Serdica J. Computing **8** (2014), No 2, 137–168

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

A COMPARATIVE ANALYSIS OF PREDICTIVE LEARNING ALGORITHMS ON HIGH-DIMENSIONAL MICROARRAY CANCER DATA

Jo Bill, Ernest Fokoué

Abstract. This research evaluates pattern recognition techniques on a subclass of big data where the dimensionality of the input space (p) is much larger than the number of observations (n). Specifically, we evaluate massive gene expression microarray cancer data where the ratio $\kappa = n/p$ is less than one. We explore the statistical and computational challenges inherent in these high dimensional low sample size (HDLSS) problems and present statistical machine learning methods used to tackle and circumvent these difficulties. Regularization and kernel algorithms were explored in this research using seven datasets where $\kappa < 1$. These techniques require special attention to tuning necessitating several extensions of cross-validation to be investigated to support better predictive performance. While no single algorithm was universally the best predictor, the regularization technique produced lower test errors in five of the seven datasets studied.

ACM Computing Classification System (1998): C.3, C.5.1, H.1.2, H.2.4., G.3.

Key words: HDLSS, machine learning algorithm, pattern recognition, classification, prediction, regularization, discriminant analysis, support vector machine, kernels, cross validation, microarray cancer data.

1. Introduction. According to *A Cancer Journal for Clinicians* [25], although cancer death rates are on the decline, cancer remains a major health problem in many parts of the world. In the United States alone one in four deaths are due to cancer. A total of 1,665,540 new cancer cases and 585,720 cancer deaths are projected to occur in the United States in 2014. With the advent of gene expression microarray technology, cancer is predominantly explored and studied through datasets gathered from microarray probes.

Microarray cancer data is an example of HDLSS data where the sample size n is very small, usually in the tens, and the number of variables p is massively large, numbering in the thousands. For this research, seven existing gene expression microarray cancer datasets were chosen, four binary and three multiclass.

Table 1. These datasets are freely available to download and use through the following resources: Prostate [26], Colon [1], Leukemia [12], WestBC [31], Lung [2], Breast [16], Brain [21]

	n	p	# Classes	κ	Distribution of classes				
Prostate Cancer	79	500	2	0.15800	37	42			
Colon Cancer	62	2000	2	0.03100	22	40			
Leukemia Cancer	72	3571	2	0.02016	47	25			
West BC Cancer	49	7129	2	0.00687	24	25			
Lung Cancer	197	1000	4	0.19700	139	17	21	20	
Breast Cancer	97	1213	3	0.07997	11	50	36		
Brain Cancer	42	5597	5	0.00750	10	10	10	4	8

When n is very small and the dimension of the space of variables p is extremely large $n \lll p$, the underlying statistical problem becomes ill-posed or ill-defined [9]. Traditional statistical techniques of relying on strong model assumptions analogous to parametric or being distribution free as with fully non-parametric are inadequate, and inference breaks down. This failure of traditional statistics is due to a phenomenon known as ill-posedness. An explanation for this circumstance can be found in the works of Hadamard [14], where he states that for a problem to be well-posed it must meet the following conditions:

- A solution must exist
- The solution must be unique
- The solution must be stable so that the inverse mapping is continuous

In dealing with HDLSS microarray cancer data these conditions were continually violated, leaving traditional methods performing badly or not at all. Fortunately, modern statistics is forging ahead where traditional methods failed and

adeptly providing computer-intensive machine learning techniques employing statistical algorithms that are proving to be quite skillful at dealing with these types of HDLSS data where $\kappa < 1$. This paper compares and contrasts the ability of various machine learning techniques to perform accurate prediction on microarray cancer data via supervised learning.

The following four approaches can be used in machine learning as techniques for dealing with ill-posed problems.

- Regularization (Introduces a small amount of bias through the constraint but stabilizes the variance) [28]
- Kernelization (Captures arbitrary nonlinear decision boundaries in high-dimensional space) [29]
- Randomized Ensemble (A collection of models averaged out to stabilize variance) [23], [6], [3]
- Feature Selection (Extraction of meaningful markers) [11], [13]

The two main goals when analyzing microarray cancer data are accurate prediction of a given disease from a set of corresponding gene profiles, and gene selection. The focus of this paper is on obtaining accurate prediction by employing the more commonly used techniques of regularization and kernelization.

The remainder of the paper is organized as follows. In section 2, we present a detailed description of the techniques used and analyzed, with a focus on the aspects that make them suitable for microarray data. Section 3 is dedicated to the description of our data preprocessing and various extensions of cross-validation used to optimize the tuning parameters. Section 4 presents the results from our computational experiments with an emphasis on the comparison of the average test errors across all the methods considered. Section 5 concludes with a discussion of our findings and the strengths and weaknesses of each of the methods. We also provide elements of our future work.

2. Predictive learning methods.

Discriminant Analysis. In general the goal of discriminant analysis is to build the best classifier function f that assigns points to one of several classes or labels $y \in \{1, 2, \ldots, k, \ldots, K\}$ based on a set of measurements $\boldsymbol{X} = (x_1, x_2, \ldots, x_p)$ such that the classification error is as small as possible [11], [9]. This function is constructed from a posterior distribution created via the majority rule concept where new points, or observations, are assigned to a class k that maximizes their probability, which equivalently minimizes their loss.

Each observation is assumed to be a member of one and only one class. An error is incurred when an observation is assigned to the incorrect class [11].

The loss from this error is defined as

$$L(k, \hat{k}), \qquad 1 \leq k, \qquad \hat{k} \leq K \tag{1}$$

Thus the expected loss or risk incurred when classifying vector $\boldsymbol{X}$ as $\hat{k}$ is

$$R(\hat{k} \mid \boldsymbol{X}) = \frac{\sum_{k=1}^{K} L\left(k, \hat{k}\right) p_k(\boldsymbol{x})\pi_k}{\sum_{k=1}^{K} p_k(\boldsymbol{x})\pi_k} \tag{2}$$

where $p_k(\boldsymbol{x}) \equiv$ Class conditional density of $\boldsymbol{x}$ in class k
and $\pi_k = \Pr[y = k]$ based on prior class membership probability
The posterior class membership probability is

$$d_k(\boldsymbol{x}) = \Pr[y = k \mid \boldsymbol{x}] = \frac{\pi_k p_k(\boldsymbol{x})}{\sum_{l=1}^{K} \pi_l p_l(\boldsymbol{x})} \tag{3}$$

By choosing the optimal $\hat{k}$ for minimization of loss, Equation (2) can be simplified to

$$\hat{f}_{DA}(\boldsymbol{X}) = \widehat{\text{class}(\boldsymbol{x})} = \hat{k} = \underset{k=1,\ldots,K}{\text{argmax}} \{\pi_k p_k(\boldsymbol{x})\} \tag{4}$$

By assigning a one-unit loss to each mistake or observation incorrectly classified, the misclassification risk simply becomes a fraction of assignments that are incorrect. The resulting rule (4) from choosing the optimal $\hat{k}$ to minimize the risk function $R(\hat{k} \mid \boldsymbol{X})$ is the Bayes rule. The Bayes rule will achieve the minimum misclassification risk among all possible rules [11].

However, since conditional densities $p_k(\boldsymbol{X})$ are seldom known the observations in the training sample are used to construct the classification rule from estimates of $p_k(\boldsymbol{X})$. This results in added complexity when dealing with high-dimensional data.

When the class prior probabilities are also unknown, we use the training data once again as a random sample from the pooled population distribution. The prior probabilities are then estimated by the fraction of each class in the pooled training set [11]. This process emphasizes the importance of using stratified sampling when creating training data sets in order to maintain the integrity of the prior probability estimates $\hat{\pi}_k$.

