

Performance Analysis of Classifying Unlabeled Data from Multiple Data Sources

M.JEEVAN BABU, K. SUVARNA VANI *

Department of Computer Science and Engineering

V. R. Siddhartha Engineering College, Vijayawada, Andhra Pradesh

ABSTRACT-In many real time applications, Data acquired about the same entity by different sources is generally partially redundant, and partially complementary, since each source has different characteristics and physical interaction mechanisms are different. The information provided by a single source is incomplete resulting in misclassification. Fusion with redundant data can help reduce ambiguity, and fusion with complementary data can provide a more complete description. In both cases, classification[3] results should be better.

In many domains, large amounts of unlabeled data are available in the real world data-mining tasks. However labeled data are often limited and time-consuming to generate, since labeling typically requires human expertise.

Consider the above requirements; we have to classify unlabelled data acquired about same entity by different sources. But existing data mining techniques (supervised learning, unsupervised learning (associate clustering), and co-training)[2] does not give good results with these requirements.

To overcome the difficulties of present data mining techniques, introduce a novel method that predicts the classification[3] of data from multiple sources[1] without class labels in each source which is called learning classification from multiple sources of unlabeled data, or simply cooperative unsupervised learning.

In this project I am going to work on 2 different datasets" classification unlabelled data of multiple data sources[3]"

Keywords: new solutions for multiple data source mining, learning from multiple sources of data, learning classifications from unlabeled data of multiple sources

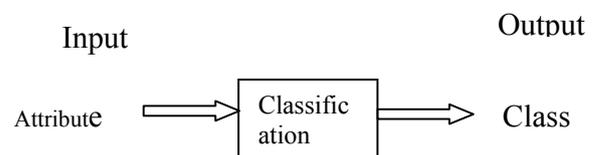
INTRODUCTION:

Classification[3] is a key data mining technique where by database tuples, acting as *training samples*, are analyzed in order to produce a model of the given data. Each tuple is assumed to belong to a predefined class, as determined by one of the attributes, called the *classifying attribute*. Once derived, the classification[3] model can be used to categorize future data samples (unknown data samples), as well as provide a better understanding of the database contents. Classification has numerous applications including credit approval, product marketing, and medical diagnosis.

Classification is the separation or ordering of objects into classes. If the classes are created without looking at the data (non-empirically), the classification is called apriori classification. If however the classes created empirically (by looking data), the classification is called posteriori classification. Generally classification is assumed that the classes have been deemed apriori, classification consists of training the system so that when a new object to one of the existing classes. This approach is called supervised learning.

Some techniques are available posteriori or unsupervised classification in which classes are determined based on the given data. In supervised learning schemes, it is assumed that we have sufficient training data to build an accurate model during the training phase.

Definition: *Classification:* Classification is the task of learning a target function f that maps each attribute set x into one of predictive class label y . the target function is also informally as a classification model.



A classification model is useful for the following purposes:

Descriptive modeling: a classification model serves as explanatory tools distinguish between objects of different classes.

Predictive Modeling: A Classification model can also be used to predict the class label of unknown records. As shown in figure 2, a classification model can be treated as a black box that automatically assign a label when presented with attribute set of unknown record.

Classification techniques are most suitable for predicting or describing data sets with binary or nominal categories. They are less effective for ordinal categories (ex: A person classify as high | medium| low income group).

Decision Tree:

A number of classification techniques (neural networks, fuzzy and or rough set theory, knowledge representation, inductive logic programming, Decision trees) from the statistics and machine learning communities have been proposed. A well-accepted method of classification is the induction of decision trees.

A decision tree is a hierarchical consecutive structure consisting of internal nodes, leaf nodes, and branches. Each internal node represents a decision or test on a data attribute, and each outgoing branch corresponds to a possible outcome of the test. Each leaf node represents a class. In order to classify an unlabeled data sample, the classifier tests the attribute values of the sample against the decision tree. A path is traced from the root to a leaf node, which holds the class predication for that sample. Decision trees can easily be converted into IF-THEN rules and used for decision-making.

