On Density Based Transforms for Uncertain Data Mining

Charu C. Aggarwal
IBM T. J. Watson Research Center
19 Skyline Drive, Hawthorne, NY 10532
charu@us.ibm.com

Abstract

In spite of the great progress in the data mining field in recent years, the problem of missing and uncertain data has remained a great challenge for data mining algorithms. Many real data sets have missing attribute values or values which are only approximately measured or imputed. In some methodologies such as privacy preserving data mining, it is desirable to explicitly add perturbations to the data in order to mask sensitive values. If the underlying data is not of high quality, one cannot expect the corresponding algorithms to perform effectively. In many cases, it may be possible to obtain quantitative measures of the errors in different entries of the data. In this paper, we will show that this is very useful information for the data mining process, since it can be leveraged to improve the quality of the results. We discuss a new method for handling error-prone and missing data with the use of density based approaches to data mining. We discuss methods for constructing error-adjusted densities of data sets, and using these densities as intermediate representations in order to perform more accurate mining. We discuss the mathematical foundations behind the method and establish ways of extending it to very large scale data mining problems. As a concrete example of our technique, we show how to apply the intermediate density representation in order to accurately solve the classification problem. We show that the error-based method can be effectively and efficiently applied to very large data sets, and turns out to be very useful as a general approach to such problems.

1 Introduction

While data collection methodologies have become increasingly sophisticated in recent years, the problem of inaccurate data continues to be a challenge for many data mining problems. This is because data collection methodologies are often inaccurate and are based on incomplete or inaccurate information. For example, the information collected from surveys is highly incomplete and either needs to be imputed or ignored altogether. In other cases, the base data for the data mining process may itself be only an estimation from other underlying phenomena. In many cases, a quantitative estimation of the noise in different fields is available. An example is illustrated in [8], in which error-driven methods are used to improve the quality of retail sales merchandising. Many scientific methods for data collection are known to have error-estimation methodologies built into the data collection and feature extraction process. We summarize a number of real applications, in which such error information can be known or estimated a-priori:

- When the inaccuracy arises out of the limitations of data collection equipment, the statistical error of data collection can be estimated by prior experimentation. In such cases, different features of observation may be collected to a different level of approximation.
- In the case of missing data, imputation procedures can be used [10] to estimate the missing values. If such procedures are used, then the statistical error of imputation for a given entry is often known a-priori.
- Many data mining methods are often applied to *derived* data sets which are generated by statistical methods such as forecasting. In such cases, the error of the data can be derived from the methodology used to construct the data.
- In many applications, the data is available only on a partially aggregated basis. For example, many demographic data sets only include the statistics of household income over different localities rather than the precise income for individuals.

The results of data mining are often subtly dependent upon the errors in the data. For example, consider the case illustrated in Figure 1. In this case, we have illustrated a two dimensional binary classification problem. It is clear that the errors along dimension 1 are higher than the errors along dimension 2. In addition to a test example X, we have illustrated two training examples Y and Z, whose errors are

illustrated in the same figure with the use of oval shapes. For a given test example X, a nearest neighbor classifier would pick the class label of data point Y. However, the data point Z may have a much higher probability of being the nearest neighbor to X than the data point Y. This is because the data point X lies within the error boundary of Z. It is important to design a technique which can use the relative errors of the different data points over the different dimensions in order to improve the accuracy of the data mining process.

Thus, it is clear that the failure to use the error information in data mining models can result in a loss of accuracy. In this paper, we will provide a general and scalable approach to uncertain data mining with the use of multivariate density estimation. We will show that density estimation methods provide an effective intermediate representation, which captures information about the noise in the underlying data. The density estimate can be leveraged by designing an algorithm which works with error-adjusted densities rather than individual data points. As a specific example, we will discuss the case of the classification problem. As we will see, only a few minor modifications to existing methods are required in order to apply the method a density based representation of the data. In general, we expect that our approach can be used for a wide variety of data mining applications which use density estimation as an intermediate representation during the analytical process.

In order to improve the scalability of our technique, we show how to use a compression approach in order to generalize it for very large data sets. This goal is achieved by developing a density estimation process which works with (error-adjusted) micro-clusters in the data. The statistics of these error-adjusted micro-clusters are used to compute a kernel function which can be used to estimate the density in a time proportional to the number of micro-clusters. Furthermore, since the micro-clusters are stored in main memory, the procedure can be efficiently applied to very large data sets, even when the probability densities need to recomputed repeatedly in different subspaces during the mining process.

This paper is organized as follows. In the next section, we will discuss the method of density based estimation of error-driven data sets. In section 3, we will discuss the application of the procedure to the problem of classification. We will show how to use the error estimates in order to gain additional information about the data mining process. In section 4, we will discuss the empirical results. Section 5 contains the conclusions and summary.

2 Kernel Density Estimation with Errors

We will first introduce some additional notations and definitions. We assume that we have a data set $\mathcal{D}$, containing N points and d dimensions. The actual records in the data are denoted by $\overline{X_1}$, $\overline{X_2}$, ... $\overline{X_N}$. We assume that the estimated error associated with the jth dimension for data point $\overline{X_i}$ is denoted by $\psi_j(\overline{X_i})$. The interpretation of this error value can vary with the nature of the data mining application. For example, in a scientific application in which the measurements can vary from one observation to another, the error value is the standard deviation of the observations over a large number of measurements. In a k-anonymity based data (or incomplete data) mining application, this is the standard deviation of the partially specified (or imputed) fields in the data. Even though the error may be defined as a function of the field (or dimension) in most applications, we have made the most general assumption in which the error is defined by both the row and the field. This can be the case in many applications in which different parts of the data are derived from heterogeneous sources.

