A Two-Stage Tree based Meta-Classifier using Stack-Generalization

B. Kalpana M.Sc.M.Phil.,
PhD Research scholar,
Bharathiar university &
A.P, Department of computer science & Applications.
PSG college of arts and science, Coimbatore- 641014

Dr .V. Saravanan
Professor & Director, MCA Dept
Dr. NGP Institute of Technology
Kalapatti Road
Coimbatore – 641048

Dr .K. Vivekanandan
Professor
School of Management
Bharathiar University
Coimbatore - 641046

ABSTRACT
Given a choice of classifiers each performing differently on different datasets the best option to assume is an ensemble of classifiers. An ensemble uses a single learning algorithm, whereas in this paper we propose a two stage stacking method with decision tree c4.5 as meta classifier. The base classifiers are Naïve Bayes, KNN and C4.5 tree. The decision tree learns from the classification output given by base classifiers after feature selection in the first stage on training data. The second stage classifies the test data using meta classifier. We prove that our algorithm provides better classification accuracy with UCI datasets.

General terms: Data mining, Classification.

Keywords: Data mining, Classification, feature selection, stack generalization.

1. INTRODUCTION
Meta-learning focuses on predicting the right algorithm for a particular problem based on characteristics of the dataset [1] or based on the performance of other, simpler learning algorithms [2]. Stacking is concerned with meta-learning of ensemble learning schemes or meta-classification schemes. Wolpert [3] introduced an approach for constructing ensembles of classifiers, known as stacked generalization or stacking. A classifier ensemble, consists of a set of n classifiers C1, …Cn, called base-level classifiers and a meta-level classifier CML that learns how to combine the predictions of the base-classifiers. The base-classifiers are generated by applying n different classification algorithms on a labeled dataset, the training set, TRAINSET ={(xk , yk)}, where xk and yk are the features and the class value for the k-th instance vector respectively. The individual predictions of the base-classifiers on a different labeled dataset TESTSET, are used to train the meta-classifier CML. The predictions of the base-classifiers on TESTSET are then transformed into a meta-level set of classification vectors. At runtime, CML combines the predictions PM(x) = {Pi(x), i = 1…n} of the n base classifiers on each new instance x, from the test data and decides upon the final class value y(x). The final predictions of the base-classifiers on x are transformed into a single vector representation, which is then classified by CML.

2. BASE CLASSIFIERS AND META CLASSIFIER
For base level classifiers we have taken Naive Bayes , KNN and c4.5 where c4.5 also acts as the meta classifier.

2.1. Naive Bayes
- A Naive Bayes classifier is a simple probabilistic classifier based on applying *Bayes' theorem***.** Naivebayes classifier assumes all features are conditionally independent given the class label. Given D features, then

$p(x|y = c) = \prod_{i=1}^{D} p(xi \mid y = c)$, is the probability of x being in class c ,given y belongs to the same class. An advantage of the Naive Bayes classifier is that it only requires a small amount of training data to estimate the parameters ie the means and variances of the variables, necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined.

2.2. C4.5 decision tree
- An extension of ID3 algorithm[4] the C4.5 algorithm is used to generate a decision tree and is developed by Ross Quinlan. C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. The training data is a set $S = s_1,s_2,...$ of already classified samples. Each sample $si = x_1,x_2,...$ is a vector where $x_1,x_2,...$ represent attributes or features of the sample. The training data is augmented with a vector $C = c_1,c_2,...$ where $c_1,c_2,...$ represent the class to which each sample belongs. At each node of the tree, one attribute of the data that effectively splits its set of samples into subsets enriched in one class or the other is chosen by C4.5. Its criterion is difference in entropy i.e the normalized information gain , which results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is chosen to make the decision node. The process is repeated again on the smaller sub lists. The algorithm has a few special cases.

a. All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.
b. None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.
c. Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