To estimate the prior probabilities of each class in the pooled sample we use the following equations where v is the i^{th} observation and $c(v)$ is the class of the v^{th} observation. W_v is a weight or mass assigned to each observation [11].

$$\hat{\pi}_k \quad = \quad W_k/W \tag{5}$$

$$W_k = \sum_{c(v)=k} w_v \tag{6}$$

$$W = \sum_{k=1}^{K} W_k \tag{7}$$

Linear Discriminant Analysis (LDA). In linear discriminant analysis (LDA) the classification rule is based upon a normal distribution with the data in each group or class following a Multivariate Gaussian distribution with means μ_k and common covariance matrix Σ [9]. As a result of the covariance matrices between the classes being relatively close and assumed equal such that $\Sigma_0 = \Sigma_1 = \Sigma$, the decision boundary and discriminant function are inherently linear. In this special case of discriminant analysis the discriminant function becomes

$$d_k(\boldsymbol{X}) = \boldsymbol{X}^\top(\Sigma^{-1}\boldsymbol{\mu}_k) - \frac{1}{2}\boldsymbol{\mu}_k^\top\Sigma^{-1}\boldsymbol{\mu}_k + \ln \pi_k \tag{8}$$

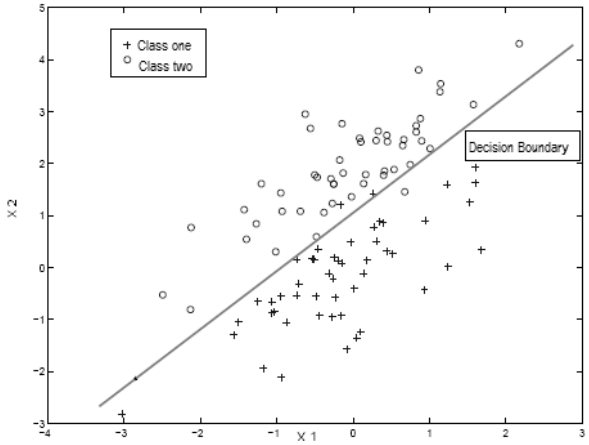

Fig. 1. Simple binary decision boundary (2-dimensions)

The assumption that the covariance matrices are equal is fundamental when dealing with sparse datasets because it means only one inverse covariance matrix needs to be estimated which is key when dealing with small sample sizes.

However, when dealing with HDLSS data a technique must also be able to handle variable sizes p several times larger than sample size n. When faced with this $n \lll p$ scenario, as in the microarray cancer data, LDA suffers from multicollinearity and is prone to overfitting the data. This happens because the

dimension d of the data vectors is much larger than the data vectors available in the sample size n.

This situation was encountered repeatedly in the research documented in this article. Every microarray cancer dataset listed here obtained warnings of collinear variables when executed with LDA. LDA struggled to classify the data due to the covariance matrix Σ^{-1} being essentially singular. In fact, no direct matrix inversion is actually used.

Even though these datasets have already been reduced through feature selection and preprocessing, for the LDA technique to work optimally, further reduction in variable size would be required. In our research, LDA is used solely for the purpose of providing a baseline.

A practical implementation of LDA is given in the R package MASS [30].

Quadratic Discriminant Analysis (QDA). In the case where the covariance matrices between classes are not equal, the quadratic discriminant analysis (QDA) technique is normally employed. The quadratic discriminant function containing the Mahalanobis distance between $\boldsymbol{X} - \boldsymbol{\mu}_k$ [11] is given as

$$d_k(\boldsymbol{X}) = \ln|\Sigma_k| + (\boldsymbol{X} - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\boldsymbol{X} - \boldsymbol{\mu}_k) - 2\ln \pi_k \tag{9}$$

However, when faced with the situation where $n \lll p$, the QDA technique will not run. The sample sizes are too small to estimate more than one inverse covariance matrix. Therefore QDA is not an option for the microarray cancer data in this research.

A practical implementation of QDA is given in the R package MASS [30].

Regularized Discriminant Analysis (RDA). The failure of LDA and QDA prompts the need for regularization to stabilize the covariance matrix Σ_k. In the presence of high dimensionality, the variance estimator is typically very high and in the process of reducing the variance of the estimating function, some bias is inevitably introduced. Regularization of the estimating parameters is therefore a common protocol for introducing a small amount of bias into the covariance matrix in exchange for the gain in stabilizing the variance and thereby reducing generalization error.

There are several variations of regularized discriminant analysis available. In this research the shrunken centroids regularized discriminant analysis from the R rda package was chosen [13].

The shrunken centroid regularized discriminant analysis, hereafter referred to as RDA, strives to minimize the within-class scatter while maximizing the

between-class scatter in order to define the feature vectors that allow the high-dimensional data to be projected onto a low-dimensional feature space to facilitate maximum class separation.

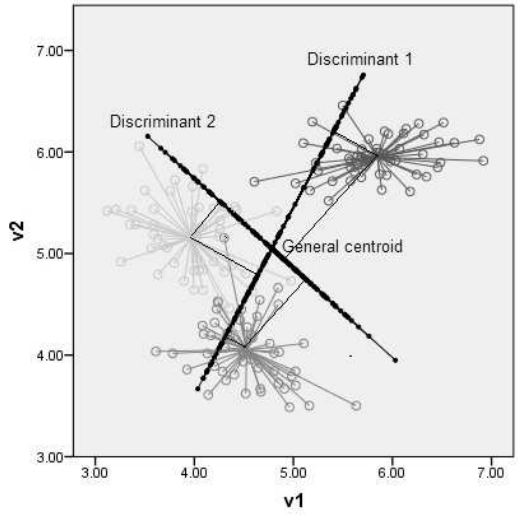

Fig. 2. Shrunken centroids for a multiclass dataset (2 dimensions)

This RDA method uses regularization of the LDA method as a way to resolve the singularity problem that occurs when the sample covariance matrix is singular and cannot be inverted. Having a covariance matrix that is singular is an expected impediment of dealing with $n \lll p$ data. The regularization hyper-parameter α is used here. Regularization of $\hat{\Sigma}$ to resolve singularity and stabilize the covariance estimate:

$$\tilde{\Sigma} = \alpha\hat{\Sigma} + (1-\alpha)I_p \quad \text{where } 0 \leq \alpha < 1 \tag{10}$$

The regularized discriminant function becomes

$$\tilde{d}_k(\boldsymbol{X}) = \boldsymbol{x}^\top \tilde{\Sigma}^{-1}\bar{x}_k - \frac{1}{2}\bar{x}_k^\top \tilde{\Sigma}^{-1}\bar{x}_k + \log \pi_k \tag{11}$$

RDA handles shrinkage slightly different than the other conventional shrinkage methods. Rather than shrinking the centroids directly as done in the "nearest shrunken centroids" method [27], the shrunken centroids RDA method [13] shrinks in the following way:

$$\text{Shrinkage of } \bar{\boldsymbol{x}}^* = \tilde{\Sigma}^{-1}\bar{\boldsymbol{x}} \tag{12}$$

Resulting in the following formula with Δ as the shrinkage hyper-parameter

$$\bar{\boldsymbol{x}}^{*\prime} = \operatorname{sgn}(\bar{\boldsymbol{x}}^{*})\,(|\bar{\boldsymbol{x}}^{*}| - \Delta)_{+} \quad \text{where} \quad 0 \leq \Delta \leq 3 \tag{13}$$

The bias variance tradeoff for RDA is now optimized through careful tuning of the two hyper-parameters, α for regularization and Δ for shrinkage (Fig. 3). The hyper-parameter values jointly producing the minimal error, thus concurrently maximizing classification performance, are determined from the use of cross-validation as explained in the *Preprocessing and Tuning* section of this paper.

