

# A Framework for Local Supervised Dimensionality Reduction of High Dimensional Data

Charu C. Aggarwal  
IBM T. J. Watson Research Center  
charu@us.ibm.com

## Abstract

High dimensional data presents a challenge to the classification problem because of the difficulty in modeling the precise relationship between the large number of feature variables and the class variable. In such cases, it may be desirable to reduce the information to a small number of dimensions in order to improve the accuracy and effectiveness of the classification process. While data reduction has been a well studied problem for the unsupervised domain, the technique has not been explored quite as extensively for the supervised case. Existing techniques which try to perform dimensionality reduction are too slow for practical use in the high dimensional case. These techniques try to find *global discriminants* in the data. However, the behavior of the data often varies considerably with data locality and different subspaces may show better discrimination in different localities. This is an even more challenging task than the global discrimination problem because of the additional issue of data localization. In this paper, we propose the novel idea of supervised subspace sampling in order to create a reduced representation of the data for classification applications in an efficient and effective way. The method exploits the natural distribution of the different classes in order to sample the best subspaces for class discrimination. Because of its sampling approach, the procedure is extremely fast and scales almost linearly both with data set size and dimensionality.

**Keywords:** classification, dimensionality reduction

## 1 Introduction

The classification problem is defined as follows: We have a set of records  $\mathcal{D}$  containing the training data, and each record is associated with a class label. The classification problem constructs a model which connects the training data to the class variables. The classification problem is a widely studied problem by the data mining, statistics and the machine learning communities [6, 8, 14]. In

this paper, we will explore the dimensionality reduction problem in the context of classification.

Dimensionality Reduction Methods have been widely studied in the unsupervised domain [9, 13, 15]. The idea in dimensionality reduction methods is to transform the data into a new orthonormal coordinate system in which the second order correlations are eliminated. In typical applications, the resulting axis-system has the property that the variance of the data along many of the new dimensions is very small [13]. These dimensions can then be eliminated, a process resulting in a compact representation of the data with some loss of representational accuracy.

Dimensionality reduction has been studied somewhat sparingly in the supervised domain. This is partially because the presence of class labels significantly complicates the reduction process. It is more important to find a new set of dimensions in which the new axis-system retains the discriminatory behavior of the data, whereas the maintenance of representational accuracy becomes secondary. Aside from the advantages of a compact representation after reduction, dimensionality reduction also serves the dual purpose of removing the irrelevant subspaces in the data. This improves the accuracy of the classifiers on the reduced data. Some techniques such as those discussed in [8] achieve this by repeated discriminant computation. This is extremely expensive for high dimensional databases.

Most data reduction methods in the supervised and unsupervised domain use *global data reduction*. In these techniques, a single axis system is constructed on which the entire data is projected. Such techniques assume uniformity of class behavior throughout the data set while computing the new representation. Recent research for the unsupervised domain has shown that different parts of the data show considerably different behavior. As a result, while global dimensionality reduction often fails to capture the important characteristics of the data in a small number of dimensions, local dimensionality reduction methods [2, 5] can often provide

a more effective solution. In these techniques, a different axis system is constructed for each data locality for more effective reduction.

For the supervised domain, the analogous intuition is that different parts of the data may show different patterns of discrimination. However, since even global reduction methods such as the Fisher method are so computationally intensive, the task of effective local reduction becomes even more intractable. In this paper, we will show that this task can actually be accomplished quite efficiently by using a sampling process in which the random process of subspace selection is biased by the underlying class distribution. We will also show that the reduction process is very useful for the classification problem itself, since it facilitates the development of some interesting decomposable classification algorithms. The overall result is a greatly improved classification process.

The technique of subspace sampling [1, 11] has recently been used to perform data reduction in the unsupervised version of the problem. In this paper, we propose an effective subspace sampling approach for supervised problems. The aim is to exploit the data distribution in such a way that the axis system of representation in each data locality is able to expose the class discrimination. Since the class discrimination can be modeled with a small number of dimensions in many parts of the data, the resulting representation is often more concise and effective for the classification process.

We will also show how existing classification algorithms can be enhanced by the local reduction approach described in this paper. Since our reduction approach decomposes the data set by optimizing the discrimination behavior in each segment, different classification techniques may vary in effectiveness on different parts of the data. This fact can be exploited in order to ensure that the particular classification model being used is best suited to its data locality. To the best of our knowledge, the *decomposable* method discussed in this paper which picks an optimal classifier depending upon locality specific properties of the compressed data is unique in its approach.

