David Pinto

Bayesian Classification with Regularized Gaussian Models

Belo Horizonte, MG, Brasil

Novembro de 2015

David Pinto

Bayesian Classification with Regularized Gaussian Models

Proposta de dissertação apresentada ao Curso de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica.

Universidade Federal de Minas Gerais – UFMG

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica – PPGEE

Supervisor: André Paim Lemos

Belo Horizonte, MG, Brasil

Novembro de 2015

David Pinto
Bayesian Classification with
Regularized Gaussian Models/ David Pinto. – Belo Horizonte, MG, Brasil, Novembro de 2015-
148 p. : il. (algumas color.) ; 30 cm.

Supervisor: André Paim Lemos

Dissertação (Mestrado) – Universidade Federal de Minas Gerais – UFMG
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica – PPGEE, Novembro de 2015.
1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 02:141:005.7

David Pinto

Bayesian Classification with Regularized Gaussian Models

Proposta de dissertação apresentada ao Curso de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica.

Trabalho aprovado. Belo Horizonte, MG, Brasil, 27 de novembro de 2015:

Belo Horizonte, MG, Brasil
Novembro de 2015

This work is dedicated to the women who are my source of power and inspiration in life: my mother Silvia, my girlfriend Marina, my sister Rhaiza, and my little niece Luiza.

Acknowledgements

This work would not have been possible without the help of many people. I would like to thank

- My adviser, Andre Lemos, for encouraging me to get my master's degree as well as for supporting me all the time;
- My girlfriend, Marina, for being incredibly supportive throughout these two years;
- My mother Silvia, the greatest teacher of all, for the countless advices;
- My sister Rhaiza and my little niece Luiza, for their love and inspiration;
- And the Nexer Labs team, Danilo Mattos, Ruhan Bidart and Sadraque Viana, for the endless support and for being great friends.

“A decision was wise, even though it led to disastrous consequences, if the evidence at hand indicated it was the best one to make; and a decision was foolish, even though it led to the happiest possible consequences, if it was unreasonable to expect those consequences.”

(Herodotus, 500 B.C.)

Abstract

This dissertation explores the use of covariance matrix regularization techniques in modeling the Gaussian Bayes classifier. The classical sample estimator is unstable in high dimensions due to the large number of parameters to be fitted, and is subjected to singularity. In order to guarantee stability and robustness of the classifier in high dimensions, positive definite and well conditioned covariance matrix estimates are needed. The naive Bayes classifier assumes conditional independence between variables and therefore, drastically reduces the number of parameters to be fitted. This classifier is stable in high dimensions. However, the conditional independence assumption is frequently violated, leading to poor generalization performance. This work provides a thorough overview of modern regularization frameworks to the sample covariance estimator, focusing on classification problems. The proposed Regularized Gaussian Bayes classifier reduces the number of covariance matrix parameters without disconsidering the correlation structure among variables. The covariance estimates are guaranteed to be positive definite and well conditioned, and the generalization performance of the classifier is significantly higher than that of the naive Bayes model. Finally, the adaptive boosting technique is applied to improve the predictive performance of the proposed method. The performance of the boosted classifier is equivalent to that of the state-of-the-art methods, *Extra Trees*, *Gradient Boosting Machines* and *Random Forests*.

Key-words: multivariate normal, covariance matrix, precision matrix, positive-definiteness, condition number, high-dimension, regularization, well-conditioned estimator, shrinkage estimator, probabilistic PCA, parsimony, sparse PCA, Cholesky decomposition, penalized likelihood, graphical lasso, banding, tapering, thresholding, Bayes classifier, naive Bayes, adaptive boosting, out-of-bag estimation, probability calibration.

Resumo

Esta dissertação explora o uso de técnicas de regularização da matriz de covariância na modelagem do classificador de Bayes Gaussiano. O estimador amostral clássico é instável em altas dimensões devido à grande quantidade de parâmetros a serem estimados, e está sujeito à singularidade. Para garantir a estabilidade e a robustez do classificador em altas dimensões, são necessárias estimativas definidas positivas e bem condicionadas da matriz de covariância. O classificador *naive* Bayes assume independência condicional entre as variáveis e, por isso, reduz drasticamente o número de parâmetros a serem estimados. Tal classificador é estável em altas dimensões. No entanto, a premissa de independência é geralmente violada, levando a baixa performance de generalização. O presente trabalho faz uma ampla revisão dos *frameworks* modernos de regularização do estimador amostral e os aplica na modelagem de problemas de classificação. O classificador de Bayes Gaussiano Regularizado proposto aqui reduz o número de parâmetros da matriz de covariância sem desprezar a estrutura de correlação existente entre as variáveis. As estimativas de covariância são garantidamente positivas definidas e bem condicionadas, e a performance de generalização do classificador é significativamente superior à do modelo *naive* Bayes. Por fim, a técnica de *adptive boosting* é aplicada para melhorar a performance preditiva do método proposto. O desempenho do classificador regularizado com *boosting* é equivalente ao dos métodos estado-da-arte, *Extra Trees*, *Gradient Boosting Machines* e *Random Forests*.

Palavras-chave: normal multivariada, matriz de covariância, matriz de precisão, definição positiva, número de condição, alta dimensão, regularização, estimador bem condicionado, estimador restrito, PCA probabilístico, parcimônia, PCA esparso, decomposição de Cholesky, verossimilhança penalizada, *graphical lasso*, *banding*, *tapering*, *thresholding*, classificador de Bayes, *naive* Bayes, *adaptive boosting*, estimativa *out-of-bag*, calibração de probabilidade.

List of Figures

List of Tables

List of abbreviations and acronyms

SVD	Singular Value Decomposition
PCA	Principal Component Analysis
PC	Principal Component
PPCA	Probabilistic Principal Component Analysis
LASSO	Least Absolute Shrinkage and Selection Operator
LARS	Least Angle Regression
SPCA	Sparse Principal Component Analysis
SPC	Sparse Principal Component
GLASSO	Graphical LASSO algorithm
UGB	Unrestricted Gaussian Bayes classifier
GNB	Gaussian Naive Bayes classifier
RGB	Regularized Gaussian Bayes classifier
BRGB	Boosted Regularized Gaussian Bayes classifier
MIC	Maximal Information Coefficient
GBM	Gradient Boosting Machine
CRAN	Comprehensive R Archive Network
ML	Maximum Likelihood
MLE	Maximum Likelihood Estimator
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
EMP	Empirical estimator
DIAG	Diagonal estimator with unequal variances
LW	Ledoit-Wolf estimator

OAS	Oracle-Approximating Shrinkage estimator
SS	Schafer-Strimmer estimator
PMD	Penalized Matrix Decomposition
QUIC	Quadratic Inverse Covariance
TIGER	Tuning-Insensitive Graph Estimation and Regression
BIC	Bayesian Information Criterion
CV	Cross-Validation
MAP	Maximum a Posteriori
LDA	Linear Discriminant Analysis
OOB	Out-of-Bag data
SVM	Support Vector Machine
PAV	Pair-Adjacent Violators

List of symbols

N	Number of data instances
p	Number of data variables
$\boldsymbol{X}$	Data matrix
$\boldsymbol{x}_n$	n^{th} data example
$\boldsymbol{\mu}$	Population vector of means
$\boldsymbol{\Sigma}$	Population covariance matrix
$\boldsymbol{\Sigma}^{-1}$, $\boldsymbol{\Omega}$	Population precision matrix
$\boldsymbol{S}$	Sample covariance matrix
$\boldsymbol{T}$	Shrinkage target covariance matrix
$\hat{\boldsymbol{\Sigma}}_{\text{O}}$	Oracle estimate of covariance matrix
α	Shrinkage/regularization tuning parameter
λ_i	i^{th} eigenvalue
q	Dimensionality of the principal component subspace
ℓ_1	LASSO penalty
ℓ_2	Ridge penalty
$\boldsymbol{L}_1$	Entropy loss
$\boldsymbol{L}_2$	Quadratic loss
$\boldsymbol{L}_3$	Modified Frobenius norm loss

Contents

1 Introduction

1.1 Covariance Matrix Estimation in High Dimensions

The covariance matrix plays a central role in many classical and modern data analysis techniques, such as feature extraction by principal component analysis (JOLLIFFE, 2002), classification by linear discriminant analysis (DUDA; HART; STORK, 2001), time series analysis (HAMILTON, 1994), spatial data analysis (CRESSIE, 1993), longitudinal data analysis (LINDSEY, 1995), multivariate Gaussian mixture models (BISHOP, 2006), Gaussian graphical models (YUAN; LIN, 2007), Kalman filtering (HAYKIN, 2004), portfolio selection (GOLDFARB; IYENGAR, 2003; LEDOIT; WOLF, 2003), quantitative genetics (LYNCH; WALSH, 1998) and many others. Some of them require estimation of the eigenstructure of the covariance matrix while others require estimation of the precision or concentration matrix (the inverse of the covariance matrix).

The sample (or empirical) covariance matrix has been the undisputed estimator in the literature, particularly when dealing with a large number of instances and a small number of variables. But it is well known that the empirical covariance matrix $\boldsymbol{S}$ for samples of size N from a p-variate Gaussian distribution, $\mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, has a number of undesirable properties when its dimension p is large. At first, as demonstrated by J. Hartlap, P. Simon and P. Schneider (2007), for $p > N - 1$ the sample covariance matrix is singular, rank deficient, and not positive-definite, so that its inverse does not exist. Secondly, as observed by Stein (1956), when p is close to N, the sample covariance matrix, though unbiased and positive-definite, distorts the eigenstructure of the population covariance matrix $\boldsymbol{\Sigma}$. Sampling variation causes the largest eigenvalues of $\boldsymbol{\Sigma}$ to be biased upward (overestimated) and the smallest eigenvalues to be biased downward (underestimated), while their mean is expected to be unbiased (JOHNSTONE, 2001). Thus, the sample covariance matrix condition-number tends to infinity (LEDOIT; WOLF, 2004). Finally, when p is comparable to N, the sample covariance matrix inverse is a poor estimator for $\boldsymbol{\Sigma}^{-1}$. According to Anderson (2003), under normality assumption, the expected value for the inverse is:

$$E[\boldsymbol{S}^{-1}] = \frac{N}{N-p-2}\boldsymbol{\Sigma}^{-1}, \text{ for } p < N-1 \tag{1.1}$$

So, when p is close to N, even though $\boldsymbol{S}$ is unbiased for $\boldsymbol{\Sigma}$, $\boldsymbol{S}^{-1}$ is highly biased for $\boldsymbol{\Sigma}^{-1}$.

1.2 Regularization of the Sample Covariance Matrix

Nowadays, it is common in many application areas to deal with the so-called "large p, small N" problems, where the number of instances can be very small relative to the dimension ($p \gg N$). Examples include microarray data, spectroscopy, neuroimaging, finance, climate studies and ecological data. In these cases, the use of the sample covariance matrix is problematic (STEIN, 1956). Since the pioneer work developed by James and Stein (1961), in order to overcome the curse of dimensionality (BELLMAN; BELLMAN, 1961), many authors have explored improved estimators by regularizing the covariance matrix, its inverse, and factors of various decompositions (see Appendix B) such as spectral, SVD, Cholesky, and variance-correlation; see Bickel and Levina (2008b), Pourahmadi (2011), Pourahmadi (2013) for a detailed review. Three broad categories of covariance estimators have emerged: those that regularize the eigenvalues of the sample covariance matrix, those that regularize the eigenvectors, and those that regularize both eigenvalues and eigenvectors.

The first category comprises Stein's family of shrinkage estimators, which focuses on improving the eigenstructure estimation by shrinking the eigenvalues of $\boldsymbol{S}$ toward a central value. Ledoit and Wolf (2003) proposed a ridge like estimator (HOERL; KENNARD, 1970) considering Steinian shrinkage toward the isotropic estimator (diagonal with common variances). The Ledoit-Wolf estimator achieves a reasonable tradeoff between low bias and low variance by shrinking the unstructured and unbiased sample estimate $\boldsymbol{S}$ towards a naive biased but well-conditioned target estimate $\boldsymbol{T}$. It is asymptotically the optimal convex combination of $\boldsymbol{S}$ and $\boldsymbol{T}$ with respect to the Frobenius norm loss function (LEDOIT; WOLF, 2004):

$$L(\hat{\boldsymbol{\Sigma}}, \boldsymbol{\Sigma}) = p^{-1}\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}\|^2 = p^{-1}\mathrm{tr}(\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma})^2 \tag{1.2}$$

resulting in the following class of positive-definite and well-conditioned estimators:

$$\hat{\boldsymbol{\Sigma}} = (1-\alpha)\boldsymbol{S} + \alpha\boldsymbol{T} \tag{1.3}$$

where $\boldsymbol{T} = p^{-1}\mathrm{tr}(\boldsymbol{S})\boldsymbol{I}$ and α is the shrinkage factor. The Ledoit-Wolf shrinkage estimator subsumes and improves many earlier Steinian and Bayesian estimators (CHEN et al., 2010).

A common framework for covariance matrix reparameterization and structure definition is the classical factor analysis model (ANDERSON, 2003), which allows modelling of a high-dimensional non-diagonal covariance matrix with a low number of parameters. The factor analysis provides parsimonious covariance structures by finding a few common (latent or observable) factors that can explain the dependence in the data. It may lead to a considerable reduction in the number of covariance parameters. A widely used approach for factor model learning involves the application of principal component analysis (PCA) (JOLLIFFE, 2002). PCA seeks a linear projection of the data on a new system of orthogonal vectors that maximizes the sample variance. These vectors are usually named *principal component loadings*, and corresponds to the eigenvectors of the sample covariance matrix. Each corresponding transformed data variable, called *principal component* (PC), will be a linear combination of all the original variables. The amount of sample variance explained by each PC is given by the associated sample covariance eigenvalue. By keeping a few PCs with the largest sample variance, PCA allows representing the data in a lower-dimensional space.

Using a latent factor framework, Tipping and Bishop (1999) derived a probabilistic formulation of PCA which efficiently computes covariance parameters that maximize the likelihood of the training data. Fan, Fan and Lv (2008) considered factor analysis to achieve a sparse representation of the covariance matrix in high dimensions, thus reducing the number of effective parameters in estimation. Mcnicholas and Murphy (2008), in an attempt to overcome the curse of dimensionality in Gaussian mixture modelling, introduced a class of eight parsimonious covariance structures based on a latent Gaussian model closely related to the factor analysis model. Kao and Roy (2013), motivated by the fact that imposing a hard constraint on the number of factors may not lead to an optimal estimate (FAN; FAN; LV, 2008), imposed sparsity on the covariance via trace penalization of the sample estimator eigendecomposition instead of constraining the number of eigenvalues and eigenvectors retained in a PCA model.

However, in high-dimensional settings when $p >> N$, it was shown by Paul (2007) followed by Johnstone and Lu (2009) that classical PCA has a critical drawback: the sample eigenvectors are not consistent estimators of their population counterparts, and provide wrong PC directions. Recently, many authors have proposed to regularize the sample covariance eigenvectors by considering only a subset of the original variables to find the PC directions, a method termed *sparse* PCA (JOLLIFFE; TRENDAFILOV; UDDIN, 2003; ZOU; HASTIE; TIBSHIRANI, 2006; SHEN; HUANG, 2008; WITTEN; TIBSHIRANI; HASTIE, 2009). Sparse PCA has been shown to be consistent in high-dimensional settings where classical PCA is not (JOHNSTONE; LU, 2009). It seeks linear projections that maximize the sample variance, but limiting the number of non-zero elements in the projection vectors. An effective method for developing sparse PCA is to formulate PCA as a regression optimization problem (WITTEN; TIBSHIRANI, 2009), and then estimate sparse PC loadings by imposing the LASSO (TIBSHIRANI, 1996) or elastic net (ZOU; HASTIE, 2005) constraint on the regression coefficients.

Sparse PCA methods regularize the eigenvectors of the sample covariance matrix. In an attempt to regularize both the eigenvectors and eigenvalues, Wu and Pourahmadi (2003) proposed a regression-based derivation and interpretation of the modified Cholesky decomposition (SCHNABEL; ESKOW, 1999) of a precision matrix. Regularizing the Cholesky factor of the covariance via this regression interpretation always results in a positive definite estimator. The simplest way to introduce sparsity in the Cholesky factor is to estimate only its first k subdiagonals and set the remaining to zero (WU; POURAHMADI, 2003; BICKEL; LEVINA, 2008b). Huang et al. (2006) imposed ℓ_1 penalties on the Cholesky factor to achieve extra parsimony. Rothman, Levina and Zhu (2010) proposed a latent variable regression model to estimate the Cholesky factor of a covariance matrix instead of its inverse. However, the Cholesky-based regularization relies on the assumption that variables have a natural ordering. The Isomap method proposed by Wagaman and Levina (2009) tries to discover an order among the variables based on their correlations. But, in applications such as time series, longitudinal data and spectroscopy, the natural ordering present in the data leads to improved performance (ROTHMAN; LEVINA; ZHU, 2010).

In applications where the variables are not naturally ordered, covariance estimators are required to be invariant under variable permutation. Maximizing a penalized normal likelihood, where a ℓ_1 penalty is added to the likelihood, does have this property. The resulting sparse covariance estimators are based on simultaneous transformation of all entries of the sample covariance matrix, and, besides being guaranteed to be positive definite, lead to regularizing both the eigenvalues and eigenvectors. These penalized likelihood estimators (YUAN; LIN, 2007; BANERJEE; GHAOUI; D'ASPREMONT, 2008; FRIEDMAN; HASTIE; TIBSHIRANI, 2008; ROTHMAN et al., 2008) are inspired by the regression-based interpretation of the entries of the precision matrix proposed by Meinshausen and Bühlmann (2006). The graphical LASSO method of Friedman, Hastie and Tibshirani (2008) is the fastest available algorithm to solve the sparse estimation of a precision matrix. A Bayesian approach for introducing sparsity in the inverse via a sparse prior was proposed by Wong, Carter and Kohn (2003).

Another large class of estimators comprises elementwise regularization, which shrinks individual entries of the sample covariance matrix and regularizes both eigenvalues and eigenvectors. The most common componentwise estimators are banding (BICKEL; LEVINA, 2004; BICKEL; LEVINA, 2008b; WU; POURAHMADI, 2003), tapering (FURRER; BENGTSSON, 2007) and thresholding (BICKEL; LEVINA, 2008a; KAROUI, 2008; ROTHMAN; LEVINA; ZHU, 2009). Banding starts estimating the covariance matrix by a diagonal matrix and then successively estimates and adds the other subdiagonals, until the optimal k number of subdiagonals has been included. Tapering is a smooth version of banding where it gradually shrinks the off-diagonal entries toward zero. Like Cholesky-based regularization, banding and tapering require a natural ordering among the variables and assume that those farther apart in the ordering are less correlated. Thresholding the sample covariance matrix does not have this premise and can be used in applications where the variables are not ordered. Choosing the threshold parameter λ is the key in implementing it. Since the entries of the sample covariance matrix have different variability, adaptive thresholding (CAI; LIU, 2011) where each entry has a different threshold parameter seems more appropriate in practice.

1.3 Gaussian Bayes Classifier in High Dimensions

In classification problems, the classical Gaussian Bayes classifier can be used if $N > p$. However, when $p > N$, regularization of the within-class covariance matrix is necessary to avoid overfitting (MOORE, 2001). The naive Bayes classifier reduces model complexity assuming independence among variables given the class label, and performs well in "large p, small N" problems (BICKEL; LEVINA, 2004). In the context of Gaussian models, it is equivalent to modelling the within-class instances using a multivariate Normal distribution with diagonal covariance matrix containing the individual attribute variances (DEMPSTER; LAIRD; RUBIN, 1977). Despite its surprisingly good performance in classification, evidence has been found that naive Bayes produces poor probability estimates (BENNETT, 2000).

Several modifications have already been proposed to improve naive Bayesian learning by relaxing the conditional attribute independence assumption (KOHAVI, 1996; FRIEDMAN; GEIGER; GOLDSZMIDT, 1997; KEOGH; PAZZANI, 1999; ZHENG; WEBB, 2000; FRANK; HALL; PFAHRINGER, 2003), transforming the original variables or selecting a subset of them (KONONENKO, 1991; LANGLEY; SAGE, 1994; PAZZANI, 1998; BOULLÉ, 2007; ZHANG; nA; ROBLES, 2009), and weighting the attributes (FERREIRA; DENISON; HAND, 2001; HALL, 2007; LEE; GUTIERREZ; DOU, 2011; ZAIDI et al., 2013). The work presented here addresses attribute independence violations in the Gaussian naive Bayes classifier by introducing regularized covariance estimates, which take into account the correlation structure between attributes. Thus, the proposed Regularized Gaussian Bayes (RGB) classifier optimizes the tradeoff between the "non-naivety" and prediction smoothing, and performs well in high-dimensional settings.

Moreover, improvements in RGB accuracy and stability are achieved using Adaptive Boosting (AdaBoost) (FREUND; SCHAPIRE et al., 1996; FREUND; SCHAPIRE, 1997), a boosting ensemble method resistant to overfitting (FRIEDMAN et al., 2000). In short, the proposed Boosted RGB (BRGB) classifier generates a sequentially weighted set of RGB base classifiers that are combined to form a robust classifier. Classification experiments have demonstrated that the BRGB achieves prediction performance comparable to the best off-the-shelf ensemble based architectures, such as Random Forests (RFs) (BREIMAN, 2001), Extremely Randomized Trees (ExtraTrees) (GEURTS; ERNST; WEHENKEL, 2006) and Gradient Boosting Machines (GBMs) (FRIEDMAN, 2001; FRIEDMAN, 2002), using few (10 to 20) base classifiers.

1.4 Organization

The remainder of this dissertation is organized as follows. In Chapter 2 the regularization frameworks for the sample covariance matrix are formally defined and statistically compared in terms of estimation accuracy and computational performance. In addition, appropriate methods to select the tuning parameters of these estimators are reviewed. In Chapter 3 the regularized covariance matrix estimators are applied to prevent overfitting and instability in the Gaussian Bayes classifier, particularly in high-dimensional data settings. Adaptive boosting is then introduced to improve classification accuracy and probability estimation of the proposed RGB classifier. The RGB and BRGB classifiers are then evaluated in Chapter 4 with synthetic and real world datasets, and the conclusions and future directions for extension of this work are presented in Chapter 5.

All experiments, analyses and statistical programming were conducted with the **R statistical software** (R Core Team, 2015). Because `R` is based on an open-source model, a large user-community contributes functions and data through add-on packages, which are available in the Comprehensive `R` Archive Network (CRAN) repository. Throughout the chapters, the description of the techniques covered in this dissertation is often supplemented by recommendations of the most appropriate `R` functions and packages to deal with them.

2 Improved Covariance Matrix Estimators

In this chapter, some properties and drawbacks of the empirical covariance matrix estimator are reviewed and the need for regularization in the high-dimensional context is pointed out. Then, a thorough overview of modern shrinkage and regularization techniques for covariance estimation is provided, followed by computer simulations to show that using these well-conditioned estimators is highly advantageous in terms of statistical risk and other performance criteria.