The field of application decision tree is quite extensive these days, but all the problems that this instrument helps to solve can be grouped into following 3 classes:

- **Data Description:** Decision tree allow storing information about data in compact form. Instead data we can store the decision tree containing exact description of the objects.
- **Classification:** Decision tree are very good at classifying objects, i.e. distributing them predefined classes. The target variable must have discrete values.
- **Regression:** if the target variable has continuous, decision trees allow determining its dependence upon independent (input) variables. This class, for example, includes the problem of numerical prediction (forecasting values of the target variable).

Today there exists quite a few algorithms implementing decision tree: CART, ID3 (Iterative Dichotomiser 3), C4.5, NewId, ITrule, CHAID and CN2 etc. However most widespread and popular are the following of them:

- **CART (Classification And Regression Tree):** This is the algorithm for constructing binary decision tree—a dichotomic classification model. Each tree node being subset gives only two offspring. As may be seen from the algorithm name, it solves problems of classification and regression.
- **ID3 and C4.5:** ID3 is an algorithm to generate decision tree. This algorithm is based on Occam's razor: it prefers smaller decision trees over larger ones. However, it does not always produce smallest tree, and is therefore a heuristic. C4.5 is an improved version of the ID3, makes use of information theoretical approach. It can deal with continuous attributes.

Building a Decision tree:

The decision tree algorithm is a relatively simple top down greedy algorithm. The aim of the algorithm is to build a tree that has leaves that are homogeneous as possible. The major step of the algorithm to continue to divide leaves those are not homogeneous into leaves that are as homogeneous as possible until no further division is possible.

The Decision Tree Algorithm is given below:

Let the training data be S. If some of the attributes are continuous-valued, they should be discretized. For example: age values may be binned into the following categories (under 18), (18-40), (41-65) and (over 65) and transformed into A, B, C, D or more descriptive labels may be chosen. Once that is done, put all of S in a single tree node.

1. If all instances are in the same class, then stop.
2. Split next node by selecting an attribute A from amongst the independent attributes that best divides or splits the objects in the node into subsets and create a decision tree node
3. Split the node according to values of A
4. Stop if either of following conditions is met, otherwise continue with step 3
 - a. If this partition divides the data into subsets that belongs to single class and no other node needs splitting
 - b. If there are no remaining attributes on which the sample may be further divided.

In the decision tree algorithm, decisions are made locally and the algorithm at no stage tries to find a globally optimum tree. The major step in the decision tree-building algorithm is step 3, where an attribute that best splits the data needs to be selected. To find a split attribute, all the attributes not yet been used to be somehow given a goodness value that shows the discrimination power of the attribute. The attributes then may be ranked according to that value and the highest ranked attribute selected. The discriminatory power may be evaluated in a number of different ways. We will only discuss “Rules based on information theory or entropy “evaluation rule.

Information theory or entropy is based on Claude shanons’s idea that if you uncertainty then you have information and if there is no uncertainty there is no information. Information is defined as $(-p_i \log p_i)$ where p_i is probability of some event. Information of any event that is likely to have several possible outcomes is given by

$$I = \sum_i (-p_i \log p_i);$$

ID3 uses the information theory or entropy for selecting the split attribute. Choose the attribute for which entropy minimum. C4.5 can deal with training sets that have records with unknown attributes values by evaluating the gain or gain ratio. Split Attribute in c4.5 algorithm the attribute, which has maximum gain.

Information gain is defined as follows:

$$\text{Gain}(S, A) = I - \sum_i (t_i / s) I_i \quad i \in \text{values}(A)$$

- I is the information before split
- $\sum_i (t_i / s) I_i$ is the sum of information after the split where I_i is the information of node i , t_i is the number of objects in node i .

Gain ratio, defined as $\text{Gain Ratio}(P|X) = \text{Gain}(P|X) / E(X)$, where $E(X)$ is the entropy of the examples relative only to the attribute X, measures the information gain of feature X relative to the "raw" information of the X distribution.

The divide and conquer algorithm partitions the data until every leaf contains cases of a single class, or until further partitioning is impossible because two cases have the same values for each attribute but belong to different classes. Consequently, if there are no conflicting cases, the decision tree will correctly classify all training cases. This so-called over fitting is generally thought to lead to a loss of predictive accuracy in most applications (Quinlan 1986).

