A Genetic Algorithm Approach for Semi-Supervised Clustering

Ayhan Demiriz
E-Business Department, Verizon Inc.,
919 Hidden Ridge, Irving, TX 75038
E-mail:ayhan.demiriz@verizon.com

Kristin P. Bennett*
Dept. of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180
E-mail: bennek@rpi.edu

Mark J. Embrechts
Dept. of Decision Sciences and Engineering Systems
Rensselaer Polytechnic Institute
Troy, NY 12180
E-mail: embrem@rpi.edu

January 27, 2002

Abstract

A novel semi-supervised clustering algorithm is proposed that synergizes the benefits of supervised and unsupervised learning methods. Data are clustered using an unsupervised learning technique biased toward producing clusters as pure as possible in terms of class distribution. These clusters can then be used to predict the class of future points. For example in database marketing, this technique can be used to identify and characterize segments of the customer population likely to respond to a specific promotion. One key additional benefit of this approach is that it allows unlabeled data with unknown class to be used to improve classification accuracy. The objective function of a traditional clustering technique, cluster dispersion in the K-means algorithm, is modified to minimize both the within cluster variance of the input attributes and a measure of cluster impurity based on the class labels. Minimizing the within cluster variance of the examples is a form of capacity control to prevent overfitting. For the the output labels, impurity measures such as the Gini index can readily be applied to this problem. In this work, a genetic algorithm is proposed to optimize such an objective function to produce clusters. Non-empty clusters are labeled with the majority class. Experimental results show that using class information often improves the generalization ability compared to unsupervised methods based only on the input attributes. Benchmark studies also indicate that the method performs very well even when few training examples are available. Training using information from unlabeled data can improve classification accuracy on that data as well.

Keywords: Semi-supervised learning, Transduction, Genetic Algorithms, Clustering, Classification.

1 Introduction

In this work, a novel approach to solve classification problems that combines supervised and unsupervised learning techniques is examined. In supervised learning, it is assumed that we are given a set of labeled training points, and the task is to construct some function that will correctly predict the labels of future points. In unsupervised learning such as clustering, the task is to segment unlabeled training data into clusters that reflect some meaningful structure in the data. For the Semi-Supervised Learning Problem (SSLP), it is assumed that we are given both labeled and unlabeled points. The task is then to predict the labels of the unlabeled points using all the available labeled data, as well as unlabeled data. One would

*http://www.math.rpi.edu/∼bennek

expect a better generalization ability of the resulting classifier due to the better understanding of the input distribution resulting from using all the available data. The semi-supervised learning problem can be used to perform transductive inference [21]. In transduction, we are given a set of training data and a set of test data, and the learning task is to predict the labels of the specific test data only. The classification function depends both on the labeled training data and the unlabeled test data. Only the labels of the training data are known. The labels of the test data are never used during training. Different test data will produce different classification functions. The intuition is that transduction is a simpler problem because we are trying to construct a function that is valid only at specific points, versus induction where the goal is to construct a function valid at all future test points.

Transduction was proposed with respect to support vector machines in [21]. The basic support vector machine for inductive inference finds a separating hyperplane between two different classes by maximizing the margin of separation between the classes and minimizing the misclassification error. The key notion is that capacity control based on maximizing the margin as measured on the labeled training data prevents overfitting. For transduction, the capacity control is done based on all of the available data, both labeled and unlabeled. There are two recent approaches for constructing semi-supervised support vector machines algorithm. Integer programming was used in [2] to construct linear support vector machines. Computational results on UC Irvine data sets [14] such as the ones investigated in this paper found small improvements using both labeled and unlabeled data. More recently, an iterative quadratic programming method [11] for constructing both linear and nonlinear support vector machines found that incorporating unlabeled data led to significant improvements on large information retrieval problems. Both of these approaches were based on incorporating unlabeled data into a classification method.

In this paper, incorporating label information into an unsupervised learning approach is studied. The goal is to group both labeled and unlabeled data into the clusters where each cluster is as pure as possible in terms of class distribution provided by labeled data. The advantage of such an approach is that it can be used for both inductive and transductive inference. In addition the clusters help characterize segments of the population likely or unlikely to possess the target characteristic represented by the label. This additional information can be useful for several applications. For example, in database marketing only the most pure clusters of customers would be included in a marketing campaign and new products may be designed to reach customers in marginal clusters.

As a base to our semi-supervised algorithm, an unsupervised clustering method optimized with a genetic algorithm is used by incorporating a measure of classification accuracy used in decision tree algorithms, the Gini index [5]. Clustering algorithms that minimize some objective function applied to k-cluster centers are examined in this paper. Each point is assigned to the nearest cluster center by Euclidean distance. The goal is to choose the cluster centers that minimize some measure of cluster quality. Typically a cluster dispersion metric is used. If the mean square error, a measure of within-cluster variance, is used then the problem becomes similar to the classic K-means clustering [10]. An alternative metric, the Davies-Bouldin Index (DBI) [8], a function of both the within cluster variance and between-cluster center distances is also examined. By minimizing an objective function that minimizes a linear combination of the cluster dispersion measure and the Gini Index, the algorithm becomes semi-supervised. The details of the problem formulation are given in Section 2. Since the objective function is highly nonlinear and discontinuous with many local minima, it is optimized by using the C++ based genetic algorithm library package GAlib [22].

