Analysis of SAGE Results with Combined Learning Techniques

Hsuan-Tien Lin and Ling Li

Learning Systems Group, California Institute of Technology, USA
htlin@caltech.edu, ling@caltech.edu

Abstract. Serial analysis of gene expression (SAGE) experiments could provide us the expression level of thousands of genes in a biological sample. However, the number of available samples is relatively small. Such undersampled problem needs to be carefully analyzed from a machine learning perspective. In this paper, we combine several state-of-the-art techniques for classification, feature selection, and error estimation, and evaluate the performance of the combined techniques on the SAGE dataset. Our results show that a novel algorithm, support vector machine with the stump kernel, performs well on the SAGE dataset both for building an accurate classifier, and for selecting relevant features.

1 Introduction

Serial analysis of gene expression (SAGE) experiments could provide us an enormous amount of information on the expression level of different genes in some cell populations. The expression level is evaluated by counting the occurrences of the SAGE tags that can identify an unique transcript [1].

Although each sample (also called library, pattern, or example) contains a large number of features, each of which corresponds to a SAGE tag, the number of available samples is relatively small. In the dataset provided for the ECML/PKDD 2005 Discovery Challenge, there are 27679 features in each of the 90 samples, of which 59 are cancerous and 31 are normal. Gamberoni and Storari [1] formulated a binary classification problem in which each sample is labeled with its cancerous status, and the goal was to see whether some learning algorithms can build a meaningful and accurate classifier for this problem.

For such an *undersampled* problem, the goal of deducing a meaningful and accurate classifier is difficult to achieve. It is shown that even if the 90 samples are labeled randomly, some very weak classifiers, say, hyperplanes in $\mathbb{R}^{27679}$, can still almost always separate the samples perfectly [2]. Thus, a classifier that exactly fits all 90 given samples is not guaranteed to generalize well. Such phenomena, called overfitting by the machine learning community, are especially troublesome when there are so few samples available.

In addition, with such a large number of features, there is usually irrelevant information in some of them. Then, learning algorithms may erroneously use the irrelevant features to construct classifiers, which also leads to poor generalization performance.

P. Berka and B. Crémilleux (Eds.): ECML/PKDD 2005 Discovery Challenge, pp. 102–113, 2005.

Several principles and techniques may help to avoid overfitting. First, the Occam's razor [3], when applied to learning, prefers simpler classifiers over more complex ones. The rationale here is that a set of simpler classifiers is less likely to overfit the given samples, and thus may have smaller discrepancy between the training and generalization errors. Techniques based on this principle, such as regularization and structural risk minimization [2], usually lead to better generalization performance. All classification algorithms we use in this study have some degree of regularization.

Secondly, some techniques may help to focus on relevant information. For example, feature selection techniques try to filter out unimportant features and leave the processing of the useful features to later learning stage [4]. Feature selection on the SAGE dataset is useful for providing the relevance of each feature (SAGE tag) when predicting the desired label (status). Thus, we shall adapt and compare some feature selection techniques in this study.

Finally, error estimation techniques, such as the leave-one-out cross-validation and the v-fold cross-validation [5], can help in estimating the generalization performance as well as in selecting good parameters. However, any single estimator suffers the risk of being fitted if we compare too many algorithms or classifiers based on the estimator. Thus, we carefully use different estimations in this study to verify our findings, in order to draw a robust conclusion.

In this paper, we combine several techniques for classification, feature selection, and error estimation, and design a systematic approach to analyze undersampled problems. We take the SAGE dataset as a case study, and focus on deducing meaningful results for the associated binary classification problem. The sections are organized as follows. Section 2 describes the classification algorithms used in this study, and Section 3 introduces the feature ranking algorithms for feature selection. Then in Section 4 we present the experimental setting as well as our findings. Finally, we conclude in Section 5.

2 Classification Algorithms

We choose several classification algorithms that have been used for similar analysis in literature. These algorithms have different views on how the relationship between a sample and its label should be modeled. Somehow their success shares a similar nature: generating a classifier that has a reasonable complexity through some degree of regularization. Each algorithm has its advantages and disadvantages for the SAGE dataset, which are discussed individually in the subsections.