Pruning:

An induced tree may overfit the training data, that tree has too many branches; some may reflect anomalies due to noise or outliers ; Resulting the Poor accuracy for unseen samples. To avoid the over fitting we use prune the tree.

The decision tree produced by ID3 is a good description of the training data. However, the tree is almost always too specific for predicting future cases. Therefore, it is often useful to optimize the tree by pruning back leaves or entire branches. There are numerous strategies for pruning. Generally, pruning involves either additional testing data or the use of statistical techniques for generalizing the tests.

The idea is to use a heuristic test for "predicting" the future error rates of a given branch in the tree. If a given branch has a higher error rate than a simple leaf would, the branch is

replaced with a leaf. By applying this heuristic from the bottom to the top of the tree, you prune back the tree for better future prediction. The error prediction is based on a binomial distribution.

Calculating the binomial probability is easy when the number of errors is zero. The error rate turns out to be a simple exponential calculation. However, when the number of errors is not zero, computing the error rate becomes much more difficult. Instead of trying to solve the error rate based on the binomial formula, I use the normal distribution for approximating the binomial error rate. This normal approximation turns out to be calculable, although for very small probabilities or very low number of cases it is not a good representation of the binomial equation. As a heuristic, though, empirical tests indicate that it is an adequate approximation.

Estimating predictive accuracy of classification model:

The accuracy of a classification is the ability of the method to correctly determine the class of a randomly selected data instance. Let us assume that the test data has a total of t objects. When testing a model we find that c of t objects are correctly classified. The error rate then may be defined as

$$\text{Error rate} = (t-c)/t$$

$$\text{Predictive accuracy} = c/t$$

We have number of methods for estimating the accuracy a model like holdout method, random sub-sampling method, k-fold cross-validation method, leave-one-out method, bootstrap method etc. In this project, I use k-fold cross-validation.

K-fold Cross Validation: In k-fold cross validation, the available data is randomly division into k disjoint subsets of approximately equal size. One of the subsets is then used as the test set and remaining $k-1$ sets are used for building the classifier. The test is then used to estimate the accuracy. This is repeated k times so that each subset is used as a test subset once. The accuracy estimate is then the mean of the estimates for each of the classifiers. Cross-validation has been tested extensively and has been found to generally work well when sufficient data is available. A value of 10 for k has been found to be adequate and accurate.

PROBLEM SPECIFICATION:

In many real time applications, Data acquired about the same entity by different sources is generally partially redundant, and partially complementary, since each source has different characteristics and physical interaction mechanisms are different. The information provided by a single source is incomplete resulting in misclassification. Fusion with redundant data can help reduce ambiguity, and fusion with complementary data can provide a more complete description. In both cases, classification results should be better.

In many domains, large amounts of unlabeled data are available in the real world data-mining tasks. However labeled data are often limited and time-consuming to generate, since labeling typically requires human expertise.

Consider the above requirements; we have to classify unlabelled data acquired about same entity by different sources. But existing data mining techniques (supervised

learning, unsupervised learning (associate clustering), and co-training) does not give good results with these requirements.

- Supervised learning from one source of data with class labels, our problem is supervised learning from multiple sources without class labels.
- Associative Clustering[5] (AC) takes two related datasets without class labels and computes clusters from each dataset. Then a contingency table is built to draw the similarity and dependency between the two sets of clusters. However, AC is still a clustering[5] or unsupervised learning method, and as we will show later, due to the lack of tight alignment between the two sources of data, the results are not reliable
- Co training utilizes a small set of labeled examples to boost the classification performance on a large set of unlabeled examples (such as texts). Similar to our methods, multiple “views” (sources) of the data are given. However, co-training assumes that small portions of examples in each view are labeled, while our method works without labels of any example in any data source.

Thus existing data mining techniques can't get better accuracy or can't work on our problem. To overcome these difficulties, introduce a novel method that predicts the classification of data from multiple sources without class labels in each source which is called learning classification from multiple sources(**CMS**) of unlabeled data, or simply cooperative unsupervised learning.

