The Generalized Dimensionality Reduction Problem

Charu C. Aggarwal*

Abstract

The dimensionality reduction problem has been widely studied in the database literature because of its application for concise data representation in a variety of database applications. The main focus in dimensionality reduction is to represent the data in a smaller number of dimensions that the least amount of information is lost. In this paper, we study the dimensionality reduction problem from an entirely different perspective. We discuss methods to find a representation of the data so that a user-defined objective function is optimized. For example, we may desire to find a reduction of the data in which a particular kind of classifier works effectively. Another example (relevant to the similarity search domain) would be a reduction in which the cluster of k closest points provides the best distance based separation from the remaining data set. We discuss a general abstraction for the problem and provide the broad framework of an evolutionary algorithm which solves this abstraction. We test our framework on two separate instantiations of this framework and provide results illustrating the effectiveness and efficiency of our method.

1 Introduction

The dimensionality reduction problem is traditionally defined as a method to represent the data in a small number of features with the least loss in information. Specifically, for a given database $\mathcal{D}$, we would like to find an orthonormal set of vectors $\mathcal{V}$, so that when the database $\mathcal{D}$ is projected onto the subspace represented by $\mathcal{V}$, the total amount of variance of the projected database $\mathcal{D}_v$ is as large as possible. Such a transformation of the data is useful in a number of content based retrieval applications. This is because the distances between pairs of points are approximately preserved by the transformation. A number of applications such as multimedia and text search, collaborative filtering and market basket applications require the use of dimensionality reduction methods.

The basic aim in dimensionality reduction is to condense the data into a few dimensions [7, 13, 17] so that the least amount of information is lost. This is achieved by applying data transformation methods such as Principal Component Analysis (PCA). This helps in several ways: first, it reduces the storage requirement; something which translates directly into improved performance scalability. Lower dimensionality also improves the retrieval efficiency of indexing structures [7, 17]. Thus, the focus of traditional dimensionality reduction methods is simply to construct a new axis-system, so that the discarded dimensions have the least amount of variance. While such an approach has served as a useful method for a number of content based retrieval applications, a variance-centered approach may not necessarily be very useful in arbitrary data mining applications. Some examples of such applications are as follows:

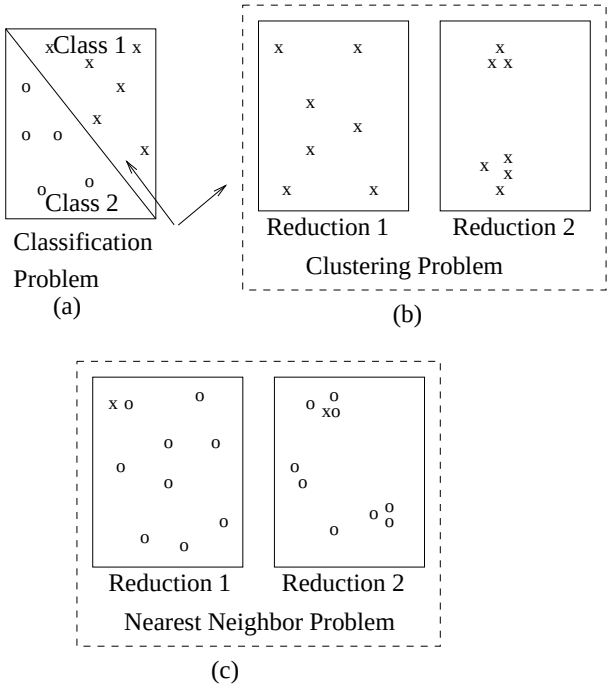

Figure 1: Arbitrary Application Scenarios

Classification Application: Find an l-dimensional set of vectors $\mathcal{V}$ so that the accuracy of a particular classifier on the projected database representation $\mathcal{D}_\mathcal{V}$ is as high as possible. We note that the traditional method of using SVD will not

*IBM T. J. Watson Research Center, charu@us.ibm.com

provide the best subspace which optimizes the class discrimination. In fact, the optimal subspace for performing the dimensionality reduction will vary not only with the nature of the class distribution, but also with the particular kind of classifier which is used for the training process. Ideally, we would like to find the subspace which provides the best discrimination for a particular kind of classifier.

Normalized Similarity Search Application: Find an l-dimensional set of vectors $\mathcal{V}$ so that the the average *normalized* distance of the k closest records to a given target point $\mathcal{T}$ in the projected database $\mathcal{D}_v$ as low as possible. The normalization is performed using the average distance to all other points in the projected database. In other words, if $Dist^k(\mathcal{T}, \mathcal{V})$ is the average (projected) distance of the k closest points to $\mathcal{T}$ in subspace $\mathcal{V}$ for a database of size N, then we would like to find the subspace $\mathcal{V}$ so that $Dist^k(\mathcal{T}, \mathcal{V})/Dist^N(\mathcal{T}, \mathcal{V})$ is minimized. Note that this approach is an effective method for high dimensional nearest neighbor search, since it is often difficult to find the full dimensional nearest neighbors in the presence of data sparsity [3, 10]. Such an approach can also be utilized in a number of applications such as collaborative filtering which are characterized by inherently high dimensional and sparse data.

In order to illustrate the above points, let us examine some cases in Figure 1. In Figure 1(a), we have illustrated a uniformly distributed 2-dimensional data set in which the classes are concentrated in two clusters. We note that traditional dimensionality reduction cannot reduce the data. However, a 1-dimensional reduction is possible in the direction of the arrow in Figure 1(a).In such a situation, it is possible to classify the data perfectly using a 1-dimensional representation. Figure 1(b) illustrates two possible 2-dimensional reductions of a higher dimensional data set. It is clear that reduction 2 is superior to reduction 1, because the data is more effectively clustered in the second case. This effectiveness can be captured in the form of an objective function which minimizes the ratio of the intra-cluster distances to that of the inter-cluster distances. Finally, in Figure 1(c), we have illustrated two possible views for a similarity search application. The target point is marked by 'x'. Again, it is clear that the second reduction is superior to the first, because a clustered group of data points can be retrieved near the target.