2.1 Adaptive Boosting with the Decision Stumps

The adaptive boosting (AdaBoost) [6] is perhaps the most popular and successful ensemble classification algorithm. For a given integer T and a weak hypothesis set $\mathcal{H}$, AdaBoost iteratively picks T weak hypotheses $h_t \in \mathcal{H}$ and weights $w_t \geq 0$

to construct an ensemble classifier

$$g(x) = \text{sign}\left(\sum_{t=1}^{T} w_t h_t(x)\right).$$

The ensemble classifier g aggregates the predictions of each individual h_t by a weighted voting. The aggregation allows AdaBoost to achieve good performance even with a simple $\mathcal{H}$. Tan and Gilbert [7] used AdaBoost with the decision trees for cancer classification with gene expressions, and found that AdaBoost can sometimes improve the generalization performance of decision trees dramatically.

Another widely used weak hypothesis set contains the decision stumps [8]. The decision stump $s_{q,d,\alpha}(x) = q \cdot \text{sign}((x)_d - \alpha)$ works on $(x)_d$, the d-th feature of a sample x, and classifies x according to $q \in \{-1, +1\}$ and the threshold α. The simplicity of decision stumps makes it possible to implement AdaBoost with the decision stumps (AdaBoost-Stump) efficiently.

Ben-Dor et al. [9] found that AdaBoost-Stump is sometimes superior to other algorithms for their tissue classification problems. If AdaBoost-Stump is applied to the SAGE dataset, each h_t in the obtained classifier can be thought as a thresholding rule for the associated SAGE tag, and the aggregation by w_t is a soft-version of the OR logic. After T iterations, AdaBoost-Stump automatically selects no more than T rules of SAGE tags that are worth investigating. That is, AdaBoost-Stump performs automatic feature selection and outputs an explainable classifier. These two properties make AdaBoost-Stump a favorable choice for gene expression datasets such as SAGE.

It was observed that AdaBoost-Stump can achieve good performances with some ad hoc values of T such as 100 or 1000 [9]. We shall present both settings in our results.

2.2 Support Vector Machine with the Linear Kernel

The support vector machine (SVM) is a classification algorithm which finds a separating hyperplane for labeled samples $(x_i, y_i) \in \mathbb{R}^D \times \{-1, +1\}$ with maximal margins [2]. That is, each sample would have a large distance to the constructed hyperplane. The obtained classifier is of the form

$$g(x) = \text{sign}(\langle w, \Phi(x) \rangle + b)$$

after solving a quadratic optimization problem for the values of w and b.

SVM is able to perform well in nonlinear classification problems. It transforms the sample x to a high-dimensional space with some mapping Φ, and constructs a hyperplane classifier there. In fact, the explicit form of Φ does not need to be used, and all necessary computation can be embedded into a special function called the kernel: $\mathcal{K}(x, x') = \langle \Phi(x), \Phi(x') \rangle$. In the most basic form, $\Phi(x) = x$ and $\mathcal{K}$, called the linear kernel, is just a basic inner product. Then, the classifier constructed by SVM is a simple hyperplane in $\mathbb{R}^D$

$$g(x) = \text{sign}\left(\sum_{d=1}^{D} w_d (x)_d + b\right) \tag{1}$$

1. Consider a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$.
2. Conduct a 5-fold cross-validation on Z with the soft-margin SVM and the linear kernel, with $C \in \{2^{-37}, 2^{-35}, \cdots, 2^{-17}\}$. Obtain C^* as the one with the lowest cross-validation error.
3. Apply the soft-margin SVM with the linear kernel and C^*, and output the obtained classifier.

