BIOINFORMATICS ORIGINAL PAPER

Vol. 21 no. 19 2005, pages 3741–3747
doi:10.1093/bioinformatics/bti618

Gene expression

Analysis of recursive gene selection approaches from microarray data

Fan Li* and Yiming Yang

Language Technology Institute, 4502 NSH, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

Received on August 13, 2004; revised on August 2, 2005; accepted on August 8, 2005
Advance Access publication August 23, 2005

ABSTRACT

Motivation: Finding a small subset of most predictive genes from microarray for disease prediction is a challenging problem. Support vector machines (SVMs) have been found to be successful with a recursive procedure in selecting important genes for cancer prediction. However, it is not well understood how much of the success depends on the choice of the specific classifier and how much on the recursive procedure. We answer this question by examining multiple classifers [SVM, ridge regression (RR) and Rocchio] with feature selection in recursive and non-recursive settings on three DNA microarray datasets (ALL-AML Leukemia data, Breast Cancer data and GCM data).

Results: We found recursive RR most effective. On the AML-ALL dataset, it achieved zero error rate on the test set using only three genes (selected from over 7000), which is more encouraging than the best published result (zero error rate using 8 genes by recursive SVM). On the Breast Cancer dataset and the two largest categories of the GCM dataset, the results achieved by recursive RR are also very encouraging. A further analysis of the experimental results shows that different classifiers penalize redundant features to different extent and this property plays an important role in the recursive feature selection process. RR classifier tends to penalize redundant features to a much larger extent than the SVM does. This may be the reason why recursive RR has a better performance in selecting genes.

Availability: The datasets are available at http://sdmc.lit.org.sg:8080/GEDatasets/Datasets.html

Contact: hustlf@cs.cmu.edu

INTRODUCTION

Gene microarray data have provided the opportunity to measure the expression level of thousands of genes simultaneously and this kind of high-throughput data has a wide application in bioinformatics research. In DNA microarray data analysis, for example, biologists measure the expression levels of genes (thousands of them) in the tissue samples from patients, and seek explanations about how the genes of patients relate to the types of cancers they had. Many genes could strongly be correlated to a particular type of cancer; however, biologists prefer to focus on a small subset of genes that dominates the outcomes before conducting in-depth analysis and expensive experiments with a larger set of genes. Therefore, automated discovery of this small subset (feature selection) is highly desirable.

Methods for automated feature selection can be roughly divided into two categories: filtering approaches, meaning that feature selection is carried out in a preprocessing step of classification, independent from the choice of the classification method, and wrapper approaches, meaning that a classifier is used to generate scores for features in the selection process and feature selection depends on the choice of the classifier. A recent overall analysis of feature selection approaches can be found in the paper by Guyon *et al*. (2003).

Both types of approaches have been applied to the extraction of gene subsets from DNA microarray data. Filtering methods, such as correlation coefficient ranking (Golub *et al*., 1999), are obviously not the best choices because they score the importance of features independently, ignoring the correlations among them. More complex filtering methods, such as Markov Blanket filtering (Xing *et al*., 2001), have also been tried. However, it has not achieved the level of the best results of wrapper approaches (Guyon *et al*., 2000; Weston *et al*., 2001) (we will compare the detailed results later). As a specific wrapper approach, recursive feature elimination using support vector machine (SVM-RFE) has been found to be successful. On the AML-ALL benchmark collection (introduced in the next section), for example, the best result ever published was by recursive SVM, with an error rate of zero when selecting eight genes from thousands in the original feature space (Guyon *et al*., 2000).

Rakotomamonjy (2003) has investigated the feature selection problem using various SVM-based criteria. His work can be seen as a generalization of the SVM-RFE algorithm. Weston *et al*. (2003) discussed the influence of norm-2, norm-1 and norm-0 regularizers in feature selection. However, none of them explored the influence of the choice of the classifier in the recursive feature selection process.

Although the above research findings provide useful insights, deeper understanding and analysis are needed. We would like to know, for instance, how much does the success of SVM in recursive feature selection on the microarray dataset come from the recursive process, and how much does it depend on the choice of the classifier. And, more generally, what property of a classifier would make it successful in recursive feature selection? Presenting such a study is the main contribution of this paper.

In the System and Methods section, we report our feature selection experiments with SVM, ridge regression (RR) and a Rocchio-style classifier on three microarray datasets (the AML-ALL microarray dataset, the Breast Cancer dataset and the GCM dataset). In the Algorithm Analysis section, we analyze the effect of redundant

*To whom correspondence should be addressed.