1.1 GA Related Clustering

Genetic Algorithms (GAs) are well known for being able to deal with complex search problems by implementing an evolutionary stochastic search. Because of this, GAs have been successfully applied to a variety of challenging optimization problems. The NP-hard nature of the clustering problem makes GA a natural choice for solving it such as in [7, 13, 15, 19]. A common objective function in these implementations is to minimize the square distance of the cluster dispersion:

$$E = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - m_k\|^2 \tag{1}$$

where K is the number of clusters and m_k is the center of cluster C_k. This is indeed the objective function for the K-means clustering algorithms. The algorithm proposed in [19] modifies this objective function by using the inverse of Davies-Bouldin index defined in [8, 10] and minimizing it by using evolutionary programming.

GAs are also implemented in [15] to minimize the function E (1). Genomes are represented by N-bit long strings where N is the number of data points. Each bit in genomes represents cluster membership. Although proper crossover and mutation operations are defined for this scheme, it is not a scalable algorithm due to the length of the genomes.

Clustering algorithms are widely used in pattern recognition. A clustering algorithm based on GAs for machine vision is introduced in [7]. The basic problem, defined in [7], is to group objects to a fixed number of clusters. A new gene representation, Boolean Matching Code (BMC), is defined in [7] as an alternative to the Linear Code (LC) used in [15]. The basic idea in BMC is that each gene represents one cluster with N bits where N is the number of objects. In this case the total size of solution space is 2^{KN}. Although the size of the solution space in LC is 2^{NlogK}, BMC reaches the convergence earlier than LC by utilizing better crossover and mutation operations. A single gene crossover operation was proposed in [7].

Since the proposed algorithm performs both supervised and semi-supervised clustering using both labeled and unlabeled data, there are some substantial differences between the proposed algorithm in this paper and the algorithms proposed in [7, 15, 19] in terms of learning tasks. Moreover, the GA approach in this paper differs in terms of genome representation compared to the ones used in [7, 15, 19]. These points are discussed further in Section 3.

1.2 Semi-supervised Clustering

Related approaches for combining supervised and unsupervised learning exist. The first known semi-supervised clustering algorithm was proposed in [16] in the fuzzy clustering (FC) context. The objective function of the fuzzy c-means clustering method was modified by using information from labeled data. To distinguish labeled and unlabeled data, a Boolean indicator vector and membership matrix for labeled data were introduced to the objective function. The terms coming from both labeled and unlabeled data were parameterized to study different scenarios in [16]. It was shown that semi-supervised clustering improved the solution by comparing average Hamming distance found after unsupervised and semi-supervised clustering methods [16].

Later semi-supervised fuzzy clustering (ssFC) was used in image segmentation successfully [3]. A small number of training data is enough to improve results by using ssFC. The training data define the number of clusters and cluster prototypes which affects cluster assignment for unlabeled data. In the case of unreliable training data, the weights coming from labeled data in the fuzzy objective function should be decreased [3]. Other applications of ssFC on different domains exist [1, 17].

There are significant differences between ssFC and the semi-supervised clustering method introduced in this paper. First of all, a hard cluster membership is used in this paper rather than a fuzzy membership. The objective function used in ssFC only includes a measure of cluster dispersion. It does not have a misclassification term. Labeled information is used indirectly to influence clustering in ssFC. The fuzzy membership matrix is defined by the user for the labeled data. This forces labeled points to be in certain clusters. In a way, the number of clusters is defined *a priori* in ssFC by labeled data. In this paper, a misclassification cost term is explicitly used in the objective function for semi-supervised clustering. Unlike ssFC, there are no constraints forcing any labeled point to be in certain clusters. Thus the number of clusters is independent of the number of classes defined by labeled data. Our algorithm starts with a maximal number of clusters and ends up with a near-optimal number of clusters. Other semi-supervised approaches [1, 3, 17, 16] have included a covariance matrix in the distance metric. The metrics investigated in this paper do not incorporate a covariance matrix. This extension is left for future work.

Semi-supervised approaches have been used in different contexts besides fuzzy clustering. For example Learning Vector Quantization (LVQ) [12] and Constrained Topological Maps (CTM) [6] also use the approach of adapting a primarily unsupervised method to perform classification. This paper helps address the interesting but still open question, of how well such methods can exploit the information in unlabeled data to support transductive inference. Moreover, nearest prototype classifiers are studied in [4, 13]. Selecting prototypes from a dataset using GA is compared with random selection in [13].

In Section 2, the problem definition and the proposed algorithm are given. Details about the GA implementation are given in Section 3. Experimental results are given in Section 4. A comparison with 3 nearest neighbor and linear and quadratic discriminant analyses is also reported in Section 4. Finally, we summarize our findings in the Section 5.

2 Problem Definition

Clustering, in general, is defined as grouping similar objects together by optimizing some similarity measure for each cluster such as within-group variance. Since clustering generally works in an unsupervised fashion, it is not necessarily guaranteed to group the same type (class) of objects together. In this case, supervision needs to be introduced to the learning scheme through some measure of cluster impurity. The basic idea is to find a set of clusters then minimize a linear combination of the cluster dispersion and cluster impurity measures. More specifically, select $K > 2$ cluster centers, m_k $(k = 1, ..., K)$, that minimize the following objective function:

$$\min_{m_k, k=1,...,K} \quad \beta * Cluster\_Dispersion \alpha * Cluster\_Impurity \tag{2}$$

where $\alpha > 0$ and $\beta > 0$ are positive regularization parameters.