The advantage of this classifier is, it can work on missing , numeric either discrete or continuous data. The missing value is not taken to consideration while calculating information gain. The type of attributes, whether numeric or nominal determines the format of the test outcomes. According to [5] for a numeric attribute *A* they are $\{A \le \theta, A > \theta\}$ where the threshold θ is found by sorting *S* on the values of *A* and choosing the split between successive values that maximizes the criterion above. An attribute *A* with discrete values has by default one outcome

for each value, but an option allows the values to be grouped into two or more subsets with one outcome for each subset.
Over fitting is avoided by pruning the initial tree. The pruning algorithm is based on a pessimistic estimate of the error rate associated with a set of *N* cases, *E* of which do not belong to the most frequent class. C4.5 determines the upper limit of the binomial probability when *E* events have been observed in *N* trials, using a user-specified confidence whose default value is 0.25 rather than using the default value of E/N..
Pruning is carried out from the leaves to the root. The estimated error at a leaf with *N* cases and *E* errors is *N* times the pessimistic error rate as above. For a sub tree, C4.5 adds the estimated errors of the branches and compares this to the estimated error if the subtree is replaced by a leaf; if the latter is no higher than the former, the subtree is pruned. Similarly, C4.5 checks the estimated error if the subtree is replaced by one of its branches and when this appears beneficial the tree is modified accordingly. The pruning process is completed in one pass through the tree[5].

2.3. KNN classifier - A drawback of rote classifier which memorizes all the result of classification and classifies data when all attributes of test data matches exactly with at least one instance of train data. In real life scenarios many test records will not be classified because they do not exactly match any of the training records. A more sophisticated approach, *k*-nearest neighbour (*k*NN) classification [6,7], finds a group of *k* objects in the training set that are closest to the test object, and bases the assignment of a label on the dominance of a particular class in this neighbourhood. There exists a set of labelled objects, e.g., a set of stored records, a distance or similarity metric to compute distance between objects, and the value of *k*, the number of nearest neighbours in this classifier. To classify an unlabeled object, the distance of this object to the labelled objects is computed, its *k*-nearest neighbours are identified, and the class labels of these nearest neighbours are then used to determine the class label of the object.

Once the nearest-neighbor list is obtained, the test object is classified based on the majority class of its nearest neighbors:

Majority Voting: $y\_ = \text{argmax}(c) = \sum_{xi,yi \in D} I \quad (c=y_i)$,

where c is a class label, *yi* is the class label for the *i*th nearest neighbors, and *I (·)* is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

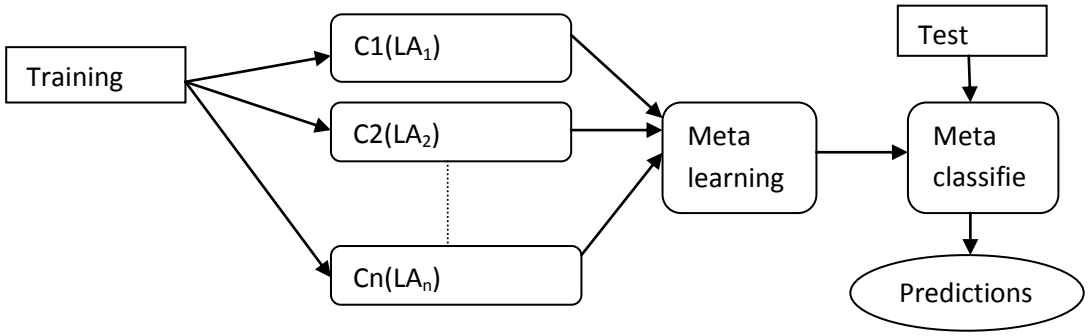
3. META CLASSIFIER AND META LEARNING IN STACKING

In stacking we use multiple algorithms and combine their results. The output from the previous layer is passed as input to the next layer. For eg. The output of a decision tree and be used as input for a neural network. This combining of algorithms helps reduce the problems of an individual algorithm. When a set of classifiers are used as a base and the output predictions are used as input for a meta learning algorithm then stacking can be visualized as in fig 1. where LA represents a learning algorithm.

4. TWO STEP TREE BASED STACK META CLASSIFIER

Although many meta classifiers have been proposed using voting ,boosting SCANN here we propose a two step stack generalization for predicting accurate results. The first step preprocess and selects the features required for classification and the second step uses base classifiers and a meta learning algorithm to induce a meta classifier based on probability distribution of base level classifiers. We have used the combine_function proposed by [8] and created the unify algorithm which basically is a meta learning step.

