High-Dimensional Data Classification

Vijay Pappu and Panos M. Pardalos

We dedicate this paper to the 70th birthday of our colleague and friend Dr. Boris Mirkin

Abstract Recently, high-dimensional classification problems have been ubiquitous due to significant advances in technology. High dimensionality poses significant statistical challenges and renders many traditional classification algorithms impractical to use. In this chapter, we present a comprehensive overview of different classifiers that have been highly successful in handling high-dimensional data classification problems. We start with popular methods such as Support Vector Machines and variants of discriminant functions and discuss in detail their applications and modifications to several problems in high-dimensional settings. We also examine regularization techniques and their integration to several existing algorithms. We then discuss more recent methods, namely the hybrid classifiers and the ensemble classifiers. Feature selection techniques, as a part of hybrid classifiers, are introduced and their relative merits and drawbacks are examined. Lastly, we describe AdaBoost and Random Forests in the ensemble classifiers and discuss their recent surge as useful algorithms for solving high-dimensional data problems.

Keywords High dimensional data classification • Ensemble methods • Feature selection • Curse of dimensionality • Regularization

V. Pappu (✉) • P.M. Pardalos
Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611, USA
e-mail: vijay.s.pappu@aexp.com

F. Aleskerov et al. (eds.), *Clusters, Orders, and Trees: Methods and Applications: In Honor of Boris Mirkin's 70th Birthday*, Springer Optimization and Its Applications 92,
DOI 10.1007/978-1-4939-0742-7_8,

1 Introduction

In the past decade, technological advances have had a profound impact on society and the research community [45]. Massive amounts of high-throughput data can be collected simultaneously and at relatively low cost. Often, each observation is characterized with thousands of variables/features. For example, in biomedical studies, huge numbers of magnetic resonance images (MRI) and functional MRI data are collected for each subject [66]. The data collected from gene expression microarrays consist of thousands of genes that constitute features [17]. Various kinds of spectral measurements including Mass Spectroscopy and Raman Spectroscopy are very common in chemometrics, where the spectra are recorded in channels that number well into the thousands [30, 80]. Satellite imagery has been used in natural resource discovery and agriculture, collecting thousands of high-resolution images. Examples of these kinds are plentiful in computational biology, climatology, geology, neurology, health science, economics, and finance among others. In several applications, the measurements tend to be very expensive and hence the number of samples in many datasets are on the order of tens, or maybe low hundreds. These datasets, often called the high-dimension low-sample size (HDLSS) datasets, are characterized with a large number of features p and a relatively small number of samples n; with $p >> n$ [98]. These massive collections of data along with many new scientific problems create golden opportunities and significant challenges for the development of mathematical sciences.

Classification is a supervised machine learning technique that maps some combination of input variables, which are measured or preset, into predefined classes. Classification problems occur in several fields of science and technology like discriminating cancerous cells from non-cancerous cells, web document classification, categorizing images in remote sensing applications among many others. Several algorithms starting from Neural Networks [44], Logistic Regression [57], linear discriminant analysis (LDA) [64], support vector machines (SVM) [92] and more recently ensemble methods like Boosting [33] and Random Forests [8], have been proposed to solve the classification problem in different contexts. However, the availability of massive data along with new scientific problems arising in the fields of computational biology, microarray gene expression analysis, etc., have reshaped statistical thinking and data analysis. The high-dimensional data has posed significant challenges to standard statistical methods and have rendered many existing classification techniques impractical [53]. Hence, researchers have proposed several novel techniques to handle the inherent difficulties of high-dimensional spaces that are discussed below.

1.1 Statistical Challenges of High-Dimensional Data Spaces

1.1.1 Curse of Dimensionality

The accuracy of classification algorithms tends to deteriorate in high dimensions due to a phenomenon called the *curse of dimensionality* [27, 60]. This phenomenon is illustrated by Trunk [90] using an example in [90]. Trunk found that (1) the best test error was achieved using a finite number of features; (2) using an infinite number of features, test error degrades to the accuracy of random guessing; and (3) the optimal dimensionality increases with increasing sample size. Also, a naive learning technique (dividing the attribute space into cells and associating a class label with each cell) that predicts using a majority voting scheme requires the number of training samples to be an exponential function of the feature dimension [50]. Thus, the ability of an algorithm to converge to a true model deteriorates rapidly as the feature dimensionality increases.

1.1.2 Poor Generalization Ability

A further challenge for modeling in high-dimensional spaces is to avoid overfitting the training data [17]. It is important to build a classification model with good generalization ability. It is expected that such a model, in addition to performing well on the training set, would also perform equally well on an independent testing set. However, often the small number of samples in high-dimensional data settings cause the classification model to overfit to the training data, thereby having poor generalization ability for the model. Two of the more common approaches to addressing these challenges of high-dimensional spaces are reducing the dimensionality of the dataset or applying methods that are independent of data dimensionality. We discuss several classifiers pertaining to these two approaches in subsequent sections.

In this survey, we present several state-of-the-art classifiers that have been very successful for classification tasks in high-dimensional data settings. The remainder of the chapter is organized as follows. Section 2 talks about SVM and its variants. Discriminant functions and their modifications including regularized techniques are discussed in Sect. 3. Section 4 discusses hybrid classifiers that include several feature selection techniques combined with other traditional classification algorithms. Recent developments in ensemble methods and their applications to high-dimensional data problems are discussed in Sect. 5. Some software packages implementing the methods in different programming languages are discussed in Sect. 6. Concluding remarks are presented in Sect. 7.

2 Support Vector Machines

2.1 Hard-Margin Support Vector Machines

In the last decade, SVM [92] have attracted the attention of many researchers with successful application to several classification problems in bioinformatics, finance and remote sensing among many others [13, 69, 89]. Standard SVM construct a hyperplane, also known as *decision boundary*, that *best* divides the input space χ into two disjoint regions. The hyperplane $f : \chi \rightarrow \Re$, is estimated from the training set S. The class membership for an unknown sample $\boldsymbol{x} \in \chi$ can be based on the classification function $g(\boldsymbol{x})$ defined as:

$$g(\boldsymbol{x}) = \begin{cases} -1, & f(\boldsymbol{x}) < 0 \\ 1, & f(\boldsymbol{x}) > 0 \end{cases} \tag{1}$$

Consider a binary classification problem with the training set S defined as:

$$S = \{(\boldsymbol{x}_i, y_i) | \boldsymbol{x}_i \in \Re^p, y_i \in \{-1, 1\}\}, \quad i = 1, 2, \ldots, n \tag{2}$$

where y_i is either -1 or 1 depending on the class that each $\boldsymbol{x}_i$ belongs to. Assume that the two classes are linearly separable and hence there exists atleast one hyperplane that separates the training data correctly. A hyperplane parameterized by the normal vector $\boldsymbol{w} \in \Re^p$ and bias $b \in \Re$ is defined as:

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle - b = 0 \tag{3}$$

where the inner product $\langle \cdot, \cdot \rangle$ is defined on $\Re^p \times \Re^p \rightarrow \Re$. The training set S satisfies the following linear inequality with respect to the hyperplane:

$$y_i (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - b) \geq 1 \quad \forall i = 1, 2, \ldots, n \tag{4}$$

where the parameters $\boldsymbol{w}$ and b are chosen such that the distance between the hyperplane and the closest point is maximized. This geometrical margin can be expressed by the quantity $\frac{1}{||\boldsymbol{w}||}$. Hence, for linearly separable set of training points, SVM can be formulated a linearly constrained quadratic convex optimization problem given as:

$$\begin{aligned} &\underset{\boldsymbol{w}, b}{\text{minimize}} \quad ||\boldsymbol{w}||_2^2 \\ &\text{subject to} \quad y_i (\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - b) \geq 1 \quad \forall i = 1, 2, \ldots, n \end{aligned} \tag{5}$$

This classical convex optimization problem can be rewritten (using the Lagrangian formulation [5]) into the following dual problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\Re^n}{\text{maximize}} \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle) \\ \text{subject to} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0, \quad \text{and}, \quad \alpha_i \geq 0, \quad i = 1, 2, \ldots, n \end{aligned} \tag{6}$$

where the Lagrange multipliers α_i $(i = 1, 2, \ldots, n)$ expressed in (6) can be estimated using quadratic programming (QP) methods [22]. The optimal hyperplane f can then be estimated using the Lagrange multipliers obtained from solving (6) and the training samples, i.e.,

$$f(\boldsymbol{x}) = \sum_{i\in S'}\alpha_i y_i(\langle \boldsymbol{x}\cdot\boldsymbol{x}_i\rangle) - b \tag{7}$$

where S' is the subset of training samples called *support vectors* that correspond to non-zero Lagrange multipliers α_i. Support vectors include the training points that exactly satisfy the inequality in (5) and lie at a distance equal to $\frac{1}{\|\boldsymbol{w}\|}$ from the optimal separating hyperplane. Since the Lagrange multipliers are non-zero only for the *support vectors* and zero for other training samples, the optimal hyperplane in (7) effectively consists of contributions from the *support vectors*. It is also important to note that the Lagrange multipliers α_i qualitatively provide relative weight of each *support vector* in determining the optimal hyperplane.

The convex optimization problem in (5) and the corresponding dual in (6) converge to a global solution *only* if the training set is linearly separable. These SVM are called *hard-margin support vector machines*.

