Data Reduction in Classification: A Simulated Annealing Based Projection Method

Tian Siva Tian [*] Rand R. Wilcox [†] GARETH M. JAMES [‡]

Abstract

This paper is concerned with classifying high dimensional data into one of two categories. In various settings, such as when dealing with fMRI and microarray data, the number of variables is very large, which makes well-known classification techniques impractical. The number of variables might be reduced via principal component analysis or some robust analog, but these methods are usually unsatisfactory for the purpose of classification because they are unsupervised learning methods and not designed to minimize classification errors. In this paper, we propose a classification guided dimensionality reduction approach incorporating a stochastic search algorithm in order to look for a "good" subspace in the context of classification. Two different versions of the simulated annealing algorithm are implemented to produce sparse and dense models, respectively. Using data from both simulation and real world studies, situations are found where the misclassification rate can be reduced by the proposed approach.

KEY WORDS: Data reduction; Classification; Stochastic search; Simulated annealing.

1 Introduction

A classic problem in data analysis is classifying high dimensional data into multiple predefined categories. In particular, two-group classification problem has received a great deal of attention.

[*]Department of Psychology, University of Houston, Houston, TX 77204, ttian@uh.edu.

[†]Department of Psychology, University of Southern California, Los Angeles, CA 90089, rwilcox@usc.edu.

[‡]Department of Information and Operations Management, University of Southern California, Los Angeles, CA 90089, gareth@marshall.usc.edu.

Many methods have been proposed. But in various situations, especially when analyzing data with a small sample size relative to the number of predictors (e.g. medical image, genetic microarray, chemometrics and text classification), many classification techniques become impractical. For example, Fisher's discriminant analysis is not applicable if the number of input variables is greater than the number of observations. Other methods, even some sophisticated methods, such as neural networks (NN) and support vector machines (SVM), do not explicitly require the data dimension smaller than the sample size, but give poor classification accuracy in practice when the data dimension is ultra high, as in fMRI and microarray data. Moreover, for the sake of model simplicity, a concise relationship between input variables and the response is required to achieve a better model interpretation.

A natural way to deal with high dimensional classification problems is to first reduce the data to a lower dimensional subspace and then apply some standard classification strategy, such as linear discriminant analysis or logistic regression (LR), to the reduced data. Data reduction as the first step is usually done by applying principal component analysis (PCA) (Hotelling, 1933; Pearson, 1901) or perhaps some robust analog. Given an input data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, classical PCA (CPCA) finds a small number p of linear combinations of the d input variables that capture most variability in the data. In other words, CPCA looks for a p-dimensional linear subspace that minimizes the sum of squared errors measured by the squared distances from the data points to the subspace. As is known, CPCA is very sensitive to outlying points, because it computes eigenvalues and eigenvectors based on the conventional covariance matrix. Consequently, there might be situations when the components explain a structure created by a relatively small number of outliers.

In recent years, several robust versions of PCA have been developed, such as the minimum volume ellipsoid (MVE) (Rousseeuw, 1985), the minimum-covariance determinant (MCD) estimator

(Rousseeuw, 1984) and the ROBPCA method (Hubert et al., 2005).

However, PCAs can perform poorly on the reduced data, even if a robust version is applied, because any *unsupervised* data reduction procedure can result in the loss of classification information. This is because unsupervised learning methods such as PCAs do not make use of the response variable, and hence may exclude components with little variance but a great deal of group information. A good data reduction method for classification would look for the optimal subspace that contains the most group information in the setting of *supervised learning*.

In this paper, we propose a classification guided dimensionality reduction approach that seeks a lower dimensional subspace that minimizes the misclassification error. By employing stochastic search algorithms, this approach uses the misclassification error to adjust the choice of the reduced subspace. Two stochastic search algorithms using simulated annealing are proposed to search for a good projection that projects the original high-dimensional data onto a lower dimensional subspace in the context of classification.

For difficult optimization problems, such as looking for the optimal subspace for classification, stochastic search are a natural approach. In order to reduce the data space in the setting of supervised learning, stochastic search algorithms have been previously implemented to perform variable subset selection in linear regression (George and McCulloch, 1995; Ntzoufras et al., 1997; Yi et al., 2003). One difficulty with these approaches is the prior specification which requires contextual interpretations of a large number of parameters. Another problem is that for variable subset selection methods in general, they only obtain relevant variables from a set of more complex variables, so they usually have difficulties in assessing the joint effect of multiple variables and can exclude potentially valuable variables which are not predictive individually but may provide significant values in conjunction with others. Instead of selecting a subset of input variables, we look

for lower-dimensional directions that make the projected data optimally classified using simulated annealing algorithms.

Unlike PCAs, the proposed method takes into account the response variable, and hence result in less loss of information than PCAs. We introduce two different simulated annealing algorithms, representing dense and sparse models, respectively. Both versions search for the optimal subspace of the data space with different levels of sparsity restrictions. Similarly to PCAs, the dense version may perform well on multicollinearities which commonly occur in high dimensional data, while the sparse version is more akin to variable selection and may deal well with informative variables and provide better interpretation. Furthermore, the proposed method makes fewer assumptions about distributions of the input data and the response. The proposed method can be used in conjunction with virtually any classification procedure. In this paper we mainly focus on examining the relative performance of the proposed method incorporated with LR, SVM and k-nearest neighbors (kNN).

The rest of the paper is organized as follows. In Section 2, we illustrate the idea of the simulated annealing based projection method, which attempts to project the data onto a lower-dimensional subspace with the goal of minimizing classification errors. We propose two different simulated annealing (SA) algorithms. In Section 3, we examine the performance of the proposed method in classification tasks on two simulation studies and two real-world studies. A further comparison of the simulated annealing based projection method with some modern classification techniques, such as margin trees (MT) (Tibshirani and Hastie, 2007), SVM (Vapnik, 1998), k-nearest neighbors (kNN) (Fix and Hodges, 1951), neural networks (NN), partial least squares regression (PLSR) (Wold, 1966) and boosting (Freund, 2001), will also be provided. In Section 4 we investigate the empirical convergence rate and the solution similarity for the proposed method. A

discussion of the significance and limitations of the proposed method will be presented in Section 5.

2 Methodology

Let $\mathbf{X}_{n,d}$ be a d-dimensional data set, where n is the sample size and d is the number of variables. Write $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d]$, where $\mathbf{x}_j$, $j = 1, \ldots, d$, is the jth component of $\mathbf{X}$. Since all vectors are assumed to be column vectors, we write $X_i = [x_{i,1}, \ldots, x_{i,d}]^T$. Hence, the ith row of $\mathbf{X}$ is X_i^T. Given $p < d$, the goal is to find a p-dimensional subspace of the original data that minimizes the classification error. In this paper, we use a 0-1 loss function indicating the misclassification rate (MCR), denoted by $\hat{e}$. Note that other options to evaluate the classification error, such as Gini index and entropy, can also be used. Let $\mathbf{A}$ be a $d \times p$ transformation matrix. The p-dimensional subspace can be obtained by the following transformation:

$$\mathbf{Z}_{n,p} = \mathbf{X}_{n,d}\mathbf{A}_{d,p}. \tag{1}$$

Equation (1) means that each of the p columns of $\mathbf{A}$ linearly transforms $\mathbf{X}$ into a single dimensional space. Hence $\mathbf{A}$ transforms $\mathbf{X}$ into a $p < d$-dimensional subspace, $\mathbf{Z}$. Ideally, $\mathbf{A}$ should be chosen to minimize MCR. Note that for different classification methods, there might be different ideal $\mathbf{A}$'s. However, finding the best subspace for a specific classification model is difficult to achieve in practice because $\mathbf{A}$ represents a very high dimensional parameter space (d-by-p dimensional). In this paper, we introduce simulated annealing algorithms for seeking a good $\mathbf{A}$ that discriminates between two classes given a pre-selected classification method.