If $\alpha = 0$, then the result is a purely unsupervised clustering algorithm. If $\beta = 0$ the result is a purely supervised algorithm that tries to minimize the cluster impurity. As in the K-means algorithm, each point is assumed to belong to the nearest cluster center as measured by Euclidean distance. Each non-empty cluster is assigned a class label corresponding to the majority class of points belonging to that cluster. Two cluster dispersion measures from the clustering literature will be examined: mean square error or distance (see Section 2.1) and Davies-Bouldin Index (see Section 2.2). It is important to note that for the induction case, cluster dispersion is based on the labeled training data. For the transduction case, the cluster dispersion is based on all available data, both labeled and unlabeled. For the cluster impurity measure, a measure of partition quality common in decision tree algorithms, the Gini Index, is used (see Section 2.3). Note that the Gini Index is particularly difficult to optimize since it is a function of the number of points from each class occurring in each cluster. For such choices of the cluster dispersion and cluster impurity measures, the objective function (Eq.2) is highly discontinuous with many local minima. Thus we optimize it using the genetic algorithm library (see Section 3) GAlib. Practical details of this algorithm are discussed in the computational results section

Like many popular clustering algorithms such as K-means and EM model mixture approaches, the GA algorithm requires a choice of the maximum number of clusters, K. The GA algorithm can effectively eliminate clusters by allowing them to become empty. Intermediary and final GA solutions may contain clusters with little or no points assigned to them. This allows the GA to prune unnecessary clusters and to construct new clusters in new parts of the search space as part of the evolutionary process. After the GA halts, clusters with little or no points are deleted and any relevant points are reassigned to their nearest cluster centers. It is not desirable to choose K arbitrarily large because this greatly increases the search complexity and may produce overfitting. Only a rough guess of the maximum number of clusters is required for good results (see Section 4).

The resulting algorithm can be summarized as follows:

Algorithm 2.1 *Semi-supervised clustering algorithm*

- *Within genetic algorithm:*
  1. *Determine cluster centers*
  2. *Partition the labeled data by distance to closest cluster center.*
  3. *Find non-empty clusters, assign a label to non-empty clusters by majority class vote within them.*
  4. *Compute dispersion and impurity measures:*
     - *Induction: Use labeled data.*
     - *Transduction: Use labeled + unlabeled data.*

- *Prune clusters with few members.*
- *Reassign the points to final non-empty clusters.*

We now discuss the cluster dispersion and impurity measures. In this paper, we investigated measures based on hard cluster assignment. Each point is assigned to its nearest cluster center based on Euclidean distance. There are many possible variations to these measures. Other distance measure could be used to determine the cluster assignment, e.g. a covariance metric could be used to produce a weighted distance. Soft cluster assignment could be done by weighting a point's membership in a cluster by its distance to the cluster center.

2.1 First Dispersion Measure: MSE

The average within cluster variance is frequently used in clustering techniques as a measure of cluster quality. Commonly known as the mean square distance or error (MSE), this quantity is defined as:

$$MSE \ = \ \frac{1}{N} \sum_{k=1}^{K} \sum_{x \in C_k} \|x - m_k\|^2, \tag{3}$$

where N is the number of points, K is the number of clusters, and m_k is the center of cluster C_k. The K-means algorithm minimizes the MSE objective. Note that the MSE objective is not convex since it is dependent on the assignment of the points to the clusters.

2.2 Second Dispersion Measure:DBI

The Davies-Bouldin index is used as an alternative to MSE. DBI is determined as follows [10] : Given a partition of the N points into K clusters, one first defines the following measure of within-to-between cluster spread for two clusters, C_j and C_k for $1 \leq j, k \leq K$ and $j \neq k$.

$$R_{j,k} = \frac{e_j + e_k}{D_{jk}}, \tag{4}$$

where e_j and e_k are the average dispersion of C_j and C_k, and D_{jk} is the Euclidean distance between C_j and C_k. If m_j and m_k are the centers of C_j and C_k, consisting of N, and N_k points respectively:

$$e_j = \frac{1}{N_j} \sum_{x \in C_j} \|x - m_j\|^2 \tag{5}$$

and $D_{jk} = \|m_j - m_k\|^2$.

The term R_k for each C_k is defined as

$$R_k = max_{j \neq k} R_{j,k}. \tag{6}$$

Now the DBI is defined as:

$$DB(K) = 1/K \sum_{k=1}^{K} R_k \tag{7}$$

The DBI can be incorporated into any clustering algorithm to evaluate a particular segmentation of data. The DBI takes into account cluster dispersion and the distance between cluster means. Well-separated compact clusters are preferred. The DBI favors small numbers of clusters. Optimizing the DBI frequently eliminates clusters by forcing them to be empty.

2.3 Impurity Measure: Gini-Index

The Gini index has been used extensively in the literature to determine the impurity of a certain split in decision trees [5]. Usually, the root and intermediate nodes are partitioned into two children nodes. In this

case, left and right nodes will have different Gini index values. If any split reduces the impurity, the decision tree at that node is partitioned further and the decision rule which yields the minimum impurity is selected. Clustering using K cluster centers partitions the input space into K regions. Therefore clustering can be considered as a K-nary partition at a particular node in a decision tree, and the Gini index can be applied to determine the impurity of such a partition. In this case, the Gini Index of a certain cluster is computed as:

$$GiniP_j \;=\; 1.0 - \sum_{i=1}^{c} (\frac{P_{ji}}{N_j})^2 \qquad j \; in \; 1, ..., K \tag{8}$$

where P_{ji} is the number of points belong to i^{th} class in cluster j. Labeled data defines c different classes. N_j is the total number of points in cluster j. The impurity measure of a particular partitioning into K clusters is:

$$impurity \;=\; \frac{\sum_{j=1}^{K} T_{P_j} * GiniP_j}{N} \tag{9}$$

where T_{P_j} is the probability of a point belonging to cluster j and N is the number of points in the dataset. The Gini index is based on the number of points of each class within a cluster so it is a discontinuous function of the centers of the points.