In order to facilitate further development of the ideas, we will introduce additional notations and definitions. We assume that the data set is denoted by  $\mathcal{D}$ . The number of points in the data set is denoted by  $N$  and the dimensionality by  $d$ . The full dimensional data space is denoted by  $\mathcal{U}$ . We define the  $l$ -dimensional hyperplane  $\mathcal{H}(\bar{y}, \mathcal{E})$  by an anchor  $\bar{y}$  and a mutually orthogonal set of vectors  $\mathcal{E} = \{\bar{e}_1 \dots \bar{e}_l\}$ . The hyperplane passes through  $\bar{y}$ , and the vectors in  $\mathcal{E}$  form the basis system for its subspace. The projection of a point  $\bar{x}$  onto this hyperplane is denoted by  $\mathcal{P}(\bar{x}, \bar{y}, \mathcal{E})$  and is the closest

approximation of  $\bar{x}$ , which lies on this hyperplane. In order to find the value of  $\mathcal{P}(\bar{x}, \bar{y}, \mathcal{E})$ , we use  $\bar{y}$  as the reference point for the computation. Specifically, we determine the projections of  $\bar{x} - \bar{y}$  onto the hyperplane defined by  $\{\bar{e}_1 \dots \bar{e}_l\}$ . Then, we translate the resulting point by the reference point  $\bar{y}$ . Therefore, we have:

$$(1.1) \quad \mathcal{P}(\bar{x}, \bar{y}, \mathcal{E}) = \bar{y} + \sum_{i=1}^l [(\bar{x} - \bar{y}) \cdot \bar{e}_i] \bar{e}_i$$

A pictorial representation of  $\bar{x}' = \mathcal{P}(\bar{x}, \bar{y}, \mathcal{E})$  is illustrated in Figure 1(a). The value of  $\bar{x}'$  can be represented in the orthonormal axis system for  $\mathcal{E}$  with the use of only  $l$  coordinates  $((\bar{x} - \bar{y}) \cdot \bar{e}_1 \dots (\bar{x} - \bar{y}) \cdot \bar{e}_l)$ . This results in an additional overhead of storing  $\bar{y}$  and  $\mathcal{E}$ . This storage overhead is however not significant, if it can be averaged over a large number of points stored on this hyperplane. While the error of approximating  $\bar{x}$  with  $\mathcal{P}(\bar{x}, \bar{y}, \mathcal{E})$  is given by the euclidean distance between  $\bar{x}$  and  $\mathcal{P}(\bar{x}, \bar{y}, \mathcal{E})$ , this measure is secondary to the classification accuracy of the reduced data.

This paper is organized as follows. In the next section, we will introduce the supervised subspace sampling technique and discuss some of its properties. Section 3 will discuss the application of the method to the classification problem. We will also discuss how the performance of the classification system can be considerably optimized by using the decomposition created by the supervised subspace sampling technique. The empirical results are discussed in section 4. Finally, we present the conclusions and summary in section 5.

**1.1 Contributions of this paper** The paper discusses a highly effective and scalable approach to the problem of supervised data reduction. While unsupervised data reduction has been well studied in the literature, the supervised problem is extremely difficult in practice because of the need to use the class distributions in the reduction process. The available discriminant methods are highly computationally intensive even on memory resident databases. In contrast, the technique discussed in this paper provides a significantly more effective reduction process, while exhibiting linear scalability with data set size and dimensionality because of its sampling approach. The process of sampling subspaces which are optimized to particular data localities results in a technique in which each segment of the data is more suited to the classification task. Furthermore, the unique decomposition created by the reduction process facilitates the creation of optimized decomposable approaches to the classification problem. Thus, the overall approach not only provides savings in terms of data compression, but also a greatly improved classification process.

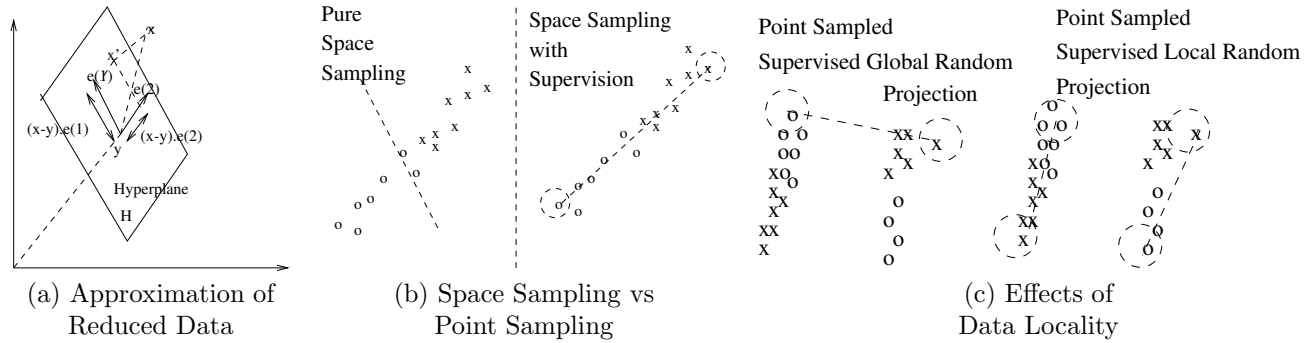


Figure 1: Illustration of Localized Sampling

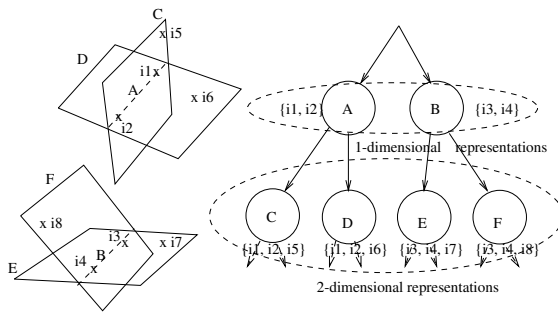


Figure 2: Subspace Tree (Example)

