Using Recursive Classification to Discover Predictive Features

Fan Li
Carnegie Mellon Univ
Pittsburgh, PA, 15213
hustlf@cs.cmu.edu

Yiming Yang
Carnegie Mellon Univ
Pittsburgh, PA, 15213
yiming@cs.cmu.edu

ABSTRACT

Finding most predictive features for statistical classification is a challenging problem and has important applications. Support Vector Machines (SVMs), for example, have been found successful with a recursive procedure in selecting most important genes for cancer prediction. It is not well understood, however, how much the success depends on the choice of the classifier, and how much on the recursive procedure. We answer this question by examining multiple classifiers (SVM, ridge regression and Rocchio) with feature selection in recursive and non-recursive settings, on a DNA microarray dataset (AMLALL) and a text categorization benchmark (Reuters-21578). We found recursive ridge regression most effective: its best classification performance (zero error) on the AMLALL dataset was obtained when using only 3 genes (selected from over 7000), which is more impressive than the best published result on the same benchmark – zero error of recursive SVM using 8 genes. On Reuters-21578, recursive ridge regression also achieves the best result ever published (the improvement was verified in a significance test). An in-depth analysis of the experimental results shows that the choice of classifier heavily influences the recursive feature selection process: the ridge regression classifier tends to penalize redundant features to a much larger extent than the SVM does.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; I.5.1 [**Pattern Recognition**]: Models-statistical; I.5.4 [**Pattern Recognition**]: Application-Text processing

General Terms

Algorithms Performance

Keywords

text categorization, feature selection, machine learning

1. INTRODUCTION

Finding a small subset of most predictive features in a high dimensional feature space is an interesting problem in statistical classification, motivated for more accurate modeling, less costly computation, and/or better explanations about the relations among input and output variables. Text categorization, for example, is an area where automated feature selection is often useful. A large document collection typically contains hundred thousands of unique words in its vocabulary, i.e., the feature space. Training a classifier with such a high-dimensional space tends to have the problems of over-fitting and costly computing. Feature selection, therefore, has been commonly used for dimensionality reduction. In the recent NIPS feature selection workshop [3], feature selection improved the classification performance on three of the five datasets(including a text dataset). Note that improving the classification performance is not the only purpose of feature selection. 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 be strongly correlated to a particular type of cancer; however, biologists prefer to focus on a small subset of genes which dominates the outcomes before conducting in-depth analysis and expensive experiments with a larger set of genes. Automated discovery of this small subset, therefore, is highly desirable.

Methods for automated feature selection can be roughly divided into two categories: *filtering* approaches, meaning that feature selection is done in a prepossessing 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.

Both types of approaches have been applied to the extraction of gene subsets from DNA microarray data. Filtering methods like “correlation coefficient ranking” [2] are obviously not the best choices because they score the importance of features independently, ignoring the correlations among them. More complex filtering methods like Markov Blanket filtering [14] has also been tried. However it has not achieved the level of the best results of wrapper approaches [4, 13] (we will compare the detailed results in section 2). As a specific wrapper approach, recursive feature elimination using SVM (SVM-REF) has been found very successful. On the AMLALL benchmark collection (Section 2), for example, the best result ever published was by recursive SVM, with an error rate of zero when selecting 8 genes from thousands in the original feature space [4].

Automated feature selection has also been studied intensively in text categorization. Because of the scaling problem in text data, most researchers have focused on filtering-style feature selection methods [9, **?**, 11]. In a recent evaluation of those methods with KNN, Naive Bayes, Rocchio and SVM classifiers on a text evaluation benchmark (Reuters-21578) [11], it was observed that although feature selection did significantly improve the performance of some classifiers such as Naive Bayes, it did not improve the results of the best performing classifiers such as SVM. This observation is consistent to an early report by [5] who found that even the features ranked the lowest still contain considerable information for SVM, and that removing those features tends to hurt the performance of SVM. Both experiments, however, were conducted only with filtering-style feature selection, leaving the question open for the performance of SVM with recursive feature selection in text categorization. A more complicated feature selection method (filtering-style), using Markov blankets, was also examined by [6] on the Reuters dataset with a Naive Bayes classifier. They observed a smaller performance improvement, compared to the improvement reported for Naive Bayes classifier in [11]

[10] has investigated the feature selection problem using various SVM-based criteria. His work can be seen as a generalization of the SVM-REF algorithm. [12] discussed the influence of norm-2, norm-1 and norm zero regularizers in feature selection. However, neither of them explored the influence of the choice of the classifier in the recursive feature selection process.