The afore-mentioned cases are only some examples of the kinds of applications which may be useful for the data reduction process. The qualitative performance of a wide variety of data mining algorithms such as clustering, classification and outlier detection are highly sensitive to the data representation which is used during execution. In such cases, it is desirable to pick an optimized representation of the data in which the corresponding data mining problem works most effectively. Some recent work has designed methods for supervised dimensionality reduction methods in classification [2, 12], and column subset selection [4] though these techniques are not applicable to general applications. In the next section, we will discuss a general formulation for performing this optimization task.

This paper is organized as follows. In the next section, we discuss the general framework for dimensionality optimization problem. In section 3, we discuss an evolutionary approach for data reduction. We also discuss particular instantiations of the method to a variety of data mining problems. In section 4 we discuss the empirical results for these instantiations. In section 5, we discuss the conclusions and summary.

2 Generalized Dimensionality Reduction

We will first discuss some notations and definitions. We assume that we have a database $\mathcal{D}$ containing N records with a dimensionality of d. Any subspace of dimensionality l is represented by a set of l orthonormal vectors represented by $\mathcal{V}$. The database $\mathcal{D}$ needs to be projected onto $\mathcal{V}$ in order to perform the dimensionality reduction process.

Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) [17] is a well known technique in order to represent the data in a lower dimensional subspace by pruning away those dimensions which result in the least loss of information. The idea is to transform the data into a new coordinate system in which the (second order) correlations in the data are minimized. This transformation is done by using a two step process.

In the first step, the $d * d$ covariance matrix is constructed for the data set. Specifically, the entry (i, j) in the matrix is equal to the covariance between the dimensions i and j. The diagonal entries correspond to the variances of the individual dimension attributes. The covariance matrix C is positive semi-definite, and can be expressed in the following form:

$$C = P \cdot \Delta \cdot P^T \tag{2.1}$$

The columns of P represent the orthonormal eigenvectors of C, whereas the diagonal entries of Δ are the eigenvalues. These eigenvectors define an orthonormal axis-system along which the second order correlations in the data are removed. The corresponding eigenvalues denote the spread (or variance) along each such newly defined dimension in this orthonormal system. Therefore, the eigenvectors with the largest eigenvalues can be chosen as the subspace in which the data is represented.

It can be shown that when the data base $\mathcal{D}$ is projected along the l vectors in P with the largest eigenvalues, the loss in variance is minimized. The standard dimensionality reduction problem can thus be formulated as the following optimization problem:

PROBLEM 2.1. *For a given database $\mathcal{D}$, find the l-dimensional subspace represented by the vectors $\mathcal{V}$, so that the variance of the projected database $\mathcal{D}_{\mathcal{V}}$ is maximized.*

Our formulation of the dimensionality reduction method as an optimization problem is especially useful since it provides a natural way to define the generic dimensionality reduction problem. This general formulation is the focus of this paper and is defined as follows:

PROBLEM 2.2. *For a given database $\mathcal{D}$, find the l-dimensional subspace represented by the vectors $\mathcal{V}$, so that the objective function $f(\mathcal{D}_{\mathcal{V}})$ is optimized.*

Instead of using a fixed dimensionality l, we may also choose to formulate the problem such that the dimensionality of the subspace $\mathcal{V}$ is *at most* l. In such a formulation, when the value of l is chosen equal to the full dimensionality d, this is essentially equivalent to finding *any* subspace of the data which optimizes the objective function $f(\mathcal{D}_{\mathcal{V}})$.

While the standard dimensionality reduction problem (with a variance maximization objective function) is optimally solvable using the SVD technique, this is not necessarily true of the more general formulation with arbitrary objective function. We note that different instantiations of the objective function provide the solution to a variety of interesting problems. Some examples are as follows:

Classification Application: In this application, we wish to find the subspace $\mathcal{V}$ such that the effectiveness of a particular classification algorithm $\mathcal{CLA}$ on the training data $\mathcal{D}_{\mathcal{V}}$ is maximized. We note that the optimal subspace depends not only on the data set being used, but also on the particular classification algorithm being used. For example, a single-attribute decision tree algorithm may work well with a subspace representation in which axis-parallel splits tend to separate large and contiguous blocks of classes well, whereas a nearest neighbor classification algorithm may work well in a subspace representation in which the classes get distributed into spherical clusters of small sizes. Thus, in this case, we define the objective function $f(\mathcal{D}_V)$ as the classification accuracy for the particular algorithm $\mathcal{CLA}$ when the training database $\mathcal{D}_{\mathcal{V}}$ is used.

Clustering Application: The problem of unsupervised feature selection [11] is well known in the clustering literature. In most cases, such methods are heuristic techniques which try to identify particular dimensions (from the original set of attributes) that are known to be noisy and non-informative for the clustering process. The problem is even more difficult for the case when generalized subspaces (that are not parallel to the original axis directions) are used. Our aim is to find a subspace $\mathcal{V}$ such that the effectiveness of a given clustering algorithm $\mathcal{CLU}$ on the database $\mathcal{D}_{\mathcal{V}}$ is maximized. We assume that the effectiveness of the clustering algorithm is measured by an objective function such as the ratio of the average cluster radius to the inter-cluster distances.

Collaborative Filtering Application: In a collaborative filtering application, we would like to find the closest cluster of peers to a given target point. This may be difficult to achieve in the full dimensional case because of data sparsity. In such cases, it may often be useful to find a subspace in which the cluster of peers has the best discrimination with respect to the other data points. One example of such an objective function is the ratio of the average radius of the k-closest points to the average radius of all points in the projected database $\mathcal{D}_{\mathcal{V}}$.

While the standard problem of principal component analysis has an optimal solution which can be expressed in closed form, the above-mentioned problems do not have natural closed form solutions. In fact, in some of the cases, even the objective function to be optimized is not defined in closed form, but is computationally defined in terms of an algorithm. (Examples include optimizing the effectiveness of a particular kind of clustering or classification algorithm.) In such cases, it is not possible to find closed form solutions to the dimensionality reduction problem. In fact, in most cases it is computationally not easy to find the optimal solution, since there can be an infinite number of possible subspaces with arbitrary objective function values. Furthermore, a non-linearity in the nature of the objective function can rule out the possibility of finding simple and efficient algorithms for the task.

Because of the inherent intractability of the underlying problem, we discuss an evolutionary framework for finding the optimal subspaces. The success of an evolutionary framework depends upon the ability to pick an effective encoding as well as the ability of design the operations of the genetic algorithm. Our evolutionary framework proposes a *point-based* encoding of the subspace determination problem and searches selectively over the space of possible encodings. In order to perform an effective evolutionary search, we will discuss effective ways of performing critical operations such as selection, crossover and mutation.

