# Supervised Learning with Unsupervised Output Separation

Nathalie Japkowicz
School of Information Technology and Engineering
University of Ottawa
150 Louis Pasteur, P.O. Box 450 Stn. A
Ottawa, Ontario, Canada K1N 6N5

ABSTRACT
In supervised learning approaches, the output labels are imposed by the knowledge engineer who prepared the data. While knowing the labels of a data set is quite useful, in cases where data points belonging to very different data distributions are agglomerated in the same class, a learning algorithm can have difficulties modeling these classes accurately. In such cases, it should be useful to separate the main classes into a number of more homogeneous subclasses. This paper assumes that the above problem is quite common and describes a simple combination method that attempts to fix it. It then tests the approach on 5 domains taken from the UCI Repository. The results show that in three out of five cases, the approach has a positive effect, in one case, it breaks even and in the fifth case, it degrades the previously established performance.

KEY WORDS
Machine Learning, Decision Trees, Combination of Classifiers, Clustering

## 1 Introduction

Supervised learning is a lot easier than unsupervised learning. Indeed, in supervised learning a function must be induced that maps training vectors to a subset of given labels. In unsupervised learning, no labels are given but a function has to be learned which first, creates a set of coherent labels and second, maps training vectors to these labels. Though easier, supervised learning lacks the flexibility offered by unsupervised learning. Indeed, what if the labels accompanying a training data set are imperfect and overly constraining? What if, for example, they lump two subcategories together which, though similar in some respect, are completely different in others? In such a case, it is conceivable that a supervised learning system would have difficulties assigning these different examples to the same class: too much variability would be present within a supposedly homogeneous class.

The purpose of this paper is to investigate this issue. In particular, we place ourselves midway between supervised and unsupervised learning by using an unsupervised method of re-labeling already labeled data sets in a way that respects both the initially given labels and the data distribution that was not necessarily taken into account by the knowledge enginner when s/he assigned the initial labels. Because, however, the distribution of the data is hidden, the "most appropriate" re-labeling is difficult to derive. We, thus, decided to generate a number of different re-labeling of the same data set. A classifier is then run on these various versions of the same data set and their results are combined using a voting technique. Several variations of this general approach are considered and the results they obtained on five data sets are reported.

The remainder of the paper is divided into four sections. Section 2 discusses previous work in the areas in which our approach lies. Section 3 describes the details of our approach, Section 4 describes the experiments we conducted and their results, Section 5 discusses our results and Section 6 concludes the paper and discusses future work.

## 2 Previous Work

The approach described in this paper is at the cross-roads of two different sub-disciplines of the field of Machine Learning. The first sub-discipline is that of combining supervised and unsupervised learning to improve classification accuracy while the second is that of combining different supervised models in order to, also, improve accuracy. In this section, we review the sate-of-the-art in both subdisciplines and locate our method within them.

### 2.1 Combination of Supervised and Unsupervised Learning

The combination of supervised and unsupervised methods is a recent approach used in the area of Machine Learning. Such a combination has different goals. On the one hand, it can be attempted in order to improve the classification accuracy of a supervised classifier by biasing that classifier using information coming from the unsupervised process (see, for example, [Stainvas99]). On the other hand, it can be used as a way to integrate large amounts of unlabeled data in the supervised learning process (see, for example, [Blum & Mitchell, 98]). In both cases, the process can be internal or external, that is, the supervised and unsupervised steps can be taken simultaneously or they can be taken independently.

The method described in this paper combines super-

vised and unsupervised learning as a way to *bias* the information fed to the supervised classifier. It is not concerned by the use of unlabeled data to enhance classification accuracy. As well, it can be described as an *external* rather than an internal method since the unsupervised step is first performed followed by the supervised one.

## 2.2 Combinations of Supervised Learning Models

As discussed by [Dietterich, 97], combining supervised classifiers is an important current research direction in Machine Learning. The approaches attempted to combine various classifiers use the idea that the combined classifiers should disagree with one another. In other words, in order for improvement to be possible, there should be some variance in the combined methods. [Dietterich, 97] lists four general ways of creating variance among a same classifier: sub-sampling the training data, manipulating the input features, manipulating the output targets, and injecting randomness. Furthermore, he lists three general combination methods: unweighted voting, weighted voting and using a gating function.