While 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 or the dataset? 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 section 2, we report our feature selection experiments with SVM, ridge regression and a Rocchio-style classifier on the AMLALL microarray dataset and the Reuters-21578 text categorization corpus. In section 3 we analyze the effect of correlated features in the process of recursive feature elimination given a classifier, and compare the differences among the classifiers with respect to their preferences for independent features against redundant features. Section 4 provides analysis on the classification models with respect to feature selection. Section 5 concludes the main findings.

2. OBSERVATIONS ON THREE CLASSIFIERS WITH FEATURE SELECTION

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 Ridge Regression (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[8].

The *recursive wrapper* procedure (the training part) is defined below[1].

```
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 $w_i$ for
           $i = 1, 2, \ldots, m$ (i.e., the regression coefficients in
           the linear model)
       (b) Delete the feature with the smallest weight in ab-
           solute value and set $m \leftarrow m - 1$
```

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.

2.1 Empirical Findings on a Microarray Dataset

The first data set is named AMLALL [1], consisting of a matrix of DNA microarray data. The rows of the matrix are genes, the columns are cancerous patients having one of the two different types of leukemia ,AML or ALL, and the elements of the matrix are 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 ([14]) and SVM based feature selection ([13, 4]); the best result so far (zero error rate) was obtained by recursive SVM, using 8 features only[4].

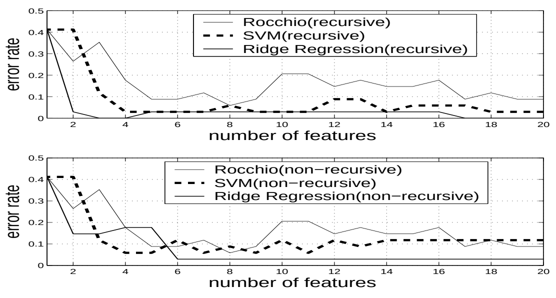

Figure 1: Performance of three classifiers on AMLALL: with recursive(upper graph) and non-recursive(lower graph) feature selection

We conducted experiments using the three classifiers on the AMLALL dataset. Figure 1 shows the classification results on the test data. We use cross-validation on training data to tune the value of a parameter, λ, the coefficient of

[1]In the real Microarray data and text data, the number of features is too large. Thus we delete $\frac{m}{2}$ features, instead of only deleting one feature in each iteration.

the regularization component in each classifier [8]. As a result, the λ value was set to 0.00001 for all the classifiers. When we use all the 7129 genes as features, SVM made 3 errors and Rocchio made 1 error on the test set. The other three classifiers classified all the test examples correctly. The results with these classifiers are shown in figure 1. These results show that the choice of classifier matters for the effectiveness, and that recursive ridge regression 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 7,000) 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[2]. Detailed information about those three genes (M27891, X51521 and Y00787) can be found from the Gene Bank database, for those who are interested (these three genes is not a subset of the 8 genes [4] reported).

2.2 Empirical Findings on Reuters 21578 Benchmark corpus

The second data set we tried is the Reuters-21578 corpus ApteMod version, which has been a benchmark in text classification evaluations. It contains 7769 training documents and 3019 testing documents. Many classifiers and feature selection methods have been evaluated on this dataset. Among those, Ridge Regression without feature selection had the best performance in macro-averaged F1 (a standard measure for text categorization performance) ever published on this corpus [8]. Support Vector Machines (SVM) also had competitive performance when using all the features.