2.1 Simulated Annealing

Simulated annealing (SA) is a generic stochastic search algorithm for global optimization problems derived by Kirkpatrick et al. (1983). As its name suggests, SA is inspired by annealing in metallurgy. At the beginning of the annealing process, a metal is heated to enable the diffusion of atoms to break bonds. As the temperature cools down slowly, the metal progresses towards its equilibrium state and thus achieves better physical properties. By mimicking this physical process, SA inherits the "temperature" concept. In each step of SA, it replaces the current solution with a random neighborhood solution with a probability that depends on the difference between the current solution and the new solution, and the temperature parameter T. The T should be large at the beginning and gradually decreased during the process. By doing this, the current solution can be randomly replaced at the beginning. This helps the algorithm escape from local optima. As T decreases, the algorithm is more likely to accept "good" solutions and discard "bad" ones. The SA algorithm has been theoretically proven to reach global optima with probability one (Geman and Geman, 1984).

2.2 Simulated Annealing Algorithm

As a heuristic algorithm, simulated annealing has been employed to solve global optimization problems. Xie et al. (1993) experimented with SA techniques for finding the optimal directions of projection pursuit based PCA. In our setting, we use SA to guide the search of $\mathbf{A}$ towards the directions that minimize MCR. The algorithm starts with an initial transformation matrix $\mathbf{A}^{(0)}$. According to Bohachevsky et al. (1986), the final results from SA are not significantly affected by the choice of the starting point. In our exploratory experiments, we tried two initial transformation matrices. One is a randomly generated matrix, and the other is the first p columns of the loading

matrix from PCA. We found the results are very similar. We will illustrate this point in Section 4. Hence, in our studies, all initial matrices are used as the first p columns of the loading matrix from PCA. Write the initial projection as $\mathbf{A}^{(0)} = [\mathbf{a}_1^{(0)}, \mathbf{a}_2^{(0)}, \ldots, \mathbf{a}_p^{(0)}]$ and the resulting lower-dimensional space as $\mathbf{Z}^{(0)}$. One would then apply a preselected classification method and compute MCR, $\hat{e}^{(0)}$, based on $\mathbf{Z}^{(0)}$. There are many classification techniques can be applied, such as LR, SVM and kNN. The main goal for this paper is to investigate the effectiveness of the proposed algorithm given a classification method is chosen. That is, the proposed method can find a proper reduced subspace based on the chosen classification method.

Suppose that the projection matrix is $\mathbf{A}^{(l)} = [\mathbf{a}_1^{(l)}, \ldots, \mathbf{a}_p^{(l)}]$ in the lth step of SA. We propose two approaches to generate the new transformation matrix in step $(l+1)$. The first approach, which we call SA-Dense, produces a relatively dense transformation matrix, $\mathbf{A}$. By a dense matrix here we refer to a matrix with few zero or close-to-zero elements. Let a scalar Δr be the step size and $\mathbf{v} = [v_1, \ldots, v_d]^T$ be a d-dimensional column vector representing a random direction. Each element of $\mathbf{v}$ is given by

$$v_j = \frac{u_j}{||\mathbf{u}||}, \tag{2}$$

where $u_j \sim \mathcal{N}(0,1)$, $j = 1, \ldots, d$, $||\cdot||$ is the L_2-norm, and $\mathbf{u} = [u_1, \ldots, u_d]^T$. Randomly select column $\mathbf{a}_j^{(l)}$ of $\mathbf{A}^{(l)}$, and compute a new column $\mathbf{a}_j^{new}$ by

$$\mathbf{a}_j^{new} = \mathbf{a}_j^{(l)} + \Delta r \mathbf{v}. \tag{3}$$

Use $\mathbf{a}_j^{new}$ to substitute $\mathbf{a}_j^{(l)}$, and then obtain a new matrix $\mathbf{A}^{new} = [\mathbf{a}_1^{(l)}, \ldots, \mathbf{a}_j^{new}, \ldots, \mathbf{a}_p^{(l)}]$.

The second approach, which we call SA-Sparse, results in a sparse transformation matrix with most elements close to zero. These close-to-zero elements contribute little to projecting the data, hence we can ignore these elements and consider the transformation matrix sparse. Instead of

revising the whole random column of the current projection matrix, it revises elements of $\mathbf{A}^{(l)}$ independently each with a small probability, ρ. If the (i,j)th element of $\mathbf{A}^{(l)}$, $a_{i,j}^{(l)}$, is replaced, it is replaced by $a_{i,j}^{new} = a_{i,j}^{(l)} + v$, where $v \sim \mathcal{N}(0, \sigma^2)$. We then rescale the new transformation matrix $\mathbf{A}^{new}$ to make each column of length one. Here σ is a tuning parameter, which is similar to the step size in the SA-Dense algorithm. When σ is small, after rescaling, the transformation matrix changes slightly and it will still be a dense matrix. However, when σ is big, after rescaling, the revised elements will dominate the whole column, so that $\mathbf{A}$ becomes relatively sparse. Hence, σ is a tuning parameter which adjusts the sparseness of the transformation matrix. By adjusting σ, we can set the algorithm from highly dense to highly sparse. In this paper, we mainly examine properties when $\mathbf{A}$ is sparse, i.e. σ is large, as oppose to the SA-Dense approach.

The acceptance for $\mathbf{A}^{new}$ is based on the cross-validation (CV) error on the data. We use a 10-fold CV method in this paper. That is, for an $\mathbf{A}^{new}$, the corresponding reduced subspace is divided into 10 folds. For each of 10 experiments, use 9 folds for training and the remaining one fold for testing. Then the CV error, $\hat{e}^{new}$, is the average over all 10 errors on the test folds. Define the perturbation of $\hat{e}$ by $\Delta\hat{e} = \hat{e}^{new} - \hat{e}_l$. If $\Delta\hat{e} < 0$, the projection matrix $\mathbf{A}^{new}$ is better than the current one and is accepted. So let $\mathbf{A}^{(l+1)} = \mathbf{A}^{new}$. If $\Delta\hat{e} \geq 0$, then accept $\mathbf{A}^{new}$ with probability

$$q_l = \exp\left(-\frac{\Delta\hat{e}}{T_l}\right), \tag{4}$$

where T_l is a decreasing sequence (called temperature), so that the search space becomes smaller. Let $0 < b < 1$ be a constant associated with the lth iteration of the method representing the decreasing rate. After L steps (e.g., 20 or 30), the temperature decreases at a rate $1-b$. Repeat this process I times until it meets some termination criterion (e.g., a pre-defined number of steps).

Systematically, the SA algorithm can be described as follows:

1. Given p, generate $\mathbf{A}^{(0)}$ from CPCA, and compute $\hat{e}_0$. Initialize T_0, L, b, and Δr (or ρ and σ).

2. Do until $l = I$.

   (a) If l is a multiple of L, $T_{l+1} \leftarrow bT_l$; otherwise, $T_{l+1} \leftarrow T_l$.

   (b) Use either algorithm 1 or 2 to obtain a new projection matrix $\mathbf{A}^{new}$.

   (c) Compute $\hat{e}^{new}$ from $\mathbf{A}^{new}$.

   (d) If $\Delta\hat{e} < 0$, $\mathbf{A}^{(l+1)} = \mathbf{A}^{new}$ and $\hat{e}_{l+1} = \hat{e}$; otherwise, accept it with probability q_l.

   (e) Set $l \leftarrow l + 1$.

