An experimental comparison of gene selection by Lasso and Dantzig selector for cancer classification

Songfeng Zheng, a,*, Weixiang Liu b

a Department of Mathematics, Missouri State University, 901 S. National Ave., Springfield, MO 65897, USA
b Biomedical Engineering Lab, School of Medicine, Shenzhen University, Shenzhen, Guangdong 518060, China

1. Introduction

Since cancer develops a cell’s genetic structure due to mutations to cells with uncontrolled growth patterns, extensive research has shown that we can identify cancer by looking at the genetic level [1,2,13,18,23,24,26,27,32,34,35]. In a microarray study we usually shown that we can identify cancer by looking at the genetic level to cells with uncontrolled growth patterns, extensive research has shown that we can identify cancer by looking at the genetic level [1,2,13,18,23,24,26,27,32,34,35]. In a microarray study we usually obtain the expression levels of different genes, based on which the decision can be made regarding whether or not this is a patient with cancer.

DNA microarray technique allows us to access the expression profiles of thousands of genes. However, due to the high cost of experiments, we usually have a limited number (50–100) of examples. A classifier built on a small training set with very high dimension is prone to overfitting. Fortunately, it is well-known that most genes are irrelevant to the cancer classification problem [38]. Thus, identifying the genes with strong discriminative power is an important step to effective classification based on the gene expression data.

Gene selection methods based on singular value decomposition [3] or principle component analysis [39] do not use class label information, thus the selected genes might be less effective for classification. To select genes with high discriminative power, most commonly used methods employ class label information, and assign each gene a score based on how well the expression of this gene can discriminate different classes and then select the genes with high scores. Popular score functions include correlation coefficients [13], fold change [4,23], absolute expression level [2,18,24,34], standard deviation [5,34], consistency in repeated data [5,23], correlation coefficient with class labels [24,32], t/F-statistics [12,21], Wilcoxon’s rank sum test statistics [7,8], the ratio of between-group over within-group sum square [9], partial least-squares [21], to name a few.

The aforementioned methods select genes one by one, and thus share some disadvantages. First, redundant genes may be selected because the mutual information between genes is not considered. Second, the interactions between genes are omitted [9,14]. In order to avoid selecting redundant genes, some methods have been introduced which can select a set of genes simultaneously. For example, Guyon et al. [14] proposed Recursive Feature Elimination method for Support Vector Machine, which recursively eliminates irrelevant genes for classification so that the survived genes are expected to have strong discriminative power.

In regression analysis, it is often very important to identify informative variables in order to explain the obtained model. In the ordinary least-square regression, if we impose 1 constraint on the regression coefficients, some of the regression coefficients in the model will shrink to zeros, thus we can automatically select a set of informative variables simultaneously (i.e. the ones with non-zero coefficients). This technique is called Lasso (least absolute shrinkage and selection operator) [30]. Dantzig selector [6] is a similar technique for linear regression model: instead of minimizing the squared error, it minimizes the norm of the gradient vector of the squared error function. With the 1 constraint on the regression coefficients, the majority of the estimated regression coefficients by Dantzig selector are exactly zeros. As such, Dantzig...
2. Lasso and Dantzig selector for variable selection

Suppose we have dataset \(\{\mathbf{x}_i, y_i\}, i = 1, \ldots, N\), with the predictor \(\mathbf{x} \in \mathbb{R}^p\) and the response \(y \in \mathbb{R}\). Without loss of generality, we assume the predictors and the response are centered, and the predictors are standardized, that is \[\sum_{i=1}^{N} y_i = 0, \quad \sum_{i=1}^{N} x_{ij} = 0 \quad \text{and} \quad \sum_{i=1}^{N} x_{ij}^2 = 1. \tag{1}\]

Consider the linear regression model \(y = \mathbf{x} \beta + \varepsilon\), where \(\beta = (\beta_1, \ldots, \beta_p)'\) is the vector of regression coefficients.