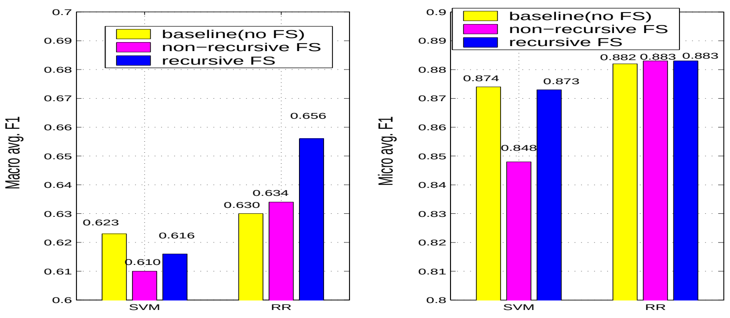

Figure 2: Macro- and micro- avg. F1 scores of RR and SVM on Reuters21578. Each classifier has three versions: the baseline version (without any feature selection), the non-recursive feature selection version and recursive feature selection version

Figure 2 shows the results of these two classifiers with feature selection in both the recursive and non-recursive settings, respectively; the results of the two classifiers without any feature selection (i.e., using all the features) are also provided as the baselines. We tuned the number of features and the classification thresholds (for converting system-generated confidence scores to binary decisions) on a per-category basis using two fold cross validation on the training data. We also tuned the regularization parameter (shared by all the categories) in the same manner. We did a t-test to compare the macro-averaged F1 scores of the baseline RR and the recursive RR, and found that the latter was significantly better than the former with a p-value of 0.0372. On the other hand , we did not find the recursion with SVM improving the average performance over the case of using the baseline SVM on this dataset.

One thing to be noticed is that the F1 scores of the baseline RR and SVM in those figures are slightly different from those reported before on the same benchmark. The reason is that when we use k-fold cross-validation to tune the SCUT thresholds, we split the training documents into k folders with equal size for each category, while [8] split training documents into k folders randomly(not necessarily with equal size). Nevertheless, the scores represented here are very close to the best results published for RR and SVM on the Reuters-21578 corpus [8], making them valid as the baselines for comparison.

Figure 3 compare the macro- and micro-averaged performance curves of RR and SVM in both the recursive and non-recursive settings, with respect to a varying number of features. Notice that when we produce results in figure 3, we force all the categories to have the same number of features in each point so that we can plot a single feature selection curve for all 90 categories. This is why the results in figure 3 are not as good as the results in figure 2, where we tune the number of features per category.

From the two figures, we can see that when using all the features, RR and SVM had a performance very close to each other. However, when using less features, recursive RR outperformed recursive SVM constantly, for the entire range of the x-axis, and in both macro- and micro-averaged F1 scores. This is a surprising observation that was not reported before on this benchmark dataset; we will discuss the implications further in Sections 3 and 4.

In the non-recursive settings, RR is better than SVM in macro-averaged F1 (which is dominated by the results on rare categories). When comparing their performance in micro-averaged F1 (which is dominated by the results on

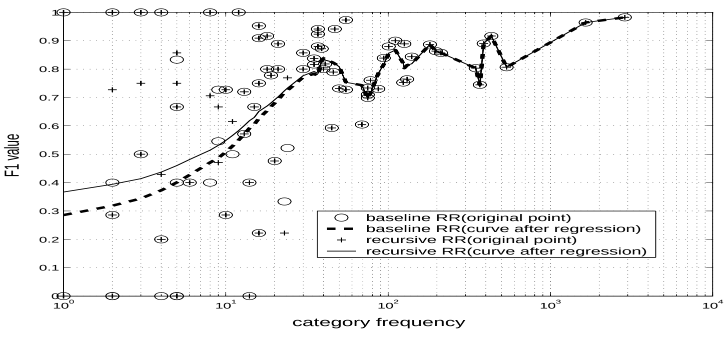

Figure 3: F1 scores of baseline RR and recursive RR on 90 categories in Reuters21578. X axis is category frequency. Two local regression curves (window width=30) are also plotted, showing the smoothed F1 values.

[2]Note that the results we had for recursive SVM are not exactly the same as those reported by [4], possibly because they used a different implementation for the SVM classifier. The SVM classifier we used is SVM-light, downloaded from http://svmlight.joachims.org.

common categories), which one is better is not so clear, depending on the ranges in the x-axis.

Figure 3 compares the performance of recursive RR with the baseline RR on individual categories. The x-axis corresponds to categories (90) sorted by the training-set frequency; the y-axis corresponds to the magnitude of the F1 measure. We plot both the F1 values on individual categories, and the interpolated and smoothed curves for the average performance. An interesting observation is that recursive RR mainly improves the performance on rare categories. In fact, the average of the F1 values was improved from 0.379 to 0.447 on the 30 most rare categories (whose training-set frequencies are less than ten) when using recursive RR instead of the baseline RR on this corpus.

