all improved after optimization, and the optimal one which proves to suit for this classification problem with respect to highly execution efficiency and promising accuracy is Random Forest.

1 Introduction

Predicting forest cover types from seven types of tree species is a classification problem. The Forest Cover Type dataset[5] used in this classification project is derived from US Forest Service inventory information which mainly focuses on four wildness areas including Neota, Rawah, Comanche Peak and Cache la Poudre situated in northern Colorado state's Roosevelt National Forest. This dataset consists of 581,012 observed samples and 54 attributes recording from 12 measures with variant units which possess 10 unprocessed data in quantitative type , 44 binary variables with qualitative processing as well as has 7 types of tree species with integers from 1 to 7.
The object of this study is to classify forest cover types building predictive models based on three different classification algorithms,Random Forest, Decision Tree and Logistic Regression and to evaluate the performance of each classifier. To some degree, classifying the diversity of forest cover by this study has an significant role in further research on taking advantages of forest resources reasonably and sustainably such as relieving deforestation concerns and avoiding the spread of forest insects[4].

2 Previous Work

Most previous works on classifying forest cover type using Cartographic Features are mainly focused on artificial neural networks, discriminant analysis, Random Forest, Support Vector Machine and Decision Tree. The first breakthrough on predicting forest cover type using artificial neural networks(ANN) and discriminant analysis predictive model was implemented significantly by Jock and Denis in 1999 and the overall accuracy of ANN based on a feedforward technique was more promising than the overall accuracy of Gaussian discriminant analysis, 70.58% and 58.38% respectively[2]. According to the mathematic properties of SVMs, the performance of SVMs classifying those large-scale dataset was inefficient. Yu et al in 2003 reported a new method, SVMs with Clustering-Based SVM (CB-SVM) applying a hierarchical micro-

clustering algorithm which could deal with the large size of data sets efficiently and receive high classification accuracy[8].Similarly, Kevin and Graham in 2004 proposed another attempt also based on SVMs predictive model. They applied K-means algorithm clustering data sets and removed all binary features to preprocess the dataset in order to increase the operating efficiency and reduce overfitting.
Ensemble algorithm used to classify the types of forest cover was evaluated by Chandra et al. which performed extremely well and obtained a maximum accuracy of 88.14%.Pruthvi et al. in 2015 applied some techniques to preprocess this dataset for extracting and select more relevant features and the result of using those preprocessing techniques shows excellent accuracies[7].

3 Classifier Technology

The technologies we used to solve this classification problem are analyzed in the following sections.

3.1 Decision Tree

Decision trees are a non-parametric supervised learning method which can be used for classification and regression. It is used as an algorithm which little data preparation and can work with both categorical and numerical data. The strategy is to create a model to learn decision rules which is derive from data features and predict the value of a target variable. This method take two arrays as input: array x of size $[n_samples, n_features]$holding the training samples, and an array of integer values, size $[n\_samples]$, holding class labels for the training samples. The model which is being fitted will be used for predicting new values. Using decision trees for over-complex trees that do not generalize the data well which is called overfitting is not stable due to the variations that can make different trees thus we used random forest to improve the performance.

3.2 Random Forest

Random Forest, aka random decision forest is one of ensemble learning methods. This method was initially proposed by Ho, Tin Kam in 1995 and was built upon decision tree theories with ensemble methodology. As ensemble methods take the strategy of grouping "weak learner" to form "strong learner", random forest groups decision tree classifiers together to improve decision tree technology itself. Random decision forest corrects for decision trees' habit of overfitting to their training set. In the training part, random forest applies bagging technology to tree learners. Given a training set X = x1, ..., xn with responses Y = y1, ..., yn, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples: For b = 1, ..., B: Sample, with replacement, B training examples from X, Y; call these Xb, Yb. Train a decision or regression tree fb on Xb, Yb. After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

or by taking the majority vote in the case of decision trees. [1]
In the learning process, when each candidate should be split, a random subset of the features are considered. This operation is called "feature bagging". In an ordinary bootstrap sample, features which are very strong predictors for the response variable always be selected in many trees. In that case, these trained trees are correlated with each other and are not able to increase accuracy. However, bagging and random subspace projection contribute to accuracy gains.