To fit the linear regression model, the ordinary least-square (OLS) estimates are obtained by minimizing the residual squared error, i.e.
\[\hat{\beta}_{\text{OLS}} = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - \mathbf{x}_i \beta)^2 = \arg\min_{\beta} \|\mathbf{Y} - \mathbf{X} \beta\|_{L_2},\]
where \(\mathbf{Y} = (y_1, \ldots, y_N) \in \mathbb{R}^N\) is the response vector, \(\mathbf{X} \in \mathbb{R}^{N \times p}\) is the design matrix, and \(\cdot \|_{L_2}\) represents the \(L_2\) norm, i.e., for vector \(\mathbf{r} = (r_1, \ldots, r_p)\), \[\|\mathbf{r}\|_{L_2} = \sum_{i=1}^{p} r_i^2.\]

However, when the dimensionality \(p\) is large, the OLS estimate \(\hat{\beta}_{\text{OLS}}\) has a large portion of non-zero components, making it difficult to interpret the obtained model.

2.1. Lasso

Lasso [30] is a technique to fit the linear regression model which minimizes the residual squared error with a constraint on the sum of the absolute value of the regression coefficients, i.e.
\[\hat{\beta}_{\text{Lasso}} = \arg\min_{\beta} \|\mathbf{Y} - \mathbf{X} \beta\|_{L_2} \quad \text{s.t.} \quad \sum_{j=1}^{p} |\beta_j| \leq t. \tag{2}\]

The Lasso penalty in Eq. (2) shrinks the fitted coefficients \(\hat{\beta}\) toward zero. If we make \(t\) in Eq. (2) small, some of the estimated coefficients \(\hat{\beta}_j\)'s will be exactly zeros. Thus, by tuning the parameter \(t\) in Lasso, we can automatically select informative variables (i.e. the ones with nonzero regression coefficients).

Efron et al. [10] proposed Least Angle Regression (LARS) and derived that Lasso is a special case of LARS with simple modification. As analyzed in [10], when there are far more variables than examples, i.e. \(p \gg N\), LARS selects at most \(N\) variables with \(O(N^3)\) operations assuming the training examples are not centralized, otherwise LARS terminates with \(N-1\) variables. In this paper, we have far more genes than examples, thus \(p \gg N\) is our interested scenario, and we will adopt LARS to find the Lasso solution.

2.2. Dantzig selector

The Dantzig selector [6] estimator of the regression coefficient vector \(\beta\) is the solution to
\[\min_{\beta \in \mathbb{R}^p} \sum_{j=1}^{p} |\beta_j| \quad \text{s.t.} \quad \|\mathbf{X}(\mathbf{Y} - \mathbf{X} \beta)\|_{L_2} \leq s, \tag{3}\]
where \(\|\cdot\|_{L_2}\) represents the \(L_2\) norm, i.e., for vector \(\mathbf{r} = (r_1, \ldots, r_p)\), \[\|\mathbf{r}\|_{L_2} = \max_{1 \leq i \leq p} |r_i|.\]
The optimization in Eq. (3) can be recast as a linear programming problem [6]:
\[\min_{\beta \in \mathbb{R}^p} \sum_{j=1}^{p} u_j \quad \text{s.t.} \quad -u \leq \beta \leq u \quad \text{and} \quad -s \leq \mathbf{X}(\mathbf{Y} - \mathbf{X} \beta) \leq s, \tag{4}\]
where \(u = (u_1, \ldots, u_p)\) with \(u_i \geq 0\), and \(1\) is a \(p\)-dimensional vector of all ones. The above minimization problem produces a large part of estimated coefficients to be exactly 0 in a similar fashion as Lasso and hence can be used as a variable selection tool. Candès and Tao [6] have provided strong theoretical justification for this property, and the Dantzig selector has shown impressive empirical performance on simulated data and real world problems involving large values of \(p\) [6,17].

The definition of Dantzig selector can be re-expressed as [11]
\[\hat{\beta}_{\text{DS}} = \arg\min_{\beta} \|\mathbf{X}(\mathbf{Y} - \mathbf{X} \beta)\|_{L_2} \quad \text{s.t.} \quad \sum_{j=1}^{p} |\beta_j| \leq s. \tag{6}\]
Comparing the minimization problem for Lasso in Eq. (2) and that for Dantzig selector in Eq. (6), we can see that with the bound on the absolute sum of the coefficient vector \(\beta\), Lasso minimizes the squared error function while Dantzig selector minimizes the maximum component of the gradient of the squared error function. The difference in the problem formulations makes Lasso and Dantzig selector select different variables, and it is claimed in [11,20] that Lasso has slight advantage over Dantzig selector on variable selection for linear regression model. The current paper attempts to compare the two variable selection methods for pattern classification problems.