3 The Evolutionary Approach

Evolutionary Algorithms can be a powerful tool for problems in which the objective functions are arbitrary or are computationally designed in the form of algorithms [9]. Evolutionary Algorithms [9] are methods which simulate the process of organic evolution in order to solve parameter optimization problems. The basic idea behind an evolutionary search technique is that every solution to an optimization problem is treated as an individual in an evolutionary system. The measure of fitness of the individual is the objective function value of the corresponding solution. As in the case of organic evolution, this individual has to compete with other individuals from a *population* of solutions to the optimization problem. Thus, unlike other optimization methods such as hill climbing or simulated annealing [14], evolutionary search methods work with an entire population of solutions rather than a single solution. As a result, evolutionary search methods are inherently more robust than hill-climbing, random search or simulated annealing techniques; they combine the power of directed random search (as in the case of hill climbing or simulated annealing) with the power of solution recombination from a population. Appropriate operations are defined in order to imitate the selection, recombination and mutation processes. These procedures are iteratively applied on this population of solutions, until the strings in the population converge to an optimal solution.

Each feasible solution to the problem is represented as a string in order to facilitate genetic processes such as crossover or mutation. The process of conversion of feasible solutions into strings is called **coding**. For example, a possible coding for a feasible solution to the travelling salesman problem could be a string containing a sequence of numbers representing the order in which he visits the cities. The value at each locus on the string is referred to as a **gene**. The measure of fitness of an individual is evaluated by the **fitness function**, and is essentially analogous to the objective function value of the corresponding solution; the higher[1] the objective function value, the better the fitness value.

The selection, crossover, and mutation processes are applied iteratively to the population of solutions. As the algorithm progresses, the individuals in the population tend to become genetically similar to one another. This phenomenon is referred to as **convergence**. Dejong [6] defined convergence of a *gene* as the stage at which 95% of the population had the same value for that gene. The population is said to have converged when all genes have converged.

We note that the quality of the genetic algorithm is highly sensitive to the nature of the coding used, and the way in which the selection, crossover and mutation operations are designed on this encoding mechanism. A poor choice of coding or operators can lead to a solution which is poor approximation of the truly optimal solution. In this section, we will design the genetic algorithm in such a way so that the search space is greatly optimized for the evolutionary search process. Furthermore, since we are discussing a general framework for dimensionality reduction for arbitrary objective functions, we will discuss the multiple design choices which arise in the context of different kinds of objective functions.

3.1 Choice of Encoding An important aspect of the evolutionary search process is the encoding used for representing solutions to the evolutionary search problem. In general, it is desirable to choose a coding of the solution which can be represented as compactly as possible. The size of the search space increases exponentially with the size of the encoding. Therefore, a compact encoding ensures that the genetic algorithm has to search over a smaller set of solutions during the search process. Since the solution to the genetic algorithm is in the form of an l-dimensional subspace $\mathcal{V}$, a natural solution is to represent the string as a concatenation of l different d-dimensional vectors. While the simplicity of the approach is appealing, it suffers from two drawbacks:

The encoding requires $d*l$ real numbers for representing a solution. This is quite a large search space, especially when considered in the context of the binary representation of each such real number. For large choices of d and l, the computational efficiency of a genetic algorithm on such a search space is greatly compromised. Furthermore, a larger size of the search space is likely to reduce effectiveness.

In many cases, we would like to choose the encoding in such a way that the search is restricted to only those l-dimensional hyperplanes which pass through $(l+1)$ linearly independent points in the underlying data. In most data mining applications, this restriction is quite reasonable since these hyperplanes should be able to provide distinguishing characteristics of the data points. (For example, an l-dimensional hyperplane on which all data points project to a single point is not of much use for most applications.) Thus, for skewed data, only a small fraction of the feasible hyperplanes are relevant. How can we design the encoding in such a way that only the relevant portion of the search space is explored

[1] Genetic Algorithms are naturally defined as maximization problems, though only trivial changes are required to make them work as minimization problems.

Algorithm *EvolutionaryReduction(Database: $\mathcal{D}$, ObjectiveFunction: $f(\cdot)$, Dimensionality: l)*
begin
$(\mathcal{P}, \mathcal{S}) = CreatePopulation(popsize, \mathcal{D}, l)$;
{ Assume that the strings in the created population are denoted by $St_1 \ldots St_{popsize}$;}
while $\mathcal{P}$ has not converged **do**
begin
$\mathcal{P} = Selection(\mathcal{P}, f(\cdot))$;
$\mathcal{P} = Crossover(\mathcal{P})$;
$\mathcal{P} = Mutation(\mathcal{P}, |\mathcal{S}|)$;
end
end

Figure 2: The Evolutionary Reduction Algorithm

during the evolutionary process?

It turns out that both of the above goals are achieved by a point sampled encoding. In the point sampled encoding, we need a string of length $(l + 1)$. This string is simply a set of $(l+1)$ data point identifiers from a sample $\mathcal{S}$ of the underlying data base $\mathcal{D}$. If these $(l + 1)$-data points are linearly independent, then they define the l-dimensional hyperplane as a feasible solution to the dimensionality reduction problem.[2] If desired, the entire database can also be used instead of a sample. However, in most cases, the use of a sample is more prudent as it results in a greater level of efficiency. We note that such an encoding results in the set of hyperplanes being skewed by the underlying data distribution. As reported in [1] point sampled hyperplanes are more effective at finding projections which expose the relevant aspects of the data in a variety of applications. For applications in which it is not desirable to bias the hyperplanes by the underlying data, a similar solution can be achieved. In this case, we can start off with a set $\mathcal{S}$ with a random sample of points. The hyperplanes are defined implicitly by sampling $(l+1)$ points from the random set $\mathcal{S}$. An even more flexible solution is to use $\mathcal{S}$ as a mixture of the points from the original data set and a random sample of points. By using such an approach, it is possible to allow the evolutionary approach to choose the search space which fits its requirements best during the optimization process. This was the choice used in our paper, since we chose the sample set $\mathcal{S}$ to be an equally weighted mixture of the two kinds of points.

3.2 Description of the Evolutionary Algorithm

The evolutionary algorithm works with a population $\mathcal{P}$

[2] We will discuss later in the description of the algorithm objective function how to deal with the case when the $(l + 1)$ points are not linearly independent.