2.2 Soft-Margin Support Vector Machines

The *maximum-margin* objective introduced in the previous subsection to obtain the *optimal* hyperplane is susceptible to the presence of outliers. Also, it is often difficult to adhere to the assumption of linear separability in real-world datasets. Hence, in order to handle nonlinearly separable datasets as well as be less sensitive to outliers, *soft-margin support vector machines* are proposed. The objective cost function in (5) is modified to represent two competing measures namely, *margin maximization* (as in the case of linearly separable data) and *error minimization* (to penalize the wrongly classified samples). The new cost function is defined as:

$$\Psi(\boldsymbol{w}, \boldsymbol{\xi}) = \frac{1}{2}||\boldsymbol{w}||_2^2 + C\sum_{i=1}^{n}\xi_i \tag{8}$$

where $\boldsymbol{\xi}$ is the *slack variable* introduced to account for the non-separability of data, and the constant C represents a regularization parameter that controls the penalty assigned to errors. The larger the C value, the higher the penalty associated to misclassified samples. The minimization of the cost function expressed in (8) is subject to the following constraints:

$$\begin{aligned} y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - b) &\geq 1 - \xi_i, \quad \forall i = 1, 2, \ldots, n \\ \xi_i &\geq 0, \quad \forall i = 1, 2, \ldots, n \end{aligned} \tag{9}$$

The convex optimization problem can then be formulated using (8) and (9) for the nonlinearly separable data as:

$$\begin{aligned} &\underset{\boldsymbol{w}, b, \boldsymbol{\xi}}{\text{minimize}} \quad \frac{1}{2}||\boldsymbol{w}||_2^2 + C \sum_{i=1}^{n} \xi_i \\ &\text{subject to} \quad y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1, 2, \ldots, n \end{aligned} \tag{10}$$

The optimization problem in (10) accounts for the outliers by adding a penalty term $C\xi_i$ for each outlier to the objective function. The corresponding dual to (10) can be written using the Lagrange formulation as:

$$\begin{aligned} &\underset{\boldsymbol{\alpha} \in \Re^n}{\text{maximize}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle) \\ &\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \text{and,} \quad 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, n \end{aligned} \tag{11}$$

The quadratic optimization problem in (11) can be solved using standard QP techniques [22] to obtain the Lagrange multipliers α_i.

2.3 Kernel Support Vector Machines

The idea of linear separation between two classes mentioned in the subsections above can be naturally extended to handle nonlinear separation as well. This is achieved by mapping the data through a particular nonlinear transformation into a higher dimensional feature space. Assuming that the data is linearly separable in this high dimensional space, a linear separation, similar to earlier subsections, can be found. Such a hyperplane can be achieved by solving a similar dual problem defined in (11) by replacing the inner products in the original space with inner products in the transformed space. However, an explicit transformation from the original space to feature space could be expensive and at times infeasible as well. The kernel method [12] provides an elegant way of dealing with such transformations.

Consider a kernel function $K(\cdot,\cdot)$, satisfying *Mercer's* theorem, that equals an inner product in the transformed higher dimensional feature space [65], i.e.,

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle \tag{12}$$

where $\Phi(\boldsymbol{x}_i)$ and $\Phi(\boldsymbol{x}_j)$ correspond to the mapping of data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ from the original space to the feature space. There are several kernel functions defined in literature that satisfy *Mercer's* conditions. One such kernel, called the Gaussian kernel is given by:

$$K(\boldsymbol{x}_i, \boldsymbol{x}) = \exp(-\sigma ||\boldsymbol{x}_i - \boldsymbol{x}||^2) \tag{13}$$

where σ is a parameter inversely proportional to the width of the Gaussian radial basis function. Another extensively studied kernel is the polynomial function of order p expressed as

$$K(\boldsymbol{x}_i, \boldsymbol{x}) = (\langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + 1)^p \tag{14}$$

Such kernel functions defined above allow for efficient estimation of inner products in feature spaces without the explicit functional form of the mapping Φ. This elegant calculation of inner products in higher dimensional feature spaces, also called the *kernel trick*, considerably simplifies the solution to the dual problem. The inner products between the training samples in the dual formulation (11) can be replaced with a kernel function K and rewritten as:

$$\begin{aligned} \underset{\boldsymbol{\alpha} \in \Re^n}{\text{maximize}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \text{and,} \quad 0 \le \alpha_i \le C, \quad i = 1, 2, \ldots, n \end{aligned} \tag{15}$$

The *optimal* hyperplane f obtained in the higher dimensional feature space can be conveniently expressed as a function of data in the original input space as:

$$f(\boldsymbol{x}) = \sum_{i \in S'} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) - b \tag{16}$$

where S' is a subset of training samples with non-zero Lagrange multipliers α_i. The shape of $f(\boldsymbol{x})$ depends on the type of kernel functions adopted.

It is important to note that the performance of kernel-based SVM is dependent on the optimal selection of multiple parameters, including the kernel parameters (e.g., σ and p parameters for the Gaussian and polynomial kernels, respectively) and the regularization parameter C. A simple and successful technique that has been employed involves a grid search over a wide range of the parameters.

The classification accuracy of SVM for every pair of parameters is estimated using a leave-one-out cross-validation technique and the pair corresponding to the highest accuracy is chosen. Also, some interesting automatic techniques have been developed to estimate these parameters [15, 16]. They involve constructing an optimization problem that would maximize the margin as well as minimize the estimate of the expected generalization error. Optimization of the parameters is then carried out using a gradient descent search over the space of the parameters. Recently, more heuristic-based approaches have been proposed to deal with this issue. A continuous version of Simulated Annealing (SA) called *Hide and Seek SA* was employed in [61] to estimate multiple parameters as well as select a subset of features to improve the classification accuracy. Similar approaches combining particle swarm optimization (PSO) with SVM are proposed in [39,62]. Furthermore, a modified Genetic Algorithm (GA) was also implemented along with SVM to estimate the optimal parameters [47].

2.4 SVM Applied to High-Dimensional Classification Problems

Support vector machines have been successfully applied to high-dimensional classification problems arising in fields like remote sensing, web document classification, microarray analysis etc. As mentioned earlier, conventional classifiers like logistic regression, maximum likelihood classification etc., on high-dimensional data tend to overfit the model using training data and run the risk of achieving lower accuracies on testing data. Hence, a pre-processing step like either feature selection and/or dimensionality reduction techniques are proposed to alleviate the problem of *curse of dimensionality* while working with these traditional classifiers. Surprisingly, SVM have been successfully applied to hyperspectral remote sensing images without any pre-processing steps [69]. Researchers show that SVM are more effective that the traditional pattern recognition approach that involves a feature selection procedure followed by a conventional classifier and are also insensitive to Hughes phenomena [49]. This is particularly helpful as it avoids the unnecessary additional computation of an intermediary step like feature selection/dimensionality reduction to achieve high classification accuracy.

Similar observations were reported in the field of document classification in [52], where SVM were trained directly on the original high-dimensional input space. Kernel SVM (Gaussian and polynomial kernels) were employed and compared with other conventional classifiers like k-NN classifiers, Naive-Bayes Classifier, Rocchio Classifier and C4.5 Decision Tree Classifier. The results show that Kernel SVM outperform the traditional classifiers. Also, in the field of microarray gene expression analysis, SVM have been successfully applied to perform classification of several cancer diagnosis tasks [9, 74].

The insensitivity of SVM to overfitting and the ability to overcome the curse of dimensionality can be explained via the generalization error bounds developed by Vapnik et al. [93]. Vapnik showed the following generalization error bounds for Large Margin Classifiers:

$$\epsilon = \tilde{O}\left(\frac{1}{m}\left(\frac{R^2}{\gamma^2} + \log\frac{1}{\delta}\right)\right) \tag{17}$$

where m is the number of training samples, γ is the margin between the parallel planes, and $(R, \delta) \in \Re^+$ with $0 < \delta \leq 1$. This error bound is inversely dependent on the sample size m and the margin γ. For a finite sample size, maximizing the margin γ (or minimizing the weight vector) would reduce the generalization error ϵ. Interestingly, this error bound does *not* depend on the dimensionality of the input space. Since, it is highly likely to linearly separate the data in higher dimensions, SVM tend to perform well with classification tasks in high dimensions.

3 Discriminant Functions

A discriminant function $g : \Re^p \rightarrow \{-1, 1\}$ assigns either class 1 or class 2 to an input vector $\boldsymbol{x} \in \Re^p$. We consider here a class of discriminant functions $\mathcal{G}$ that are well studied in literature and traditionally applied to binary classification problems.

3.1 Quadratic and Linear Discriminant Analysis

Consider a binary classification problem with classes $\mathcal{C}_1$ and $\mathcal{C}_2$ and prior probabilities given as π_1 and π_2. Assume the class conditional probability densities $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ to be normally distributed with mean vectors $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ and covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, respectively:

$$f_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}_k|^{1/2}}\exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_k)\right) \quad k = 1, 2. \tag{18}$$

where, $|\boldsymbol{\Sigma}_k|$ is the determinant of the covariance matrix $\boldsymbol{\Sigma}_k$. Following *Bayes* optimal rule [3], *quadratic discriminant analysis (QDA)* [64] assigns class 1 to an input vector $\boldsymbol{x}$ if the following condition holds:

$$\pi_1 f_1(\boldsymbol{x}) \geq \pi_2 f_2(\boldsymbol{x}) \tag{19}$$

Linear discriminant analysis [64] further assumes the covariances $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are equal to $\boldsymbol{\Sigma}$ and classifies an input vector again in accordance to *Bayes* optimal rule. The condition in (19) can then be rewritten as:

$$\log\frac{\pi_1}{\pi_2} + (\boldsymbol{x} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \geq 0, \quad \boldsymbol{\mu} = \frac{1}{2}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2). \tag{20}$$

Assuming the prior probabilities to be equal, (20) is equivalent to:

$$(\boldsymbol{x} - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_1) \leq (\boldsymbol{x} - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_2) \tag{21}$$

It is interesting to note that LDA compares the *squared Mahalanobis distance* [21] of $\boldsymbol{x}$ from the class means $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ and assigns the class that is closest. The squared Mahalanobis distance of a point $\boldsymbol{x}$ from a distribution $\mathcal{P}$ characterized by mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is defined as:

$$d_M(\boldsymbol{x}, \mathcal{P}) = (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \tag{22}$$

This distance measure, unlike *Euclidean distance measure*, accounts for correlations among different dimensions of $\boldsymbol{x}$. Equation (21) shows how LDA differs from other *distance-based* classifiers like *k-NN* classifier [3] which measures Euclidean distance to assign the class.