The method described in this paper creates variance by manipulating the *output* targets and uses both *unweighted* and *weighted voting* for combining the different decisions. In addition, as suggested by [Shapire, 97], AdaBoost (a sub-sampling, weighted voting combination method) is used in some experiments in conjunction with the output manipulation approach.

While manipulating the output has previously been used in the combination framework (see [Dietterich & Bakiri, 95]) the purpose for which that approach was used is quite different from ours. [Dietterich & Bakiri, 95] devised their method to improve the classification accuracy in multi-class problems by reducing them to binary class problems. We, on the other hand, expand binary classification problems[1] into multi-class classification ones with the hope of increasing their classification accuracy.[2]

# 3 The Approach

The approach we propose can be divided into three steps. In a first step, we separate each class into a number of subclasses, using an unsupervised learning technique, and we re-label each training example as a function of these new subclasses. In a second step, supervised learning is applied to various versions of these new problems. Finally, the results obtained on each version of the problem are combined in a decisive vote. Each step of this process are described in more detail below.

## 3.1 Step 1: Output Separation

In this part of our approach, a simple unsupervised learning system, k-means, is used. k-means requires the number of clusters to be specified in advance. However, because the appropriate number of clusters required to best characterize the two classes is not clear, we experimented with two types of subdivisions: the first one subdivided the positive and the negative classes into 5 subclusters each while the second subdivided each class into 3 subclusters each. The clustering was performed separately on each class, since, while we wanted to separate the classes into subclasses, we did not want any cross-clustering to occur (i.e., we wanted to avoid creating sub-clusters containing both positive and negative examples). In addition, because the optimal clustering is not known, we introduced some variability by running k-means 5 times within each experiment and, subsequently, combining the results obtained on each variation (see Step 3).

## 3.2 Step 2: Supervised Multi-class Learning

As just described, the first step of our method created 5 different multi-class learning problems per experiment. Each of these problems contains 10 (sub)classes (5 positive and 5 negative ones) in the first version of our experiments and 6 (sub)classes (3 positive and 3 negative ones) in the second.

The purpose of the second step of our method is to apply a supervised classifier to each of these new multi-class problems with the hope that learning how to classify subclasses of the positive or the negative classes is an easier task than classifying an example directly as positive or negative. C5.0 and C5.0-Boosted were used in this part of our work since both approaches apply to multi-class problems.

An important aspect of our experiments to mention at this point is the fact that we are not concerned about one example being misclassified as belonging to a subclass to which it does not belong, as long as its real subclass and its assigned subclass both belong to the same higher class (i.e., they are both positive or negative subclasses). For example, assume that the positive class was subdivided into subclasses 1, 2 and 3 while the negative class was subdivided into subclasses 11, 12 and 13. Misclassifying a class 3 example for a class 1 or 2 example or misclassifying a class 11 example for a class 12 or 13 examples are not errors that should be accounted for. On the other hand, misclassifying a class 3 example for a class 11, 12 or 13 example or a class 12 example for a class 1, 2 or 3 example should be accounted for since the positive/negative boundary, this time, has been crossed. This emphasizes the fact that our output separation method was designed to help classification but not to impose yet another strict labeling of the data.

[1] But we could also work with multi-class problems.

[2] Actually, once our multi-class problems are generated, nothing prevents us from using [Dietterich & Bakiri, 95]'s approach to improve our handling of these new problems. This was not attempted in the experiments reported in this paper, but we believe that it could yield some improvement, especially when the number of created subclasses grows significantly.

### 3.3 Step 3: Voting

In the last step of our approach, the results obtained for each example on the five different versions of the initial problem are combined in a weighted or a non-weighted fashion, using the certainty factor output by C5.0 or C5.0-Boosted as the weights in the first one of these cases. As mentioned previously, classification errors that do not cross class boundaries are not taken into consideration: all we are interested in is whether the subclass assigned to an example is positive or negative. The results of these votes constitutes the final decision on each example.

In more detail, in the weighted case, the decision for a given example E is calculated by applying the following formula:

$$\sum_{i=version1}^{version5} certainty_i(E)$$

where the function $certainty_i$ returns the probability value returned by C5.0 or C5.0-Boosted on version *i* of the problem for example E multiplied by -1 if E was classified as negative in version *i* and by 1 if it was classified as positive. The overall classification of E is positive if the above formula yields a positive number and negative, otherwise.