Algorithm *CreatePopulation(Population Size: popsize, Database: $\mathcal{D}$, Dimensionality: l)*
begin
‘ Pick a sample $\mathcal{S}$ from the database $\mathcal{D}$;
{ The sample $\mathcal{S}$ may also be chosen to be a random set of data points when it is desirable to find solution hyperplanes which are unbiased from the original data distribution; }
$\mathcal{P} = \{\}$;
for $i = 1$ **to** *popsize* **do**
begin
Construct a string St containing the *ids* of $(l + 1)$ data points from $\mathcal{S}$;
Add St to $\mathcal{P}$;
end
return($\mathcal{P}$, $\mathcal{S}$);
end

Figure 3: Initializing the Population

Algorithm *Selection(Population: $\mathcal{P}$)*;
begin
Determine rank $r(i)$ for each string $St_i \in \mathcal{P}$;
$\mathcal{P}_{new} = \{\}$;
for $i = 1$ **to** $|\mathcal{P}|$ **do**
begin
Sample a string St_{samp} from $\mathcal{P}$ with sampling probability proportional to $\max\{rank\_min, |\mathcal{P}| - r(i)\}$;
Add St_{samp} to $\mathcal{P}_{new}$;
end
return($\mathcal{P}_{new}$);
end

Figure 4: The Selection Operation

Algorithm *Crossover(Population: $\mathcal{P}$)*;
begin
Pick a fraction $p_{crossover}$ of the population $\mathcal{P}$ and pair them;
For each pair of strings create a basket of $(2 \cdot l + 2)$ values by computing the union of values in parent strings;
For each basket of values created, create k' new children strings by choosing k' possible samples of length $(l + 1)$ from the basket of size $2 \cdot (l + 1)$
For each basket of values created, pick the highest fitness child from the k' possibilities, and construct the complement of this child as the second child;
Replace parent strings by children strings in population $\mathcal{P}$;
return($\mathcal{P}$);
end

Figure 5: The Crossover Operation

```
Algorithm Mutation(Population: P,
    Samplesize: |S|)
begin
  for each string St_i ∈ P do
  begin
    Sample a set of positions from St_i with sampling
      probability p_mutate;
    for each such sampled position change it to a randomly
      picked position between 1 and |S|;
  end
end
```

Figure 6: The Mutation Operation

of solutions on which it repeatedly applies the procedures of selection, crossover and mutation. Therefore, the first step is to create an initial population of solutions of size *popsize*. This procedure is illustrated by the procedure *CreatePopulation* of Figure 3. In this procedure, we first create a sample $\mathcal{S}$ of data points. This sample $\mathcal{S}$ can either be chosen from the original database (if it is desirable to choose the hyperplanes from the original distribution of points), or it can be chosen as a random sample of uniformly distributed points. Next, a population of *popsize* solutions is created by repeatedly picking $(l+1)$ data points from the sample $\mathcal{S}$.

Once the initial population has been determined, the selection, crossover and mutation operations are applied iteratively to this population $\mathcal{P}$ of solutions. The overall procedure is illustrated in Figure 2. The procedure takes as input the database $\mathcal{D}$ and the objection function $f(\cdot)$ which needs to be optimized. We note that this objective function may not necessarily be computable in closed form, but may simply be computationally defined in terms of optimizing the behavior of a particular kind of data mining algorithm such as clustering, classification or collaborative filtering. The objective function computation is required during the selection procedure; a point which we will discuss in detail slightly later.

For the purpose of selection, we use a rank-selection mechanism discussed in [9]. Thus procedure is illustrated in Figure 4. Let $r(i)$ be the rank of population member i in order of rank in reducing order of fitness. In the rank selection mechanism, we randomly sample (with replacement) the solutions in the population, while weighting the bias in the sampling by $\max\{rank\_min, |P| - r(i)\}$. Typically, the value of $rank\_min$ is chosen to be the (lowest) quarter percentile [9] by default. This is done in order to ensure that the least fit solutions are not prematurely removed from the population. The effect of the selection procedure is to bias the population in favor of more favorable solutions. When used in combination with the other operators of selection and crossover, the characteristics of fit solutions are combined in order to create combinations of even fitter solutions. We note that in the process of selection, the objection function $f(\cdot)$ needs to be computed. In the event that this objective function is defined algorithmically (such as the effectiveness of a particular data mining algorithm such as classification), the subspace for each string needs to be computed, and the corresponding algorithm needs to be executed on a sample of the data. For example, in one of our implementations for the use of a classification application, we tested the classification accuracy on a small sample of the data points in order to evaluate the corresponding objective function.

We note that each string of length $(l+1)$ represents a subspace of dimensionality *at most* $(l+1)$. This is because the data points comprising the solution string may not be linearly independent or there may be repetitions among the $(l+1)$ point ids represented in the strings. When it is desirable to find only subspaces of dimensionality *exactly* $(l+1)$, this can be achieved by setting the objective functions of solutions with dimensionality less than l to arbitrarily low values. Otherwise, the genetic algorithm may converge to an optimal subspace with dimensionality at most l, but not necessarily equal to l. Thus, by choosing l to be equal to the full dimensionality d it is possible to find the optimal subspace of any dimensionality satisfying a pre-defined objective function.

After the selection process, a crossover mechanism is applied in order to combine the solutions from the population. In the crossover operation, the strings in the population are paired randomly with one another and are recombined in order to create new solutions which combine features from both parents. The crossover operation is a key operation in evolutionary search methods which is not present in methods such as simulated annealing. The ability to create solutions which combine characteristics from two solutions of good quality often creates even better solutions than can be obtained from the neighborhood search of either solution. For the particular case of the dimensionality reduction problem, we would like to combine the two solution subspaces in such a way so that the children strings contain large subspaces in common with the parent strings. Fortunately, our encoding provides a natural way to perform the crossover operation so as to create a subspace which combines features from both subspaces.