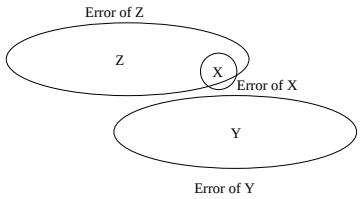

Figure 1. Effect of Errors on Classification

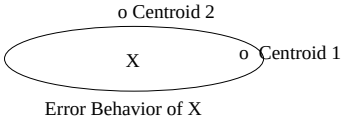

Figure 2. Effect of Errors on Clustering

The idea in kernel density estimation [11] is to provide a continuous estimate of the density of the data at a given point. The value of the density at a given point is estimated as the sum of the smoothed values of kernel functions $K'_h(\cdot)$ associated with each point in the data set. Each kernel function is associated with a kernel width h which determines the level of smoothing created by the function. The kernel estimation $\overline{f}(x)$ based on N data points and kernel function $K'_h(\cdot)$ is defined as follows:

$$\overline{f}(x) = (1/N) \cdot \sum_{i=1}^{N} K'_h(x - \overline{X_i}) \quad (1)$$

Thus, each discrete point $\overline{X_i}$ in the data set is replaced by a continuous function $K'_h(\cdot)$ which peaks at X_i and has a variance which is determined by the smoothing parameter h. An example of such a distribution would be a gaussian kernel with width h.

$$K'_h(x - \overline{X_i}) = (1/\sqrt{2\pi} \cdot h) \cdot e^{-(x-\overline{X_i})^2/(2h^2)} \quad (2)$$

The overall effect of kernel density estimation is to convert the (discrete) data set into a continuous density estimate by replacing each data point with a smoothed "bump", whose width is determined by h. The density distribution at a given coordinate is equal to the sum of the contributions of all the "bumps" represented by the data points. The result is a continuous distribution in which the random artifacts are suppressed and the density behavior provides a global overview of the dense as well as sparsely populated regions of the data. The estimation error depends upon the kernel width h which is chosen in a data driven manner. A widely used rule for approximating the bandwidth is the Silverman approximation rule [11] for which h may be chosen to be $1.06 \cdot \sigma \cdot N^{-1/5}$, where σ^2 is the variance of the N data points. It has been shown [11] that for most smooth functions $K'_h(\cdot)$, when the number of data points goes to infinity, the estimator $\overline{f}(x)$ asymptotically converges to the true density function $f(x)$, provided that the width h is chosen using the above relationship. For the d-dimensional case, the kernel function is chosen to be the product of d identical kernels $K_i(\cdot)$, each with its own smoothing parameter h_i.

The presence of errors can change the density estimates because of the different levels of error in different entries or fields. For example a data point or field with very large error should affect the density estimation of its locality to a smaller extent than one which contains small errors. When estimations of such errors are available, it is desirable to incorporate them into the estimation process. A direct way of doing so is to adapt the kernel function so that the measurement errors are taken into account during the calculation of the density distribution. Correspondingly, we define the following error-based kernel $Q'_h(x - X_i, \psi(\overline{X_i})$ function, which depends both upon the error as well as the values of the underlying data points.

$$Q'_h(x-X_i, \psi(\overline{X_i}) = (1/\sqrt{2\pi} \cdot (h + \psi(\overline{X_i}))) \cdot e^{\frac{-(x-X_i)^2}{(2\cdot(h^2+\psi(\overline{X_i})^2))}} \quad (3)$$

The overall effect of changing the kernel function is that the width of the bandwidth along the corresponding dimension is increased by $\psi(\overline{X_i})$. The intuition behind this choice of modifying the kernel is to adjust the contributions of the kernel function for a point depending upon its (error-based) probability density. Note that in the limiting case, when there are a large number of data points N, the value of the bandwidth h goes to zero, and this kernel function has a gaussian distribution with standard error exactly equal to the standard error of the data point. Conversely, the error-based kernel function converges to the standard kernel function when the value of the error $\psi(\overline{X_i})$ is 0. Therefore, in these boundary cases, the direct error-based generalization of the kernel function has a probability distribution with the same standard error as the data point. It is also clear that in the limiting case of a large number of data points, (when the bandwidth h tends to zero by the Silverman rule) the kernel function reflects the errors in each data point accurately. As in the previous case, the error-based density at a given data point is defined as the sum of the error-based kernels over different data points. Therefore, we define the error based density $\overline{f^Q}$ at point x as follows:

$$\overline{f^Q}(x, \psi(\overline{X_i})) = (1/N) \cdot \sum_{i=1}^{N} Q'_h(x - \overline{X_i}, \psi(\overline{X_i})) \quad (4)$$

As in the previous case, we can easily generalize the definition to the multi-dimensional case. Specifically, the error for the jth dimension is denoted by $\psi_j(\overline{X_i})$. The overall kernel function is defined as the product of the kernel function for the different dimensions.

Our aim is to use the joint probability densities over different subspaces in order to design data mining algorithms. This is much more general than the *univariate* approach for privacy preserving data mining discussed in [3], and is applicable to a much wider range of error-based and privacy-preserving approaches, since it does not require the probability distribution of the noise in the data. This joint probability density may need to be repeatedly computed over different subsets of dimensions for particular data mining problems. If the data set is too large to maintain in main memory, we need to perform repeated passes over the data for the computation of the density over different subsets of dimensions.

Since this option does not scale very well for large data sets, we need to develop methods to condense the data points into a smaller number of *pseudo-points*, but with slightly larger error. Such an approach is easily implementable for larger data sets, since the pseudo-points can be maintained in main memory for the computation of the density over different subsets of dimensions. However, this requires us to modify the method of computation of the error-based densities using micro-clusters [2] instead of individual data points.