3.2 Fisher Linear Discriminant Analysis

Fisher linear discriminant analysis (FLDA) [3], unlike LDA, does not make assumptions on the class conditional densities. Instead, it estimates the class means from the training set. In practice, the most commonly used estimators are their maximum-likelihood estimates, given by:

$$\hat{\boldsymbol{\mu}}_1 = \frac{1}{N_1} \sum_{k \in \mathcal{C}_1} \boldsymbol{x}_k, \qquad \hat{\boldsymbol{\mu}}_2 = \frac{1}{N_2} \sum_{k \in \mathcal{C}_2} \boldsymbol{x}_k. \tag{23}$$

Fisher linear discriminant analysis attempts to find a projection vector $\boldsymbol{w}$ that maximizes the class separation. In particular, it maximizes the following *Fisher criterion* given as:

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}} \tag{24}$$

where $\boldsymbol{S}_B$ is the *between-class* covariance matrix and is given by:

$$\boldsymbol{S}_B = (\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1)(\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1)^T \tag{25}$$

and $\boldsymbol{S}_W$ is the *within-class* covariance matrix and is given by:

$$\boldsymbol{S}_W = \sum_{k \in \mathcal{C}_1} (\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_1)(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_1)^T + \sum_{k \in \mathcal{C}_2} (\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_2)(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}_2)^T \tag{26}$$

The optimal Fisher discriminant $\boldsymbol{w}^*$ can be obtained by maximizing the *Fisher criterion*:

$$\underset{\boldsymbol{w}}{\text{maximize}} \quad J(\boldsymbol{w}) \tag{27}$$

An important property to notice about the objective function $J(\boldsymbol{w})$ is that it is invariant to the rescalings of the vector $\boldsymbol{w} \rightarrow \alpha \boldsymbol{w}, \forall \alpha \in \Re$. Hence, $\boldsymbol{w}$ can be chosen in a way that the denominator is simply $\boldsymbol{w}^T S_W \boldsymbol{w} = 1$, since it is a scalar itself. For this reason, we can transform the problem of maximizing *Fisher criterion J* into the following constrained optimization problem,

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{maximize}} \quad & \boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w} \\ \text{subject to} \quad & \boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w} = 1 \end{aligned} \tag{28}$$

The KKT conditions for (28) can be solved to obtain the following generalized eigenvalue problem, given as:

$$\boldsymbol{S}_B \boldsymbol{w} = \lambda \boldsymbol{S}_W \boldsymbol{w} \tag{29}$$

where λ represents the eigenvalue and the optimal vector $\boldsymbol{w}^*$ corresponds to the eigenvector with the largest eigenvalue $\lambda_{\max}$ and is proportional to:

$$\boldsymbol{w}^* \propto \boldsymbol{S}_W^{-1} (\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1) \tag{30}$$

The class of an input vector $\boldsymbol{x}$ is determined using the following condition:

$$\langle \boldsymbol{w}^*, \boldsymbol{x} \rangle < c \tag{31}$$

where $c \in \Re$ is a threshold constant.

3.3 Diagonal Linear Discriminant Analysis

Diagonal linear discriminant analysis (DLDA) extends on LDA and assumes independence among the features [35]. In particular, the discriminant rule in (20) is replaced with:

$$\log \frac{\pi_1}{\pi_2} + (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{D}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \geq 0 \tag{32}$$

where $\boldsymbol{D} = \text{diag}(\boldsymbol{\Sigma})$. The off-diagonal elements of the covariance matrix $\boldsymbol{\Sigma}$ are replaced with zeros by independence assumption.

Similarly, *diagonal quadratic discriminant analysis* (DQDA) [28] assumes the *independence rule* for QDA. The discriminant rule in this case is given by:

$$\log \frac{\pi_1}{\pi_2} + (\boldsymbol{x} - \boldsymbol{\mu}_2)^T \boldsymbol{D}_2^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_2) - (\boldsymbol{x} - \boldsymbol{\mu}_1)^T \boldsymbol{D}_1^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_1) \geq 0 \tag{33}$$

where $\boldsymbol{D}_1 = \text{diag}(\boldsymbol{\Sigma}_1)$, and $\boldsymbol{D}_2 = \text{diag}(\boldsymbol{\Sigma}_2)$.

Diagonal quadratic discriminant analysis and DLDA classifiers are sometimes called "naive Bayes" classifiers because they can arise in a Bayesian setting [2]. Additionally, it is important to note that FLDA and Diagonal Discriminant analysis (DLDA and DQDA) are commonly generalized to handle multi-class problems as well.

3.4 Sparse Discriminant Analysis

The optimal discriminant vector in FLDA (30) involves estimating the inverse of covariance matrix obtained from sample data. However the high dimensionality in some classification problems poses the threat of singularity and thus leads to poor classification performance. One approach to overcome singularity involves a variable selection procedure that selects a subset of variables most appropriate for classification. Such a *sparse* solution has several advantages including better classification accuracy as well as interpretability of the model. One of the ways to induce sparsity is via the path of regularization. Regularization techniques have been traditionally used to prevent overfitting in classification models, but recently, they have been extended to induce sparsity as well in high-dimensional classification problems. Here, we briefly discuss some standard regularization techniques that facilitate variable selection and prevent overfitting.

Given a set of instance-label pairs $(\boldsymbol{x}_i, y_i)$; $i = 1, 2, \ldots, n$; a regularized classifier optimizes the following unconstrained optimization problem:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad \Phi(\boldsymbol{x}, y, \boldsymbol{\beta}) + \lambda ||\boldsymbol{\beta}||_p \tag{34}$$

where Φ represents a non-negative loss function, $(p, \lambda) \in \Re$ and $\boldsymbol{\beta}$ is the coefficient vector. Classifiers with $p = 1$ (*Lasso-penalty*) and $p = 2$ (*ridge-penalty*) have been successfully applied to several classification problems [99].

In a regression setting, Tibshirani [85] introduced variable selection via the framework of regularized classifiers using the l_1-norm. This method, also called *least absolute shrinkage and selection operator (LASSO)*, considers the least-squares error as the loss function. The user-defined parameter λ balances the regularization and the loss terms. The l_1-norm in Lasso produces some coefficients that are exactly 0 thus facilitating the selection of only a subset of variables useful for regression. The Lasso regression, in addition to providing a sparse model, also shares the stability of ridge regression. Several algorithms have been successfully

employed to solve the Lasso regression in the past decade. Efron et al. [29] showed that, starting from zero, the Lasso solution paths grow piecewise linearly in a predictable way and hence exploit this predictability to propose a new algorithm called *Least Angle Regression* that solves the entire Lasso path efficiently. The Lasso framework has been further extended to several classification problems by considering different loss functions, and has been highly successful in producing sparse models with high classification accuracy.

A Lasso-type framework, however, is not without its limitations. Zou and Hastie [99] mention that a Lasso framework, in high-dimensional problems, suffers from two drawbacks namely, the number of variables selected is limited by the number of samples n, and in the case of highly correlated features, the method selects one of them, neglecting the rest and also does not care about the one selected. The second limitation, also called the *grouping effect*, is very common in high-dimensional classification problems like microarray gene analysis where a group of variables are highly correlated to each other. The authors propose a new technique that overcomes the limitations of Lasso. The technique, called *elastic-net*, considers a convex combination of l_1 and l_2-norms to induce sparsity. In particular, in an *elastic-net* framework, the following optimization problem is minimized:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad \Phi(\boldsymbol{x}, y, \boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_1 + (1-\lambda)||\boldsymbol{\beta}||_2 \tag{35}$$

where Φ is the loss function, and $0 \leq \lambda \leq 1$. When $\lambda = 0$ (or $= 1$), the elastic-net framework simplifies to Lasso (or ridge) frameworks. The method could simultaneously perform variable selection along with continuous shrinkage and also select groups of correlated variables. An efficient algorithm, called LARS-EN, along the lines of LARS, was proposed to solve the elastic-net problem. It is important to note that these regularized frameworks are very general and can be added to models that suffer from overfitting. They provide better generalization performance by inherently performing variable selection and thus also producing better interpretable models.

Sparsity can be induced to the solution of FLDA using regularization techniques described above. One such method called *sparse linear discriminant analysis (SLDA)*, is inspired from *penalized least squares* where regularization is applied to the solution of least squares problem via *Lasso-penalty*. The penalized least squares problem is formulated as:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1 \tag{36}$$

where $\boldsymbol{X}$ represents the data matrix and $\boldsymbol{y}$ is the outcome vector. The second term in (36) is assumed to induce sparsity to the optimal $\boldsymbol{\beta}$.