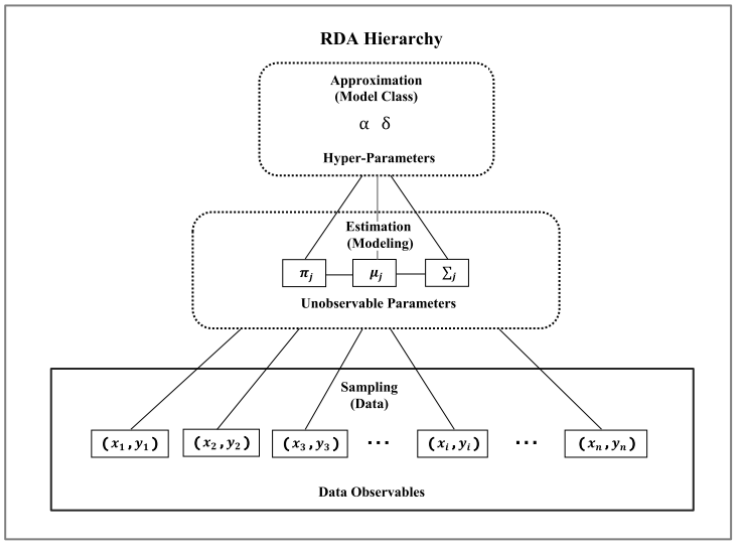

Fig. 3 The RDA technique represented pictorially to show the hierarchy leading to the two tuning hyper-parameters alpha and delta

Support Vector Machines (SVM). When using the support vector machine (SVM) technique for classification the goal is to use $\boldsymbol{x}_i$, the vector of explanatory variables, to estimate the decision boundary that best separates the classes or y_i labels [29], [9]. In the simple binary case (Figure 4) where $p = 2$, the two classes separate linearly and the boundary between the two classes is called the hyperplane represented by $\boldsymbol{w}^{\top}\Phi(\boldsymbol{x}) + b = 0$, where b is equivalent to $\boldsymbol{w}_0$ and Φ is the function that maps the vector of $\boldsymbol{x}_i$'s into a higher dimensional space [29].

For support vector machines, optimal prediction comes from jointly maximizing the margin while minimizing the misclassified points or observations. In

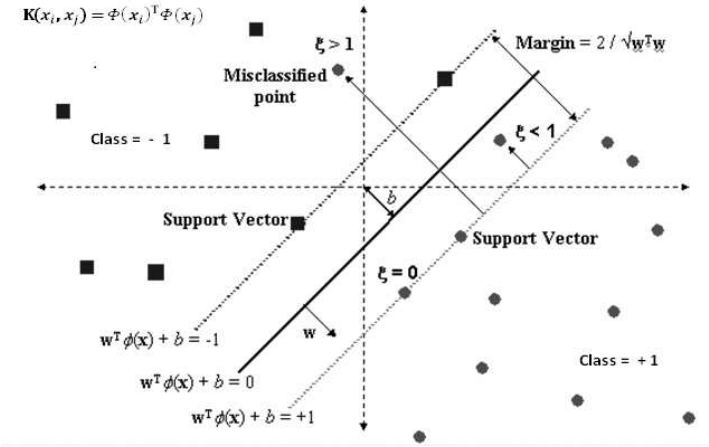

Fig. 4 Support vector machine

Figure 4, the margin is identified as the distance between the $\boldsymbol{w}^\top \Phi(\boldsymbol{x}) + b = -1$ line and $\boldsymbol{w}^\top \Phi(\boldsymbol{x}) + b = +1$ line. The misclassified points are shown by the arrows pointing to the two observation points labeled $\xi > 1$ and $\xi < 1$. The four observation points found on the two lines defining the margin are called support vectors. These observations or support vectors help to define the margin. In general, if two models perform equally in prediction and are the same with respect to complexity, it is best to choose the model defined with the least amount of support vectors.

Reiterated, with support vector machines the prediction goal is to find the optimal hyperplane that separates the data with the least errors while simultaneously maximizing the margin (14). It can be found by satisfying the conditions of formula (16) while minimizing (15) [29].

Maximize the margin

$$\rho = \frac{2}{\|\boldsymbol{w}\|} = \frac{2}{\sqrt{\boldsymbol{w}^\top \boldsymbol{w}}}. \tag{14}$$

The SVM binary classifier is found by minimizing

$$\frac{1}{2}(\boldsymbol{w}^\top \boldsymbol{w}) + \frac{C}{n}\sum_{i=1}^{n} \xi_i \tag{15}$$

subject to the constraints

$$y_i(\boldsymbol{w}^\top \Phi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i, \text{ and } \xi_i > 0, \ i = 1, \ldots, n. \tag{16}$$

The estimating equation is given by

$$\hat{f}_{SVM}(\boldsymbol{x}_{new}) = \text{sign}\left(\sum_{j=1}^{n} \hat{\alpha}_j y_j \Phi(\boldsymbol{x}_j)^\top \Phi(\boldsymbol{x}_{new}) + \hat{b}\right) \tag{17}$$

To this purpose, attempting to identify this optimal hyperplane in the original q-dimensional input space can be extremely cumbersome and difficult, if possible at all. Additionally, in the case of multiclass data, there are multiple intersecting hyperplanes that need to be found. The actual computations involved in training a support vector machine to find a hyperplane require solving a quadratic optimization problem [29], [18]. Attempting to train an algorithm on a HDLSS dataset using a standard quadratic problem solver for training would be unmanageable and impractical.

Opportunely, since the inner product in feature space is exactly equal to a non-linear transformation in input space [29], by projecting the vector of $\boldsymbol{x}_i$'s from input space into a higher dimensional feature space, solving a quadratic optimization problem becomes attainable. As illustrated in Figure 5, for each point $\boldsymbol{x}_i$ in q-dimensional space there is an equivalent $\Phi(\boldsymbol{x}_i)$ in feature space. The projection or mapping of these $\boldsymbol{x}_i$ points into a higher dimension makes it possible to find the hyperplane that separates the data optimally in feature space and consequently avoids the much more complicated decision boundary shown in the input space (Fig. 5).

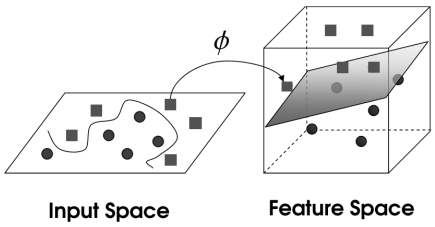

Fig. 5. Mapping input space to feature space

Unquestionably the explicit mapping of the $\boldsymbol{x}_i$'s is labor-intensive and problematic due to the non-linear, high dimensionality of the mapping. Fortunately, the kernel function Φ facilitates implicitly mapping the input data points into feature space and returning the inner product between the images of two data points from feature space [18]. This kernelization of the SVM classifier enables each $\boldsymbol{x}$ to obtain an estimated response. This is often referred to in literature as the "kernel trick" [24].

Projection Φ

$$\forall \Phi : \mathcal{X} \to \mathcal{F}$$

Kernel function of K

$$\exists K(\cdot,\cdot) \text{ such that } K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle = \Phi(\boldsymbol{x}_i)^\top \Phi(\boldsymbol{x}_j)$$

Kernelized SVM classifier

$$\hat{f}_{SVM}(\boldsymbol{x}_{new}) = \text{sign}\left(\sum_{j=1}^{n} \hat{\alpha}_j y_j K(\boldsymbol{x}_j, \boldsymbol{x}_{new}) + \hat{b} \right) \tag{18}$$

Each kernel $K(\cdot,\cdot)$ defines a flexible class of base functions indexed by one or more tuning parameters. Once the proper kernel for the data is identified, the calculation of the mapping of Φ happens implicitly through tuning the kernel. Given the critical importance of choosing the best kernel for a given dataset, the following kernels were explored in this research.

• Vanilla or linear kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle = \boldsymbol{x}_i^\top \boldsymbol{x}_j \tag{19}$$

• Polynomial kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\text{scale} \cdot \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \text{ offset})^{\text{degree}} = (\gamma \cdot \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \tau)^{\text{degree}} \tag{20}$$

where $\gamma \equiv$ scale and typically set to 1 and τ is set to 1 or 0

• Gaussian Radial Basis Function (RBF) kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2) \tag{21}$$

where $\gamma \equiv$ bandwidth and

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \sum_{\ell=1}^{P} (\boldsymbol{x}_{i\ell} - \boldsymbol{x}_{j\ell})^2$$

• Laplace RBF kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|) \tag{22}$$

where $\gamma \equiv$ bandwidth and

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\| = \sum_{\ell=1}^{P} |\boldsymbol{x}_{i\ell} - \boldsymbol{x}_{j\ell}|$$

• Hyperbolic tangent or sigmoid kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\text{scale} \cdot \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \text{offset}) = \tanh(\gamma \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \tau) \tag{23}$$

where $\gamma \equiv$ scale and τ is the offset

• ANOVA radial basis kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\sum_{k=1}^{n} \exp(-\gamma(\boldsymbol{x}_i^k - \boldsymbol{x}_j^k)^2) \right)^d \tag{24}$$

where $\gamma \equiv$ bandwidth and d is the degree.