Fig. 1. Steps of the SVM-Linear algorithm

In our study, we use the soft-margin SVM which has a parameter C that controls the trade-off between the magnitude of margin and the amount of margin violations allowed. Gamberoni and Storari [1] analyzed the SAGE dataset with the soft-margin SVM and the linear kernel using the default parameter ($C = 1$ with feature scaling). They found that this setting leads to good leave-one-out performance for the task. However, it is not clear whether such parameter would always be a good choice. Hsu and Lin [10] suggested selecting the value of C that achieves the lowest cross-validation error on the training set. In this study we adapt this procedure to our SVM with the linear kernel, denoted SVM-Linear, as shown in Fig. 1. Note that the parameter C is searched within some small values because we do not perform explicit feature scaling on the dataset.

As shown in (1), SVM-Linear outputs a hyperplane classifier in $\mathbb{R}^D$. Thus, it is possible to analyze the weight w_d of each feature (SAGE tag) for building the hyperplane. When $|w_d|$ is larger, the prediction of the hyperplane is more sensitive to the value of feature d, which implies that tag d is more important.

2.3 Support Vector Machine with the Gaussian Kernel

The linear kernel $\mathcal{K}(x, x') = \langle x, x' \rangle$ is not the only kernel that can be used with SVM. One of the most popular kernels is the Gaussian kernel, which is of the form $\mathcal{K}(x, x') = \exp(-\gamma \|x - x'\|_2^2)$. The Gaussian kernel implicitly defines a mapping Φ such that $\Phi(x)$ is a vector in an infinite dimensional space. Such property makes it possible to classify the labeled samples in a nonlinear and powerful manner. SVM with the Gaussian kernel was shown to have superior performance in some analysis of microarray gene expression data [11].

The Gaussian kernel has an additional parameter γ, which, similar to the parameter C, also needs to be decided carefully. It is known that the linear kernel can be thought as a special case of the Gaussian kernel under thorough parameter selection [12]. Thus, the Gaussian kernel is often considered to be more useful than the linear kernel [10]. However, the additional parameter γ makes the parameter selection step more time-consuming. We shall also adapt the procedure of Hsu and Lin [10] for SVM with the Gaussian kernel (SVM-Gauss), in which the best parameter pair (C^*, γ^*) would be searched with $C \in \{2^{-5}, 2^{-3}, \cdots, 2^{15}\}$ and $\gamma \in \{2^{-34}, 2^{-32}, \cdots, 2^{-16}\}$.

Another drawback of SVM-Gauss is that it is difficult to interpret the obtained classifier. The classifier embeds an infinite number of interactions between features in the kernel, and those interactions cannot be directly accessed or evaluated.

2.4 Support Vector Machine with the Stump Kernel

Lin and Li [13] constructed the stump kernel $\mathcal{K}(x, x') = -\|x - x'\|_1$ and showed that SVM with the stump kernel (SVM-Stump) has a strong connection to AdaBoost-Stump. They found that SVM-Stump usually performs better than AdaBoost-Stump due to the ability to construct infinite and dense ensemble of the decision stumps. They also showed that SVM-Stump has comparable classification power with SVM-Gauss, while having the advantage of faster parameter selection. Thus, we want to see whether SVM-Stump can also have those advantages on the SAGE dataset. We set up SVM-Stump with similar steps to SVM-Linear, with the difference that SVM-Stump uses the stump kernel and searches for C^* with $C \in \{2^{-17}, 2^{-15}, \cdots, 2^2\}$.

Although the stump kernel is nonlinear and defines a mapping to an infinite dimensional space, the classifier obtained from SVM-Stump can be partially interpreted. The classifier is proved to be equivalent to an ensemble classifier [13]

$$g(x) = \operatorname{sign}\left(\sum_{d=1}^{D} \sum_{q \in \pm 1} \left(\int w_{q,d}(\alpha) s_{q,d,\alpha}(x)\, d\alpha\right) + b\right).$$

Here, $s_{q,d,\alpha}$ is the decision stump mentioned in Subsection 2.1, and $w_{q,d}(\alpha)\, d\alpha$ is its infinitesimal weight. Unlike AdaBoost-Stump, in which only a finite T decision stumps are included in the ensemble classifier, SVM-Stump averages over an infinite number of them. Note that it is not hard to compute the aggregated weight

$$W_d = \sum_{q \in \pm 1} \int w_{q,d}^2(\alpha)\, d\alpha$$

from the classifier. The value of W_d measures the amount of decision stumps that are needed for feature (SAGE tag) number d. This is similar to the classifier constructed by AdaBoost-Stump, where each nonzero w_t indicates that the associated stump and the feature being worked on are important.