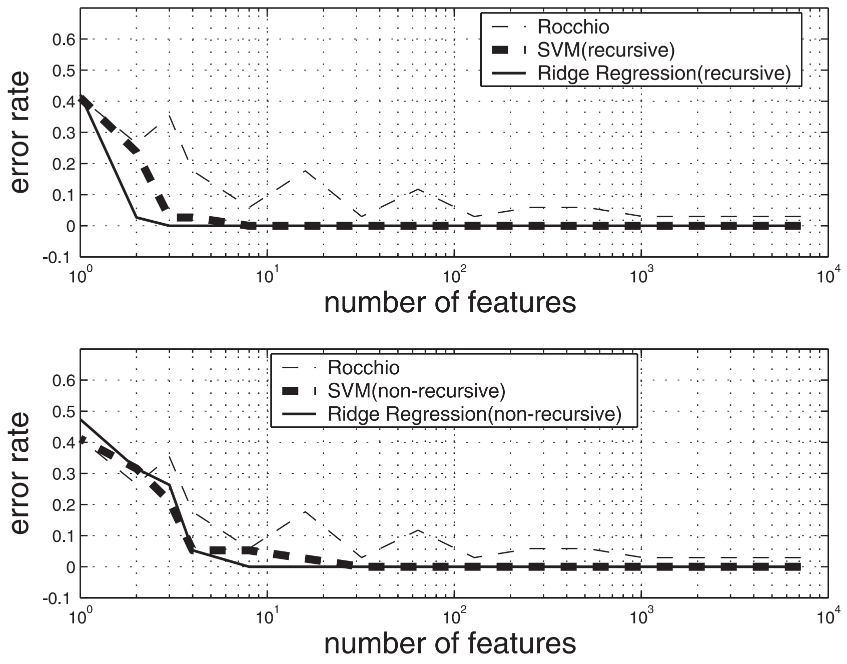

Fig. 1. Performance of three classifiers on AML-ALL dataset: with recursive (upper graph) and non-recursive (lower graph) feature selection.

features in the process of recursive feature selection and compare the differences among the classifiers with respect to their extent to penalize redundant features. We also provide analysis on the classification models with respect to feature selection. In the last section we conclude with the main findings.

SYSTEMS AND METHODS

We conducted a set of experiments for wrapper-style feature selection with three classifiers, in both recursive and non-recursive ways. The three classifiers are Rocchio, SVM and RR. More specifically we choose to examine the linear version of those classification methods, since linear classifiers are relatively simple, easy to interpret and can be enriched through the use of kernel functions for solving non-linear problems. Details about these classifiers can be found in a previous paper (Li and Yang, 2003).

Algorithm 1 Recursive wrapper for feature selection

```
(1) Let $m$ be the initial number of features and $t$ be the number of
    features we want to get.
(2) While ($m \geq t$)
    (a) Train the classifier and get feature weights $w_i$
    (b) Delete the feature with the smallest weight in absolute value
        and set $m \leftarrow m - 1$. (In fact, more than one feature can
        be deleted in each iteration. In this paper, we delete $m/2$
        features when $m \geq 8$ and delete only one feature when $m$
        becomes <8.)
```

The recursive wrapper procedure (the training part) is shown in Algorithm 1. By non-recursive wrapper approach, we mean that we stop the above procedure after the first iteration and select the t top-ranking features based on their weights in absolute value.

Empirical findings on an AML-ALL microarray dataset

The first dataset is named AML-ALL (Fodor, 1997), consisting of a matrix of DNA microarray data. The rows of the matrix represent genes, the columns represent cancerous patients having one of the two different types of leukemia, AML or ALL, and the elements of the matrix represent the gene expression levels in the corresponding patients. There are a total of 7129 genes (features) and 72 patients (examples) split into a training set of 38 examples (27 belong to ALL and 11 belong to AML) and a test set of 34 examples (20 belong to ALL and 14 belong to AML). The classification task is to predict the disease type (ALL or AML) for an arbitrary patient, given the gene expression levels in the tissue sample from that patient. Different feature selection methods have been evaluated on this dataset, including the Markov blanket algorithm (Xing *et al*., 2001) and SVM-based feature selection (Weston *et al*., 2001; Guyon *et al*., 2000); the best result so far (zero error rate) was obtained by recursive SVM, using eight features only (Guyon *et al*., 2000).

We conducted experiments using the three classifiers on the AML-ALL dataset. We use cross-validation on training data to tune regularization parameter λ [for details readers are referred to Li and Yang (2003)] in each classifier. In the feature elimination process, the value of parameter λ will not change. Figure 1 shows the classification results on the test data.

We can see that when we use all the 7129 genes as features, all the three classifiers can predict diseases very well (SVM and RR with zero error rate and Rocchio with very low error rate). However, when we want to find a small subset of genes to predict the disease, the choice of recursive RR is the best in the sense of using the minimum number of features to obtain the lowest error rate in the classification. It is quite impressive that only three genes (selected from over 7000) were needed for this classifier to achieve the error rate of zero, outperforming the best result for recursive SVM ever reported on the same data. Recursive SVM has a worse performance, achieving zero error rate with eight genes. Rocchio[1] has the worst performance when only using a small set of genes. We can also see that recursive process did help RR and

[1]The recursive and non-recursive version for Rocchio classifiers are exactly the same, which would be mentioned in the following sections.