2.1 The Sample Covariance Matrix

Let $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_p\}$ be random variables of a p-variate Gaussian distributed population centered in $\boldsymbol{\mu} = \{\mu_1 = \boldsymbol{E}(\boldsymbol{x}_1), \dots, \mu_p = \boldsymbol{E}(\boldsymbol{x}_p)\}$. The population covariance matrix is:

$$\boldsymbol{\Sigma} = (\sigma_{i,j})_{1 \le i,j \le p} \tag{2.1}$$

where $\sigma_{i,j} = \mathrm{cov}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{E}(\boldsymbol{x}_i \boldsymbol{x}_j) - \boldsymbol{E}(\boldsymbol{x}_i)\boldsymbol{E}(\boldsymbol{x}_j)$, $1 \le i, j \le p$, is the covariance function. Given a random sample $\{\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N\}, \boldsymbol{x}_i \in \mathbb{R}^p$ of size N from this population, the sample covariance matrix is defined as

$$\boldsymbol{S} = \frac{1}{N-1} \sum_{i=1}^{N} (\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T \tag{2.2}$$

which is a unbiased estimate of $\boldsymbol{\Sigma}$, and the maximum likelihood estimator of $\boldsymbol{\Sigma}$ is

$$\boldsymbol{S}^{ML} = (\sigma_{i,j}^{ML})_{1 \le i,j \le p} = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T = \frac{N-1}{N} \boldsymbol{S} \tag{2.3}$$

where $\overline{\boldsymbol{x}} = \{N^{-1} \sum_{i=1}^N \boldsymbol{x}_{i,1}, \dots, N^{-1} \sum_{i=1}^N \boldsymbol{x}_{i,p}\}$ is the sample mean vector. These two estimators are similar for large N, and are reliable and undisputed estimators of the population covariance matrix when the number of variables p is small relative to the sample size N (ANDERSON, 2003). However, as noted by Stein (1956), both $\boldsymbol{S}$ and $\boldsymbol{S}^{ML}$ exhibit serious deficiencies in the “large p, small N” data setting. In particular, in this case the sample eigenstructure (see Section B.2 of Appendix B) tends to be systematically distorted, resulting in numerically ill-conditioned estimates of $\boldsymbol{\Sigma}$. This phenomenon, so-called “Stein phenomenon”, is illustrated in Figure 2.1 where the population covariance matrix is taken to be the identity matrix with all its eigenvalues equal to one. It is evident by inspection that the range of the sample eigenvalues gets larger for larger values of the ratio p/N, and some eigenvalues go to zero when $p > n$. A covariance matrix estimate will be positive definite and thus of full-rank and invertible if and only if all its eigenvalues are greater than

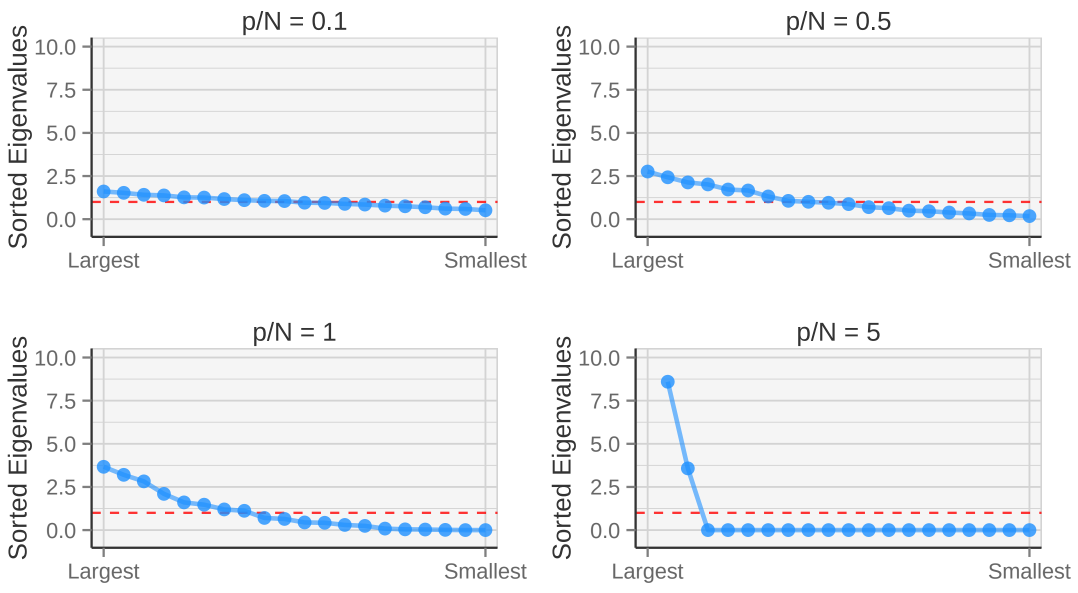


Figure 2.1 – Distribution of the sample eigenvalues against the true eigenvalues (red dashed line) for different values of the ratio p/N.

zero. Moreover, it will be well-conditioned, which means that inverting it will not amplify the estimation error, if and only if its condition number (the ratio of its maximum and minimum eigenvalue) is not much larger than one (ABABOU; BAGTZOGLOU; WOOD, 1994).

A naive strategy to obtain a positive definite estimator of the covariance is to take the sample covariance matrix $\boldsymbol{S}$ and apply, for example, the algorithm proposed by Higham (1988), which computes the nearest symmetric positive definite matrix to $\boldsymbol{S}$ in the Frobenius norm. This will adjust all eigenvalues to be larger than some prespecified threshold and thus guarantee positive definiteness. However, the resulting matrix will not be well conditioned (SCHÄFER; STRIMMER, 2005).

Regularized and well-conditioned covariance estimates are usually obtained by minimizing certain risk functions. The two commonly loss functions considered in the literature are (YANG; BERGER, 1994)

$$\boldsymbol{L}_1(\hat{\boldsymbol{\Sigma}},\boldsymbol{\Sigma}) = \mathrm{tr}(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Sigma}^{-1}) - \log|\hat{\boldsymbol{\Sigma}}\boldsymbol{\Sigma}^{-1}| - p, \tag{2.4}$$

$$\boldsymbol{L}_2(\hat{\boldsymbol{\Sigma}},\boldsymbol{\Sigma}) = \mathrm{tr}(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Sigma}^{-1} - \boldsymbol{I})^2 \tag{2.5}$$

where $\hat{\boldsymbol{\Sigma}}$ is an arbitrary estimator. Given a loss function $\boldsymbol{L}_i$, the risk of the estimator $\hat{\boldsymbol{\Sigma}}$ is defined by the expected loss

$$\boldsymbol{R}_i(\hat{\boldsymbol{\Sigma}},\boldsymbol{\Sigma}) = \boldsymbol{E}[\boldsymbol{L}_i(\hat{\boldsymbol{\Sigma}},\boldsymbol{\Sigma})],\ i = 1,2 \tag{2.6}$$

An estimator $\hat{\boldsymbol{\Sigma}}$ is considered better than the sample covariance matrix $\boldsymbol{S}$, if its risk function is smaller than that of $\boldsymbol{S}$ (POURAHMADI, 2011). The entropy loss function $\boldsymbol{L}_1$ (STEIN, 1956), also known as Stein's loss function, can be seen as the Kullback-Leibler divergence between two multivariate normal distributions corresponding to the two covariance matrices. The second, called the quadratic loss function (JAMES; STEIN, 1961), is the Euclidean or the Frobenius norm of its matrix argument, which involves squaring the difference between entries of the estimator and the target. Consequently, it penalizes overestimates more than underestimates, while downward biased estimates are less favored under $\boldsymbol{L}_1$ (POURAHMADI, 2013). For example, among all scalar multiples $c\boldsymbol{S}$ of the sample covariance matrix, the usual unbiased estimator $\boldsymbol{S}$ minimizes the $\boldsymbol{R}_1$ risk, while $N\boldsymbol{S}/(N+p+1)$ yields the minimum risk estimator under loss function $\boldsymbol{L}_2$ (HAFF, 1980).

2.2 Ledoit-Wolf Shrinkage Estimator

To ensure positive-definiteness and well-condition of the estimated covariance matrix for moderate and large dimensionality, Ledoit and Wolf (2004) proposed a ridge-type shrinkage estimator of the form

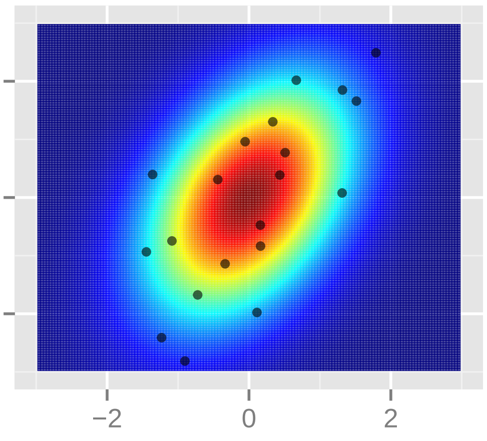
$$\hat{\boldsymbol{\Sigma}}_{LW} = \alpha\boldsymbol{T} + (1-\alpha)\boldsymbol{S},\ \boldsymbol{T} = p^{-1}\mathrm{tr}(\boldsymbol{S})\boldsymbol{I} \tag{2.7}$$

which is asymptotically the optimal convex linear combination of the sample covariance with the identity matrix. The shrinkage intensity parameter $\alpha \in [0,1]$ is chosen to minimize the risk function corresponding to a slightly modified Frobenius norm loss function for high-dimensional data

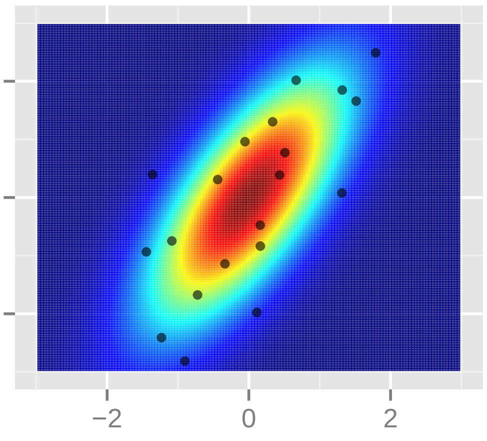
$$\boldsymbol{L}_3 = p^{-1}\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}\|_F^2 = p^{-1}\mathrm{tr}(\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma})^2 \tag{2.8}$$

which does not involve matrix inversion, preventing high computational costs when $p \gg N$. The heuristics behind this loss function are the same as those for $\boldsymbol{L}_2$. However, as evidenced by Warton (2008), the optimal covariance estimator under $\boldsymbol{L}_3$ is the maximum penalized likelihood estimator with penalty term proportional to $-\mathrm{tr}(\boldsymbol{\Sigma}^{-1})$ (the sum of inverse eigenvalues). Since the penalty function becomes large when $\boldsymbol{\Sigma}$ gets closer to singularity, such a penalty forces the covariance estimate to be positive-definite and better conditioned (POURAHMADI, 2013).

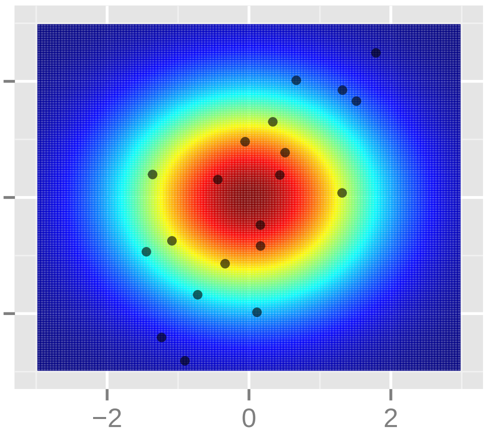
The Ledoit and Wolf (2004) shrinkage estimator can be seen as a weighted average of a variance term and a bias term, where the weights are chosen to optimize the bias-variance tradeoff, and consequently minimizes the mean squared error (MSE). Since the unconstrained estimate $\boldsymbol{S}$ is unbiased for $\boldsymbol{\Sigma}$, but unstable with high variance due to the large number of parameters that need to be fitted ($\frac{p(p+1)}{2}$), whereas the constrained target



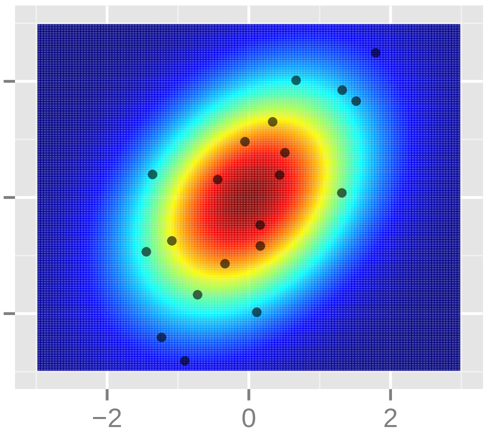
(a) True covariance matrix $\mathbf{\Sigma}$



(b) Sample covariance estimator $\boldsymbol{S}$ ($\|\hat{\mathbf{\Sigma}} - \mathbf{\Sigma}\|_F^2 = 0.58$)



(c) Isotropic covariance estimator $\boldsymbol{T}$ ($\|\hat{\mathbf{\Sigma}} - \mathbf{\Sigma}\|_F^2 = 1.18$)



(d) Ledoit-Wolf covariance estimator $\mathbf{\Sigma}_{LW} = 0.53\boldsymbol{S} + 0.47\boldsymbol{T}$ ($\|\hat{\mathbf{\Sigma}} - \mathbf{\Sigma}\|_F^2 = 0.08$)

Figure 2.2 – Density estimation of a bivariate Gaussian distribution from 10 training observations using, respectively, the sample, isotropic and shrinkage covariance estimators.

estimator $\boldsymbol{T}$ has very little variability (only one parameter to be fitted), but can be severely biased, a reasonable tradeoff between low bias and low variance is achieved by shrinking $\boldsymbol{S}$ towards $\boldsymbol{T}$. A key question in this procedure is how to select an optimal value for the shrinkage parameter to obtain a regularized estimate $\hat{\mathbf{\Sigma}}_{LW}$ that outperforms the individual estimators $\boldsymbol{S}$ and $\boldsymbol{T}$ in terms of statistical risk. The shrinkage effect on density estimation is illustrated in Figure 2.2, where the Ledoit-Wolf estimator achieves the lowest $\boldsymbol{L}_3$ loss.

The oracle estimate $\hat{\mathbf{\Sigma}}_O$ which minimizes the MSE, or the $\boldsymbol{R}_3$ risk, is the solution to

$$\arg\min_{\alpha} \boldsymbol{E}[\|\hat{\mathbf{\Sigma}}_O - \mathbf{\Sigma}\|_F^2],\ \hat{\mathbf{\Sigma}}_O = (1-\alpha)\boldsymbol{S} + \alpha\boldsymbol{T} \tag{2.9}$$

where the optimal parameter α_O, under the assumption that $\boldsymbol{S}$ is the sample covariance of a set of p-dimensional i.i.d Gaussian vectors $\{\boldsymbol{x}_i\}_{i=1}^N$ with covariance $\boldsymbol{\Sigma}$, is given by (CHEN et al., 2010):

$$\alpha_O = \frac{(1-2/p)\mathrm{tr}(\boldsymbol{\Sigma}^2) + \mathrm{tr}(\boldsymbol{\Sigma})^2}{(N+1-2/p)\mathrm{tr}(\boldsymbol{\Sigma}^2) + (1-N/p)\mathrm{tr}(\boldsymbol{\Sigma})^2} \tag{2.10}$$

Despite being optimal, the oracle estimator cannot be implemented, since the solution specified in (2.10) depends on the unknown true covariance matrix $\boldsymbol{\Sigma}$. Without assuming any sample distribution, Ledoit and Wolf (2004) proposed to approximate the oracle using the following N-consistent estimator:

$$\alpha_{LW} = \frac{\sum_{j=1}^{p}\sum_{i=1}^{p}\left[(\boldsymbol{Y}^T\boldsymbol{Y})/(N-1) - \boldsymbol{S}^2\right]_{ij}}{\|\boldsymbol{S}-\boldsymbol{T}\|_F^2} \tag{2.11}$$

where the $N \times p$ matrix $\boldsymbol{Y} = \boldsymbol{X}^2$ is obtained by squaring the training data $\boldsymbol{X}$, provided that all columns of $\boldsymbol{X}$ have been previously centered to zero mean. The estimator $\hat{\boldsymbol{\Sigma}}_{LW}$ defined by plugging α_{LW} into (2.7) is distribution-free and not restricted to Gaussian assumptions, and asymptotically minimizes the MSE. Ledoit and Wolf (2004) also showed that the optimal shrinkage intensity α_O is always between 0 and 1. To guarantee this constraint, they suggested using a truncated estimate

$$\alpha_{LW}^* = \min(\alpha_{LW}, 1) \tag{2.12}$$

Under Gaussian assumption, Chen et al. (2010) showed that the LW estimation can be improved using the Rao-Blackwell theorem (TREES, 2004). They proposed an alternative estimator for (2.10) that is a function of only $\boldsymbol{S}$, the Oracle-Approximating Shrinkage (OAS) estimator. The resulting covariance estimate $\hat{\boldsymbol{\Sigma}}_{OAS}$ is obtained by plugging the following tuning parameter into (2.7)

$$\alpha_{OAS} = \min\left(\frac{\overline{s} - p^{-2}\mathrm{tr}(\boldsymbol{S})^2}{(N+1)[\overline{s} - p^{-3}\mathrm{tr}(\boldsymbol{S})^2]}, 1\right), \ \overline{s} = p^{-2}\sum_{j=1}^{p}\sum_{i=1}^{p}[\boldsymbol{S}^2]_{ij} \tag{2.13}$$

Chen et al. (2010) empirically shown that when N is small, the OAS significantly outperforms the LW in terms of MSE. This behaviour is illustrated in Figure 2.3, which compares the MSE performance of both LW and OAS shrinkage estimators on the task of estimating a 100-variate covariance matrix from 20 i.i.d Gaussian observations. The true covariance matrix has been generated using the methodology proposed by Joe (2006), which first randomly generates p eigenvalues in $[1, 10]$, then uses columns of a randomly generated orthogonal matrix as eigenvectors. This algorithm is implemented in the `R` package `clusterGeneration` (QIU; JOE., 2015).

Another widely used but computationally intensive approach to infer the shrinkage intensity α is cross-validation. Warton (2008) proposed the maximizer of the normal

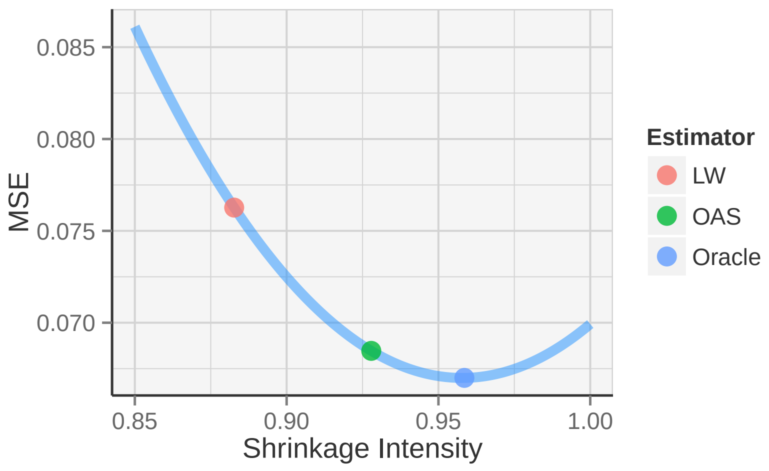


Figure 2.3 – Performance of the LW and OAS estimators in terms of MSE against the oracle estimator for $p/N = 5$.

likelihood estimated by K-fold cross-validation (FRIEDMAN; HASTIE; TIBSHIRANI, 2001) as an estimator of the the penalty parameter α. In this procedure the data is randomly partitioned into K parts of roughly equal size $\mathcal{A}_1, \dots, \mathcal{A}_K$, with N_k observations in the k^{th} subset. In a first step, the group $\mathcal{A}_1$ is retained as the validation data, and the remaining $K-1$ ones are used as training data to estimate $\hat{\boldsymbol{\mu}}^{(1)}$ and $\hat{\boldsymbol{\Sigma}}_\alpha^{(1)}$. The held-out samples are used to compute the observed likelihood $L(\hat{\boldsymbol{\mu}}^{(1)}, \hat{\boldsymbol{\Sigma}}_\alpha^{(1)}, \mathcal{A}_1)$. Then the first subset is returned to the training set and procedure repeats with the second subset held out, and so on. Finally, the shrinkage parameter α is taken to maximize the cross-validate likelihood function

$$\hat{\alpha} = \arg\max_{\alpha} \sum_{k=1}^{K} \log L(\hat{\boldsymbol{\mu}}^{(k)}, \hat{\boldsymbol{\Sigma}}_\alpha^{(k)}, \mathcal{A}_k) \tag{2.14}$$

where

$$\begin{aligned} -2\log L(\hat{\boldsymbol{\mu}}^{(k)}, \hat{\boldsymbol{\Sigma}}_\alpha^{(k)}, \mathcal{A}_k) = N_k p \log(2\pi) + N_k \log|\hat{\boldsymbol{\Sigma}}_\alpha^{(k)}| + \\ \operatorname{tr}\left\{(\mathcal{A}_k - \hat{\boldsymbol{\mu}}^{(k)})(\hat{\boldsymbol{\Sigma}}_\alpha^{(k)})^{-1}(\mathcal{A}_k - \hat{\boldsymbol{\mu}}^{(k)})^T\right\} \end{aligned} \tag{2.15}$$

This cross-validation procedure is provided by the `R` package `mvabund` (WANG et al., 2015) through the function `ridgeParamEst`.

Bickel and Levina (2008b) proposed a random cross-validation procedure to select the tuning parameter b of the regularized banding estimator $\hat{\boldsymbol{\Sigma}}_b$. Here this procedure will be adopted to choose the shrinkage parameter α in (2.7). So, α will be selected to minimize the risk

$$\boldsymbol{R}(\alpha) = E[\|\hat{\boldsymbol{\Sigma}}_\alpha - \boldsymbol{\Sigma}\|_1] \tag{2.16}$$

where $\|\cdot\|_1$ is the ℓ_1 matrix norm. To estimate the risk and thus the "oracle" α, $\alpha_O = \arg\min_{\alpha} \boldsymbol{R}(\alpha)$, the resampling scheme proposed by Bickel and Levina (2008b) is considered: divide the original sample into two samples at random and use the sample covariance matrix of one sample as the target to choose the best α for the other sample. More precisely, let N_1 and $N_2 = N - N_1$ be the two sample sizes for the random split, and let $\boldsymbol{S}_1^{(i)}$ and $\boldsymbol{S}_2^{(i)}$ be the two sample covariance matrices from the i^{th} split, for $i = 1, \ldots, B$. Then the risk (2.16) can be estimated by

$$\hat{\boldsymbol{R}}(\alpha) = \frac{1}{B} \sum_{i=1}^{B} \| \left(\alpha \boldsymbol{T}_1^{(i)} + (1-\alpha)\boldsymbol{S}_1^{(i)} \right) - \boldsymbol{S}_2^{(i)} \|_1 \tag{2.17}$$

and α is selected as

$$\hat{\alpha} = \arg\min_{\alpha} \hat{\boldsymbol{R}}(\alpha) \tag{2.18}$$

In (BICKEL; LEVINA, 2008b) the authors suggest to use $N_1 = N/3$. The original algorithm for the choice of the banding parameter is provided by the `R` package `CVTuningCov` (WANG, 2014).

In order to compare the different approaches described so far to define the shrinkage intensity in (2.7), a 100-variate population covariance matrix with randomly assigned eigenvalues in $[1, 10]$ have been considered. The shrinkage estimators have been fitted over normally distributed observations $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ considering four values of the sample size $N = 20, 100, 200, 1000$. Table 2.1 shows average losses and standard deviations over 100 replications, as measured by the squared Frobenius matrix norm $p\boldsymbol{L}_3 = \|\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}\|_F^2$. The results suggest that, like found by Wasserman and Roeder (2009) in high-dimensional regression problems, the cross-validation leads to overfitting. On the other hand, choosing the regularization parameter analytically, besides not requiring computationally expensive procedures, proved to be more accurate in estimating the oracle α, with the OAS estimator achieving the best performance. The results also suggest that when N is small, the OAS outperforms the LW in terms of MSE.

Since its proposal, several improvements have been addressed to the Ledoit-Wolf shrinkage estimator (2.7) in an attempt to enhance its estimation accuracy. Fisher and Sun (2011) extended the Ledoit-Wolf estimator to other target matrices and proposed new optimal shrinkage intensities for them. The authors considered the optimal convex combination with three commonly used positive-definite and well-conditioned target matrices: the diagonal with unit variance ($\boldsymbol{T} = \boldsymbol{I}$), the diagonal with common variance

Table 2.1 – Averages and standard errors over 100 replications of performance measure for different estimators of the shrinkage intensity.

	Oracle	LW	OAS	5-fold CV	Random CV
$p/N = 5$					
α	0.97 (0)	0.87 (0.02)	0.92 (0.02)	0.92 (0.08)	0.98 (0.02)
$p\boldsymbol{L}_3$	578.73 (6.23)	763.45 (80.59)	622.92 (35.45)	744.08 (426.49)	**588.60 (10.35)**
$p/N = 1$					
α	0.87 (0)	0.84 (0.02)	0.85 (0.02)	0.88 (0.08)	0.97 (0.02)
$p\boldsymbol{L}_3$	511.68 (4.63)	515.15 (7.46)	**513.83 (6.29)**	541.38 (29.63)	566.77 (19.29)
$p/N = 0.5$					
α	0.74 (0)	0.73 (0.01)	0.73 (0.01)	0.83 (0.11)	0.97 (0.02)
$p\boldsymbol{L}_3$	453.46 (5.18)	454.01 (5.51)	**453.90 (5.45)**	501.22 (46.14)	577.86 (23.07)
$p/N = 0.1$					
α	0.27 (0)	0.26 (0.00)	0.26 (0.00)	0.52 (0.12)	0.84 (0.03)
$p\boldsymbol{L}_3$	190.77 (4.09)	**190.61 (4.16)**	**190.61 (4.16)**	273.54 (69.37)	528.91 (29.99)

($\boldsymbol{T} = p^{-1}\text{tr}\{\boldsymbol{S}\}\boldsymbol{I}$) and the diagonal with unequal variance ($\boldsymbol{T} = \text{diag}\{\boldsymbol{S}\}$). The `R` package `HiDimDA` provides all these estimators.