IMPLEMENTATION CMS:

Classification of unlabeled data from multiple data sources give a model that are in the form of tree is used in number of application domains. Data mining categorical problems can be solved either classification or clustering[Classification problems require labels for training data but present problem did not have label data. Clustering problems will not achieve tight arrangement between multiple data sources. If we apply the present problem to clustering then we got either too many clusters or too small clusters formed.

Thus a new class of algorithm called classification from multiple data sources introduced which gives exact no of classes as in original dataset with better accuracy or give sub-classification, which forms clusters with less impurity in each group. In this algorithm, has 2 steps first one is partitioning tree construction for using all data sources attributes selected attributes for splitting data at internal node test which gives uni-class clusters. In each partition tree construction, the node is split based on only data source attributes. Second one is merging the partition tree leafs to be merged to identifying which clusters are related.

Partition Tree Construction: A *partition tree* is similar to a decision tree (Quinlan, 1993) since it partitions examples into subsets by attribute values. The major difference is that partition trees do not have class labels in leaves.

The partition tree construction is follows:

- Initially read data from data sources and store in object, construct root node of partition tree for all data sources.
- Construction of partition tree using divide and conquer greedy algorithm, decomposition of node continuous while

tree node attribute relevancy criteria satisfies some threshold condition

- Decomposition of tree node by selected one attribute values, selected attribute based on Maximum attribute relevancy criteria (ARC) value.
- ARC of an attribute evaluate by 2 calculations
- Calculate probability of most frequent value of this attribute say 1
- Decision tree constructed using present attribute is class attribute. Calculate average predictive accuracy of 10-fold cross-validated decision tree. Say 2
- $2 - 1$ value gives the ARC of this attribute

In Partition tree construction, we use a threshold on ARC to either decompose the node or leave it as leaf. This threshold is in between 0 and 1, which is based on randomness of the dataset.

Pseudo code for partition tree algorithm for a modality:

Partition (*set*) For each A_i Do
 Calculate $p(A_i)$
 Use 10-fold cross-validation of C4.5 to estimate $p(A_i | A_i', B)$
 $ARC(A_i) = p(A_i | A_i', B) - p(A_i)$
 If there exists no A_i such that $ARC(A_i) > \alpha$
 Then
 Label all instances of set as a single class
 Else
 $A_j = \max A_i$
 Split *set* into subsets according to values of A_j
 Apply **partition** to subsets recursively

Table 1: Pseudo code for partition tree

Merging:

On completion partition tree algorithm, partition tree was build for each data source attribute. In each partition tree we got uni-class clusters at leafs. To identify which all leafs are formed as same category we implement merging algorithm here. If one assigns distinctive class labels to leaves in the partition tree in either source, such labeling in the two sources may not be consistent. Since the training examples in the different clusters in one source can appear in the same cluster in the other modality, we can “merge” these different clusters to be the same class.

In real world datasets, noise may exist. Therefore, even if L_{Ai} and L_{Bj} intersect, we cannot simply conclude L_{Ai} and L_{Bj} to be labeled by the same class. The intersection must be large enough to warrant such merging. A threshold is thus set in CMS. If

$$|L_{Ai} \cap L_{Bj}| / \min(|L_{Ai}|, |L_{Bj}|) > \beta$$

- Where β is a small constant between 0 and 1.

Then the intersection is regarded as trustable, and L_{Ai} and L_{Bj} are labeled as the same class. As C4.5 is used to estimate $p(A_i | A_i', B)$ in CMS, the time complexity of this probability estimation step is $O(ea)$ where e is the number of examples and a is the total number of attributes in both modalities. In the merge algorithm, all the merge iterations depending on the

threshold and leafs in 2 partition trees of different data sources, will reduce the number of classes in each modality, and therefore, the iteration will be at most e times for e examples. Thus, the time complexity of the merge algorithm is $O(e^2)$. As usually $e > a$, the time complexity of CMS is $O(e^2)$.

OBJECTIVE:

The objective of our project is classifying unlabeled data from multiple data sources.