2.1 Scalability for Large Data Sets

The method can be generalized to very large data sets and data streams. In order to generalize the approach to very large data sets, we condense the data into a smaller number of micro-clusters. While the concept of micro-clustering has been discussed in [2, 12], our aim in this paper is to modify and leverage it for error-based density estimation. In order to achieve this goal, we need to discuss how the error-based density may be computed using micro-cluster statistics instead of individual data points. Our first step is to define the micro-clusters in the data as suggested in [2]. However, since the work in [2] does not use errors, the definition of a micro-cluster needs to be modified correspondingly.

It is assumed that the data stream consists of a set of multi-dimensional records $\overline{X}_1 \ldots \overline{X}_k \ldots$ arriving at time stamps $T_1 \ldots T_k \ldots$. Each $\overline{X}_i$ is a multi-dimensional record containing d dimensions which are denoted by $\overline{X}_i = (x_i^1 \ldots x_i^d)$.

We will first begin by defining the concept of error-based micro-clusters more precisely.

Definition 1 *A micro-cluster for a set of d-dimensional points $X_{i_1} \ldots X_{i_n}$ with time stamps $T_{i_1} \ldots T_{i_n}$ is defined as the $(3 \cdot d + 1)$ tuple $(\overline{CF2^x}(\mathcal{C}), \overline{EF2^x}(\mathcal{C}), \overline{CF1^x}(\mathcal{C}), n(\mathcal{C})$, wherein $\overline{CF2^x}(\mathcal{C})$, $\overline{EF2^x}(\mathcal{C})$, and $\overline{CF1^x}(\mathcal{C})$ each correspond to a vector of d entries. The definition of each of these entries is as follows:*

- *For each dimension, the sum of the squares of the data values is maintained in $\overline{CF2^x}(\mathcal{C})$. Thus, $\overline{CF2^x}(\mathcal{C})$ contains d values. The p-th entry of $\overline{CF2^x}(\mathcal{C})$ is equal to $\sum_{j=1}^{n} (x_{i_j}^p)^2$.*
- *For each dimension, the sum of the squares of the errors in the data values is maintained in $\overline{EF2^x}(\mathcal{C})$. Thus, $\overline{EF2^x}(\mathcal{C})$ contains d values. The p-th entry of $\overline{EF2^x}(\mathcal{C})$ is equal to $\sum_{j=1}^{n} \psi_p(X_{i_j})^2$.*
- *For each dimension, the sum of the data values is maintained in $\overline{CF1^x}(\mathcal{C})$. Thus, $\overline{CF1^x}(\mathcal{C})$ contains d values. The p-th entry of $\overline{CF1^x}(\mathcal{C})$ is equal to $\sum_{j=1}^{n} x_{i_j}^p$.*
- *The number of points in the data is maintained in $n(\mathcal{C})$.*

We will refer to the micro-cluster for a set of points $\mathcal{C}$ by $\overline{CFT}(\mathcal{C})$. As in [12], this summary information can be expressed in an additive way over the different data points. This makes it very easy to create and maintain the clusters using a single pass of the data. The actual maintenance of the micro-clusters is a variation of the approach discussed in [2]. In this variation, we maintain the micro-cluster statistics for the q different centroids. These q centroids are chosen randomly. Each incoming data point is always assigned to its closest micro-cluster centroid using a nearest neighbor algorithm, and is never allowed to create a new micro-cluster. This is different from [2] in which a new micro-cluster is created whenever the incoming data point does not naturally fit in a micro-cluster. Similarly, clusters are never discarded as in [2]. This is required to ensure that all data points are reflected in the micro-cluster statistics. In addition, it is necessary to take the errors into account during the computation of the micro-cluster statistics. Consider the example illustrated in Figure 2 in which we have shown the error behavior of data point X by an elliptical error region. While the data point X is closer to centroid 2, it is more likely to belong to the cluster corresponding to centroid 1. This is because the error behavior of the data point is skewed in such a way that it would have been more likely to coincide with centroid 1 simply because of an error in measurement. Thus, we need to adjust for the errors corresponding to the different dimensions during the distance calculations. Thus, let us consider the data point $\overline{X}$ and centroid $\overline{c} = (c_1 \ldots c_d)$. Then, the distance $dist(\overline{Y}, \overline{c})$ between data point $\overline{Y} = (Y_1 \ldots Y_d)$ and $\overline{c}$ is given by the following relationship:

$$dist(\overline{Y}, \overline{c}) = \sum_{j=1}^{d} \max\{0, (Y_j - c_j)^2 - \psi_j(\overline{Y})^2\} \quad (5)$$

We note that this is an error-adjusted variation of the Euclidean distance metric. The error adjustment ensures that the dimensions which have large errors do not contribute significantly to the distance function. If the true distance along a particular dimension j is less than the average error $\psi_j(\overline{Y})$, then the value of the error-adjusted distance is defined to be zero. Therefore, we use a distance function which reflects the best-case scenario along each dimension. Such an approach turns out to be more effective for noisy data sets in high dimensionality [1].

The aim of the micro-clustering process is to compress the data so that the resulting statistics can be held in main memory for repeated passes during the density estimation process over different subspaces. Therefore, the number of micro-clusters q is defined by the amount of main memory available. Given the large memory sizes available even on modest desktop hardware today, this corresponds to thousands of micro-clusters for data sets containing hundreds of dimensions. This means that a high level of granularity in data representation can be maintained. This level of granularity is critical in using the micro-clustering as a surrogate for the original data.