Focusing on the diagonal with unequal variance shrinkage target, Schäfer and Strimmer (2005) suggest an unbiased, rather than N-consistent, estimator for the penalty parameter. In this formulation, regularization is applied separately to the sample correlations r_{ij} rather than the entire covariances, shrinking them to zero. In a subsequent work Opgen-Rhein and Strimmer (2007) proposed to shrink the sample variances v_i as well, towards their median. The resulting covariance estimator $\hat{\boldsymbol{\Sigma}}_{SS}$ is always positive-definite and well-conditioned, and their elements are of the form $s_{ij}^{SS} = r_{ij}^{SS}\sqrt{v_i^{SS}v_j^{SS}}$ with $i, j = 1, \ldots, p$ and components

$$r_{ij}^{SS} = (1 - \hat{\alpha}_1)\, r_{ij}, \tag{2.19}$$

$$v_i^{SS} = \hat{\alpha}_2 \text{median}\,(\text{tr}\{\boldsymbol{S}\}) + (1 - \hat{\alpha}_2)\, v_i \tag{2.20}$$

The shrinkage intensities are estimated via

$$\hat{\alpha}_1 = \min\left(1, \frac{\sum_{i\neq j} \widehat{\text{var}}(r_{ij})}{\sum_{i\neq j} r_{ij}^2}\right), \tag{2.21}$$

$$\hat{\alpha}_2 = \min\left(1, \frac{\sum_{i=1}^{p} \widehat{\text{var}}(v_i)}{\sum_{i=1}^{p} (v_i - v_{\text{median}})^2}\right) \tag{2.22}$$

This Stein-type shrinkage estimator is available in the `R` package `corpcor` (SCHÄFER et al., 2014).

More recently, Donoho, Gavish and Johnstone (2013) studied a variety of loss functions obtained from or inspired by the literature, and derived a set of new optimal closed form eigenvalue shrinkers, resulting in relatively simple methods that can be effectively used for very large covariance matrices. Some of them are available in the recent `R` package `covmat` developed in the Google Summer of Code 2015 (GSoC 2015). Another recent contribution is given by Anestis Touloumis (2015) who developed nonparametric consistent estimators for the optimal shrinkage intensity considering three popular target matrices: the identity matrix, the diagonal matrix with equal variances and the diagonal matrix with unequal variances. The resulting covariance matrix estimators are available in the `R` package `ShrinkCovMat` (Anestis Touloumis, 2015).

2.3 Latent Factor Models and Probabilistic PCA

The factor analysis model (KNOTT; BARTHOLOMEW, 1999) assumes that a p-dimensional random vector $\boldsymbol{x}_i$ can be modelled using a q-dimensional vector of latent (or unobserved) factors $\boldsymbol{f}_i$, where $q \ll p$. The model can be written as

$$\boldsymbol{x}_i = \boldsymbol{W}\boldsymbol{f}_i + \boldsymbol{\mu} + \boldsymbol{\epsilon}_i \tag{2.23}$$

where $\boldsymbol{W}$ is a $p \times q$ matrix of unobserved factor loadings. Conventionally, the latent variables are defined to be independent and Gaussian with unit variance, so $\boldsymbol{f}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$. The noise model is also Gaussian such that $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Psi})$, where $\boldsymbol{\Psi}$ is a $p \times p$ diagonal matrix. The parameter $\boldsymbol{\mu}$ permits the data model to have non-zero mean. Given this formulation, the observation vectors are also normally distributed $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{C})$, where the model covariance is $\boldsymbol{C} = \boldsymbol{W}\boldsymbol{W}^T + \boldsymbol{\Psi}$.

Tipping and Bishop (1999) demonstrated that there is a link between latent factor models and principal component analysis (PCA), and for the case of isotropic noise $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$, the columns of the maximum-likelihood estimator $\boldsymbol{W}_{\text{ML}}$ span the principal subspace of the data even when $\boldsymbol{C}$ is different from the sample covariance $\boldsymbol{S}$. So, (2.23) can be efficiently computed using an analytical solution via PCA. Tipping and Bishop (1999) termed this model probabilistic PCA (PPCA), which represents covariance by

$$\boldsymbol{\Sigma}^q_{\text{PPCA}} = \boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I} \tag{2.24}$$

The parameters $\boldsymbol{W}$ and σ are defined by their maximum likelihood estimates

$$\begin{aligned} \boldsymbol{W}_{\text{ML}} &= \boldsymbol{Q}_q\left(\boldsymbol{\Lambda}_q - \sigma^2\boldsymbol{I}\right)^{1/2}, & (2.25)\\ \sigma^2_{\text{ML}} &= \frac{1}{p-q}\sum_{j=q+1}^{p}\lambda_j & (2.26) \end{aligned}$$

where the q column vectors in the $p\times q$ matrix $\boldsymbol{Q}_q$ are the principal eigenvectors of $\boldsymbol{S}$, with corresponding eigenvalues $\lambda_1, \ldots, \lambda_q$ in the $q\times q$ matrix $\boldsymbol{\Lambda}_q$. The noise variance estimate σ^2_{ML} has a clear interpretation as the variance lost in the projection, averaged over the lost dimensions. From (2.25), the latent factor model defined in (2.23) seeks a mapping from the latent space into the principal subspace of the observed data. The standard PCA can be seen as a zero-noise limit of PPCA, when $\sigma^2 \to 0$. The probabilistic PCA and other variations of the standard PCA are implemented in the `R` package `pcaMethods` (STACKLIES et al., 2007).

The PPCA model gives a rank-constrained representation of the covariance. In other words, it is the rank-q maximum likelihood estimate of covariance, with an equality constraint among the $p - q$ smallest eigenvalues, which are set to their average. So, the eigenvalues of $\boldsymbol{\Sigma}^q_{\mathrm{PPCA}}$ become $[\lambda_1, \ldots, \lambda_q, \sigma^2_{\mathrm{ML}}, \ldots, \sigma^2_{\mathrm{ML}}]$. PPCA can thus be defined as a constrained (or parsimonious) Gaussian model, with few parameters. While the sample covariance matrix has $p(p+1)/2$ parameters, for PPCA there are only $pq + 1$ parameters in the covariance representation. The diagonal covariance model has an even smaller structure, with p parameters, but cannot capture correlations. On the other hand, PPCA can represent q most significant correlations.

Using a generalized representation for the factor model noise covariance $\boldsymbol{\Psi}$, Mcnicholas and Murphy (2008) developed a collection of eight parsimonious covariance structures applied to Gaussian mixture models. These covariance structures can have as few as $pq - q(q-1)/2 + 1$ parameters or as many as $pq - q(q-1)/2 + p$ parameters. The resulting parsimonious Gaussian mixture models are provided by the `R` package `pgmm` (MCNICHOLAS et al., 2015). Kao and Roy (2013) referred PPCA as a *uniform-residual rank-constrained maximum-likelihood* (URM) covariance estimator, and advocated that, as shown in (FAN; FAN; LV, 2008), imposing a hard constraint on the number of factors will not lead to an optimal estimate. Motivated by this premise, they proposed an *uniform-residual trace-penalized* (UTM) covariance estimator, which set the $p - q$ smallest eigenvalues to a consistent estimated constant and subtract a penalty from the q largest ones, without any eigenvector constraint. This approach provides a better conditioned covariance estimate (because it reduces the condition number), but like the Ledoit-Wolf estimator, it regularizes only the eigenvalues, keeping the eigenvectors intact.

A central issue in PCA is choosing the optimal number q of principal component (PC) loadings to be retained. Many methods, both heuristic and statistically based, have been proposed to help determining this quantity. For a detailed review see (JOLLIFFE, 2002; JACKSON, 2005). Some of these heuristics and strategies are provided by the `R` package `nFactors` (RAICHE, 2010). Due to its intuitively interpretation and the ease to compute, the most widely used criterion for choosing q is to select a cumulative percentage of total variation explained by the principal components. Given that the variance of the

j^{th} PC is λ_j, the percentage of variation accounted by the first q PCs is therefore

$$t_q = 100\frac{\sum_{j=1}^{q}\lambda_j}{\sum_{j=1}^{p}\lambda_j} = 100\frac{\sum_{j=1}^{q}\lambda_j}{\text{tr}\{\boldsymbol{S}\}} \tag{2.27}$$

Choosing a cut-off t^* somewhere between 70% and 90% and retaining q PCs, where q is the smallest integer for which $t_q > t^*$, provides a rule which in practice preserves in the first q eigenvalues and eigenvectors most of the information in the sample covariance $\boldsymbol{S}$ (JOLLIFFE, 2002). Cangelosi and Goriely (2007) empirically found that the cumulative percent of variance at the 90% level retains the greatest number of components. However, there is no rule of thumb for defining t^*, and the use of criteria based on the cumulative percent of variation explained are not recommended. In practice, the resulting rules are completely arbitrary (CANGELOSI; GORIELY, 2007).

Given that the probabilistic formulation of PCA can be interpreted as maximum-likelihood density estimation (TIPPING; BISHOP, 1999), Minka (2000) demonstrated that Bayesian model selection can be applied to the PPCA model to accurately determine the true dimensionality of the principal subspace, given enough data. The oracle q is then selected as

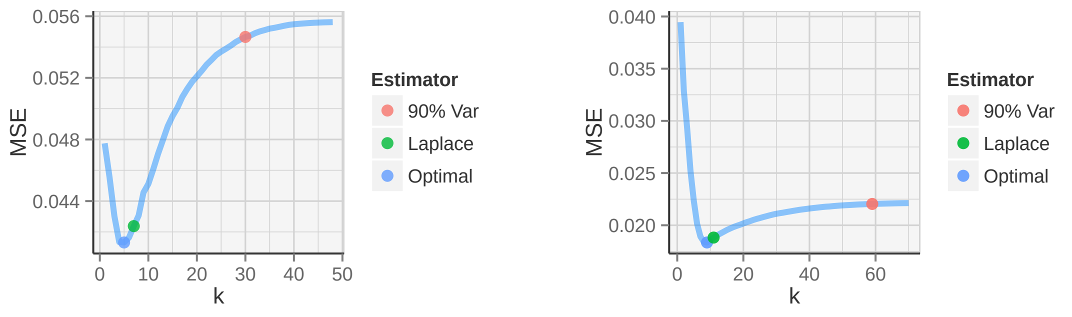
$$q = \arg\min_{q}\ -\log\left(p(\boldsymbol{X}|q)\right) \tag{2.28}$$

For a training data set $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$ and a subspace dimensionality q, $p(\boldsymbol{X}|q)$ is the probability of observing $\boldsymbol{X}$ given q. Minka (2000) derived a Laplace approximation for $p(\boldsymbol{X}|q)$. To avoid arithmetic overflow is recommended to take the logarithm before computing it:

$$\begin{aligned}\log\left(p(\boldsymbol{X}|q)\right) \approx\ & \frac{2q+m}{2}\log(2) + \frac{q+m}{2}\log(\pi) + \sum_{i=1}^{q}\log\Gamma\left((p-i+1)/2\right) \\ & -\frac{\log(\pi)}{2}\sum_{i=1}^{q}(p-i+1) - \frac{N}{2}\sum_{j=1}^{q}\log(\lambda_j) - \frac{N(p-q)}{2}\log(\sigma_{\text{ML}}) \\ & -\frac{1}{2}\sum_{i=1}^{q}\sum_{j=i+1}^{p}\Big(\log\left(\lambda_j^{-1}-\lambda_i^{-1}\right) + \log\left(\lambda_i-\lambda_j\right) + \log(N)\Big) - \frac{q}{2}\log(N)\end{aligned} \tag{2.29}$$

where $m = p(p-1)/2 - (p-q)(p-q-1)/2$ and $\Gamma(x)$ is the gamma function. Given the sample eigenvalues, the cost of computing $\log\left(p(\boldsymbol{X}|q)\right)$ is $O\left(\min(p,N)q\right)$, which is less than one loop over the training data.

To test the performance of these two algorithms for model selection, data has been generated from a 100-dimensional Gaussian distribution with the 10 largest covariance matrix eigenvalues given by $[10, 9, \ldots, 1]$ and the remaining 90 given by 0.01, with the addition of a Gaussian white noise $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{100})$ to the samples. Figure 2.4 shows the MSE between the true covariance matrix and the PPCA estimate for each choice of



(a) Laplace evidence and 90% of the cumulative variance for $p/N = 2$

(b) Laplace evidence and 90% of the cumulative variance for $p/N = 0.5$

Figure 2.4 – MSE of the PPCA covariance estimation considering both component selection methodologies, evaluated in two different settings: $p > N$ and $p < N$.

Table 2.2 – Averages and standard errors over 100 replications of performance measure for different methodologies to select the dimensionality.

	Laplace Evidence	90% Variability
$p/N = 5$		
q	2.30 (0.75)	14.52 (0.50)
$p\boldsymbol{L}_3$	**913.13 (202.56)**	1398.72 (197.78)
$p/N = 1$		
q	7.70 (0.59)	45.33 (0.59)
$p\boldsymbol{L}_3$	**280.98 (21.11)**	349.96 (21.22)
$p/N = 0.5$		
q	8.92 (0.31)	59.13 (0.53)
$p\boldsymbol{L}_3$	**187.67 (10.90)**	224.92 (11.08)
$p/N = 0.1$		
q	10.00 (0.00)	76.02 (0.14)
$p\boldsymbol{L}_3$	**117.19 (2.71)**	125.21 (2.69)

dimensionality, with sample size $N = [50, 200]$. The results over 100 replications are reported in Table 2.2. It is evident that the Laplace's method proposed by Minka (2000) outperforms the traditional cumulative percent of variance criterion, and picks q quite close to the minimal MSE solution.

2.4 Sparse Principal Component Analysis

Principal component analysis (PCA) (JOLLIFFE, 2002; JACKSON, 2005; BISHOP, 2006) is a data-processing technique that is widely used for applications such as dimensionality reduction, lossy data compression, feature extraction, and data visualization. PCA seeks a linear projection of the original variables such that the derived variables capture maximal variance. Let the p random variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p$, with sample size N, be represented by the $N \times p$ data matrix $\boldsymbol{X}$. Now, without loss of generality, assume that the variable means are all 0. PCA is concerned with explaining the covariance matrix $\boldsymbol{\Sigma}$ of $\boldsymbol{X}$ through a new considerably smaller set of variables, commonly referred as *principal components* (PCs), describing the maximum variability in the original data. Specifically, the PCs of $\boldsymbol{X}$ are just a particular set of linear combinations of the original variables, say

$$\begin{aligned}
\boldsymbol{z}_1 &= \boldsymbol{c}_1^T\boldsymbol{X} = c_{11}\boldsymbol{x}_1 + c_{12}\boldsymbol{x}_2 \ + \ldots + c_{1p}\boldsymbol{x}_p, \\
\boldsymbol{z}_2 &= \boldsymbol{c}_2^T\boldsymbol{X} = c_{21}\boldsymbol{x}_1 + c_{22}\boldsymbol{x}_2 \ + \ldots + c_{2p}\boldsymbol{x}_p, \\
&\vdots \\
\boldsymbol{z}_i &= \boldsymbol{c}_i^T\boldsymbol{X} = c_{i1}\boldsymbol{x}_1 + c_{i2}\boldsymbol{x}_2 \ + \ldots + c_{ip}\boldsymbol{x}_p, \\
&\vdots \\
\boldsymbol{z}_p &= \boldsymbol{c}_p^T\boldsymbol{X} = c_{p1}\boldsymbol{x}_1 + c_{p2}\boldsymbol{x}_2 \ + \ldots + c_{pp}\boldsymbol{x}_p,
\end{aligned} \tag{2.30}$$

with

$$\begin{aligned}
\operatorname{var}(\boldsymbol{z}_i) &= \boldsymbol{c}_i^T\boldsymbol{\Sigma}\boldsymbol{c}_i, \ i = 1, \ldots, p, \\
\operatorname{cov}(\boldsymbol{z}_i, \boldsymbol{z}_j) &= \boldsymbol{c}_i^T\boldsymbol{\Sigma}\boldsymbol{c}_j, \ i, j = 1, \ldots, p.
\end{aligned} \tag{2.31}$$

Then, the PCs of $\boldsymbol{X}$ are those uncorrelated random variables $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_p$ whose variances are as large as possible. More precisely, the i^{th} principal component is obtained by finding $\boldsymbol{c}_i$ so that $\operatorname{var}(\boldsymbol{z}_i) = \boldsymbol{c}_i^T\boldsymbol{\Sigma}\boldsymbol{c}_i$ is maximized subjected to

$$\boldsymbol{c}_i^T\boldsymbol{c}_i = 1, \tag{2.32}$$

and

$$\operatorname{cov}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \boldsymbol{c}_i^T\boldsymbol{\Sigma}\boldsymbol{c}_j = 0, \text{ for } i = 1, 2, \ldots, j-1 \tag{2.33}$$

where the unknown covariance $\boldsymbol{\Sigma}$ is simply replaced by it empirical estimate $\boldsymbol{S}$. It can be show (BISHOP, 2006) that $\boldsymbol{c}_i$ is just the i^{th} normalized eigenvector of $\boldsymbol{S}$ and $\operatorname{var}(\boldsymbol{z}_i) = \lambda_i$ is the corresponding i^{th} largest eigenvalue. Geometrically, the principal component variables $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_p$ are the axes of a new coordinate system obtained by rotating the axes of the original system (the $\boldsymbol{x}$'s). The new axes represent the directions of maximum variability, and often lead to a satisfactory description of the data requiring considerably fewer variables, say q, than the original p variables (MONTGOMERY, 2007).

The *singular value decomposition* (SVD) of the data matrix $\boldsymbol{X}$ with centered columns is another way of expressing the principal components of the variables in $\boldsymbol{X}$ (see Section B.4 of Appendix B for more details about SVD). The SVD of $\boldsymbol{X}$ has the form

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T, \tag{2.34}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are $N \times p$ and $p \times p$ orthogonal matrices, respectively. The columns $\boldsymbol{v}_i$'s of $\boldsymbol{V}$ are the normalized eigenvectors of $\boldsymbol{S}$, $\boldsymbol{c}_i$'s, also called the principal component *loadings*. Under (2.34), the PCs can be expressed as the columns of $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{V} = \boldsymbol{U}\boldsymbol{D}$. The i^{th} eigenvalue of $\boldsymbol{S}$, and hence the sample variance of the i^{th} PC, is $\text{var}(\boldsymbol{z}_i) = \boldsymbol{D}_{ii}^2/(N-1)$. Usually only the first q PCs, with $q \ll \min(N, p)$, are chosen to represent the data, thus a great dimensionality reduction is achieved. The method proposed by Minka (2000), discussed in Section 2.3, is a effective approach for choosing q.

The success of PCA lies in the fact that new variables with maximum variance will lead to: (i) a simpler and more parsimonious approximation to $\boldsymbol{\Sigma}$, (ii) a low-dimensional data representation, easier to interpret and visualize when $q = [2, 3]$, with minimal information loss. However, as pointed out by Zou, Hastie and Tibshirani (2006), the standard PCA has an obvious drawback, since each PC is a linear combination of all p variables and the loading entries are typically nonzero. This makes it often difficult to interpret the derived PCs. Moreover, in high-dimensional settings when $p >> N$, the traditional estimates of the PC directions are inconsistent estimators of their population counterparts, as demonstrated by Johnstone and Lu (2009). Therefore, deriving principal components with regularized and sparse loadings could enhance their interpretability and make the problem well-posed in the "large p, small N" data setting.

Ideally, one would define the i^{th} sparse vector of PC loadings $\boldsymbol{v}_i$ with a large number of zero entries as the solution to

$$\boldsymbol{v}_i = \arg\max_{\boldsymbol{v}} \boldsymbol{v}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v}, \tag{2.35}$$

subject to

$$\boldsymbol{v}_i^T\boldsymbol{v}_i = 1,\ \boldsymbol{v}_{i-1}^T\boldsymbol{v}_i = 0 \text{ for } i \geq 2, \text{ and } \|\boldsymbol{v}\|_0 \leq t, \tag{2.36}$$

where the ℓ_0-norm $\|\boldsymbol{v}\|_0$ simply counts the number of nonzero entries of $\boldsymbol{v}$ and t is a integer. The optimization amounts to seek a linear combination of variables with the highest variance among those with at most t nonzero loadings. However, this problem is doubly nonconvex, since it involves maximizing (as opposed to minimizing) a convex function with a combinatorial constraint (HASTIE; TIBSHIRANI; WAINWRIGHT, 2015). The SCoTLASS procedure of Jolliffe, Trendafilov and Uddin (2003) is a natural relaxation of this objective, based on replacing the ℓ_0-norm by the ℓ_1-norm $\|\boldsymbol{v}_i\|_1 = \sum_{j=1}^{p} |v_{ij}|$. The

ℓ_1-constraint encourages some of the loadings to be zero and hence $\boldsymbol{v}_i$ to be sparse. Although the ℓ_1-norm is convex, the SCoTLASS reamains a nonconvex optimization problem, and moreover is not well-suited to simple iterative algorithms. In fact, Zou, Hastie and Tibshirani (2006) have shown that PCA can be formulated as a regression-type optimization problem, and sparse loadings can be then obtained imposing the LASSO or the elastic net constraint on the regression coefficients (see Appendix C for a brief review of regularized regression models). More precisely, denoting by $\boldsymbol{z}_i = \boldsymbol{X}\boldsymbol{v}_i = \boldsymbol{u}_i\boldsymbol{D}_{ii}$ the i^{th} principal component, and considering a positive α penalty, the ridge estimates $\hat{\boldsymbol{\beta}}(\alpha)$ given by

$$\hat{\boldsymbol{\beta}}(\alpha) = \arg\min_{\boldsymbol{\beta}} \left\{(\boldsymbol{z}_i - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{z}_i - \boldsymbol{X}\boldsymbol{\beta}) + \alpha\boldsymbol{\beta}^T\boldsymbol{\beta}\right\} = (\boldsymbol{X}^T\boldsymbol{X} + \alpha\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{z}_i \tag{2.37}$$

can be applied to exactly compute the i^{th} loading vector (ZOU; HASTIE; TIBSHIRANI, 2006)

$$\boldsymbol{v_i} = \frac{\hat{\boldsymbol{\beta}}(\alpha)}{\|\hat{\boldsymbol{\beta}}(\alpha)\|_2} \tag{2.38}$$

In this case, the ridge penalty is not used to penalize the regression coefficients but to ensure the reconstruction of principal components in rank-deficient data settings. Particularly, after normalization in (2.38) the coefficients are independent of α. On the other hand, adding the ℓ_1-penalty α_1 to (2.37) leads to the following optimization problem

$$\hat{\boldsymbol{\beta}}(\alpha, \alpha_1) = \arg\min_{\boldsymbol{\beta}} \left\{(\boldsymbol{z}_i - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{z}_i - \boldsymbol{X}\boldsymbol{\beta}) + \alpha\boldsymbol{\beta}^T\boldsymbol{\beta} + \alpha_1\sum_{j=1}^{p}|\beta_j|\right\} \tag{2.39}$$

which gives

$$\hat{\boldsymbol{v}}_i = \frac{\hat{\boldsymbol{\beta}}(\alpha, \alpha_1)}{\|\hat{\boldsymbol{\beta}}(\alpha, \alpha_1)\|_2} \tag{2.40}$$

as an approximation to $\boldsymbol{v}_i$, and $\boldsymbol{X}\hat{\boldsymbol{v}}_i$ as the i^{th} approximated principal component. Clearly, large enough α_1 gives a sparse $\hat{\boldsymbol{\beta}}$, and hence a sparse $\hat{\boldsymbol{v}}_i$. Given a fixed α, (2.39) is efficiently solved for all α_1 by using the LARS-EN algorithm (ZOU; HASTIE, 2005). However, since (2.39) depends on the results of PCA, it is not a genuine approach to impose sparsity on the principal components. Thus, Zou, Hastie and Tibshirani (2006) alternatively proposed to compute multiple sparse principal components, say q, simultaneously by minimizing

$$\sum_{i=1}^{N}\|\boldsymbol{x}_i - \boldsymbol{\Theta}\boldsymbol{B}^T\boldsymbol{x}_i\|_2^2 + \alpha\sum_{j=1}^{q}\|\boldsymbol{\beta}_j\|_2^2 + \sum_{j=1}^{q}\alpha_1^{(j)}\|\boldsymbol{\beta}_j\|_1 \tag{2.41}$$

subject to $\boldsymbol{\Theta}^T\boldsymbol{\Theta} = \boldsymbol{I}_q$. Here $\boldsymbol{x}_i$ is the i^{th} row of $\boldsymbol{X}$, $\boldsymbol{B}$ is a $p \times q$ matrix with the coefficients $\boldsymbol{\beta}_j$ as its columns, and $\boldsymbol{\Theta}$ is also a $p \times q$ matrix initialized as $\boldsymbol{V}[, 1 : q]$ (the loadings of the first q ordinary principal components). An efficient solution to this optimization problem is provided by the R package `elasticnet` (ZOU; HASTIE, 2012).