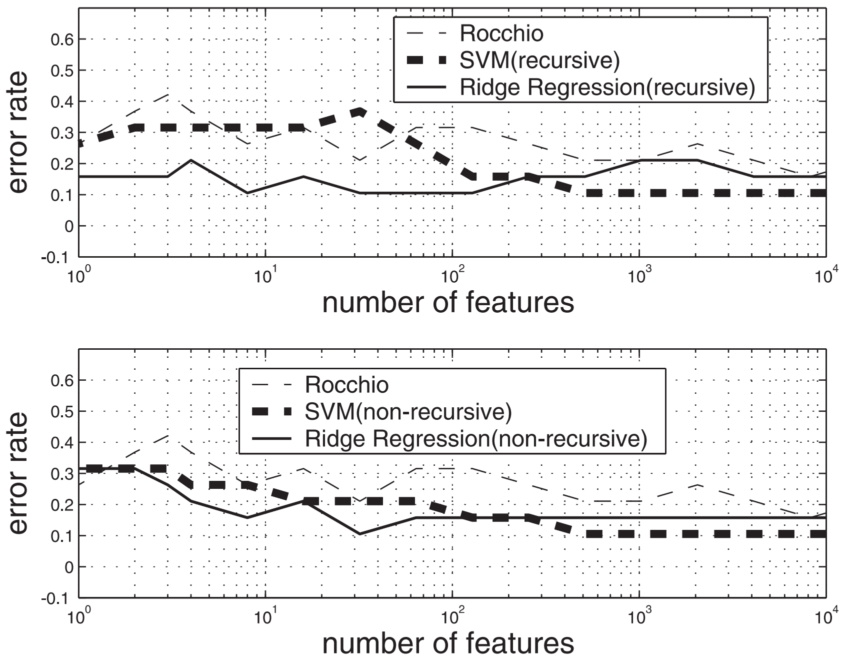

Fig. 2. Performance of three classifiers on Breast Cancer dataset: with recursive (upper graph) and non-recursive (lower graph) feature selection.

SVM to increase the gene selection performance a lot with small number of features.

The three genes found by recursive RR are M27891, X51521 and Y00787. Note that M27891 and Y00787 belong to the 50 genes Golub *et al.* (1999) has identified (which have prediction powers), while X51521 was not reported by Golub *et al.* (1999).

We also merged the training and the test set to get a leave one out cross-validation result using the three classifiers. With three genes, recursive RR gets a leave one out cross-validation error rate of 0.0417, while recursive SVM gets 0.0833 and Rocchio gets 0.2917. This is consistent with the above results on the test set only.

Empirical findings on a breast cancer dataset

The second microarray dataset is a more difficult one, which comes from Van't Veer *et al.* (2002). The training data contains 78 patient samples, 34 of which are from patients who had developed distant metastases within 5 years (labeled as 'relapse'), the rest 44 samples are from patients who remained healthy from the disease after their initial diagnosis, for an interval of at least 5 years (labeled as 'non-relapse'). Correspondingly, there are 12 relapse and 7 non-relapse samples in the testing dataset. The total number of genes is 24 481. Van't Veer *et al.* (2002) have extracted 70 genes that can correctly predict the outcome in 17 of the 19 patients in the test set (thus the error rate is 0.1053).

We conducted experiments using the above three classifiers on this Breast Cancer dataset. The regularization parameter λ is tuned in the same way as described above. Results on the test set are shown in Figure 2. We can see that recursive RR achieved the error rate of 0.1053 with only eight genes, while Van't Veer *et al.* (2002) got the same error rate using 70 genes. However, recursive SVM could not find a small set of genes to achieve this classification accuracy.

Empirical findings on the two largest categories of GCM data

The third microarray dataset [named GCM data by Ramaswamy *et al.* (2001)], spanning 14 different tumor classes, was obtained from NCI and many other institutes. The training data contain 144 patient samples and the test data contain 54 patient samples. The total number of genes is 16 063.

Since this dataset contains 14 categories while the feature selection methods introduced in this paper only focus on binary classification problems, we use one versus all strategy to do feature selection for each category separately. Owing to the space limitation, we only focus on the two largest categories (Lymphoma and Leukemia) in this paper. In the training data, Lymphoma category contains 16 samples and Leukemia category contains 24 samples. In the test data, Lymphoma category contains six samples and Leukemia category contains six samples.

We conducted experiments using the three classifiers for Lymphoma category and Leukemia category, respectively. The regularization parameter λ is tuned in the same way as described above. Results on the test set are shown in Figures 3 and 4. We can see that recursive RR achieved better performance than recursive SVM with small number of features.