3. CORRELATION AMONG SELECTED FEATURES

To visualize the differences among the three classifiers in selecting features, we show the correlation matrix for the top 32 features selected by each classifier on the AMLALL dataset (in figure 4). The correlation matrices constructed from Reuster21578 also show similar patterns.

In each matrix, the features are sorted according to the order (dependent on the classifier) of features being eliminated in the recursive process. 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; those feature vectors come from the training data, i.e., the row vectors in the DNA microarray for the training examples define the gene vectors. 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.

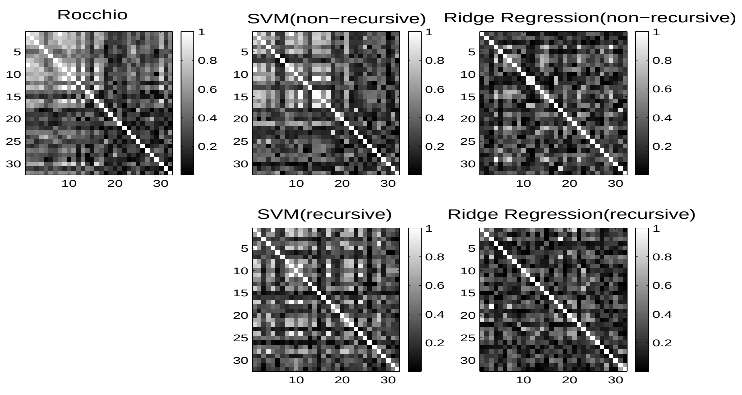

Figure 4: The correlation matrix of three different classifiers (non-recursive) on ALLAML dataset

There are two things from these graphs which we should notice. First, we can easily see a major difference between recursive Rocchio and recursive ridge regression (RR): the former has more bright pixels in the upper-left corner of its matrix, while the latter has more evenly distributed bright pixels. This means that recursive Rocchio tends to reserve correlated features among its choices during the feature elimination process, and that RR tends to reserve non-correlated features during the process. This observation gives an intuitive explanation for the superior performance of RR in figure 1. That is, the fewer redundant features a small subset includes, the more information it would offer for accurate prediction. From these graphs we can also see that SVM is worse than RR but better than Rocchio, with respect to their preference in choosing non-redundant features over redundant ones.

Second, if we compare the recursive version and non-recursive version of these three classifiers, we can see from figure 4 that in recursive version, the bright pixels tend to evenly distribute , which means that the recursive process will increase the tendency of the classifiers to choose non-redundant features over redundant ones. In fact, this is the reason why recursive feature selection is often preferred than the non-recursive version. Figure 1 compares the performance curves of RR and SVM in the recursive setting and the non-recursive setting on AMLALL dataset. Clearly, the recursive process helped to improve the performance for both RR and SVM, particularly when the size of the feature subset is small. Rocchio is the only exception, whose recursive version and non-recursive version are exactly the same(which we will explain later).

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 that the influence, desirably, should penalize redundant features and promote non-redundant ones. The influence depends on the choice of the classifier: some are better than others in this aspect. In the next section, we analyze Rocchio, ridge regression and SVM in detail, exploring why they have different influences in the recursive feature selection process.

4. LOSS FUNCTION ANALYSIS

Rocchio-style classifiers are commonly used for their simplicity, efficiency and reasonable performance [7]. 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

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

Obviously, the weight of each feature only depends on the training examples, and remains constant during the recursive process. In other words, feature weights and their ranks relative to each other are not influenced by the iterative feature elimination process, thus the recursive and non-recursive settings do not make any difference for Rocchio, as long as the size of the feature subset is fixed.

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

$$L_{RR} = \sum_{i=1}^{n} (1 - y_i \langle \vec{\beta}, \vec{x}_i \rangle)^2 + \lambda \|\vec{\beta}\|^2$$

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

$$\beta_q = \frac{\sum_{i=1}^{n} x_{iq} y_i - \sum_{p \neq q} \sum_{i=1}^{n} x_{iq} x_{ip} \beta_p}{\sum_{i=1}^{n} x_{iq}^2 - \lambda}$$

Now focus on the second term in the numerator of the above formula. Notice 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 $sim(p,q)$, and that $\sum_{i=1}^{n} x_{iq}x_{ip}\beta_p = sim(p,q)\beta_p$ 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

$$\beta_q = \frac{\sum_{i=1}^{n} x_{iq}y_i - \sum_{p \neq q} sim(p,q)\beta_p}{\sum_{i=1}^{n} x_{iq}^2 - \lambda}$$

Clearly, without the second term in the numerator of the formula for β_q, ridge regression is very similar to Rocchio; with the second term, however, 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.