Figure 1. Learning of meta classifiers



Proposed algorithm- TTS

Two_step_meta_classifier(D, C_1,C_2,…Cn, MC)

{

C1..Cn : Classifiers to be used as base classifiers

MC: Meta classifiers

D : Dataset to work on

1. Choose the C4.5 as meta-classifier and use *Algorithm 2* for preprocessing the data set D
2. Select base classifiers for stacking and run them on training dataset using *unify* function

3. Classify test data using meta classifier using CML obtained in previous step

4. Repeat the process for desired number of runs and output result

}

Algorithm 2(A,D,Train,test,Percent_split)
{
A: meta classifier algorithm to be used //input
D: the dataset with full features //input
Train: Contains training dataset // return argument
Test contains test dataset // return argument

1. First run the classifier on full dataset D, with 10 fold cross validation and calculate BEST
2. FS=null
3. While the expected evaluation criteria not met
   a. Choose the highest information gain attribute a_i
   b. FS= FS $\bigcup$ a_i
   c. Classify the dataset with only features in set FS with wrapper based approach
   d. If classifying criteria >=BEST exit loop
4. Create a new dataset D_{NEW} with selected FS
5. Divide data as training dataset and test dataset with required Percent_split

}

Algorithm unify(L, {$A1$,$A2$, . . .AN }, m, A_{ML})
{$L1$, $L2$, . . . Lm} = stratified partition of L into m different sets)
A_{ML} : Learning algorithm for C_{ML}
L_{ML} = { }

a. for k = 1 to m //no of datasets
b. for j = 1 to N do //No of Learning algorithm
c. Let Cj,k be the classifier obtained by applying Aj to L / Lk
d. Let $CV\ j,k$ be class values predicted by $Cj\ k$ on examples in Lk
e. Let CDj,k be class distributions predicted by Cj,k on examples in Lk
f. MAj,k = meta_ attributes($CV\ j,k$, CDj,k , Lk)
g. endfor
h. $L_{ML} = L_{ML}\ U$ join$^{N}_{j=1}\ MA_{j,k}$
i. endfor
j. Apply A_{ML} to L_{ML} in order to induce the combiner C_{ML}
k. return C_{ML}

end
}

Given the class probability function and class for each classifier it is easy to classify the new dataset. First, the predictions of the base-level classifiers are obtained on the given data set. These include predicted class probability distributions as well as class values themselves. In the meta-level data set *M, the* meta level attributes *C*1 and *C*2 are the class value predictions of two base-level classifiers *C*1 and *C*2 for a given example. The two additional meta-level attributes *Conf* 1 and *Conf* 2 measure the confidence of the predictions of *C*1 and *C*2 for a given example. The highest class probability, predicted by a base-level classifier, is used as a measure of its prediction confidence. For example in table a the result of predictions of a binary class dataset by two classifiers C1 and C2 is given. The tree based stacking classifier will choose result based on the confidence level given by the prediction.

Table a. Classifier C1

Conf1	Class
0.545	1
0.845	1
0.684	0
0.345	1
0.546	0

Table b. Classifier C2

Conf2	Class
0.645	1
0.645	1
0.484	1
0.445	0
0.746	0

Table c. Meta Classifier

Class
1
1
0
0
0

5. EXPERIMENTS AND RESULTS

For our experiment we have chosen three UCI datasets. The program was implemented using Java 1.6 . The software was run on Pentium Dual processor machine with 3GB RAM and clock speed of 2.16 and 2.17 MHz respectively. The selected dataset for test is presented in table d. The performance of meta classifier without using algorithm 2 and run on all attributes is shown in table e. Datasets with minimum selected attributes and positive result are shown in table f.

Table d . Selected UCI datasets

Dataset	Instances	Attributes	No. of classes
Iris	150	5	3
soybean	683	63	19
segment	1500	20	7

Table e . Classifier performance without feature selection training and test set – training set -66% split

Dataset	Naive Bayes	C4.5	KNN	META-CLASSIFIER
Iris	95.53	94.73	95.73	95.33
soybean	91.36	88.74	84.31	90.50
Segment	80.17	96.79	95.25	95.69

Table f . Classifier performance with feature selection training and test set – training set -66% split

Dataset	Naive Bayes	C4.5	KNN	TTS
Iris(2,cfs)	96.08	96.08	96.08	96.08
Soyabean(13,IG)	90.99	94.1	95.94	98.19
Segment(7,Cfs)	83.53	95.88	92.16	96.86

The results show that more number of attributes can be reduced using algorithm 2 and the classification rate is higher using tree based meta classifier with attribute selection. In table f ,the number of attributes selected and the feature selection evaluation criteria used in wrapper approach is shown. Large datasets for information extraction using stacking based methods have been analyzed by[9]. We have shown that class probabilities that can act as confidence levels for prediction to be made as used by [10] .The percent correct of different base classifiers and meta classifier without attribute selection is shown in figure b. The percent correct of different base classifiers and meta classifier with attribute selection is shown in figure c.