The support vector packages e1071 and kernlab were evaluated in this research for their ability to correctly classify HDLSS microarray cancer data through supervised learning. Once again these tuning parameters (Figure 6) are optimized utilizing cross-validation as explained in the *Preprocessing and Tuning* section of this paper.

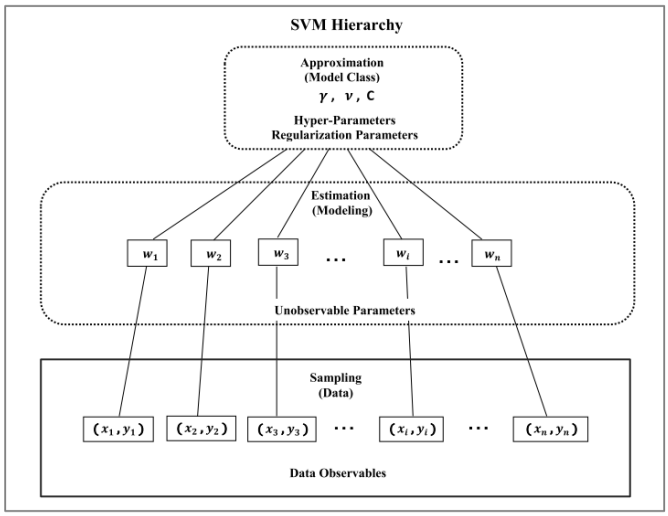

Fig. 6. The SVM technique is represented pictorially to show the hierarchy leading to tuning parameters γ, ν, and C or cost

SVM (e1071) and (kernlab). These two packages are very similar in mathematical theory, as outlined previously, but have different technical implementations. Both packages offer a C and ν classification type for binary datasets with C being the default for both when y is a factor. The C classification uses a

tuning parameter as a regularization parameter to control the cost of a constraint violation in an attempt to balance the tradeoff between a wide margin and classification errors [20]. Essentially, this is the 'C'-constant of the regularization term in the Lagrange formulation [19]. The C formulation is trying to control the dimensionality of the p by p matrix and will perform better on data that has a more discrete distribution [9]. The nu classification type works by controlling the error rather than the dimensionality directly [9]. It uses the tuning parameter ν to control the ratio of support vectors to data points [20]. The ν parameter is bounded between 0 and 1 and the higher the value of ν is the wider the margin will be [19].

Classification for multiclass datasets is offered by both packages. Package e1071 documents that SVMs can only solve binary classification problems and therefore handles multiclass datasets using a one-against-one technique that fits all of the binary subclassifiers and then finds the correct class via a voting mechanism [20]. The kernlab package offers a native multiclass classification formulation through two additional classification types: spoc-svc, based on Crammer, Singer native multi-class and kbb-svc, based on Weston, Watkins native multiclass [19]. These additional classification types utilize a chunking algorithm based on TRON QP solver and work by solving a single quadratic problem involving all the classes [19].

Each package offers a standard set of kernels. In our research several kernels were investigated for each package due to the vital importance of choosing the optimal kernel function Φ for mapping the $\boldsymbol{x}_i$ vector into feature space. The linear, polynomial, radial basis and sigmoid kernels were investigated for e1071. The vanilla, polynomial, laplace, ANOVA, radial basis, and hyperbolic tangent kernels were investigated for kernlab.

Package e1071 allows tuning of the hyper-parameter γ which is used in all kernels except the linear. Kernlab does not provide tuning of the γ parameter directly. Also, the e1071 package implements an interface to the libsvm C++ code [8] for support vector machines.

3. Preprocessing and tuning parameter optimization. Preprocessing data and properly tuning the algorithms is a very important and often overlooked step in performing accurate prediction. One of the key tasks in preprocessing data is to scale it. One of the main advantages of scaling the data is to prevent a feature with a large range of values from dominating a feature with a small range of values [17]. In the case that different measurement scales are involved, scaling the data will also transform all of the features to the same scale

of measurement for a fair comparison between them [29], [22]. It essentially "levels the playing field" so all of the features or input variables are treated equally with a consistent reference of measurement. Furthermore, scaling the data provides the added benefit of avoiding numerical difficulties in calculations [17]. This is key when dealing with kernel techniques as kernels rely on the inner products of the features vectors, as was shown in the *Predictive Learning Methods* section of this paper.

The authors of e1071 and kernlab deemed scaling the data so important that their software scales the input data by default. However, it was discovered that the rda package did not scale automatically and some anomalies in our input data were discovered that effected prediction in a way that sometimes lead to lower errors during cross-validation with the unscaled data. While it is our belief that scaling the data is best, in the principle of authenticity, we report both the non-standardized and the standardized values in our results where applicable.

Understanding the sparsity and unevenness between class labels in a given dataset is also a key component in performing accurate tuning and prediction in classification. Initial investigation of a dataset should identify the proportions in each class label. It is vitally important to then diligently ensure these proportions are properly maintained when performing cross validation and replication. In other words, when dealing with this microarray cancer data, or any HDLSS data, stratification of the sampling is central to obtaining minimal misclassifications. This is highlighted in detail in a later example (Table 2) of the *Stratified sampling* subsection.

Recall from Hadamard [14] that for a problem to be well-posed a solution must not only exist, but it must be unique and stable. In our research, classification with this microarray cancer data proved to be anything but stable and returned multiple minimums when searching for the optimal tuning parameters via cross-validation. This led to the use of various cross-validation implementations in an attempt to determine which one was the best performer for this HDLSS data.

Undeniably, the arduous process of selecting the correct hyper-parameters to optimize accurate classification prediction proved to be one of the most intensive and crucial portions of this research. This level of tuning involves dealing with the ubiquitous bias-variance tradeoff. This is one of the most important concepts in machine learning as it helps explain why there is no universally optimal learning method.

Machine learning algorithms must generalize from the training data rather than just memorize it. Regularization, although it introduces a small amount of

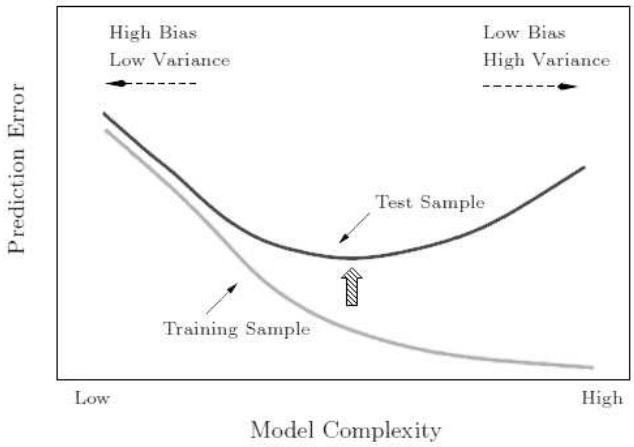

Fig. 7. Bias Variance Tradeoff (Striped arrow indicates the lowest test error) [15], [4]

bias, facilitates the training algorithm's ability to generalize and accurately predict out-of-sample observations. Identifying the optimal hyper-parameters and tuning parameters via cross-validation becomes vital in order to find the optimal bias-variance tradeoff for a particular technique and dataset.

As seen in Figure 7, the training error will decrease as the model becomes more complex and lowers the bias. Conversely, this will result in a proportionally higher amount of variability. Once the test sample, containing data the algorithm has never seen, is introduced the test error shoots up quickly. Thus, the challenge becomes to find the balancing point between bias and variance that correspondingly gives the lowest test error as indicated by the stripped arrow in the diagram.

As mentioned previously, several extensions of cross-validation were explored to determine which machine learning technique would produce the lowest prediction error when executed for classification against microarray cancer data. Leave-one-out cross-validation (LOOCV) and three types of k-fold cross-validation were evaluated.