In order to induce sparsity in FLDA via the l_1 penalty, the generalized eigenvalue problem in (29) is first reformulated as an equivalent least squares regression problem and is shown that the optimal discriminant vector of FLDA is equivalent to

the optimal regression coefficient vector. This is achieved by applying the following theorem:

Theorem. *Assume the between-class covariance matrix* $\boldsymbol{S}_B \in \Re^{p\times p}$ *and the within-class covariance matrix* $\boldsymbol{S}_W \in \Re^{p\times p}$ *be given by* (25) *and* (26). *Also, assume* $\boldsymbol{S}_W$ *is positive definite and denote its Cholesky decomposition as* $\boldsymbol{S}_W = \boldsymbol{R}_W^T \boldsymbol{R}_W$ *where* $\boldsymbol{R}_W \in \Re^{p\times p}$ *is an upper triangular matrix. Let* $\boldsymbol{H}_B \in \Re^{n\times p}$ *satisfy* $\boldsymbol{S}_B = \boldsymbol{H}_B^T \boldsymbol{H}_B$. *Let* $\boldsymbol{v}_1, \boldsymbol{v}_2 \ldots, \boldsymbol{v}_q (q \leq min(p, n-1))$ *denote the eigenvectors of problem* (29) *corresponding to the* q *largest eigenvalues* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_q$. *Let* $\boldsymbol{A} \in \Re^{p\times q} = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_q]$ *and* $\boldsymbol{B} \in \Re^{p\times q} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_q]$. *For* $\lambda > 0$, *let* $\hat{\boldsymbol{A}}$ *and* $\hat{\boldsymbol{B}}$ *be the solution to the following least squares regression problem:*

$$\begin{aligned} &\underset{\boldsymbol{A},\boldsymbol{B}}{\text{minimize}} \ \sum_{i=1}^{n} \| \boldsymbol{R}_W^{-T} \boldsymbol{H}_{B,i} - \boldsymbol{A}\boldsymbol{B}^T \boldsymbol{H}_{B,i} \|^2 + \sum_{j=1}^{q} \boldsymbol{\beta}_j^T \boldsymbol{S}_W \boldsymbol{\beta}_j, \\ &\text{subject to} \ \ \boldsymbol{A}^T \boldsymbol{A} = I \end{aligned} \tag{37}$$

where, $\boldsymbol{H}_{B,i}$ *is the* i*th row of* $\boldsymbol{H}_B$. *Then* $\hat{\boldsymbol{\beta}}_j, j = 1, 2, \ldots, q$, *span the same subspace as* $\boldsymbol{v}_j, j = 1, 2, \ldots, q$. [refer to [73] for the proof].

After establishing the equivalence, the regularization is applied on the *least squares* formulation in (37) via the $Lasso\text{-}penalty$ as shown below:

$$\begin{aligned} &\underset{\boldsymbol{A},\boldsymbol{B}}{\text{minimize}} \ \sum_{i=1}^{n} \| \boldsymbol{R}_W^{-T} \boldsymbol{H}_{B,i} - \boldsymbol{A}\boldsymbol{B}^T \boldsymbol{H}_{B,i} \|^2 + \sum_{j=1}^{q} \boldsymbol{\beta}_j^T \boldsymbol{S}_W \boldsymbol{\beta}_j, + \sum_{j=1}^{q} \lambda_{j,1} \| \boldsymbol{\beta}_j \|_1, \\ &\text{subject to} \ \ \boldsymbol{A}^T \boldsymbol{A} = I \end{aligned} \tag{38}$$

Since (38) is non-convex, finding the global optimum is often difficult. Qiao et al. [73], suggest a technique to obtain a local optimum by alternating optimization over $\boldsymbol{A}$ and $\boldsymbol{B}$. We refer readers to their article for details on their implementation.

Clemmensen et al. [18] also propose a similar sparse model using FLDA for classification problems. They also follow the approach of re-casting the optimization problem of FLDA into an equivalent least squares problem and then inducing sparsity by introducing a regularization term. However, the reformulation is achieved via an *optimal scoring* function that maps categorical variables to continuous variables via a sequence of *scorings*. Given a data matrix $\boldsymbol{X} \in \Re^{n\times p}$ and the samples belonging to one of the K classes, the equivalent regression problem can be formulated as:

$$\begin{aligned} \underset{\boldsymbol{\beta}_k, \boldsymbol{\theta}_k}{\text{minimize}} \quad & ||\boldsymbol{Y}\boldsymbol{\theta}_k - \boldsymbol{X}\boldsymbol{\beta}_k||_2^2 \\ \text{subject to} \quad & \frac{1}{n}\boldsymbol{\theta}_k^T \boldsymbol{Y}^T \boldsymbol{Y}\boldsymbol{\theta}_k = 1 \\ & \boldsymbol{\theta}_k^T \boldsymbol{Y}^T \boldsymbol{Y}\boldsymbol{\theta}_l = 0, \quad \forall l < k, \end{aligned} \tag{39}$$

where $\boldsymbol{\theta}_k$ is the score vector and $\boldsymbol{\beta}_k$ is the coefficient vector. It can be shown that the optimal vector $\boldsymbol{\beta}_k$ from (39) is also optimal to FLDA formulation in (28). Sparse discriminant vectors are then obtained by adding an l_1-penalty to the objective function in (39) as:

$$\begin{aligned} \underset{\boldsymbol{\beta}_k, \boldsymbol{\theta}_k}{\text{minimize}} \quad & ||\boldsymbol{Y}\boldsymbol{\theta}_k - \boldsymbol{X}\boldsymbol{\beta}_k||_2^2 + \gamma \boldsymbol{\beta}_k^T \boldsymbol{\Omega} \boldsymbol{\beta}_k + \lambda ||\boldsymbol{\beta}_k||_1 \\ \text{subject to} \quad & \frac{1}{n}\boldsymbol{\theta}_k^T \boldsymbol{Y}^T \boldsymbol{Y}\boldsymbol{\theta}_k = 1 \\ & \boldsymbol{\theta}_k^T \boldsymbol{Y}^T \boldsymbol{Y}\boldsymbol{\theta}_l = 0, \quad \forall l < k, \end{aligned} \tag{40}$$

where $\boldsymbol{\Omega}$ is a positive-definite matrix. The authors propose a simple iterative algorithm to obtain a local minima for the optimization problem in (40). The algorithm involves holding $\boldsymbol{\theta}_k$ fixed and optimizing with respect to $\boldsymbol{\beta}_k$, and holding $\boldsymbol{\beta}_k$ fixed and optimizing with respect to $\boldsymbol{\theta}_k$ until a pre-defined convergence criteria is met.

3.5 Discriminant Functions for High-Dimensional Data Classification

Linear discriminant analysis and QDA require the covariance within classes to be known a priori in order to establish a discriminant rule in classification problems. In many problems, since the covariance is not known a priori, researchers often attempt to estimate the covariance from the sample data. However, in high-dimensional problems, the sample covariance matrix is ill-conditioned and hence induces singularity in the estimation of the inverse covariance matrix. FLDA also faces similar challenges since *within-scatter* and *in-between scatter* are estimated from the sample data. In fact, even if the true covariance matrix is not ill-conditioned, the singularity of the sample covariance matrix will make these methods inapplicable when the dimensionality is larger than the sample size. Several authors performed a theoretical study on the performance of FLDA in high-dimensional classification settings. Bickel and Levina [2] showed that under some regularity conditions, as the ratio of features p and the number of samples n tend to infinity, the worst case misclassification rate tends to 0.5. This proves that as the dimensionality increases, FLDA is only as good as random guessing.

Several alternatives have been proposed to overcome the problem of singularity in LDA and QDA. Thomaz and Gillies [84] propose a new LDA algorithm (NLDA), which replaces the less reliable smaller eigenvalues of the sample covariance matrix with the grand mean of all eigenvalues and keeps larger eigenvalues unchanged. NLDA has been used successfully in face recognition problems. Xu et al. [94] state the lack of theoretical basis for NLDA and introduced a modified version of LDA called MLDA, which is based on a well-conditioned estimator for high-dimensional covariance matrices. This estimator has been shown to be more accurate than the sample covariance matrix asymptotically.

The assumption of independence in DLDA greatly reduces the number of parameters in the model and often results in an effective and interpretable classifier. Despite the fact that features will rarely be independent within a class, in the case of high-dimensional classification problems, the dependencies cannot be estimated due to lack of data. DLDA is shown to perform well for high-dimensional classification setting in spite of this naive assumption. Bickel and Levina [2] theoretically showed that it will outperform classical discriminant analysis in high-dimensional problems. However, one shortcoming of DLDA is that it uses all features and hence is not convenient for interpretation. Tibshirani et al. [86] introduced further regularization in DLDA using a procedure called *nearest shrunken centroids (NSC)* in order to improve misclassification error as well as interpretability. The regularization is introduced in a way that automatically assigns a weight *zero* to features that do not contribute to the class predictions. This is achieved by shrinking the classwise mean toward the overall mean, for each feature separately. We refer readers to [86] for a complete description of the method. DLDA integrated with *NSC* was applied to gene expression array analysis and is shown to be more accurate than other competing methods. The authors prove that the method is highly efficient in finding genes representative of small round blue cell tumors and leukemias. Several variations of NSC also exist in literature, for example [19,87]. Interestingly, NSC is also shown to be highly successful in open-set classification problems [77, 78] where the number of classes is not necessarily closed.

Another framework applied to high-dimensional classification problems include combining DLDA with shrinkage [71, 88]. Pang et al. [71] combined the shrinkage estimates of variances with diagonal discriminant scores to define two shrinkage-based discriminant rules called shrinkage-based DQDA (SDQDA) and shrinkage-based DLDA (SDLDA). Furthermore, the authors also applied regularization to further improve the performance of SDQDA and SDLDA. The discriminant rule combining shrinkage-based variances and regularization in diagonal discriminant analysis showed improvement over the original DQDA and DLDA, SVM, and k-Nearest Neighbors in many classification problems. Recently, Huang et al. [48] observed that the diagonal discriminant analysis suffers from serious drawback of having biased discriminant scores. Hence, they proposed bias-corrected diagonal discriminant rules by considering unbiased estimates for the discriminant scores. Especially in the case of highly unbalanced classification problems, the bias corrected rule is shown to outperform the standard rules.

Recently, SLDA has shown promise in high-dimensional classification problems. In [73], SLDA was applied to synthetic and real-world datasets including wine datasets and gene expression datasets and is shown to perform very well on training and testing data with lesser number of significant variables. The authors in [18] compared SLDA obtained via optimal scoring to other methods like shrunken centroid regularized discriminant analysis, sparse partial least squares regression and the elastic-net regression on a number of high-dimensional datasets and is shown to have comparable performance to other methods but with lesser number of significant variables.