Preliminary experiments indicated that the Gini index is generally preferable over other impurity measures such as the number of misclassified points. If the simple number of misclassified points is used, then the misclassified points may be distributed evenly throughout all the clusters. The Gini Index favors solutions with pure clusters even at the expense of total classification error. Other decision tree splitting criterion such as Information Gain could also be used for cluster impurity measures [18].

3 Genetic Representation and Algorithm

The objective function (Eq. 2) defined in the previous section is a function of the discontinuous and non-convex impurity and dispersion measures. Finding an optimal solution to this problem is extremely difficult, so heuristic search is desirable. Heuristic search approaches such as genetic algorithms (GA), evolutionary programming, simulated annealing and tabu search have been used extensively in the literature to optimize related problems. A genetic algorithm approach is utilized because the objective function defined can be readily used as a fitness function in the GA. As opposed to developing a genetic algorithm from scratch, a general purpose GA library, GAlib [22], is customized by utilizing the floating-point representation and Goldberg's simple GA approach [9]. This algorithm uses non-overlapping populations. In each generation, the algorithm creates an entirely new population of individuals by selecting from the previous population then mating to produce the new offspring for the new population. This process continues until the stopping criteria have been met. An elitist strategy was applied that allows the best individual to pass to the new generation.

In a genetic algorithm application major concerns are genome representation, initialization, selection, crossover and mutation operators, stopping criteria and most importantly the fitness function. The objective function (Eq.2), defined above, is directly used as the fitness function without any scaling.

In most of the GA applications, one would find a binary representation as opposed to a floating-point one [19, 15, 7]. In the binary case, the GA explicitly represents the assignments of the data points to the clusters. In our floating point representation, we represent the cluster centers. Cluster assignment is done implicitly based on distance. Many crossover, mutation, and other operators are defined for binary representation. Fortunately for the floating point representation, GAlib comes with a built-in real genome class which allows users to represent the individuals in the population by real number arrays. Our representation consists of an array of Kd real numbers, where d is the number of dimensions in the data and K is the maximal number of clusters. K need not be very close to the true number of clusters. Our algorithm prunes unnecessary clusters. We use the heuristic of choosing K based on the number of points in the data. For larger datasets one can afford to have a broader search region without risk of overfitting. But having a larger search space may lead the GA to get stuck in a poor local minimum as well. So there is always a trade-off between a

throrough search in a small space and diverse enough search space for the GA algorithm. Each set of d numbers represents one cluster center.

The selection of K in this type of representation has several advantages over the prior discrete GA representation of the cluster membership. First, cluster memberships are assigned based on the Euclidean distance metric in this case instead of assigning them based on the values of the genome. Second, each genome requires less search space than previous applications for large datasets, since the length of the genomes depends only on the number of clusters (K) and the dimensionality (d) of the dataset, not on the number of data points. It is therefore possible to handle large datasets with this representation.

The default genetic operators defined for the `GARealGenome` class in GAlib were used in our GA implementation. Mutation with Gaussian noise is the default in this case. GAlib has built-in crossover operators that include partial match, ordered, cycle, single point, two point, even, odd, uniform, node- and subtree-single point. Uniform crossover was used as the default crossover operation in `GARealGenome`. The crossover operator defines the procedure for generating a child from two parent genomes. A Roulette Wheel selection rule was used for selecting mating individuals (parents). This selection method picks an individual based on the ratio of the individual's fitness score to the sum of fitness scores of the population. The higher the fitness score is, the more likely an individual will be selected [22]. Although the uniform initializer is used by default, the population was initialized by sampling from the data.

Two stopping criteria were utilized as in most GA applications. These are the maximum number of generations and the convergence after a certain number of consecutive generations. The algorithm stops when either of these is satisfied.

The genetic algorithm yields reasonable results for both induction and transduction problems. The experimental findings are summarized in the next section.

4 Experimental Results

Our goal in the experiments was to determine if using unlabeled data in addition to the labeled training data can improve generalization. More specifically, we were investigating whether transduction works or not. Can using the unlabeled data in training improve the generalization ability? For each dataset, two scenarios have been tested to understand the differences between inductive and transductive inference. In order to ascertain the differences, the original datasets were divided into three parts: training, working and test sets. The training set is the only labeled data used to construct the classifiers. Working and test sets form our unlabeled data during training. For the inductive inference, the algorithm is applied to labeled training data and then tested on the test set. For the transductive inference, the algorithm is applied to all the available data on hand (training,working and test sets), and then tested on the test set. The same labeled training data are used for both methods. But in transduction, the unlabeled test and working set data is also used for training. In any phase of transductive inference, no labeled information is used from the unlabeled data. The same amount of labeled data is used for both the induction and transduction experiments. Previous work on semi-supervised clustering approaches did not investigate transduction. In [16, 3, 17, 1] clusters were constructed using only labeled training and some unlabeled data and then tested on a completely different set of data. In this work, to investigate transduction, the test set is used as the unlabeled data during training. In short, for both inductive and transductive inference prediction is done only on the test set. In addition to the test set, working set is also used only in transduction. No prediction is done for the working set in both inductive and transductive inference.