For classification techniques the goal is to minimize the error over the whole population. This is often referred to as minimizing the generalization error. This is achieved by determining the optimal hyper-parameter(s) for a given method (Figure 3,6). For example, in the simple case of binary classification each observation in the validation case is predicted either correctly or incorrectly. The misclassifications are summarized and the average cross-validation error for each hyper-parameter or hyper-parameter combination is calculated. The hyper-

parameter or combination of hyper-parameters producing the lowest misclassification rate is chosen as optimal. For all cross-validation forms the following zero-one loss function applies:

$$\ell(y_i, f(\boldsymbol{x}_i)) = 1_{\{y_i \neq f(\boldsymbol{x}_i)\}} = \begin{cases} 1 & \text{if } y_i \neq f(\boldsymbol{x}_i) \\ 0 & \text{if } y_i = f(\boldsymbol{x}_i) \end{cases} \tag{25}$$

Leave-One-Out Cross-Validation (LOOCV). The LOOCV form basically performs k-fold $= n$ iterations on a given dataset where one observation is held out as a test case and the remaining observations are used for training. The label of the held-out observation is then predicted using the trained classifier and compared to the actual label. Accuracy of prediction is assessed and the process is repeated for each observation. The mean misclassification rate for a given parameter or set of parameters is then assessed at the end of n iterations.

$$\text{Err}_i \equiv \ell(y_i, \widehat{y}_i) \tag{26}$$

$$\text{Err} = \frac{1}{n}\sum_{i=1}^{n} \text{Err}_i \tag{27}$$

LOOCV Pseudocode

Given a specific parameter or parameter set execute the following

For $i = 1$ to n

- Hold out the i^{th} observation
- Build a classifier using the remaining $n - i^{\text{th}}$ observations for a specific technique
- Give the i^{th} held out $\boldsymbol{x}_i$ variables to the trained classifier to predict the y_i label.
- If $\hat{y}_i \neq y_i$
    - Total Misclassifications = Total Misclassifications + 1
- End If

End Loop

Mean CV error for the parameter(s) = Total Misclassifications/n

K-fold cross-validation methods. The following three k-fold forms involve splitting the data into k equal, or nearly equal, folds and following a similar process as above for cross-validation. These three k-fold methods all perform

cross-validation exactly the same and the difference lies in how the samples are split into folds. When splitting or partitioning the data the mean misclassification rate for a given parameter or set of parameters is assessed at the end of K iterations.

$$\text{Err}_\ell \equiv \frac{1}{n_\ell}\sum_{\ell=1}^{n_\ell} \ell(y_i, \hat{f}^{(-\ell)}(\boldsymbol{x}_i))1_{\{(\boldsymbol{x}_i,\boldsymbol{y}_i)\in Out_\ell\}} \tag{28}$$

$$\text{Err} = \frac{1}{K}\sum_{\ell=1}^{K} \text{Err}_\ell. \tag{29}$$

Traditionally, when choosing the number of k folds to partition the data into for cross-validation the industry standard has been to use 5 or 10 folds [7], [5], [9] with $k = 10$ being the default in most software packages. However, with this microarray cancer data where $\kappa < 1$, the class proportions are often extremely uneven, and there is great sparsity of data, the choice of k becomes important for optimal prediction. In this research $k = 5$ or smaller was found to perform the best. This is explained in more detail in the *Stratified sampling* section of this paper.

K-fold Cross-Validation Pseudocode

For $i = 1$ to K

- Hold out the i^{th} fold of observations
- n_i = number of observations in the i^{th} fold
- Build a classifier for a given technique using the observations in the remaining $K - i^{\text{th}}$ folds
- Give the $\boldsymbol{x}_i$ variables for all observations in the held out i^{th} fold to the trained classifier for prediction of the y_i labels
- Average i^{th} Misclassifications = $\sum(\hat{Y}_i$ label predictions $\neq y_i$ actual labels$)/n_i$
- Total Average Misclassifications = Total Average Misclassification + Average i^{th} misclassifications

End i Loop

Mean CV error for the parameter(s) = Total Misclassifications$/K$

Shuffle and Split (SS). This is the form of cross-validation typically used by default in existing machine learning software in both R and Matlab. In

this process the data is randomized and then simply split into k folds without stratification.

Each fold is held out in turn and predicted by the remaining $k-1$ folds for each unique parameter or set parameters as described above in the K-fold Pseudocode.

SS Pseudocode

- Shuffle the observations into random order
- Split the observations into K fold partitions
- Given a specific parameter or parameter set execute
  - K-fold Cross-Validation Pseudocode

Stratified Cross Validate (SCV) and Balanced Stratified Cross-Validate (BSCV). These methods differ from the SS method above in that both these techniques use stratification when splitting the samples into k folds. This means that these two techniques will maintain the dataset class proportions within each fold when the data is split. This ensures that the probability of each class label occurring in a fold is kept consistent to its probability of occurring in the original dataset. The difference in the two methods is in how they handle uneven sample splits of classes.

The SCV form of cross-validation performs a strictly theoretical stratification which results in folds that are slightly uneven but it maintains pure statistical stratification of classes. The BSCV performs a more computational stratification when splitting the data. BSCV adheres to the statistical stratification on splits for an even multiple of k. It will then gather any remainder from uneven class splits and spread them evenly across all folds at the end of the process. This results in folds of equal or almost equal sizes that are also stratified. These subtleties will become apparent from the example in the *Stratified sampling* subsection (Table 2).

SCV and BSCV Pseudocode

For $j = 1$ to $\#$ of classes

- For the j^{th} class label shuffle the observations into random order for that class
- Split the randomized samples for the j^{th} class label into K fold partitions
- If SCV

- When the observations for the j^{th} class do not split evenly across the folds, the extra observations for the j^{th} class are distributed one at a time from the 1^{st} fold, in sequence until they are exhausted

Else BSCV

- When the observations for the j^{th} class do not split evenly across the folds, the extra observations for the j^{th} class are stored in a matrix

End If

- Concatenate this j^{th} class's K fold splits together with the previous $j^{\text{th}} - 1$ class's K fold class splits such that they accumulate together

End j Loop
If BSCV

- Any remaining observations from the classes that were stored in the matrix are now Shuffled randomly and then distributed one at a time from the 1^{st} fold, in sequence until they are exhausted

End If
Given a specific parameter or set of parameters execute

- K-fold Cross-Validation Pseudocode

Stratified sampling. The importance of stratifying the sampling when partitioning data for cross-validation and replication was discussed previously in other sections. Below is an explanation as to why stratifying the samples is vital to obtaining optimal prediction when classifying data.

Following is a brief explanation of the various cross-validation techniques which will illuminate the need for stratification especially when dealing with very sparse data with uneven class sizes. For this example the Brain Cancer Data is used.

Table 2. Brain label partition splits

Label	1	2	3	4	5
Class Size	10	10	10	4	8
Partition Splits	Partition 1	Partition 2	Partition 3	Partition 4	Partition 5
SS Results (k=5)	31114235	45552224	23153512	34213132	5153312321
SS Results (k=4)	31212212523	12212341115	1314255534	5323533534	
SCV Results (k=4)	111222333455	111222333455	112233455	112233455	
BSCV Results (k=4)	22515411323	41315323152	5423321513	5351212432	

Table 2 aids in illustrating the importance of stratification and careful selection of the k-fold value when dealing with the kind of sparse data that is encountered in this research. Given that the brain class label 4 only contains four observations, if it is split into folds of five or more then not all of the folds will accurately represent the label 4. As seen in the table when executing SS with $k = 5$ partitions, partition 3 and 5 are not representing label 4 at all and partition 2 is inaccurately representing its true proportion. From this we can conclude that for this dataset k-fold must be set to 4.

Continuing in this table we discover the importance of stratifying the sample in conjunction with choosing the proper amount of k-folds. In observing the three k-fold extensions of cross-validation represented in the last three rows it can be seen clearly that only the stratified sampling methods (SCV and BSCV) correctly maintain the class proportions and represent label 4 in each fold or partition.

In the early works of Friedman [11] when he discusses estimation of the prior probabilities he emphasizes the importance of maintaining the correct class proportions so that the π matrix is accurately preserved. Upholding the integrity of the π matrix is a key component to accurate classification.