3. Classifiers

Suppose we have a dataset \(\{\mathbf{x}_i, y_i\}, i = 1, \ldots, N\), where \(\mathbf{x}_i \in \mathbb{R}^d\) and \(y_i \in \{0, 1\}\). The purpose of supervised learning is to fit a classifier from the training set and apply it to the unseen testing set. In this paper, we consider three linear classifiers, which include Linear Regression as Classifier, linear Support Vector Machines (SVM), and Logistic Regression; we will also apply the selected genes to nonlinear SVM with Gaussian kernel.

3.1. Linear regression as classifier

In the binary classification problems, similar to [29], we treat the class label \(y_i \in \{0, 1\}\) as the probability of \(\mathbf{x}_i\) being a positive
example, thus $y_i$ can take continuous values in $[0, 1]$. We then fit a linear regression model to represent $y_i$ (the probability of being positive) as a linear function of $x_i$. In the testing stage, we apply the threshold value 0.5 to the predicted value for obtaining the class label.

We notice that [15] suggests name the class label as $y_i \in \{1, -1\}$ and treat it as continuous variable, and use the sign of the predicted value as the class label. There is no difference between the 0/1 labeling system and the $\pm 1$ naming system in the final results because there is a linear relation between them. However, the probabilistic interpretation of the 0/1 labeling system is more natural and easier to understand.

### 3.2. Support vector machines

In the formulation of Support Vector Machine (SVM), the class label is named as $y_i \in \{1, -1\}$ for convenience. SVM regards the linear classifier $\text{sign}(\mathbf{x}^\top \mathbf{w} + b)$ as a hyperplane in the p-dimensional space, and chooses the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is clearly of interest and is known as the maximum-margin hyperplane. When the training set is not separable, SVM will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples. The method introduces slack variable $\xi_i$, which measures the degree of misclassification of the datum $x_i$. The optimization becomes a trade-off between a large margin, and a small error penalty. If the penalty function is linear, the optimization problem becomes

$$\min_{\mathbf{w}, \xi} \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^{N} \xi_i$$

subject to $y_i(\mathbf{x}_i^\top \mathbf{w} + b) \geq 1 - \xi_i$ for $1 \leq i \leq N$,

where $\xi = (\xi_1, \ldots, \xi_N)^\top$, and $C$ is a parameter to balance the margin of the hyperplane and the errors made on the training set. The optimization can be solved by the dual problem via quadratic programming:

$$\max_{\mathbf{a}} \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j y_i y_j x_i^\top x_j$$

subject to $ \sum_{i=1}^{N} a_i y_i = 0$ and $0 \leq a_i \leq C$ for $1 \leq i \leq N$,

where $\mathbf{a} = (a_1, \ldots, a_N)$ with $a_i$ being the Lagrangian multiplier of the $i$th constraint in Eq. (8). Finally, the coefficient vector of the linear classifier can be calculated by

$$\mathbf{w} = \sum_{i=1}^{N} a_i y_i \mathbf{x}_i,$$

and $b$ can be estimated from the set of support vectors (with $a_i \neq 0$).

Linear SVM can be extended to nonlinear case by using the kernel trick. Let $\phi(x)$ be the mapped feature vector for $x$ in the reproducing kernel Hilbert space, and let $k(x, x_i) = \phi(x)^\top \phi(x_i)$ be the kernel function. Nonlinear SVM is learned by fitting the following maximization problem:

$$\max_{\mathbf{a}} \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j y_i y_j k(x_i, x_j),$$

with the same constraints as in Eq. (10). The final nonlinear SVM classifier has the form

$$\hat{y} = \text{sign}\left(\sum_{i=1}^{N} a_i y_i k(x_i, x) + b\right),$$

where $\hat{y}$ is the predicted label for the testing object $x$.

### 3.3. Logistic regression

For a predictor/feature vector $x$, we can directly model the posterior probability of the class label $y \in \{0, 1\}$ as logistic response function

$$p(y = 1|x) = \frac{\exp(w_0^\top x)}{1 + \exp(w_0^\top x)} = p(x; w_0).$$

The log likelihood function of $\{(x_i, y_i), i = 1, \ldots, N\}$ is

$$l(w_0; \mathbf{x}) = \sum_{i=1}^{N} y_i \log\left(p(x_i; w_0)\right) + \left(1 - y_i\right)\log\left(1 - p(x_i; w_0)\right).$$

The parameter $(w_0, \mathbf{w})$ can be obtained by maximizing the log likelihood function $l(w_0, \mathbf{w})$, and the prediction could be made by looking at the sign of $w_0 + x^\top \mathbf{w}$.

