Gene expression

Improved centroids estimation for the nearest shrunken centroid classifier

Sijian Wang¹ and Ji Zhu²,*

¹Department of Biostatistics and ²Department of Statistics, University of Michigan, Ann Arbor, MI, 48109, USA

Received on November 17, 2006; revised on January 19, 2007; accepted on February 4, 2007

Advance Access publication March 24, 2007

Associate Editor: John Quackenbush

ABSTRACT

Motivation: The nearest shrunken centroid (NSC) method has been successfully applied in many DNA-microarray classification problems. The NSC uses ‘shrunken’ centroids as prototypes for each class and identifies subsets of genes that best characterize each class. Classification is then made to the nearest (shrunken) centroid. The NSC is very easy to implement and very easy to interpret, however, it has drawbacks.

Results: We show that the NSC method can be interpreted in the framework of LASSO regression. Based on that, we consider two new methods, adaptive L∞-norm penalized NSC (ALP-NSC) and adaptive hierarchically penalized NSC (AHP-NSC), with two different penalty functions for microarray classification, which improve over the NSC. Unlike the L1-norm penalty used in LASSO, the penalty terms that we consider make use of the fact that parameters belonging to one gene should be treated as a natural group. Numerical results indicate that the two new methods tend to remove irrelevant genes more effectively and provide better classification results than the L1-norm approach.

Availability: R code for the ALP-NSC and the AHP-NSC algorithms are available from authors upon request.

Contact: jizhu@umich.edu

Supplementary information: Supplementary data are available at Bioinformatics online.

1 INTRODUCTION

Class prediction with high-dimensional microarray data has recently received much attention in many fields, such as bioinformatics, machine learning, medicine and statistics (Alizadeh et al., 2000; Dabney, 2005; Dudoit et al., 2002; Eisen et al., 1998; Golub et al., 1999; Hastie et al., 2001; Khan et al., 2001; Liu and Shen, 2006; Pan, 2002; Shen et al., 2006; Wu, 2006; Zhang et al., 2006a). It is considered very helpful for medical research if one can classify and predict the clinical category of a sample based on its gene expression profile. The microarray classification problem is a very challenging task, however, because there are a huge number of variables (genes) but a much smaller number of samples. Hence, finding relevant genes that distinguish samples is also greatly desired in practice.

Tibshirani et al. (2002) proposed the nearest shrunken centroid (NSC) method for class prediction in DNA-microarray studies. The NSC uses ‘shrunken’ centroids as prototypes for each class and identifies subsets of genes that best characterize each class. We describe the NSC algorithm briefly in the next subsection.

1.1 The nearest shrunken centroids method

Assuming we have n samples, and for each sample, we have expressions for p genes. Let \( x_{ij} \) be the expression for the \( j \)th gene and the \( i \)th sample. Each sample belongs to one of \( K \) classes \( 1, 2, \ldots, K \). Let \( C_k \) be the set of indices of the \( n_k \) samples in class \( k \). The \( j \)th component of the centroid for class \( k \) is \( \bar{x}_{kj} = \frac{1}{n_k} \sum_{i \in C_k} x_{ij} \), the mean expression in class \( k \) for gene \( j \); the \( j \)th component of the overall centroid is \( \bar{x}_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij} \).

Let

\[
\mu_{kj}^0 = \frac{\bar{x}_{kj} - \bar{x}_j}{m_k \cdot S_j},
\]

where, \( S_j \) is the pooled within-class SD for the \( j \)th gene:

\[
S_j^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i \in C_k} (x_{ij} - \bar{x}_{kj})^2,
\]

and \( m_k = \sqrt{1/n_k - 1/n} \).

The NSC shrinks each \( \mu_{kj}^0 \) to

\[
\hat{\mu}_{kj} = \text{sgn}(\mu_{kj}^0)(|\mu_{kj}^0| - \lambda)_+,
\]

where \( \lambda \) is a tuning parameter, and the shrunken centroid in class \( k \) for gene \( j \) is constructed as:

\[
\hat{x}_{kj} = \bar{x}_j + \hat{\mu}_{kj} \cdot m_k \cdot S_j.
\]

Because many of the \( \hat{x}_{kj} \) values are noisy and close to the overall mean \( \bar{x}_j \), soft thresholding usually produces more reliable estimates of the true means, and if \( \lambda \) is large enough, some of the \( \hat{\mu}_{kj} \) can be shrunk to zero, hence the corresponding \( \hat{x}_{kj} \) are equal to \( \bar{x}_j \).

For the classification of a test sample, suppose we have one with expression levels \( x^* = (x^*_1, x^*_2, \ldots, x^*_p) \). We define the discriminant score for class \( k \) as

\[
\delta_k(x^*) = \sum_{j=1}^{p} \left( x^*_j - \hat{x}_{kj} \right)^2 / S_j^2 - 2 \log \pi_k,
\]

where \( \pi_k \) is the prior probability of class \( k \).
where \( \pi_k = n_k/n \), and the classification rule is given by
\[
C(x^*) = k^*, \quad \text{where } k^* = \arg\min_k \delta_k(x^*).
\] (6)

So we can see, if for some \( j \), all \( \hat{\mu}_{kj}, k = 1, \ldots, K \), are zero, hence all \( \hat{\xi}_j \) are equal to \( \hat{x}_j \), then the \( j \)th gene will not contribute to the discriminant score, and it can be removed.