Each micro-cluster is subsequently used as a summary representation in order to compute the densities over different subspaces. The key is to design a kernel function which adapts to the variance of the data points within a cluster as well as their errors. At the same time, the kernel function should be computable *using only the micro-cluster statistics of the individual clusters*. The first step is to recognize that each micro-cluster is treated as a pseudo-point with a new error defined both by the variance of the points in it as well as their individual errors. Thus, we need to compute the error of the pseudo-point corresponding to a micro-cluster. For this purpose, we assume that each data point $\overline{X}$ in a micro-cluster is a mutually independent observation with bias equal to its distance from the centroid and a variance equal to the error $\psi(\overline{X})$. Therefore, the true error $\phi(\overline{X}, \mathcal{C})^2$ of the data point $\overline{X}$ assuming that it is an instantiation of the pseudo-observation corresponding to the micro-cluster $\mathcal{C}$ is given by:

$$\phi_j(\overline{X}, \mathcal{C})^2 = bias_j(\overline{X}, \mathcal{C})^2 + \psi_j(\overline{X})^2 \quad (6)$$

Here $\phi_j(\overline{X}, \overline{C})$ refers to the error in the jth dimension. We note that this result follows from the well known statistical relationship that the true error is defined by the squared sum

of the bias and the variance. Once we have established the above relationship, we can use the independence assumption to derive the overall error for the micro-cluster pseudo-point in terms of the micro-cluster summary statistics:

Lemma 1 *The true error $\Delta(\mathcal{C})$ for the pseudo-observation for a micro-cluster $\mathcal{C} = \{Y_1 \ldots Y_r\}$ is given by the relationship:*

$$\Delta_j(\mathcal{C}) = \sum_{i=1}^{r} \frac{\phi_j(\overline{Y_i}, \mathcal{C})^2}{r} = \frac{CF2_j^x(\mathcal{C})}{r} - \frac{CF1_j^x(\mathcal{C})^2}{r^2} + \frac{EF2_j(\mathcal{C})}{r} \tag{7}$$

Proof: These results easily follow from Equation 6. By averaging the results of Equation 6 over different data points, we get:

$$\sum_{i=1}^{r} \phi_j(\overline{Y_i}, \mathcal{C})^2/r = \sum_{i=1}^{r} bias_j(\overline{Y_i}, \mathcal{C})^2/r + \sum_{j=1}^{r} \psi_j(\overline{Y_i})^2/r \tag{8}$$

We will now examine the individual terms on the right hand side of Equation 8. We first note that the value of $\sum_{i=1}^{r} bias_j(\overline{Y_i}, \mathcal{C})^2/r$ corresponds to the variance of the data points in cluster $\mathcal{C}$. From [12], we know that this corresponds to $CF2_j^x(\mathcal{C})/r - CF1_j^x(\mathcal{C})^2/r^2$. It further follows from the definition of a micro-cluster that the expression $\sum_{j=1}^{r} \psi_j(\overline{Y_i})^2/r$ evaluates to $EF2_j(\mathcal{C})/r$. By substituting the corresponding values in Equation 8, the result follows. ■

The above result on the true error of a micro-cluster pseudo-observation can then be used in order to compute the error-based kernel function for a micro-cluster. We denote the centroid of the cluster $\mathcal{C}$ by $c(\mathcal{C})$. Correspondingly, we define the kernel function for the micro-cluster $\mathcal{C}$ in an analogous way to the error-based definition of a data point:

$$Q'_h(x - c(\mathcal{C}), \Delta(\mathcal{C}) = (1/\sqrt{2\pi} \cdot (h + \Delta(\mathcal{C}))) \cdot e^{-\frac{(x - X_j)^2}{(2 \cdot (h^2 + \Delta(\mathcal{C})^2))}} \tag{9}$$

As in the previous case, we need to define the overall density as a sum of the densities of the corresponding micro-clusters. The only difference is that we need to define the overall density as the weighted sum of the corresponding kernel functions. Therefore, if the data contains the clusters $\mathcal{C}_1 \ldots \mathcal{C}_m$, then we define the density estimate at x as follows:

$$\overline{f^Q}(x, \Delta(\mathcal{C})) = (1/N) \cdot \sum_{i=1}^{m} n(\mathcal{C}_i) \cdot Q'_h(x - c(\mathcal{C}_i), \Delta(\overline{X_i})) \tag{10}$$

The density estimate is defined by the weighted estimate of the contributions from the different micro-clusters. This estimate can be used for a variety of data mining purposes. In the next section, we will describe one such application.

Algorithm *DensityBasedClassification*(Test Point: x, Accuracy Threshold: a);
begin
  $C_1 = \{1, \ldots d\}$;
  for each dimension S in C_1 and label l_j
    compute $\mathcal{A}(x, S, l_j)$;
  L_1 is the set of dimensions in C_1
   for which for some $j \in \{1 \ldots k\}$
    $\mathcal{A}(x, S, l_j) > a$;
  i=1;
  while L_i is not empty
  begin
    Generate C_{i+1} by joining L_i with L_1;
    for each subset S of dimensions in C_{i+1}
      and class label l_j compute $\mathcal{A}(x, S, l_j)$;
    L_{i+1} is the set of dimensions in C_{i+1}
     for which for some $j \in \{1 \ldots k\}$ we
      have $\mathcal{A}(x, S, l_j) > a$;
     $i = i + 1$;
  end;
  $\mathcal{L} = \cup_i L_i$;
  $N = \{\}$;
  while $\mathcal{L}$ is not empty
  begin
    Add set with highest local accuracy in $\mathcal{L}$ to N;
    Remove all sets in $\mathcal{L}$ which overlap with sets in N;
  end;
  report majority class label in N;
end

Figure 3. Density Based Classification of Data

3 Leveraging Density Estimation for Mining

We note that error-based densities can be used for a variety of data mining purposes. This is because the density distribution of the data set is a surrogate for the actual points in it. When joint distributions over different subspaces are available, then many data mining algorithms can be redesigned to work with density based methods. Thus, the key approach to apply the method to an arbitrary data mining problem is to design a *density based algorithm* for the problem. We further note that clustering algorithms such as DBSCAN [5] and many other data mining approaches work with joint probability densities as intermediate representations. In all these cases, our approach provides a direct (and scalable) solution to the corresponding problem. We further note that unlike the work in [3], which is inherently designed to work with univariate re-construction, our results (which are derived from only simple error statistics rather than entire distributions) are tailored to a far wider class of methods. The joint probability distribution makes it easier to directly generate error-based generalizations of more complex methods such as multi-variate classifiers.