3 Feature Selection with Ranking Algorithms

We use feature selection techniques on the SAGE dataset to select most relevant features. Since we only have a limited number of samples, using a complicated feature selection technique, similar to using a complicated classification algorithm, increases the risk to overfit. Thus, we apply a rather simple technique: using a feature ranking algorithm to rank the features according to their importance, and selecting the best M features for learning. In the following subsections, we shall introduce three different feature ranking algorithms for our needs.

3.1 Feature Ranking with F-score

In NIPS 2003 Feature Selection Challenge, Chen and Lin [14] observed that the F-score could be a useful measure for ranking the features. They used the features with the best F-scores, and achieved good classification performance with SVM-Gauss. The F-score measures the discriminative power of a feature when only that feature is used for classification. When a feature d is more discriminative, its associated F-score F_d is larger. The disadvantage of F-score is that mutual information among features is not considered [14]. In this paper, we use *ranking with F-score* (RFS) for baseline comparison.

3.2 Feature Ranking with Linear Weights

We mentioned in Subsection 2.2 that the weight w_d in the classifier obtained from SVM-Linear measures the importance of feature d for the hyperplane. For example, Guyon et al. [15] used the measure with a backward elimination algorithm to select good features for gene classification problems.

For a given set of labeled samples with all features, we can run SVM-Linear (Fig. 1) on the samples, and use the magnitude of the linear weights $|w_d|$ for ranking the features. We call this procedure *ranking with linear weights* (RLW). It should be emphasized that the use of parameter selection in SVM-Linear is important for obtaining robust and representative values of w_d. If the parameters are not well-selected, the hyperplane classifier constructed by SVM could perform poorly and hence w_d is not meaningful anymore.

3.3 Ranking with Aggregated Stump Weights

Similar to the linear weight in SVM-Linear, we can also compute the aggregated stump weight W_d of feature d for the classifier obtained from SVM-Stump. As mentioned in Subsection 2.4, the aggregated stump weight represents the amount of decision stumps needed to describe the associated feature within the classifier. The larger W_d is, the more stumps on feature d are needed, which implies that feature d is more important. We follow similar steps as RLW and call this *ranking with aggregated stump weights* (RSW).

4 Experiment

4.1 Setting

In general, the generalization performance of an algorithm should be estimated using *unseen* samples. A common procedure is to use only parts of the given samples for training, and test with the holdout samples. However, such procedure would make inefficient use of the samples [5], and is especially wasteful when we only have a small dataset.

The v-fold cross-validation makes a more thorough use of the samples. The dataset $\mathcal{D}$ is randomly split into v mutually exclusive and approximately equal-size subsets (folds) $\mathcal{D}_1$, $\mathcal{D}_2$, …, $\mathcal{D}_v$. For iteration $k = 1, 2, \ldots, v$, the subset $\mathcal{D} \setminus \mathcal{D}_k$ is used for training the algorithm, and $\mathcal{D}_k$ is used for testing. Let μ_k be the test error on fold $\mathcal{D}_k$. The cross-validation error is $\frac{1}{|\mathcal{D}|} \sum_{k=1}^{v} |\mathcal{D}_k| \mu_k$, which is an estimator for the true generalization error.

An extreme of the v-fold cross-validation is called leave-one-out (LOO), in which v equals the number of samples. During the computation of the LOO estimator, the inner training set $\mathcal{D} \setminus \mathcal{D}_k$ is almost as large as $\mathcal{D}$. That is, almost all the given samples are used in training. This makes the LOO estimator theoretically profound and almost unbiased for estimating the generalization performance. The LOO estimator was used for some gene classification problems [1, 9]. However, it suffers from huge amount of computation when the number of samples is large, and from high variance when the number of samples is small [5].