3.3 Logistic Regression

Logistic regression is a technique for binary classification problems. More specifically, logistic regression seeks to describe data and predict the probability of a series of independent variables on a binary response variable, where two values "0" and "1" can be taken to represent two outcome categories, and classify the observations via the probability that has been estimated[6]. However, in this section, logistic regression could also be used in multi-classification problem. In order to train and optimize logistic regression classifier, the concept below should be considered:

- Hypothesis representation

The hypothesis in logistic regression could be represented by Sigmoid function seem like a S shape whose values returned increasing from 0 to 1 on $\mathbb{R}$ monotonically. Combining the hypothesis representation in linear regression with sigmoid function , the hypothesis used in logistic regression should be shown as the two equations below if only two classes'1' and '0' need to classified:

$$P(y=1|x) = h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \equiv g((\theta^T x))$$

$$P(y=0|x) = 1 - P(y=1|x) = 1 - h_\theta(x)$$

• Cost function
In order to optimize logistic regression classifier using gradient descent method to reach a global minimum, the cost function used for logistic regression can be put forward and simplified below:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}, y^{(i)}))$$

$$Cost(h_\theta(x), y) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

After integrate those equations together, the cost function with parameter θ could be represented.
• Gradient descent method
Gradient descent method is based on the cost function with θ parameter as mentioned. The object of Gradient descent method is to find a fit θ parameter with reducing cost function to minimum repeatedly.

$$Repeat\{\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)\}$$

4 Experiment and Discussion

The experiment section heavily relies on third party library scikit-learning[3].

4.1 Preprocessing Details

The previous three methods are applied in our experiment. Before training our dataset, preprocessors are conducted to the dataset as follows:

1. Naive feature extraction: Given reduction of binary data does not damage performance heavily. In that case, binary data consists of 44 columns are wiped out of dataset to improve executing runtime. The comparison between with and without 44 binary columns is covered in Table 1.

with 44 columns of binary data	61.9%
without 44 columns of binary data	61.0%

Table 1: Compare accuracy between with or without 44 columns based on Random Forest with default parameter

2. Means of all columns are subtracted in order to center the dataset.

3. All columns are scaled to unit variance. Step 2 and 3 are designed as regulation routine.

4. PCA is applied to reduce dimension from 10 to 3. However this operation is applied for trial only in that the following performance is based on 10 dimension dataset.

4.2 Cross Validation Details

After three classifiers above are operated on k-fold cross-validator, we found that not all confusion matrices of each fold share the same dimension. That means labels are not distributed randomly in dataset in that case some folds leak of specific labels. To leverage the impact from imbalance of label distribution, a stratified k-fold cross-validator is applied to our experiment to ensure labels are sampled evenly in every fold. In that all performance measure below is based on this kind of k-fold function in the scikit-learning library.

4.3 Experiment Process

4.3.1 Decision Tree

Decision tree is one of quite simple classifier with limited parameters to tune. The considered parameters are listed as follows:

1. Criterion: This is the parameter for measuring the quality of a split. Gini and Entropy are typical functions for this aim. Gini relies on CART algorithm to calculate impurity while the common algorithm behind Entropy are ID3, C4.5. We conducted trial on both of them and the result lists in Table 2. From the result, entropy is more suitable to this dataset.

Type	Accuracy	Time
Gini	57.53%	59s
Entropy	58.99%	64s

Table 2: Type of Decision Tree Criterion with 10-fold validation

2. Feature number: This is another parameter we tuned in this experiment. Since 10 columns are extracted from original dataset, we brute force all possible feature numbers and the result is presented like Table 3, plotted in Figure 1.

Feature Number	Accuracy	Runtime
1	52.34%	15s
2	55.73%	18s
3	57.37%	22s
4	57.30%	27s
5	58.13%	32s
6	57.9%	41s
7	58.0%	44s
8	58.41%	49s
9	58.24%	53s
10	58.88%	59s

Table 3: Accuracy Runtime by different feature numbers(10 fold validation)

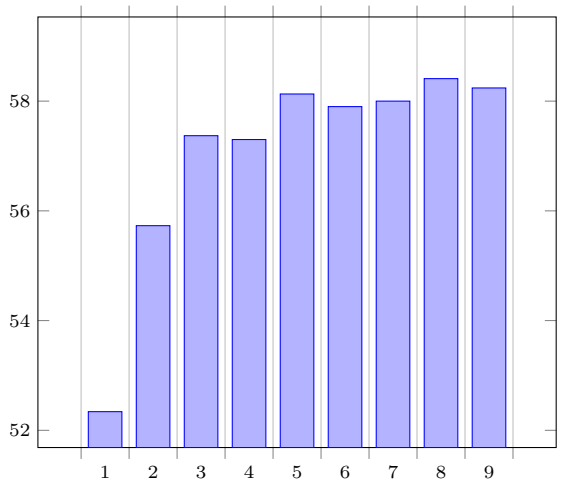

Figure 1: Accuracy(%) Feature(n) Relationship

3. Class Weight: This parameter help to set weight for every class. The class(label) distribution is not even in the covtype dataset. To deal with this problem, 'balanced' value is set to balance the weight of each class.