Comparison of cross-validation performance in this research. Although LOOCV is an unbiased estimator that is extensively used in data mining, when it comes to HDLSS space it often performs sub-optimally. According to Breiman [7], [5], leave-one-out cross-validation is less accurate than leave-many-out. In this research the LOOCV extension had difficulty finding the minimum error and often returned 30, 40, or even over 100 optimal minimum parameters. The 3 k-fold extensions also struggled to find the optimal minimum but usually returned far fewer possible optima than LOOCV. The normal amount returned for the k-fold cross-validates was from 3 to 10 depending on the technique investigated.

Due to the complexity inherent in dealing with this type of massive data, it was discovered that replicating the k-fold extensions 3 to 5 times provided a way of controlling some of the instability and high variability. The replication of the k-fold cross-validations also provided an opportunity to recognize any repeating minimum tuning parameters, thus identifying them as more plausible candidates for the true minimum.

Once a reasonable group of *possible* optimum tuning parameters or parameter sets was determined from all cross-validation extensions, they were put through a replication loop of $R = 500$ to determine the true average test error (TE) across replications. These results would then substantiate the final choice

of the optimal tuning parameter or parameter set. The SCV and BSCV extensions were the most successful at consistently identifying the minimum parameters across classification techniques. Neither the BSCV nor SCV outperformed the other in every scenario. It is not surprising that these two extensions were more successful in determining the optimal tuning parameters as they both perform stratified sampling which is crucial to maintaining the class probabilities and thus performing more accurate classification prediction.

It should also be noted that the LOOCV method is much more computationally intensive and time consuming than the k-fold split techniques. The LOOCV option took anywhere from 3 times up to 8 or 9 times as long to run as 3 replicates of the k-fold splits depending on the classification technique and dataset combination being evaluated. All performance testing was completed on a Unix server with dedicated core and adequate memory.

Hyper-parameter optimization summary. To reiterate, prior to choosing the optimal hyper-parameters and regularization parameters for a given classification technique the data should be scaled, unless done within the technique, and any transformations or other preprocessing steps should be completed. For the actual cross-validation process, due to the high multiplicity inherent in this data, it is recommended to use replicates of either BSCV or SCV in order to consistently obtain the optimal tuning parameters across techniques. If time and resources allow, perform both BSCV and SCV on the given data.

The need for such careful optimization of the tuning parameters stems from the fact that in this type of high-dimensional space, the objective function of the model space is bound to be complex with potentially many local minima and saddle points.

4. Computational experiments. The following tables contain the culmination of our intensive cross-validation analysis to obtain the optimal tuning parameters for each predictive technique across datasets.

Recalling that RDA is a regularization technique with an algorithm to shrink the centers, it is interesting to note that δ, the shrinkage parameter, is zero for three of the datasets: Leukemia, Lung, and Breast(NS). All three of these datasets also use an α regularization parameter of only 0.05. Essentially, for these three datasets, the parameter responsible for shrinking the centers is not used. When looking at the actual prediction error for these datasets in Table 6, it can be seen that the Leukemia and Lung datasets have an average TE of less than 3% and the breast dataset's average TE is less than 8% whether standardized or not. These were the lowest prediction errors for RDA and two

Table 3. RDA Optimal Hyper-Parameters

Regularized Discriminant Analysis (RDA)			
	Class	α	δ
Prostate Cancer	Binomial	0.30	0.20
Colon Cancer (NS)	Binomial	0.10	0.15
Colon Cancer (S)		0.05	0.20
Leukemia Cancer	Binomial	0.05	0
West Breast Cancer	Binomial	0.00	0.55
Lung Cancer	Multiclass	0.05	0
Breast Cancer (NS)	Multiclass	0.05	0
Breast Cancer (S)		0	0.10
Brain Cancer (NS)	Multiclass	0.15	0.10
Brain Cancer (S)		0.05	0.55

of the datasets were actually predicted more accurately by another technique. It could be theorized that when a dataset is relatively easy to classify or separate, then the corresponding penalization applied on the parameter space by RDA regularization and shrinkage algorithms may turn out to be unnecessary.

Another noteworthy observation is the effect that standardizing the data has on the α regularization parameter. In all three datasets that initially resisted standardization it can be observed that the NS version requires a stronger, or larger, regularization parameter than the S version. This would be expected, but it underscores the importance of scaling the data when performing classification.

Following in Table 4 and Table 5 are the optimal SVM tuning parameters for e1071 and kernlab respectively.

Table 4. SVM (e1071) Optimal Hyper-Parameters and Regularization Parameters

Support Vector Machine (e1071)					
	Class	Classification	Kernel	γ	Cost/ν
Prostate Cancer	Binomial	nu-classification	radial	$\gamma = 0.002$	$\nu = 0.5$
Colon Cancer	Binomial	C-classification	sigmoid	$\gamma = 0.0005$	cost = 1
Leukemia Cancer	Binomial	nu-classification	linear		$\nu = 0.5$
West Breast Cancer	Binomial	C-classification	sigmoid	$\gamma = 0.0001402721$	cost = 2
Lung Cancer	Multiclass	nu-classification	sigmoid	$\gamma = 0.001$	$\nu = 0.125$
Breast Cancer	Multiclass	nu-classification	linear		$\nu = 0.0625$
Brain Cancer	Multiclass	C-classification	sigmoid	$\gamma = 0.0001786671$	cost = 2

Table 5. SVM (kernlab) Optimal Regularization Parameters

Support Vector Machine (kernlab)				
	Class	Classification	Kernel	Regularization Parameter (C/ν)
Prostate Cancer	Binomial	C-svc	vanilladot	$C = 0.0008302176$
Colon Cancer	Binomial	C-svc	vanilladot	$C = 0.0006579332$
Leukemia Cancer	Binomial	C-svc	vanilladot	$C = 0.0001629751$
West Breast Cancer	Binomial	C-svc	vanilladot	$C = 0.0002595024$
Lung Cancer	Multiclass	kbb-svc	anovadot	—
Breast Cancer	Multiclass	spoc-svc	vanilladot	—
Brain Cancer	Multiclass	spoc-svc	vanilladot	—

The author of package e1071 emphasizes the importance of choosing the correct kernel parameters for the data and recommends an extensive grid search on a range of parameter values before the results are to be trusted [20]. Stated another way, simply executing this code off-the-shelf and accepting all defaults would not provide optimal results.

It can be seen in Table 4 that there is no single best classification type or kernel for all datasets. The cost and ν values also vary between datasets. The γ value is the only uniformly consistent tuning parameter with all datasets using the default value of $1/p$, where p is the dimensionality or number of features in the dataset.

Of the packages evaluated in this research, e1071 was the simplest to tune because the stability of the results coupled with lower levels of multiplicity made obtaining the optimal hyper-parameter and tuning parameters a far easier and smoother process across all datasets.

For the kernlab package the tuning results are very consistent and standard across the datasets. The C classification type that uses regularization to control the dimensionality is chosen for all binary datasets. The simple linear kernel (vanilladot) is chosen almost exclusively for all datasets in this study. Interestingly, the linear kernel does not use the gamma parameter. Recall that there was no direct method to tune gamma with this package.

Evaluating the predictive results between just the two SVM kernel methods studied in this research, it can be observed that the e1071 had the lowest test error on 4 of 7 datasets. The remaining 3 datasets were better classified by kernlab then e1071 (Table 6).

In summary, no classification technique chose the default settings for all of the tuning parameter values on any dataset and none of them executed optimally off-the-shelf without tuning.

Following in Table 6 is the performance comparison of the classification techniques evaluated in this research. The final evaluation is based on the mean or average test error (TE) across replications. Throughout this section we use R = 1000 replicates of the data partitioned into a 2/3 training dataset and a 1/3 test dataset. Performing replication is believed to help stabilize unstable procedures because the averaged predictors are a more stable sequence with lower predictive loss and less biased predictive estimates [5]. In the interest of reproducibility and unbiased comparison between techniques, the same seed (1398398) is used for all random number generators across classification techniques during comparisons.

The algorithm below summarizes the comparison procedure used in this research.

Test Error (TE) Replication Loop Pseudocode

$R = 1000$
For $r = 1$ to R

- **Using stratification**, randomly split the data into a 2/3 and 1/3 partition
- Build the model with the chosen technique and optimal tuning parameters using the 2/3 partition as training data
- Test the model created by the training data with the held out 1/3 partition
- Store the r^{th} error in a matrix

End Loop

$$\text{Error(TE)} = \frac{1}{R}\sum_{r=1}^{R}\text{error}_r$$

Following are the final results of our comparative analysis of predictive learning classification algorithms on HDLSS microarray cancer data.