## 2 Supervised Subspace Sampling

An interesting approach for dimensionality reduction of the unsupervised version of the problem is that of random projections [11, 12]. In this class of techniques, we repeatedly sample spherically symmetric random directions in order to determine an optimum hyperplane on which the data is projected. These methods can also be extended to the classification problem by projection of the data onto random subspaces and measuring the discrimination of such spaces. However, such a direct extension of the random projection technique [11, 12] may often turn out to be ineffective in practice, since it is blind both to the data and class distributions of the points. We will try to explain this point by using 1-dimensional projections of 2-dimensional data. Consider the data set illustrated in Figure 1(b) in which we have illustrated two kinds of projections. In the left figure, the data *space* is sampled in order to find a 1-dimensional line along which the projection is performed. In data space sampling, random projections are chosen in a spherically symmetric fashion irrespective of the data distribution. The reduced data in this 1-dimensional representation is simply the projection of the data points onto the line. We note that such a pro-

jection neither follows the basic pattern in the data, nor does it provide a direction along which the class distributions are well discriminated. For high dimensional cases, such a projection may be poor at distinguishing the different classes even after repeated subspace sampling. In the other case of Figure 1(b), we have sampled the *points* in order to create a random projection. The sampled subspace is defined as the  $(l - 1)$ -dimensional hyperplane containing  $l$  (locally proximate) points (of different classes) from the data.<sup>1</sup> The reason for picking points from different classes is that we would like the resulting subspace to represent the discrimination behavior of different classes more effectively. At the same time these points should be picked carefully only from a local segment of the data in order to ensure that the class discrimination is determined by data locality. For example, in Figure 1(b), the 1-dimensional line obtained by sampling two points of different classes picks the direction of greater discrimination most effectively than the space sampled projection in the same figure.

While it is intuitively clear that point sampling is more effective than space sampling for variance preservation, the advantages are limited when the data distribution varies considerably with locality. For example, in Figure 1(c), even the optimal 1-dimensional random projection cannot represent all points without losing a substantial amount of class discrimination. In fact, there is no 1-dimensional line along which a projection can effectively separate out the classes. In Figure 1(c), we have used the random projection technique locally in conjunction with data partitioning. In this technique, each data point is projected on the closest of a number of point sampled hyperplanes. In this case, it is evident that the data points are often well discriminated along each of the sampled hyperplanes, while they may be

<sup>1</sup>The actual methodology of choosing the points is discussed at a later stage. At this point, we are concerned only with choosing the points in such a way that the chosen subspace is naturally biased by the original data distribution.

poorly discriminated at a global level. This is because the nature of the class distribution varies considerably with data locality, and a global dimensionality reduction method cannot reduce the representation below the original set of two dimensions. On the other hand, the 1-dimensional representation of the data created by local projection of the data points along the sampled lines represents the class discrimination very well.

It should be noted that the improvements of the localized subspace sampling technique come at the additional storage costs of the different hyperplanes. This limits the number of hyperplanes which can be retained from the sampling process, and requires us to make judicious choices in picking these hyperplanes. A second important issue is that even the implicit dimensionalities of the different data localities may be different. Therefore, we need a mechanism by which the sampling process is able to effectively choose hyperplanes of the lowest possible dimensionality for each data locality. This is an issue which we will discuss after developing some additional notational machinery:

**DEFINITION 1.** *Let  $P = (\overline{x_1} \dots \overline{x_{l+1}})$  be a set of  $(l + 1)$  linearly independent points. The representative hyperplane  $\mathcal{R}(P)$  of  $P$  is defined as the  $l$ -dimensional hyperplane which passes through each of these  $(l + 1)$  points.*

The hyperplane  $R(P)$  can also be represented with the use of any point  $\overline{y}$  on the hyperplane, and an orthonormal set of vectors  $\mathcal{E} = \{\overline{e_1} \dots \overline{e_l}\}$ , which lie on the hyperplane. We shall call  $(\overline{y}, \mathcal{E})$  the *axis* representation of the hyperplane, whereas the set  $P$  is referred to as the *point* representation. Thus,  $\mathcal{R}(P)$  (point representation) is the same as  $\mathcal{H}(\overline{y}, \mathcal{E})$  (axis representation). We note that there can be infinitely many point or axis representations of the same hyperplane. The axis representation is more useful for performing distance computations of the hyperplane from individual points in the database, whereas the point representation has advantages in storage efficiency in the context of a hierarchical arrangement of subspaces. We will discuss this issue in a later section.