Alternatively, Witten, Tibshirani and Hastie (2009) proposed to regularize eigenvectors and develop sparse PCA based on the connection between SVD and the principal components. They approximate the data matrix $\boldsymbol{X}$ as $\hat{\boldsymbol{X}} = \sum_{j=1}^{q} d_j \boldsymbol{u}_j \boldsymbol{v}_j^T$, where d_j, $\boldsymbol{u}_j$, and $\boldsymbol{v}_j$ minimize the squared Frobenius norm of $\boldsymbol{X} - \hat{\boldsymbol{X}}$, subject to ℓ_1-penalties on $\boldsymbol{u}_j$ and $\boldsymbol{v}_j$. This results in a regularized version of SVD, referred to as *penalized matrix decomposition* (PMD). When PMD is applied using a ℓ_1-penalty on $\boldsymbol{v}_j$ but with no constraint on $\boldsymbol{u}_j$, it results in a sparse principal components (SPC) method. In this case, a sparse PCA solution can be obtained from solving the following optimization problem:

$$\underset{\|\boldsymbol{u}\|_2=\|\boldsymbol{v}\|_2=1}{\text{maximize}} \left\{\boldsymbol{u}^T\boldsymbol{X}\boldsymbol{v}\right\} \text{ subject to } \|\boldsymbol{v}\|_1 \leq t. \tag{2.42}$$

Any optimal solution $\hat{\boldsymbol{v}}$ to this problem is also optimal for the SCoTLASS program (HASTIE; TIBSHIRANI; WAINWRIGHT, 2015). The algorithm to solve (2.42) consists of the following two steps:

1. Initialize $\boldsymbol{v} \in \mathbb{R}^p$ with $\|\boldsymbol{v}\|_2 = 1$.
2. Iterate until changes in $\boldsymbol{u}$ and $\boldsymbol{v}$ are sufficiently small:
   a) Update $\boldsymbol{u} \in \mathbb{R}^N$ via $\boldsymbol{u} \leftarrow \dfrac{\boldsymbol{X}\boldsymbol{v}}{\|\boldsymbol{X}\boldsymbol{v}\|_2}$.
   b) Update $\boldsymbol{v} \in \mathbb{R}^p$ via
   $$\boldsymbol{v} \leftarrow \boldsymbol{v}(\alpha, \boldsymbol{u}) = \frac{\mathcal{S}_\alpha\left(\boldsymbol{X}^T\boldsymbol{u}\right)}{\|\mathcal{S}_\alpha\left(\boldsymbol{X}^T\boldsymbol{u}\right)\|_2}, \tag{2.43}$$
   where $\alpha = 0$ if $\|\boldsymbol{X}^T\boldsymbol{u}\|_1 \leq t$, and otherwise $\alpha > 0$ is chosen such that $\|\boldsymbol{v}(\alpha, \boldsymbol{u})\|_1 = t$.

Here $\mathcal{S}_\alpha(x) = sign(x)(|x| - \alpha)_+$ is the familiar soft-thresholding operator at level α. After convergence of (2.43), the corresponding singular value is obtained by $d \leftarrow \boldsymbol{u}^T\boldsymbol{X}\boldsymbol{v}$. Witten, Tibshirani and Hastie (2009) also extended this method to achieve orthogonality among subsequent $\boldsymbol{v}_i$'s. The PMD algorithm for sparse PCA and sparse SVD is implemented in the R package `PMA` (*penalized multivariate analysis*) (WITTEN et al., 2013).

Figure 2.5 shows the results of PCA and SPCA applied to the DNA microarray dataset described in (CHRISTENSEN et al., 2009). The data consists of 1,413 expression measurements on 773 genes from 217 human tissue samples, in other words $\boldsymbol{X}_{217\times1413} = \{\boldsymbol{x}_n \in \mathbb{R}^{1413}\}_{n=1}^{217}$, classified into 3 subgroups. This and other small-sample, high-dimensional microarray datasets are available in the R package `datamicroarray` (RAMEY, 2013). In an attempt to achieve a bi-dimensional representation of the data, four different dimensionality reduction approaches have been considered: best subset selection using information gain (MITCHELL, 1997), standard PCA, sparse PCA (SPCA), and the two variables associated with the highest absolute values of the first and second sparse PC loading coefficients, respectively. As can be seen, only the feature extraction techniques using principal components are able to completely separate the classes using only two predictors. However, PCA is not able to discard irrelevant variables from the principal component estimates, as shown in Figure 2.6 in which each entry of the loading vectors is colored accordingly to its absolute value (white for zero and proportional gray scale for non-zero values). On the other hand, SPCA selects few variables to represent the principal component directions, leading to more parsimonious and interpretable data representations.

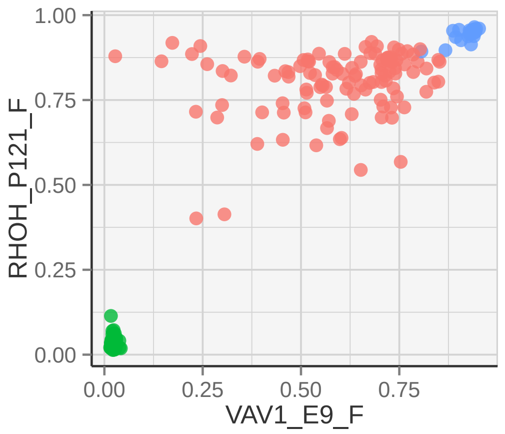
In order to achieve a consistent covariance estimate in high dimensions, a natural idea is to replace the sample eigenvectors in the probabilistic PCA formulation (2.24) by their sparse estimates. Although this approach regularizes covariance, it is no longer a probabilistic formulation. Following Zou, Hastie and Tibshirani (2006), Guan and Dy (2009) developed a probabilistic formulation for sparse PCA by introducing a Laplacian prior to each element $\boldsymbol{W}_{ij}$ of the transformation matrix $\boldsymbol{W}$ in (2.24). This results in an equivalent ℓ_1 constraint, since Laplacian priors are equivalent to ℓ_1 regularization (PARK; CASELLA, 2008). Variational inference (BISHOP, 1999) is applied to learn the resulting probabilistic model. In this work, for simplicity, the non-probabilistic approach will be adopted to estimate the covariance.

2.5 Regularized Cholesky Factors

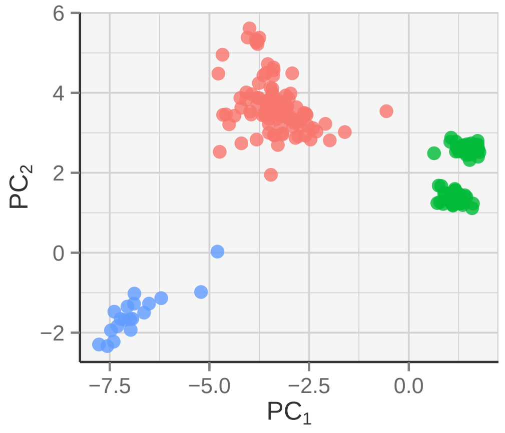
In many applications, the need for the precision matrix $\boldsymbol{\Sigma}^{-1}$ is stronger than that for $\boldsymbol{\Sigma}$ itself. However, computing the former as the inverse of the latter can take $\mathcal{O}(p^3)$ operations, and should be avoided when p is large. As a tool for regularizing the precision matrix directly, Wu and Pourahmadi (2003) developed a regression-based derivation and interpretation of its modified Cholesky decomposition (see Section B.3 of Appendix B). The latter can be written in the form

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{P}^T\boldsymbol{D}^{-1}\boldsymbol{P} \tag{2.44}$$

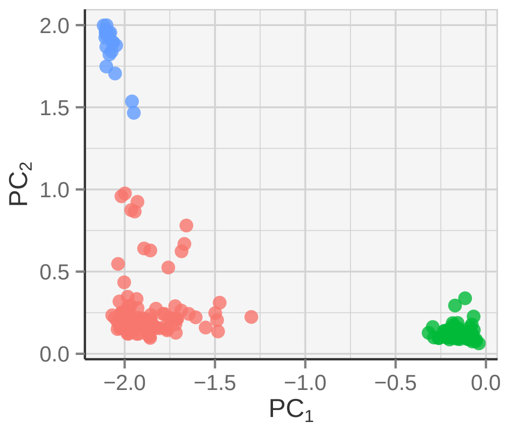
where $\boldsymbol{P}$ is a unit triangular matrix and $\boldsymbol{D}$ is diagonal. For a data matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)$ with centered columns and a positive-definite covariance matrix $\boldsymbol{\Sigma}$, this factorization arises



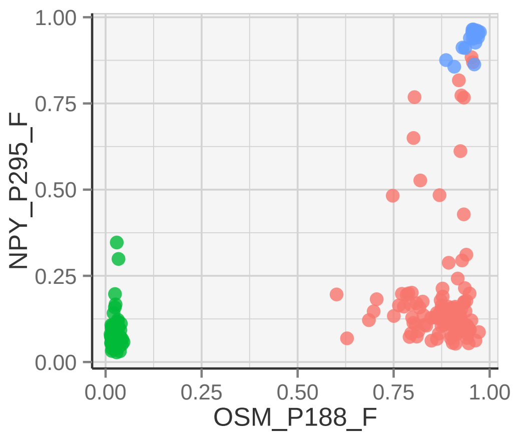
(a) Two most discriminative features according to information gain.



(b) First two principal components.



(c) First two sparse principal components.



(d) Two most important predictors to the first two sparse PCs.

Figure 2.5 – Class separation after dimensionality reduction of the Christensen's microarray dataset (p=1413, N=217).

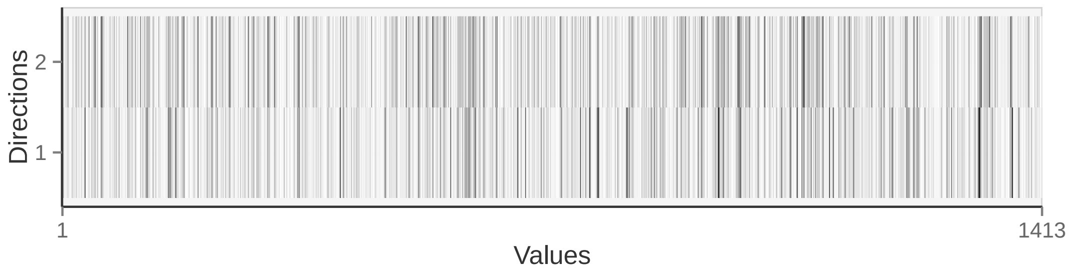
from regressing each variable $\boldsymbol{x}_t$ on its predecessors $\boldsymbol{x}_{t-1}, \ldots, \boldsymbol{x}_1$, that is, fitting regressions

$$\boldsymbol{x}_t = \sum_{j=1}^{t-1} \phi_{tj} \boldsymbol{x}_j + \epsilon_t, \; t = 1, \ldots, p \tag{2.45}$$

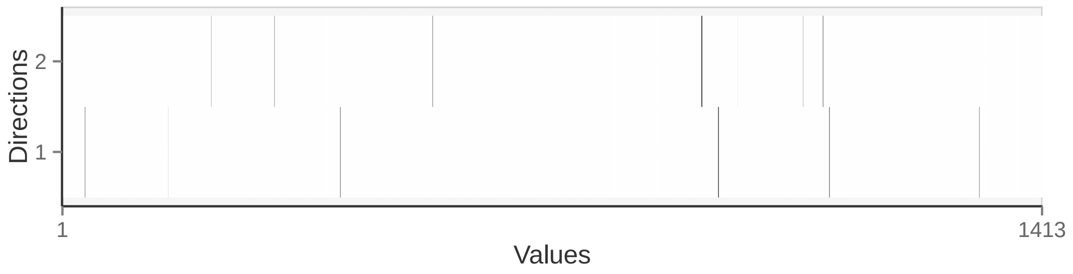
where $\epsilon_t = \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t$ denotes the linear least-squares prediction error with variance $\sigma_t^2 = \text{var}(\epsilon_t)$, and $\epsilon_1 = \boldsymbol{x}_1$. Let $\boldsymbol{\epsilon} = \boldsymbol{X} - \hat{\boldsymbol{X}} = (\epsilon_1, \ldots, \epsilon_p)$ be the vector of successive uncorrelated prediction errors with

$$\text{cov}(\boldsymbol{\epsilon}) = \text{diag}(\sigma_1^2, \ldots, \sigma_p^2) = \boldsymbol{D}. \tag{2.46}$$

Then, (2.45) can be rewritten in matrix form as $\boldsymbol{PX} = \boldsymbol{\epsilon}$, where $\boldsymbol{P}$ is the following unit lower triangular matrix with the negative of the unique regression coefficients ϕ_{tj} as its



(a) PCA loadings.



(b) SPCA loadings.

Figure 2.6 – Hinton diagrams of the loading matrix $\boldsymbol{Q}_{p\times 2}$ with empirical and sparse eigenvectors, respectively. The whiter the diagram, the greater the sparsity level.

entries:

$$\boldsymbol{P} = \begin{bmatrix} 1 & & & & \\ -\phi_{21} & 1 & & & \\ -\phi_{31} & -\phi_{32} & 1 & & \\ \vdots & & & \ddots & \\ -\phi_{p1} & -\phi_{p2} & \cdots & -\phi_{p,p-1} & 1 \end{bmatrix} \tag{2.47}$$

Now, using the fact that the errors are uncorrelated,

$$\boldsymbol{D} = \text{cov}(\boldsymbol{\epsilon}) = \text{cov}(\boldsymbol{PX}) = \boldsymbol{P}\text{cov}(\boldsymbol{X})\boldsymbol{P}^T = \boldsymbol{P}\Sigma\boldsymbol{P}^T, \tag{2.48}$$

and thus

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{P}^T\boldsymbol{D}^{-1}\boldsymbol{P}. \tag{2.49}$$

This decomposition transforms the precision matrix estimation into a regression problem, and hence regularization approaches for regression can be applied. If these regressions are not regularized, the resulting estimate is simply the inverse of the sample covariance $\boldsymbol{S}^{-1}$. An important consequence of (2.49) is that for any estimate $\left(\hat{\boldsymbol{P}}, \hat{\boldsymbol{D}}\right)$ of the Cholesky factors, the estimated precision matrix, $\hat{\boldsymbol{\Sigma}}^{-1} = \hat{\boldsymbol{P}}^T\hat{\boldsymbol{D}}^{-1}\hat{\boldsymbol{P}}$, is guaranteed to be positive-definite. In particular, the LASSO penalty can be imposed on the regression coefficients in (2.45) to introduce sparsity directly in the inverse, as in (HUANG et al., 2006). Alternatively, Bickel and Levina (2008b) proposed banding the Cholesky factor $\boldsymbol{P}$, which refers to regularizing the inverse by only including the immediate k predecessors in the regression, $\boldsymbol{x}_{t-1}, \ldots, \boldsymbol{x}_{t-k}$, for some fixed k. While the covariance estimators presented so far regularize either eigenvalues or eigenvectors, these two Cholesky based approaches regularize both eigenvectors and eigenvalues.

Rothman, Levina and Zhu (2010) demonstrated that the modified Cholesky factor of the covariance matrix, rather than its inverse, also has a natural regression interpretation, and therefore all Cholesky-based regularization methods can be applied to the covariance matrix itself instead of its inverse to obtain a sparse estimator with guaranteed positive definiteness. Particularly, the modified Cholesky factorization of $\boldsymbol{\Sigma}$ can be obtained from a latent variable regression model. Let $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$ be the modified Cholesky decomposition of $\boldsymbol{\Sigma}$, where $\boldsymbol{D}$ is diagonal and $\boldsymbol{L}$ is unit lower triangular. Now, let $\boldsymbol{\epsilon}$ be a normal vector with independent components, $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{D})$, and $\boldsymbol{X} = \boldsymbol{L}\boldsymbol{\epsilon}$. This leads to

$$\boldsymbol{\Sigma} = \operatorname{cov}(\boldsymbol{L}\boldsymbol{\epsilon}) = \boldsymbol{L}\operatorname{cov}(\boldsymbol{\epsilon})\boldsymbol{L}^T = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T \tag{2.50}$$

where $\boldsymbol{\epsilon}$ is a latent or unobserved vector. However, since $\boldsymbol{L}$ is lower triangular, (2.50) can be seen as a sequence of p varying-coefficient and varying-order regressions, where each variable $\boldsymbol{x}_t$ is regressed on the previous regression errors $\epsilon_{t-1}, \ldots, \epsilon_1$, as follows:

$$\boldsymbol{x}_t = \sum_{j=1}^{t-1} l_{tj}\epsilon_j + \epsilon_t = \hat{\boldsymbol{x}}_t + \epsilon_t,\ t = 1, \ldots, p \tag{2.51}$$

with $\epsilon_1 = \boldsymbol{x}_1$. Given a $N \times p$ data matrix $\boldsymbol{X}$, with all columns $\boldsymbol{x}_t$'s centered by their sample means, the regression coefficients and the residuals can be computed as

$$\hat{\boldsymbol{l}}_t = \arg\min_{\boldsymbol{l}_t} \|\boldsymbol{x}_t - \boldsymbol{Z}_t\boldsymbol{l}_t\|^2,\ \boldsymbol{e}_t = \boldsymbol{x}_t - \boldsymbol{Z}_t\hat{\boldsymbol{l}}_t, \tag{2.52}$$

where $\boldsymbol{l}_t = (l_{t1}, \ldots, l_{t,t-1})^T$ and $\boldsymbol{Z}_t = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{t-1})$. The variances are estimated as $\hat{\sigma}_t^2 = (N-1)^{-1}\|\boldsymbol{e}_t\|^2$. If all model matrices $\boldsymbol{Z}_t$'s are of full rank, and all regressions are fitted by least squares, the resulting estimate recovers the sample covariance matrix $\boldsymbol{S} = (N-1)^{-1}\boldsymbol{X}^T\boldsymbol{X} = \hat{\boldsymbol{L}}\hat{\boldsymbol{D}}\hat{\boldsymbol{L}}^T$. Again, for any estimate $(\hat{\boldsymbol{L}}, \hat{\boldsymbol{D}})$ of the Cholesky factors, the estimated covariance matrix $\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{L}}\hat{\boldsymbol{D}}\hat{\boldsymbol{L}}^T$ is guaranteed to be positive definite. Following Wu and Pourahmadi (2003) and Bickel and Levina (2008b), Rothman, Levina and Zhu (2010) proposed to introduce sparsity in the Cholesky factor $\boldsymbol{L}$ by estimating only their first k sub-diagonals and setting the rest to zero.

However, the derivation of the Cholesky factor parameters from (2.45) and (2.51) is dependent on the order of the variables in $\boldsymbol{X}$ (POURAHMADI, 2011). In some situations, for instance time series analysis, the variables have a natural order and so using it makes sense. However, in many data analytic problems, there is no canonical ordering of the variables. Wagaman and Levina (2009) have proposed an Isomap method for discovering an order among the variables based on their correlations. But finding a permutation that makes the analysis particularly simple would be practically infeasible. In this case, as pointed out by Karoui (2008), a natural requirement is to find an estimator which is invariant under permutations of the order of the variables, referred to as *permutation-equivariant* estimator. Since there is no guarantee of natural ordering among variables in classification problems, in this work only permutation-equivariant estimators will be considered.

2.6 Penalized Likelihood Estimation

A natural way to estimate the precision matrix $\mathbf{\Sigma}^{-1}$ is by maximizing the log-likelihood of the data. For an independent and identically distributed (i.i.d.) random sample of size N, say $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$, from a $\mathcal{N}_p(\mathbf{0}, \mathbf{\Sigma})$ distribution, the log-likelihood takes the form

$$l(\mathbf{\Omega}) = \log|\mathbf{\Omega}| - \mathrm{tr}(\boldsymbol{S}\mathbf{\Omega}), \tag{2.53}$$

where $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$ with determinant $|\mathbf{\Omega}|$, and $\boldsymbol{S} = \boldsymbol{X}^T\boldsymbol{X}/N$ is the maximum likelihood estimate of the covariance matrix of the data. The function $-l(\mathbf{\Omega})$ is known to be convex in $\mathbf{\Omega}$. Then minimizing it with respect to $\mathbf{\Omega}$ leads to the maximum likelihood estimate $\hat{\mathbf{\Omega}} = \boldsymbol{S}^{-1}$, which will in general contain no entries exactly equal to zero. Moreover, it is well known (LEDOIT; WOLF, 2004) that the empirical covariance matrix $\boldsymbol{S}$ performs poorly when p is large, and become singular when $p > N$. In this case, the maximum likelihood estimate cannot be computed. To regularize $\boldsymbol{S}$ and induce sparsity in $\mathbf{\Omega}$, one may maximize the following LASSO-type penalized log-likelihood (YUAN; LIN, 2007):

$$l(\mathbf{\Omega}, \alpha) = \log|\mathbf{\Omega}| - \mathrm{tr}(\boldsymbol{S}\mathbf{\Omega}) - \alpha||\mathbf{\Omega}||_1 \tag{2.54}$$

where α is a nonnegative tuning parameter that controls sparsity, and $||\mathbf{\Omega}||_1$ is the ℓ_1-norm of a suitable factor of the precision matrix. The maximizer $\hat{\mathbf{\Omega}}$ of (2.54) is positive-definite for all $\alpha > 0$ even if $\boldsymbol{S}$ is singular. Three types of estimators can be derived from (2.54) considering an ℓ_1-penalty on (POURAHMADI; WU, 2013):

1. the modified Cholesky factor of $\mathbf{\Omega}$
2. the off-diagonal entries of $\mathbf{\Omega}$
3. all entries of $\mathbf{\Omega}$.

Imposing sparsity indirectly on the precision matrix by regularizing its Cholesky factor has two main disadvantages (POURAHMADI, 2011): (1) a sparse factor does not necessarily imply sparsity on the precision matrix, and (2) the factor sparsity structure could be sensitive to permutations of the order of the variables in $\boldsymbol{X}$. The *graphical LASSO* (glasso) algorithm, proposed by Friedman, Hastie and Tibshirani (2008), imposes a penalty on all entries of $\boldsymbol{\Omega}$, and produces sparse (when α is large enough) and permutation-equivariant estimators of the precision matrix. The transformation of all entries at the same time leads to regularizing both eigenvalues and eigenvectors of the sample covariance matrix. The glasso is inspired by the LASSO regression-based interpretation of the entries of $\boldsymbol{\Omega}$ proposed by Meinshausen and Bühlmann (2006), and is one of the fastest algorithms for sparse estimation of a precision matrix. The algorithm is available in the `R` package `glasso` (FRIEDMAN; HASTIE; TIBSHIRANI, 2014), which uses a block diagonal screening rule to speed up computations considerably, as proposed in (WITTEN; FRIEDMAN; SIMON, 2011).

As a promising alternative to glasso, Hsieh et al. (2011) proposed a quadratic approximation method to solve (2.54), the *Quadratic Inverse Covariance* (QUIC) algorithm. The `R` package `QUIC` provides an interface to the original `C++` implementation. QUIC is much faster and outperforms glasso in many situations (HSIEH et al., 2014), and will be preferred in this work.

In applying glasso and QUIC, one will usually need to select an appropriate value for the sparsity level tuning parameter α. Partitioning the random sample $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ into K subsets, $\mathcal{A}_1, \ldots, \mathcal{A}_K$, a good choice for α is

$$\hat{\alpha}_{\text{CV}} = \arg\max_{\alpha} \{K^{-1} \sum_{k=1}^{K} l\left(\hat{\boldsymbol{\Omega}}_\alpha(\boldsymbol{S}_{\mathcal{A}_k^C}); \boldsymbol{S}_{\mathcal{A}_k}\right)\} \tag{2.55}$$

where $\mathcal{A}_k^C$ is the complement of the subset $\mathcal{A}_k$. This procedure, also known as K-fold cross-validation, selects a value of α that maximizes the validation log-likelihood

$$l\left(\hat{\boldsymbol{\Omega}}_\alpha(\boldsymbol{S}_{\text{train}}); \boldsymbol{S}_{\text{valid}}\right) = \log|\hat{\boldsymbol{\Omega}}_\alpha(\boldsymbol{S}_{\text{train}})| - \operatorname{tr}\left(\boldsymbol{S}_{\text{valid}}\hat{\boldsymbol{\Omega}}_\alpha(\boldsymbol{S}_{\text{train}})\right). \tag{2.56}$$

The number of subsets (folds) is typically chosen as 5 or 10 (HASTIE; TIBSHIRANI; WAINWRIGHT, 2015). Given the maximum absolute value of $\boldsymbol{S}$, $s_{\max}$, a reasonable range for the regularization parameter α may be $[s_{\max}/10, s_{\max}]$, as suggested in the package `glasso`. Two optional regularization parameter selection methods are provided by the `R` package `huge` (ZHAO et al., 2012; ZHAO et al., 2014): the stability approach for regularization selection (StARS) (LIU; ROEDER; WASSERMAN, 2010), and rotation information criterion (RIC). Moreover, `huge` provides a likelihood-based extended Bayesian information criterion (EBIC), as proposed in (FOYGEL; DRTON, 2010).

Using cross-validation to choose the tuning parameter α could be very timing consuming as it requires computing the solutions over a full regularization path. On the other hand, the procedure proposed by Liu and Wang (2012), referred to as the *Tuning-Insensitive Graph Estimation and Regression* (TIGER), has a tuning-insensitive property in the sense that it automatically adapts to the unknown sparsity pattern and is asymptotically tuning-free. So, in the finite sample settings, it may require less effort on tuning the regularization parameter. However, in practice, this procedure has a slower convergence when compared to glasso and QUIC. The TIGER methods are available in the recent `R` package `flare` (LI et al., 2014; LI et al., 2015).