From the results, it is evident that there is no one universally optimal classification algorithm or technique. However, regularization is proving to be a strong contender by producing the lowest classification error in 5 of the 7 datasets investigated in this research. Following is a graphical representation of the test error results ordered by increasing dimensionality.

Note that only the optimal performer between non-standardized and standardized data for the three RDA datasets is carried forward in the remainder of our research results.

Table 6. Average Test Error for 1000 Replicates with seed (1398398)

Comparison of classification techniques

	Class	n	p	κ	Dimension	LDA	RDA	SVM (e1071)	KSVM (kernlab)
Prostate Cancer	Binomial	79	500	0.15800	10^{-1}	29.36%	***28.39%***	30.20%	29.90%
Colon Cancer (NS/S)	Binomial	62	2000	0.03100	10^{-2}	13.96%	***11.05%***/13.14%	11.77%	11.85%
Leukemia Cancer	Binomial	72	3571	0.02016	10^{-2}	3.43%	1.99%	1.47%	***1.45%***
West Breast Cancer	Binomial	49	7129	0.00687	10^{-3}	38.90%	***19.99%***	35.99%	37.31%
Lung Cancer	Multiclass	197	1000	0.19700	10^{-1}	3.93%	***2.88%***	3.11%	3.62%
Breast Cancer (NS/S)	Multiclass	97	1213	0.07997	10^{-2}	9.22%	7.15%/7.55%	6.43%	***5.96%***
Brain Cancer (NS/S)	Multiclass	42	5597	0.00750	10^{-3}	31.18%	16.74%/***16.71%***	16.73%	17.07%

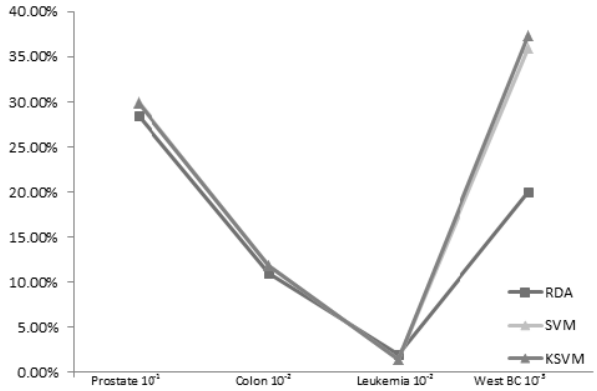

Fig. 8. Raw TE outcome for binary datasets

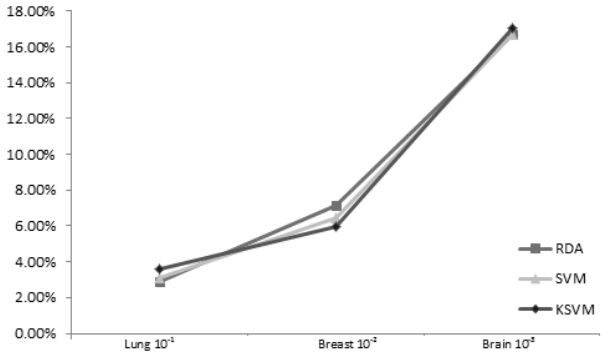

Fig. 9. Raw TE outcome for multiclass datasets

From the average test error graphs it is easier to observe how close these techniques are in raw predictive ability and that RDA was the strongest classifier overall by a narrow margin.

Table 7 details the percent reduction in error from the baseline LDA technique. Note that the two kernel techniques actually had an increase in percent error from the baseline for Prostate Cancer.

From the graphs of percent reduction, the differences in techniques from the LDA baseline give a better representation of the differences in classification techniques.

Following is a comparison of average test error from our research compared to two other publications using some of the same datasets. Our test errors were

Table 7. Percent reduction in Average Test Error from LDA baseline

Percent Reduction in Average Test Error					
	LDA-RDA	LDA-SVM	LDA-kSVM	Dimension	κ
Prostate Cancer	***3.30%***	−2.25%	−1.84%	10^{-1}	0.15800
Colon Cancer	***20.85%***	15.69%	15.11%	10^{-2}	0.03100
Leukemia Cancer	41.98%	57.14%	***57.73%***	10^{-2}	0.02016
West Breast Cancer	***48.61%***	7.48%	4.09%	10^{-3}	0.00687
Lung Cancer	***26.72%***	20.87%	7.89%	10^{-1}	0.19700
Breast Cancer	22.45%	30.26%	***35.36%***	10^{-2}	0.07997
Brain Cancer	***46.41%***	46.34%	45.25%	10^{-3}	0.00750

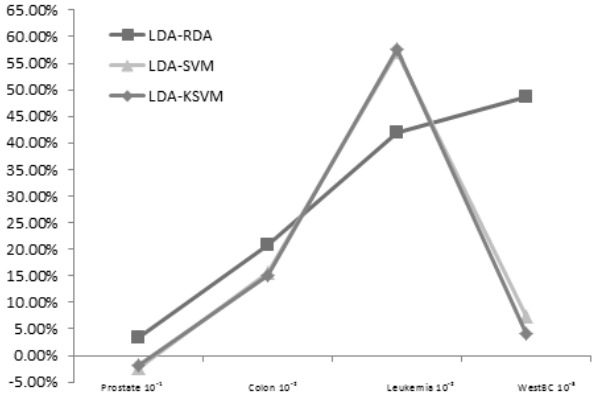

Fig. 10. Multiclass percent reduction in average test error

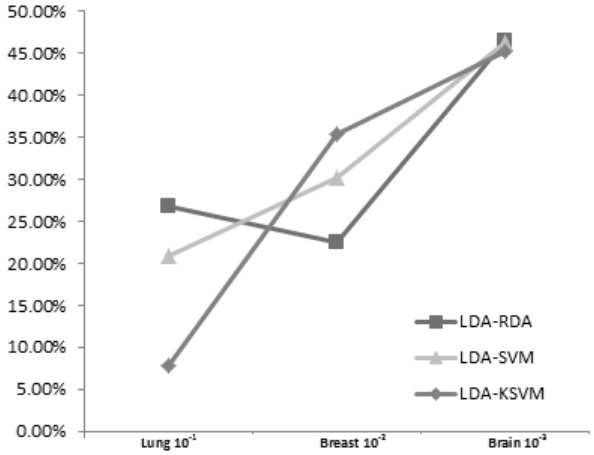

Fig. 11. Binary percent reduction in average test error

Table 8. Comparison of SVM errors across studies.

Support Vector Machine (SVM) Results[1]				
SVM	Bill/Fokoue (e1071)	Bill/Fokoue (kernlab)	Guo/Hastie/ Tibshirani	Dettling
Colon	11.71%	11.85%	$6/22 \sim 27.27\%$	16.67%
Leukemia	1.47%	1.45%	$0/23 \sim 0.00\%^{**}$	3.50%
Brain	16.73%	17.07%	$3/14 \sim 21.43\%$	28.14%

** Note that the Leukemia data in the Guo et al. study used a training to test ratio of 1:1.

Table 9. Comparison of RDA errors across studies

Regularized Discriminant Analysis (RDA) Results[1]		
RDA	Bill/Fokoue	Guo/Hastie/Tibshirani (SCRDA)
Colon	11.05/13.14	$3/22 \sim 13.64\%$
Leukemia	1.99%	$2/23 \sim 8.70\%$
Brain	16.717%	$5/14 \sim 35.71\%$

generated on all of the features with a 2/3 training, 1/3 test split based on 1000 replicates. The test errors from Guo [13] were divided into training and test subsets with a ratio of 2:1 with an unknown number of replicates. Dettling's [10] test errors are using a 2/3, 1/3 split with 50 replicates.

The result of the average test error comparison above strongly supports the importance of properly tuning the regularization parameters and hyper-parameters to facilitate better classification performance. A striking finding of this research is the fact that by carefully optimizing these tuning parameters we consistently outperformed the previous research cited in this paper.