A simple way to do this is as follows. First, we pick the $(l+1)$ point ids from each parent and combine them to create a basket of $(2 \cdot l + 2)$ point ids. This

set of $(2 \cdot l + 2)$ point ids may possibly have repetitions. The children are created by randomly partitioning the $(2 \cdot l + 2)$ points into two sets of $(l + 1)$ points. We note that the children may possibly have repetitions of point ids, resulting in less than l-dimensional string solutions. Such solutions are automatically selected out by the evolutionary algorithm, when only subspaces of dimensionality exactly l are required. When l_1 of the point ids in a child solution are drawn from one of the two parents, this means that the parent and child hyperplane share at least an $(l_1 - 1)$-dimensional subspace in common. In many cases, this re-combination may result in a child solution which combines the fit aspects of both parents. It is these cases, which are useful to the evolutionary approach. Therefore, we improved the quality of the crossover further by using the following optimization mechanism. From the basket of $(2 \cdot l + 2)$ values, we created the pairwise partitions in $k' \geq 1$ distinct ways. Of these k' distinct pairs of children, we picked the child pair which included the fittest children from all $2 \cdot k'$ possible children. The crossover operation is described in Figure 5.

As is evident from the description of the crossover operation in Figure 5, all pairs of solutions are not necessarily recombined during an iteration of the crossover procedure on the population. Only a fraction $p_{crossover}$ of the population of solutions are picked by the algorithm and recombined in each iteration. This ensures that solutions of high quality within the population have a non-zero probability of surviving intact without recombination. The crossover operation is followed by the mutation operation.

The purpose of the mutation operation is to use the recombination procedure in conjunction with a directed neighborhood search of the population of solutions. In order to perform the mutation, we flip a biased coin with success probability equal to p_{mutate} for each position in the string. When the flip results in a success, that position is modified to randomly picked point id from the sample $\mathcal{S}$. This kind of procedure ensures that the changed subspace is quite similar to the original subspaces, but with minor variations once in a while. When such mutations improve the quality of the objective function, this is likely to be retained by the population because of the selection procedure. The overall process of mutation is illustrated in Figure 6.

The processes of selection, crossover and mutation are repeatedly executed until the population of solutions shows convergence behavior. As defined in [6], we considered the population to have converged when 95% of the genes have converged. During the process of execution of the genetic algorithm, we saved the best 10 strings in the population. In the next section, we will discuss two specific instantiations of the dimensionality reduction problem.

3.3 Specific Instantiations We applied our approach to two specific instantiations of the dimensionality reduction problem: (1) Standard Dimensionality Reduction for Principal Component analysis. (2) A dimensionality reduction which optimizes the accuracy of a nearest neighbor classifier.

The first instantiation would seem like a futile exercise since the problem is already efficiently and optimally solvable in closed form by SVD based methods. We do not expect an evolutionary approach to provide exactly optimal solutions as is the case with an SVD method. However, the closed form solvability of the principal component analysis problem also provides us with an opportunity to judge the general effectiveness of an evolutionary search strategy on the broad problem of dimensionality reduction. For arbitrary objective functions, we simply do not know the value of the optimal solution, while for the case of SVD, the optimal solution is known. While it is not certain that the evolutionary search method would work equally well with all objective functions, it is generally the case that the effectiveness of an evolutionary approach is more sensitive to the nature of the solution and its encoding as opposed to the objective function associated with the problem [9]. Therefore, if we are able to use the evolutionary approach to find solutions which are close to optimal in the case of principal component analysis, it provides evidence that the approach is likely to provide near optimal solutions with other objective functions.

An attempt was made to use a minimal level of difference between the implementations for the two different instantiations. Therefore, the code used for both instantiations was essentially identical with the only difference being the objective function module. Both the instantiations were run with the same set of parameters over all runs. Specifically, we used $p_{crossover} = 0.6$, $p_{mutate} = 0.002$, and $popsize = 500$ which are typical parameter settings for an evolutionary approach. We note that it may be possible to improve the performance of the technique further by optimizing the specific implementations in a problem or data specific way. However, our aim here is to show that very effective results were obtained with a *general* framework requiring minimum changes across different problems.

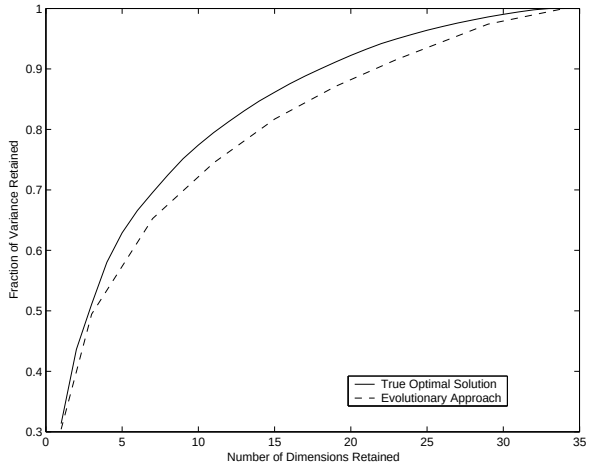

Figure 7: Standard Dimensionality Reduction (Ionosphere)

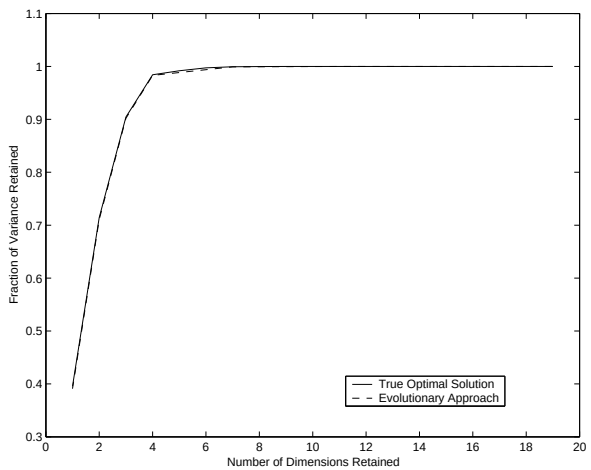

Figure 8: Standard Dimensionality Reduction (Segmentation)

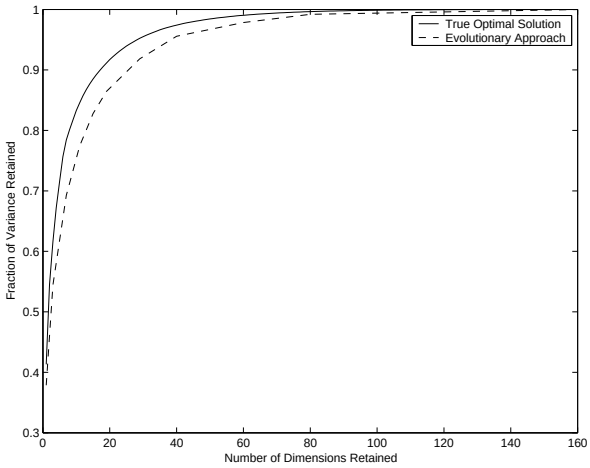

Figure 9: Standard Dimensionality Reduction (Musk)