The NSC classifier is very easy to implement and very easy to interpret; it has been shown to be very successful in many high-dimensional classification problems. However, it has drawbacks.

In this article, we re-derive the NSC method as a LASSO regression on gene expression profiles. This re-interpretation allows us to notice that the \( L_1 \)-norm penalty used by NSC may not be the most effective way in analyzing microarray data. The \( L_1 \)-norm penalty treats all centroids equally or ‘flatly’, but centroids for the same gene are naturally ‘grouped’ together and intuitively should be considered as a group. Also, centroids for different genes, say relevant ones and irrelevant ones, should be treated differently. Enlightened by these observations, we consider two different penalty functions different from the \( L_1 \)-norm penalty to make use of natural grouping information within the data. As we will see in the numerical study, the methods we consider tend to remove irrelevant genes more effectively and provide better classification results.

The remainder of this article is organized as follows. In Section 2, we show how the NSC can be represented as a LASSO regression, and based on that, we consider two new methods: the adaptive \( L_{\infty} \)-norm penalized NSC (ALP-NSC) and the adaptive hierarchically penalized NSC (AHP-NSC), which improve over the NSC. In Section 3, we derive algorithms for the two methods in detail. Numerical results are in Sections 4 and 5. A brief discussion is in Section 6.

2 METHODS

2.1 LASSO interpretation of the nearest shrunk centroids

Assume that the observation \( x_i = (x_{ij1}, \ldots, x_{ijp}) \) from class \( k \) follows a multivariate normal distribution: \( \text{MVN}(\nu_k, \Sigma_k) \), where \( \nu_k = (\nu_{k1}, \ldots, \nu_{kp}) \) is the mean vector for class \( k \), and \( \Sigma_k \) is the covariance matrix for class \( k \). We further assume that \( \Sigma_k = \Sigma = \text{diag}(\sigma_1^2, \ldots, \sigma_p^2) \), i.e. the covariance matrices are the same across different classes and are diagonal. Such assumption is common when one works with high-dimensional low sample size data (Marron and Todd, 2002). Some theoretical justification for this assumption can be found in Bickel and Levina (2004).

We first center and scale each \( x_{ij} \) to be \( y_{ij} = (x_{ij} - \bar{x}_j)/(m_k \cdot s_j) \), and consider the linear regression:
\[
y_{ij} = \sum_{k=1}^{K} z_{ik} \mu_{kj} + \epsilon_{ij},
\] (7)
where \( \mu_{kj} = (v_{kj1} - \bar{x}_j)/(m_k \cdot s_j) \); \( z_{ik} \) is the indicator for whether the \( i \)th sample is in class \( k \), i.e. \( z_{ik} = 1 \) if the \( i \)th sample belongs to class \( k \), and \( z_{ik} = 0 \) otherwise; \( \epsilon_{ij} \) are independent of each other and approximately follow \( N(0, 1/m_k^2) \), if sample \( i \) belongs to class \( k \).

Now we consider an LASSO-(Tibshirani, 1996) type estimator for \( \mu_{kj} \):
\[
\hat{\mu}_{kj} = \arg\min_{\mu_{kj}} \sum_{i=1}^{n} \sum_{j=1}^{p} z_{ik} y_{ij} - \mu_{kj})^2 + \lambda \sum_{j=1}^{p} \sum_{k=1}^{K} |\mu_{kj}|.
\] (8)

After some algebra, one can show that the solutions to (8) are
\[
\hat{\mu}_{kj} = \text{sgn}\left( \sum_{i=1}^{n} z_{ij} y_{ij} \right) \left( \frac{\sum_{i=1}^{n} z_{ij} y_{ij}}{\sum_{i=1}^{n} z_{ij}^2} - \lambda \right)_+, \quad k = 1, \ldots, K; \ j = 1, \ldots, p
\] (9)
which matches exactly with (3). Therefore, the shrunk centroids used in the NSC can be considered as the solutions to (8). We acknowledge that Wu (2006) presented a similar interpretation of NSC, but used different values of \( \lambda \) for different classes.

We can see from (8) that by using the \( L_1 \)-norm penalty, the NSC shrinks \( \mu_{kj} \) continuously towards zero, and shrinks some of the fitted \( \mu_{kj} \) to be exactly zero when making \( \lambda \) sufficiently large. In order to remove the \( j \)th gene, we require all \( \mu_{kj}, \ k = 1, \ldots, K \), to be zero. However, we can also see from (8) that the \( L_1 \)-norm penalty treats all \( \mu_{kj} \) the same, i.e. it does not use the information that \( \mu_{kj} \) and \( \mu_{kj} \) are associated with the same gene \( j \). Intuitively, they belong to one ‘group’ and should be treated differently from \( \mu_{kj} \), which is associated with a different gene \( j \). In the next two subsections, we consider two different penalty functions, i.e. the \( L_{\infty} \)-norm penalty and the hierarchical penalty that incorporate this information into the modeling procedure. In general, we consider
\[
\hat{\mu}_{kj} = \arg\min_{\mu_{kj}} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{p} z_{ik} (y_{ij} - \mu_{kj})^2 + \lambda \cdot J(\Omega) \right\},
\] (11)
where \( \Omega = \{ \mu_{kj}, k = 1, \ldots, K; j = 1, \ldots, p \} \), and \( J(\Omega) \) is a penalty function.