In this paper, we will discuss such a generalization for the classification problem [4, 7]. We define the notations for the classification problem as follows: We have a data

set $\mathcal{D}$ containing a set of d-dimensional records, and k class labels which are denoted by $l_1 \ldots l_k$. The subset of the data corresponding to the class label l_i is denoted by $\mathcal{D}_i$. In this paper, we will design a density based adaptation of rule-based classifiers. Therefore, in order to perform the classification, we need to find relevant classification rules for a particular test instance. The challenge is to use the density based approach in order to find the particular subsets of dimensions that are most discriminatory for a particular test instance. Therefore, we need to find those subsets of dimensions in which the instance-specific local density of the data for a particular class is significantly higher than its density in the overall data. The first step is to compute the density over a subset of dimensions $\mathcal{S}$. We denote the density at a given point x, subset of dimensions S, and data set $\mathcal{D}$ by $g(x, S, \mathcal{D})$. The process of computation of the density over a given subset of dimensions is exactly similar to our discussion of the full dimensional case, except that we use only the subset of dimensions S rather than the full dimensionality.

The first step in the classification process is to pre-compute the error-based micro-clusters together with the corresponding statistics. Furthermore, the micro-clusters are computed separately for each of the data sets $\mathcal{D}_1 \ldots \mathcal{D}_k$ and $\mathcal{D}$. We note that this process is performed only *once* as a pre-processing step. Subsequently, the compressed micro-clusters for the different classes are used in order to generate the accuracy density estimates over the different subspaces. The statistics of these micro-clusters are used in order to compute the densities using the relations discussed in the previous section. However, since the densities need to be computed in an example-specific way, the density calculation is performed during the classification process itself. The process of data compression makes the density computation particularly efficient for large data sets. We also note that in order to calculate the density over a particular subspace, we can use Equations 9, and 10, except that we only need to apply them to a subspace of the data rather than the entire data dimensionality.

In order to construct the final set of rules, we use an iterative approach in which we find the most relevant subspaces for the classification process. In order to define the relevance of a particular subspace, we define its density based local accuracy $\mathcal{A}(x, S, l_i)$ as follows:

$$\mathcal{A}(x, S, l_i) = \frac{|\mathcal{D}_i| \cdot g(x, S, \mathcal{D}_i)}{|\mathcal{D}| \cdot g(x, S, \mathcal{D})} \tag{11}$$

This dominant class $dom(x, S)$ at point x is defined as the class label with the highest accuracy. Correspondingly, we have:

$$dom(x, S) = \text{argmax}_i \mathcal{A}(x, S, l_i) \tag{12}$$

In order to determine the most relevant subspaces, we perform a bottom up methodology to find those combinations of dimensions which have high accuracy for a given test example. We also impose the requirement that in order for a subspace to be considered in the set of $(i+1)$-dimensional variations, at least one subset of it needs to satisfy the accuracy threshold requirements. We impose this constraint in order to facilitate the use of a roll-up technique in our algorithm. In most cases, this does not affect the use of the technique when only lower dimensional projections of the data are used for the classification process. In the roll-up process, we assume that C_i is a set of candidate i-dimensional subspaces, and L_i is a subset of C_i, which have sufficient discriminatory power for that test instance. We iterate over increasing values of i, and join the candidate set L_i with the set L_1 in order to determine C_{i+1}. We find the subspaces in C_{i+1} which have accuracy above a certain threshold (subject to the additional constraints discussed). In order to find such subspaces, we use the relationships discussed in Equations 11 and 12. Thus, we need to compute the local density accuracy over each set of dimensions in C_{i+1} in order to determine the set L_{i+1}. We note that this can be achieved quite efficiently because of the fact that the computations in Figure 11 can be performed directly over the micro-clusters rather than the original data points. Since there are significantly fewer number of micro-clusters (which reside in main memory), the density calculation can be performed very efficiently. Once the accuracy density for each subspace in C_{i+1} has been computed, we retain only the dimension subsets for which the accuracy density is above the threshold a. This process is repeated for higher dimensionalities until the set C_{i+1} is empty. We assume that the final set of discriminatory dimensions are stored in $\mathcal{L} = \cup_i L_i$.

Finally, the non-overlapping sets of dimensions with the highest accuracy above the pre-defined threshold of a are used in order to predict the class label. In order to achieve this goal, we first determine the subspace of dimensions with the highest local accuracy. Then, we repeatedly find the next non-overlapping subset of dimensions with the highest level of accuracy until all possibilities are exhausted. The majority class label from these non-overlapping subsets of dimensions are reported as the relevant class label. We note that we do not necessarily need to continue with the process of finding the next overlapping subspace repeatedly until all possibilities are exhausted. Rather, it is possible to terminate the process after finding at most p non-overlapping subsets of dimensions.

This kind of approach can in fact be generalized to any data mining approach which use probability densities instead of individual data points. Many data mining algorithms can be naturally extended from point based processing to density based processing in order to apply this kind of approach. This is useful in order to leverage this methodology in general and practical settings. The overall algorithm is illustrated in Figure 3.