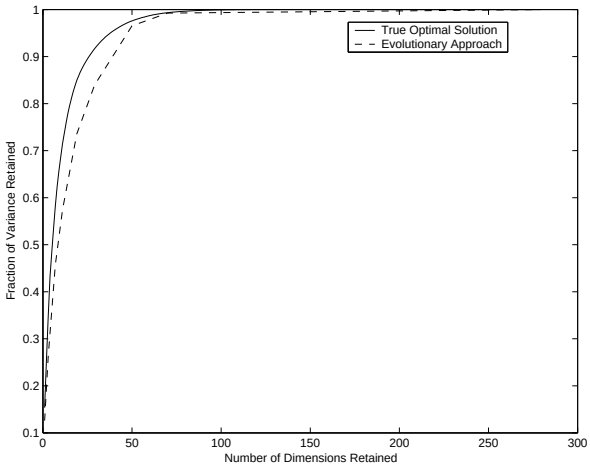

Figure 10: Standard Dimensionality Reduction (Arrythmia)

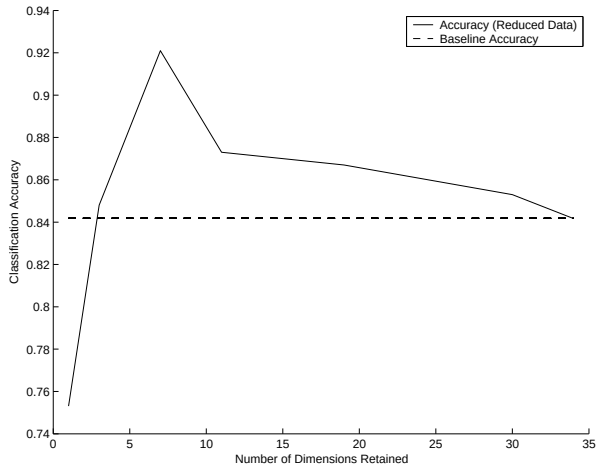

Figure 11: Optimizing a Nearest Neighbor Classifier with Dimensionality Reduction (Ionosphere)

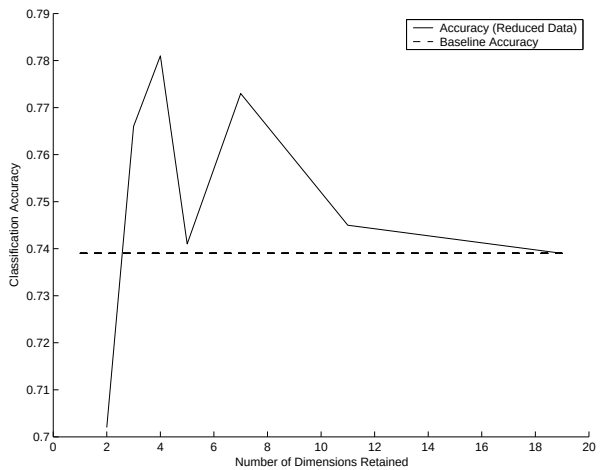

Figure 12: Optimizing a Nearest Neighbor Classifier with Dimensionality Reduction (Segmentation)

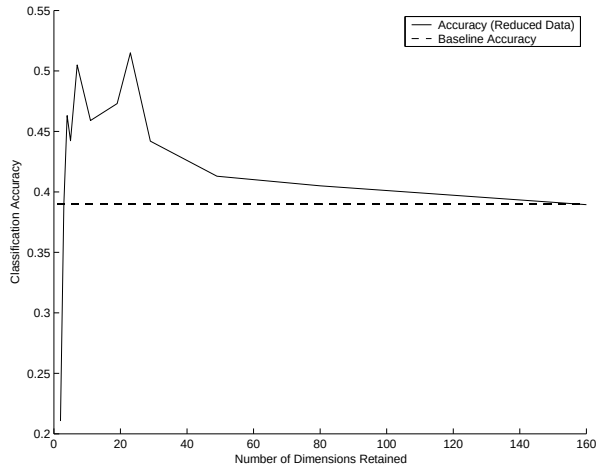

Figure 13: Optimizing a Nearest Neighbor Classifier with Dimensionality Reduction (Musk)

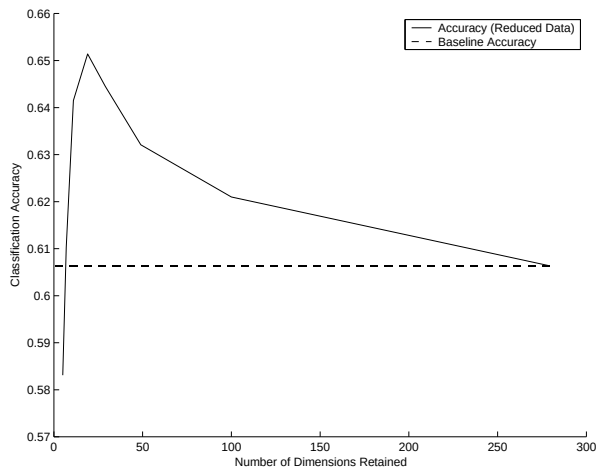

Figure 14: Optimizing a Nearest Neighbor Classifier with Dimensionality Reduction (Arrythmia)