2.2 Method I: the adaptive \( L_{\infty} \)-norm penalized NSC (ALP-NSC)

For the ALP-NSC, we consider to estimate \( \mu_{kj} \) by
\[
\hat{\mu}_{kj} = \arg\min_{\mu_{kj}} \left\{ \lambda \cdot J(\Omega) + \sum_{k=1}^{K} \max_{j=1}^{p} |\mu_{kj}| \right\},
\] (12)
where \( \max(|\mu_{t1}|, \ldots, |\mu_{tK}|) = |\{\mu_{t1}, \ldots, \mu_{tK}\}|_{\infty} \). Different from penalizing every \( \mu_{kj} \) individually, the \( L_{\infty} \)-norm penalizes the maximum absolute value of \( \mu_{kj}, k = 1, \ldots, K \), for the \( j \)th gene. If the maximum of \( |\mu_{kj}|, k = 1, \ldots, K \) is shrunk to zero, all \( \mu_{kj} \) are automatically shrunk to zero. The \( L_{\infty} \)-norm penalty has also been used in Zhang et al. (2006b), Zhao et al. (2006) and Zou and Yuan (2006) for other supervised problems.

To further improve the model (12), we borrow the adaptive idea from Shen and Ye (2002) and Zou (2006), i.e. to penalize different genes differently. We consider
\[
\max_{\mu_{kj}} \left\{ \lambda \cdot J(\Omega) + \sum_{k=1}^{K} w_k \cdot \max_{j=1}^{p} |\mu_{kj}| \right\},
\] (13)
where \( w_k \) are pre-specified weights. The intuition is that if the \( j \)th gene is relevant for distinguishing different classes from each other, we would like the corresponding \( w_k \) to be small, hence the \( j \)th gene is lightly penalized, while if the \( j \)th gene is irrelevant and expressed similarly...
across different classes, we would like the corresponding \( w_j \) to be large, hence the \( j \)th gene is heavily penalized. How to pre-specify \( w_j \) from the data will be discussed in Section 2.4.

### 2.3 Method II: the adaptive hierarchically penalized NSC (AHP-NSC)

The \( L_1 \)-norm penalty makes use of the information that \( \mu_{kj} \) and \( \mu_{kj} \) are associated with the same gene by shrinking the maximum absolute value of \( \mu_{kj} \) within the \( j \)th gene. If we denote \( M_j = \max(\{\mu_{kj}, \ldots, \mu_{kj}\}) \), the corresponding \( L_\infty \)-norm penalty on the \( j \)th gene is \( \lambda M_j \), and we can write \( \mu_{kj} = M_j \theta_{kj} \), where \( -1 \leq \theta_{kj} \leq 1 \). This motivates us to reparameterize \( \mu_{kj} \) in a more general way:

\[
\mu_{kj} = y_j \theta_{kj}, \quad j = 1, \ldots, p; \quad k = 1, \ldots, K,
\]

where \( y_j \geq 0 \) (for identifiability reasons). Notice that here \( y_j \) plays a similar role as \( M_p \) but it does not have to be the maximum of \( \{\mu_{kj}\} \); similarly \( \theta_{kj} \) does not have to be bounded between \(-1\) and \( 1 \). This decomposition reflects the information that \( \mu_{kj} \), \( k = 1, \ldots, K \), all belong to a single gene \( x_j \), by treating each \( \mu_{kj} \) hierarchically. \( y_j \) is at the first level of the hierarchy, controlling \( \mu_{kj}, k = 1, \ldots, K \), as a group; \( \theta_{kj} \)s are at the second level of the hierarchy, reflecting differences within the \( j \)th gene.

To estimate \( y_j \) and \( \theta_{kj} \), we consider

\[
\min_{y_j, \theta_{kj}} \left( \ell(\Omega) + \lambda_y \sum_{j=1}^p y_j + \lambda_\theta \sum_{j=1}^p \sum_{k=1}^K |\theta_{kj}| \right),
\]

subject to \( y_j \geq 0 \). Notice that there are two tuning parameters \( \lambda_y \) and \( \lambda_\theta \), \( \lambda_\theta \) controls the estimates at the gene-specific level, and it can effectively remove irrelevant genes: if \( y_j \) is shrunk to zero, all \( \mu_{kj} \) for the \( j \)th gene will be equal to zero. \( \lambda_y \) controls the estimates at the class-specific level: if \( y_j \) is not equal to zero, some of the \( \theta_{kj} \) hence some of the \( \mu_{kj} \), \( k = 1, \ldots, K \), still have the possibility of being zero; in this sense, the hierarchical penalty shares some of the properties of the \( L_1 \)-norm penalty.