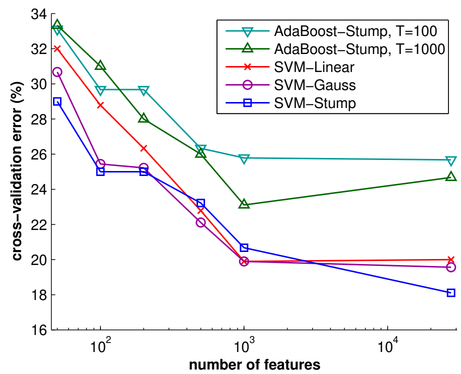
Other common choices of v are 3, 5, and 10. Kohavi [5] experimentally found that the 10-fold cross-validation performs most well on real-world datasets; Tan and Gilbert [7] applied the 10-fold cross-validation for their study of a gene expression dataset. Note that it is not possible to have a definitely best choice for estimation, because no estimator can be correct all the time [5]. Only looking at one estimator also suffers the risk of being fitted if we evaluate too many algorithms, and causes wrong interpretation on the results. The risk is especially high when the dataset is small and the estimator may have a large variance (e.g., LOO). Thus, for undersampled problems, results using different v should be compared in order to draw a robust conclusion. With similar computational time, we could do a 90-fold cross-validation on the SAGE dataset (LOO), a 10-fold cross-validation 10 times, or a 5-fold cross-validation 20 times. We shall report the results of all these experiment settings.

To summarize, our experiment setting is based on the v-fold cross-validation, where $v = 5, 10$ or 90. A complete algorithm is constructed by cascading ranking and classifying steps, where the ranking step is one of RFS, RLW, or RSW, and the classifying step is one of AdaBoost-Stump, SVM-Linear, SVM-Gauss, or SVM-Stump. An algorithm only has access to the training folds, and the test folds is kept unseen both to the ranking step and to the classifying step. After the ranking step, we select the top M features, where M varies in $\{50, 100, 200, 500, 1000, 27679\}$. When $M = 27679$, it is as if no feature ranking/selection is done, and the full set is used. After feature ranking and selection, the classification algorithm is applied to the simplified training folds with only those M features, and the error is estimated on the simplified test fold with M features.

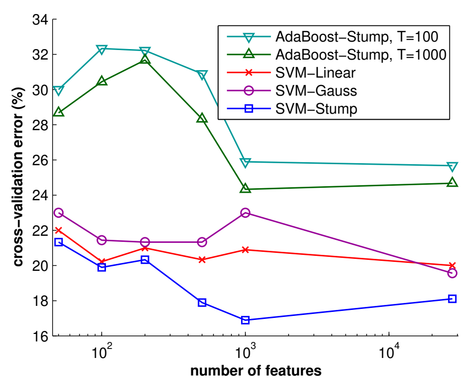
We use LIBSVM [16] for the SVM algorithms. The RFS and RLW procedures are also downloaded in the associated tools page of LIBSVM.

4.2 Result

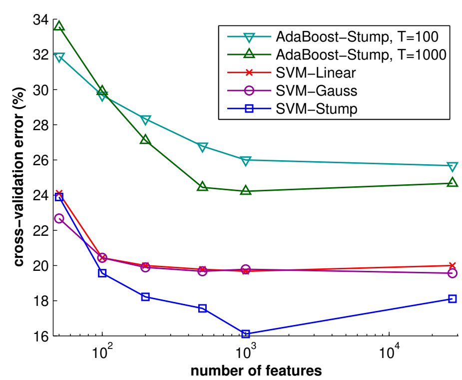
We start from analyzing the 10-fold cross-validation result, which seems to be the most reliable one among the three cross-validation settings we have tried.