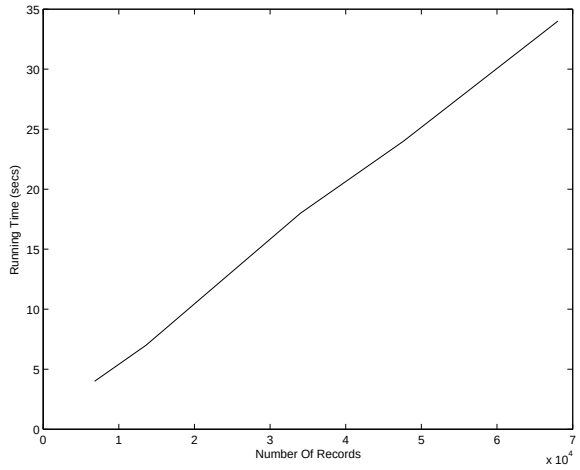

Figure 15: Scalability of Evolutionary Approach with Increasing Data Size

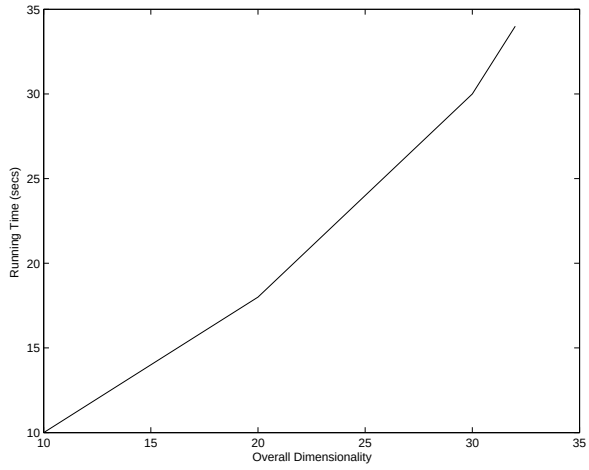

Figure 16: Scalability of Evolutionary Approach with Increasing Data Dimensionality

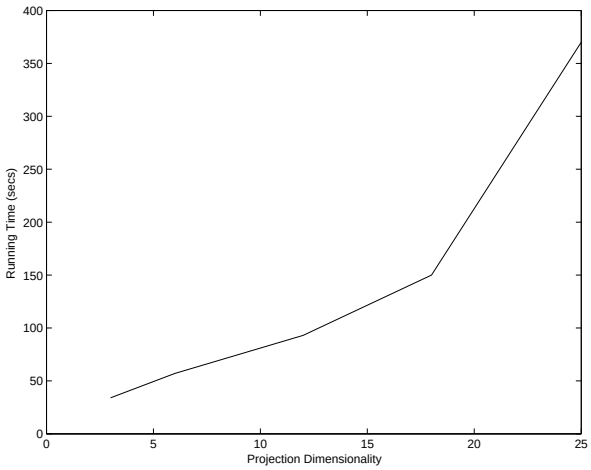

Figure 17: Scalability of Evolutionary Approach with Increasing Projection Dimensionality

4 Empirical Results

We tested the algorithm over a number of data sets from the UCI machine learning repository.[3] We tested the algorithm for a number effectiveness and efficiency measures. For the purpose of effectiveness, we tested the algorithm for two different objective functions:

1. We tested the standard dimensionality reduction scenario in which the variance of the data reduced data needs to be maximized. In this case, the problem is optimally solvable in closed form using singular value decomposition. As discussed earlier, we do not expect an evolutionary approach to provide exactly optimal solutions as is the case with an SVD method. However, the closed form solvability of this problem also provides us with an opportunity to judge the general effectiveness of our evolutionary search strategy on the dimensionality reduction problem. In many cases, a particular encoding may not work effectively simply because it is not suited to the solution space for that problem [9]. If the evolutionary search technique performs almost as well as the "gold standard" SVD method, this provides some evidence for the effectiveness of the technique over arbitrary objective functions as well. If we do not test on optimally solvable cases, there is no way of knowing how closely the evolutionary search method approximates the true optimal value for different kinds of problems.

2. We also tested the case in which we wished to find a reduction such that the effectiveness of a nearest neighbor classifier is optimized. In this case, the truly optimal representation is unknown. However, we can measure

[3]The UCI machine learning repository is available at *http*;//*www.cs.uci.edu/*~*mlearn*.

the effectiveness of the approach by its usefulness in improving the classification accuracy over the baseline classification accuracy (using the entire set of dimensions).

We also tested the approach for efficiency by testing the scalability with data size and dimensionality. The results seem to indicate that the evolutionary search method can be used efficiently over a wide variety of data sets.

We first ran the evolutionary search method for a number of different data dimensionalities and compared the results to the SVD method. The results for the Ionosphere, Segmentation, Musk, and Arrythmia data sets are illustrated in Figures 7, 8, 9, and 10 respectively. In each graph, the X-axis represents the dimensionality of the reduction. The Y-axis represents the fraction of the variance which is retained in that reduction. This variance fraction is defined by the variance of the data in the reduced space divided by the variance of the data in the original space. In each graph, we have illustrated the results for both the evolutionary approach as well as our "gold standard"- the truly optimal solution using SVD based dimensionality reduction. It is interesting to note that in each case, the evolutionary approach provides results which are almost as effective as the optimal reduction. Furthermore, solution provided by the evolutionary approach was within 5% of the truly optimal solution. In most cases, the approximations were much better. For example, in the case of the segmentation data set, the reduction chosen by the evolutionary approach was optimal within 0.5% for every case. As a result, the optimal solution cannot be distinguished from the evolutionary solution in Figure 8. Another salient observation from Figures 7, 8, 9, and 10 is that the evolutionary approach provided the best approximation when the number of reduced dimensions were sufficient to capture 90% or more of the original variance in the data. This shows that almost the same number of dimensions are required by the evolutionary approach to capture most of the variance in the data.

We also tested the evolutionary approach for finding a reduction which optimizes the classification accuracy of a nearest neighbor classifier. For a data set of dimensionality d the optimal reduction was determined on a small sample set of $\max\{200, 6 * d\}$ points. A separate test set was used for evaluating the quality of the nearest neighbor classifier. In Figures 11, 12, 13, and 14 we have illustrated the classification accuracy of the method in the reduced space for different dimensionalities. We have also illustrated the baseline classification accuracy of a full dimensional nearest neighbor classifier using the same test set. It is clear that in each case, the classification accuracy can be significantly improved with the use of a dimensionality reduction technique. Another interesting observation is that *very low* dimensional projections yield nearest neighbor classifiers which are competitive to the full dimensional nearest neighbor classifier. For example, for the case of the (34-dimensional) ionosphere and (19-dimensional) segmentation data sets, the use of 2-dimensional projections was sufficient to match the effectiveness of a full dimensional nearest neighbor classifier. Note that these dimensionalities of projection are *significantly* lower than those needed to capture the variance in the data (as illustrated in Figures 7, 8, 9, and 10). For projections of even slightly higher dimensionality, the classification accuracy was greatly improved. This is because such lower dimensional projections can effectively mask out the noise in the data without losing any of the relevant information needed for classification purposes. The behavior for the 160-dimensional musk data set, and 279-dimensional arrythmia data set was even more interesting. For the case of the 160-dimensional musk data set, the optimal 3-dimensional projection performed competitively to the full dimensional nearest neighbor classifier. A 4-dimensional projection was able to provide a classification accuracy which was significantly higher than the full dimensional case. For the case of the 279-dimensional arrythmia data set, a 7-dimensional projection was able to outperform the nearest neighbor classifier. These results provide overwhelming evidence about the level of redundancy which is inherently present in data sets for a variety of applications. Techniques such as those discussed in this paper can be leveraged to not only mine out the redundancy but also provide a crisp and concise data representation which is optimized to a particular kind of application.