2.7 Elementwise Regularization

Elementwize or componentwize regularization aims to shrink individual entries of the sample covariance matrix. Popular examples are *banding* (BICKEL; LEVINA, 2008b), *tapering* (BICKEL; LEVINA, 2008b; CAI et al., 2010) and *thresholding* (BICKEL; LEVINA, 2008a; KAROUI, 2008; ROTHMAN; LEVINA; ZHU, 2009; ROTHMAN; LEVINA; ZHU, 2010; CAI; LIU, 2011). These covariance estimators require minimal amount of computation, and regularize both eigenvalues and eigenvectors. However, due to their emphasis on elementwise transformations, such estimators are not guaranteed to be positive definite (POURAHMADI, 2011).

Given a $p \times p$ sample covariance matrix $\boldsymbol{S} = (s_{ij})$ and any integer k, $0 \leq k < p$, Bickel and Levina (2008b) defined its k-banded version by

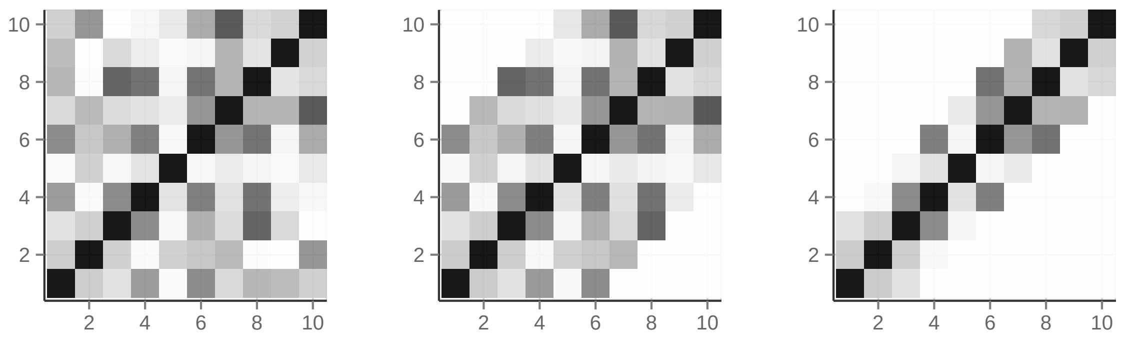
$$\boldsymbol{B}_k(\boldsymbol{S}) = \begin{cases} s_{ij}, & \text{if} \quad |i-j| \leq k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.57}$$

which is a sparse estimator for $\boldsymbol{\Sigma}$. Banding starts estimating the covariance matrix by a diagonal matrix and then successively adds the other k sub-diagonals, as shown in Figure 2.7. This kind of regularization is ideal in the situation where the indexes have been arranged in such a way that in $\boldsymbol{\Sigma} = (\sigma_{ij})$

$$|i-j| > k \Rightarrow \sigma_{ij} = 0. \tag{2.58}$$

This is equivalent to have a natural ordering among variables in $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p]$ so that those farther apart in the ordering are less correlated or have no correlation, like in time series and longitudinal data.

One problem with an k-banded covariance estimate $\boldsymbol{B}_k(\boldsymbol{S})$ is the lack of assured positive definiteness. The idea of banding and regularizing the lower triangular matrix of the modified Cholesky decomposition have been studied in (WU; POURAHMADI, 2003; HUANG et al., 2006; BICKEL; LEVINA, 2008b) to estimate $\boldsymbol{\Sigma}^{-1}$, and then in



(a) 10×10 sample covariance estimate

(b) 5-banded estimate

(c) 2-banded estimate

Figure 2.7 – Banding estimation of the covariance matrix of a 10-variate synthetic data. Zero entries are represented by white pixels while non-zero ones are represented by proportionally gray scaled pixels.

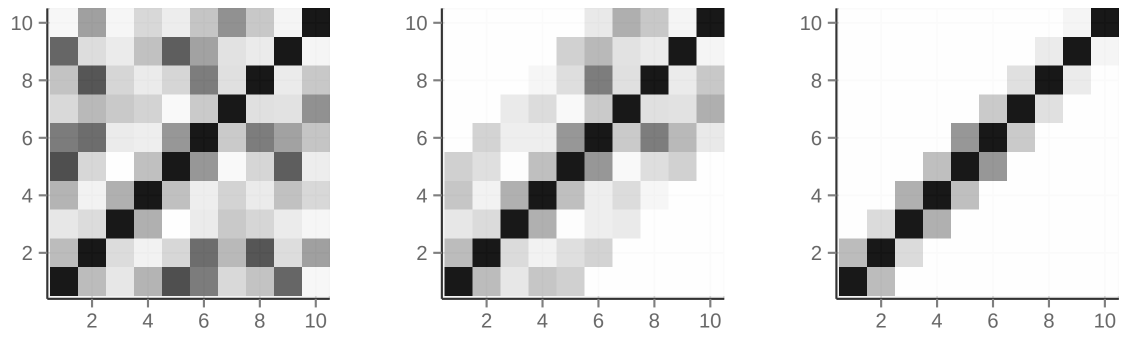
(ROTHMAN; LEVINA; ZHU, 2010) to estimate $\mathbf{\Sigma}$, which is effective in guaranteeing the positive-definiteness of the estimated covariance matrix. Bickel and Levina (2008b) have shown that, for normally distributed data, the banded estimator is consistent in the operator norm (spectral norm). The consistency in operator norm guarantees the consistency of principal component analysis (JOHNSTONE; LU, 2009) and other related methods in multivariate statistics when N is small and p is large.

Banding is a special case of tapering the covariance matrix. A tapered estimator of $\mathbf{\Sigma}$ for a tapering matrix $\boldsymbol{R} = (r_{ij})$ replaces the empirical estimate $\boldsymbol{S} = (s_{ij})$ by

$$\boldsymbol{S}_R = \boldsymbol{S} * \boldsymbol{R} = (s_{ij} r_{ij}), \tag{2.59}$$

where $(*)$ denotes the Schur (coordinate-wise) matrix multiplication. When $\boldsymbol{R}$ is a positive-definite symmetric matrix, then $\boldsymbol{S}_R$ as the Schur product of two positive-definite matrices is guaranteed to be positive definite (FURRER; BENGTSSON, 2007). Banding corresponds to using $\boldsymbol{R} = (r_{ij}) = (\mathbf{1}(|i-j| \leq k))$, which is not a positive-definite matrix. Tapering is a smooth version of banding where it gradually shrinks the off-diagonal entries toward zero. The choice of a smoother positive-definite tapering matrix $\boldsymbol{R}$ with off-diagonal entries gradually decaying to zero will ensure positive-definiteness as well as optimal rate of convergence of the tapered estimator. Cai et al. (2010) studied the asymptotic properties of the following tapering estimator

$$\boldsymbol{T}_k(\boldsymbol{S}) = (s_{ij} r_{ij}), \quad r_{ij} = \begin{cases} 1, & \text{when} \quad |i-j| \leq k/2, \\ \left(2 - \dfrac{2|i-j|}{k}\right), & \text{when} \quad k/2 < |i-j| \leq k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.60}$$



(a) 10×10 sample covariance estimate (b) Tapering with $k = 5$ (c) Tapering with $k = 2$

Figure 2.8 – Covariance matrix estimation with tapering for a 10-variate synthetic data. Zero entries are represented by white pixels while non-zero ones are represented by proportionally gray scaled pixels.

where the integer $0 \leq k < p$ is the tapering parameter, which controls the rate of decay of the sample covariance entries s_{ij} as one moves away from the main diagonal. They empirically showed that their proposed estimator is consistent in the operator norm, has good numerical performance, and nearly uniformly outperforms the banding estimator of (BICKEL; LEVINA, 2008b). Figure 2.8 shows an example of tapering the sample covariance matrix.

However, banding and tapering are not permutation-equivariant. These approaches require a natural ordering among the variables and assume that variables farther apart in the ordering are less correlated. They are appropriate for time series, longitudinal and spectroscopic data, or in situations where there is a metric on the variable indexes. In applications where the variables are not ordered, a common situation in classification problems, covariance estimators are required to be invariant under variable permutations. Regularizing a covariance estimate by applying the same function to each of the entries of the matrix leads to permutation-equivariant estimators (KAROUI, 2008). In particular, the idea of thresholding individuals entries of $\boldsymbol{S}$ has been used in the estimation of large covariance matrices. In high dimensions, it is plausible that many elements of the population covariance matrix are small or zero, and hence $\boldsymbol{\Sigma}$ is sparse (POURAHMADI; WU, 2013).

For a sample covariance matrix $\boldsymbol{S} = (s_{ij})$, the hard-thresholding operator $\boldsymbol{T}_\alpha$ with penalty parameter $\alpha \geq 0$ is defined by (BICKEL; LEVINA, 2008a):

$$\boldsymbol{T}_\alpha(\boldsymbol{S}) = \begin{cases} s_{ij}, & \text{if} \quad |s_{ij}| \geq \alpha, \\ 0, & \text{otherwise.} \end{cases} \tag{2.61}$$

so that thresholding $\boldsymbol{S}$ at α amounts to replacing by zero all entries with absolute value less than α. Such estimator preserves symmetry and is permutation-equivariant with desirable asymptotic properties. Just as in banding, (BICKEL; LEVINA, 2008a) have established the consistency of the threshold estimator in the operator norm, provided that $\log(p)/N \to 0$. An important consequence of this result is that a threshold estimator will be positive definite with probability tending to one for large N and p.

Rothman, Levina and Zhu (2009) introduced a broader class of generalized thresholding operators that combine thresholding with shrinkage, including the soft-thresholding estimator,

$$\boldsymbol{T}_{\alpha}^{(\text{soft})}(\boldsymbol{S}) = \left(\text{sign}(s_{ij})(|s_{ij} - \alpha|)_{+}\right), \tag{2.62}$$

which empirically outperformed the hard thresholding estimator under operator norm loss in the experiments carried out in (ROTHMAN; LEVINA; ZHU, 2009).

Since the entries of the sample covariance matrix have different variability, a natural requirement is to use thresholding rules with entry-dependent thresholds which automatically adapt to the variability of the individual elements of $\boldsymbol{S}$. More recently, Cai and Liu (2011) proposed to use an adaptive thresholding estimator

$$\boldsymbol{T}_{\alpha_{11},\ldots,\alpha_{pp}}^{(\text{adapt})}(\boldsymbol{S}) = \left(s_{ij}\mathbf{1}(|s_{ij}| \geq \alpha_{ij})\right), \tag{2.63}$$

where the individual thresholds α_{ij} are fully data-driven and adapt to the variability of individual entries of the sample covariance matrix $\boldsymbol{S}$. They can be estimated as follows

$$\alpha_{ij} = \alpha_{ij}(\delta) = \delta\sqrt{\frac{\theta_{ij}\log p}{N}}, \tag{2.64}$$

where $\delta > 0$ is a regularization parameter. It can be fixed at $\delta = 2$ or can be chosen through cross-validation. In practise, $\theta_{ij} = \text{var}\left((x_{ni} - \mu_i)(x_{nj} - \mu_j)\right)$ are typically unknown, but can be well estimated by

$$\hat{\theta}_{ij} = \frac{1}{N}\sum_{n=1}^{N}\left[(x_{ni} - \overline{x}_i)(x_{nj} - \overline{x}_j) - s_{ij}\right]^2, \quad \overline{x}_i = \frac{1}{N}\sum_{n=1}^{N} x_{ni}. \tag{2.65}$$

The thresholding covariance estimator, besides being permutation-equivariant, has nice asymptotic properties for estimating sparse large covariance matrices. However, as pointed out by Xue, Ma and Zou (2012), it often has negative eigenvalues when used in real data analysis.

The elementwise regularized covariance estimators require a minimal amount of computation, except in the cross-validation procedure for selecting the tuning parameters. The performance of such estimators depends heavily on the quality of tuning parameter selection. The most common approach to select the regularization parameters is to use

K-fold cross-validation. This technique first splits data into K subsets $\{\mathcal{A}_1, \ldots, \mathcal{A}_K\}$, and then selects the tuning parameter in the regularized estimator $\hat{\mathbf{\Sigma}}(\alpha)$ as

$$\alpha_F^{(\mathrm{CV})} = \arg\min_{\alpha} \left\{ \frac{1}{K} \sum_{k=1}^{K} \|\hat{\mathbf{\Sigma}}^{(-k)}(\alpha) - \boldsymbol{S}^{(k)}\|_F^2 \right\}, \tag{2.66}$$

where $\boldsymbol{S}^{(k)}$ is the sample estimator based on $\mathcal{A}_k$, and $\hat{\mathbf{\Sigma}}^{(-k)}(\alpha)$ is the regularized estimator based on data without $\mathcal{A}_k$. Here the size of the validation set is about N/K. More details about cross-validation in elementwise covariance estimation can be found in (FANG; WANG; FENG, 2015). The `R` packages `CVTuningCov` (WANG, 2014) and `FinCovRegularization` (YAN, 2015) provide cross-validation procedures to select the tuning parameters of banding, tapering, and hard or soft thresholding.

2.8 Summary of the Regularized Estimators

This chapter presented improved estimators that regularize both the eigenvalues and eigenvectors of the sample covariance matrix. These have been obtained by minimizing certain risk functions, penalized likelihood functions, and elementwise shrinkage. Table 2.3 summarizes the regularized estimators to be used in the rest of the dissertation. Only permutation-equivariant estimators that guarantees positive-definiteness have been considered.

Table 2.3 – Permutation invariant estimators that are guaranteed to preserve symmetry and positive-definiteness.

Estimator	**Description**	**Type of Regularization**	**Parameter Tuning**	**R Packages**
$\hat{\mathbf{\Sigma}}_{\text{OAS}}$	Stein-type shrinkage estimator	Eigenvalues	Oracle approximating shrinkage (OAS) estimator	`HiDimDA`
$\hat{\mathbf{\Sigma}}_{\text{SS}}$	Stein-type shrinkage estimator	Variances and correlations	Analytic approach by Ledoit and Wolf (2003)	`corpcor` `SHIP` `ShrinkCovMat`
$\hat{\mathbf{\Sigma}}_{\text{PPCA}}$	Parsimonious Gaussian estimator	Low-rank estimation with a uniform constraint on the smallest eigenvalues	Laplace evidence	`pcaMethods`
$\hat{\mathbf{\Sigma}}_{\text{SPC}}$	Sparse eigenvectors estimator	Eigenvectors	K-fold CV and Laplace evidence	`PMA` `elasticnet` `spca`
$\hat{\mathbf{\Sigma}}_{\text{QUIC}}$	Penalized likelihood sparse estimator	Eigenvalues and eigenvectors	K-fold CV	`glasso` `huge` `QUIC`

3 Regularized Gaussian Bayes Classifiers

In this chapter, the improved covariance matrix estimators discussed above are applied to perform accurate classification in the high-dimensional setting. A family of regularized Gaussian generative classifiers is proposed, and empirically shown to outperform the Gaussian Naive Bayes (GNB) classifier in low, medium and high-dimensional data settings.

3.1 Generative Classification

Consider the problem of predicting a label $y \in \{C_1, \ldots, C_K\}$ on the basis of a vector of continuous real valued features $\boldsymbol{x} = (x_1, \ldots, x_p) \in \mathbb{R}^p$. A generative classifier (NG; JORDAN, 2002; MITCHELL, 2015) learns a model of the joint probability $\boldsymbol{P}(\boldsymbol{x}, y)$, of the inputs $\boldsymbol{x}$ and the label y, and makes its predictions by using Bayes rule to calculate $\boldsymbol{P}(y|\boldsymbol{x})$, and then picking the most likely label y. More precisely, applying Bayes rule, the posterior probability $\boldsymbol{P}(y = C_k|\boldsymbol{x})$ for the class C_k can be written as

$$\boldsymbol{P}(y = C_k|\boldsymbol{x}) = \frac{\boldsymbol{P}(\boldsymbol{x}|y = C_k)\boldsymbol{P}(y = C_k)}{\sum_{j=1}^{K} \boldsymbol{P}(\boldsymbol{x}|y = C_j)\boldsymbol{P}(y = C_j)} \tag{3.1}$$

where C_k denotes the k^{th} possible value for y, and the summation in the denominator is over all legal values of y. The distribution $\boldsymbol{P}(y)$ gives the prior probability of y and $\boldsymbol{P}(\boldsymbol{x}|y)$ models the class-conditional density or the likelihood of observing $\boldsymbol{x}$ given y. Both of these distributions can be estimated using training data. One can then use these estimates, together with Bayes rule above, to determine $\boldsymbol{P}(y|\boldsymbol{x} = \boldsymbol{x}_{\text{new}})$ for any new unlabeled instance $\boldsymbol{x}_{\text{new}}$. The corresponding classifier, the *Bayes optimal classifier*, is the function that assigns a class label $\hat{y}$ to $\boldsymbol{x}_{\text{new}}$ based on the following classification rule:

$$\hat{y} = \arg\max_{C_k} \boldsymbol{P}(y = C_k)\boldsymbol{P}(\boldsymbol{x} = \boldsymbol{x}_{\text{new}}|y = C_k),\ k \in \{1, \ldots, K\} \tag{3.2}$$

which is known as the maximum a posteriori or MAP decision rule. This rule suppresses the denominator of (3.1) since, for a given observation $\boldsymbol{x}_{\text{new}}$, it is a constant. In other words, the optimal Bayes decision rule is to choose the class presenting the maximum posterior probability, given the particular observation at hand. It is also common to see the logarithmic version of the MAP criterion:

$$\hat{y} = \arg\max_{C_k} \{\log \boldsymbol{P}(y = C_k) + \log \boldsymbol{P}(\boldsymbol{x} = \boldsymbol{x}_{\text{new}}|y = C_k)\},\ k \in \{1, \ldots, K\} \tag{3.3}$$

which makes more clear that the MAP decision rule tries to reach a compromise between the *a priori* expectations carried by $\boldsymbol{P}(y)$ and the evidence provided by the data via the likelihood function $\boldsymbol{P}(\boldsymbol{x}|y)$ (FIGUEIREDO, 2004).

When dealing with continuous data, a typical assumption is that the class-conditional densities are Gaussian (DUDA; HART; STORK, 2001) (see Appendix A). Thus the density for the class C_k is given by (BISHOP, 2006):

$$\boldsymbol{P}(\boldsymbol{x}|y=C_k) = \frac{1}{\sqrt{(2\pi)^p|\boldsymbol{\Sigma}_k|}}\exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right\}. \tag{3.4}$$

Given a set of training pairs $\mathcal{S} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ with N_k independent observations of the class C_k, the maximum likelihood estimators (MLEs) for $\boldsymbol{P}(y=C_k)$, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are given, respectively, by:

$$\hat{\pi}_k = \frac{N_k}{N}, \tag{3.5}$$

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k}\sum_{n=1}^{N_k}\boldsymbol{x}_n, \tag{3.6}$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k}\sum_{n=1}^{N_k}(\boldsymbol{x}_n-\hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_n-\hat{\boldsymbol{\mu}}_k)^T. \tag{3.7}$$

considering only the observations $\boldsymbol{x}_n$'s of class C_k, that is, $y_n = C_k$ for $n = 1, \ldots, N_k$.

The maximum likelihood estimate of the covariance matrix requires the estimation of $p(p+1)/2$ parameters, p diagonal elements and $p(p-1)/2$ independent off-diagonal elements. This estimate can be seen as the sum of $N-1$ independent p-by-p rank-one matrices, and thus is guaranteed to be singular if $N \leq p$ (DUDA; HART; STORK, 2001). Since $\hat{\boldsymbol{\Sigma}}_k$ must be inverted in (3.4) to obtain the class-conditional densities, there is an algebraic requirement for at least $p+1$ training observations. Thus, in the "large p, small N" data setting, one must apply a constraint over the number of parameters of the covariance matrix to ensure the existence of its inverse.

The Gaussian Bayes classifier with unrestricted covariance structure, the Unrestricted Gaussian Bayes (UGB) classifier, suffers from severe drawbacks. At first, for large p, since the number of parameters is quadratic with the data dimension, the UGB classifier tends to overfit (MOORE, 2001). Moreover, the covariance estimate is subjected to all the problems pointed out on Chapter 2. Finally, when $p \geq N$, the UGB decision rule can be no longer evaluated.

3.2 The Gaussian Naive Bayes Classifier

Given the intractable sample complexity for learning the UGB classifier, the Gaussian Naive Bayes (GNB) model tries to minimize this complexity by making a conditional independence assumption that dramatically reduces the number of covariance parameters when modeling $\boldsymbol{P}(\boldsymbol{x}|y)$, from the original $p(p+1)/2$ to just p, the p diagonal elements. So, the GNB is equivalent to a Gaussian Bayes classifier with diagonal covariance matrix (DEMPSTER; LAIRD; RUBIN, 1977).

More precisely, in the naive Bayes approach (also known as *simple Bayes* and *Idiot's Bayes*) each feature x_i is assumed to be conditionally independent of every other feature x_j for $j \neq i$, given the class label. That is,

$$\boldsymbol{P}(\boldsymbol{x}|y) = \prod_{i=1}^{p} \boldsymbol{P}(x_i|y). \tag{3.8}$$

With this assumption and using Bayes rule (3.1), the Bayes optimal classifier in (3.2), for a given observation $\boldsymbol{x}_{\text{new}} = (x_1^{\text{new}}, \ldots, x_p^{\text{new}})$, can be simplified to

$$\hat{y} = \arg\max_{C_k} \boldsymbol{P}(y = C_k) \prod_{i=1}^{p} \boldsymbol{P}(x_i = x_i^{\text{new}}|y = C_k),\ k \in \{1, \ldots, K\}. \tag{3.9}$$

One common approach for computing the probabilities $\boldsymbol{P}(x_i = x_i^{\text{new}}|y)$ is to assume that for each possible discrete value C_k of y, the distribution of each continuous x_i is Gaussian, and is defined by a mean μ_{ki} and standard deviation σ_{ki} specific to x_i and C_k:

$$\boldsymbol{P}(x_i|y = C_k) = \frac{1}{\sigma_{ki}\sqrt{2\pi}} \exp\left\{-\frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2}\right\}. \tag{3.10}$$

Again, the maximum likelihood approach can be used to estimate these parameters from training data:

$$\hat{\mu}_{ki} = \frac{1}{N_k} \sum_{n=1}^{N_k} x_{ni}, \tag{3.11}$$

$$\hat{\sigma}_{ki} = \sqrt{\frac{1}{N_k} \sum_{n=1}^{N_k} (x_{ni} - \hat{\mu}_{ki})^2}, \tag{3.12}$$

with all x_{ni}'s in class C_k. The assumption that the values of the attributes are conditionally independent given the class label dramatically reduces the complexity of learning the class posterior probability density function. When this assumption is met, the naive Bayes classifier outputs the optimal Bayes classification (MITCHELL, 1997).

However, despite its surprisingly good performance in classification (MICHIE et al., 1994; CARUANA; NICULESCU-MIZIL, 2006), specially in the high-dimensional data setting (CARUANA; KARAMPATZIAKIS; YESSENALINA, 2008), evidence has been found that the GNB classifier produces poor probability estimates (BENNETT, 2000). Because attributes tend to be positively correlated, the predicted posterior probabilities $\boldsymbol{P}(y = C_k|\boldsymbol{x})$ are typically too extreme: near 0 when $\hat{y} \neq C_k$ or near 1 when $\hat{y} = C_k$ (ZADROZNY; ELKAN, 2001). Moreover, in practice, naive Bayes' attribute independence assumption is often violated, and as a result its prediction accuracy is often suboptimal.

A large literature addresses approaches to reducing the inaccuracies that result from the conditional independence assumption. According to Zaidi et al. (2013), such approaches can be placed into two categories. The first category comprises *semi-naive Bayes* methods, which enhance naive Bayes' accuracy by relaxing the assumption of conditional independence between attributes given the class label (KOHAVI, 1996; FRIEDMAN; GEIGER; GOLDSZMIDT, 1997; KEOGH; PAZZANI, 1999; ZHENG; WEBB, 2000; FRANK; HALL; PFAHRINGER, 2003; WEBB; BOUGHTON; WANG, 2005; JIANG; ZHANG; CAI, 2009; WEBB et al., 2012; ZHENG et al., 2012) or by applying naive Bayes over a new feature set achieved via feature selection, elimination and joining techniques (KONONENKO, 1991; LANGLEY; SAGE, 1994; PAZZANI, 1998; RATANAMAHATANA; GUNOPULOS, 2003; BOULLÉ, 2007; ZHANG; nA; ROBLES, 2009) or via feature extraction techniques (FAN; POH, 2007). The second category comprises *attribute weighting* methods (FERREIRA; DENISON; HAND, 2001; HALL, 2007; LEE; GUTIERREZ; DOU, 2011; ZAIDI et al., 2013), which add a weight to each attribute when computing the class posterior probabilities. However, most of these approaches require the domain of the attribute values to be discrete.