### 4. Experiments

This section applies Lasso and Dantzig selector to select the most informative genes for the classifiers introduced in Section 3, and reports the results on six public cancer datasets.

#### 4.1. The datasets

We perform binary classification on six datasets which include DLBCL, Leukemia, Prostate, Colon, Lymphoma, and Estrogen. Table 1 summarizes the basic information about the datasets, and we provide the descriptions of the datasets and the preprocessing procedures as follows:

**DLBCL**: The DLBCL dataset [26] contains in total 77 examples in two classes, diffuse large B-cell lymphomas (DLBCL) and follicular lymphoma (FL) which have 58 and 19 examples, respectively. The original dataset contains 7129 genes. We thresholded the intensities at 20 and 16,000 units, then we filtered out genes with max = min = 9, $\text{min} / \text{max} \leq 3$ or max- min $\leq 100$. After preprocessing, we obtained a dataset with 77 examples and 6285 genes.

**Leukemia**: This dataset was first presented in [13], which contains gene expression levels of 72 patients of which 47 with acute lymphoblastic leukemia (ALL) and 25 with acute myeloid leukemia (AML). Following the preprocessing strategy in [9], we processed them by thresholding, filtering and logarithm transformation, and standardizing each tissue example to have zero mean and unit variance across the genes. The processed data finally contain the expression values of 3571 genes.

**Prostate**: The Prostate dataset [27] contains in total 102 examples in two classes, i.e. tumor and normal, which have 52 and 50 examples, respectively. The original dataset contains 12,600 genes. In our experiment, intensities were thresholded at 100 and 16,000 units. Then we filtered out the genes with max/min $\leq 5$ or max- min $\leq 50$. After preprocessing, we obtained a dataset with 102 examples and 5966 genes. The DLBCL, Leukemia, and Prostate datasets are available in MATLAB format at [http://www.biomedcentral.com/1471-2105/7/228/additional/](http://www.biomedcentral.com/1471-2105/7/228/additional/).

**Colon**: In this dataset, expression levels of 40 tumors and 22 normal colon tissues for 6500 genes were measured using the Affymetrix gene chip technique. A subset of 2000 genes with

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DLBCL</th>
<th>Leukemia</th>
<th>Prostate</th>
<th>Colon</th>
<th>Lymphoma</th>
<th>Estrogen</th>
</tr>
</thead>
<tbody>
<tr>
<td># of genes</td>
<td>6285</td>
<td>3571</td>
<td>5966</td>
<td>2000</td>
<td>4026</td>
<td>7129</td>
</tr>
<tr>
<td># of positive</td>
<td>19</td>
<td>25</td>
<td>52</td>
<td>22</td>
<td>62</td>
<td>25</td>
</tr>
<tr>
<td># of negative</td>
<td>58</td>
<td>47</td>
<td>50</td>
<td>40</td>
<td>34</td>
<td>24</td>
</tr>
</tbody>
</table>
highest minimal intensity across the examples has been selected in [2], and the selected subset can be downloaded from http://microarray.princeton.edu/oncology/affydata/index.html. We further process the data as in [8]: applying a logarithm transformation and standardizing each tissue example to have zero mean and unit variance across the genes.

**Lymphoma:** The Lymphoma dataset [1] contains 62 malignant and 34 normal examples, and each example consists of 4026 gene expression measures. The data are preprocessed to give each feature mean zero and unit standard deviation. This dataset is made publicly available in MATLAB format by [36] at http://www.kyb.tuebingen.mpg.de/bs/people/weston/l0/.

**Estrogen:** The Estrogen dataset was first presented in [35], and is available at http://data.cgt.duke.edu/west.php. The dataset contains 7129 gene expression values obtained by applying the Affymetrix gene chip technique to 49 breast tumor examples. We thresholded the raw data with a floor of 100 and a ceiling of 16,000 and then applied a logarithm transformation. Finally, each

![Graphs showing average testing error curves](image.png)

**Fig. 1.** Average testing error curves with the number of selected genes on the DLBCL, Leukemia, Prostate, and Colon datasets.
tissue example was standardized to have zero mean and unit variance across the genes. The response variable describes the status of the estrogen receptor (ER), and 25 examples are ER+, whereas the remaining 24 examples are ER−.