In this project, we present a new method, called CMS (Classification from Multiple Sources), that reconstructs class labels from unlabeled data of two sources, where the two sources are aligned through a common index. CMS consists of two major steps. In the first step, it uses a partition algorithm

WORKING WITH CMS:

We conducted experiments on CMS using both artificial and real world datasets. The artificial dataset experiments are designed to evaluate how well the method works under various controlled situations (such as incomplete or noisy datasets). This allows us to study, in a great depth, the behavior and source of power of the algorithm, and to compare it with COBWEB and AUTOCLASS. CMS is then tested on several real world datasets with unknown properties.

I worked on 2 different problems on “classification of multiple data sources with unlabeled data”. All are from the UC Irvine Machine Learning Repository.

Real world datasets[9]:

-  Voting dataset
-  Mushroom dataset

The mushroom and voting datasets, examples are described by only one set of attributes. To overcome this problem, we first partition the whole set of attributes into two disjoint subsets, and use the two subsets as the descriptions for the two modalities. To insure that the attributes in each set are sufficient to decide the classification

Mushroom dataset. The first dataset tested is the mushroom dataset from the UCI Repository. After removing examples with unknown attribute values, we have a total of 5,644 examples and each is described by 22 symbolic attributes. The original dataset has two classes: 3,488 examples with class Edible (E) and 2,156 examples with class Poisonous (P). As discussed, we need to partition the 22 attributes (called A_1 to A_{22} here corresponding to the attributes in the name file with the same order) into two disjoint subsets (two modalities) so that either set produces high predictive accuracy in predicting the class. After several experiments, we find that if the first subset contains $A_2, A_9, A_{11}, A_{17}, A_{18},$ and A_{20} , and the second subset contains the rest of the attributes, it meets our requirement.

If we set the α value at 30% and β at 15% or 20%, the partition trees constructed from both Modalities are give quite good sub classification.

Relation the original labels and labeling generated by CMS (c1to C4) on mushroom dataset at alpha=0.30 and beta=0.15 show s the following table:

Table 2 :

	C1	C2	C3	C4
P	1584	328	192	52
E	0	864	1824	800

Relation the original labels and labeling generated by CMS (C1to C6) on mushroom dataset at alpha=0.30 and beta=0.15 show s the following table:

Table 3:

	C1	C2	C3	C4	C5	C6
P	1584	328	192	52	8	44
E	0	864	1776	800	48	752

While analyzing the above results are the good sub classification and good accuracy maintained an Impurity over leafs is: 10.1% at $\alpha=0.30$ and $\beta=0.15$.

Voting dataset.

The second real-world dataset is the voting dataset from the UCI database. This dataset has 16 discrete attributes, each of which has 3 possible values. Unlike the mushroom dataset, the voting dataset has only 435 examples (168 are republican and 267 are democrat), and it is more difficult to split the attributes into two subsets while retaining high predictive accuracy. After several experiments, we find a reasonable partition which contains attributes “physician fee freeze”, “religious groups in schools”, “mix missile”, “education spending”, “water project cost sharing”, and “duty free exports” in one subset, and the rest of the attributes in the other subset for the two modalities respectively.

Table 4 :

	$\alpha=0.04$	$\alpha=0.08$	$\alpha=0.12$	$\alpha=0.16$	$\alpha=0.30$
$\beta=0.05$	s1-170(0.0%) s2- 75(0.0%) m-40(0.0%)	s1-130(0.0%) s2- 60(0.0%) m-25(0.0%)	s1-110(0.0%) s2- 40(0.0%) m-10(0.0%)	s1-100(0.0%) s2- 16(0.0%) m-6(0.0%)	s1-21(33%) s2- 12(33%) m-1(40%)
$\beta=0.10$	s1-170(0.0%) s2- 75(0.0%) m-40(0.0%)	s1-130(0.0%) s2- 60(0.0%) m-25(0.0%)	s1-110(0.0%) s2- 40(0.0%) m-10(0.0%)	s1-100(0.0%) s2- 16(0.0%) m-7(0.0%)	s1-21(33%) s2- 12(33%) m-1(40%)
$\beta=0.15$	s1-170(0.0%) s2- 75(0.0%) m-40(0.0%)	s1-130(0.0%) s2- 60(0.0%) m-25(0.0%)	s1-110(0.0%) s2- 40(0.0%) m-10(0.0%)	s1-100(0.0%) s2- 16(0.0%) m-7(0.0%)	s1-21(33%) s2- 12(33%) m-2(16.5%)
$\beta=0.20$	s1-170(0.0%) s2- 75(0.0%) m-40(0.0%)	s1-130(0.0%) s2- 60(0.0%) m-25(0.0%)	s1-110(0.0%) s2- 40(0.0%) m-10(0.0%)	s1-100(0.0%) s2- 16(0.0%) m-7(0.0%)	s1-21(33%) s2- 12(33%) m-2(18%)