The adaptive idea in (13) also applies here. If the \( j \)th gene is relevant, we would like to penalize its \( y_j \) and \( \theta_{kj} \) lightly, and if the \( j \)th gene is irrelevant, we would like to penalize its \( y_j \) and \( \theta_{kj} \) heavily. Hence, we consider the AHP-NSC:

\[
\min_{y_j, \theta_{kj}} \left( \ell(\Omega) + \lambda_y \sum_{j=1}^p y_j + \lambda_\theta \sum_{j=1}^p \sum_{k=1}^K w_{kj} |\theta_{kj}| \right),
\]

where \( w_j \) and \( w_{kj} \) are pre-specified weights.

### 2.4 Computing the adaptive weights

Regarding the adaptive weights \( w_j \) in (13), \( w_j \) and \( w_{kj} \) in (16), following Breiman (1995) and Zou (2006), we can choose them using the un-penalized estimates \( \hat{\mu}_{kj}^0 \). Specifically:

\[
M_j^0 = \max(\{\mu_{kj}^0, \ldots, \mu_{kj}^0\}),
\]

\[
w_j = 1/M_j^0,
\]

\[
w_{kj} = 1/M_j^0,
\]

\[
w_{kj} = 1/|\mu_{kj}^0|.
\]

### 3 ALGORITHMS

In this section, we describe details of our algorithms for estimating \( \mu_{kj} \) in ALP-NSC and AHP-NSC. Notice that both the \( L_\infty \)-norm penalty and the hierarchical penalty have non-differential points, so they pose optimization challenges.

#### 3.1 Estimating \( \mu_{kj} \) in ALP-NSC

In ALP-NSC, (13) can be decomposed into \( p \) separate minimization problems

\[
\min_{\mu_{kj}, kj} \frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K \frac{1}{nk} (y_{ij} - \mu_{kj})^2 + \lambda \cdot w_j \cdot \max(|\mu_{kj}|, \ldots, |\mu_{kj}|),
\]

(17)

where \( j = 1, \ldots, p \). For each \( j \), (17) can be transformed into a quadratic programming problem:

\[
\min_{\mu_{kj}, kj} \frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K \frac{1}{nk} (y_{ij} - \mu_{kj})^2 + \lambda \cdot w_j \cdot M_j
\]

subject to \( -M_j \leq \mu_{kj} \leq M_j, \quad k = 1, \ldots, K \)

(19)

\( M_j \geq 0 \)

(20)

Hence, most commercially available packages can be used to solve it.

We have also explored explicit forms for the solutions to (17), which help us gain more insights into the nature of the \( L_\infty \)-norm penalty. We can show that \( \hat{\mu}_{kj} \), the solution to the minimization problem (17), can be achieved by shrinking an average of \( \mu_{kj} \) (1), i.e. the solution to (17) when there is no penalty (or \( \lambda = 0 \)).

**Theorem 1.** For the \( j \)th minimization problem (17), if there exists an indices set \( C = [k_1, \ldots, k_r] \), such that

\[
|\hat{\mu}_{k_1}| = \cdots = |\hat{\mu}_{k_r}| > |\hat{\mu}_{kj}|, \quad \text{for } k \notin C
\]

(21)

then

\[
\hat{\mu}_{kj} = \begin{cases} \mu_{kj}, & k \notin C \\ \text{sgn}(\mu_{kj}^0) \left[ \frac{\sum_{j=1}^p |\mu_{kj}^0| - \lambda w_j}{\lambda w_j} \right] + k \in C \end{cases}
\]

(22)

where \( \text{sgn}(\cdot) \) is the positive part of the argument.

Details of the proof are in the Supplementary Material. From Theorem 1, we can see when there are \( r \) maximums among \( |\hat{\mu}_{kj}| \), only the corresponding \( \mu_{kj} \) will be shrunk by the \( L_\infty \)-norm penalty, and they are shrunk to the same absolute value. This value is based on an average of \( \mu_{kj} \) of the corresponding \( r \) classes. We can also see that if the \( j \)th gene is irrelevant and all \( |\mu_{kj}| \) are close to zero, then the \( L_\infty \)-norm penalty tends to shrink all of them to zero (with an appropriately chosen \( \lambda w_j \)).

To implement Theorem 1, we need to decide \( r \), the number of maximums among \( |\hat{\mu}_{kj}| \), and the set \( [k_1, \ldots, k_r] \), which indicates which \( r \mu_{kj} \) should be shrunk. When \( K \) is not very large, say \( K \leq 20 \), we can use an exhaustive search to find \( r \) and \( [k_1, \ldots, k_r] \), i.e. for each \( 1 \leq r \leq K \), we search over all possible sets \( [k_1, \ldots, k_r] \). For each possible set, we estimate \( \hat{\mu}_{kj} \) using (22), then check whether the estimates satisfy the assumption (21). If the assumption is satisfied, we compute the corresponding value for the objective function (17). Finally, we choose
\( \hat{\mu}_{kj} \) that give the smallest value for the objective function. When \( K \) is large, we will resort to the quadratic programming (18)–(20).