	1	2	3	4	5	6	7
1	7204	12797	7	0	977	10	189
2	1108	21442	194	2	5310	273	2
3	0	162	2932	114	4	364	0
4	0	0	81	172	0	22	0
5	26	126	0	0	795	3	0
6	3	115	162	12	2	1443	0
7	130	20	0	0	0	0	1901

Table 4: Confusion Matrix of Decision Tree of 1st Fold

4.3.2 Random Forest

Random Forest is ensemble version of decision tree. Partial parameters from these two classifier are overlapped. To fine tune the Random Forest classifier, follow parameters are considered in this experiment:

1. max_features: Sqrt are set in this experiment because to improve the diversity of trees feature selection should represent difference in different trees. sqrt is recommended value for random forest and we test this parameter from sqrt to maximum value which is 10 in our experiment. The accuracy is listed as Table 5.

Max Feature	Accuracy
sqrt(3.16)	61.01%
log2(3.3)	61.01%
5	60.90%
10	58.72%

Table 5: Accuracy by different max feature(n_estimators = 10)

2. n_estimators: This parameter reflects the amount of trees in trained for Random Forest. According to theory of Random Forest, the more trees introduced into the algorithm, the better the performance will achieve. The relationship between them are listed in the Table 6, plotted on Figure 2. The runtime of every test includes result analysis period. From that table, it's found that the accuracy increases as more trees are used for training while the runtime boosts either.

Number	Accuracy	Runtime
1	55.18%	19s
2	56.53%	26s
3	58.46%	25s
4	59.08%	35s
5	60.08%	48s
6	60.18%	46s
7	60.7%	52s
8	60.65%	61s
9	60.97%	76s
10	61.01%	75s
15	61.65%	100s
20	61.8%	167s
50	62.46%	374s
100	62.59%	667s

Table 6: Accuracy Runtime by different estimator numbers(10 fold validation)

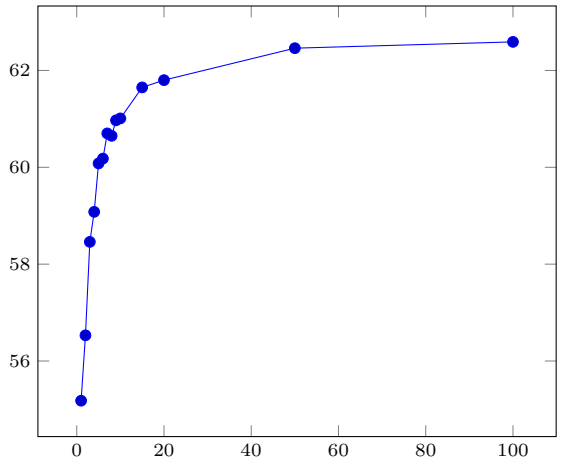

Figure 2: Accuracy(%) Estimators(n) Relationship

3. n_jobs: This parameter is designed for training the classifier with more cores. The number is the parallel threads. To make use of potential capability of the machine, -1 is set maximize the computation.

	1	2	3	4	5	6	7
1	7929	12971	0	0	257	0	27
2	733	23545	145	0	3854	53	1
3	0	151	3253	23	2	147	0
4	0	0	109	154	0	12	0
5	15	203	0	0	730	2	0
6	1	106	140	12	1	1477	0
7	139	16	0	0	0	0	1896

Table 7: Confusion Matrix of Random Forest of 1st Fold

4.3.3 Logistic Regression

Logistic Regression is binary classifier initially. To apply Logistic Regression to this multi-classification problem, some parameters are set and other parameters are considered to optimize the performance.

1. multi_class: This parameter decides the options that we can use to implement classification task, one-vs-rest(OvR) and many-vs-many(MvM). The difference between those two methods is mainly reflected on multinomial Logistic Regression. Applying for 'multinomial' is more accurate and the operating efficiency of this method is lower comparing with 'ovr' method.

2. penalty: This parameter is used to represent the norm used in the penalization. 'l2' can limit model spaces and avoid overfitting to some degrees.