This work presents a novel approach to reduce the effects of the violations of the attribute independence assumption on which naive Bayes is based. A Regularized Gaussian Bayes (RGB) algorithm is introduced, that considers the correlation structure among variables to learn the class posterior probabilities. The proposed RGB classifier is similar to the UGB model, but avoids overfitting by replacing the unrestricted covariance estimate (3.7) with well-conditioned regularized ones, as suggested in Chapter 2. So, RGB aims to find the best trade-off between non-naivety (high variance) and prediction accuracy (low bias).

3.3 The Regularized Gaussian Bayes Classifier

The RGB classifier provides a intermediate approach that is less constraining than the global assumption of conditional independence made by the GNB classifier, but more tractable than assuming an unrestricted correlation structure among variables. The regularized covariance estimators allow modeling attribute dependence, but shrinking model complexity by: penalizing covariance matrix eigenvalues through James-Stein-type shrinkage estimates ($\hat{\boldsymbol{\Sigma}}_{\text{OAS}}$ and $\hat{\boldsymbol{\Sigma}}_{\text{SS}}$), reducing the number of covariance matrix parameters to be fitted through parsimonious estimates ($\hat{\boldsymbol{\Sigma}}_{\text{PPCA}}$), or assuming independence between some variables through sparse estimates ($\hat{\boldsymbol{\Sigma}}_{\text{SPC}}$ and $\hat{\boldsymbol{\Sigma}}_{\text{QUIC}}$).

Figures 3.1 and 3.2 compare the decision boundaries of the GNB, UGB and RGB (with the estimator $\hat{\boldsymbol{\Sigma}}_{\text{OAS}}$) classifiers for two types of problems with different complexities: a linearly separable problem (two Gaussians) and a nonlinearly separable problem (XOR problem). The datasets have been generated using the R package `mlbench` (LEISCH; DIMITRIADOU, 2010) through the functions `mlbench.2dnormals` and `mlbench.xor`, respectively. The classifiers were trained using 50 (25 from each class) and 200 (100 from each class) randomly selected examples, respectively, from a set of 1,000 data points generated for each problem. The points plotted on the decision boundaries are the held out observations. One can note that RGB generates an adequate response given the complexity of the problem. It is smoother than UGB when predicting the linearly separable classes, and is more complex than GNB when predicting the nonlinearly separable classes.

In order to make RGB even more robust, specially in the high-dimensional data setting, two other regularization strategies are employed in the form of James-Stein-type shrinkage estimators (JAMES; STEIN, 1961):

1. Regularization of the maximum likelihood class priors $\hat{\pi}_k$ in (3.5), using the shrinkage estimator of Hausser and Strimmer (2009);

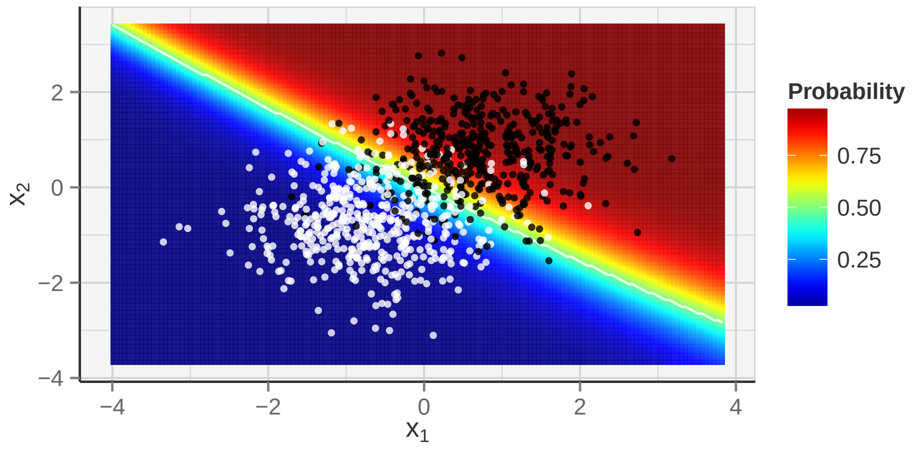
2. Regularization of the sample means $\hat{\boldsymbol{\mu}}_k$ in (3.6), using the shrinkage estimator of DeMiguel, Martin-Utrera and Nogales (2013).

Such estimators are obtained by shrinking the maximum likelihood estimators (3.5) and (3.6) towards target estimators. The advantage is that while the shrinkage target is usually biased, it also contains less variance than the sample estimator. Thus, under general conditions, there exists a shrinkage intensity for which the resulting shrinkage estimator contains less estimation error than the original sample estimator (JAMES; STEIN, 1961).

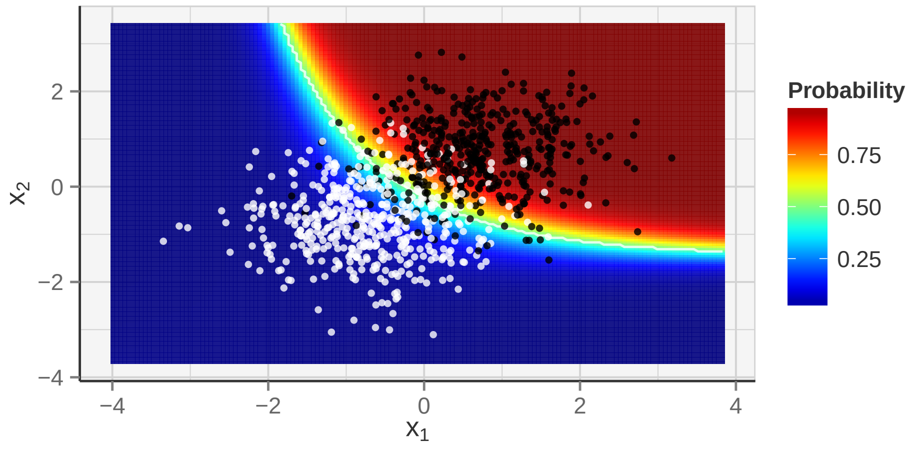
Motivated by the James-Stein principle, Hausser and Strimmer (2009) derived a shrinkage estimator that provides effective frequency estimates of discrete variables from small-sample data:

$$\hat{\pi}_k^{\text{Shrink}} = \alpha_1 t_k^{\text{prior}} + (1 - \alpha_1)\hat{\pi}_k^{\text{ML}}, \tag{3.13}$$

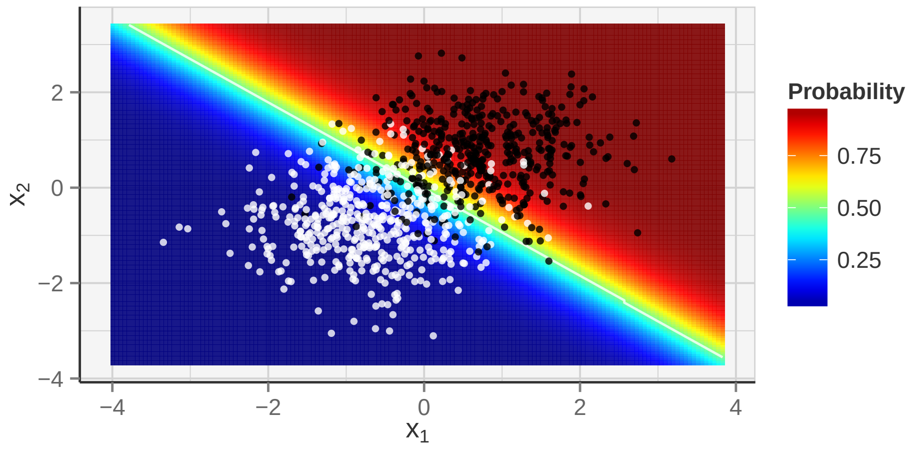
where $\alpha_1 \in [0, 1]$ is the shrinkage intensity that takes on a value between 0 (no shrinkage) and 1 (full shrinkage), and t_k^{prior} is the shrinkage target. A convenient choice of t_k^{prior} is the uniform distribution $t_k^{\text{prior}} = 1/K$, which is the maximum entropy target. Hausser and



(a) GNB decision boundary.

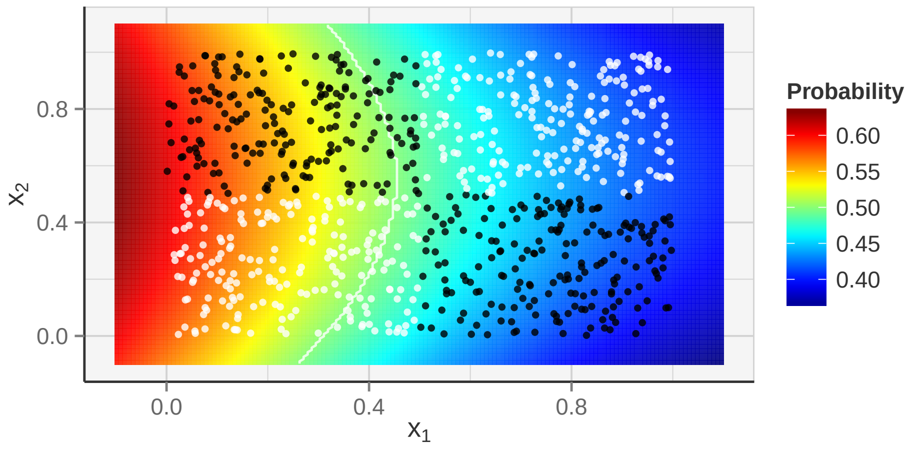


(b) UGB decision boundary.

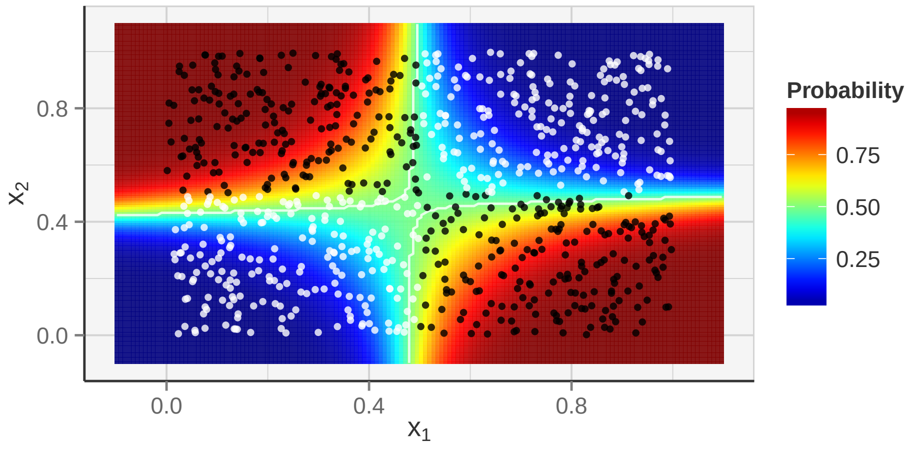


(c) RGB decision boundary.

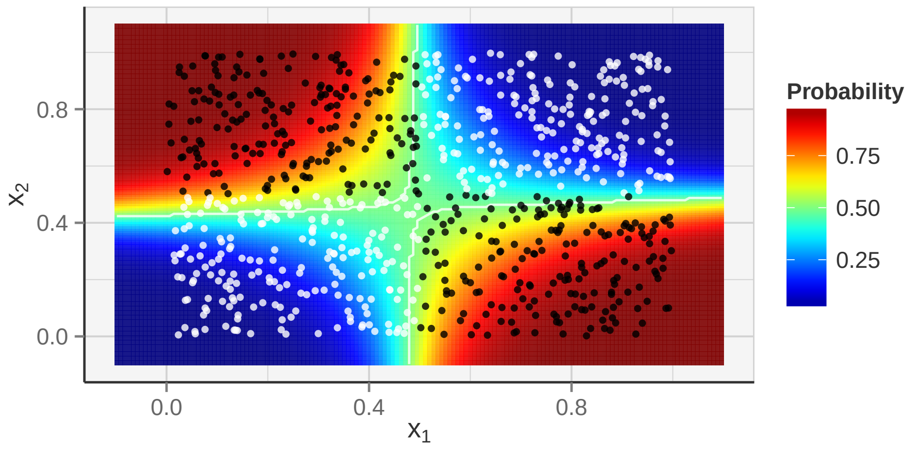
Figure 3.1 – Classification decision boundary for the two-class 2D Gaussian problem. The classifiers were trained using 50 data points, 25 from each class. The color-coded plot indicates the posterior probability for the "black" class.



(a) GNB decision boundary.



(b) UGB decision boundary.



(c) RGB decision boundary.

Figure 3.2 – Classification decision boundary for the 2D XOR problem. The classifiers were trained using 200 data points, 100 from each class. The color-coded plot indicates the posterior probability for the "black" class.

Strimmer (2009) suggest to calculate the optimal shrinkage intensity analytically, using the following estimator:

$$\hat{\alpha}_1 = \min \left\{ \frac{1 - \sum_{k=1}^{K} (\hat{\pi}_k^{\text{ML}})^2}{(N-1) \sum_{k=1}^{K} (t_k^{\text{prior}} - \hat{\pi}_k^{\text{ML}})^2}, 1 \right\}. \tag{3.14}$$

The frequency estimates obtained by plugging $\hat{\alpha}_1$ into (3.13) are used by Hausser and Strimmer (2009) to derive shrinkage estimators for entropy, mutual information and related quantities. Such estimators are available in the `R` package `entropy` (HAUSSER; STRIMMER, 2014). Here, the estimator (3.13) will be adopted to shrink the class prior probability estimates, as proposed in (AHDESMÄKI; STRIMMER et al., 2010) to learn the LDA (Linear Discriminant Analysis) discriminant function. The later is implemented in the `R` package `sda` (AHDESMAKI et al., 2015).

DeMiguel, Martin-Utrera and Nogales (2013) consider a shrinkage estimator for the vector of means that is a weighted average of the sample or maximum likelihood mean (3.6) and the shrinkage target $\boldsymbol{t}_k^{\text{mean}} = \mathbf{1}(1/p) \sum_{j=1}^{p} \hat{\mu}_{kj}$, where $\mathbf{1}$ is a $1 \times p$ vector of ones. Such estimator can be equivalently defined as the optimal convex combination between the sample estimator and the shrinkage target:

$$\hat{\boldsymbol{\mu}}_k^{\text{Shrink}} = \alpha_2 \boldsymbol{t}_k^{\text{mean}} + (1 - \alpha_2) \hat{\boldsymbol{\mu}}_k^{\text{ML}}, \tag{3.15}$$

where $\alpha_2 \in [0, 1]$ is the shrinkage intensity that determines the "strength" with which the sample estimator is shrunk towards the shrinkage target. The optimal value of α_2 that minimizes the expected quadratic loss is:

$$\hat{\alpha}_2 = \min \left\{ \frac{\bar{\sigma}^2}{\bar{\sigma}^2 + (N/p) \| \boldsymbol{t}_k^{\text{mean}} - \hat{\boldsymbol{\mu}}_k^{\text{ML}} \|_2^2}, 1 \right\} \tag{3.16}$$

where $\bar{\sigma}^2 = (1/p) \text{tr} \left(\hat{\boldsymbol{\Sigma}}_k^{\text{ML}} \right)$. One can observe that the shrinkage intensity increases with the number of variables p, and decreases with the number of observations N.

Once well-conditioned estimators for (3.5), (3.6) and (3.7) have been established, a more robust regularized Gaussian Bayes classifier for continuous input variables can be obtained, which is more accurate than GNB and performs well in high-dimensional data settings. In summary, the proposed RGB algorithm is given in Algorithm 3.1.

In order to illustrate the performance of the RGB classifier in the "small N, large p" setting, four high-dimensional classification problems, with $p \gg N$, have been considered: DBWorld ($N = 64$, $p = 4702$, $K = 2$) from the UCI repository (LICHMAN, 2013), Arcene ($N = 100$, $p = 10000$, $K = 2$) from the NIPS Challenge (GUYON et al., 2004), Leukemia ($N = 72$, $p = 7129$, $K = 2$) (GOLUB et al., 1999), and Sorlie ($N = 85$, $p = 456$, $K = 5$) (SØRLIE et al., 2001). The last two are available in the `R` package `datamicroarray` (RAMEY, 2013). For each problem, the features have been ordered beforehand according

Algorithm 3.1: Regularized Gaussian Bayes (RGB) Classifier

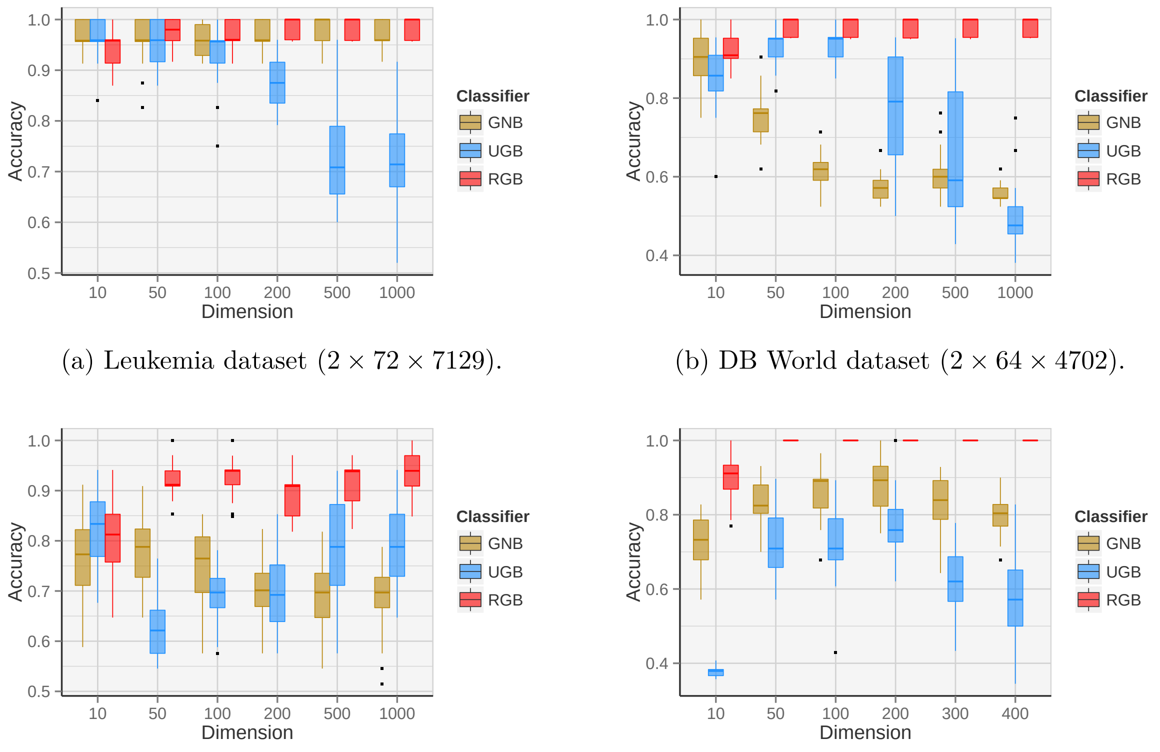
```
1. Let $\boldsymbol{y} \in \{C_1, \dots, C_k, \dots, C_K\}$ be a $N \times 1$ categorical variable and
   $\boldsymbol{X}_k = \{\boldsymbol{x}_n\}_{n=1}^{N_k}$ be a $N_k \times p$ matrix containing the $N_k$ observations in class $C_k$.
2. For $k = 1, \dots, K$, fit a Gaussian model to the class $C_k$ as follows:
    a) Compute the class prior probability $\hat{\pi}_k$ using (3.13);
    b) Compute the vector of means $\hat{\boldsymbol{\mu}}_k$ using (3.15);
    c) Compute the covariance matrix $\hat{\boldsymbol{\Sigma}}_k$ using one of those improved
       estimators listed in Table 2.3.
3. For a new unlabeled observation $\boldsymbol{x}_{\text{new}}$, assign a label $\hat{y}$ as follows:
    a) Compute the class-conditional densities $\boldsymbol{P}(\boldsymbol{x}_{\text{new}}|y = C_k)$ using (3.4);
    b) Predict the class label according to the MAP decision rule (3.2).
```

to their correlation with the class labels, measured by the Maximal Information Coefficient (MIC[1]) (RESHEF et al., 2011). Then the algorithms UGB, GNB and RGB were evaluated under the $m \in \{10, 50, 100, 200, 300, 400, 500, 1000\}$ best ranked features, in an attempt to assess the effect of increasing data dimension on the test prediction accuracy. The nearest positive-definite matrix approximation method of Higham (1988) has been adopted to ensure non-singularity of the UGB covariance matrix. The test classification accuracy of the algorithms was estimated doing 6 runs of 5-fold cross-validation, generating a sample of 30 results. The boxplots in Figure 3.3 show the distribution of these results. One can observe that, in general, the predictive performance of GNB and UGB decreases with increasing m, while the performance of RGB is surprisingly robust to an increase in the number of features, even when $p \gg N$.

3.4 Improving Classification Accuracy with Adaptive Boosting

Boosting is one of the most powerful learning ideas introduced in the last twenty years (HASTIE; TIBSHIRANI; FRIEDMAN, 2009a; FERREIRA; FIGUEIREDO, 2012; SCHAPIRE; FREUND, 2012). The motivation for boosting was a procedure that combines the outputs of many *weak* classifiers to produce a powerful *committee.* A *weak learner* (WL) is a learning algorithm capable of producing classifiers with probability of error strictly (but only slightly) less than that of random guessing (0.5, in the binary case). On the other hand, a *strong learner* (SL) is able (given enough training data) to yield classifiers with

[1] MIC was touted by Speed (2011) as a "correlation for the 21st century". The `R` package `minerva` (FILOSI; VISINTAINER; ALBANESE, 2014) is a wrapper for the `C` engine `cmine`, an implementation of Maximal Information-Based Non-parametric Exploration (MINE) statistics that allows computing MIC.



(a) Leukemia dataset ($2 \times 72 \times 7129$).

(b) DB World dataset ($2 \times 64 \times 4702$).

(c) Arcene dataset ($2 \times 100 \times 10000$).

(d) Sorlie dataset ($5 \times 85 \times 456$).

Figure 3.3 – Classification experiment with high-dimensional datasets varying the number of features, which have been ordered beforehand using the MIC statistic.

arbitrarily small error probability. In general, it is easier to train several simple classifiers and combine them into a more complex classifier than to learn a single complex classifier. So, the strategy of boosting is to learn many weak classifiers and combine them in some way, instead of trying to learn a single strong classifier.

The most popular boosting algorithm is AdaBoost (adaptive boosting) (FREUND; SCHAPIRE et al., 1996; FREUND; SCHAPIRE, 1997), commonly referred to as “AdaBoost.M1” or “Discrete AdaBoost”. In fact, Breiman (2004) referred to the AdaBoost with decision trees as “the most accurate general purpose classification algorithm available”. Algorithm 3.2 shows the details of the AdaBoost.M1 algorithm in the two-class classification setting. The AdaBoost procedure takes as input a set of training examples $\mathcal{S} = \{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$, where $\boldsymbol{x}_n$ is a vector valued feature and $y_n \in \{-1, 1\}$ is the associated class label, and outputs $F(\boldsymbol{x}_{\text{new}}) = \sum_{m=1}^{M} c_m f_m(\boldsymbol{x}_{\text{new}})$ for any new unseen instance $\boldsymbol{x}_{\text{new}}$, with the corresponding label prediction given by $\operatorname{sign}(F(\boldsymbol{x}_{\text{new}}))$. Each $f_m(\boldsymbol{x})$ is a classifier producing values ± 1 and c_m is a constant that measures the importance assigned to $f_m(\boldsymbol{x})$. AdaBoost trains the classifiers $f_m(\boldsymbol{x})$ on modified versions of the training sample. The data modifications at each step consist of applying weights $w_1, w_2, \ldots, w_N$ to each of the training observations. Initially all of the weights are set to $w_n = 1/N$, so that the first step simply trains the classifier on the data in the usual manner. For each successive iteration

$m = 2, 3, \ldots, M$ the weights are individually modified according to the (mis)classification of the related observation by the previous classifiers. Thus as iterations proceed, observations that are difficult to classify correctly receive ever-increasing influence. Each successive classifier is thereby forced to concentrate on those training observations that are missed by previous ones in the sequence. The more accurate the base classifier $f_m(\boldsymbol{x})$, the more importance is assigned to it by increasing its contribution c_m to the committee response.