3. End do.

2.3 Tuning parameters

We have several tuning parameters in our algorithms. First and foremost, the choice of the number of components p has a significant effect on the results. In general p can be set to be either greater or smaller than the sample size, but in practice p should be selected to be cost-effective. Certainly, a smaller p will have less computational expense, but if p is too small, there will be more chance of excluding important information. As p increases, information loss will be reduced, but the chance of including unnecessary variables and components will increase. The inclusion of unnecessary variables will complicate or mask the group structure and hence destroy the classification accuracy (Tadesse et al., 2005). In addition, the computational cost will increase too. Therefore, p should be selected as a good balance between computational time and classification accuracy. In this paper, we mainly explore when p is given, the merits of the proposed method relative to other unsupervised data reduction methods such as CPCA. In our simulation studies we examine the model

performance by selecting different values of p. In real world studies, some empirical knowledge about p may be applied. For example, in fMRI studies, we may have prior knowledge about how many brain regions might contribute to a specific task. And that number would be the best prior guess of p. On our real examples, we will investigate several values for p that may be appropriate.

Another important tuning parameter is the number of iterations, I. We choose to use $I = 2000$ based on our experiments. On the one hand, we will see in the next section when using $I = 2000$, $\hat{e}$ has already converged to some low level for both simulated and real world data. For some moderate-scale data, an even smaller I can be used (e.g., in our simulated data example when $d = 50$). On the other hand, an ultra large I may cause overfitting problems which we demonstrate in Section 4.

There are additional tuning parameters in the SA algorithms. In the SA-Dense algorithm, Δr is a constant that reflects the precision of the optimization results. The magnitude of the step size Δr depends on the properties of the objective function that we want to minimize. The determination of Δr depends on some experimentation, as well as the decreasing speed of T_l. A good value for Δr is one that allows the algorithm to escape from a local minimum in a few steps, normally 2-3 steps (Bohachevsky et al., 1986). In our settings, the transformation matrix has unit length. So the value of Δr should not be larger than 1. We examined many values of Δr and found that as long as it is not too small (< 0.1) or too large (> 0.8), the results are not influenced much. We use $\Delta r = 0.5$ in our examples. In the SA-Sparse algorithm, the choice of ρ should be some number between $1/(dp)$ and $d/(dp)$, because we expect that there should be at least one element and at most d elements to be changed at each step. Setting it between $1/(dp)$ and $1/p$ will maintain an expected number of updates between 1 and d. In particular, in this paper, we choose $\rho = 1/p$ to obtain roughly the same degree of modification as with SA-Dense. In order to make comparisons

to the SA-Dense algorithm, we choose $\sigma = 10$ to make the transformation matrix relatively sparse. Note that by choosing different values of σ, the model can be set to different sparsity levels.

The initial T_0 should be high enough to make q_0 in Equation (4) lie between 0.5 and 0.9 (Bohachevsky et al., 1986). Hence $T_0 = 1$ is used. Note that the smaller the T_l, the smaller the probability of transition from a lower $\hat{e}$ state to a higher one. The constants b and L should be chosen so that q_l will be close to zero at the end of the process. This means near the end of the process almost all bad moves will be rejected and the search space will be located in a small region. As for the implementation of SA algorithms, $b = 0.9$ and $L = 30$ guarantee the procedure cools down gradually.

As for tuning parameters in the classification models, since our goal is to demonstrate that the proposed method dominates unsupervised methods such as conventional PCA and robust PCA given a specific classification strategy, we do not pay too much attention to the choice of tuning parameters for classification models. However, we examine several different combinations of parameters, and choose a roughly good set of parameters for the classification model.

3 Applications

Using data from two simulation studies and two real world studies, this section investigates the relative merits of the proposed method versus CPCA and a robust version of PCA, the ROBPCA method (Hubert et al., 2005). In order to show data reduction is prerequisite to classification in ultra high dimensional data, we also implement seven widely used classification methods: logistic regression (LR), support vector machine (SVM), k-nearest neighbor (kNN), neural network (NN), partial least squares regression (PLSR), margin tree (MT), and boosting, on the full dimensional data. Results show that the proposed method obtained better results than all other approaches on

most of the examples.

3.1 Simulated Data I

In this study, we start by simulating 20 training data sets with each having 100 observations from a multivariate g-and-h distribution with $\boldsymbol{g} = \boldsymbol{h} = [0.5, \ldots, 0.5]^T \in \mathbb{R}^d$, where $d = 50$. This is a skewed, heavy-tailed distribution that has a large departure from normality. In symbols, the marginal g-and-h distribution can be generated by

$$\tau_{g,h}(Z) = \begin{cases} \left(\frac{\exp(gZ)-1}{g}\right)\exp\left(\frac{hZ^2}{2}\right) & g \neq 0, \\ Z\exp\left(\frac{hZ^2}{2}\right) & g = 0 \end{cases} \tag{5}$$

where Z has a standard normal distribution. Let Σ be an arbitrary covariance matrix and $\boldsymbol{\mu}$ be an arbitrary location. The general multivariate g-and-h distribution is represented as

$$\mathbf{X} = \Sigma^{1/2}\boldsymbol{\tau}_{\boldsymbol{g},\boldsymbol{h}}(\mathbf{Z}) + \boldsymbol{\mu}, \tag{6}$$

where $\mathbf{Z} = [Z_1, \ldots, Z_d]^T$ and $\boldsymbol{\tau} = [\tau_{g_1,h_1}, \ldots, \tau_{g_d,h_d}]$. For more details about the g-and-h distribution, see Field and Genton (2006); Wilcox (2005).

In this simulation we let $\boldsymbol{\mu}$ be the zero vector. In order to introduce multicollinearity, we let the diagonal elements of Σ be 1 and the off-diagonal elements be 0.5. Furthermore, we rescaled the first $p = 5$ columns to have a standard deviation of 10. In principle PCAs should be able to extract linear combinations of the first p columns. The projection matrix obtained by PCAs should be similar to $[\mathbf{B}_1, \mathbf{0}_{p\times(d-p)}]^T$, where $\mathbf{B}_1 \in \mathbb{R}^{p\times p}$ is some full rank squared matrix. We create two scenarios, which match PCAs' working mechanism to different degrees.

In scenario 1, we design the true transformation matrix $\mathbf{A}_1 = [\mathbf{I}_p, \mathbf{0}]^T$. And the true reduced

data is denoted by $\mathbf{Z}_1$. It is easy to see that $\mathbf{Z}_1$ will only contain the information from the first p columns of $\mathbf{X}$. Presumably, PCAs should perform well on $\mathbf{Z}_1$, because all group information is associated with the first p variables having the most variability, and PCAs should be able to pick them out. While in scenario 2, we design the true transformation matrix $\mathbf{A}_2 = [\mathbf{0}_{p\times(d-p)}, \mathbf{B}_2]^T$ where $\mathbf{B}_2 \in \mathbb{R}^{p\times p}$ is a squared matrix with elements from $\mathscr{N}(0,1)$. $\mathbf{B}_2$ has been rescaled so that each column has length one. Then the reduced data $\mathbf{Z}_2$ only contains information from the last five columns with little variance. That is, the group information only resides in the last five variables with least data variance. CPCA and ROBPCA should perform poorly in this scenario, because PCA methods will still pick out the first five variables with the most variance, but there is no group information in these variables. We expect SA to be able to find better solutions in scenario 2.

In order to create group labels, we use a logistic regression model

$$\Pr(y_i = 1|X_i) = \frac{\exp(X_i^T \mathbf{A}\boldsymbol{\beta})}{1+\exp(X_i^T \mathbf{A}\boldsymbol{\beta})} = \frac{\exp(Z_i\boldsymbol{\beta})}{1+\exp(Z_i\boldsymbol{\beta})}, \tag{7}$$

where Z_i is the ith data point in the true reduced data, and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_p]$ with each element randomly generated from the uniform distribution $\mathscr{U}(-0.5, 0.5)$ for the first situation and from $\mathscr{U}(-2,2)$ for the second. We make this change in order to make the Bayes error remain around 0.1.