(a) RFS



(b) RLW

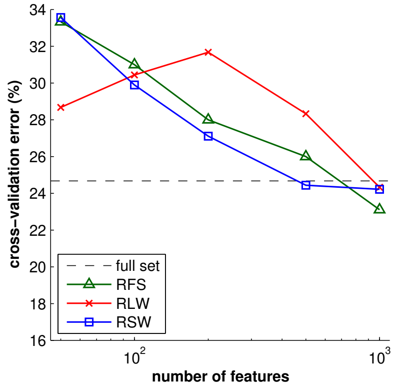


(c) RSW

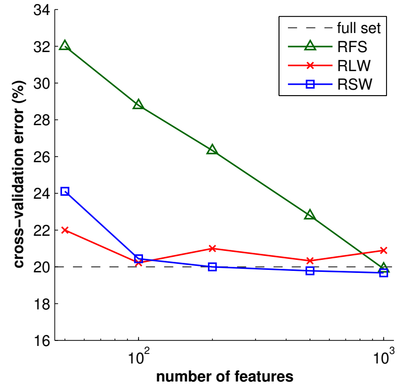
Fig. 2. The 10-fold cross-validation errors on the SAGE dataset. Markers on each curve mark the errors of feature set size 50, 100, 200, 500, 1000, and the full dataset.

In Fig. 2 we show the averaged 10-fold cross-validation error of different classification and feature ranking algorithms. In general, the error goes down when M, the number of features selected, gets larger. but the error with full features is usually comparable to the one with 1000 features, if not higher, which validates that learning might be deteriorated by irrelevant features. Overall, it seems that 500–1000 features can represent the dataset well.

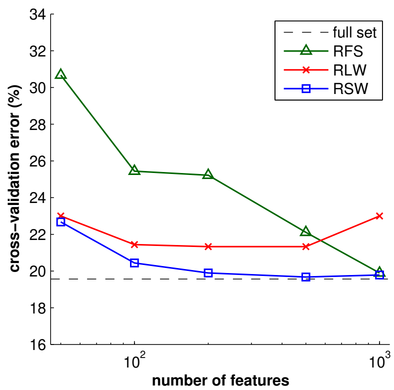
Although the standard error is around 1.3%, we can still conclude that AdaBoost-Stump performs much worse than the SVM algorithms. In addition, SVM-Stump is almost always the best, although the edge is not significant when the feature ranking is done with RFS. SVM-Linear and SVM-Gauss perform



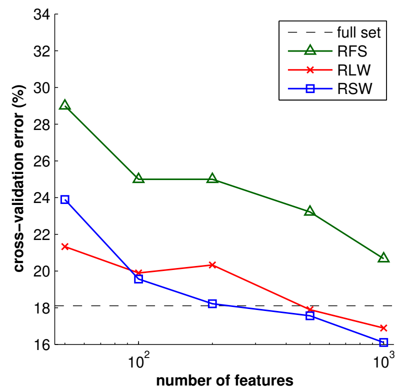
(a) AdaBoost-Stump, $T = 1000$



(b) SVM-Linear



(c) SVM-Gauss



(d) SVM-Stump

Fig. 3. The 10-fold cross-validation errors on the SAGE dataset. The error of using the full dataset is plot as the broken horizontal line.

quite similarly. Since SVM-Gauss requires much more computational effort in parameter selection, and the obtained classifier is less explainable, SVM-Linear is preferable to SVM-Gauss.

Another way to look at these errors is to organize them according to the learning algorithms. In Fig. 3 we show the 10-fold cross-validation errors of each learning algorithm with different feature ranking algorithms. It is quite clear that feature selection with RFS works the worst, and feature selection with RSW overall could achieve the lowest error. However, RLW is comparable to

Table 1. The 5-fold cross-validation errors on the SAGE dataset, with standard error shown in parenthesis.