In the non-weighted case, the decision for E is calculated by the following formula:

$$\sum_{i=version1}^{version5} class_i(E)$$

where $class_i(E)$ is +1 if E is classified as positive in version *i* of the problem and -1, otherwise. Again, the overall classification of E is positive if the above formula yields a positive number and negative, otherwise.

## 4 Experiments

As discussed in the previous section, several types of combinations were considered:

- Non-boosted and Non-Weighted (NB-NW)
- Non-Boosted and Weighted (NB-W)
- Boosted and Non-Weighted (B-NW)
- Boosted and Weighted (B-W)

These four schemes were evaluated on five different data sets all obtained from the UCI Repository for Machine Learning: Haberman, Ionosphere, Pima, Sonar, Wisconsin Breast Cancer Diagnostic (WBCD). These data sets were chosen because they are all the UCI data sets that 1) contain only continuous attributes (necessary condition for the k-means process); 2) contain no missing attributes (again, a requirement of the k-means process); and 3) are binary classification processes.[3]

[3] As mentioned previously, it would have been possible to use data sets containing more than two classes, but we decided to restrict this study to two-class problems so as to limit the growth of class numbers once the outputs get subdivided. A more detailed study could also include data sets defined over more than two classes.

The experiments were conducted using 10-fold cross-validation and the folds were paired from one experiment setting to another (for example, fold 3 for a given data set contains the same training and testing examples in all the settings of our experiments).

The results for the four schemes are reported in terms of accuracy in columns 4-7 of table 1: In more detail, column 4 reports the results obtained in the non-weighted non-boosted case; column 5 reports the results obtained in the weighted non-boosted case; column 6 reports the results obtained in the non-weighted boosted case; and column 7 reports the results obtained in the weighted boosted case. In addition, columns 2 and 3 report the results obtained by C5.0 and C5.0-Boosted, respectively and column 8 reports the percentage difference in accuracy observed between the best of columns 2 and 3 (that do not use our scheme) on the one hand and, the best of columns 4-7 (that do use our scheme) on the other hand. In every case, the best result is indicated in the table in boldface characters and a positive value in column 8 indicates a better performance by some instance of our scheme while a negative value indicates a better performance by regular or boosted C5.0. Finally, column 1 simply lists the name of the data set tested along with the subdivision used (5 positive and 5 negative subclusters or 3 positive and 3 negative ones). For each data set, the first line corresponds to the experiments obtained with the 5-5 subdivision while the second corresponds to those obtained with the 3-3 subdivision. Since the results reported in columns 2 and 3 are not affected by these subdivisions, they are reported only once, on the first line.

## 5 Discussion

The results reported in Table 1 suggest that the combination scheme described in this paper can be quite useful. Indeed, it yields over 10% increase (and in one case, over 25% increase) over C5.0-Boosted (the better of the two standard classifiers—C5.0 and C5.0-Boosted—for all data sets) in three out of 5 cases (Pima, Sonar, WDBC). In another case, it breaks about even (Haberman) and in the last case, it deteriorates the C5.0-boosted results by about 16% (Ionosphere). Though quite encouraging, these results, however, could be shown to be significant at the s=.05 significance level only in the case of the Pima data set. Significance was established at significance level s=.1 in the case of the Sonar data set while in all other cases, no reasonable significance level could be established.[4] This is caused by the high degree of variance displayed at the various folds of the cross-validation experiments probably caused by the fact that the data sets did not contain enough data points, especially in light of the large number of subclasses generated by the unsupervised process.

All these observations lead us to believe that while the approach we described has some merit, it would perform

[4] A paired t-test for one sided intervals was applied to determine the significance of our results.

better with larger data sets and/or in conjunction with a multi-class stabilizing method such as [Dietterich & Bakiri, 95]'s output correcting codes approach. Another way to improve the results would be to determine carefully how many subclusters constitute an optimal subdivision of each class. For the time being, in particular, we subdivided the positive and the negative classes into the same number of subclusters. However, it is quite likely that a different number of subclusters for each class would be more appropriate. If enough data are available, optimal numbers of subclusters can be determined by cross-validation experiments. Alternatively, unsupervised learning methods that determine on their own the optimal number of subclusters into which a data set should be partitioned could also be used.