Table: Results of applying CMS with various alpha and beta values to the voting dataset. In each entry, the first value is the number of leaves in the partition tree from the first modality; the percent in parenthesis is the impurity rate comparing to underlying classes. The second value is the number of leaves from the second modality. The third line is number of classes after applying the merge algorithm. The percent in parenthesis is the impurity rate of the final results.

While analyzing the table 4 results are the good sub classification and good accuracy maintained an Impurity over leafs is: 16.5% at $\alpha=0.30$ and $\beta=0.15$.

In observations, at $\alpha = 0.3$ and $\beta=0.20$ most of real world examples satisfied with number of labels. At $\alpha = 0.04$, $\alpha = 0.08$, $\alpha = 0.12$ give not good results because the voting dataset has 435 records, which have many missed values and small set. This is may reasoned as randomness of dataset; i.e, the dataset is not properly have the class distribution as in whole data.

HYPOTHESIS:

Instead of supervised learning from one source of data with class labels, our problem is supervised learning from multiple sources without class labels. More specifically, given two sets of attributes, each describing the same training examples with a common index such as the customer ID, the task is to learn the appropriate class labels, as well as the hypothesis of the classification (such as decision trees) that predicts the class labels. We call this type of learning task *learning classification from multiple sources of unlabeled data*, or simply *cooperative unsupervised learning*. The two attribute sets are cooperative since they describe the same examples. Cooperative unsupervised learning is an interesting and important problem—unlike supervised learning, the class labeling is not available; yet unlike unsupervised learning, multiple data sources are given and the classification to be learned must be consistent between them.

CONCLUSION & FUTURE WORK:

Conclusion:

In many real-world applications there are often no explicit class labels; instead, we are given unsupervised data originating from different sources. We design and implement a new learning algorithm CMS that can discover classification from unsupervised data of multiple sources. Extensive experiments on real-world datasets (voting data set and mushroom data set) show that CMS is robust and effective in discovering class labels accurately.

Future Work:

- ✚ In our future work, we plan to improve CMS with a variety of learning algorithms (in addition to C4.5).
- ✚ The stopping criterion for growing the partition tree, currently using α , can be improved.
- ✚ The current merging criterion with β can also be made more sophisticated. We also plan to apply CMS to real-world problems with very large datasets from multiple sources.
- ✚ Improve decision about α , β values which are depends on the randomness and noisiness of data set.

REFERENCES:

- [1]. "Discovering Classification from Data of Multiple Sources", CHARLES X. LING
- [2]. Blum, A. and Mitchell, T. 1998. Combining labeled and unlabeled data with co-training. In Proceedings of the Eleventh Annual Conference on Computational Learning Theory,
- [3]. Tan , Steinbach, Kumar "Introduction to Data Mining"
- [4] de Sa, V. 1994a. Learning classification with unlabeled data. In Advances in Neural Information Processing Systems, J. Cowan, G. Tesauro, and J. Alspector (Eds.), vol. 6
- [5] Fisher, D. 1987. Knowledge acquisition via incremental conceptual clustering. Machine Learning,
- [6]. David Hand, "Principles of Data Mining" Mellon University.
- [7]. Pankaj Jalote, "An Integrated Approach to Software Engineering", Narosa Publishing House, 1998.
- [8]. Reich, Y. and Fennes, S. 1992. Inductive learning of synthesis knowledge. International Journal of Expert Systems: Research and Applications,
- [9]. UCI Repository data sets from <http://www.ics.uci.edu/~mllearn/MLRepository.html>