The experimental study was done using nine datasets from the UC-Irvine Machine Learning Repository. The datasets and their corresponding sizes are: Bright (14 variables, 2462 points), Sonar (60 variables, 208 points), Cleveland Heart(13 variables, 297 points), Ionosphere (34 variables, 351 points), Boston Housing (13 variables, 506 points), House Votes (16 variables, 435 points), Breast Cancer Diagnostic (30 variables, 569 points), Pima Diabetes (8 variables, 769 points), and Iris (4 variables, 150 points). The datasets have categorical dependent variables except Housing. The continuous dependent variable for this dataset was categorized at the level of 21.5. Iris is a three class problem. The other datasets are two class problems. Each dataset was divided into three subsets after standard normalization. These subsets are called the training, test and working sets. Currently 40% of data is for the training set, 30% is for the test set and remaining 30% is for the working set. For each of the 10 trials, the training set is used as labeled data and

the test and work sets are used as unlabeled data. After the clusters are constructed, the labels of the testing set are used to compute the testing error.

Results from seven different fitness functions are reported. The two different cluster dispersion measures, MSE (Eq.3) and Davies-Bouldin Index (Eq.7), are applied to induction in a completely unsupervised mode ($\beta = 1, \alpha = 0$) and semi-supervised mode ($\alpha > 0, \beta > 0$), and transduction in a semi-supervised mode ($\alpha > 0, \beta > 0$). We also tried the completely supervised case based on only the Gini index ($\beta = 0, \alpha = 1$). For transduction, both the cluster dispersion measure and the Gini index are based on the labeled and unlabeled data. In transduction, the Gini index (Eq.8) becomes: $GiniP_j = 1.0 - \sum_{i=1}^{k} (\frac{P_{ji}}{\hat{N}_j})^2$ where $\hat{N}_j$ is equal to number of labeled and unlabeled points in cluster j^{th}.

The best parameter set for the problem was picked by trial and error to have roughly equal weight for both dispersion and impurity measures. Since the dispersion and accuracy measures have different scales, this corresponds to the fixed values of $\beta = 0.01$ and $\alpha = 1$. We use the same set of GA parameters for each dataset. The maximum number of generations is 500, the mutation probability is 0.01, probability of crossover is 0.95, and the number of generations to converge is 50. Experiments are conducted based on 10 random partitions of each dataset. For brevity only the average test set error results are reported here. A paired t-test was used to assess the significance of differences between transductive and inductive learning. Errors with a p-value less than 0.2 were considered significant. To ensure that the weaker performance of MSE was not based on poor choice of parameters, (K, β, α) for each dataset were chosen based on trials with the inductive MSE with Gini index[1].

To be consistent in our experiments, we use $K = 7$, $K = 11$ and $K = 15$ for small, medium and large datasets respectively. Again we should emphasize that the choice of K is a trade-off. Using a very large value creates a very complex and large search space which might result in very poor GA performance. On the other hand, using very small values will result in a very limited search space for the GA. In addition, the values of K used in this paper do not represent the underlying true number of clusters in any given dataset. These numbers are large enough to have a meaningful search space. Another reason for having larger Ks is to show that our algorithm finds the near optimum number of clusters.

Parameters β and α were picked to ensure approximately equal weight on both dispersion and impurity measures in the objective function. Since dispersion and impurity measures have different magnitude, we need to select β and α accordingly to ensure approximately equal weight from both dispersion and impurity measures in the objective function. Basically, by choosing β and α accordingly, we scale up and down the components of the objective function to have approximately equal influence from both of them. For the DBI-based results, the same values of K were used for each dataset as in the MSE case, and the fixed values of $\beta = 0.01$ and $\alpha = 1$ were used for all datasets.

4.1 First Dispersion Measure:MSE

The results using the first dispersion measure, MSE, are reported in Table 1. The first column, MSE-only, indicates how the totally unsupervised approach of clustering based on only the unlabeled training data would perform. The second column, Gini-only, shows how the completely supervised approach of clustering using the Gini index on the labeled training data performs. The third column is the proposed approach using both the MSE and Gini based on the labeled training data. The fourth column indicates how MSE+Gini performs transductive inference when all the available data are used. A **bold** number is the minimum error for a given dataset, a * indicates that the result is significantly different from the transduction result. Conclusively both supervised and semi-supervised clustering perform better than the unsupervised MSE-only approach. Apparently, completely supervised clustering (the Gini-only) approach was the best on five of the nine datasets. The transductive MSE+Gini method based on all available data showed no consistent improvements over the induction approach.

The MSE-based approaches showed poor performance. To examine why consider the results of the MSE-based fitness functions on the cartoon example shown in Figure 1. The top plot shows the induction result (using just labeled data) and the bottom plot shows the transduction result (using both labeled and unlabeled data) cases. Note that on the center cluster transduction does work appropriately. The inductive

[1]The (k, β, α) values applied for each dataset were bright (15, 0.01,0.99), sonar (7,0.1,1), heart (7,0.25,0.75), ionosphere (7, 0.01,0.99), house (7,0.1,0.9), housing (11,0.01, 0.99), diagnostic (11,0.4,0.6), pima (11,0.01,0.99) and iris(7,0.01,1).