Two different classification methods are applied to the reduced data: LR and SVM. We examined different kernels and different margins for SVM, but the MCR is not sensitive to these tuning parameters. We use a radial kernel and a margin of 1 in this study. We examine $p = 2, 5, 10$, and 20. For each training data set, we generate a test sample with $n = 1000$ observations to evaluate the model performance. We evaluate methods based on the average test MCR over the 20 test data sets. Since the true p for these data is 5, we would expect worse results when $p = 2, 10, 20$.

Figure 1 shows the box plots for SA-Dense and SA-Sparse with different p's for both scenarios using LR. Similar results were obtained when SVM is used. As these box plots show, the proposed method gives the best classification accuracy when $p = 5$. The test errors produced by $p = 5$ are constantly smaller than using any other p's. This indicates the proposed method is able to produce a good classification accuracy when an appropriate number of dimensions is chosen.

We also computed the test MCR using the data reduction methods: CPCA and ROBPCA, and using seven popular classification methods: LR, SVM, kNN, NN, PLSR, MT and boosting, on the full dimensional data (FD). We intend to show that without data reduction, even an advanced classification technique may produce poor classification accuracy. The tuning parameters of these classification methods are selected roughly by examining several different sets of values and choosing the best set.

Tables 1 and 2 list the Bayes error rates, the average test MCR for CPCA, ROBPCA, and the two versions of the SA-based method when $p = 5$ and 10. The average test MCR for $p = 2$ and 20 cases are not reported here. Similarly to the $p = 5$ and the $p = 10$ cases, in the $p = 2$ and the $p = 20$ cases, CPCA and ROBPCA do not show any advantages over SA. In addition, the MCR for these two cases are generally higher than that for the case when $p = 5$.

As expected, PCAs perform well in scenario 1. In particular, ROBPCA does well on these data. The reason is that the data from this g-and-h distribution contain a large number of extreme values, for which CPCA would not be able to handle. However, SA methods perform as well as, if not better than, PCAs. In scenario 2, both SA-Sparse and SA-Dense outperform CPCA and ROBPCA significantly. In particular, SA-Sparse dominates others in both scenarios. This is because this experiment is a highly sparse case, which should favor SA-Sparse' working mechanism. Overall, SA methods are more stable over different conditions. Figure 2 displays the average MCR for SA-

Dense and SA-Sparse on the 20 training data sets as functions of the number of steps in scenario 1. Note the training MCR are obtained by performing a 10-fold CV on the training data as we described in Section 2.2. From these plots, we see that the MCR decreases rapidly and then levels off. Similar trends are observed in scenario 2 and hence the figures are omitted here.

Table 3 shows results for the seven classification methods on the full-dimensional data. As can be seen, most classification methods are unsatisfactory. MT performs the best among all these methods in both scenarios, but it is still worse the SA approach. This example is an intermediate data dimension example, which many advanced classification methods are able to handle. In Sections 3.3 and 3.4, we will give two real world examples on ultra-high dimensional data, where conventional classification methods have predictive difficulties.

Table 1: Average test MCR and standard errors on the 5-dimensional reduced data in Simulation I.

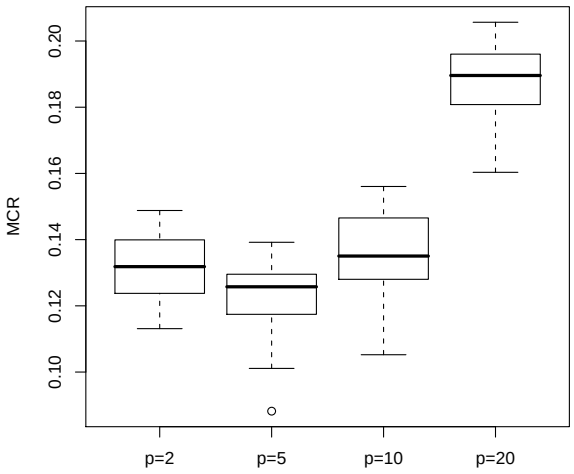
Scenario	Classifier	CPCA	ROBPCA	SA-Dense	SA-Sparse	Bayes
1	LR	0.132(0.010)	0.084(0.008)	0.125(0.008)	0.115(0.009)	0.061(0.003)
	SVM	0.168(0.015)	0.142(0.009)	0.138(0.008)	0.152(0.010)	
2	LR	0.464(0.009)	0.475(0.007)	0.320(0.010)	0.162(0.012)	0.083(0.004)
	SVM	0.469(0.008)	0.467(0.008)	0.358(0.009)	0.197(0.014)	

Table 2: Average test MCR and standard errors on the 10-dimensional reduced data in Simulation I.

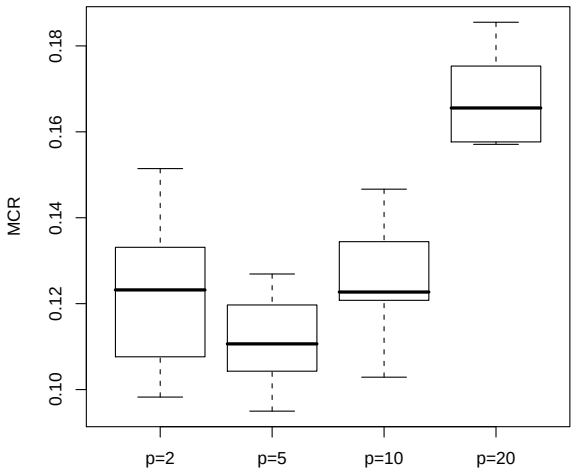
Scenario	Classifier	CPCA	ROBPCA	SA-Dense	SA-Sparse
1	LR	0.138(0.006)	0.102(0.004)	0.136(0.008)	0.120(0.006)
	SVM	0.209(0.007)	0.139(0.007)	0.142(0.009)	0.161(0.009)
2	LR	0.444(0.008)	0.442(0.009)	0.329(0.012)	0.233(0.013)
	SVM	0.448(0.007)	0.448(0.008)	0.371(0.006)	0.246(0.010)

Figure 1: Box plots for SA-Dense and SA-Sparse in Simulation I when $p = 2, 5, 10, 20$ using LR.