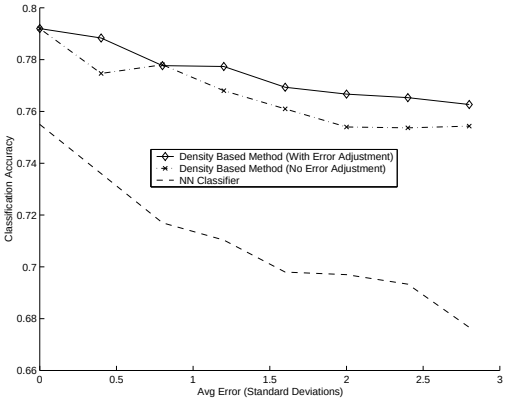

Figure 4. Error Based Classification for Different Error Levels (Adult Data Set)

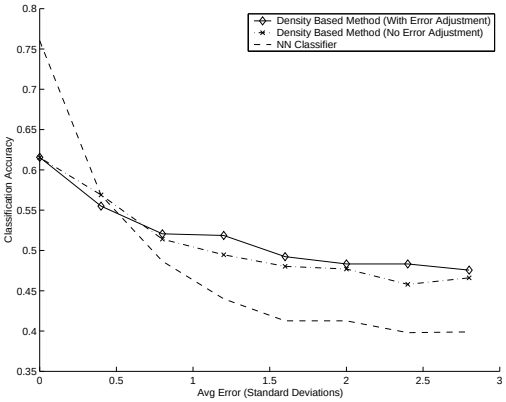

Figure 6. Error Based Classification for Different Error Levels (Forest Cover Data Set)

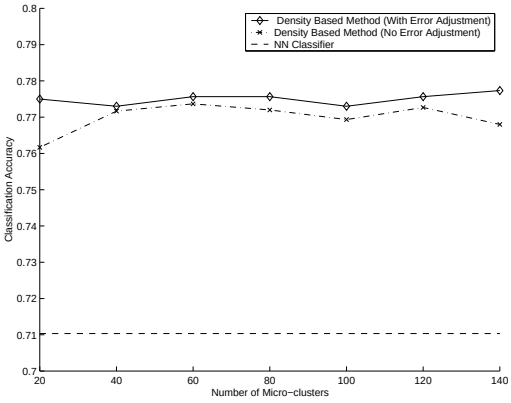

Figure 5. Error Based Classification for Different Number Of Clusters (Adult Data Set)

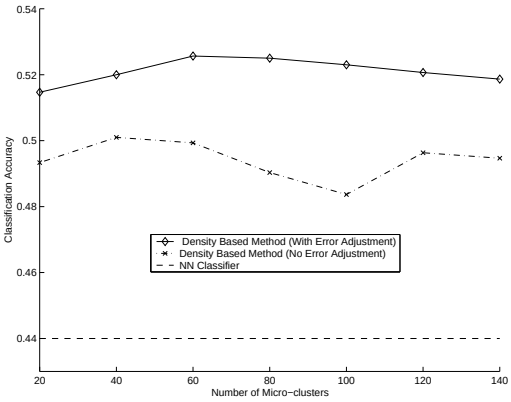

Figure 7. Error Based Classification for Different Number Of Clusters (Forest Cover Data Set)

4 Experimental Results

In this section, we will present the experimental results for the error-based mining approach. Our primary aim was to measure the effectiveness of the method in the presence of increasing error. In addition, we would like to check the effectiveness and efficiency of our micro-clustering strategy for data summarization. Therefore, we will present the following results: (1) Change in classification accuracy with increasing error rate. (2) Change in classification accuracy with increasing number of micro-clusters. (3) Efficiency of training and testing process with increasing number of micro-clusters (4) Efficiency of training and testing process with increasing data size and dimensionality.

All results were tested on an IBM T41p laptop running Windows XP with 1.60 GHz processor speed with 512 MB of main memory. The results were tested on a variety of data