ALGORITHM ANALYSIS

Analysis of the recursive feature elimination procedure

Our experiment implies that, given a small subset of genes, most other relevant genes are redundant in the sense of predicting disease. In order to find this small subset, a classifier should be able to select features that are both relevant and not redundant. All the three classifiers we discussed in the paper tend to assign high coefficients (thus high ranks) to most relevant features. However, they have very different strategy to penalize redundant features, which lead to their very different gene selection performance.

To visualize the differences among the three classifiers in selecting features, we show the correlation matrix for the top 50 genes selected by each classifier on the AML-ALL dataset (Fig. 5)[2].

[2]The correlation matrices constructed from the Breast Cancer data and the GCM data also show similar patterns.

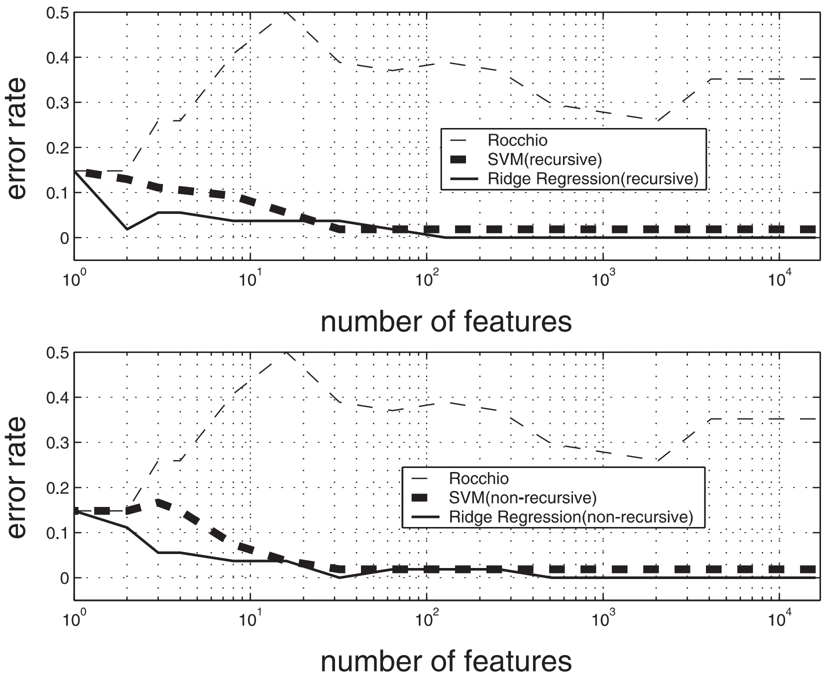

Fig. 3. Performance of three classifiers on Leukemia category of the GCM dataset: with recursive (upper graph) and non-recursive (lower graph) feature selection.

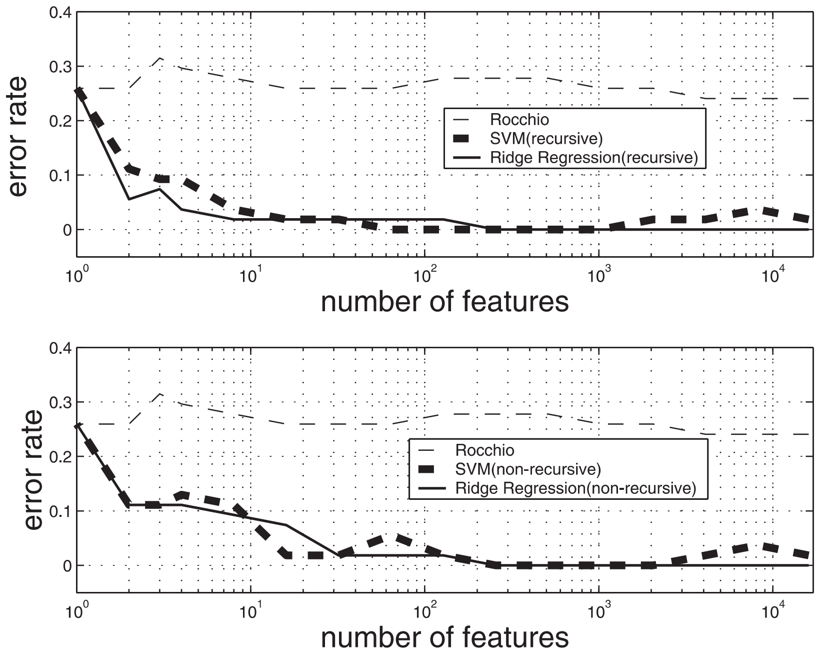

Fig. 4. Performance of three classifiers on Lymphoma category of the GCM dataset: with recursive (upper graph) and non-recursive (lower graph) feature selection.