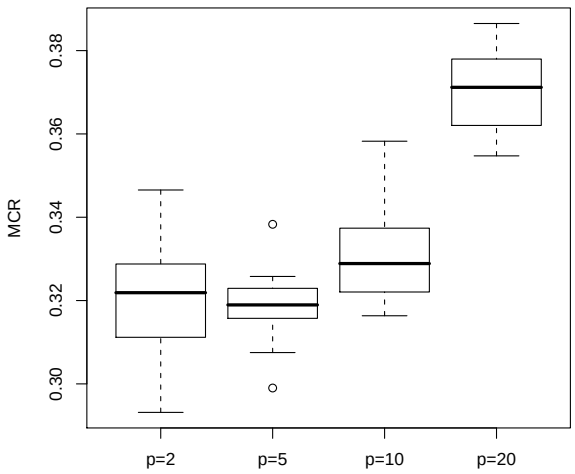
(a) SA-Dense in scenario 1



(b) SA-Sparse in scenario 1



(c) SA-Dense in scenario 2



(d) SA-Sparse in scenario 2

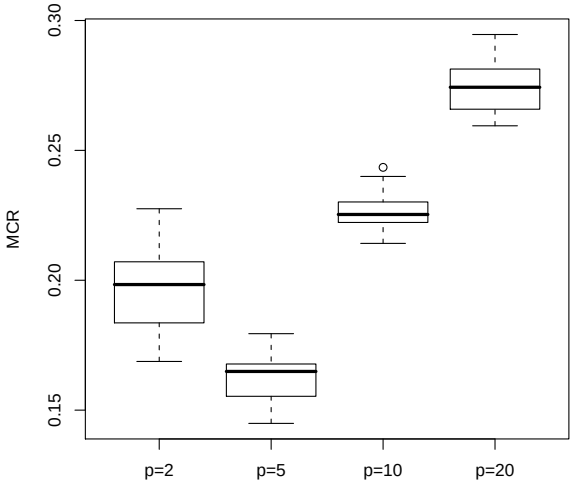
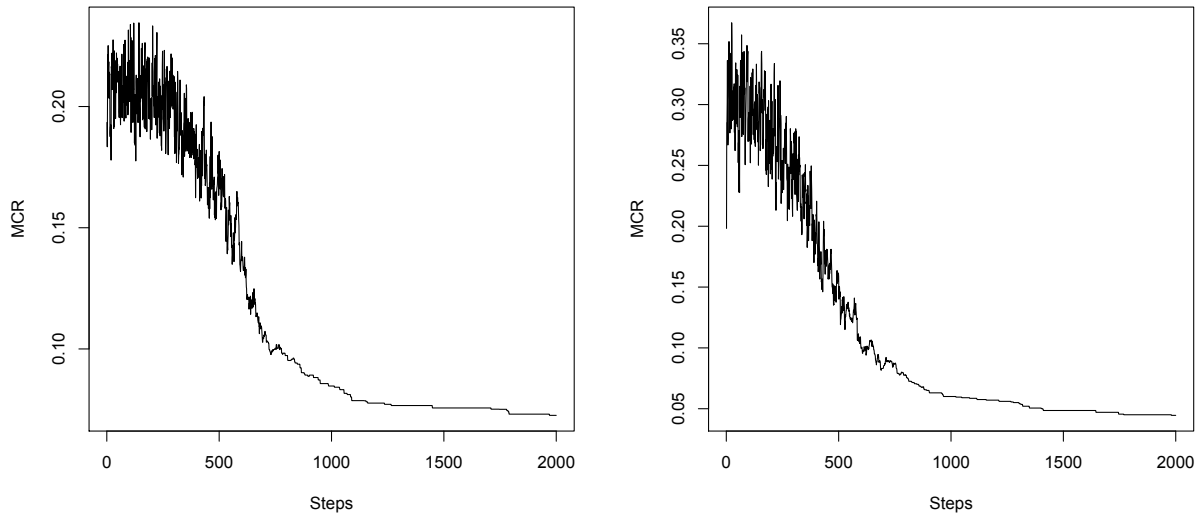


Table 3: Average test MCR and standard errors on the full dimensional data in Simulation I.

Scenario	LR	SVM	kNN	NN	PLSR	MT	Boosting
1	0.327 (0.008)	0.398 (0.007)	0.206 (0.009)	0.201 (0.010)	0.218 (0.009)	0.198 (0.005)	0.258 (0.024)
2	0.295 (0.009)	0.435 (0.011)	0.482 (0.003)	0.407 (0.006)	0.397 (0.006)	0.285 (0.008)	0.312 (0.006)

Figure 2: Average $\hat{e}_{CV}$ on training data using LR models in Simulation I Scenario 1.

(a) SA-Dense in scenario 1 (b) SA-Sparse in scenario 1



3.2 Simulated Data II

In this study, we create a dense situation where the data variance is equally distributed among all the variables. We simulated the input training data $\mathbf{X} \in \mathbb{R}^{100\times 50}$ from the same multivariate *g*-and-*h* distribution as in Section 3.1. Unlike the previous case, the elements of the 5-dimensional true $\mathbf{A}$ were generated from the standard normal distribution. This setting results in linear combinations of columns in $\mathbf{X}$. We use a non-linear LR model to generate group labels:

$$\Pr(y_i = 1|Z_i) = \frac{e^{g(Z_i)}}{1+e^{g(Z_i)}}. \tag{8}$$

where $g(Z_i) = \sin(0.05\pi Z_i)^T \boldsymbol{\beta}$, and elements of $\boldsymbol{\beta}$ are generated from $\mathscr{U}(-1,1)$. We use this non-linear LR to avoid the linear combinations of the columns of $\mathbf{Z}$ when generating group labels.

Since all the variables have roughly equal variances, PCAs should have difficulties accessing group information. Twenty pairs of training and test data sets with the same sizes as Simulation I are generated. We examined $p = 2, 5, 10, 20$. As with Simulation I, we report results from $p = 5, 10$ cases. The average Bayes rate, the average test MCR and standard errors on the reduced data are listed in Table 4. As we can see, both versions of the SA method outperform PCAs. Furthermore, SA methods perform better in the $p = 5$ case than $p = 10$ indicating that a correct dimension can be identified by SA. However, PCAs perform better in the $p = 10$ case. This is because since the data variance is equally distributed, more principal components provides better information to identify the groups. In particular, SA-Dense performs better than SA-Sparse in this simulation because the true transformation matrix is dense. It is interesting to see that the results from the LR classifier are better than that from SVM in general. One possible reason is that the curvature for the sine function that we used to introduce non-linearity is not too large. In addition, results from SVM may be improved by selecting the tuning parameters more carefully. Table 5 shows different classification methods on the full-dimensional data.

Table 4: Average test MCR and standard errors on the reduced data in Simulation II.

p	Classifier	CPCA	ROBPCA	SA-Dense	SA-Sparse	Bayes
5	LR	0.372(0.005)	0.360(0.005)	0.163(0.005)	0.177(0.006)	0.101(0.002)
	SVM	0.379(0.006)	0.365(0.006)	0.164(0.008)	0.189(0.009)	
10	LR	0.292(0.008)	0.294(0.008)	0.169(0.009)	0.185(0.009)	
	SVM	0.296(0.009)	0.294(0.008)	0.175(0.009)	0.197(0.010)	

Table 5: Average test MCR and standard errors on the full dimensional data in Simulation II.

LR	SVM	kNN	NN	PLSR	MT	Boosting
0.319	0.291	0.213	0.210	0.289	0.191	0.204
(0.006)	(0.005)	(0.005)	(0.007)	(0.007)	(0.005)	(0.008)

3.3 fMRI Data

Exploiting temporal information in fMRI has several difficulties: First, for whole brain studies, thousands of voxels can be involved. Second, the sample size is usually very small. Third, brains differ in subjects. Even for one subject, variations can occur in different runs. These differences lead to large inter-subject variability and intra-subject variability. Finally, the recording errors and the system noises can make classification tasks even more difficult. Consequently, most classification methods can not be readily applied, and data reduction steps are needed.

Our data were collected from the Image Center of the University of Southern California. The participant was shown four kinds of figures: faces, objects, scenes and scrambled images. The experiment consisted of eight blocks with each block showing one of the four figures for 12 time points followed by 12 time points of resting (baseline). The first eight time points were resting, and then from time point nine the experiment began. A total of 200 time points with eight blocks are available. Pre-processing was done by BrainVoyager and MATLAB 7.1. It included slice time correction, motion correction, highpass filter, linear trend removal and background removal. Finally, the 4-D image was expanded. A data matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ was obtained, where $n = 200$ represents the number of time points and $d = 11,383$ represents the number of voxels. Our goal is to do classification over time. That is, classify the time points into two classes: when the subject is doing the task and when the subject is resting.