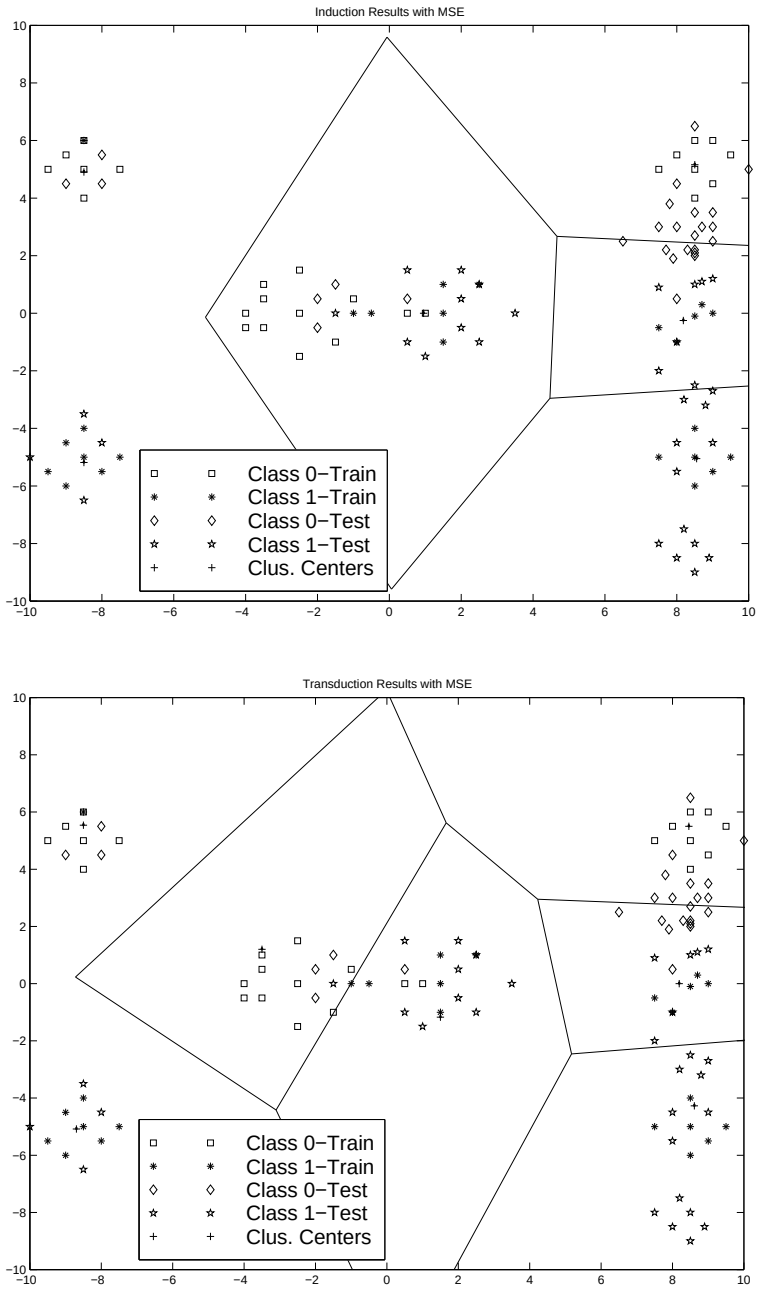

Figure 1: Induction and Transduction Results using MSE

MSE+Gini method does not separate the cluster in the center of the figure, because such separation will result an impure cluster on the right. Using the additional unlabeled data, the transductive MSE+Gini reduces cluster dispersion by splitting the center cluster (bottom of Figure 1). On the other hand, both the transductive MSE+Gini methods do not catch the downward shift of the top right cluster. Because the added unlabeled points are roughly equal distance from two top right cluster centers, adding unlabeled data has little effect despite the fact that a natural gap exits between the two clusters. The MSE minimizes only the compactness of the clusters. It is necessary to find clusters that are both compact and well separated. The DBI is much more effective in this regard and the computational results are greatly improved when this cluster dispersion metric is applied.

4.2 Second Dispersion Measure:DBI

The DBI dispersion measure was much more effective than the MSE with regards to transduction. For the cartoon example, the top of Figure 2 shows the resulting partition after using DBI in fitness function for the induction case. By using DBI, the method was able to catch the vertical shift within the data in transductive inference. The results for the DBI dispersion measure (Eq.7) on the UC-Irvine Data are reported in Table 2. The same experimental setup and parameters as mentioned above were used except that the maximum number of generations is set to 300. As a purely unsupervised approach DBI-only was even worse than MSE-only for classification. The inductive DBI+Gini approach was not significantly different from the Gini-Only approach. The transductive DBI+Gini was either better or not significantly worse than the Gini-only approach. Transductive DBI+Gini consistently produced the best classification results of all the seven approaches tested primarily due to the more compact and better-separated clusters found by DBI over MSE. The evidence indicates that capacity control based on both labeled and unlabeled data is much more effective using the DBI criterion than MSE.

Another finding from the results is that the DBI dispersion measure favors a fewer number of non-empty clusters compared to the MSE dispersion measure. On average, using the DBI dispersion measure resulted in 53%, 33%, 28%, 18%, 31%, 33%, 51%, 51% and 37% fewer non-empty clusters than the MSE dispersion measure for transductive inference on the UCI datasets respectively. The initial number of clusters, K, is picked large enough to show that our approach ends up with a near-optimal non-empty clusters. Experiments with Iris data have shown that the DBI dispersion measure ended up finding 4.1 clusters out of 7 on average.

In Table 3, results from some other classification techniques are compared with the transductive DBI+Gini – 3 nearest neighbor classifier, linear and quadratic discriminant classifiers. The discriminant analysis was done using the SAS procedure DISCRIM [20]. All results are reported on the test datasets. The transductive DBI-Gini is consistently one of the best methods. It performed the best or second best on 8 of the 9 datasets. The linear classifier also performed very well (except for a catastrophic failure on the Sonar data). The proposed semi-supervised clustering approach has the advantage over the linear approach in that it performs segmentation. It determines a set of clusters representing each class. The cluster centers can be used to characterize each class center. The resulting information can be useful in applications such as database marketing where ranking and segmentation are part of the classification task.