ranking	classifying	50	100	200	500	1000	27697 [full]
RLW	SVM-Linear	22.9(1.0)	21.3(0.9)	22.2(0.9)	21.0(0.9)	21.4(1.0)	22.1(0.9)
RSW	SVM-Linear	25.6(1.0)	22.6(1.0)	21.3(0.9)	21.3(1.0)	21.8(0.9)	22.1(0.9)
RLW	SVM-Stump	22.5(0.9)	21.6(0.8)	21.3(0.8)	20.4(0.8)	19.1(0.9)	19.1(0.8)
RSW	SVM-Stump	22.6(0.8)	21.1(0.9)	20.2(0.9)	19.1(0.9)	18.7(0.8)	19.1(0.8)

Table 2. The 10-fold cross-validation errors on the SAGE dataset, with standard error shown in parenthesis.

ranking	classifying	50	100	200	500	1000	27697 [full]
RLW	SVM-Linear	22.0(1.3)	20.2(1.3)	21.0(1.4)	20.3(1.3)	20.9(1.3)	20.0(1.3)
RSW	SVM-Linear	24.1(1.4)	20.4(1.2)	20.0(1.2)	19.8(1.3)	19.7(1.3)	20.0(1.3)
RLW	SVM-Stump	21.3(1.3)	19.9(1.2)	20.3(1.3)	17.9(1.2)	16.9(1.2)	18.1(1.2)
RSW	SVM-Stump	23.9(1.2)	19.6(1.4)	18.2(1.2)	17.6(1.2)	16.1(1.2)	18.1(1.2)

RSW in some settings. For example, when the number of selected features is small (say 50), RLW usually works the best.

Since the 5-fold cross-validation uses less samples for training, the estimated errors have larger positive bias and are thus higher than those from the 10-fold cross-validation. However, the conclusion from these numbers on the best learning or feature ranking algorithms is similar. Thus, we do not include plots, but just list the 5-fold cross-validation errors of the best two classification algorithms and the best two ranking algorithms in Table 1. The 10-fold cross-validation results are similarly shown in Table 2.

If we carefully look at the 5-fold and 10-fold cross-validation results, we can see that SVM-Stump is useful for classification, and RSW is useful for feature selection. However, we should also notice that none of the differences between SVM-Linear and SVM-Stump, as well as the differences between RLW and RSW, are significant with a confidence level of 95% and corrected resampled t-test [17, 18].

In Table 3, we show the errors of the LOO cross-validation. While some observations confirm our findings in the 5-fold and 10-fold settings, the large standard error as well as the oscillating estimates make it hard to draw a reliable conclusion on, for example, a suitable size for selecting features. The instability of LOO results are also observed in the experiments with other classification and feature ranking algorithms. Thus, even if LOO results are almost unbiased, it can be risky to rely on these results for evaluating the performance of algorithms on the SAGE dataset.

5 Conclusion

In this paper, we combined several state-of-the-art techniques for classification, feature selection, and error estimation, and applied them for studying SAGE results.

Table 3. The leave-one-out cross-validation errors on the SAGE dataset, with standard error shown in parenthesis.

ranking	classifying	50	100	200	500	1000	27697 [full]
RLW	SVM-Linear	21.1(4.3)	18.9(4.1)	17.8(4.0)	21.1(4.3)	22.2(4.4)	21.1(4.3)
RSW	SVM-Linear	24.4(4.5)	18.9(4.1)	18.9(4.1)	20.0(4.2)	18.9(4.1)	21.1(4.3)
RLW	SVM-Stump	22.2(4.4)	22.2(4.4)	20.0(4.2)	21.1(4.3)	14.4(3.7)	17.8(4.0)
RSW	SVM-Stump	27.8(4.7)	20.0(4.2)	21.1(4.3)	16.7(3.9)	16.7(3.9)	17.8(4.0)

We have shown that some classification techniques can achieve good performance. For example, SVM-Stump can achieve promising performance either when applied to the full set or when applied to the dataset with selected features. Nevertheless, it is beyond our knowledge to assess whether this modeling is reasonable.

In many cases, SVM-Linear also achieves similar performance to SVM-Stump. However, Ng [19] showed that SVM-Linear, which is a rotational invariant algorithm, requires a worst-case sample complexity that grows at least linearly in the number of features. In this sense, SVM-Linear may not be a good choice for undersampled datasets like SAGE.