**2.1 The Supervised Subspace Tree** The Supervised Subspace Tree is a conceptual organization of subspaces used in the data reduction process. This conceptual organization imposes a hierarchical arrangement of the subspaces of different dimensionalities. Each such subspace provides effective data discrimination which is specific to a particular locality of the data. Since the subspaces have different data dimensionalities, this results in a variable dimensionality decomposition of the data. The nodes at level- $m$  in the subspace tree correspond to  $m$ -dimensional subspaces. The root node

corresponds to the null subspace. Thus, the dimensionality of the hyperplane for any node in the tree is determined by its depth. The subspace at a node is hierarchically related to that of its immediate parent. Each subspace other than the null subspace at the root is a 1-dimensional extension of its parent hyperplane. This 1-dimensional extension is obtained by adding a sampled data point to the representative set of the parent hyperplane. In order to elucidate the concept of a subspace tree, we will use an example. In Figure 2, we have illustrated a hierarchically arranged set of subspaces. The figure contains a two-level tree structure which corresponds to 1- and 2-dimensional subspaces. For each level-1 node in the tree, we store two points which correspond to the 1-dimensional line for that node. For each lower level node, we store an additional data point which increases the dimensionality of its parent subspace by 1. Therefore, a level- $m$  node has a representative set of cardinality  $(m + 1)$ . For example, in the case of Figure 2, the node  $A$  in the subspace tree (with representative set  $\{i_1, i_2\}$ ) corresponds to the 1-dimensional line defined by  $\{i_1, i_2\}$ . This node is extended to a 2-dimensional hyperplane in two possible ways corresponding to the nodes  $C$  and  $D$ . In each case, an extra point needs to be added to the representative set for creating the 1-dimensional extension. In order to extend to the 2-dimensional hyperplane for node  $C$ , we use the point  $i_5$ , whereas in order to extend to the hyperplane for node  $D$ , we use the point  $i_6$ . Note from Figure 2(a) that the intersection of the 2-dimensional hyperplanes  $C$  and  $D$  is the 1-dimensional line  $A$ . Thus, each node in the subspace tree corresponds to a hyperplane which is defined by its representative set drawn from the database  $\mathcal{D}$ . The representative set for a given hyperplane is obtained by adding one point to the representative set of its immediate parent. The subspace tree is formally defined as follows:

**DEFINITION 2.** *The subspace tree is a hierarchical arrangement of subspaces with the following properties: (1) Nodes at level- $m$  correspond to  $m$ -dimensional hyperplanes (2) Nodes at level- $(m + 1)$  correspond to 1-dimensional extensions of their parent hyperplanes at level- $m$ . (3) The point representative set of a level- $(m + 1)$  node is obtained by adding a sampled data point to the representative set of its  $m$ -dimensional parent subspace.*

The data points in  $\mathcal{D}$  are partitioned among the different nodes of the subspace tree. We note that since the hyperplane is a subspace of the full dimensional space, it has a lower dimensional axis system in terms of which the coordinates of  $\overline{x}$  are represented. Since the dimensionality of a hyperplane depends directly on the

distance of the node to the root, higher levels of the tree provide greater advantages in the reduction process.

**2.2 Subspace Tree Construction** Each node of the subspace tree corresponds to a hyperplane defined by the sequence of representative points sampled, starting from the root up to that node. The terms hyperplane and node are therefore used interchangeably through this paper.

In order to measure the quality of a given node  $N$  for the classification process, a *discrimination index*  $\beta(N)$  is maintained along with each node. This discrimination index  $\beta(N)$  always lies between 0 and 1, and is a measure of how well the different classes are separated out in the data set for node  $N$ . A value of 1 indicates perfect discrimination among the classes, whereas a value of 0 indicates very poor discrimination. We will discuss the methodology for computation of the discriminant in a later section.

The input to the subspace sampling algorithm for tree construction is the compression tolerance parameter  $\epsilon$ , the data set  $\mathcal{D}$ , the maximum number of nodes  $L$ , a discrimination tolerance  $\gamma_1$ , and a discrimination target  $\gamma_2$ . The value of  $\gamma_2$  is always larger than  $\gamma_1$ . Each of these discrimination thresholds lie between 0 and 1. Intuitively, these discrimination thresholds impose a minimum and maximum threshold on the quality of class separation classes in the individual nodes. Correspondingly, each node  $N$  is classified into one of three types which is recorded by a variable called the *Status*( $\cdot$ ) vector: (1) *Default Node*: In this case, the discrimination index  $\beta(N)$  lies between  $\gamma_1$  and  $\gamma_2$ . The value of *Status*( $N$ ) is set to 0. (2) *Discriminative Node*: Such a node is good for the classification process. The discrimination index  $\beta(N)$  is larger than  $\gamma_2$  in this case. the variable *Status*( $N$ ) is set to 2. (3) *Forbidden Node*: Such a node is bad for the classification process. In this case, the discriminant index  $\beta(N)$  is smaller than  $\gamma_1$ . The value of *Status*( $N$ ) is set to 1.

A top-down algorithm is used to construct the nodes of the subspace tree, and the data set  $\mathcal{D}$  is partitioned along this hierarchy in order to maximize the localized discrimination during the dimensionality reduction process. A *discrimination index*  $\beta(N)$  is maintained with each node  $N$  in the tree. At each stage of the algorithm, every node  $N$  in the subspace tree has a set of *descendent assignments*  $\mathcal{T}(N) \subseteq \mathcal{D}$  from the database  $\mathcal{D}$ . These are the data points which will be assigned to one of the descendants of node  $N$  during the tree construction process, but not to node  $N$  itself. In addition, each node also has a set of *direct assignments*  $\mathcal{Q}(N)$ , which are data points that are reduced onto node  $N$ . A data point becomes a direct assignment of node  $N$ , when