To examine the relationship between $\hat{e}$ and p, we consider $p = 20, 50, 80, 100, 150$. SVM and kNN are applied in our SA algorithms. A radial kernel version of the SVM and a 3-nearest-

neighbor method seem to provide relatively good results. In addition, we randomize the 200 observations and take 150 for the training set, and the rest as the test set.

In order to compare the proposed method with conventional dimensionality reduction methods on classification tasks, CPCA and ROBPCA are also applied. The resulting test MCR for all four approaches are presented in Table 6. In general, the proposed method gives the best overall results: the test MCR are all smaller than those obtained from PCA when $p < 80$. For the two SA algorithms, one might not have strong preference for one or the other. Figure 3 displays $\hat{e}$ paths on training data for both SA-Dense and SA-Sparse using SVM, when $p = 50$. There are declines in $\hat{e}$'s for both algorithms. Similarly, the MCR paths using kNN have the same pattern as using SVM, so we do not present them here.

The estimated MCR from CPCA and ROBPCA are very similar. One reason might be that this data set does not contain many extreme values. When the goal is classification, both methods can fail. As we can see, the minimal test MCR for all methods are obtained when $p = 50$. As for the full-dimensional case (see Table 7), all methods fail, some methods such as LR, NN, PLSR, MT, and boosting, cannot be applied on our Dell Precision workstation (CPU 3.00GHz; RAM 2.99GHz, 16.0GB).

3.4 Microarray Data

Our second real-world example is a microarray data set. Usually microarray data sets include only a handful of observations, but several thousand variables, which necessitates a dimension reduction technique. A well-known microarray data set on Colon cancer (Alon et al., 1999) is used in our study. Colon adenocarcinoma tissues were collected from patients, paired normal colon tissue was also obtained. Gene expression in 40 tumor and 22 normal colon tissue samples was

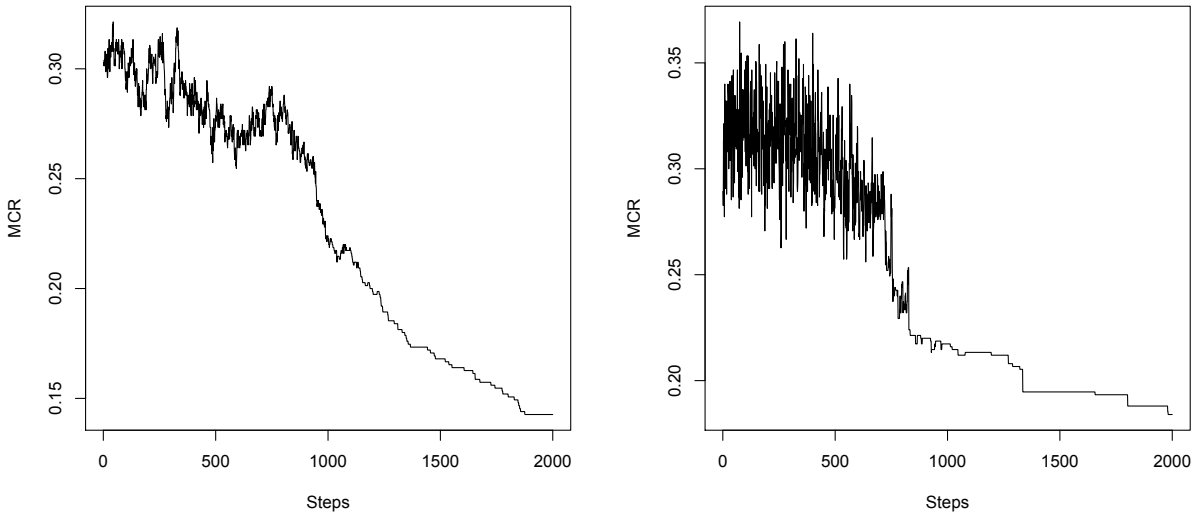
Table 6: Test MCR on fMRI data.

Classifier	p	CPCA	ROBPCA	SA-Dense	SA-Sparse
SVM	20	0.360	0.340	0.260	0.240
	50	0.220	0.240	0.200	0.200
	80	0.260	0.260	0.240	0.260
	100	0.320	/	0.340	0.360
	150	0.320	/	0.360	0.400
kNN	20	0.320	0.360	0.240	0.200
	50	0.340	0.420	0.160	0.200
	80	0.340	0.320	0.240	0.240
	100	0.340	/	0.320	0.260
	150	0.380	/	0.340	0.280

Table 7: Test MCR on the full dimensional data for fMRI and Colon data.

	LR	SVM	kNN	NN	PLSR	MT	Boosting
fMRI	/	0.42	0.46	/	/	/	/
colon	7/12	5/12	5/12	6/12	4/12	4/12	4/12

analyzed with a microarray consisting of more than 6500 human genes. The data set contains the expression of the 2000 genes with highest minimal intensity across the 62 tissue samples. Each gene intensity has been derived from about 20 feature pairs that correspond to the gene on the chip by using a filtering process. The goal is to classify the tissues as being cancerous or noncancerous. In previous studies, data reduction for gene microarray data had mostly focused on selecting a subset of relevant genes having higher variances than irrelevant ones (Dettling and Bühlmann, 2003; Hedenfalk et al., 2001). This is an example of selection technique. Although the microarray data have numerous genes, only a small number of genes are important in terms of classification (West et al., 2001). Selecting a small subset of genes will identify the ones most associated with colon cancer. However, as we mentioned earlier, a disadvantage of variable selection is that it is usually difficult to access the joint effect of genes when the data have multicollinearities. In our study, we use combinations of genes as data reduction for classification.



(a) SA-Dense (b) SA-Sparse

Figure 3: $\hat{e}$ paths using SVM when $p = 50$ on the fMRI data.

After a rough parameter investigation, SVM with radial kernels and kNN with $k = 5$ are applied in this example. The training set consists of 50 randomly selected tissues with the remainder used as the test set. We use $p = 10, 30, 50, 80$. Table 8 reports the test MCR for all four data reduction methods after applying SVM or kNN. Similarly to the fMRI example, it is found that SAs outperform other methods in general. We could roughly observe that the best p lies around 10 to 30. Note that the SVM models might not be the best ones for all the examples. Instead, our goal is to compare the performance of the four dimensionality reduction approaches under some given classification models. The full dimensional case (second row of Table 7) shows that even modern classification methods, such as MT and boosting, without dimension reduction perform worse than more classical methods with an appropriate dimension reduction. The $\hat{e}$ paths are also plotted in Figure 4. Similar to all previous studies, $\hat{e}$'s decrease rapidly.

Table 8: Test MCR on Colon data.