Figure.b. Dataset classification using stack and c4.5 as meta classifier without attribute selection

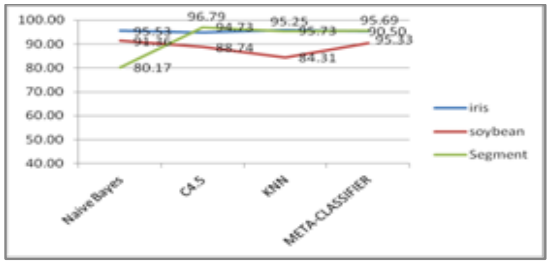
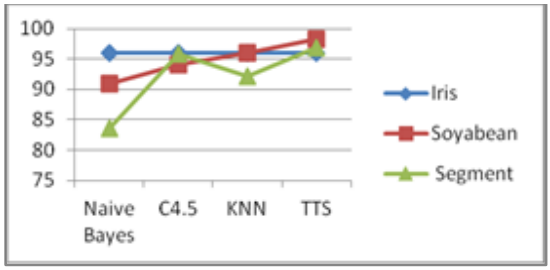
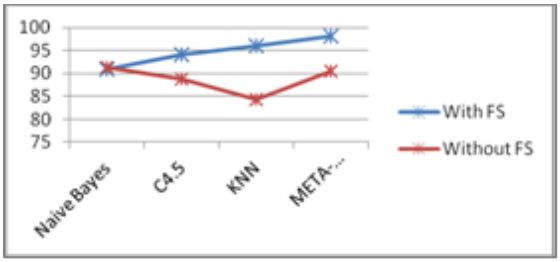


Figure.c. Dataset classification using stack and c4.5 as meta classifier with attribute selection



The results clearly show that that there is a tremendous improvement in the classification of datasets by using the TTS algorithm. The algorithm 2 can be clearly improved in time by using only wrapper based attribute selection method or we can use a genetic algorithm to select features. We plan to continue the work in finding a suitable genetic algorithm for improving the time efficiency of the TTS algorithm. The graph below shows the soybean improvement classification using TTS algorithm with and without attribute selection.

Figure d. Soybean dataset classification using TTS



The soybean dataset has acquired a huge difference in correct percent on test data. Soybean has large number of attributes as well as classes. This shows that this algorithm will work well on datasets having many number of attributes and classes.

6. CONCLUSION

The results show that the feature selection algorithms when used with stacking algorithm with different base classifier can produce more accurate results. Though feature selection with different algorithms may take time, this is compensated by the accuracy of the classification result. In this paper we have proved that the meta classifier should be included as base classifier so that any learning that has to be done and discrepancy encountered can be compensated from other learning algorithms. We plan to use this stacking method in real life business environment for further study.

7. REFERENCES

[1] Brazdil. P., Gama. J. & Henery. R,"Characterizing the applicability of classification algorithms using meta level learning", in European conference on machine learning, ECML-94, pp 83-102.

[2] Bernhard Pfahringer, Hilan Bensusan, Christophe Giraud-Carrier, "Meta-learning by landmarking various learning algorithms".*Proceedings of the Seventeenth International Conference on Machine Learning, ICML'2000*. ISBN 1-55860-707-2, pp. 743–750. June 2000

[3] Wolpert, D.H (1992). "Stacked Generalization". *Neural Networks*, 5(2):241–260.

[4] J. R. Quinlan," Improved use of continuous attributes in c4.5". Journal of Artificial Intelligence Research, 4:77-90, 1996

[5] XindongWu et al "Top 10 algorithms in data mining", Survey paper , Springer-Verlag London Limited 2007, Published online: 4 December 2007.

[6] Fix E, Hodges JL, Jr (1951) Discriminatory analysis, and nonparametric discrimination. USAF School of Aviation Medicine, Randolph Field, Tex., Project 21-49-004, Rept. 4, Contract AF41(128)-31, February1951

[7] Tan P-N, Steinbach M, Kumar V (2006) Introduction to data mining. Pearson Addison-Wesley.

[8] Todorovski.L , Dzeroski.S.,,"Combining Classifiers with Meta Decision Trees", Machine Learning Journal, volume 50, pp 223-249,2003

[9] Sigletos G, Paliouras G, Constantine D spyropoulus " Combining Information Extraction Systems Using Voting and Stacked Generalization", Journal of machine learning research, Volume 6,1751-1782,2005

[10] Ting K.M , Ian H Witten, "Issues in Stack Generalization", Journal of artificial intelligence research, volume 10, pp 271-281,1999