### 3.2 Estimating \( \mu_{kj} \) in AHP-NSC

In AHP-NSC (16), we can use an iterative approach to estimate \( \gamma_j \) and \( \theta_{kj} \), i.e. we first fix \( \theta_{kj} \) and estimate \( \gamma_j \), then we fix \( \gamma_j \) and estimate \( \theta_{kj} \), and we iterate between these two steps until the solution converges. Since at each step, the value of the objective function (16) decreases, the solution is guaranteed to converge. We have the following theorem that helps us solve for \( \gamma_j \) and \( \theta_{kj} \) at each step.

**Theorem 2.**

- When \( \theta_{kj}, j = 1, \ldots, p \) and \( k = 1, \ldots, K \), are fixed,
  \[ \hat{\gamma}_j = \Pi_{(\xi_k, \theta_{kj} \neq 0)} \left( \sum_{k=1}^{K} \xi_k \frac{\mu_{kj}}{\theta_{kj}} - \frac{\lambda \gamma^j w_j}{\gamma^j} \right), \tag{23} \]

  where \( \xi_k = \theta_{kj}^2 \sum_{k=1}^{K} \theta_{kj}^2 \).

- When \( \gamma_j, j = 1, \ldots, p \), are fixed,
  \[ \hat{\theta}_{kj} = \Pi_{(\gamma_j > 0)} \cdot \text{sgn}(\mu_{kj}) \left( \frac{\mu_{kj}}{\gamma_j} \frac{\lambda \gamma^j w_j}{\gamma^j} \right) \tag{24} \]

Equations (23) and (24) show that both \( \hat{\gamma}_j \) and \( \hat{\theta}_{kj} \) are soft-thresholding estimates. Details of the proof are in the Supplementary Material. Here we give some intuitive explanation.

We first look at \( \hat{\gamma}_j \) (23). If all \( \theta_{kj} \) are zero, it is natural to estimate \( \gamma_j \) also to be zero because of the penalty on \( \gamma_j \). If not all \( \theta_{kj} \) are zero, then from our reparameterization, we have \( \gamma_j = \mu_{kj}/\theta_{kj}, 1 \leq s \leq r \). Plugging in \( \mu_{kj} \) for \( \mu_{kj} \), we obtain \( r \) estimates for \( \gamma_j \): \( \hat{\gamma}_j = \mu_{kj}/\theta_{kj}, 1 \leq s \leq r \). A natural estimate for \( \gamma_j \) is then a weighted average of the \( \hat{\gamma}_j \), and Equation (23) provides such a (shrunken) average, with weights proportional to \( \xi_k \).

Now considering \( \hat{\theta}_{kj} \) (24). If \( \gamma = 0 \), it is natural to estimate all \( \theta_{kj} \) also to be zero because of the penalty on \( \theta_{kj} \). When \( \gamma > 0 \), we have \( \hat{\theta}_{kj} = \mu_{kj}/\gamma_j \). Again, plugging in \( \mu_{kj} \) for \( \mu_{kj} \), we obtain \( \hat{\theta}_{kj} = \mu_{kj}/\gamma_j \). Equation (24) shrinks \( \hat{\theta}_{kj} \) and the amount of shrinkage is inversely proportional to \( \gamma_j^2 \). When \( \gamma_j \) is large, which indicates the \( j \)th gene is relevant, the amount of shrinkage is small, while when \( \gamma_j \) is small, which indicates the \( j \)th gene is less relevant, the amount of shrinkage is large.

### 4 SIMULATION STUDY

In this section, we use simulated data to demonstrate our methods ALP-NSC and AHP-NSC, and compare the results with that of the NSC. We also compare our methods with an improved version of NSC, i.e. NSC with adaptive choice of thresholds (Tibshirani et al., 2003). The ‘adaptive’ in Tibshirani et al. (2003) has a different meaning from that in our two methods: it still treats different genes ‘flatly’, but uses large thresholds for classes easy to classify and small thresholds for classes difficult to classify.

We first considered a two-class classification scenario. There were a total of \( p = 10000 \) variables with only the first 20 relevant while the other 9980 irrelevant in forming two classes. Specifically, the first 20 variables were i.i.d. \( N(0, 1) \) for the first class and i.i.d. \( N(1, 1) \) for the second class, whereas the remaining 9980 variables were all i.i.d. \( N(0, 1) \) for both classes.

We generated \( n = 100 \) training observations, with 70 in the first class and 30 in the second one; similarly, we also generated 1000 test observations, with the same class prior and the same within-class distribution. We denote this as the ‘70-30’ example. Tuning parameters were chosen using 5-fold cross-validation (CV) on the training data. We then computed the misclassification error rate of the chosen model on the test data. We also recorded both the number of relevant variables and the number of irrelevant variables that were selected. We repeated this 100 times. The results are summarized in Table 1.