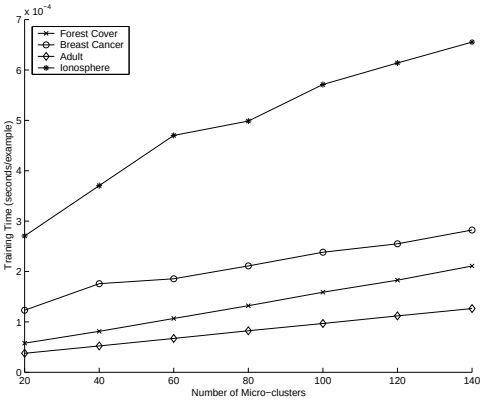

Figure 8. Training Time with Increasing Number Of Micro-clusters

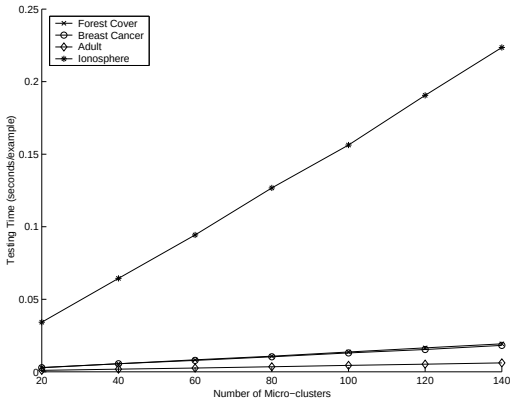

Figure 9. Testing Time with Increasing Number Of Micro-clusters

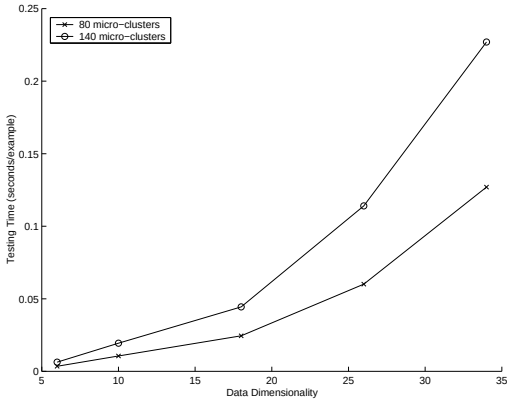

Figure 10. Testing Time with Increasing Data Dimensionality

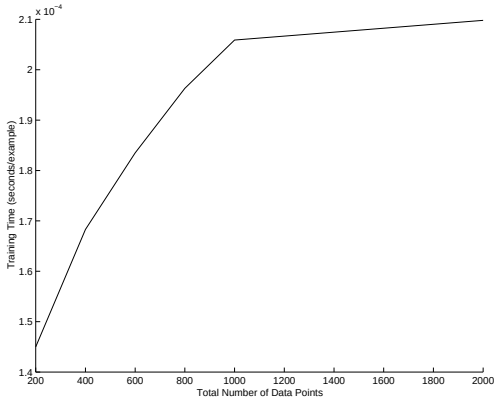

Figure 11. Training Rate with Increasing number of data points

sets from the UCI machine learning repository[1] in order to test the effects of varying data set sizes and dimensionalities. Specifically, we used the adult, ionosphere, wisconsin breast cancer, and forest cover data sets from the repository. In each case, we used only the quantitative variables for the testing process. We note that the forest cover and adult data sets were quite large, and were specially useful for testing the effects of micro-cluster summarization on the error-based classification process. For the purpose of experimentation, errors were added to the data set from a normal distribution with zero mean, and a standard deviation whose parameter was chosen as follows. For each entry, the standard deviation parameter of the normal distribution was chosen from a uniform distribution in the range $[0, 2 \cdot f] \cdot \sigma$, where σ is the standard deviation of that dimension in the underlying data. Thus, by changing the value of f, it is possible to vary the error level of the data set. We note that when the value of f is chosen to be 3, the majority of entries in the data are distorted by as much as 3 standard deviations of the original distribution. At this error level, the distorted value of the entry could lie outside the tail of the original distribution even at 99.99% level of confidence under the normal distribution assumption. In some of the scenarios discussed in this paper, this level of distortion reduces the effectiveness of a classifier to almost that of a random classifier, if the errors are not taken into account. In the testing process, we varied the value of the noise parameter f from 0 to 3 units to show the effects of a wide range of distortions.

In order to test the effectiveness of the method in the presence of errors, we used two other implementations: (1) A standard nearest neighbor classification algorithm which reported the class label of its nearest record. (2) A version of our density based classification algorithm which was not adjusted for errors. This was exactly the same algorithm as our error-based algorithm except that all the entries in the data were assumed to have an error of zero. The aim of using two different implementations was to account for differences which were obtained simply because of the use of a density based classifcation process. As our results will show, different classifiers are more effective for different data sets both in the presence and absence of errors, if no adjustments are made for the errors. In addition, we will show that the error based method was always superior to both classifiers in the presence of errors.

First, we will discuss the results from varying the value of the error parameter f. In these cases, we will fix the number of micro-clusters at 140. This effectively summarizes the entire data set in only 140 error-based points. This is a very high level of summarization, especially for data sets such as the forest cover set which contain more than half a million points. The purpose of choosing a high level of summarization was to illustrate the effectiveness

[1] http://www.ics.uci.edu/~mlearn

of the approach even in the case where the efficiency of the approach was considerably scaled up. In Figures 4, and 6, we have illustrated the classification accuracy with increasing error parameter f for the different data sets. The number of micro-clusters was fixed at 140. In each graph, the error parameter f is illustrated on the X-axis, and the classification accuracy is illustrated on the Y-axis. We make some observations about the common trends across different data sets:

(1) In the case of each classifier, an increase in the error parameter reduced the classification accuracy. We further note that the two density based classifiers had exactly the same accuracy when the error-parameter was zero. However, as the error parameter was increased, the differences between the two density based classifiers increased in each case. In addition, the accuracy of the nearest neighbor classifier worsened drastically with increasing error in all data sets.
(2) In the zero-error case, the nearest neighbor classifier could be better or worse than the density based classifier. For example, in the case of the forest cover data set (Figure 6), the nearest neighbor classifier is more effective than the density based classifier when there are no errors in the data set. However, an increased level of error reduced the effectiveness of the nearest neighbor classifier to a point where it was worse than both classifiers. In some of the data sets, an increase in error to an average of 3 standard deviations reduced the effectiveness of the nearest neighbor classifier to that of a random classifier. However, in each case, the error-based classifier continued to be effective even in the case of large errors in the data. Thus, the order of effectiveness of different classification methodologies could vary over different data sets, though the error adjustment process provides a definite advantage in terms of the quality of classification.
(3) One interesting characteristic of the error-based method is that even when the errors in the data are 2 to 3 times the standard deviation of the corresponding dimension, the classifier is still able to meaningfully classify the data at an accuracy significantly higher than that of a random classifier. As illustrated by our results, this is not the case for either the nearest neighbor or the (non-adjusted) density based classifier. In either case, the results of classification could be arbitrarily poor for data sets with high error.

The above results show the robustness of the approach to increasing levels of error in the data. Next, we study the effects of using different levels of micro-clustering on classification accuracy. A greater number of micro-clusters increases both the training and testing time, but it increases the level of granularity at which the data is represented. Naturally, a higher level of granularity leads to a greater level of accuracy, though the advantages are limited beyond a certain point. The results are also not completely monotonic because of the underlying noise in the data. The classification accuracy with increasing number of micro-clusters are illustrated in Figures 5, and 7. Since the nearest neighbor approach is not dependent on the micro-clustering process, its classification accuracy is illustrated as a horizontal baseline, which is independent of the number of micro-clusters. In each case, the value of the error parameter f was fixed at 1.2. On the X-axis, we have illustrated the number of micro-clusters, whereas the classification accuracy is illustrated in the Y-axis. We make the following observations:

(1) In each data set, the classification accuracy of the error-adjusted method increased with the number of micro-clusters. This is because the effectiveness of the summarization methodology improved with an increasing number of micro-clusters. However, the effects of the micro-clustering process seemed to level off after about 100 micro-clusters in each case.
(2) When error-adjustments were not applied to the density based classifier, the noise in the classification process exceeded the effects of increasing number of micro-clusters.