4.2. Results of linear classifiers

For each dataset, we randomly select 80% of the examples as training set, and the remaining 20% are used for the testing purpose. In the gene selection stage, we let the response 

\[ y_i = \begin{cases} 1 & \text{if the } i\text{th patient has cancer}, \\ 0 & \text{otherwise.} \end{cases} \]

As discussed in Section 3.1 (also refer to [29]), we understand the response \( y_i \) as the probability of the \( i\)th example being cancerous. On the training set, we apply Lasso and Dantzig selector to regress \( y_i \) on the gene data, which allows us to select informative genes for expressing the probability of being cancerous with the gene expression data by a linear equation. We then feed the expression values of the selected genes to the training algorithms of the three linear classifiers introduced in Section 3. In the testing stage, only the expression levels of the selected genes are used for decision making. The partition-selection-training-testing process is repeated 100 times.

If Lasso and Dantzig selector pick the same set of genes, the performances of a specific classifier on the gene set selected by Lasso and that selected by Dantzig selector are expected to be the same. In the case of Lasso and Dantzig selector select different sets of genes, this paper studies which gene set can give us better classification result, thus compares the gene selection ability of Lasso and Dantzig selector. Starting with \( t = 0 \) or \( s = 0 \), i.e., no gene is selected, we gradually increase the parameters. The continuity of the solution paths of Lasso and Dantzig Selector [11,20] indicates that the number of selected genes increases gradually. That is, if the number of selected genes is fixed, tuning the parameter might improve the performance of the regression, but cannot change the selected genes, thus has no effect on the final classification result because the classifiers are trained on the selected gene set. As such, it is not necessary to tune the parameter (s or t) as long as the number of selected genes by Lasso or Dantzig selector is fixed.

For each number of selected genes, we calculate the average testing error rate for every classifier on each of the six datasets. Figs. 1 and 2 show the average testing error rate curves of the 100 runs for every classifier on the six datasets. From the figures, we observe that on particular datasets (e.g. Colon), for particular classifiers, Dantzig selector has advantage when the number of selected genes is in a certain range, while in all other cases, Lasso based classifiers have better performance in terms of average testing error rate. This shows that in general, compared to Dantzig selector, Lasso is more efficient at picking informative genes for linear classifier.

4.3. Results of nonlinear support vector machines

We also ran the experiment in Section 4.2 with nonlinear SVM with Gaussian kernel function

\[ k(x_i, x_j) = \exp \left( -\frac{|x_i - x_j|^2}{2\sigma^2} \right). \]

with \( \sigma = 1 \). As in the experiments in Section 4.2, we fix the penalty parameter \( C = 15 \) in the nonlinear SVM for a fair comparison to linear SVM. Fig. 3 shows the average testing error curves of the 100 runs with different number of selected genes. From Fig. 3, we observe that, same as for linear classifiers, in general, the...
genes selected by Lasso are more informative than those selected by Dantzig selector for nonlinear SVM.

Comparing the testing error curves of nonlinear SVM in Fig. 3 to those of linear SVM in Figs. 1 and 2, we see that for the same set of selected genes by Lasso or Dantzig selector, on Leukemia dataset, nonlinear SVM has slightly better performance than linear SVM, but on other datasets, the performances of the two types of classifiers are similar.

In summary, from the performance curves on six public datasets with three linear classifiers and nonlinear SVM, we conclude that in general, the genes selected by Lasso are more informative than those selected by Dantzig selector for cancer classification.

4.4. Comparing Lasso and Dantzig selector to elastic net

There are variable selection methods closely related to Lasso, for example, fused Lasso [31], adaptive Lasso [40], and Elastic net [41]. Under the assumption in Eq. (1) and assuming the linear model, these methods could be respectively summarized as

\[
\hat{\beta}_{\text{fused}} = \arg\min_{\beta} \|Y - X\beta\|_2^2 + \lambda_1 \sum_{j=1}^p |\beta_j| + \lambda_2 \sum_{j=2} |\beta_j - \beta_{j-1}|, \tag{13}
\]

\[
\hat{\beta}_{\text{aen}} = \arg\min_{\beta} \|Y - X\beta\|_2^2 + \lambda \sum_{j=1}^p w_j |\beta_j| \tag{14}
\]

and

\[
\hat{\beta}_{\text{Enet}} = \arg\min_{\beta} \|Y - X\beta\|_2^2 + \lambda_1 \sum_{j=1}^p |\beta_j| + \lambda_2 \sum_{j=1}^p \beta_j^2. \tag{15}
\]