As we can see, our ALP-NSC and AHP-NSC methods performed similarly to the NSC method in terms of selecting relevant variables, but tended to keep much fewer irrelevant variables. The error rates of ALP-NSC and AHP-NSC also seemed to be smaller than that of the NSC.

We then considered two three-class classification scenarios, with highly unbalanced classes. In both scenarios, there were a total of \( p = 4000 \) variables with the first 2 relevant while the other 3998 irrelevant in forming three classes. The first 2 variables were i.i.d. from \( N(0, 1) \) for the first class, i.i.d. \( N(2.5, 1) \) for the second class and i.i.d. \( N(5, 1) \) for the third class, whereas the remaining 3998 variables were all i.i.d. from \( N(0, 1) \) for all three classes. In the first scenario, we generated 20 observations for each of the first class and the third class, and 100 for the second class We denote it as the ‘20-100-20’ example. In the second scenario, we generated 100 observations for each of the first class and the third class, and 20 for the second class. We denote it as the ‘100-20-100’ example. For each scenario, we also generated test observations with the same class prior and the same within-class distribution, but with the sample size 10 times larger than that of the training datasets. We repeated this 100 times. The results are summarized in Table 2.

### Table 1. Simulation results for the ‘70-30’ example

<table>
<thead>
<tr>
<th>Method</th>
<th># Info</th>
<th># Non-info</th>
<th>CV error</th>
<th>Test error</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSC-No-Noise</td>
<td>–</td>
<td>–</td>
<td>0.03 (0.01)</td>
<td></td>
</tr>
<tr>
<td>NSC</td>
<td>19 (1)</td>
<td>90 (89)</td>
<td>0.10 (0.03)</td>
<td>0.09 (0.02)</td>
</tr>
<tr>
<td>NSC-Ada</td>
<td>19 (1)</td>
<td>90 (89)</td>
<td>0.10 (0.03)</td>
<td>0.09 (0.02)</td>
</tr>
<tr>
<td>ALP-NSC</td>
<td>18 (2)</td>
<td>39 (38)</td>
<td>0.04 (0.02)</td>
<td>0.04 (0.01)</td>
</tr>
<tr>
<td>AHP-NSC</td>
<td>16 (3)</td>
<td>10 (8)</td>
<td>0.00 (0.01)</td>
<td>0.04 (0.02)</td>
</tr>
</tbody>
</table>

# Info’ is the average number of selected relevant variables (out of 20) over 100 repetitions. ‘# Non-info’ is the average number of irrelevant variables (out of 9980) that were kept. ‘CV error’ is the average misclassification error rate in 5-fold cross-validation. ‘Test error’ is the average misclassification error rate on test data. The numbers in the parentheses are the corresponding SDs. ‘NSC-No-Noise’ is to apply the NSC method to the dataset with only the first 20 relevant variables, and its ‘Test Error’ can be considered as an ‘oracle’ benchmark. ‘NSC-Ada’ refers to the NSC with adaptive thresholds.
As we can see, our ALP-NSC and AHP-NSC methods removed all 3998 irrelevant variables for every repetition (out of 100), and the classification error rates were just slightly higher than that of the NSC using only the first 2 relevant variables, i.e., the ‘oracle’. In contrast, the NSC method and the NSC with adaptive thresholds tended to select more noise variables and have higher error rates.

5 REAL DATA ANALYSIS

In this section, we apply the ALP-NSC and the AHP-NSC methods to three gene microarray datasets.

The first dataset we considered is the Leukemia dataset in Golub et al. (1999). This dataset consists of 38 training data and 34 test data for two types of acute leukemia, acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). Each sample is a vector of \( p = 7129 \) genes. We applied the ALP-NSC and the AHP-NSC methods to the training data. Tuning parameters were chosen using 10-fold CV, and the chosen models were evaluated on the test data. The results are summarized in the upper part of Table 3. Both the ALP-NSC and the AHP-NSC had two misclassification on the test data, and they selected 16 exactly same genes. Figure 1 shows the corresponding heatmap. Clear separation of the two classes is evident.

The second dataset we considered consists of microarray experiments of small round blue cell tumors (SRBCT) of childhood cancer (Khan et al., 2001). The tumors are classified as Burkitt lymphoma (BL), Ewing sarcoma (EWS), neuroblastoma (NB) or rhabdomyosarcoma (RMS). A total of 63 training samples and 20 test samples were provided. Each sample consists of expression measurements on \( p = 2308 \) genes. We analyzed this dataset in the similar way as with the Leukemia dataset. Tuning parameters were chosen using 8-fold CV. The results are summarized in Table 3. The CV errors and the test errors are all zero. The ALP-NSC selected 38 genes, and the AHP-NSC selected 40 genes. The 38 genes selected by ALP-NSC and the 40 genes selected by AHP-NSC have 34 overlapping genes. Figures 2 and 3 show the heatmaps of the selected genes. Similar as in Figure 1, genes that distinguish each class from other classes are also evident.