Algorithm 3.2: AdaBoost.M1

```
1. Initialize the observation weights $w_n = 1/N$, $n = 1, \ldots, N$.
2. For $m = 1$ to $M$:
    a) Fit a classifier $f_m(\boldsymbol{x})$ to the training data using weights $w_n$.
    b) Compute the weighted training classification error $\epsilon_m$:
           $\epsilon_m = \sum_{i:\, f_m(\boldsymbol{x}_i) \neq y_i} w_i.$
    c) Compute the classifier importance $c_m$:
           $c_m = \gamma \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right),$
       where $\gamma = 1$ is suggested by Freund, Schapire et al. (1996), but
       Breiman et al. (1998) uses $\gamma = 0.5$.
    d) For $n = 1, \ldots, N$, update the weights $w_n$:
           $w_n^{(m+1)} = w_n^{(m)} \times \begin{cases} e^{-c_m}, & \text{if } f_m(\boldsymbol{x}_n) = y_n, \\ e^{c_m}, & \text{if } f_m(\boldsymbol{x}_n) \neq y_n \end{cases}$
    e) Normalize the weights $w_n^{(m+1)}$ so that $\sum_{n=1}^{N} w_n^{(m+1)} = 1$.
3. Combine the base classifier responses through a *weighted majority vote* to produce
   the final prediction:
           $\hat{y} = \operatorname{sign}\left(\sum_{m=1}^{M} c_m f_m(\boldsymbol{x})\right).$
```

Notice that the AdaBoost algorithm can handle weak classifiers with weighted error rate larger than 0.5. By simply inverting the output of such a classifier, a weighted error rate less than 0.5 is obtained. Such an inversion is automatically performed by AdaBoost, because if $\epsilon_m > 0.5$, the corresponding weight c_m is negative. On the other hand, a base classifier that plays identically to random guessing ($\epsilon_m = 0.5$) receives a weight of zero, since it produces outputs completely at random. A perfect classifier ($\epsilon = 0$) in turn receives an infinite weight, since it is the only committee member needed. Finally, a perfect liar ($\epsilon = 1$) has its decision reversed in order to get a perfect classifier.

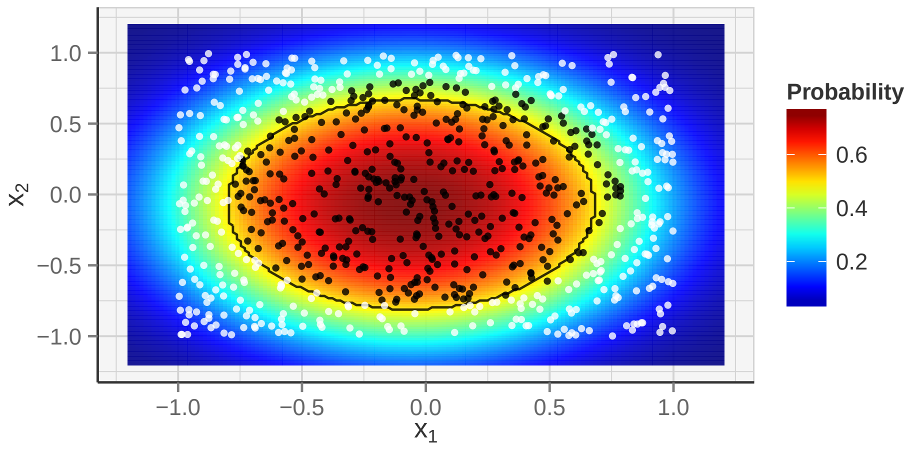
The c_m constant can be also interpreted as a learning rate calculated as a function of the error made in each step. Thus, the step-size factor $\gamma \in (0, 1]$ can be seen as a shrinkage of the learning procedure, and sometimes a small value of γ can reduce overfitting. In fact, empirical evidence was presented by Friedman (2001) that a small γ can be often beneficial and almost never yields substantially worse predictive performance of boosting estimates in the regression context. However, a smaller value of γ, such as $\gamma = 0.1$, typically requires a larger number of boosting iterations and thus more computing time. In the classification setting, the AdaBoost step-size is typically set as 1 (FREUND; SCHAPIRE et al., 1996) or 0.5 (BREIMAN et al., 1998).

As for the weights that AdaBoost calculates on the training examples, in practice, there are several ways in which these can be used by the base learner (SCHAPIRE; FREUND, 2012). In some cases, the base learner can use these weights directly. This approach, called *boosting by reweighting*, has the advantage of being direct and exact. In other cases, an unweighted training set is generated for the base learner by selecting examples at random from the original training set. The probability of selecting an example in this case is set to be proportional to the weight of the example. This approach, called boosting by resampling, is often useful when the chosen base learner cannot easily be modified to handle the given weights directly.

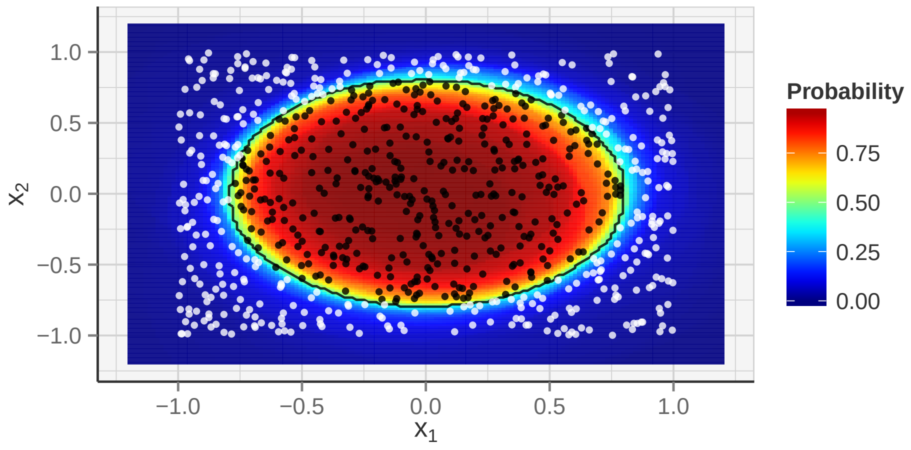
It is well known that a tree-based AdaBoost classifier consistently produces significantly lower error rates than a single decision tree (HASTIE; TIBSHIRANI; FRIEDMAN, 2009a). Moreover, for some reason, it seems that AdaBoost is resistant to overfitting (FRIEDMAN et al., 2000) and generally tends to increase the margins of all training examples (SCHAPIRE, 2013). In this work, the Adaptive Boosting procedure will be adopted to improve prediction accuracy of the RGB classifier. In addition, considering a simple modification on the committee output, the class posterior probability estimates will also be significantly improved. The proposed Boosted RGB (BRGB) algorithm for the two-class classification setting is detailed in Algorithm 3.3.

Figures 3.4 and 3.5 compare the decision boundaries of the RGB against the BRGB for two toy problems generated with the `R` package `mlbench`. The classifiers were trained using a sample of size 200 (100 instances from each class) randomly selected from a set of 1000 data points. The remaining 800 examples were then plotted on the decision boundaries. As it can be observed, in both cases the predictions produced by the RGB classifier present considerable bias, which seems to be corrected by boosting.

The RGB classifier can be easily modified to handle the instance weights directly. Such modification is achieved by simply replacing the maximum likelihood estimators (3.5), (3.6) and (3.7) by their weighted forms (BISHOP, 2006):



(a) RGB decision boundary.



(b) BRGB decision boundary.

Figure 3.4 – RGB and BRGB decision boundaries for the two-class 2D Circle problem. The classifiers were trained using 200 data points, 100 from each class. The color-coded plot indicates the posterior probability for the "black" class.

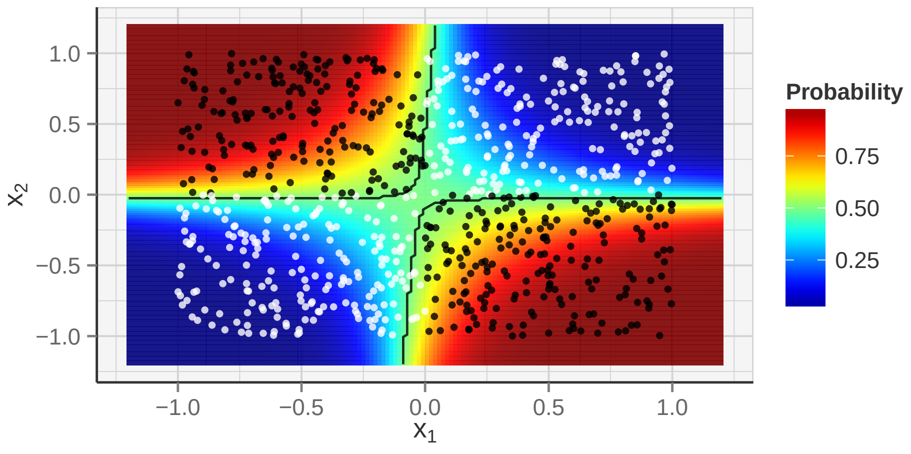
$$\hat{\pi}_k^{\mathrm{ML}} = \frac{\sum_{n=1}^{N_k} w_n}{\sum_{n=1}^{N} w_n} \tag{3.17}$$

$$\hat{\boldsymbol{\mu}}_k^{\mathrm{ML}} = \frac{\sum_{n=1}^{N_k} w_n \boldsymbol{x}_n^{(k)}}{\sum_{n=1}^{N_k} w_n}, \tag{3.18}$$

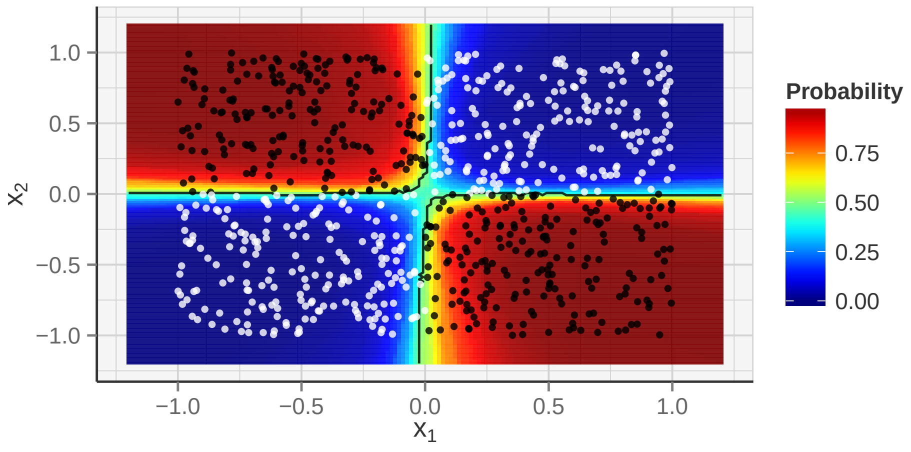
$$\hat{\boldsymbol{\Sigma}}_k^{\mathrm{ML}} = \frac{\sum_{n=1}^{N_k} w_n (\boldsymbol{x}_n^{(k)} - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_n^{(k)} - \hat{\boldsymbol{\mu}}_k)^T}{\sum_{n=1}^{N_k} w_n}. \tag{3.19}$$

followed by the regularization procedures in step 2 of Algorithm 3.1, in order to get robust estimates.

However, boosting by resampling empirically demonstrated to consistently outperform the reweighting approach for the BRGB algorithm, and will be preferred here.



(a) RGB decision boundary.



(b) BRGB decision boundary.

Figure 3.5 – RGB and BRGB decision boundaries for the two-class 2D XOR problem. The classifiers were trained using 200 data points, 100 from each class. The color-coded plot indicates the posterior probability for the "black" class.

In addition, the resampling approach has the advantage that, at each boosting iteration, some training examples in $\mathcal{S}$ are not in $\mathcal{S}^{(m)}$ (what Breiman (1997) calls out-of-bag (OOB) data) and can be used as a separate validation set to form accurate estimates of important quantities. More precisely, in the first step of boosting, sampling from the training set $\mathcal{S}$ with replacement using equal probabilities $w_n = 1/N$ (bootstrap sampling) leaves out roughly 37% of the examples in $\mathcal{S}$ (FERREIRA; FIGUEIREDO, 2012). But, as iterations proceed, those cases that have been most frequently missclassifed receives ever-increasing resampling probabilities, and thus will appear more often in the subsequent training sets $\mathcal{S}^{(m+1)}$. As a consequence, the number of left-out examples tend to grow through the boosting iterations.

The OOB estimation has gained popularity with the success of the Random Forest

Algorithm 3.3: Boosted Regularized Gaussian Bayes (BRGB) Classifier

```
1. Consider a training set $\mathcal{S} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ with instance labels $y_n \in \{0, 1\}$.
2. Initialize the observation weights $w_n = 1/N$, $n = 1, \ldots, N$.
3. For $m = 1$ to $M$:
    a) Sample $N$ observations with replacement from $\mathcal{S}$ to get the training
       set $\mathcal{S}^{(m)}$, setting weight $w_n^{(m)}$ as the probability of selecting the $n^{\text{th}}$
       observation;
    b) Fit a RGB classifier (see Algorithm 3.1) $f_m(\boldsymbol{x})$ to $\mathcal{S}^{(m)}$.
    c) (Optional) Calibrate the fitted model $f_m(\boldsymbol{x})$ using Platt Scaling or
       Isotonic Regression considering the predictions for the out-of-bag
       observations (i.e. the training examples that are not in $\mathcal{S}^{(m)}$).
    d) Compute the weighted training classification error $\epsilon_m$:
           $$\epsilon_m = \sum_{i:\, f_m(\boldsymbol{x}_i) \neq y_i} w_i.$$
    e) Compute the classifier importance $c_m$:
           $$c_m = \gamma \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right),$$
       with $\gamma$ in $(0, 1]$.
    f) For $n = 1, \ldots, N$, update the weights $w_n$:
           $$w_n^{(m+1)} = w_n^{(m)} \times \begin{cases} e^{-c_m}, & \text{if} \quad f_m(\boldsymbol{x}_n) = y_n, \\ e^{c_m}, & \text{if} \quad f_m(\boldsymbol{x}_n) \neq y_n \end{cases}$$
    g) Normalize the weights $w_n^{(m+1)}$ so that $\sum_{n=1}^N w_n^{(m+1)} = 1$.
4. Combine the base classifier responses to get a *weighted average class
   posterior probability* estimate:
       $$F(\boldsymbol{x}) = \boldsymbol{P}(y=1|\boldsymbol{x}) = \frac{1}{\sum_{m=1}^M c_m} \sum_{m=1}^M c_m \boldsymbol{P}_m(y=1|\boldsymbol{x})$$
5. Predict the class labels $\hat{y}$ according to the MAP rule using a probability
   cutoff $\delta$:
       $$\hat{y} = \begin{cases} 1, & \text{if} \quad F(\boldsymbol{x}) \geq \delta, \\ 0, & \text{if} \quad F(\boldsymbol{x}) < \delta \end{cases}$$
```

algorithm (BREIMAN, 2001), in which the OOB examples are used to get an unbiased estimate of the test prediction error, and also to measure the prediction strength of the attributes. This work proposes an alternative strategy to take advantage of the OOB data. Since the assumption that numeric attributes obey a Gaussian distribution may not hold for some domains, the probability estimates that the RGB classifier produces can occasionally be poorly calibrated. One approach to address this problem is by means of calibration (ZADROZNY; ELKAN, 2001; NICULESCU-MIZIL; CARUANA, 2005), i.e., learning a function that maps the predicted probabilities into more accurate estimates. Thus, the OOB data produced at the m^{th} boosting iteration can be used as a separate set to calibrate the probability estimates produced by the m^{th} base classifier, as suggest in step 3(c) of Algorithm 3.3.

Formally, a classifier is considered well-calibrated (MURPHY; WINKLER, 1977), if the empirical class membership probability $\boldsymbol{P}\left(y|\hat{\boldsymbol{P}}(y|\boldsymbol{x})\right)$ conditioned on the predicted probability $\hat{\boldsymbol{P}}(y|\boldsymbol{x})$ converges to the latter, as the number of training examples goes to infinity. Putting it more simply, consider all examples to which a classifier assigned a particular class probability of say $\hat{\boldsymbol{P}}(y|\boldsymbol{x}) = 0.7$, then approximately 70% of these examples are expected to be members of the class in question. Calibration is important if the class posterior probabilities are required to be interpretable as the chances of membership in the class.

The *Logarithmic Loss* (log-loss), also called logistic regression loss, Kullback-Leibler loss or cross-entropy, is a commonly used measure for assessing the quality of the posterior probability estimates of a given classifier. Let the true labels for a set of N data points be encoded as a 1-of-K binary indicator matrix $\boldsymbol{Y}$, i.e., $y_{n,k} = 1$ if the n^{th} instance has label C_k taken from a set of K labels $\{C_1, \ldots, C_K\}$. Now, let $\boldsymbol{P}$ be a matrix of probability estimates, with $p_{n,k} = \boldsymbol{P}(y_{n,k} = 1|\boldsymbol{x}_n)$. Then the log-loss of the whole set is given by:

$$\boldsymbol{L}_{\log}(\boldsymbol{Y}, \boldsymbol{P}) = -\log \boldsymbol{P}(\boldsymbol{Y}|\boldsymbol{P}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k}\log(p_{n,k}). \tag{3.20}$$

The most widely used calibration methods are *Platt Scaling* (PLATT et al., 1999) and *Isotonic Regression* (ROBERTSON et al., 1988). Let the output of a binary classifier be $f(\boldsymbol{x})$. To get calibrated probabilities, Platt et al. (1999) suggested to pass $f(\boldsymbol{x})$ through a sigmoid:

$$\boldsymbol{P}(y|\boldsymbol{x}) = \frac{1}{1 + \exp(\beta_1 f(\boldsymbol{x}) + \beta_0)} \tag{3.21}$$

where the parameters β_0 and β_1 are fitted using maximum likelihood estimation from a set of data examples $(\boldsymbol{x}_n, y_n)$, with $y_n \in \{0, 1\}$. In order to avoid unwanted bias and get good posterior probabilities, a independent calibration set is needed, different from the data used to train the classifier.

Platt Calibration (or Platt Scaling) essentially involves fitting a logistic regression on the classifier estimates, and was originally proposed to map the SVM (Support Vector

Machines) output scores into probability estimates. However, as observed by Zadrozny and Elkan (2001), the sigmoidal shape does not fit naive Bayes outputs as well as it fits SVM scores. In fact, a sigmoid might not be the correct transformation for all learning algorithms.

Zadrozny and Elkan (2001) successfully used a more general method based on Isotonic Regression to calibrate predictions of many classifiers, including naive Bayes and boosted naive Bayes. Given the predictions $f_n = f(\boldsymbol{x}_n)$ from a classifier and the true targets $y_n \in \{0, 1\}$, the basic assumption in Isotonic Regression is that:

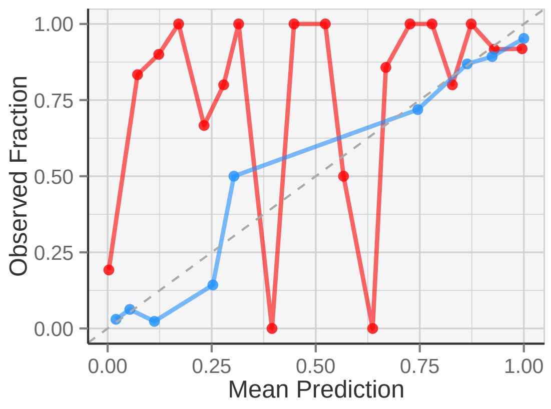
$$y_n = g(f_n) + \epsilon_n \tag{3.22}$$

where g is a isotonic (monotonically increasing) function and ϵ_n is an individual error term. Isotonic Regression is more general in that the only restriction it makes is that the mapping function be isotonic. Assuming that the classifier labels the data points correctly, the mapping from the uncalibrated predictions into posterior probabilities is non-decreasing, and Isotonic Regression can be used to learn this mapping. More precisely, given a training set $\{f_n, y_n\}_{n=1}^N$ with predicted membership values f_n and binary class labels y_n, the Isotonic Regression problem is finding a non-decreasing mapping function $\hat{g}$ so that:

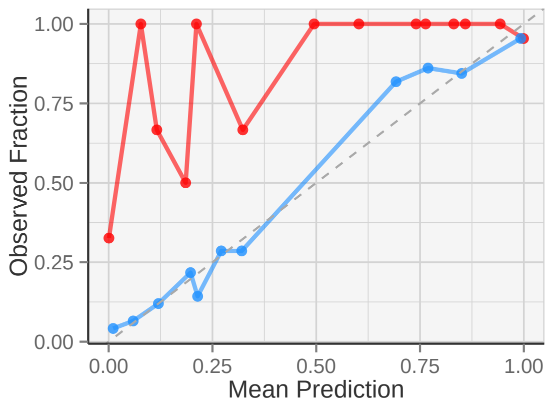
$$\hat{g} = \arg\min_z \sum_{n=1}^{N} (y_n - z(f_n))^2 \tag{3.23}$$

holds. Again, a independent calibration set is needed to train the isotonic function. A commonly used algorithm for computing the isotonic regression is *Pair-Adjacent Violators* (PAV) (AYER et al., 1955). The PAV algorithm first sorts the examples according to their scores f_n and assigns probabilities $z(f_n) = y_n$, $y_n \in \{0, 1\}$. If z is already isotonic, the algorithm then returns $\hat{g} = z$. Otherwise, there must be a subscript n such that $z(f_{n-1}) > z(f_n)$. The examples associated with the scores f_{n-1} and f_n are called pair-adjacent violators, because they violate the isotonic assumption. In this case, the values of $z(f_{n-1})$ and $z(f_n)$ are replaced by their average, so that the $(n-1)^{\text{th}}$ and n^{th} examples now comply with the isotonic assumption. This procedure is repeated using the new values until an isotonic set of values is obtained.

The PAV algorithm finds the stepwise-constant isotonic function that best fits the data according to a square loss criterion. It returns a set of intervals and an estimate $\hat{g}(j)$ for each interval j, such that $\hat{g}(j+1) \geq \hat{g}(j)$. To obtain a calibrated posterior probability for a test example $\boldsymbol{x}_{\text{new}}$, one must find the interval j for which $f(\boldsymbol{x}_{\text{new}})$ is between the lowest and highest scores in the interval and assign $\hat{g}(j)$ as the probability estimate for $\boldsymbol{x}_{\text{new}}$. The `R` package `CORElearn` provides probability calibration with Isotonic Regression for binary



(a) Chess dataset ($2 \times 3196 \times 37$).



(b) Spam dataset ($2 \times 4601 \times 57$).

Figure 3.6 – Reliability diagrams of RGB (uncalibrated in red line, and calibrated in blue line) on two real data sets.

classification problems through the function `calibrate`. Since Isotonic calibration is a more general method, it will be preferred here.

The calibration of a classifier can be visualized through a reliability diagram (DEGROOT; FIENBERG, 1983). Reliability diagrams plot empirical class membership probability $\boldsymbol{P}\left(y|\hat{\boldsymbol{P}}(y|\boldsymbol{x})\right)$ versus predicted probability $\hat{\boldsymbol{P}}(y|\boldsymbol{x})$ at various levels of the latter. If a classifier is well-calibrated, all points will lie on the diagonal indicating that estimates are equal to their empirical probability. However, since the number of different predicted values is large compared to the number of data points, reliable empirical probabilities can not be computed for each possible probability estimate. A typical strategy is to discretize the prediction space and then compute reliable empirical probability estimates for each bin.

Figure 3.6 shows reliability diagrams for two well-known datasets from the UCI repository (LICHMAN, 2013): Chess ($N = 3196$, $p = 36$, $K = 2$) and Spambase ($N = 4601$, $p = 57$, $K = 2$), which have been randomly divided into three parts: a training set (60%), a test set (20%) and a calibration set (20%). The diagrams compare the quality of probability estimates produced by the uncalibrated RGB (in red) against those produced by the RGB calibrated with Isotonic Regression (in blue). The prediction space has been discretized into 20 bins of equal size. For each bin, the mean predicted value is plotted against the observed fraction of positive cases. One can see that the probabilities estimated by the RGB are poorly calibrated. On the other hand, the points for the calibrated RGB (ISO-RGB) are considerably closer to the diagonal line. In addition, calibration has shown to surprisingly improve classification accuracy on both problems. The performance of both RGB and ISO-RGB on the test set over 30 simulation runs is summarized in Table 3.1.

The calibration methods described so far were designed exclusively for two-class

Table 3.1 – Averages and standard errors over 30 replications of performance measure for uncalibrated and calibrated RGB classifiers.

	Dataset	RGB	ISO-RGB
Accuracy			
	Chess	0.784 (0.024)	0.929 (0.014)
	Spambase	0.790 (0.013)	0.900 (0.010)
Log-loss			
	Chess	1.137 (0.234)	0.277 (0.083)
	Spambase	3.043 (0.369)	0.312 (0.056)

problems. This, however, is not a drawback, since AdaBoost is in essence a binary classifier, and most methods of extending AdaBoost for multi-class learning work by reducing the multi-class problem to multiple two-class problems.

One popular strategy, called "AdaBoost.MH" (SCHAPIRE; SINGER, 1999), is based on the *one-against-all* technique (RIFKIN; KLAUTAU, 2004), where each individual class (typically coded as 1) is modeled against all the remaining classes (each coded as zero), and K different ensembles are constructed. The values $F_k(\boldsymbol{x})$ returned by each ensemble are compared and the value corresponding to the maximum F_k is given as the class label (FRIEDMAN et al., 2000). In contrast, Zhu et al. (2009) proposed to directly extend the AdaBoost algorithm to the multi-class case, without reducing it to multiple two-class problems. However, for the BRGB algorithm, this method, referred to as SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function), performed worse than the one-vs-all approach. Thus, the later will be adopted for the remainder of this work in the multi-class classification setting.