## 6 Conclusion and Future Work

This paper presented a new classification method that combines unsupervised and supervised learning in the hope of improving the performance (in terms of classification accuracy) of the supervised classifier. The paper is based on the assumption that class labels assigned by knowledge engineers do not always give rise to homogeneous classes and that this constitutes a problem for classifiers that do expect homogeneous classes. The problem is tackled by using an unsupervised learning technique to subdivide each class into a series of more homogeneous subclasses and applying a multi-class classifier on these new problems. Because the unsupervised process can lead to variations, this technique is implemented within a combination framework.

The preliminary results we obtained show that, in three out of five cases, the method is quite helpful, but that in two others, it either does not help or it hurts the classification performance.

There are several areas of future work that could be considered. First, it would be useful, as mentioned previously, to create more flexible partitions, not imposing, for example, the same number of subclusters onto each class. A second line of future research could take into account the fact that multi-class classification is not, generally, as accurate as binary class classification and use, for example, [Dietterich & Bakiri, 95]'s method to deal with this problem. In a third line of enquiry, the method we proposed could be included in a wrapper system that would allow the unsupervised learning system to conform implicitly to the particular constraints of the supervised classifier. Finally, it would be useful to contrast the proposed approach to research on radial basis functions (RBFs) that present some similarities. In particular, like our approach, RBFs use an unsupervised and a supervised step. However, our approach is a generalization of the RBF method since it can bestow more flexibility onto the process by allowing the use of any unsupervised or supervised classification approach (in RBFs, the unsupervised approach is Gaussian, and the supervised one is linear). Furthermore, our process allows for combinations of different subclustering of the same data set whereas RBFs do not.

## Acknowledgements

This research was funded by a grant from the Natural Science and Engineering Research Council of Canada. We thank Chris Drummond for useful cooments about this research.

## References

[Blum & Mitchell, 98] Blum, A., Mitchell, T. (1998) Combining Labeled and Unlabeled Data with Co-Training, *COLT: Proceedings of the Workshop on Computational Learning Theory*.

[Dietterich & Bakiri, 95] Dietterich, T. G., Bakiri, G. (1995) Solving Multi-class Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research* 2: 263-286.

[Dietterich, 97] Dietterich, T. G., (1997). Machine Learning Research: Four Current Directions *AI Magazine*. 18 (4), 97-136.

[Domingos, 99] Domingos, Pedro (1999): Metacost: A general method for making classifiers cost sensitive, *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, 155–164.

[Estabrooks, 00] Estabrooks, A. (2000): *A Combination Scheme for Inductive Learning from Imbalanced Data Sets*, MCS Thesis, Faculty of Computer Science, Dalhousie University.

[Stainvas, 99] Stainvas, I. (1999): Blurred Face Recognition via a Hybrid Network Architecture, *Neural Computation in Science and Technology*,

[Shapire, 97] Shapire, (1997): Using output codes to boost multi-class learning problems. *Tech Report AT & T Research*

| Data Set | C5.0-NB | C5.0-B | NW-NB | W-NB | NW-B | W-B | Improv. |
|---|---|---|---|---|---|---|---|
| haberman-5-5 | 28.82 | 27.20 | 29.51 | 28.87 | 28.50 | 28.50 | |
| haberman-3-3 | | | **26.83** | 27.16 | 28.40 | 28.72 | + 1.36% |
| ionosphere-5-5 | 8.06 | **5.19** | 8.33 | 9.17 | 6.94 | 6.94 | |
| ionosphere-3-3 | | | 9.17 | 8.33 | 6.02 | 6.02 | - 15.99% |
| pima-5-5 | 27.10 | 25.80 | 27.35 | 25.66 | 27.20 | 27.47 | |
| pima-3-3 | | | 25.91 | 25.52 | **23.17** | **23.17** | + 10.19% |
| sonar-5-5 | 23.63 | 16.87 | 18.34 | 18.77 | 13.58 | **12.63** | |
| sonar-3-3 | | | 17.77 | 15.82 | 15.01 | 15.54 | + 25.13% |
| wdbc-5-5 | 6.50 | 4.21 | 5.27 | 5.27 | 4.39 | 4.39 | |
| wdbc-3-3 | | | 4.93 | 4.93 | **3.70** | **3.70** | + 12.11% |

Table 1. Unsupervised/Supervised Combination Scheme (boosted or not and weighted or not) versus C5.0 (boosted or not)