To further assess the genes that were selected from the Leukemia and the SRBCT datasets, we randomly split each dataset into the training and the test sets for 100 times. The sizes of the two sets and the class priors were kept the same as the original training/test split. The results are summarized in Tables 4–6. For the Leukemia dataset, out of the 16 genes selected from the original training/test split, 10 genes were selected for more than 70 times out of the 100 random training/test splits. For the SRBCT dataset, out of the 44 genes that were selected by ALP-NSC and AHP-NSC from the original training/test split, 29 genes were
selected for more than 70 times out of the 100 random splits. These results imply that the highly frequently selected genes may have certain power in differentiating the corresponding cancer classes from each other.

The Leukemia and the SRBCT datasets are relatively ‘easy’ for classification. We also considered the NCI-60 dataset (Dudoit et al., 2002). In this study, cDNA microarrays were used to examine the variation in gene expressions among 60 cell lines from the National Cancer Institute’s anti-cancer drug screen. The cell lines are derived from tumors with eight different sites of origin: breast, central nervous system (CNS), colon, leukemia, melanoma, non-small-cell-lung-carcinoma (NSCLC), ovarian and prostate. A total of 61 samples were provided. Each sample consists of expression measurements on p = 5244 genes. The class sizes are all small, and some of the classes (e.g. breast and NSCLS) are known to be heterogeneous. So this is a more difficult dataset for classification than the Leukemia and the SRBCT datasets. The NCI-60 dataset came without pre-specified training/test split, so we randomly split the data into the training and the test sets comma; with sample sizes 40 and 21, respectively. We repeat it 100 times. The results are summarized in the lower part of Table 4 and Figure 4. We can see that ALP-NSC and AHP-NSC had similar misclassification errors as NSC and adaptive-NSC, but selected much fewer genes. Then, 777 genes were selected for more than 70 times out of the 100 trials. Our methods selected on average about 1000 genes, while NSC selected more than 3000 genes.

6 CONCLUSION
In this article, we first re-interpreted the popular NSC method in the framework of LASSO regression. Based on the penalized linear regression framework, we have considered two new methods for microarray classification, which improve over the NSC. Unlike the $L_1$-norm penalty used in LASSO, the penalty terms that we consider make use of the fact that parameters belonging to one gene should be treated as a natural group. We have presented some evidence that the two new methods tend to remove irrelevant genes more effectively and provide better classification results than the $L_1$-norm approach.

ACKNOWLEDGEMENT
We would like to thank the Associate Editor and two reviewers for their thoughtful and useful comments. We would also like to thank Trevor Hastie and Rob Tibshirani for helpful discussions. S.W and J.Z are partially supported by grant DMS-0505432 from the National Science Foundation and grant NHLBI-18 HL60884 from National Institutes of Health.

Conflict of Interest: none declared.
Table 5. The 16 genes that were selected by the ALP-NSC and the AHP-NSC from the Leukemia dataset

<table>
<thead>
<tr>
<th>Gene accession number</th>
<th>Selection frequency</th>
<th>Gene description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X95735_at</td>
<td>100</td>
<td>Zyxin</td>
</tr>
<tr>
<td>M23197_at</td>
<td>98</td>
<td>CD33 CD33 antigen (differentiation antigen)</td>
</tr>
<tr>
<td>M63136_at</td>
<td>98</td>
<td>CTSD Cathepsin D (lysosomal aspartyl protease)</td>
</tr>
<tr>
<td>M27891_at</td>
<td>97</td>
<td>CST3 Cystatin C (amyloid angio pathy and cerebral hemorrhage)</td>
</tr>
<tr>
<td>M16038_at</td>
<td>97</td>
<td>LYN V-yes-1 Yamaguchi sarcoma viral-related oncogene homolog</td>
</tr>
<tr>
<td>X17042_at</td>
<td>91</td>
<td>PRG1 Proteoglycan 1, secretory granule</td>
</tr>
<tr>
<td>M84526_at</td>
<td>90</td>
<td>DF D component of complement (adipin)</td>
</tr>
<tr>
<td>M55150_at</td>
<td>85</td>
<td>FAH Fumarylacetoacetate</td>
</tr>
<tr>
<td>U50136_rna1_at</td>
<td>80</td>
<td>Leukotriene C synthase (LTC4S) gene</td>
</tr>
<tr>
<td>M62762_at</td>
<td>75</td>
<td>ATP6C Vacuolar H+ ATPase proton channel subunit</td>
</tr>
<tr>
<td>Y00787_s_at</td>
<td>53</td>
<td>INTERLEUKIN-8 PRECURSOR</td>
</tr>
<tr>
<td>U82759_at</td>
<td>51</td>
<td>GB DEF = Homeodomain protein HoxA9 mRNA</td>
</tr>
<tr>
<td>U46751_at</td>
<td>35</td>
<td>Phosphotyrosine independent ligand p62 for the Lek SH2 domain mRNA</td>
</tr>
<tr>
<td>M80254_at</td>
<td>29</td>
<td>PEPTIDYL-PROLYL CIS-TRANS ISOMERASE, MITOCHONDRIAL PRECURSOR</td>
</tr>
<tr>
<td>L08246_at</td>
<td>27</td>
<td>INDUCED MYELOID LEUKEMIA CELL DIFFERENTIATION PROTEIN MCL1</td>
</tr>
<tr>
<td>Y12670_at</td>
<td>25</td>
<td>LEPR Leptin receptor</td>
</tr>
</tbody>
</table>

‘Selection frequency’ is the number of times that the corresponding gene was selected out of 100 random splits of the training and testing data.