In each matrix, the features are sorted according to their ranks assigned by the classifiers (feature with rank $= 1$ is the most important feature). The (i, j) element of the matrix is the absolute value of the correlation coefficient between the i-th feature vector and the j-th feature vector in the training data; the color intensity in those graphs reflects the magnitude of gene–gene correlation coefficients: the brighter the color, the stronger the correlation for either positively or negatively correlated genes.

First, we can observe the extent to which different classifiers penalize redundant features intuitively from these correlation matrices. For Rocchio classifier, the bright pixels are clustered in the upper-left

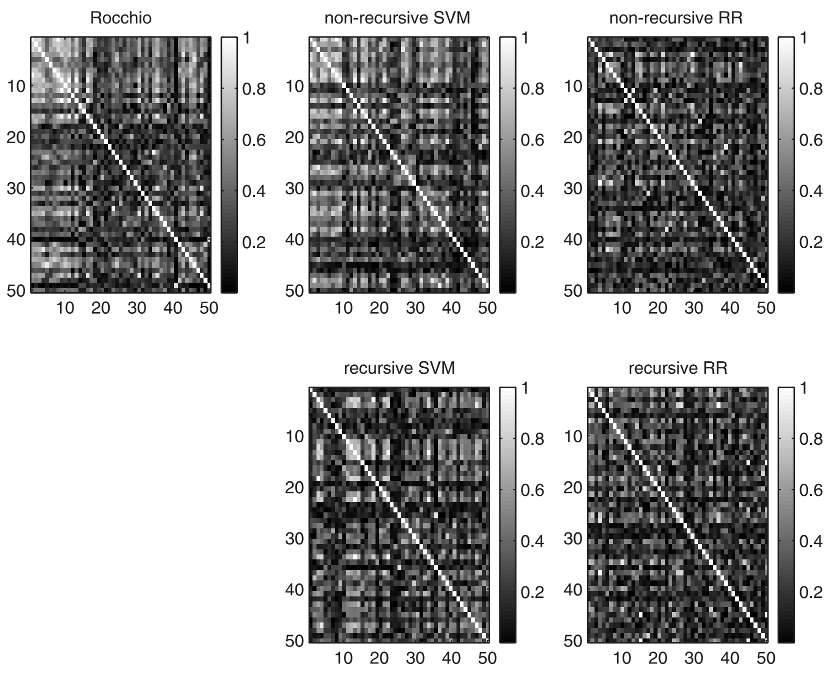

Fig. 5. The correlation matrix of three different classifiers on the ALLAML dataset. The recursive and non-recursive version of Rocchio classifier are exactly the same; thus, we only plot one matrix for Rocchio.

corner of the matrix. This implies that the highest ranked genes for Rocchio classifiers are very similar to each other and contain much redundant information. However, for RR classifier the bright pixels are much more evenly distributed[3], which implies that the highest ranked genes for RR classifier contain much less redundant information. We can also see that the extent to which SVM penalizes redundant features is between Rocchio and RR classifier. Recall that RR classifier has the best gene selection performance, followed by SVM and then by Rocchio. This fact suggests that how classifiers penalize redundant features is closely related to the success of their gene selection strategy.

Second, we can identify the role of recursive process by comparing correlation matrices in recursive and non-recursive versions. For clarity, we first give a simple example here. Let us assume there are three genes (A, B and C) that are relevant to the disease. Among them, genes B and C are similar to each other and contain redundant information. Then, a classifier that will penalize redundant features would assign relatively lower coefficients to genes B and C because of their redundancy. If we use a non-recursive version of the classifier, both genes B and C may be deleted due to their relatively low coefficients (thus low ranks). Then the information they contain may be lost and the classification performance suffers. However, if we use a recursive version of the classifier, the coefficients of the classifier are continuously being re-trained, while the features are deleted one by one. Let us suppose that gene C is deleted before gene B. Then without gene C, gene B is no longer redundant and its rank will be improved. Ideally, the recursive classifier would finally keep genes A and B as its top-ranked features. This is the principle on which the recursive procedure works. From the graphs, we can see that in the recursive version the bright pixels tend to evenly distribute. The reason is that the classifiers have deleted part of the genes with redundant information and kept the left ones (which are no longer redundant). Our experiments show that when the number of features is large, the performance of recursive classifiers and non-recursive classifiers is similar. When the number of features becomes very small (<10), recursive classifiers (especially RR classifier) often achieve better performance than non-recursive classifiers.

[3]In fact we can see less bright pixels because we only show the top 50 features. If we show all the 7129 features, then the number of bright pixels would be the same for all the classifiers.

Generally speaking, for the recursive feature selection to succeed eliminated features at certain point in the process must have some influence on the re-adjusted weights of the remaining features, and the influence, desirably, should penalize redundant features and promote non-redundant ones. The influence depends on the choice of the classifier since different classifiers have different penalization strategies to redundant features: some are better than others in this aspect. In the next subsection, we analyze the penalization property of Rocchio, RR and SVM in detail.