5 Conclusion

A novel method for semi-supervised learning that combines aspects of supervised and unsupervised learning techniques has been introduced in this paper. The basic idea is to take an unsupervised clustering method, label each cluster with the class membership, and simultaneously optimize the misclassification error of the resulting clusters. The intuition behind this approach is that the unsupervised component of the objective function acts as a form of regularization or capacity control during supervised learning to avoid overfitting. The objective function now is a linear combination of a measure of cluster dispersion and a measure of cluster impurity. The method can exploit any available unlabeled data during training since the cluster dispersion measure does not require class labels. This allows the approach to be used for transductive inference, the process of constructing a classifier using both the labeled training data and the unlabeled test data. Experimental results also show that using the Davies-Bouldin Index for cluster dispersion instead of Mean Square Error improves transductive inference. This is due to the compact and well-separated clusters

Table 1: Results Using MSE in Fitness Function

	Induction			Transduction
Data Set	MSE-Only	Gini-Only	MSE+Gini	MSE+Gini
Bright	0.06585*	**0.01084 ***	0.02507	0.02263
Sonar	0.43279*	0.25410	**0.22951***	0.26066
Heart	0.23636*	0.21477*	0.20000	**0.19659**
Iono.	0.25673*	0.14423	**0.12788**	0.12981
Housing	0.25828*	**0.15629***	0.18874*	0.16887
House	0.09846*	0.06692	**0.06000**	0.06308
Diagnos.	0.10000*	**0.05059**	0.06235*	0.05235
Pima	0.32402*	**0.27118 ***	0.30131	0.30393
Iris	0.19111*	**0.07111**	0.08000	0.07556

*Significant difference from Transduction denoted by **

Table 2: Results Using DBI in Fitness Function

	Induction			Transduction
Data Set	DBI-Only	Gini-Only	DBI+Gini	DBI+Gini
Bright	0.26897	**0.01084**	0.01992*	0.01165
Sonar	0.50656*	0.25410	0.27049*	**0.23771**
Heart	0.38410*	0.21477*	0.21136*	**0.19155**
Iono.	0.34327*	0.14423	**0.12885**	0.13558
Housing	0.45630*	0.15629	0.17086*	**0.15497**
House	0.11769*	**0.06692**	0.07462	0.06923
Diagnos.	0.38059*	0.05059*	0.04941*	**0.04353**
Pima	0.34585*	**0.27118**	0.28428	0.28122
Iris	0.20222*	0.06444*	0.06444*	**0.04889**

*Significant difference from Transduction denoted by **

Table 3: Comparison between Transductive DBI+Gini and 3-NN, LD, and QD

Data Set	3-NN	LinDisc	QuadDisc	DBI+Gini
Bright	0.01247*	0.02387 *	0.02112*	**0.01165**
Sonar	**0.20980***	0.38025*	0.35256*	0.23771
Heart	0.19773	**0.17450***	0.22334*	0.19155
Iono.	0.18846*	0.14624	**0.12940**	0.13558
Housing	0.16291*	0.16013	0.19946*	**0.15497**
House	0.06154	**0.04140***	0.06995	0.06923
Diagnos.	**0.04235**	0.04797	0.05348*	0.04353
Pima	0.28777	**0.23130 ***	0.26401*	0.28122
Iris	0.06666*	*0.03300* *	0.11232*	0.04889