Table 6. The 44 genes that were selected by the ALP-NSC and the AHP-NSC from the SRBCT dataset

<table>
<thead>
<tr>
<th>Unique gene ID</th>
<th>Gene symbol</th>
<th>Selection frequency</th>
<th>Unique gene ID</th>
<th>Gene symbol</th>
<th>Selection frequency</th>
<th>Unique gene ID</th>
<th>Gene symbol</th>
<th>Selection frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hs.1517</td>
<td>WAS</td>
<td>100</td>
<td>Hs.356717</td>
<td>MYL4</td>
<td>99</td>
<td>Hs.408222</td>
<td>PBX1</td>
<td>62</td>
</tr>
<tr>
<td>Hs.349109</td>
<td>IGFG1</td>
<td>100</td>
<td>Hs.221889</td>
<td>CSDA</td>
<td>99</td>
<td>Hs.65029</td>
<td>GSI</td>
<td>46</td>
</tr>
<tr>
<td>Hs.74034</td>
<td>CAV1</td>
<td>100</td>
<td>Hs.74376</td>
<td>OLFM1</td>
<td>99</td>
<td>Hs.351279</td>
<td>HLA-DMA</td>
<td>43</td>
</tr>
<tr>
<td>Hs.80205</td>
<td>PIM2</td>
<td>100</td>
<td>Hs.274520</td>
<td>ANTXR1</td>
<td>98</td>
<td>Hs.236361</td>
<td>RNP C1</td>
<td>38</td>
</tr>
<tr>
<td>Hs.251664</td>
<td>–</td>
<td>100</td>
<td>Hs.254321</td>
<td>CTNNA1</td>
<td>97</td>
<td>Hs.169907</td>
<td>GSTA4</td>
<td>37</td>
</tr>
<tr>
<td>Hs.283477</td>
<td>CD99</td>
<td>100</td>
<td>Hs.902</td>
<td>NF2</td>
<td>97</td>
<td>Hs.312102</td>
<td>ALDH7A1</td>
<td>32</td>
</tr>
<tr>
<td>Hs.75823</td>
<td>AF1Q</td>
<td>100</td>
<td>Hs.124030</td>
<td>ELF1</td>
<td>96</td>
<td>Hs.268515</td>
<td>MN1</td>
<td>22</td>
</tr>
<tr>
<td>Hs.99931</td>
<td>SGCA</td>
<td>100</td>
<td>Hs.170548</td>
<td>YAP1</td>
<td>94</td>
<td>Hs.75216</td>
<td>PTPR B</td>
<td>21</td>
</tr>
<tr>
<td>Hs.387553</td>
<td>PTPN13</td>
<td>100</td>
<td>Hs.407132</td>
<td>TNNT2</td>
<td>93</td>
<td>In multiple clusters –</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Hs.105434</td>
<td>ISG20</td>
<td>100</td>
<td>Hs.50649</td>
<td>TP513</td>
<td>88</td>
<td>Hs.121576</td>
<td>MYO1B</td>
<td>13</td>
</tr>
<tr>
<td>Hs.111903</td>
<td>FCGRT</td>
<td>100</td>
<td>Hs.407546</td>
<td>TNAFAP6</td>
<td>83</td>
<td>Hs.386092</td>
<td>NCOA1</td>
<td>5</td>
</tr>
<tr>
<td>Hs.74050</td>
<td>FVT1</td>
<td>100</td>
<td>Hs.14601</td>
<td>HCLS1</td>
<td>80</td>
<td>Hs.25363</td>
<td>PSEN2</td>
<td>2</td>
</tr>
<tr>
<td>Hs.165950</td>
<td>FGF4</td>
<td>100</td>
<td>Hs.380757</td>
<td>GYG2</td>
<td>77</td>
<td>Hs.91747</td>
<td>PFN2</td>
<td>2</td>
</tr>
<tr>
<td>Hs.15463</td>
<td>FNDC5</td>
<td>100</td>
<td>Hs.440459</td>
<td>MEST</td>
<td>76</td>
<td>Hs.62</td>
<td>PTPN12</td>
<td>2</td>
</tr>
<tr>
<td>Hs.388761</td>
<td>CDK6</td>
<td>100</td>
<td>Hs.334131</td>
<td>CDH2</td>
<td>62</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

‘Selection frequency’ is the number of times that the corresponding gene was selected out of 100 random splits of the training and testing data.

Fig. 4. Selection frequency for ALP-NSC and AHP-NSC out of 100 trials for the NCI-60 dataset.

REFERENCES

Zhang, H. et al. (2006b) Variable selection for multicategory SVM via sup-norm regularization. Institute of Statistics Mimeo Series 2596, NCSU.