one of the following two properties is satisfied: (1) The data point is at most a distance of  $\epsilon$  from the hyperplane corresponding to node  $N$  and the discrimination factor  $\beta(N)$  for the node is larger than  $\gamma_1$ . (2) The discrimination factor  $\beta(N)$  for the node is larger than  $\gamma_2$ . All assignments of the node which are not direct assignments automatically become descendent assignments. In each iteration, the descendent assignments  $\mathcal{T}(N)$  of the nodes at a given level of the tree are partitioned further into at most  $k_{max}$  children of node  $N$ . This partitioning is based on the distance of the data points to the hyperplanes corresponding to the  $k_{max}$  children of  $N$ . Specifically, each data point is assigned to the hyperplane from which it has the least distance. The assigned points are then classified either as a descendent or direct assignments depending upon the distance from the hyperplane and corresponding discrimination index. As noted earlier, the latter value determines whether a node is a default node, a discriminative node, or forbidden node. Forbidden nodes do not have any direct assignments and therefore all data points at forbidden nodes automatically become descendent assignments. On the other hand, in the case of a discriminative node, the reverse is true and all points become direct assignments. For the case of default nodes, a point becomes a direct assignment only if it is at a distance of at most  $\epsilon$  from the corresponding hyperplane. This process continues until each data point becomes the direct assignment of some node, or is identified in the anomaly set. The overall algorithm for subspace tree construction is illustrated in Figure 3.

A levelwise algorithm is used during the tree construction phase. The reason for this levelwise approach is that the database operations during the construction of a given level of nodes can be consolidated into a single database pass. The actual construction of the  $m$ th level is achieved by sampling one representative point for each of the  $k_{max}$  children of the level- $(m-1)$  nodes. This representative point is added in order to create the corresponding 1-dimensional extension. These  $k_{max}$  representative points are sampled from the local segment  $\mathcal{T}(N)$  of the database. Picking these representative points at a node  $N$  is tricky, since we would like to ensure that they satisfy the following two properties: (1) The points represent the behavior of localized regions in the data. (2) The points are sufficiently representative of the different classes in that data locality. In order to achieve this, we would like to ensure that the set  $\mathcal{R}(N)$  represents as many different classes as possible. Since the representative extensions at a node  $N$  are sampled from the local segment  $\mathcal{T}(N)$  only, the first property is satisfied. In order to satisfy the second property, we choose the class from which the data points are sampled

**Algorithm** *SampleSubspaceTree*(CompressionTolerance:  $\epsilon$ , MaximumTreeDegree:  $k_{max}$ , Database:  $\mathcal{D}$ , Node Limit:  $L$ , Discrimination Tolerance:  $\gamma_1$ , Discrimination Target:  $\gamma_2$ )

**begin**

Sample  $2 * k_{max} * sampfactor$  points from  $\mathcal{D}$  and pair up points randomly to create

$k_{max} * sampfactor$  1-dim. point representative hyperplanes (lines) denoted by  $\mathcal{S}$ ;

$(\mathcal{S}, \beta(S_1) \dots \beta(S_{k_{max}})) = SelectSubspaces(\mathcal{S}, k_{max})$ ;

$(\mathcal{T}(S_1), \dots, \mathcal{T}(S_{k_{max}}), \mathcal{Q}(S_1), \dots, \mathcal{Q}(S_{k_{max}})) = PartitionData(\mathcal{D}, \mathcal{S})$ ;

**for**  $i = 1$  **to**  $k_{max}$  **do** **if**  $\beta(S_i) \geq \gamma_2$  **then** { Discriminative Node }  $\mathcal{Q}(S_i) = \mathcal{Q}(S_i) \cup \mathcal{T}(S_i)$ ;  $\mathcal{T}(S_i) = \phi$ ;  $Status(S_i) = 2$ ;

**else if**  $\beta(S_i) < \gamma_1$  **then** { Forbidden Node }  $\mathcal{T}(S_i) = \mathcal{T}(S_i) \cup \mathcal{Q}(S_i)$ ;  $\mathcal{Q}(S_i) = \phi$ ;  $Status(S_i) = 1$ ;

**else**  $Status(S_i) = 0$ ;

$\mathcal{S} = DeleteNodes(S_1 \dots S_{k_{max}}, min-thresh)$ ; {  $\mathcal{L}_m$  is the set of level- $m$  nodes }

$m = 1$ ;  $\mathcal{L}_1 = \mathcal{S}$ ; { Each hyperplane (line) in  $\mathcal{S}$  is the child of *Root* };

**while** ( $\mathcal{L}_m \neq \{\}$ ) and (less than  $L$  nodes have been generated) **do** **begin**

**for** each non-null level- $m$  node  $R \in \mathcal{L}_m$  **do** **begin**

Sample  $k_{max} * sampfactor$  points from  $\mathcal{T}(R)$ ;

Extend the node  $R$  by each of these  $k_{max} * sampfactor$  points (in turn) to create the  $k_{max} * sampfactor$  corresponding  $(m + 1)$ -dimensional hyperplanes denoted by  $\mathcal{S}$ ;

$(\mathcal{S}, \beta(S_1) \dots \beta(S_{k_{max}})) = SelectSubspaces(\mathcal{S}, k_{max})$ ;