3. solver: This parameter is used to optimize loss function and avoid overfitting. 'liblinear' can match penalty 'l1'and 'l2' as well as support 'multinomial' only. 'newton-cg','lbfgs' and 'sag' can match penalty 'l2' only. For large-scale dataset, 'sag' is the most ideal choice.

4. class_weight: This parameter can help to adjust and weight highly unbalanced samples.Classifier could calculate weight according to the number of training samples for each class if selecting'balanced'.

5. C: This parameter is the backward of regularization coefficient which means the lower the value of C is, the stronger regularization is. The comparison is presented in Table 8, plotted in Figure 3.When the value of C is 0.1, the classifier receives the highest accuracy.

C	Accuracy	Runtime
10^{-4}	41.09%	587s
10^{-3}	47.26%	584s
10^{-2}	49.39%	532s
10^{-1}	49.52%	573s
10^{0}	47.70%	565s
10^{1}	47.93%	596s
10^{2}	48.94%	597s
10^{3}	48.08%	556s
10^{4}	48.46%	578s

Table 8: Accuracy Runtime by different values of C (10 fold validation)

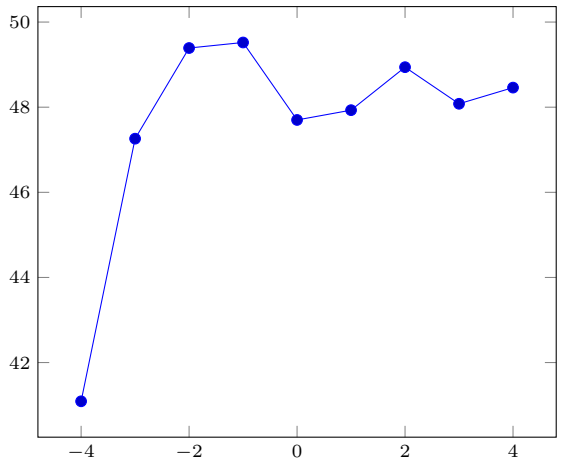

Figure 3: Average accuracy(%) and C(10^n)

6. n_jobs: This parameter is designed for training the classifier with more cores. The number is the parallel threads. -1 is set to maximize the speed of execution.

	1	2	3	4	5	6	7
1	7708	12213	1	1	686	9	566
2	1360	18638	75	364	5848	1998	48
3	0	1	2332	739	201	303	0
4	0	0	84	189	0	2	0
5	5	176	9	8	718	34	0
6	0	46	870	136	127	558	0
7	318	7	0	0	7	0	1719

Table 9: Confusion Matrix of Logistic Regression of 1st Fold

4.4 Comparisons and Evaluation

Since Decision Tree and Random Forests perform similarly, initial comparison begins with these two classifiers. After parameters are tuned according to experiments conducted previously, accuracy results are list as follows.

Type	Accuracy	Precision	Recall	Fscore
DT	58.99%	59.29%	66.12%	60.68%
RF	62.18%	67.02%	64.52%	63.67%

Obviously, Random Forest holds higher accuracy while there is a huge time consumption gap. Due to time consumption of training multiple trees in Random Forest, it runs slower than the simple Decision Tree classifier. To be more specific, the gap of run time on two classifier is exaggerated because 10-fold validation is applied in our experiment and the time tables displayed in the previous sections are the total run time of 10-fold validation.

When we compare Random Forest and Logistic Regression, performance index are considered as follows:

Type	Accuracy	Precision	Recall	Fscore
RF	62.18%	67.02%	64.52%	63.67%
LR	48.95%	40.92%	58.30%	40.12%

From the Figure 4 and Figure 5, we can analysis as follows:

- Compared to the Logistic Regression, The Random Forest performs better on this dataset with higher area under curve.