4 Hybrid Classifiers

We now discuss an important set of classifiers that are frequently used for classification in the context of high-dimensional data problems. High dimensional datasets usually consist of irrelevant and redundant features that adversely effect the performance of traditional classifiers. Also, the high dimensionality of the data makes the estimation of statistical measures difficult. Hence, several techniques have been proposed in the literature to perform feature selection that selects relevant features suitable for classification [46]. Generally, feature selection is performed as a dimensionality reduction step prior to building the classification model using the traditional classifiers. Unlike other dimensionality reduction techniques like those based on transformation (e.g., principal component analysis) or compression (e.g., based on information theory), feature selection techniques do not alter the original dimensional space of the features, but merely select a subset of them [76]. Thus, they offer the advantage of interpretability by a domain expert as they preserve the original feature space. Also, feature selection helps to gain a deeper insight into the underlying processes that generated the data and thus plays a vital role in the discovery of *biomarkers* especially in biomedical applications [30]. Thus the classification framework can be viewed as a two-stage process with dimensionality reduction via feature selection being the first step followed by a classification model. We call these set of classifiers as *hybrid classifiers*, as different techniques pertaining to two stages have been combined to produce classification frameworks that have been successful in several high-dimensional problems. We briefly describe various feature selection techniques and also review the hybrid classifiers developed using these techniques for high-dimensional data problems.

4.1 Feature Selection Methods

Recently, feature selection has been an active area of research among many researchers due to tremendous advances in technology enabling collecting samples with hundreds and thousands of attributes in a single experiment. The goal of

feature selection techniques is to find an *optimal* set of features based on different measures of *optimality*. Irrespective of the measure of optimality, the selected subset of features should ideally possess the following characteristics [40]:

- The cardinality of the subset should be minimal such that it is necessary and sufficient to accurately predict the class of unknown samples,
- The subset of features should improve the prediction accuracy of the classifier run on data containing only these features rather than on the original dataset with all the features,
- The resulting class distribution, given only the values for the selected features, is as close as possible to the original class distribution given all feature values.

Based on the above feature characteristics, it is obvious that *irrelevant* features would not be part of the optimal set of features, where an irrelevant feature with respect to the target class is defined as follows [97].

Let F be the full set of features and C be the target class. Define $F_i \in F$ and $S_i = F - F_i$.

Definition 1 (Irrelevance). A feature F_i is irrelevant if and only if

$$\forall S_i^{'} \subseteq S_i, \quad \mathbf{P}(C|F_i, S_i^{'}) = \mathbf{P}(C|S_i^{'})$$

Irrelevance simply means that it is not necessary for classification since the class distribution given any subset of other features does not change after eliminating the feature.

The definition of relevance is not as straightforward as irrelevance. There have been several definitions for relevance in the past; however, Kohavi and John [58] argued that the earlier definitions weren't adequate to accurately classify the features. Hence, they defined relevance in terms of an optimal Bayes classifier. A feature F_i is strongly relevant if removal of F_i alone will result in decrease of performance of an optimal Bayes classifier. A feature F_i is weakly relevant if it is not strongly relevant and there exists a subset of features, $S_i^{'}$, such that the performance of a Bayes classifier on $S_i^{'}$ is worse than the performance on $S_i^{'}$ U $\{F_i\}$.

Definition 2 (Strong Relevance). A feature F_i is strongly relevant if only and if:

$$\mathbf{P}(C|F_i, S_i^{'}) \neq \mathbf{P}(C|S_i^{'}), \quad S_i^{'} \subseteq S_i \tag{41}$$

Definition 3 (Weak Relevance). A feature F_i is weakly relevant if only and if:

$$\mathbf{P}(C|F_i, S_i) = \mathbf{P}(C|S_i) \quad \text{and,} \quad \exists S_i^{'} \subseteq S_i, \quad \mathbf{P}(C|F_i, S_i^{'}) \neq \mathbf{P}(C|S_i^{'}) \tag{42}$$

Strong relevance implies that the feature is indispensable and is required for an optimal set, while weak relevance implies that the feature may be required sometimes to improve the prediction accuracy. From this, one may conclude that the optimal set should consist of all the strongly relevant features, none of the irrelevant

features and some of the weakly irrelevant features. However, the definitions do not explicitly mention which of the weakly relevant features should be included and which of them excluded. Hence, Yu and Liu [97] claim that the weakly relevant features should be further classified to discriminate among the redundant features and the non-redundant features, since earlier research efforts showed that along with irrelevant features, redundant features also adversely affect the classifier performance. Before we provide definitions, we introduce another concept called feature's *Markov Blanket* as defined by Koller and Sahami [59].

Definition 4 (Markov Blanket). Given a feature F_i, let $M_i \subset F(F_i \notin M_i)$, M_i is said to be a Markov blanket for F_i if only and if:

$$\mathbf{P}(F - M_i - \{F_i\}, C | F_i, M_i) = P(F - M_i - \{F_i\}, C | M_i) \tag{43}$$

The Markov blanket M_i could be imagined as a *blanket* for the feature F_i that subsumes not only the information that F_i possesses about target class C, but also about other features. It is also important to note that the strongly relevant features cannot have a Markov Blanket. Since the irrelevant features do not contribute to classification, Yu and Liu [97] further classified the weakly relevant features into either *redundant* or *non-redundant* using the concept of Markov blanket:

Definition 5 (Redundant Feature). Given a set of current features G, a feature is redundant and hence should be removed from G if and only if it has a Markov Blanket within G.

From the above definitions, it is clear that the optimal set of features should consist of all of the strongly relevant features and the weakly relevant non-redundant features. However, an exhaustive search over the feature space is intractable since there are 2^p possibilities with p being the number of features. Hence, over the past decade, several heuristic and approximate methods have been developed to perform feature selection. In the context of classification, feature selection techniques can be organized into three categories, depending on how they combine the feature selection search with the construction of the classification model: filter methods, wrapper methods and embedded methods [76]. While all methods define some criterion measure to eliminate the irrelevant features, very few methods attempt to eliminate the redundant features as well. Here, we briefly describe methods in each of the three categories.

4.1.1 Filter Methods

Filter methods assess feature relevance from the intrinsic properties of the data. In most cases the features are ranked using a feature relevance score and the low-scoring features are removed. The reduced data obtained from considering only the selected features are then presented as an input to the classification algorithm. Filter techniques offer several advantages including scalability to high-dimensional

datasets, being computationally efficient, and are independent of the classification algorithm. This independency offers the advantage of performing feature selection only once and then evaluating different classifiers.

Some univariate filter techniques perform simple hypothesis testing like Chi-Square (χ^2) test or t-test to eliminate the irrelevant features, while other techniques estimate information theoretic measures like information gain and gain-ratio to perform the filtering process [1]. Although these techniques are simple, fast and highly scalable, they ignore feature dependencies which may lead to worse classification performance as compared with other feature selection techniques. In order to account for feature dependencies, a number of multivariate filter techniques were introduced. The multivariate filter methods range from accounting for simple mutual interactions [4] to more advanced solutions exploring higher order interactions. One such technique called correlation-based feature selection (CFS) introduced by Hall [42], evaluates a subset of features by considering the individual predictive ability of each feature along with the degree of redundancy between them:

$$\text{CFS}_S = \frac{k\Phi_{cf}}{\sqrt{k + k(k-1)\Phi_{ff}}} \tag{44}$$

where CFS_S is the score of a feature subset S containing k features, Φ_{cf} is the average feature-to-class correlation ($f \in S$), and Φ_{ff} is the average feature-to-feature correlation. Unlike the univariate filter methods, CFS presents a score for a subset of features. Since, exhaustive search is intractable, several heuristic techniques like greedy hill-climbing or best-first search have been proposed to find the feature subset with the highest CFS score.

Another important multivariate filter method called Markov blanket filtering was introduced by Koller and Sahami [59]. The idea here being that once we find a Markov blanket of feature F_i in a feature set G, we can safely remove F_i from G without compromising on the class distribution. Since estimating the Markov blanket for a feature is hard, Koller and Sahami propose a simple iterative algorithm that starts with the full feature set $F = G$ and then repeatedly eliminates one feature at a time based on cross-entropy of each feature until a pre-selected number of features are removed.

Koller and Sahami further prove that in such a sequential elimination process in which unnecessary features are removed one by one, a feature tagged as unnecessary based on the existence of a Markov blanket M_i remains unnecessary in later stages when more features have been removed. Also, the authors claim that the process removes all the irrelevant as well as redundant features. Several variations to the Markov blanket filtering method like Grow-Shrink (GS) algorithms, incremental association Markov blanket (IAMB), Fast-IAMB and recently λ-IAMB have been proposed by other authors [36]. Due to space constraints, we mention other interesting multivariate filter methods like fast-correlation-based feature selection (FCBF) ([96]), minimum redundancy-maximum relevance (MRMR) [26], and uncorrelated shrunken centroid (USC) [95] algorithms.

Statnikov et al. [82] recently performed a comprehensive comparative study between Random Forests [8] and SVM for microarray-based cancer classification. They adopt several filter methods like sequential filtering techniques as a pre-processing step to select a subset of features which are then used as input to the classifiers. It is shown that on an average, SVM outperform Random Forests on most microarray datasets. Recently, Pal and Moody [68] studied the effect of dimensionality on performance of SVM using four feature selection techniques namely CFS, MRMR, Random Forests and SVM-RFE [41] on hyperspectral data. Unlike earlier findings, they show that dimensionality might affect the performance of SVM and hence a pre-processing step like feature selection might still be useful to improve the performance.

4.1.2 Wrapper Methods

As seen in the earlier section, filter methods treat the problem of finding a good feature subset independently of the classifier building step. Wrapper methods, on the other hand, integrate the classifier hypothesis search within the feature subset search. In this framework, a search procedure in the feature space is first defined, and various subsets of features are generated and evaluated. The evaluation of a specific feature subset is obtained by training and testing a specific classification model, making this approach tailored to a specific classification algorithm [58, 76]. Advantages of wrapper methods include consideration of feature dependencies and the ability to include interactions between the feature subset search and model selection. A common drawback includes the risk of higher overfitting than the filter methods and could be computationally intensive if the classification model especially has a high computational cost.