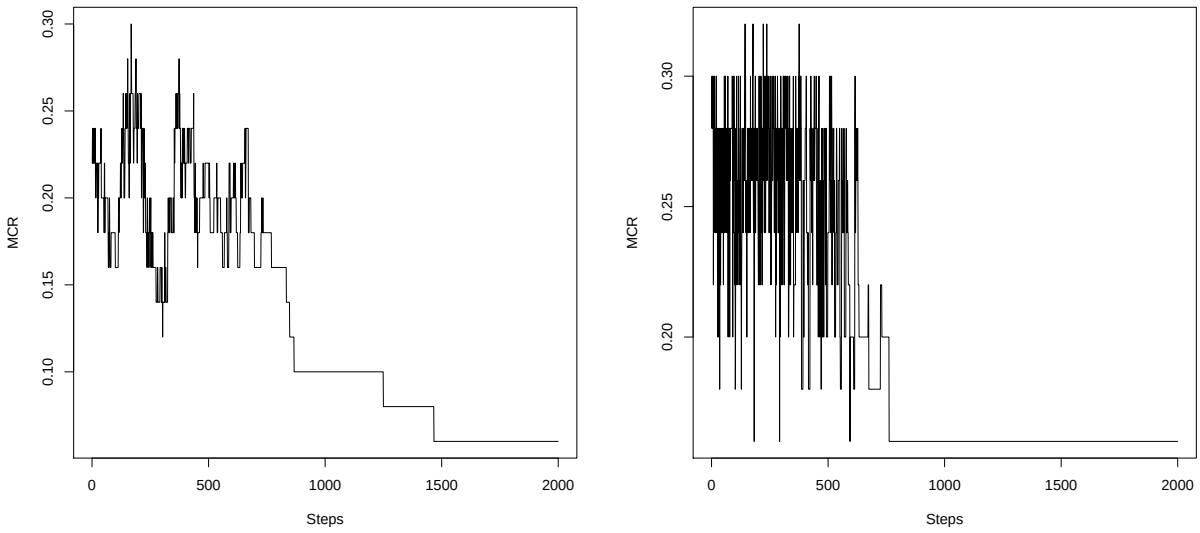
Classifier	p	CPCA	ROBPCA	SA-Dense	SA-Sparse
SVM	10	$4/12$	$4/12$	$2/12$	$3/12$
	30	$6/12$	/	$1/12$	$2/12$
	50	$7/12$	/	$1/12$	$2/12$
	80	/	/	$4/12$	$4/12$
kNN	10	$2/12$	$2/12$	$1/12$	$2/12$
	30	$2/12$	/	$2/12$	$2/12$
	50	$3/12$	/	$3/12$	$2/12$
	80	/	/	$5/12$	$4/12$

4 Investigating Convergence

Since SA is a stochastic based approach, solutions from different runs may be different. However, as long as they span the same space, they may achieve the same classification accuracy. In this section we examine the empirical convergence rate and the solution similarity.

4.1 Empirical Convergence

We used $I = 2000$ in our studies. One may question whether or not this number is large enough to achieve relatively good accuracy. We examine situations when $I = 20,000$, on Simulations 1 and 2. We use LR as the classifier and let $p = 5$ in both cases. The same data as in previous simulations are used in this study. That is, we use the same 20 training data sets and the same 20 test sets in each simulation study. Table 9 lists test MCR when $I = 2000$ and $I = 20,000$. As we can see, in all simulation studies, there are not big differences between $I = 2000$ cases and $I = 20,000$ cases. In particular, when $I = 20,000$ is applied, the test MCR are generally higher than the test MCR when $I = 2000$ is applied. This trend may suggest the problem of overfitting when I is too large. Since the algorithm itself is rather complex and there are several tuning parameters in the model, overfitting is a potential problem. This can be seen more obviously by plotting the training and test



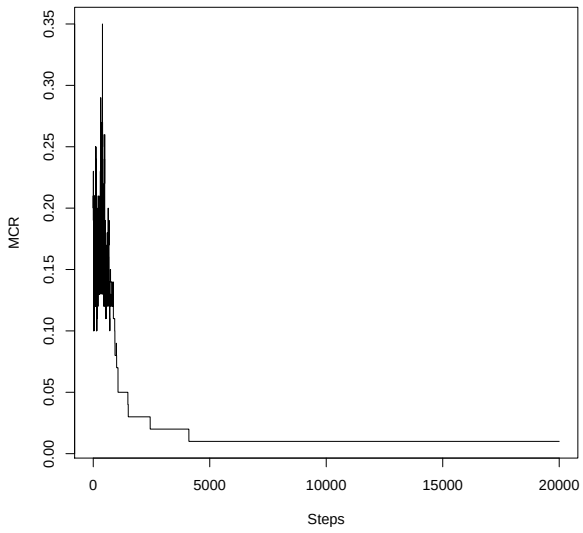
(a) SA-Dense (b) SA-Sparse

Figure 4: $\hat{e}$ paths using SVM when $p = 30$ on the microarray data.

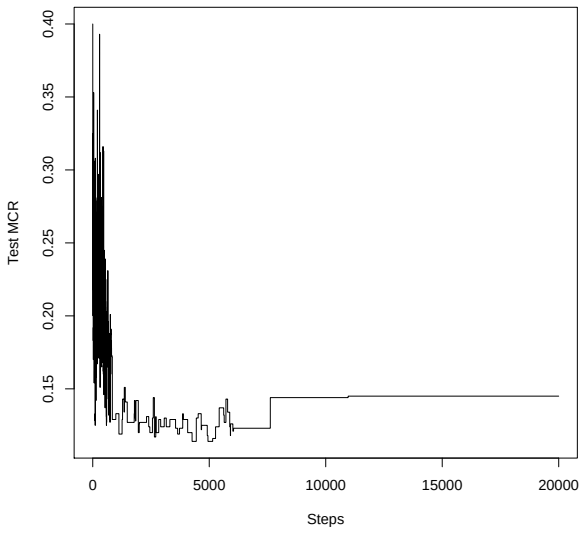
MCR paths over iterations.

Figure 5 (a) shows the CV error on the training data as a function of the number of iterations in Simulation I Scenario 1. Figures in other simulations are similar. The training MCR continuously goes down and then levels off after about 1000 steps. We also plot the test MCR path in Simulation I Scenario 1 [see Figure 5 (b)]. That is, at each iteration, we obtain an $\mathbf{A}^{new}$, we then apply this transformation matrix to the test data. Note that the testing step is an additional step for the purpose of visualization, and is not a part of the algorithm. As we can see, the test MCR decreases rapidly at the beginning, then it levels off, and it jumps to about 0.15 and levels off again. Overfitting may be avoided by applying a test step at each iteration and stop the algorithm when the test MCR reaches the lowest point. In our simulation studies, $I = 2000$ is a better number than $I = 20,000$.

We also investigate the computational cost of the proposed method by examining the CPU time given different values of d, I and p. The data we used are from Simulation I scenario 1 and the fMRI study in Section 3.3. Table 10 lists the CPU time in seconds for SA-Sparse. As we can see,



(a) SA-Sparse training MCR.



(b) SA-Sparse test MCR.

Figure 5: Training and test error paths using $I = 20,000$ in Simulation 1 (1).

the computation time linearly increases with I. When $I = 2000$, the computational cost for the proposed method are not very high, and even with $I = 20,000$ are acceptable.

Table 9: Test MCR on the simulated data.

Method	Sim I (1)		Sim I (2)		Sim II	
	$I = 2000$	$I = 20,000$	$I = 2000$	$I = 20,000$	$I = 2000$	$I = 20,000$
SA-Dense	0.125(0.008)	0.157(0.009)	0.320(0.101)	0.332(0.012)	0.163(0.005)	0.191(0.010)
SA-Sparse	0.115(0.009)	0.149(0.008)	0.162(0.012)	0.187(0.010)	0.177(0.006)	0.210(0.011)

4.2 Solution Similarity

Since there is randomness in our algorithm, one may be also interested in how the solutions from different runs differ from each other. Note that different transformation matrices can produce the same reduced subspace. Hence two $\mathbf{Z}$'s are equivalent as long as they have the same orthogonal bases even though they may be rotated differently. Therefore, we investigate solution stability by computing the principal components (PC) for different $\mathbf{Z}$'s and calculating their pairwise correla-

Table 10: CPU time (sec.) when running SA-Sparse with LR.

	$d = 50$		$d = 11,383$	
	$p = 5$	$p = 20$	$p = 10$	$p = 80$
$I = 2000$	80.28	118.20	175.75	796.34
$I = 20,000$	793.18	1171.12	1700.21	7642.31

tions. We define the average of absolute pairwise correlation for the kth PC, γ_k as

$$\gamma_k = \frac{1}{\binom{2}{N}} \sum_{i<j} |\xi_{i,j}|, \tag{9}$$

where $\xi_{i,j}$ is the correlation coefficient between the kth PC from the ith and the jth runs. We use γ_k as one of the criteria to evaluate the solution similarity. The larger the γ_k is, the more similar the solutions would be. Another criterion is the proportion of the data variance that a specific PC captures. If, for example, the kth PC from different runs captures 90% of the data variance on average, and its γ_k is 0.98, then one may conclude the solutions from different runs are similar to each other and the algorithm is stable.