$(\mathcal{T}(S_1), \dots, \mathcal{T}(S_{k_{max}}), \mathcal{Q}(S_1), \dots, \mathcal{Q}(S_{k_{max}})) = PartitionData(\mathcal{T}(R), \mathcal{S})$ ;

**for**  $i = 1$  **to**  $k_{max}$  **do** **if**  $\beta(S_i) \geq \gamma_2$  **then** { Discriminative Node }  $\mathcal{Q}(S_i) = \mathcal{Q}(S_i) \cup \mathcal{T}(S_i)$ ;  $\mathcal{T}(S_i) = \phi$ ;  $Status(S_i) = 2$ ;

**else if**  $\beta(S_i) < \gamma_1$  **then** { Forbidden Node }  $\mathcal{T}(S_i) = \mathcal{T}(S_i) \cup \mathcal{Q}(S_i)$ ;  $\mathcal{Q}(S_i) = \phi$ ;  $Status(S_i) = 1$ ;

**else**  $Status(S_i) = 0$ ;

$\mathcal{S} = DeleteNodes(S_1 \dots S_{k_{max}}, min-thresh)$ ; { Thus  $\mathcal{S}$  contains at most  $k_{max}$  children of  $R$  }

$\mathcal{L}_{m+1} = \mathcal{L}_{m+1} \cup \mathcal{S}$ ;  $m = m + 1$ ;

**end**;

**end**;

**end**

Figure 3: Supervised Subspace Tree Construction

as follows: Let  $f_1^R \dots f_k^R$  be the fractional class distributions in  $\mathcal{R}(N)$  and  $f_1^T \dots f_k^T$  be the fractional class distributions in  $\mathcal{T}(N)$ . We sample a point belonging to the class  $i$  in  $\mathcal{T}(N)$  which always belongs to the class with the least value of  $f_i^R / f_i^T$ . Thus, this process picks the point from the class which is most under-represented in  $\mathcal{R}(N)$  relative to  $\mathcal{T}(N)$ .

A total of  $k_{max} * sampfactor$  points (belonging to the selected class) are picked for extension of the nodes from level- $(m - 1)$  to level- $m$ . Thus, a total of  $k_{max} * sampfactor$   $m$ -dimensional hyperplanes can be generated by combining the representative set  $\mathcal{R}(N)$  of node  $N$  with each of these sampled points. The purpose of oversampling by a factor of  $sampfactor$  is to increase the effectiveness of the final children subspaces which are picked. The larger the value of  $sampfactor$ , the better the sampled subspaces, but the greater the computational requirement. Next, the procedure *SelectSubspaces* picks  $k_{max}$  hyperplanes out of these  $k_{max} * sampfactor$  possibilities so that the different classes are as well separated as possible. The first task is to partition the  $k_{max} * sampfactor$  hyperplanes into  $sampfactor$  sets of  $k_{max}$  hyperplanes. We will pick one of these partitions depending upon the quality of the assignment. In order to achieve this, the distance of the data point  $\bar{x}$  to each of the  $k_{max} * sampfactor$

hyperplanes is determined. For each of the  $sampfactor$  sets of hyperplanes, we assign the data point  $\bar{x}$  to the closest hyperplane from that partition. This results in a total of  $sampfactor$  possible assignments of the data points. The quality of the assignment depends upon how well the different classes are discriminated from one another in the resulting localized data sets. The *SelectSubspaces* procedure quantifies this separation in terms of the discrimination index  $\beta(\cdot)$  for each of these nodes. The average discrimination index for each of the  $sampfactor$  sets of nodes is calculated. The set of  $k_{max}$  hyperplanes with the smallest average discrimination index is chosen for the purpose of reduction. These hyperplanes are returned as the set  $\mathcal{S} = (S_1 \dots S_{k_{max}})$ . In addition, the discrimination index of each of these nodes is returned as  $(\beta_1 \dots \beta_{k_{max}})$ .

Once the hyperplanes which form the optimal extensions of node  $N$  have been determined, each point in  $\mathcal{T}(N)$  is re-assigned to one of the children of  $N$  by the use of the procedure *PartitionData*. Specifically, each point in  $\mathcal{T}(N)$  is assigned to the child node to which it is the least distance. Furthermore, the assigned nodes are classified into direct assignments  $\mathcal{T}(S_i)$  and descendent assignments  $\mathcal{Q}(S_i)$ . Initially, the *PartitionData* procedure returns the direct assignments  $\mathcal{T}(S_i)$  and descendent assignments  $\mathcal{Q}(S_i)$  using only the distance of the

data points from the corresponding hyperplanes. Specifically, the *PartitionData* procedure returns a point as a direct assignment, if it is at a distance of at most  $\epsilon$  from the corresponding hyperplane. Otherwise, it returns the data point as a descendent assignment. After application of the *PartitionData* procedure, we further re-adjust the direct and descendent assignments using the discrimination levels  $\beta(S_i)$  of each child node  $S_i$ . If the node  $S_i$  is a forbidden node, then the direct assignment set  $\mathcal{Q}(S_i)$  is reset to null, and all points become descendent assignments. Therefore  $\mathcal{Q}(S_i)$  is added to  $\mathcal{T}(S_i)$ . The reverse is true when the node is a discriminative node. In that case, all points become direct assignments and  $\mathcal{T}(S_i)$  is set to null.