The wrapper methods generally employ a search algorithm in order to search through the space of all feature subsets. The search algorithm is *wrapped* around the classification model which provides a feature subset that can be evaluated by the classification algorithm. As mentioned earlier, since an exhaustive search is not practical, heuristic search methods are used to guide the search. These search methods can be broadly classified as deterministic and randomized search algorithms. Deterministic search methods include a set of sequential search techniques like the Sequential Forward Selection [56], Sequential Backward Selection [56], Plus-l Minus-r Selection [31], Bidirectional Search, Sequential Floating Selection [72] etc., where the features are either sequentially added or removed based on some criterion measure. Randomized Search algorithms include popular techniques like Genetic Algorithms [20], Simulated Annealing [55], Randomized Hill Climbing [81], etc.

4.1.3 Embedded Methods

Embedded methods integrate the search for an optimal subset of features into the classifier construction and can be seen as a search in the combined space of feature subsets and hypotheses. Similar to wrapper methods, embedded approaches are also specific to a given learning algorithm. The advantages of embedded methods include the interaction with the classification model, but unlike the wrapper methods, also has the advantage to be less computationally intensive [76].

Recently embedded methods have gained importance among the research community due to their advantages. The embedded characteristic of several classifiers to eliminate input features futile to classification and thus select a subset of features, has been exploited by several authors. Examples include the use of random forests (discussed later) in an embedded way to calculate the importance of each feature [24, 51]. Another line of embedded feature selection techniques uses the weights of each feature in linear classifiers, such as SVM [41] and logistic regression [63]. These weights are used as a measure of relevance of each feature, and thus allow for the removal of features with very small weights. Also, recently regularized classifiers like Lasso and elastic-net have also been successfully employed in performing feature selection in microarray gene analysis [99]. Another interesting technique called feature selection via sparse SVM has been recently proposed by Tan et al. [83]. This technique called the feature Generating machine (FGM) adds a binary variable for every feature in the sparse formulation of SVM via l_0-norm and the authors propose a cutting plane algorithm combined with multiple kernel learning to efficiently solve the convex relaxation of the optimization problem.

5 Ensemble Classifiers

Ensemble classifiers have gained increasing attention from the research community over the past years, ranging from simple averaging of individually trained neural networks to the combination of thousands of decision trees to build Random Forests [8], to the boosting of weak classifiers to build a strong classifier where the training of each subsequent classifier depends on the results of all previously trained classifiers [75]. The main idea of an ensemble methodology is to combine a set of models, each of which solves the same original task, in order to obtain a better composite global model, with more accurate and reliable estimates or decisions. They combine multiple hypotheses of different models with the hope to form a better classifier. Alternatively, an ensemble classifier can also be viewed as a technique for combining many weak learners in an attempt to produce a strong learner. Hence an ensemble classifier is itself a supervised learning algorithm capable of making prediction on unknown sample data. The trained ensemble classifier, therefore, represents a single hypothesis that is not necessarily contained within the hypothesis space of the constituent models. This flexibility of ensemble classifiers can theoretically overfit to the training data more than a single model

would, but however surprisingly, in practice, some ensemble techniques (especially bagging and Random Forests) tend to reduce problems related to overfitting of the training data.

In the past few years, experimental studies show that combining the outputs of multiple classifiers similar to ensemble methods reduces the generalization error [25]. Ensemble methods are particularly effective due to the phenomenon that various types of classifiers have different inductive biases. Additionally, ensemble methods can effectively make use of such diversity to reduce the variance-error while keeping the bias-error in check. In certain situations, an ensemble can also reduce bias-error, as shown by the theory of large margin classifiers. So, diversified classifiers help in building a lesser number of classifiers, especially in the case of Random Forests. The increase in prediction accuracy does come at a cost of performing more calculations in comparison to a single model. So, the ensemble methods can be thought of as a way to compensate for a poor learner by performing a lot of computations. So, a fast poor learner like decision trees have certainly gained from ensemble methods; although slow algorithms can also benefit from ensemble techniques.

Recently, ensemble methods have shown promise in high-dimensional data classification problems. In particular, bagging methods, random forests and boosting have been particularly impressive due to their flexibility to create stronger classifiers from weak classifiers. Here, we describe two methods: AdaBoost and Random Forests, and show their importance in high-dimensional problems.

5.1 AdaBoost

Boosting [33, 79] is a general method which attempts to boost the accuracy of any given learning algorithm. The inception of boosting can be traced back to a theoretical framework for studying machine learning called the "PAC" learning model, [91]. Kearns and Valiant [54] were among the first authors to pose the question of whether a *weak* learner which is only slightly correlated with the true classification and performs just slightly better than random guessing in the PAC model can be *boosted* into an accurate *strong* learning algorithm that is arbitrarily well-correlated with true classification. Schapire [79] proposed the first provable polynomial-time boosting algorithm in 1989. A year later, Freund [32] developed a much more efficient boosting algorithm which, although optimal in a certain sense, nevertheless suffered from certain practical drawbacks.

Boosting encompasses a family of methods that produces a series of classifiers. The training set used for each member of the series is chosen based on the performance of the earlier classifier(s) in the series. Unlike other committee methods like bagging [6], in boosting, the base classifiers are trained in sequence, and each base classifier is trained using a weighted variant of the dataset in which the individual weighting coefficient depends on the performance of previous classifiers.

In particular, points that are misclassified by one of the base classifiers are given greater weight when used to train the next classifier in the sequence. Once all the classifiers have been trained, their predictions are then combined through a weighted majority voting scheme.

AdaBoost, short for Adaptive Boosting, formulated by Yoav Freund and Robert Schapire [33], solved many of the practical difficulties of earlier boosting algorithms. It can be considered as classification framework that can be used in conjunction with many other learners to improve their performance. AdaBoost is adaptive in the sense that subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifiers. The framework provides a new weak classifier with a form of training set that is representative of the performance of previous classifiers. The weights of those training samples that are misclassified by earlier weak learners are given higher values than those that are correctly classified. This allows the new classifier to adapt to the misclassified training samples and focus on predicting them correctly. After the training phase is complete, each classifier is assigned a weight and their outputs are linearly combined to make predictions on the unknown sample. Generally, it provides a significant performance boost to weak learners that are only slightly better than random guessing. Even classifiers with a higher error rate could also be useful as they will have negative coefficients in the final linear combination of classifiers and hence behave like their inverses.The precise form of the AdaBoost algorithm is described below.

Consider a binary classification problem, in which the training data comprises input vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ along with corresponding binary target variables given by t where $t_n \in \{-1, 1\}$. Each data point is given an associated weighting parameter w_n, which is initially set 1/N for all data points. We assume that we have a procedure available for training a base classifier using weighted data to give a function $y(\boldsymbol{x}) \in \{-1, 1\}$.

- Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, 2, \ldots, N$.
- For $m = 1, \ldots, M$:
  (a) Fit a classifier $y_m(\boldsymbol{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\boldsymbol{x}_n) \neq t_n) \tag{45}$$

  where $I(y_m(\boldsymbol{x}_n) \neq t_n)$ is the indicator function and equals 1 when $(y_m(\boldsymbol{x}_n) \neq t_n)$ and 0 otherwise.
  (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\boldsymbol{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} \tag{46}$$

and then use ϵ_m to evaluate

$$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\} \tag{47}$$

(c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(\boldsymbol{x}_n) \neq t_n)\} \tag{48}$$

- Make predictions using the final model, which is given by:

$$Y_M^{(\boldsymbol{x})} = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x})\right) \tag{49}$$

We see that the first weak learner $y_1(\boldsymbol{x})$ is trained using weighting coefficients $w_n^{(1)}$ that are all equal and hence is similar to training a single classifier. From (48), we see that in subsequent iterations the weighting coefficients $w_n^{(m)}$ are increased for data points that are misclassified and decreased for data points that are correctly classified. Successive classifiers are therefore forced to focus on points that have been misclassified by previous classifiers, and data points that continue to be misclassified by successive classifiers receive even greater weight. The quantities ϵ_m represent weighted measures of the error rates of each of the base classifiers on the dataset. We therefore see that the weighting coefficients α_m defined by (47) give greater weight to more accurate classifiers when computing the overall output for unknown samples given by (49). AdaBoost is sensitive to noisy data and outliers. In some problems, however, it can be less susceptible to the overfitting problem than most learning algorithms. We refer readers to [34] for a more theoretical discussion on the performance of the AdaBoost algorithm.

Boosting framework in conjunction with several classifiers have been successfully applied to high-dimensional data problems. As discussed in [7] boosting framework can be viewed as a functional gradient descent technique. This analysis of boosting connects the method to more common optimization view of statistical inference. Bühlmann and Yu [11] investigate one such computationally simple variant of boosting called L_2Boost, which is constructed from a functional gradient descent algorithm with the L_2-loss function. In particular, they study the algorithm with cubic smoothing spline as the base learner and show empirically on real and simulation datasets the effectiveness of the algorithm in high-dimensional predictors. Bühlmann [10] presented an interesting review on how the boosting methods can be useful for high-dimensional problems. He proposes that inherent variable selection and assigning variable amount of degrees of freedom to the selected variables by boosting algorithms could be a reason for high performance in high-dimensional problems. Additionally, he suggests that boosting yields consistent function approximations even when the number of predictors grow fast to infinity,

where the underlying true function is *sparse*. Dettling and Bühlmann [23] applied boosting to perform classification tasks with gene expression data. A modified boosting framework in conjunction with decision trees that does pre-selection was proposed and shown to yield slight to drastic improvement in performance on several publicly available datasets.