- The Random Forest owns better generalization level due to its stable performance between 10-folds

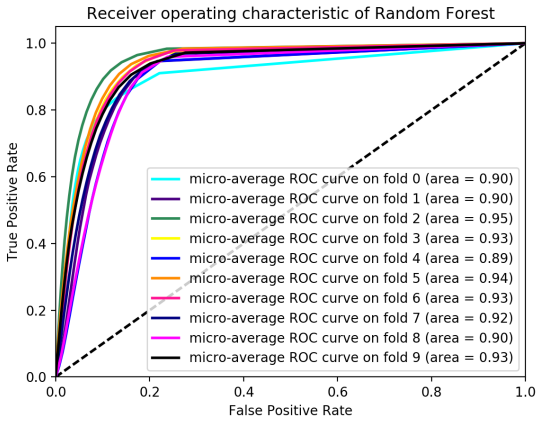

Figure 4: Random Forest ROC

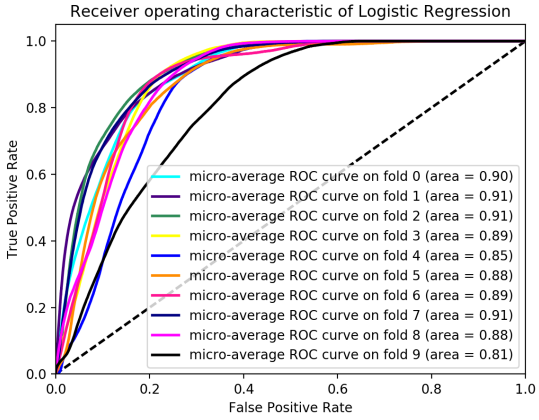

Figure 5: Logistic Regression ROC

5 Conclusions and the Future

According to performance of above three classifiers, the Random Forest Classifier is the best among them towards the covtype dataset. When experiments are conducted we found that:

1. The pre-process is not necessary for Decision Tree and Random Forests especially for the scale preprocessing stage. Since these two classifiers treating every attribute independently and building leave nodes based on individual calculation, the scale operation does no mean to the training of them. The accuracy of training with or without the scale of data stands still.
   The operation of scale rewards dataset transforms the original data type integer to float ranging from 0 to 1. This slow the training process heavily in that the execution time is almost 10 timed.
   In the future, pre-process will be dealt with separately to improve speed of Decision Tree and Random forest.

2. Decent feature extraction is helpful to increase overall classification performance. In our experiment, binary data is dropped but taken use of. Though, reduction of binary data slightly damage the performance, a improvement of model should be achieved if we combine them reasonably.
   In the future, feature extraction engineer will be conducted to filter and reorganize raw data such as splitting binary attributes into groups and reassigning symbols like the work described in previous work section.

3. The performance score is sensitive to selection of training and test dataset. Over-fitting and imbalance classification impact the mark of each classifier.
   In the future, a more precise performance score may be achieved if labels are evenly distributed.

4. To speed up the program, n-jobs value is tuned in the program which takes use of threads or processes of the local machine. However the optimization is limited.
   In the future, distributed computation infrastructure can be applied to boost the execution time.

6 Hardware & Software

The specific of hardware we conducted experiments upon is listed below:

Processor	2.7GHz Intel Core i5
Memory	8GB 1867 MHz DDR3
Storage	128GB SSD
Interpreter	python3.6.0(64bit)
Libraries	Numpy, Pandas, Matplotlib, Scikit-learning

A Code Usage Instructions

1. Append covtype dataset under data folder
2. Change work directory to code folder
3. Execute 3 separate classifiers through python 3 interpretor with third part library mentioned in Hardware & Software part.
4. "python lr.py" for run Logistic Regression
5. "python dt.py" for Decision Tree
6. "python random-forest.py" for Random Forest

B Contribution

Shen, Chun – Random Forest
Shahrasar, Delaram – Decision Tree
Zhou, Chengcheng – Logistic Regression

References

[1] A. GANDHIMATHI A. MALATHI. Design and development of an enhanced random forest method to reduce the attributes. Ho Chi Minh, Vietnam, 2016. Basha Research Corporation.

[2] Jock A Blackard and Denis J Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.

[3] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[4] RH Crockford and DP Richardson. Partitioning of rainfall into throughfall, stemflow and interception: effect of forest type, ground cover and climate. *Hydrological processes*, 14(16-17):2903–2920, 2000.

[5] M. Lichman. UCI machine learning repository, 2013.

[6] Llew Mason, Jonathan Baxter, Peter L Bartlett, and Marcus R Frean. Boosting algorithms as gradient descent. In *Advances in neural information processing systems*, pages 512–518, 2000.

[7] HR Pruthvi, KK Nisha, TL Chandana, K Navami, and RM Biju. Feature engineering on forest cover type data with ensemble of decision trees. In *Advance Computing Conference (IACC), 2015 IEEE International*, pages 1093–1098. IEEE, 2015.

[8] Hwanjo Yu, Jiong Yang, and Jiawei Han. Classifying large data sets using svms with hierarchical clusters. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 306–315. ACM, 2003.