However, it is discussed in [31] that the computational speed of fused Lasso is a practical difficulty, especially when the parameters in fused Lasso are determined by 5 or 10-fold cross validation. In the adaptive Lasso, the weights \(w_j\) are determined from the ordinary least-square (OLS) regression [40], however, the OLS might be unstable in the case of high dimensional data. By using a data augmentation technique, Elastic net can be solved efficiently in the framework of LARS [41]. A variation of the Elastic net was proposed in [19] for genomic data analysis. As such, we experimentally compare the performance of Elastic net in gene selection.

In the Elastic net formulation Eq. (15), there are two tuning parameters, \(\lambda_1\) and \(\lambda_2\). For each fixed \(\lambda_2\), we solve an augmented \(L_1\) constrained regression problem using LARS. In our work, similar to [19], \(\lambda_2\) is selected from \([0.001, 0.01, 0.1, 1, 10, 100]\) by 10-fold cross validation on the training set, and the chosen \(\lambda_2\) is the one giving the smallest cross validation error. For the chosen \(\lambda_2\), we repeat the experiments in Sections 4.2 and 4.3. For illustration purpose, we performed the experiments on the DLBCL and Prostate datasets.

Fig. 4 shows the average testing error rate curves of the 100 runs for every linear classifier using the genes selected by Lasso, Dantzig selector, and Elastic net. From Fig. 4, we observe that in most cases, Dantzig selector performs worse than Lasso and Elastic net at picking discriminative genes for linear classifiers, and the Elastic net performs very similar to Lasso. For Gaussian kernel SVM, Fig. 5 shows the average testing error curves of the 100 runs with different number of selected genes. For nonlinear SVM, we observe that, same as for linear classifiers, in general, Lasso and Elastic net have similar gene selection ability, and both are more effective than Dantzig selector.

Although Elastic net can be solved by using the time-efficient LARS algorithm, the optimization problem has a large size due to the use of augmented data, thus Elastic net takes more time than Lasso. Furthermore, Elastic net has two parameters, and we employed cross validation to select \(\lambda_2\), which consumes extra time. Overall, compared to Lasso, Elastic net is more time expensive although they have the similar gene selection performance.

![Average testing error curves of the nonlinear SVM with the number of selected genes on the six datasets.](image-url)
5. Conclusion and discussion

In linear regression models, Lasso and Dantzig selector impose upper bound on the $L_1$-norm of the regression coefficients, such that a part of the regression coefficients are shrunk to zeros, therefore Lasso and Dantzig selector implicitly perform variable selection. This paper applies Lasso and Dantzig selector to regress the class label (which is interpreted as the probability of being positive example) on the gene expression data, thus selecting informative genes. The selected genes are further used by different classifiers for the purpose of cancer classification. On six public cancer datasets, we conducted a comparative study on the gene selection ability of Lasso and Dantzig selector. For three linear classifiers and nonlinear SVM, the testing error rate curves demonstrate that, in general, Lasso is more capable at selecting informative genes than Dantzig selector.

The result presented in this paper is consistent with the conclusion in the statistics literature [11,20]. However, the researches in [11,20] study the variable selection performance of Lasso and Dantzig selector for linear regression model, while this paper compares the feature selection ability of Lasso and Dantzig selector for pattern classification problems.

The proposed approach has two distinct stages, i.e., gene selection then classification. There are strategies which combine feature (gene) selection and classification in a unified framework by adding the $L_1$ [22,28,25,33,37] or $L_{\infty}$ constraint [16] directly to the classification models such as logistic regression or support vector machines. By this way, the classification performance might be improved because the selected genes the most suited for the specified classifier (e.g., SVM, logistic regression). This paper chose the two-stage strategy because we intend to compare the gene selection ability of Lasso and Dantzig selector in general, not for a particular classifier. Moreover, the two-stage method is easy to understand and implement. Nevertheless, it is instructive to compare classification performances of the two-stage strategy and the unified strategy, and it is part of our future project.

Conflict of interest statement

None.
Acknowledgement

The authors thank the anonymous reviewers for their comments. S. Zheng was supported by a 2011 Summer Faculty Fellowship awarded by Missouri State University. W. Liu was supported by SZU R&D Fund (201054) and the National Natural Science Foundation of China (60903113).

References