5. Conclusion and discussion. Due to the extreme multiplicity inherent in HDLSS data, a more rigorous and methodical approach was needed when tuning algorithms for classification. Given that any one cross-validation run could bring back one to one-hundred or more parameters as optimally minimum, we discovered that replicating the cross-validation process multiple times was beneficial in enabling us to identify the one optimal parameter or parameter set.

[1]It was our desire to perform a thorough comparison of our results against the results of other authors' research for all seven datasets evaluated in this paper. Unfortunately, the datasets listed in these tables were the only ones we were able to locate research papers on that used the same techniques covered in this paper and reported the average test error or average test accuracy.

Additionally, when predicting with classification algorithms, performing stratified sampling for cross-validation as well as the average test error calculations proved crucial to maintaining class probabilities for optimal algorithm performance.

The regularization technique outperformed both variations of the kernelization algorithms on 5 of the 7 HDLSS microarray cancer datasets. Still, no single universally superior learning method was discovered during this research.

Currently we are continuing our exploration by conducting preliminary investigations into ensemble methods, as well as a fully Bayesian technique, in our quest to find the lowest test errors for classification of HDLSS microarray cancer data.

REFERENCES

[1] Alon U. Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays. *PNAS,* **96** (1999), 6745–6750. http://microarray.princeton.edu/oncology/

[2] Bhattacharjee A. Classification of Human Lung Carcinomas by mRNA Expression Profiling Reveals Distinct Adenocarcinoma Subclasses. *PNAS,* **98** (2001), No 24, 13790–13795. http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi

[3] Breiman L. Random Forests. Berkeley, University of California, USA, 2001. https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

[4] Breiman L. Bias, Variance, and arcing classifiers. Technical Report 460, Statistics, Berkeley, University of California, USA, 1996. https://www.stat.berkeley.edu/~breiman/arcall96.pdf

[5] Breiman L. Heuristics of instability and stabilization in model selection. *The Annals of Statistics,* **24** (1996), No 6, 2350–2383. http://sci2s.ugr.es/keel/pdf/specific/articulo/breiman_heuristics_instability_stabilization1996.pdf

[6] Breiman L. Bagging Predictors. Technical Report 421, Berkeley, University of California, USA, 1994. http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf

[7] Brieman L., P. Spector. Submodel Selection and Evaluation in Regression. The X-Random Case. *International Statistical Review/Revue Internationale de Statistique,* **60** (1992), No 3, 291–319.

http://www.jstor.org/discover/10.2307/1403680?uid=3739832&uid=2&uid=4&uid=3739256&sid=21104917461267

[8] Chang C.-C., C.-J. Lin. LIBSVM: a library for support vector machines (Software). http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001
Detailed documentation found in http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.ps.gz

[9] Clarke B., E. Fokoue, H. Zhang. Principles and Theory for Data Mining and Machine Learning. Springer Verlag, Chapter 1, Chapter 5, New York, 2009.

[10] Dettling M. BagBoosting for Tumor Classification with Gene Expression Data. ETH Zurich, CH-8092 Switzerland, 2004. ftp://ftp.math.ethz.ch/sfs/Manuscripts/dettling/bagboost.pdf

[11] Friedman J. H. Regularized Discriminant Analysis. *Journal of the American Statistical Association,* **84**(1989), No 405, 165–175.
http://www.tandfonline.com/doi/abs/10.1080/01621459.1989.10478752#.VFDjI8np9x0

[12] Golub T. R. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science,* **286** (1999), No 5439, 531–537. http://www.sciencemag.org/content/286/5439/531.short

[13] Guo Y., T. Hastie, R. Tibshirani. Regularized Discriminant Analysis and Its Applications in Microarrays. *Biostatistics*, **1** (2005), No 1, 1–18. http://web.stanford.edu/~hastie/Papers/RDA-6.pdf
Revision: *Biostatistics,* **8** (2006), No 1, 86–100. http://biostatistics.oxfordjournals.org/content/8/1/86.full

[14] Hadamard J. Lectures on Cauchy's Problem in Linear Partial Differential Equations. Yale University Press, New Haven, Connecticut, USA, 1923.

[15] Hastie T., R. Tibshirani, J. Friedman. The Elements of Statistical Learning. Springer Verlag, New York, 2001. http://web.stanford.edu/~hastie/local.ftp/Springer/OLD/ESLII_print4.pdf

[16] Hoshida Y., J. P. Brunet, P. Tamayo, T. R. Golub, J. P. Mesirov. Subclass mapping: Identifying common subtypes in independent disease datasets. *PLoS ONE,* **2** (2007), No 11. http://www.plosone.org/article/info%3A

[17] Hsu C.-W., C.-C. Chang, C.-J. Lin. A Practical Guide to Support Vector Classification. National Taiwan University, Taiwan. http://www.csie.ntu.edu.tw/~cjlin, 2010

[18] Karatzoglou A., D. Meyer, K. Hornik. Support Vector Machines in R. *Journal of Statistical Software,* **15** (2006), No 9, 1–28. http://www.jstatsoft.org/v15/i09/

[19] Karatzoglou A., A. Smola, K. Hornik, A. Zeileis. kernlab – An S4 Package for Kernel Methods in R. *Journal of Statistical Software,* **11** (2004), No 9, 1–20. http://www.jstatsoft.org/v11/i09/

[20] Meyer D. Support Vector Machines. The Interface to libsvm in package e1071, FH Technikum Wien, Austria, 2014. http://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf

[21] Pomeroy S. Prediction of Central Nervous System Embryonal Tumor Outcome Based on Gene Expression. *Nature,* **415** (2002), 436–442. http://www.broad.mit.edu/mpr/CNS/

[22] Ripley B. D. Pattern Recognition and Neural Networks. Cambridge University Press, 1995.

[23] Schapire R. The Strength of Weak Learnability. *Machine Learning*, **5** (1990), 197–227. http://machine-learning.martinsewell.com/ensembles/boosting/Schapire1990.pdf

[24] Schölkopf B., A. Smola. Learning with Kernels. MIT Press, USA, 2002.

[25] Siegel R., J. Ma, Z. Zou, A. Jemal. Cancer statistics. *A Cancer Journal for Clinicians*, **64** (2014), 9–29. doi: 10.3322/caac.21208

[26] Singh D. Gene Expression Correlates of Clinical Prostate Cancer Behavior. *Cancer Cell,* **1** (2002), No 2, 203–209. http://www.sciencedirect.com/science/article/pii/S1535610802000302

[27] Tibshirani R., T. Hastie, B. Narashimhan, G. Chu. Class prediction by nearest shrunken centroids with applications to DNA microarrays. *Statistical Science,* **18** (2003), 104–117. http://www.jstor.org/discover/10.2307/3182873?uid=3739832&uid=2&uid=4&uid=3739256&sid=21104917461267

[28] Tikhonov A. Solution of incorrectly formulated problems and the regularization method. *Dokl. Akad. Nauk SSSR*, **151** (1963), 501–504 (in Russian); English translation in: *Sov. Math., Dokl.* **5** (1963), 1035–1038.

[29] Vapnik V. The Nature of Statistical Learning Theory. Springer, New York, 1995.

[30] Venables W. N., B. D. Ripley. Modern Applied Statistics with S. (Fourth ed.) Springer, 2002. http://cran.r-project.org/web/packages/MASS/MASS.pdf

[31] West M., C. Blanchette, H. Dressman, C. Huang, S. Ishida, R. Spang, H. Zuzan, J. Olson, J. Marks, J. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. In: Proceedings of the National Academy of Sciences, 98, 11462–11467. http://www.pnas.org/content/98/20/11462.full#abstract-1, 2001
Dataset taken from: mboost R package of Hofner, Mayr, Robinzonov, & Schmid, 2014

Jo Bill
Rochester Institute of Technology
Center for Quality and Applied Statistics
98, Lomb Memorial Drive, Rochester
New York, USA
e-mail: jab1470@rit.edu

Ernest Fokoué
Rochester Institute of Technology
School of Mathematical Sciences
98 Lomb Memorial Drive, Rochester
New York, USA
e-mail: epfeqa@rit.edu

Received October 27, 2014
Final Accepted December 18, 2014