We also tested the efficiency of the training and testing process. In Figure 8, we have illustrated the training efficiency with increasing number of micro-clusters. The number of micro-clusters is illustrated on the X-axis, and the average time for processing each data point is illustrated on the Y-axis. It is clear that the processing time scaled linearly with the number of micro-clusters. The efficiency results for all data sets are illustrated in the same chart (Figure 8). We note that none of the data sets required more than $7 * 10^{-4}$ seconds per data point for processing. For some of the data sets, this translates to a rate of more than several thousand data points per second. The difference in running time among the different data sets is because of the varying dimensionality. This is because the number of operations per data point is proportional to data dimensionality. The running times for the adult data set are the least since it contains only 6 dimensions, whereas the 34-dimensional ionosphere data set takes the longest processing time per record.

In Figure 9, we have illustrated the testing time per record for the different data sets. On the X-axis, we have illustrated the number of micro-clusters, whereas the testing time per record is illustrated on the Y-axis. As in the previous case, the testing time is proportional to the number of micro-clusters. However, in this case, the difference in running times for the different data sets is much larger. This is because the testing time is more sensitive to data dimensionality. This trend can be observed more clearly from Figure 10, in which we have illustrated the running times for dif-

ferent data dimensionalities. We have illustrated two cases corresponding to 80 and 140 micro-clusters respectively. In order to derive the results of Figure 10, we used projections of different dimensionalities from the ionosphere data set. The dimensionality is illustrated on the X-axis, whereas the testing time per record is illustrated on the Y-axis. The trends in running time reflect the use of a roll-up algorithm during the knowledge discovery process. This results in a non-linear variation in the testing time with data dimensionality.

Finally, we tested the scalability of training times with data set size. We have illustrated the training times for the case of the forest cover data set using 140 micro-clusters. We used samples of different sizes from the forest cover data set, and calculated the training time per example. This training time per example is illustrated on the Y-axis of Figure 11, whereas the sample size is illustrated on the X-axis. The average training time per example was lower for smaller sample sizes. This is because the maximum number of micro-clusters were not used with very small sample sizes. At the earlier stages of the micro-clustering algorithm, only a small number of micro-clusters were created, but this gradually increased to the maximum number over time. Therefore, a fewer number of distance calculations were performed at the beginning of the algorithm. This resulted in a higher training rate at the beginning of the algorithm. However, as the sample size increased, the average training time per example stabilized to the steady-state rate of the maximum number of micro-clusters. This trend is illustrated in Figure 11. As evident from the trend, the training time per example is only of the order of 10^{-4} seconds. This efficiency is because of the use of the micro-clustering approach which allows efficient construction of the error-based densities. As discussed earlier in this section, the use of micro-cluster summarization in the design of error-based densities does not significantly reduce the accuracy of the overall approach. Thus, the error-based density estimation can be used for very large data sets without compromising effectiveness.

5 Conclusions and Discussion

In this paper, we discussed a general framework for uncertain data mining. We showed that the quality of data mining approaches can be substantially improved when the quantitative disposition of the errors in different fields are leveraged for the mining process. These errors can be captured in the intermediate representation of a density estimate of the data set. This intermediate representation can be used to construct accurate solutions for any data mining problem which works with a multi-variate estimation of the densities. We used a micro-clustering approach in order to scale our method to very large data sets. This increases the applicability of the error based procedure to very large number of data sets in which density estimates may need to be repeatedly computed over possibly different subsets of dimensions. As a concrete example of our technique, we constructed a classifier which uses error adjusted densities instead of the original data points. We tested our algorithm on a number of real data sets, and show that the error-adjusted approach turns out to be significantly more effective in practice. Given the ubiquity of uncertain data, we expect this method to have wide applicability for many future applications.

References

[1] C. C. Aggarwal, A. Hinneburg, D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. *ICDT Conference*, 2001.

[2] C. C. Aggarwal, J. Han, J. Wang, P. Yu. A Framework for Clustering Evolving Data Streams. *VLDB Conference*, 2003.

[3] R. Agrawal, R. Srikant. Privacy Preserving Data Mining. *ACM SIGMOD Conference*, 2000.

[4] R. Duda, P. Hart. *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

[5] M. Ester, H.-P. Kriegel, J. Sander, X. Xu: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, *KDD Conference*, 1996.

[6] D. Burdick, P. Deshpande, T. Jayram, R. Ramakrishnan, S. Vaithyanathan. OLAP Over Uncertain and Imprecise Data. *VLDB Conference*, 2005.

[7] M. James. Classification Algorithms, *Wiley*, 1985.

[8] M. Kumar, N. Patel, J. Woo. Clustering Seasonality Patterns in the presence of Errors. *KDD Conference*, 2002.

[9] H.-P. Kriegel, M. Pfeifle. Density-Based Clustering of Uncertain Data. *ACM KDD Conference*, 2005.

[10] R. Little, D. Rubin. Statistical Analysis with Missing Data Values. *Wiley Series in Prob. and Stats.*, 1987.

[11] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[12] T. Zhang, R. Ramakrishnan, M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *ACM SIGMOD Conference*, 1996.

[13] T. Zhang, R. Ramakrishnan, M. Livny. Fast Density Estimation Using CF-Kernel for Very Large Databases. *ACM KDD Conference*, 1999.