Analysis of Rocchio

The following notations will be used in the rest of this paper: The training data consist of n pairs of $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n)$, where $\vec{x}_i = (x_{i1}, \ldots, x_{im})$ represents the values of the m input features in the i-th training example, and $y_i \in \{-1, 1\}$ is the class label. We use vector $\vec{\beta} = (\beta_1, \ldots, \beta_m)$ to represent the parameters of the linear classifier. n_+ and n_- are used to represent the number of positive and negative training examples, respectively.

Rocchio-style classifiers are commonly used for their simplicity, efficiency and reasonable performance (Li and Yang, 2003). A prototype vector is constructed for each class in the form $\vec{\beta} = \vec{u} - b\vec{v}$, where $\vec{u}$ and $\vec{v}$ are the centroids of positive and negative training examples, respectively, and b is the weight of the negative centroid relative to the positive centroid. By centroid we mean the vector average of training examples. The weight of the p-th feature can be computed as follows:

$$\beta_p = \frac{1}{n_+}\sum_{y_i=1} y_i x_{ip} + \frac{b}{n_-}\sum_{y_i=-1} y_i x_{ip}.$$

Obviously, the weight of each feature is independent from others and remains constant during the recursive process. In other words, Rocchio does not give any penalization to redundant features; thus, the recursive feature selection procedure cannot work at all (its recursive and non-recursive procedure are exactly the same). This may explain the bad performance of Rocchio in gene selection—genes with redundant information dominate its highest ranked features.

Analysis of ridge regression

Term weights in RR are determined by the minimization of its loss function, defined as follows:

$$L_{\mathrm{RR}} = \sum_{i=1}^{n}\left(1 - y_i\vec{\beta}\vec{x}_i\right)^2 + \lambda\|\vec{\beta}\|^2$$
$$= \sum_{i=1}^{n}\left(1 - y_i\sum_{p=1}^{m} x_{ip}\beta_p\right)^2 + \lambda\sum_{p=1}^{m}\beta_p^2.$$

To minimize L_{RR} we need to set its partial derivative with respect to each term weight (β_q) to zero, which yields

$$\sum_{i=1}^{n} x_{iq}y_i\left(1 - y_i\sum_{p=1}^{m} x_{ip}\beta_p\right) = \lambda\beta_q. \quad (1)$$

Thus

$$\beta_q = \frac{\sum_{i=1}^{n} x_{iq}y_i - \sum_{p=1}^{m}\sum_{i=1}^{n} x_{iq}x_{ip}\beta_p}{\lambda}. \quad (2)$$

Now we focus on the second term in the numerator of the above formula. Note that $\sum_{i=1}^{n} x_{iq}x_{ip}$ is the dot-product of two 'feature vectors', reflecting the similarity between features p and q in the n training examples, which can be replaced by token $\mathrm{sim}(p,q)$, and that $\sum_{i=1}^{n} x_{iq}x_{ip}\beta_p = \mathrm{sim}(p,q)\beta_p$ reflects how much the correlation and the weight of feature p jointly deduct the weight of feature q. To be clearer, we can write the above formula as follows:

$$\beta_q = \frac{\sum_{i=1}^{n} x_{iq}y_i - \sum_{p=1}^{m}\mathrm{sim}(p,q)\beta_p}{\lambda}.$$

Clearly, without the second term in the numerator of the formula for β_q, RR is very similar to Rocchio; with the second term, however, the redundant features would be penalized, and the elimination of features during the recursive process has the effect of boosting the remaining features that are correlated to the eliminated ones. In other words, the iterative process has the effect of boosting the weights for relatively non-redundant features in the remaining set.

Analysis of SVM

Although SVM has been widely used, not much work has been reported for an explicit analysis of how SVM penalizes redundant features. The loss function of SVM (Vapnik *et al.*, 1995) has the form of