5.2 Random Forests

Random forests are an ensemble classifier that consists of many tree-type classifiers with each classifier being trained on a bootstrapped sample of the original training data, and searches only across a randomly selected subset of the input variables to determine a split (for each node). For classification, each tree in the Random Forest casts a unit vote for the most popular class at input x. The output of the Random Forest for an unknown sample is then determined by a majority vote of the trees. The algorithm for inducing Random Forests was developed by Leo Breiman [8] and can be summarized as below:

Assume the number of training samples be N, and the number of features be given by M. Also, assume that random m number of features ($m < M$) used for decision at each split. Each tree in the Random Forest is constructed as follows:

- Choose a training set for this tree by bootstrapping the original training set n times. The rest of the samples are used as a testing set to estimate the error of the tree.
- For each node of the tree, the best split is based on randomly choosing m features for each training sample and the tree is fully grown without pruning.

For prediction, a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the class obtained from majority vote of all the trees is reported as Random Forest prediction.

Random Forests are considered one of the most accurate classifiers and are reported to have several advantages. Random Forests are shown to handle many features and also assign a weight relative to their importance in classification tasks which can be further explored for feature selection. The computational complexity of the algorithm is reduced as the number of features used for each split is bounded by m. Also, non-pruning of the trees also helps in reducing the computational complexity further. Such random selection of features to build the trees also limits the correlation among the trees thus resulting in error rates similar to those of AdaBoost. The analysis of Random Forests shows that its computational time is $cT\sqrt{M}N\ \log(N)$ where c is a constant, T is the number of trees in the ensemble, M is the number of features and N is the number of training samples in the dataset. It should be noted that although Random Forests are not computationally intensive, they require a fair amount of memory as they store an N by T matrix in memory. Also, Random Forests have sometimes been shown to overfit to the data in some classification problems.

Random Forests, due to the aforementioned advantages, can handle high-dimensional data by building a large number of trees using only a subset of features. This combined with the fact that the random selection of features for a split seeks to minimize the correlation between the trees in the ensemble, certainly helps in building an ensemble classifier with high generalization accuracy for high-dimensional data problems. Gislason et al. [38] performed a comparative study among Random Forests and other well-known ensemble methods for multisource remote sensing and geographic data. They show that Random Forests outperform a single CART classifier and perform on par with other ensemble methods like bagging and boosting. On a related remote sensing application, Pal [67] investigated the use of Random Forests for classification tasks and compared their performance with SVM. Pal showed that Random Forests perform equally well to SVM in terms of classification accuracy and training time. Additionally, Pal concludes that the user-defined parameters in Random Forests are less than those required for SVM. Pang et al. [70] proposed a pathway-based classification and regression method using Random Forests to analyze gene expression data. The proposed method allows to rank important pathways, discover important genes and find pathway-based outlying cases. Random Forests, in comparison with other machine learning algorithms, were shown to have either lower or second-lowest classification error rates. Recently, Genuer et al. [37] used Random Forests to perform feature selection as well. The authors propose a strategy involving ranking of the explanatory variables using the Random Forests score of importance.

6 Software Packages

We briefly describe some publicly available resources that have implemented the methods discussed here. These packages are available in several programming languages including Java, Matlab and R softwares. LibSVM [14] is an integrated software that implements SVM and offers several extensions to Java, C++, Python, R and Matlab. Weka [43] is a collection of machine learning algorithms for data mining tasks implemented in Java. It contains methods to perform classification as well as feature selection on high-dimensional datasets. The FSelector package in R language offers several algorithms to perform filter, wrapper and embedded feature selection. Several packages are also available to perform regularization. Glmnet for Matlab[1] solves for regularized paths in Generalized Linear models while Lasso2[2] and LARS[3] packages provide similar algorithms in R language. Random Forests and AdaBoost algorithms are also available via randomForest[4] and adabag[5] packages in R language.

[1] http://www-stat.stanford.edu/~tibs/glmnet-matlab/.

[2] http://cran.r-project.org/web/packages/lasso2/index.html.

[3] http://cran.r-project.org/web/packages/lars/index.html.

[4] http://cran.r-project.org/web/packages/randomForest/index.html.

[5] http://cran.r-project.org/web/packages/adabag/index.html.

7 Concluding Remarks

We have presented several classification problems for high-dimensional data problems. Several researchers have focused on extended the traditional algorithms like LDA and Logistic Regression in the context of high-dimensional data settings. Though some success is seen on this front, recently, the focus has shifted to applying regularization techniques and ensemble type methods to make more accurate predictions. Though the progress made so far is encouraging, we believe that high-dimensional data classification would continue to be an active area of research as the technological innovations continue to evolve and become more effective.

Acknowledgements This research is partially supported by NSF & DTRA grants

References

1. Ben-Bassat, M.: 35 use of distance measures, information measures and error bounds in feature evaluation. In: Handbook of Statistics, vol. 2, pp. 773–791. North-Holland, Amsterdam (1982)
2. Bickel, P., Levina, E.: Some theory for fisher's linear discriminant function, Naive Bayes', and some alternatives when there are many more variables than observations. Bernoulli **10**(6), 989–1010 (2004)
3. Bishop, C.: Pattern Recognition and Machine Learning. Springer, New York (2006)
4. Bo, T., Jonassen, I.: New feature subset selection procedures for classification of expression profiles. Genome Biol. **3**(4), 1–11 (2002)
5. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
6. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
7. Breiman, L.: Prediction games and arcing algorithms. Neural Comput. **11**(7), 1493–1517 (1999)
8. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
9. Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., Furey, T., Ares, M., Haussler, D.: Knowledge-based analysis of microarray gene expression data by using support vector machines. Proc. Natl. Acad. Sci. USA **97**(1), 262 (2000)
10. Bühlmann, P.: Boosting methods: why they can be useful for high-dimensional data. In: Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC) (2003)
11. Bühlmann, P., Yu, B.: Boosting with the l 2 loss: regression and classification. J. Am. Stat. Assoc. **98**(462), 324–339 (2003)
12. Burges, C.: Advances in Kernel Methods: Support Vector Learning. The MIT Press, Cambridge (1999)
13. Byvatov, E., Schneider, G., et al.: Support vector machine applications in bioinformatics. Appl. Bioinformatics **2**(2), 67–77 (2003)
14. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), 1–27 (2011)
15. Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing multiple parameters for support vector machines. Mach. Learn. **46**(1), 131–159 (2002)
16. Chung, K., Kao, W., Sun, C., Wang, L., Lin, C.: Radius margin bounds for support vector machines with the rbf kernel. Neural Comput. **15**(11), 2643–2681 (2003)

17. Clarke, R., Ressom, H., Wang, A., Xuan, J., Liu, M., Gehan, E., Wang, Y.: The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. Nat. Rev. Cancer **8**(1), 37–49 (2008)
18. Clemmensen, L., Hastie, T., Witten, D., Ersbøll, B.: Sparse discriminant analysis. Technometrics **53**(4), 406–413 (2011)
19. Dabney, A.: Classification of microarrays to nearest centroids. Bioinformatics **21**(22), 4148–4154 (2005)
20. Davis, L., Mitchell, M.: Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York (1991)
21. De Maesschalck, R., Jouan-Rimbaud, D., Massart, D.: The mahalanobis distance. Chemometr. Intell. Lab. Syst. **50**(1), 1–18 (2000)
22. Den Hertog, D.: Interior Point Approach to Linear, Quadratic and Convex Programming: Algorithms and Complexity. Kluwer Academic, Norwell (1992)
23. Dettling, M., Bühlmann, P.: Boosting for tumor classification with gene expression data. Bioinformatics **19**(9), 1061–1069 (2003)
24. Díaz-Uriarte, R., De Andres, S.: Gene selection and classification of microarray data using random forest. BMC Bioinformatics **7**(3), 1–13 (2006)
25. Dietterich, T.: Ensemble methods in machine learning. In: Multiple Classifier Systems, pp. 1–15. Springer, Heidelberg (2000)
26. Ding, C., Peng, H.: Minimum redundancy feature selection from microarray gene expression data. J. Bioinforma. Comput. Biol. **3**(2), 185–205 (2005)
27. Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley-Interscience, London (2001)
28. Dudoit, S., Fridlyand, J., Speed, T.: Comparison of discrimination methods for the classification of tumors using gene expression data. J. Am. Stat. Assoc. **97**(457), 77–87 (2002)
29. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. Ann. Stat. **32**(2), 407–499 (2004)
30. Fenn, M., Pappu, V.: Data mining for cancer biomarkers with raman spectroscopy. In: Data Mining for Biomarker Discovery, pp. 143–168. Springer, Berlin (2012)
31. Ferri, F., Pudil, P., Hatef, M., Kittler, J.: Comparative study of techniques for large-scale feature selection. In: Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies, and Hybrid Systems, pp. 403–413. IEEE Xplore (1994)
32. Freund, Y.: Boosting a weak learning algorithm by majority. Inf. Comput. **121**(2), 256–285 (1995)
33. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Proceedings of the 13th International Conference on Machine Learning, pp. 148–156. Morgan Kaufmann, Los Altos (1996)
34. Freund, Y., Schapire, R., Abe, N.: A short introduction to boosting. J. Jpn. Soc. Artif. Intell. **14**(1612), 771–780 (1999)
35. Friedman, J., Hastie, T., Tibshirani, R.: The Elements of Statistical Learning. Springer Series in Statistics. Springer, Berlin (2001)
36. Fu, S., Desmarais, M.: Markov blanket based feature selection: a review of past decade. In: Proceedings of the World Congress on Engineering, vol. 1, pp. 321–328 (2010). Citeseer
37. Genuer, R., Poggi, J., Tuleau-Malot, C.: Variable selection using random forests. Pattern Recognit. Lett. **31**(14), 2225–2236 (2010)
38. Gislason, P., Benediktsson, J., Sveinsson, J.: Random forests for land cover classification. Pattern Recognit. Lett. **27**(4), 294–300 (2006)
39. Guo, X., Yang, J., Wu, C., Wang, C., Liang, Y.: A novel ls-svms hyper-parameter selection based on particle swarm optimization. Neurocomputing **71**(16), 3211–3215 (2008)
40. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)
41. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Mach. Learn. **46**(1), 389–422 (2002)
42. Hall, M.: Correlation-based feature selection for machine learning. Ph.D. thesis, The University of Waikato (1999)

43. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD Explor. Newslett. **11**(1), 10–18 (2009)
44. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall, Englewood (2004)
45. Herbert, P., Tiejun, T.: Recent advances in discriminant analysis for high-dimensional data classification. J. Biom. Biostat. **3**(2), 1–2 (2012)
46. Hua, J., Tembe, W., Dougherty, E.: Performance of feature-selection methods in the classification of high-dimension data. Pattern Recognit. **42**(3), 409–424 (2009)
47. Huang, C., Wang, C.: A ga-based feature selection and parameters optimization for support vector machines. Expert Syst. Appl. **31**(2), 231–240 (2006)
48. Huang, S., Tong, T., Zhao, H.: Bias-corrected diagonal discriminant rules for high-dimensional classification. Biometrics **66**(4), 1096–1106 (2010)
49. Hughes, G.: On the mean accuracy of statistical pattern recognizers. IEEE Trans. Inf. Theory **14**(1), 55–63 (1968)
50. Jain, A., Duin, R., Mao, J.: Statistical pattern recognition: a review. IEEE Trans. Pattern Anal. Mach. Intell. **22**(1), 4–37 (2000)
51. Jiang, H., Deng, Y., Chen, H., Tao, L., Sha, Q., Chen, J., Tsai, C., Zhang, S.: Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes. BMC Bioinformatics **5**(81), 1–12 (2004)
52. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Machine Learning: ECML-98, pp. 137–142. Springer, Berlin (1998)
53. Johnstone, I., Titterington, D.: Statistical challenges of high-dimensional data. Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. **367**(1906), 4237–4253 (2009)
54. Kearns, M., Valiant, L.: Learning Boolean formulae or finite automata is as hard as factoring. Center for Research in Computing Technology, Aiken Computation Laboratory, Harvard University (1988)
55. Kirkpatrick, S., Gelatt, C. Jr., Vecchi, M.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
56. Kittler, J.: Feature set search algorithms. In: Pattern Recognition and Signal Processing, pp. 41–60. Sijthoff and Noordhoff, Alphen aan den Rijn (1978)
57. Kleinbaum, D., Klein, M., Pryor, E.: Logistic Regression: A Self-learning Text. Springer, Berlin (2002)
58. Kohavi, R., John, G.: Wrappers for feature subset selection. Artif. Intell. **97**(1–2), 273–324 (1997)
59. Koller, D., Sahami, M.: Toward optimal feature selection. In: Proceedings of the 13th International Conference on Machine Learning, pp. 284–292 (1996)
60. Köppen, M.: The curse of dimensionality. In: Proceedings of the 5th Online World Conference on Soft Computing in Industrial Applications (WSC5), pp. 4–8 (2000)
61. Lin, S., Lee, Z., Chen, S., Tseng, T.: Parameter determination of support vector machine and feature selection using simulated annealing approach. Appl. Soft Comput. **8**(4), 1505–1512 (2008)
62. Lin, S., Ying, K., Chen, S., Lee, Z.: Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Syst. Appl. **35**(4), 1817–1824 (2008)
63. Ma, S., Huang, J.: Regularized roc method for disease classification and biomarker selection with microarray data. Bioinformatics **21**(24), 4356–4362 (2005)
64. McLachlan, G., Wiley, J.: Discriminant Analysis and Statistical Pattern Recognition. Wiley Online Library, New York (1992)
65. Minh, H., Niyogi, P., Yao, Y.: Mercer's theorem, feature maps, and smoothing. In: Learning Theory, pp. 154–168. Springer Berlin Heidelberg (2006)
66. Mourão-Miranda, J., Bokde, A., Born, C., Hampel, H., Stetter, M.: Classifying brain states and determining the discriminating activation patterns: support vector machine on functional MRI data. NeuroImage **28**(4), 980–995 (2005)
67. Pal, M.: Support vector machine-based feature selection for land cover classification: a case study with dais hyperspectral data. Int. J. Remote Sens. **27**(14), 2877–2894 (2006)

68. Pal, M., Foody, G.: Feature selection for classification of hyperspectral data by svm. IEEE Trans. Geosci. Remote Sens. **48**(5), 2297–2307 (2010)
69. Pal, M., Mather, P.: Support vector machines for classification in remote sensing. Int. J. Remote Sens. **26**(5), 1007–1011 (2005)
70. Pang, H., Lin, A., Holford, M., Enerson, B., Lu, B., Lawton, M., Floyd, E., Zhao, H.: Pathway analysis using random forests classification and regression. Bioinformatics **22**(16), 2028–2036 (2006)
71. Pang, H., Tong, T., Zhao, H.: Shrinkage-based diagonal discriminant analysis and its applications in high-dimensional data. Biometrics **65**(4), 1021–1029 (2009)
72. Pudil, P., Novovičová, J., Kittler, J.: Floating search methods in feature selection. Pattern Recognit. Lett. **15**(11), 1119–1125 (1994)
73. Qiao, Z., Zhou, L., Huang, J.: Sparse linear discriminant analysis with applications to high dimensional low sample size data. Int. J. Appl. Math. **39**(1), 6–29 (2009)
74. Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C., Angelo, M., Ladd, C., Reich, M., Latulippe, E., Mesirov, J., et al.: Multiclass cancer diagnosis using tumor gene expression signatures. Proc. Natl. Acad. Sci. USA **98**(26), 15149–15154 (2001)
75. Rokach, L.: Ensemble-based classifiers. Artif. Intell. Rev. **33**(1), 1–39 (2010)
76. Saeys, Y., Inza, I., Larrañaga, P.: A review of feature selection techniques in bioinformatics. Bioinformatics **23**(19), 2507–2517 (2007)
77. Schaalje, G., Fields, P.: Open-set nearest shrunken centroid classification. Commun. Stat. Theory Methods **41**(4), 638–652 (2012)
78. Schaalje, G., Fields, P., Roper, M., Snow, G.: Extended nearest shrunken centroid classification: a new method for open-set authorship attribution of texts of varying sizes. Lit. Linguist. Comput. **26**(1), 71–88 (2011)
79. Schapire, R.: The strength of weak learnability. Mach. Learn. **5**(2), 197–227 (1990)
80. Schoonover, J., Marx, R., Zhang, S.: Multivariate curve resolution in the analysis of vibrational spectroscopy data files. Appl. Spectrosc. **57**(5), 483–490 (2003)
81. Skalak, D.: Prototype and feature selection by sampling and random mutation hill climbing algorithms. In: Proceedings of the 11th International Conference on Machine Learning, pp. 293–301 (1994). Citeseer
82. Statnikov, A., Wang, L., Aliferis, C.: A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. BMC Bioinformatics **9**(319), 1–10 (2008)
83. Tan, M., Wang, L., Tsang, I.: Learning sparse svm for feature selection on very high dimensional datasets. In: Proceedings of the 27th International Conference on Machine Learning, pp. 1047–1054 (2010)
84. Thomaz, C., Gillies, D.: A maximum uncertainty lda-based approach for limited sample size problems - with application to face recognition. In: Proceedings of the 18th Brazilian Symposium on Computer Graphics and Image Processing, pp. 89–96. IEEE, Natal (2005)
85. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Series B Methodol. **58**, 267–288 (1996)
86. Tibshirani, R., Hastie, T., Narasimhan, B., Chu, G.: Diagnosis of multiple cancer types by shrunken centroids of gene expression. Proc. Natl. Acad. Sci. **99**(10), 6567–6572 (2002)
87. Tibshirani, R., Hastie, T., Narasimhan, B., Chu, G.: Class prediction by nearest shrunken centroids, with applications to dna microarrays. Stat. Sci. **18**, 104–117 (2003)
88. Tong, T., Chen, L., Zhao, H.: Improved mean estimation and its application to diagonal discriminant analysis. Bioinformatics **28**(4), 531–537 (2012)
89. Trafalis, T., Ince, H.: Support vector machine for regression and applications to financial forecasting. In: Proceedings of the International Joint Conference on Neural Networks, vol. 6, pp. 348–353. IEEE, New York (2000)
90. Trunk, G.: A problem of dimensionality: a simple example. IEEE Trans. Pattern Anal. Mach. Intell. **3**(3), 306–307 (1979)
91. Valiant, L.: A theory of the learnable. Commun. ACM **27**(11), 1134–1142 (1984)
92. Vapnik, V.: The nature of statistical learning theory. springer (2000)

93. Vapnik, V., Chapelle, O.: Bounds on error expectation for support vector machines. Neural Comput. **12**(9), 2013–2036 (2000)
94. Xu, P., Brock, G., Parrish, R.: Modified linear discriminant analysis approaches for classification of high-dimensional microarray data. Comput. Stat. Data Anal. **53**(5), 1674–1687 (2009)
95. Yeung, K., Bumgarner, R., et al.: Multiclass classification of microarray data with repeated measurements: application to cancer. Genome Biol. **4**(12), R83 (2003)
96. Yu, L., Liu, H.: Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proceedings of the 20th International Conference on Machine Learning, pp. 856–863 (2003)
97. Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. J. Mach. Learn. Res. **5**, 1205–1224 (2004)
98. Zhang, L., Lin, X.: Some considerations of classification for high dimension low-sample size data. Stat. Methods Med. Res. **22**, 537–550 (2011)
99. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. R. Stat. Soc. Ser. B Stat. Methodol. **67**(2), 301–320 (2005)