Nodes which have too few points assigned to them are not useful for the dimensionality reduction process. Such nodes are deleted by the procedure *DeleteNodes*. The corresponding data points are considered exceptions which are stored separately by the algorithm.

The first iteration of the algorithm ( $m = 1$ ) is special in which we sample  $2 * k_{max} * sampfactor$  points in order to create the initial set of  $k_{max} * sampfactor$  lines. Thus, the only difference is that twice the number of points need to be sampled in order to create the 1-dimensional hyperplanes used by the algorithm. The procedure for selection of these points is exactly similar to that of the general case. The other procedures such as selection and deletion of subspaces and data partitioning are also the same as in the general case.

In order to ease conceptual abstraction, we have presented the *PartitionData* and *SelectSubspaces* procedures separately for each node. In the actual implementation, this procedure is executed simultaneously for all nodes at a given level in one scan. Similarly, the process of picking the best hyperplanes for all nodes at a given level is executed simultaneously in a single scan of the data.

The process of levelwise tree construction continues until no node in the current level can be extended any further, or the maximum limit  $L$  for the number of nodes has been reached. We would like this limit  $L$  to be determined by main memory limitations, since it will be found useful in applying it effectively during the classification process. For our implementation, we used a conservative limit of only  $L = 10,000$  nodes, which was well within current main memory limitations for even 1000-dimensional data sets.

Each of the procedures *SelectSubspaces* and *PartitionData* require the computation of distances of data points  $\bar{x}$  to the representative hyperplanes. In order to perform these distance computations, the axis representations of the hyperplanes need to be determined. A hyperplane node  $N$  at level- $m$  is only implicitly de-

defined by the  $(m + 1)$  data points  $\{\bar{z}_1 \dots \bar{z}_{m+1}\}$  stored at the nodes along the path from the root to  $N$ . The next tricky issue is to compute the axis representation  $(\bar{y}, \mathcal{E} = \{\bar{e}_1 \dots \bar{e}_m\})$  of the points  $\{\bar{z}_1 \dots \bar{z}_{m+1}\}$  efficiently in a way that can be replicated exactly at the time of data reconstruction. This is especially important, since there can be an infinite number of axis representations of the same hyperplane, but the projection coordinates are computed only with respect to a particular axis-representation. The corresponding representation  $(\bar{y}, \mathcal{E} = \{\bar{e}_1 \dots \bar{e}_m\})$  is computed as follows:

We first set  $\bar{y} = \bar{z}_1$  and  $\bar{e}_1 = (\bar{z}_2 - \bar{z}_1) / \|\bar{z}_2 - \bar{z}_1\|$ . Next, we iteratively compute  $\bar{e}_i$  from  $\bar{e}_1 \dots \bar{e}_{i-1}$  as follows:

$$(2.2) \quad \bar{e}_i = \frac{\bar{z}_{i+1} - \bar{z}_1 - \sum_{j=1}^{i-1} [(\bar{z}_{i+1} - \bar{z}_1) \cdot \bar{e}_j] \bar{e}_j}{\|\bar{z}_{i+1} - \bar{z}_1 - \sum_{j=1}^{i-1} [(\bar{z}_{i+1} - \bar{z}_1) \cdot \bar{e}_j] \bar{e}_j\|}$$

Equation 2.2 is essentially the iteration for the Gram-Schmidt orthogonalization process [10]. The following observation is a direct consequence of this fact:

**OBSERVATION 2.1.** *The set  $(\bar{z}_1, \mathcal{E})$  generated by Equation 2.2 is an axis representation of the hyperplane  $\mathcal{R}(\bar{z}_1 \dots \bar{z}_{m+1})$ .*

Many axis representations can be generated using Equation 2.2 for the same hyperplane  $\mathcal{R}(\{\bar{z}_1 \dots \bar{z}_{m+1}\})$  depending upon the ordering of  $\{\bar{z}_1 \dots \bar{z}_{m+1}\}$ . Since we need to convert from point representations to axis representations in a consistent way for both data reduction and reconstruction, this ordering needs to be fixed in advance. For the purpose of this paper, we will assume that the point ordering is always the same as one in which it was sampled during the top-down tree construction process. This leads to representative points sampled at higher levels of the tree to be ordered first, and points at lower levels to be ordered last. The only ambiguity is for the level-1 nodes at which 2 points are stored instead of one. In that case, the record which is lexicographically smaller is ordered earlier. We shall refer to this particular convention for axis representation as the *path-ordered axis representation*.

**2.3 Computation of Discrimination Index** Several methods can be used in order to calculate the discrimination index of the reduced data at a given node. The most popularly used method is the Fisher's linear discriminant [7] of the data points in the node. This discriminant minimizes the ratio of the intra-class distance to the inter-class distance. Our approach however uses a nearest neighbor discriminant. In this technique, we find the nearest neighbor to each data point in the dimensionality reduced database, and calculate the fraction of data points which share the same class as their

nearest neighbor. This fraction is reported as the discrimination index.