*Significant difference from Transductive DBI+Gini denoted by **

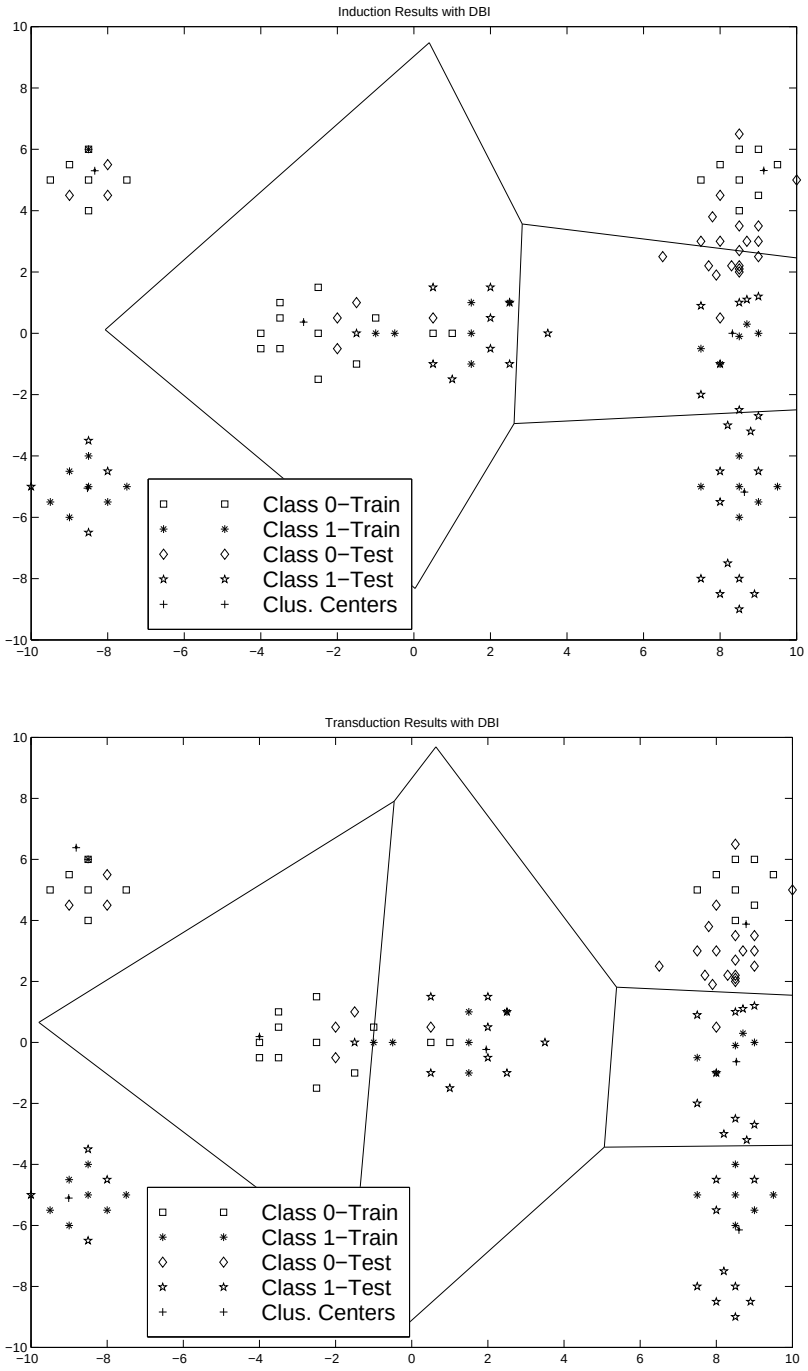

Figure 2: Induction and Transduction Results using DBI

found by minimizing DBI. DBI finds solution using much fewer clusters than MSE with much greater accuracy. Conclusively, using labeled data improved classification accuracy compared to completely unsupervised clustering. The basic ideas in this paper, incorporating class information into an unsupervised algorithm and using the resulting algorithm for transductive inference, are applicable to many types of unsupervised learning and are promising areas of future research.

Acknowledgements

This work was partially supported by NSF IRI-9702306, NSF IIS-9979860 and NSF DMS-9872019. Many thanks to the referees for their helpful comments.

References

[1] M. Benkhalifa, A. M. Bensaid, and A. Mouradi. Text categorization using the semi-supervised fuzzy c-means algorithm. In *18th International Conference of the North American Fuzzy Information*, pages 561–565, 1999.

[2] K.P. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems,11*, pages 368–374, Cambridge, MA, 1999. MIT Press.

[3] A.M. Bensaid, L.O. Hall, J.C. Bezdek, and L.P. Clarke. "Partially Supervised Clustering for Image Segmentation". *Pattern Recognition*, 29(5):859–871, 1996.

[4] J.C. Bezdek, T.R. Reichherzer, G.S. Lim, and Y. Attikiouzel. "Multiple Prototype Classifier Design". *IEEE Transactions on Systems, Man, and Cybernetics*, 28(1):67–79, 1998.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International, California, 1984.

[6] V. Cherkassky, Y. Kim, and F Mulier. Constrained topological maps for regression and classification. In *Proceedings of Int. Conf. on Neural Information Processing, New Zealand*, November 1997.

[7] R. Cucchiara. "Genetic Algorithms for Clustering in Machine Vision". *Machine Vision and Applications*, 11:1–6, 1998.

[8] D.L. Davies and D.W. Bouldin. "A Cluster Separation Measure". *IEEE Transcations on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

[9] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison–Wesley, Reading, MA, 1989.

[10] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, N.J., 1988.

[11] T. Joachims. "Transductive Inference for Text Classification using Support Vector Machines". In *The Proceedings of 16th International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann, San Fransisco, CA, 1999.

[12] T. Kohonen. "Learning Vector Quantitization". *Neural Networks*, 1, 1988.

[13] L.I. Kuncheva and J.C. Bezdek. "Nearest Prototype Classification: Clustering, Genetic Algorithms, or Random Search?". *IEEE Transactions on Systems, Man, and Cybernetics*, 28(1):160–164, 1998.

[14] P.M. Murphy and D.W. Aha. *UCI repository of machine learning databases*. Department of Information and Computer Science, University of California, Irvine, California, 1992.

[15] C.A. Murthy and N. Chowdhury. "In Search of Optimal Clusters Using Genetic Algorithms". *Pattern Recognition Letters*, 17:825–832, 1996.

[16] W. Pedrycz. "Algorithms of Fuzzy Clustering with Partial Supervision". *Pattern Recognition Letters*, 3(1):13–20, 1985.

[17] W. Pedrycz and J. Waletzky. "Fuzzy Clustering with Partial Supervision". *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 27(5):787–795, 1997.

[18] J. R. Quinlan. "Induction of Decision Trees". *Machine Learning*, 1:81–106, 1984.

[19] M. Sarkar, B. Yegnanarayana, and D. Khemani. "A Clustering Algorithm Using an Evolutionary Programming-based Approach". *Pattern Recognition Letters*, 18:975–986, 1997.

[20] SAS Institute Inc., Cary, North Carolina. *SAS Language and Procedures:Usage, Version 6*, 1989.

[21] V.N. Vapnik. *Statistical Learning Theory*. Wiley Inter-Science, 1998.

[22] M. Wall. *GAlib: A C++ Library of Genetic Algorithm Components*. MIT, http://lancet.mit.edu/ga/, 1996.