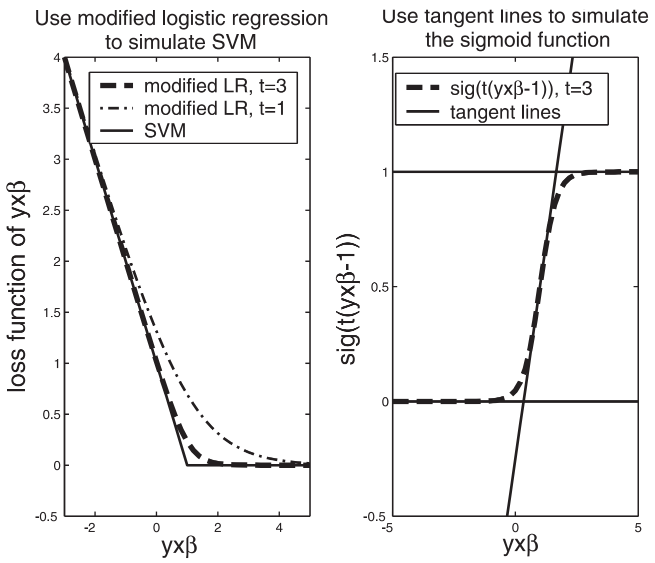

Fig. 6. The left graph shows the loss function of modified logistic regression (with different t-values) and SVM, without the regularization part. We can see that when t gets larger, the loss function of modified LR would converge to SVM. The right graph shows the transformed sigmoid function $\mathrm{sig}(t(y_i\vec{\beta}\vec{x}_i - 1))$ for $t = 3$ and three tangent lines. The $a_i t$ values are about 0, 0.75 and 0, and b_i values are about 0, 0.5 and 1 for these three lines.

$$L_{\mathrm{svm}} = \sum_{i=1}^{n}(1 - y_i\vec{\beta}\vec{x}_i)_+ + \lambda\|\vec{\beta}\|^2.$$

This formula appears to be similar to the loss function of RR except that the first term on the right-hand side is not differentiable. This makes it hard to directly apply the same kind of analysis to SVM as we did for RR. Here, we used a modified logistic regression to simulate SVM. The loss function of the modified logistic regression (LR) is

$$L_{\log} = \sum_{i=1}^{n}\frac{1}{t}\ln(1 + \exp(t(1 - y_i\vec{\beta}\vec{x}_i))) + \lambda\|\vec{\beta}\|^2.$$

Zhang *et al.* (2003) have shown that when t gets large enough the solution of modified LR will converge to the solution of SVM. In other words, when t is large enough, the modified LR ranks features in the same way as SVM does. The left graph of Figure 6 intuitively shows that the loss function of modified LR will converge to the loss function of SVM when t gets large.

To minimize $L_{\log}$ we need to set its partial derivative with respect to each term weight (β_q) to zero, which yields

$$\sum_{i=1}^{n} x_{iq}y_i\left(1 - \frac{1}{1 + \exp(t(1 - y_i\vec{\beta}\vec{x}_i))}\right) = 2\lambda\beta_q.$$

We use the sigmoid function $\mathrm{sig}(z) = 1/[1 + \exp(-z)]$ to simplify the above formula and get

$$\sum_{i=1}^{n} x_{iq}y_i\left(1 - \mathrm{sig}\left(t\left(y_i\sum_{p=1}^{m} x_{ip}\beta_p - 1\right)\right)\right) = 2\lambda\beta_q. \quad (3)$$

The sigmoid function is non-linear and this makes it difficult for us to analyze Formula 3 as we analyze Formula 1. Thus we use multiple tangent lines to simulate the sigmoid function. In particular, we use $\mathrm{sig}(z) = a_z z + b_z$ to replace $\mathrm{sig}(z) = 1/1 + [\exp(-z)]$, where $a_z = [\mathrm{dsig}(z)]/\mathrm{d}z$ and $b_z = \mathrm{sig}(z) - a_z z$. Thus $\mathrm{sig}(t(y_i\vec{\beta}\vec{x}_i - 1))$ can be re-written as $a_i t y_i \vec{\beta}\vec{x}_i - a_i t + b_i$. Here a_i and b_i are functions of $t(y_i\vec{\beta}\vec{x}_i - 1)$. It is easy to see that when the value of $y_i\vec{\beta}\vec{x}_i$ (this term is often called the margin value of $\vec{x}_i$ and it reflects the goodness with which $\vec{x}_i$ is classified) varies from $-\infty$ to ∞, a_i would vary from 0 to 0.25 and then back to 0, while b_i would vary from 0 to 1. The right graph of Figure 6 shows the sigmoid function $\mathrm{sig}(t(y_i\vec{\beta}\vec{x}_i - 1))$ and three tangent lines.

Now we re-write Formula 3 as

$$\sum_{i=1}^{n} x_{iq} y_i \left(1 - a_i t y_i \sum_{p=1}^{m} x_{ip}\beta_p + a_i t - b_I\right) = 2\lambda\beta_q.$$

Finally, we get

$$\beta_q = \frac{\sum_{i=1}^{n}(a_i t + 1 - b_i)x_{iq}y_i - \sum_{p=1}^{m}\sum_{i=1}^{n}(a_i t)x_{iq}x_{ip}\beta_p}{2\lambda}. \quad (4)$$

We can see that Formula 4 has the same form as Formula 2. The first term (relevancy term) in the numerator measures the extent that feature q is relevant and the second term (redundancy penalization term) measures the extent that feature q is redundant. However, unlike the strategy used in Formula 2, Formula 4 assigns weight $a_i t + 1 - b_i$ to the relevancy term and weight $a_i t$ to the redundancy penalization term for each sample i. Note that $1 - b_i > 0$ always holds, which means that the weight for the relevancy term is always larger than the weight for the redundancy penalization term. More specially, suppose t is large enough, then

(1) For samples whose margin values $y_i\vec{x}_i\beta$ are obviously less than one, $a_i t$ and b_i will be close to zero[4], which means that the redundancy penalization term almost vanishes while the relevancy term still holds (its weight will be close to one). Thus, modified LR mainly models the relevancy of features and almost ignores the redundancy among features in these samples. This strategy is similar to Rocchio.