**2.4 Storage of Compressed Representation** The storage algorithm needs to store the tree structure and the set of points associated with each node of the tree. Each of these components are stored as follows:

(1) The Subspace Tree: If the axis-systems are explicitly stored at each node, the storage requirements for the tree structure could be considerable. This is because the axis representation  $(\bar{y}, \mathcal{E})$  for an  $m$ -dimensional node requires  $m+1$  orthonormal vectors (including the origin  $\bar{y}$  for the corresponding hyperplane). However, it turns out that we do not need to store the axis representations explicitly, For each level- $m$  node, we only maintain the additional data point which increases the dimensionality of the corresponding subspace by one. In addition, we need to maintain the identity of the node, its immediate parent and the status of the node (corresponding to whether it is forbidden, discriminative, or a default node). Thus, a total of  $(d+3)$  values are required for each node. For the (at most  $k_{max}$ ) level-1 nodes, the storage requires  $(2 \cdot d + 3)$  values since we need to maintain the *two* points which define the sampled line in lexicographic ordering.

The subspaces in the tree structure are thus only *implicitly* defined by the sequence of points from the root to that node. Thus, by using this method, we require only one vector for an  $m$ -dimensional node rather than than  $O(m)$  vectors. Since most nodes in the tree are at lower levels, such savings add up considerably over the entire tree structure. The reason for this storage efficiency is the implicit representation of the subspace tree. This results in the reuse of the vector stored at a given node for all descendents of that node.

(2) The Reduced Database: Each data point is either an exception or is associated with one node in the tree. We store the identity of the node for which it is a direct assignment. In addition, we maintain the coordinates of the data point for the axis representation  $(\bar{y}, \mathcal{E})$  of this hyperplane in accordance with Equation 1.1. The projection coordinates of  $\bar{x}$  on  $(\bar{y}, \mathcal{E})$  are given by  $(c_1 \dots c_m) = \{\bar{e}_1 \cdot (\bar{x} - \bar{y}) \dots \bar{e}_m \cdot (\bar{x} - \bar{y})\}$ . The class labels were stored separately.

**2.5 Reconstruction Algorithm** Since the subspace tree is represented in an implicit format, the axis representations of the nodes need to be reconstructed. It is possible to do this efficiently because of the use of the path-ordered axis convention for representation. Since there are a large number of nodes, it would seem that the initial phase could be quite expensive to perform for each node. However, it turns out that because of the use

**Algorithm *SelectClassifierModels***(Subspace Tree:  $ST$ )  
**begin**

**for** each node  $N$  in the subspace tree  $ST$  **do**  
**begin**  
Divide the data set  $\mathcal{T}(N)$  into two parts  
 $\mathcal{T}_1(N)$  and  $\mathcal{T}_2(N)$  with ratio  $r : 1$ ;  
Use classification training algorithms  $\mathcal{A}_1 \dots \mathcal{A}_m$   
on  $\mathcal{T}_1(N)$  to create models  $\mathcal{M}_1(N) \dots \mathcal{M}_m(N)$ ;  
Compute accuracy of models  $\mathcal{M}_1(N) \dots \mathcal{M}_m(N)$   
on  $\mathcal{T}_2(N)$ ;  
Pick model with highest classification accuracy on  $\mathcal{T}_2(N)$   
and denote as  $\mathcal{CL}(N)$ ;

**end;**  
**end**

Figure 4: Node Specific Classification

of the path-ordered convention for axis-representations, this can be achieved in a time complexity which requires the computation of only one axis per node. The trick is to construct the axis representations of the nodes in the tree in a top-down fashion. This is because the Equation 2.2 computes the axis representation  $\{\bar{e}_1 \dots \bar{e}_i\}$  of a node by using the axis representation  $\{\bar{e}_1 \dots \bar{e}_{i-1}\}$  of its parent and the point  $\bar{z}$  stored at that node. (For the nodes at level-1, lexicographic ordering of the representative points is assumed.) It is easy to verify that this order of processing results in the path-ordered axis representation.

Once the axis representations of the nodes have been constructed, it is simple to perform the necessary axis transformations which represent the reconstructed database in terms of the original attributes. Recall that for each database point  $\bar{x}$ , the identity of the corresponding node is also stored along with it. Let  $(\bar{y}, \mathcal{E})$  be the corresponding hyperplane and  $(c_1 \dots c_m) = \{\bar{e}_1 \cdot (\bar{x} - \bar{y}) \dots \bar{e}_m \cdot (\bar{x} - \bar{y})\}$  be the coordinates of  $\bar{x}$  along this  $m$ -dimensional axis representation. Then, as evident from Equation 1.1, the reconstructed point  $\bar{x}'$  is given by  $\bar{x}' = \bar{y} + \sum_{i=1}^m [c_i] \bar{e}_i$

### 3 Effective Classification by Locality Specific Decomposition

It turns out that the locality specific decomposition approach of the dimensionality reduction problem not only provides a reduced representation of the data, but can also be leveraged for effective classification. This is because the dimensionality reduction process decomposes the data into a number of different parts each of which have a unique and distinctive class distribution. The use of different training phases on different nodes may be quite effective in these cases. In fact, the fundamental variation in the dimensionalities and data characteristics of the different nodes makes