We use two data sets from Simulation I Scenario 1 and Simulation II, respectively. For each data set, we run SA-Dense and SA-Sparse 50 times and obtain 50 estimated transformation matrices, and hence, 50 reduced subspaces. We compute the average of absolute pairwise correlations for the first PC, γ_1, because this PC captures the most data variance. Table 11 lists γ_1 and the average variance that PC1 captures over all 50 runs for Simulation I Scenario 1 and Simulation II. We examine the performance of the proposed method given different initial transformation matrices. First, we use the first p columns of the loading matrix from PCA for all 50 runs, and, second, we use randomly generated matrices for each run. Note that in the former case, the initial $\mathbf{A}^{(0)}$'s are all the same, but in the later case, the initial $\mathbf{A}^{(0)}$'s are different for the 50 runs. Our intention is to investigate if the choice of the starting point will affect the results. As we can see, in both

Simulations I and II γ_1 is considerably large (more than 0.9 in most cases) for PC1, which captures a large fraction of the data variance. This indicates that the solutions from different runs are similar to each other and the proposed algorithm is stable. It is not very surprising that the change of the starting point does not affect the results much.

Table 11: γ_1 (standard errors) and average percentage of data variance (standard errors).

$\mathbf{A}^{(0)}$	Method	Sim I (1)		Sim II	
		γ_1	% Variance	γ_1	% Variance
PC	SA-Dense	0.903(0.001)	73.8(0.010)	0.945(0.001)	82.5(0.011)
	SA-Sparse	0.956(0.001)	89.2(0.008)	0.897(0.002)	74.3(0.011)
Random	SA-Dense	0.907(0.002)	74.9(0.011)	0.935(0.001)	75.6(0.012)
	SA-Sparse	0.935(0.001)	85.2(0.010)	0.802(0.005)	68.8(0.013)

5 Conclusions

We have introduced a new approach for dealing with classification problems with high dimensional data. Because the commonly used data reduction methods, such as PCAs, are not designed for minimizing classification errors, they do not consider the response variable when reducing the data space. Therefore, the intermediate dimensionality reduction stage may remove some useful information, and their first directions might not (and in practice often will not) reveal the class structures that are needed for proper classification. Unlike PCA methods, the proposed method uses classification to guide dimensionality reduction. It projects the original data onto a lower-dimensional space with the goal of minimizing MCR. We have presented a simulated annealing algorithm for exploring a "good" projection using MCR from the classification model to guide the search path. In our simulation studies and real data experiments, it has demonstrated the ability to find a relatively good projection of the data.

One advantage of SA is that the number of steps required to find the solution does not increase

very rapidly with the dimensionality of the problem (Bohachevsky et al., 1986). In our experiments, the number of steps is 2000, which has been shown to be enough. Our results indicate that SA generally performs better than PCAs.

In this paper, we have only considered highly dense and highly sparse algorithms, which result in linear combinations of variables and subsets of input variables, respectively. The advantages and disadvantages of these two kinds of models are opposing. Dense models might deal well with multicollinearities, but fail to select informative variables. Moreover, it may be difficult to interpret the model itself. Sparse models might benefit from informative variables and better model interpretations, but may have trouble with multicollinearities. In our future work, we will investigate methods that lie between highly dense and highly sparse, and hence gain the benefits of both methods.

We have seen from Section 4 that the computational cost for the proposed method is higher than most of the deterministic methods (e.g., PCA), but it is acceptable. However, for many real-time applications, any stochastic search based method is hard to apply. This drawback may be cured by adding a preliminary reduction step before the SA algorithm is applied. This preliminary reduction step reduces the ultra-large data into an intermediate sized dimension. As Fan and Lv (2008) noted, as long as the intermediate space is not too small, the classification accuracy will not be significantly affected. Then the proposed method can be applied on this intermediate space without losing much classification information, while we can obtain a tremendous time saving. Further investigation is planned.

References

Alon, U., N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States 96*, 6745–6750.

Bohachevsky, I. O., J. M.E., and M. L. Stein (1986). Generalized simulated annealing for function optimization. *Technometrics 28*, 209–217.

Dettling, M. and P. Bühlmann (2003). Boosting for tumor classification with gene expression data. *Bioinformatics 19*, 1061–1069.

Fan, J. and J. Lv (2008). Sure independence screening for ultra-high dimensional feature space. *Journal of the Royal Statistical Society: Series B 70*, 849–911.

Field, C. and M. G. Genton (2006). The multivariate g-and-h distribution. *Technometrics 48*, 104–111.

Fix, E. and J. Hodges (1951). Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas.

Freund, Y. (2001). An adaptive version of the boost by majority algorithm. *Machine Learning 43*, 293–318.

Geman, S. and D. Geman (1984). Stochastic relaxation, gibbs distribution and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 721–741.

George, E. I. and R. E. McCulloch (1995). Stochastic search variable selection. *Practical Markov*

Chain Monte Carlo in Practice, 203–214. Eds: Gilks, W. R., Richardson, S. and Spiegelhalter, D. J.

Hedenfalk, I., D. Duggan, Y. Chen, M. Radmacher, M. Bittner, R. Simon, P. Meltzer, B. Gusterson, M. Esteller, O. Kallioniemi, B. Wilfond, A. Borg, and J. Trent (2001). Gene expression profiles distinguish hereditary breast cancers. *New England Journal of Medicine 244*, 539–548.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology 24*, 417–441.

Hubert, M., P. J. Rousseeuw, and K. Vanden Branden (2005). Robpca: A new approach to robust principal component analysis. *Technometrics 47*, 64–79.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. *Science 220*, 671–680.

Ntzoufras, I., J. J. Forster, and P. Dellaportas (1997). Stochastic search variable selection for log-linear models. Technical Report, Faculty of Mathematics, Southampton University, Southampton, UK.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine 2*, 559–572.

Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American Statistical Association 79*, 871–880.

Rousseeuw, P. J. (1985). Multivariate estimation with high breakdown point. *Mathematical Statistics and Probability B. Reidel: Dordrecht*, 283–297. Eds: Grossman, W., Pflug, G., Vincze, I. and Wertz, W.

Tadesse, M. G., N. Sha, and M. Vannucci (2005). Bayesian variable selection in clustering high-dimensional data. *Journal of the American Statistical Association 100*, 602–617.

Tibshirani, R. and T. Hastie (2007). Margin trees for high-dimensional classification. *Journal of Machine Learning Research 8*, 637–652.

Vapnik, V. (1998). *Statistical Learning Theory*. Wiley-Interscience.

West, M., C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. A. Olson, J. R. Marks, and J. R. Nevins (2001). Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences of the United States 98*, 11462–11467.

Wilcox, R. R. (2005). *Introduction to Robust Estimation and Hypothesis Testing*. San Diego: Academic Press. 2nd Edition.

Wold, H. (1966). Estimation of principal components and related models by iterative least squares. *Multivariate Analysis*, 391–420. Academic Press, New York.

Xie, Y., J. Wang, Y. Liang, X. Song, and R. Yu (1993). Robust principal components analysis by projection pursuit. *Journal of Chemometrics 7*, 527–541.

Yi, N., V. George, and D. B. Allison (2003). Stochastic search variable selection for identifying multiple quantitative trait loci. *Genetics 164*, 1129–1138.