Although SVM has been widely used, not much work has been reported for explicit analysis of how SVM penalizes redundant features. In this section, we will show that SVM may or may not penalize redundant features. The extent of penalization depends on the specific distribution of support vectors.

The loss function of SVM has the form:

$$L_c = \sum_{i=1}^{n} (1 - y_i \langle \vec{\beta}, \vec{x}_i \rangle)_+ + \lambda \|\vec{\beta}\|^2$$

It appears to be similar to the loss function of ridge regression except that the first term on the right hand side. This non-differentiable term gives SVM a special property: only a small portion of training examples (which are called support vectors) are really used to train the classification boundary and calculate coefficients for features. Most other training examples which are non-support vectors would not be used at all. Although this strategy leads to a sparse solution in the example space and has several advantages in classification tasks, it is not always a good choice in a feature selection task. We know support vectors is often a small portion of training examples and contain all the information about the classification boundary in SVM framework. However, this does not mean they also contain all the information about the redundancy relationship among features. The non-support vectors may also contain useful information reflecting the redundancy relationship among features and this part of information would be lost since non-support vectors are not used in SVM.

5. CONCLUSIONS

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 reached the following conclusions. The ability of a classifier for penalizing correlated features and promoting independent features in the recursive process has a strong influence on its success. Ridge regression, having an explicit penalty of correlated features in its loss function minimization, is a good choice for recursive feature selection. Our experimental results strongly support this point. SVM is not as effective as ridge regression in terms of finding non-redundant features because it ignores non-support vectors that often contain important information about correlated features. Rocchio, with a constant feature weighting (or ranking) scheme, makes a recursive process not effective, i.e., not different from using a non-recursive approach. Correlation matrices with features sorted in the order of elimination during the recursive process are useful for visualizing the strengths/weaknesses of classifiers in finding non-redundant features.

6. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0225656.

7. REFERENCES

[1] S. Fodor. Massively parallel genomics. In *Science.277.*, pages 393–395, 1997.

[2] T. Golub, D. Slonim, P. Tamayo, and C. Huard. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. In *Science, 286(15)*, pages 531–537, 1999.

[3] I. Guyon, S. Gunn, and A. B. Hur. Benchmark datasets and challenge result summary. In *NIPS 2003 feature selection workshop*, 2003.

[4] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. In *Machine Learning 2000*, 2000.

[5] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *LS8-Report 23, Universit?t Dortmund, LS VIII-Report*, 1997.

[6] D. Koller and M. Sahami. Toward optimal feature selection. In *International Conference on Machine Learning 1996*, 1996.

[7] D. Lewis, Y. Yang, T. Rose, and F. Li. Rcv1: A new text categorization test collection. In *Journal of Machine Learning Research 2003 (accepted)*, 2002.

[8] F. Li and Y. Yang. A loss function analysis for classification methods in text categorization. In *The Twentieth International Conference on Machine Learning (ICML-2003)*, 2003.

[9] D. Mladenic. Feature subset selection in text learning. In *ECML98*, 1998.

[10] A. Rakotomamonly. Variable selection using svm-based criteria. In *Journal of Machine Learning Research 3(2003)*, pages 1357–1370, 2003.

[11] M. Rogati and Y. Yang. High-performing feature selection for text classification. In *ACM CIKM 2002*, 2002.

[12] J. Weston and B. S. Andre Elisseeff. Use of the zero-norm with linear models and kernel methods. In *Journal of Machine Learning Research 3(2003)*, pages 1439–1461, 2003.

[13] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, and V. V. T. Poggio. Feature selection for svms. In *NIPS 2001*, 2001.

[14] E. Xing, M. Jordan, and R. Karp. Feature selection for high-dimensional genomic microarray data. In *In the Eighteenth International Conference on Machine Learning, in press.*, 2001.