Another interesting observation from Figures 12 and 13 that the behavior of classification accuracy with projection dimensionality does not show the kind of parabolic behavior shown by the standard dimensionality reduction problem. In some cases, the classification accuracy was almost equally high for different dimensionalities of projection. This tends to indicate the best hyperplanes for projection are hidden in subspaces of varying dimensionality. This makes the problem particularly difficult to solve, since it means that the class of procedures which build subspaces in a bottom-up or top-down fashion are unlikely to work (if computationally feasible at all).

We tested the scheme for its scalability with data size, data dimensionality and projection dimensionality on the conventional dimensionality reduction problem. Since none of the UCI data sets are particularly large, we used a 32-dimensional color histogram data set discussed in [1]. In order to construct data sets of

varying size and dimensionality we used two methods: **(1)** We constructed random samples of varying number of records from the data set. **(2)** We constructed data sets of varying dimensionality from the data set.

In Figures 15, we have illustrated the scalability with increasing data size. The results in Figure 15 reflect the running time for finding a 3-dimensional projection in each case. The running time scaled linearly with increasing data size. This is simply because the computation of the evaluation function scaled in this way with increasing data size. The number of evaluations of the objective function did not change significantly. In Figure 16, we have illustrated the scalability behavior of the algorithm with increasing data dimensionality. As in the previous case, the *projection* dimensionality was fixed at 3. It is clear from Figure 16 that the algorithm scaled slightly more than linearly with data dimensionality. This scalability increase rate is however still less than quadratic. This is because the main effect of increased dimensionality was a linear increase in the computation time of the objective function. The convergence behavior of the algorithm was also increased slightly because of the increase in search space size. As a result, the scalability was only slightly worse than linear. In Figure 17, we have illustrated the scalability of the algorithm with increasing projection dimensionality when the original data set was used. An increase in the projection dimensionality increased both the size of the encoding (which slowed the algorithm convergence) as well as the computation of the objective function. As a result the algorithm scaled superlinearly. The overall scalability was still better than quadratic. It is also clear from Figures 15, 16 and 17 that in each case, the dimensionality reduction method was very efficient and converged to an optimal solution in a few minutes.

5 Conclusions and Summary

In this paper, we proposed the generalized dimensionality reduction problem. While traditional dimensionality reduction is effective for preserving the variance in the data, it is often desirable to reduce the data differently in order to improve the effectiveness of different kinds of data mining algorithms such as clustering, classification or collaborative filtering. As a concrete example, we implemented a dimensionality reduction strategy which optimizes the effectiveness of a nearest neighbor classifier. This classifier creates a data representation which significantly improves over the behavior of a baseline classifier using only a small number of dimensions. We discussed an effective implementation of the evolutionary approach which is able to converge to a near optimal solution in most cases. The method is very efficient, and scales well with data set size and dimensionality.

References

[1] C. C. Aggarwal, *Hierarchical Subspace Sampling: A Unified Framework for High Dimensional Data Reduction, Selectivity Estimation and Nearest Neighbor Search*, ACM SIGMOD Conference, (2002), pp. 452–463.

[2] C. C. Aggarwal, *A Framework for Local Supervised Dimensionality Reduction of High Dimensional Data*, SIAM Conference on Data Mining, (2006), pp. 360–371.

[3] K. Beyer, J. Goldstein, R. Ramakrishan, and U. Shaft, *When is Nearest Neighbors Meaningful?* ICDT Conference, (1999), pp. 217–235.

[4] C. Boutsidis, M. W. Mahoney, and P. Drineas, *An improved approximation algorithm for the column subset selection problem*, SODA Conference, (2009), pp. 968-977.

[5] K. Chakrabarti, and S. Mehrotra, *Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces*, VLDB Conference, (2000), pp. 89–100.

[6] K. A. De Jong, *Analysis of the behaviour of a class of Genetic Adaptive Systems*, Ph.D. Thesis, Univ. of Michigan, Ann Arbor, (1975).

[7] C. Faloutsos, and K.-I. Lin, *FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets*, SIGMOD Conference, (1995), pp. 163–174.

[8] J. J. Grefenstette, *Genesis Software Version 5.0*, Available at http://www.santafe.edu.

[9] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Addison Wesley, Reading, MA, (1989).

[10] A. Hinneburg, C. C. Aggarwal, and D. A. Keim, *What is the Nearest Neighbor in High Dimensional Spaces?* VLDB Conference, (2000), pp. 506–515.

[11] A. Jain, and R. Dubes, *Algorithms for Clustering Data*, Prentice Hall, New Jersey, (1998).

[12] S. Ji, and J. Ye, *Linear Dimensionality Reduction for Multi-label Classification*, IJCAI Conference, (2009), pp. 1077–1082.

[13] I. T. Jolliffe, *Principal Component Analysis*, Springer-Verlag, New York, (1986).

[14] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by Simulated Annealing*, Science, (220) (4589), (1983), pp. 671–680.

[15] J. Kleinberg, and A. Tomkins, *Applications of Linear Algebra in Information Retrieval and Hypertext Analysis*, ACM PODS Conference, (1999), pp. 185–193.

[16] C.-H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, *Latent Semantic Indexing: A Probabilistic Analysis*, ACM PODS Conference, (1998), pp. 159–168.

[17] K. V. Ravi Kanth, D. Agrawal, and A. Singh, *Dimensionality Reduction for Similarity Search in Dynamic Databases*, ACM SIGMOD Conference, (1998), pp. 166–176.