We have also demonstrated that feature selection is important for the SAGE dataset, and a good feature selection algorithm alone with a suitable size of features can achieve better performance than the full set of features. We have shown that a feature set of size 500 to 1000 from RSW feature ranking, smaller than 4% of the original SAGE dataset, is good enough for predicting the cancerous status of the sample.

We have estimated the generalization errors of each algorithm by trying different folds of cross-validation. We have shown that the commonly-used leave-one-out estimator, albeit theoretically almost unbiased, has a large variation and makes it hard to interpret the results. Hence, a v-fold cross-validation with a smaller v would be more desirable.

Although the techniques provide promising results, we hope to know from SAGE experts on whether the modeling of those learning algorithms is reasonable. It is known in machine learning that any small piece of expert information may dramatically improve the performance of learning algorithms [20]. The SAGE dataset only provides the algorithms limited number of samples to conquer the task. Thus, more information, whether in the form of samples, known importance of features, or the reasonability of classifiers, could possibly help us design better techniques to further analyze the SAGE dataset.

Acknowledgment

We thank Yaser Abu-Mostafa, Amrit Pratap, and the anonymous reviewers for valuable suggestions. This work has been mainly supported by the Caltech Center for Neuromorphic Systems Engineering under the US NSF Cooperative

Agreement EEC-9402726. Ling Li is currently sponsored by the Caltech SISL Graduate Fellowship.

References

1. Gamberoni, G., Storari, S.: Supervised and unsupervised learning techniques for profiling SAGE results (2004) Presented in workshop on the ECML/PKDD 2004 Discovery Challenge.
2. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, New York (1998)
3. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Occam's razor. Information Processing Letter **24** (1987) 377–380
4. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3** (2003) 1157–1182
5. Kohavi, R.: A study on cross-validation and bootstrap for accuracy estimation and model selection. In Mellish, C.S., ed.: IJCAI 95. Volume 2., Montréal, Québec, Canada, Morgan Kaufmann (1995) 1137–1145
6. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Machine Learning: Proceedings of the 13th International Conference. (1996) 148–156
7. Tan, A.C., Gilbert, D.: Ensemble machine learning on gene expression data for cancer classification. Applied Bioinformatics **2** (2003) S75–S83
8. Holte, R.C.: Very simple classification rules perform well on most commonly used datasets. Machine Learning **11** (1993) 63–91
9. Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M., Yakhini, Z.: Tissue classification with gene expression profiles. Journal of Computational Biology **7** (2000) 559–584
10. Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification. Technical report, National Taiwan University (2003)
11. Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., Furey, T., M. Ares, J., Haussler, D.: Knowledge-based analysis of microarray gene expression data by using support vector machines. Proceedings of the National Academy of Sciences of the United States of America **97** (2000) 262–267
12. Keerthi, S.S., Lin, C.J.: Asymptotic behaviors of support vector machines with Gaussian kernel. Neural Computation **15** (2003) 1667–1689
13. Lin, H.T., Li, L.: Infinite ensemble learning with support vector machines. In: Machine Learning: ECML 2005, Porto, Portugal, Springer-Verlag (2005) 242–254
14. Chen, Y.W., Lin, C.J.: Combining SVMs with various feature selection strategies (2005) To appear in the book "Feature extraction, foundations and applications".
15. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning **46** (2002) 389–422
16. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. (2001) Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
17. Weston, J., Elisseff, A., Schölkopf, B., Tipping, M.: Use of the zero-norm with linear models and kernel methods. Journal of Machine Learning Research **3** (2003) 1439–1461
18. Nadeau, C., Bengio, Y.: Inference for the generalization error. Machine Learning **52** (2001) 239–281
19. Ng, A.Y.: Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In: Machine Learning: Proceedings of the 21th International Conference. (2004)
20. Abu-Mostafa, Y.S.: Learning from hints. Journal of Complexity **10** (1994) 168–178