(2) For samples whose margin values $y_i\vec{x}_i\beta$ become close to one, $a_i t$ will become larger; thus, the redundancy penalization term will play a role. In fact, when $y_i\vec{x}_i\beta$ is equal to one, $a_i t$ will take its largest value $0.25t$, while b_i will be equal to 0.5. The redundancy penalization term has the largest influence at this point. For these samples, both the relevancy and the redundancy of features are modeled, and modified LR will penalize redundant features in a way similar to RR.

(3) For samples whose margin values $y_i\vec{x}_i\beta$ become obviously larger than one, $a_i t$ will again become close to zero while b_i will become close to one. It is easy to see that the weights for the relevancy term and the redundancy penalization term will both be close to zero. In other words, modified LR tends to ignore these samples.

[4]This is easy to see from the fact that $a_i t = t[\exp(t(1 - y_i\vec{x}_i\vec{\beta}))]/[(1 + \exp(t(1 - y_i\vec{x}_i\vec{\beta}))^2]$.

From the above analysis, we can see that modified LR penalizes redundant features in a way between Rocchio and RR. Note when t is large enough, modified LR will rank features in the same way as SVM will do. Thus the above analysis is also suitable for SVM. In fact, SVM only models the redundancy among features for the samples exactly on the margin ($y_i\vec{x}_i\vec{\beta} = 1$). The feature redundancy information in other samples (with larger or smaller $y_i\vec{x}_i\vec{\beta}$ values), which may be very valuable, is totally ignored. This may lead to unsatisfied performance when SVM is used for feature selection.

DISCUSSION

In this paper, we addressed a key question for wrapper-style feature selection: what property of a classifier would lead to the success of recursive feature elimination? By analyzing three different classifiers, we find that the ability of a classifier for penalizing redundant features in the recursive process has a strong influence on its success. RR, having an explicit penalty of correlated features in its loss function minimization, is a good choice for recursive feature selection, and shows best performance in our experiments. SVM is not as effective as RR in terms of finding non-redundant features in the recursive gene selection process. Part of the reason may be that it only penalizes redundant features for samples exactly on the margin. Rocchio will not penalize redundant features at all. This property makes its recursive process not effective, i.e. not different from using a non-recursive approach.

Conflict of Interest: none declared.

REFERENCES

Fodor,S. (1997) Massively parallel genomics. *Science*, **277**, 393–395.

Golub,T.R. *et al*. (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531–537.

Guyon,I. *et al*. (2000) Gene selection for cancer classification using support vector machines. *Machine Learning*, **46**, 389–422.

Guyon,I., Gunn,S., Ben-Hur,A. and Dror,G. (2004) Result analysis of the NIPS 2004 feature selection challenge. *NIPS*, MIT Press, vol. 17, pp. 545–552.

Li,F. and Yang,Y. (2003) A loss function analysis for classification methods in text categorization. In *Twentieth International Conference on Machine Learning*, Washington, DC, pp. 472–479.

Rakotomamonjy,A. (2003) Variable selection using SVM-based criteria. *J. Mach. Learn. Res.*, 1357–1370.

Ramaswamy,S. *et al*. (2001) Multiclass cancer diagnosis using tumor gene expression signatures. *Proc. Natl Acad. Sci. USA*, **98**, 15149–15154.

Van't Veer,L.J. *et al*. (2002) Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, **415**, 530–536.

Vapnik,V. *et al*. (1995) *The Nature of Statistical Learning Theory*. Springer, NY.

Weston,J., Mukherjee,S., Chapelle,O., Pontil,M., Poggio,T. and Vapnik,V. (2000) Feature selection for SVMs. In *NIPS*, MIT Press, vol 13, pp. 668–674.

Weston,J. *et al*. (2003) Use of the zero-norm with linear models and kernel methods. *J. Mach. Learn. Res.*, 1439–1461.

Xing,E., Jordan,M. and Karp,R. (2001) Feature selection for high-dimensional genomic microarray data. In *Eighteenth International Conference on Machine Learning*, MA, pp. 53–64.

Zhang,J., Jin,R. and Yang,Y. (2003) Modified logistic regression: an approximation to SVM and its applications in large-scale text categorization. In *Twentieth International Conference on Machine Learning*, Washington, DC, pp. 888–897.