4 Experimental Results

In this chapter, a large, carefully controlled and carefully measured set of experiments, as well as analysis of these experiments are presented. Firstly, the improved covariance estimators discussed in Chapter 2 are compared in terms of their prediction performance and computational complexity. Then, the Regularized Gaussian Bayes (RGB) classifiers derived from these estimators are evaluated on artificially generated data, in terms of their prediction accuracy and quality of their probability estimates. The classifier with the best performance on test data, as well as its boosted version (BRGB), are then compared against state-of-the-art classification algorithms on a large set of popular real world problems. Finally, the performance of RGB and BRGB is evaluated on hard classification problems taken from two famous Kaggle[1] machine learning competitions.

All experiments were carried out on a Linux machine with Intel i7-2600S 2.80GHz **CPU** and 12GB **RAM**.

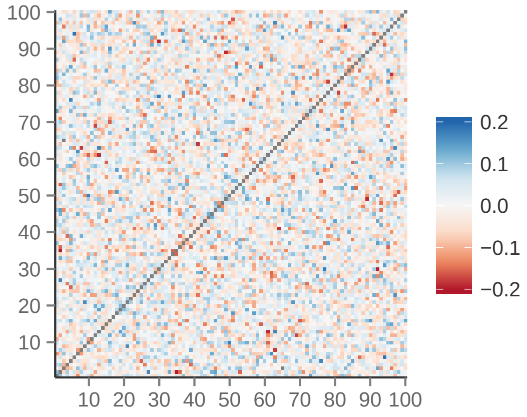
4.1 Experiment with Synthetic Covariance Matrices

The goal of this first experiment is to compare the expected loss (or risk) of the improved covariance estimators discussed in Chapter 2 (see Table 2.3) across a wide range of situations, using artificially generated Gaussian data. The number of variables is taken as $p = 100$. Two different structures are considered for the true covariance matrix $\mathbf{\Sigma}$: a dense structure (only non-zero valued entries) and a sparse structure (many zero valued entries). The first is obtained by combining a randomly generated correlation matrix (JOE, 2006) with randomly assigned attribute variances in $[1, 10]$, using the `R` package `clusterGeneration` (QIU; JOE., 2015). The second is generated using the `R` package `spcov` (BIEN; TIBSHIRANI, 2012), considering a block diagonal structure with 4 blocks. The corresponding population correlation matrices are illustrated in Figure 4.1.

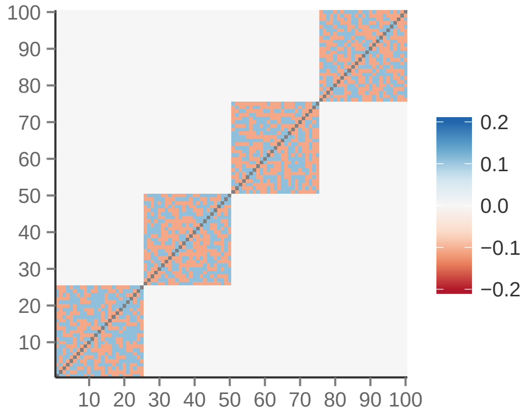
For each covariance matrix structure, the covariance estimators are fitted over multivariate Gaussian observations drawn randomly and independently from $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$. The `R` package `mvnfast` (FASIOLO, 2014) provides fast routines for simulation of multivariate normal random variables. In order to evaluate predictions at different levels of the p/N ratio, five sample sizes are considered: $N = \{1000, 200, 100, 50, 10\}$. The average losses and standard deviations over 50 simulations of performance measures are detailed in Annex A.

The estimators are evaluated in terms of the three well known loss functions $\boldsymbol{L}_1$ (2.4), $\boldsymbol{L}_2$ (2.5) and $\boldsymbol{L}_3$ (2.8). Additionally, the sparsity level of the estimates is measured

[1] https://www.kaggle.com



(a) Dense (non-sparse) 100 × 100 correlation matrix target.



(b) Sparse 100 × 100 correlation matrix target.

Figure 4.1 – Synthetic correlation matrices used to form the covariance matrix targets to compare the prediction performance of the improved covariance estimators.

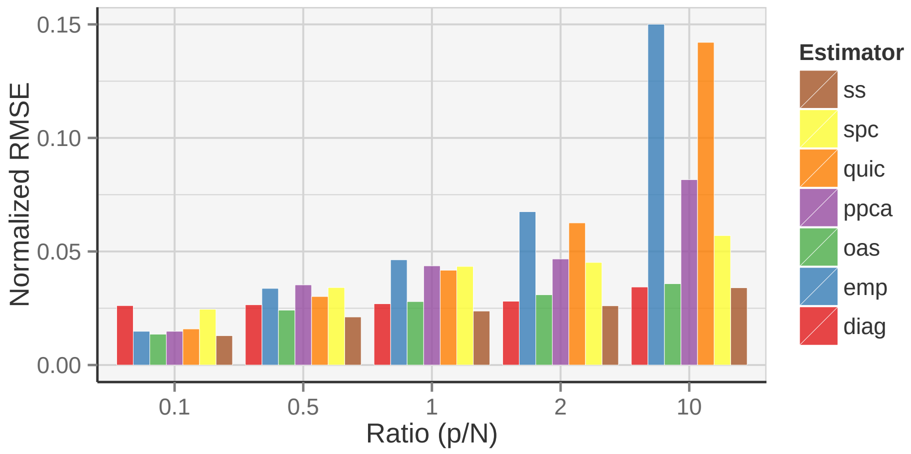
by means of $\boldsymbol{L}_0$, which is simply the percentage of zero valued entries of $\hat{\boldsymbol{\Sigma}}$. Two other performance measures are reported, the absolute difference in the largest eigenvalue, $|\lambda_{\max}(\hat{\boldsymbol{\Sigma}}) - \lambda_{\max}(\boldsymbol{\Sigma})|$, and the absolute value of the cosine of the angle between the estimated and true eigenvectors corresponding to the first eigenvalue. This assesses how accurate each of the estimators would be in estimating the first principal component. For the EMP and DIAG estimators, that is, the empirical estimator and its diagonal, the nearest positive-definite matrix approximation method of Higham (1988) has been adopted to ensure non-singularity of the covariance estimate when $p > N$.

Figure 4.2 compares the average performances over the 50 simulations, as measured by the Normalized Root MSE (or the expected $\boldsymbol{L}_3$ loss), as the p/N ratio increases. The $\boldsymbol{L}_3$ loss is preferred here since it has been empirically observed that covariance estimates with minimal $\boldsymbol{L}_3$ loss often leads to more accurate classifiers. It can be noted that, as the sample size N becomes small relative to the number of variables p, the OAS and SS estimators dominate the others on both dense and sparse targets, except the DIAG estimator, which demonstrated to perform well on large p/N ratios. However, this estimator is dominated by OAS and SS in small p/N settings.

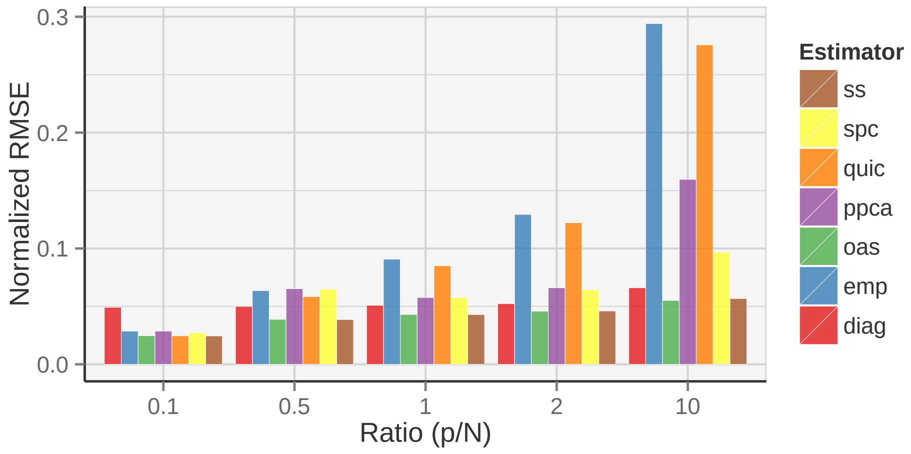
4.2 Experiment with Artificial Classification Datasets

The goal is to compare the prediction performance of the RGB classifiers formed with different covariance estimators, using simulated data. A brief description of the artificially generated datasets follows.

- *twonorm*: This is a simulated 20 variable data with 7400 cases and 2 classes each



(a) Prediction performance in the dense setting.



(b) Prediction performance in the sparse setting.

Figure 4.2 – Average performance of the covariance estimators as the p/N ratio increases.

having probability 1/2. It is described in (BREIMAN, 1996). Each class is drawn from a multivariate normal distribution with unit variance. Class 1 has mean $(a, a, \ldots, a)$ while Class 2 has mean $(-a, -a, \ldots, -a)$, where $a = 2/\sqrt{20}$. Breiman (1996) reports the Optimal Bayes classification rate as 97.7%. The function `mlbench.twonorm` implemented in the `R` package `mlbench` allows simulating this dataset;

- *threenorm*: This is a simulated 20 variable data with 7400 cases and 2 classes each having probability 1/2. It is described in (BREIMAN, 1996). Class 1 is drawn with equal probability from an unit (identity covariance matrix) multivariate normal with mean $(a, a, \ldots, a)$ and from an unit multivariate normal with mean $(-a, -a, \ldots, -a)$. Class 2 is drawn from an unit multivariate normal with mean at $(a, -a, a, -a, \ldots, a, -a)$, where $a = 2/\sqrt{20}$. Breiman (1996) reports the Optimal Bayes classification rate as 89.5%. The function `mlbench.threenorm` implemented in the `R` package `mlbench` allows simulating this dataset;

- *ringnorm*: This is a simulated 20 variable data with 7400 cases and 2 classes each having probability 1/2. It is described in (BREIMAN, 1996). Each class is drawn from a multivariate normal distribution. Class 1 has mean zero and covariance 4 times the identity. Class 2 has mean $(a, a, \ldots, a)$ and unit covariance, where $a = 2/\sqrt{20}$. Breiman (1996) reports the Optimal Bayes classification rate as 98.7%. The function `mlbench.ringnorm` implemented in the `R` package `mlbench` allows simulating this dataset;

- *waveform*: This is a simulated 21 variable data with 5000 cases and 3 classes each having probability 1/3. It is described in (BREIMAN et al., 1984). The Optimal Bayes classification rate is reported as 86%. The function `mlbench.waveform` implemented in the `R` package `mlbench` allows simulating this dataset;

- *waveform-noise*: This is a simulated 40 variable data with 5000 cases and 3 classes each having probability 1/3. It is described in (BREIMAN et al., 1984). The latter 19 attributes are all noise attributes with zero mean and unit variance. The Optimal Bayes classification rate is reported as 86%. Both *waveform* and *waveform-noise* datasets are available in the UCI repository (LICHMAN, 2013);

- *corelearn*: This is a simulated 10 variable data with 5000 cases and 2 classes each having probability 1/2. It was generated through the function `classDataGen` available in the `R` package `CORElearn`. The generator produces 7 categorical and 3 numeric attributes. A final dataset with 16 numerical-valued attributes has been obtained by expanding the categorical variables into individual dummy variables;

- *xor*: This is a simulated 4 variable data with 5000 cases and 8 classes each having probability 1/8. It was generated through the function `mlbench.xor` available in the `R` package `mlbench`.

The experimental procedure to be used on these datasets consists of 30 iterations of the following steps:

1. Select at random 300 data points for training, as suggested in (BREIMAN, 1996), and set the left out examples as a test set;

2. Run RGB (Algorithm 3.1) on the training data considering the seven covariance estimators studied in the previous section;

3. Get accuracy rates and log-loss on test data.

The performance measures from step 3 are averaged over the 30 iterations to get the final results shown in Annex B. In order to conduct a pairwise statistical comparison between the classifiers, the *bootstrap* technique (EFRON; TIBSHIRANI, 1994; EFRON, 1987) was applied to estimate the 95% confidence intervals (CIs) for the differences in average performance between pairs of classifiers, on the seven datasets. The algorithm with the lowest computational cost, the OAS-RGB, has been taken as the control level. More precisely, the technique carries out the following steps:

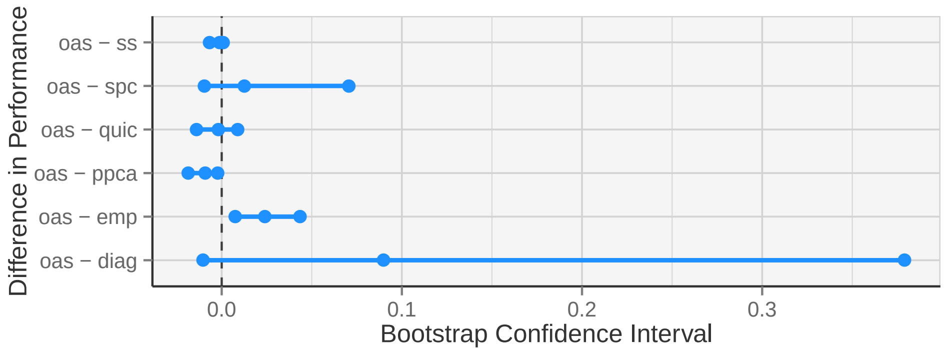
1. Choose a classification performance measure: accuracy or log-loss;
2. For each classifier excluding the control algorithm:
    a) Take the average results computed on the seven datasets and subtract them from the results obtained with the OAS-RGB algorithm;
    b) Draw, with replacement, $10,000$ samples of size 7 from the differences obtained in step 2(a);
    c) Take the mean of each bootstrap sample;
    d) The limits of the 95% CI are the quantiles of 2.5% and 97.5% of the estimated means.

The `R` package `boot` (CANTY; RIPLEY, 2012) provides a variety of methods to compute bootstrapped confidence intervals.

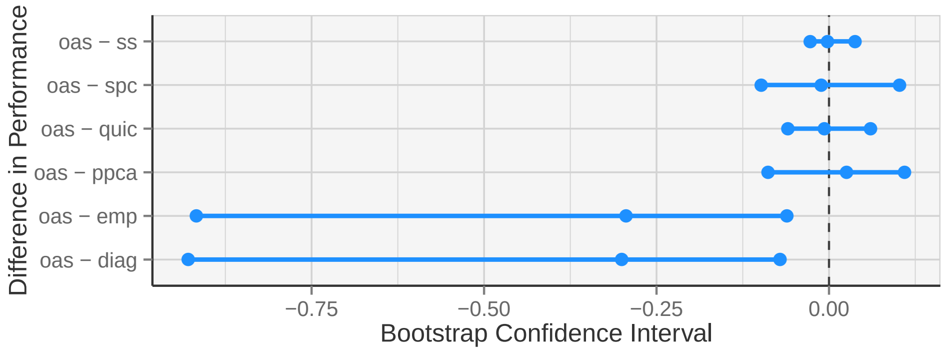
Basically, considering classification accuracy as the performance measure, 95% confidence intervals containing the zero suggest small or no difference between algorithms, while strictly negative or strictly positive confidence intervals suggest, respectively, average performance greater or smaller than the control algorithm, at a significance level of 0.05.

Figure 4.3 shows, respectively, the estimated CIs for the mean differences in average accuracy and average log-loss. It can be note that the OAS-RGB significantly outperforms the EMP-RGB in terms of classification accuracy and log-loss. The DIAG-RGB algorithm is outperformed only in terms of log-loss. For the remaining algorithms, there is not enough evidence to reject the null hypothesis of similarity with the OAS-RGB in terms of both accuracy and log-loss. Thus, the latter will be preferred in the real world data experiments because of its low computational cost.

Since bootstrap is a non-parametric method, it is not subjected to the assumptions which frequently limit parametric methods, like normality and homoscedasticity. A parametric method assumes that data comes from a type of probability distribution and makes inferences about the parameters of the distribution. In fact, if the assumptions required for a parametric method hold, the later should always be preferred over a non-parametric one, in that it will have a lower Type I error and higher power (GARCÍA et al., 2010). However, in most cases these conditions are not easy to meet.



(a) Comparison of **accuracy** between the control algorithm OAS-RGB and the rest of algorithms.



(b) Comparison of **log-loss** between the control algorithm OAS-RGB and the rest of algorithms.

Figure 4.3 – Bootstrapped 95% confidence intervals of the differences in average performance between the control algorithm OAS-RGB and the other RGB classifiers.

4.3 Experiment with Real World Classification Datasets

The goal is to compare the prediction performance of the RGB and BRGB against state-of-the-art classification algorithms over a wide range of data settings. For this purpose, 33 classification problems are considered, whose within-class size ranges from 10 to 45586 and dimensionality ranges from 4 to 180. Datasets are from the UCI machine learning repository (LICHMAN, 2013), all of them valid for classification tasks. They are detailed in Annex C. The first 25 problems are moderate sized datasets and the last 8 are larger ones. The computational burden for the latter make them intractable for some learning algorithms, as it will be seen.

In order to establish how well RGB (RegGaussBayes) and BRGB (BoostedRgb) perform on each of these 33 datasets, they are compared with 12 well-established learning algorithms implemented in `R`: *Gaussian Naive Bayes* (GaussNaiveBayes or GNB), *Vanilla* SVM (LinearSvm or SVM.VAN) and *Radial Basis Function* SVM (NonlinearRbfSvm or SVM.RBF) from the package `e1071` (MEYER et al., 2015), *Kernel Naive Bayes*[2] (KernelNaiveBayes or KNB) from the package `klaR` (WEIHS et al., 2005), Shrinkage *Linear Discriminant Analysis* (LinearDiscriminant or LDA) from the package `sda` (AHDESMAKI et al., 2015), Regularized *Generalized Linear Model* (GlmNet or GLMNET) from the package `glmnet` (FRIEDMAN; HASTIE; TIBSHIRANI, 2010), Pruned *Cart* Decision Tree (Cart or CART) from the package `rpart` (THERNEAU; ATKINSON; RIPLEY, 2015), Pruned *C4.5* Decision Tree (C4.5) and *AdaBoost* (Adaboost or ADA) with C4.5 as base learner from the package `RWeka` (HORNIK; BUCHTA; ZEILEIS, 2009), *Extremely Randomized Trees* (ExtraTrees or EXT) from the package `extraTrees` (SIMM; de Abril; SUGIYAMA, 2014), *Gradient Boosting Machine* (GradientBoosting or GBM) from the package `gbm` (OTHERS, 2015), and *Random Forest* (RandomForest or RF) from the package `randomForest` (LIAW; WIENER, 2002). The parameters used for each learning algorithm are summarized in Annex D. `R` provides a variety of packages containing alternative implementations for these algorithms, but the previously listed packages are strongly recommended, since other packages tested in this study gave errors for some datasets.

The RGB and BRGB algorithms are modeled using the OAS covariance estimator. Simplified structures are also considered. The *RegGaussBayes-diag* (RGB.D) and *BoostedRgb-diag* (BRGB.D) use a diagonal estimator for the covariance matrix. However, it has been empirically observed in some datasets that the diagonal of the sample covariance matrix is a numerically unstable estimate, then it was replaced by the diagonal of the OAS estimator. Finally, the *RegGaussBayes-lowrank* (RGB.LR) and *BoostedRgb-lowrank* (BRGB.LR) use a rank-2 PPCA estimator.

For the learning procedure of the BRGB classifiers, 20 boosting iterations are considered with a learning rate step $\gamma = 0.5$, except on *Soybean*, *Satellite* and *Texture*. In particular, shrinkage is necessary to make BRGB work properly on these datasets. This is achieved by simply reducing the step-size to $\gamma = 0.1$. The probability cutoff of the committee decision rule is set as $\delta = 0.5$ for all problems.

4.3.1 Experimental Protocol

All datasets were pre-processed in the same way. Firstly, for datasets with missing values, all observations containing missing entries were discarded. Then, all nominal/categorical attributes taking on k different values were expanded into $k-1$ individual dummy (0-1 coded) variables, with arbitrary selection of the control category. Finally, constant valued attributes were discarded.

[2] John and Langley (1995) showed that the Bayesian classifier's performance can be much improved if the traditional treatment of numeric attributes, which assumes Gaussian distributions, is replaced by kernel density estimation.

It has been observed that, after converting the categorical variables to binary ones, it is common to obtain attributes with constant within-class entries (all 0 or all 1), that is, attributes with zero within-class sample variance. This introduces a zero on the main-diagonal of the sample covariance estimate. In this case, the empirical within-class covariance matrix is always singular, even when $N \gg p$. The same is not true for the improved covariance estimators, since they add a constraint on the variance estimates.

After preprocessing the data, the experimental procedure to be used on the 33 datasets consists of 6 runs of 5-fold cross-validation, in order to obtain a sample of 30 results. The detailed description of the experiment follows.

1. Randomly break the dataset into five roughly equal-sized subsets, respecting as closely as possible the class frequencies in the original data;

2. Get five train/test splits by taking four of the subsets as training data (roughly 80% of the cases) and the fifth as test data (roughly 20% of the cases);

3. For each train/test split:

    a) Normalize the continuous real valued attributes to have mean 0 and variance 1. Apply the same scaling to the training and test sets, but determine the scaling using only the training set;

    b) Run the learning algorithms on the training data;

    c) Get accuracy rates and log-loss on test data.

The algorithms GLMNET, SVM.VAN, SVM.RBF and ADA are left out from the analysis of the large-sized datasets: *Adult*, *Dna*, *Letter*, *Optdigits*, *Penbased*, *Satellite*, *Shuttle*, *Texture*. Such methods are implemented in R for moderate-sized data. They are memory inefficient and have slow training speed for large datasets.

The BRGB with Isotonic Regression is evaluated only on the large datasets, since small samples of OOB data do not provide enough information to obtain well-calibrated base learners. In (ZADROZNY; ELKAN, 2001), for example, the calibration sets considered in the classification experiments range from 3498 to 96367 data points. Caruana and Niculescu-Mizil (2006) successfully calibrated popular classifiers, like SVM and Naive Bayes, on 11 binary classification problems, using 1000 separated observations. Suppose for simplicity that in each boosting iteration the examples are sampled uniformly and roughly 37% of the observations are left out from the training set. Thus, about 2700 training observations are necessary to get 1000 OOB examples.

4.3.2 Results

The performance measures computed in step 3(c) for each learning algorithm on each of the 33 problems are averaged over the 30 replicates to get the final results shown in Annex E. Excluding the trials developed for parameter tuning in some classifiers, the total number of experimental runs is $18,810$.

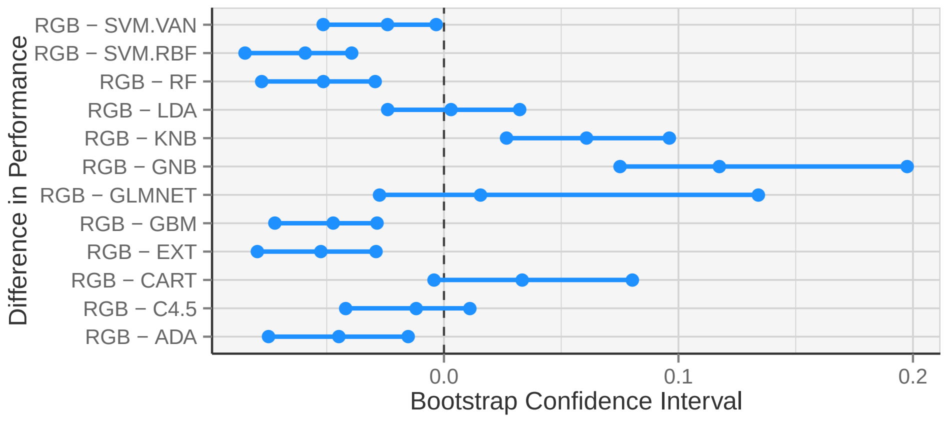
It can be observed that, as the No-Free-Lunch Theorem (WOLPERT; MACREADY, 1997) suggests, there is no universally best learning algorithm. Formally, this theorem states that for every learner, there exists a task on which it fails, even though that task can be successfully learned by another learner. In fact, even the best models (*ExtraTrees*, *GradientBoosting*, and *RandomForest*) perform poorly on some problems, and models that have poor average performance perform well on a few problems. For example, the best models on *Vehicle* are *RegGaussBayes*, *BoostedRgb* and *NonlinearRbfSvm*. Tree based learning algorithms perform much worse. On the other hand, these classifiers perform very well on *Chess* and *Phoneme*. On *Balance*, *GaussNaiveBayes* performs better than *ExtraTress*, *Adaboost* and *RandomForest*, even though it has poor average performance in general. *BoostedRgb-lowrank* is outperformed by *BoostedRgb* in most problems, but on *Spam* and *Dna* it performs much better.

However, in practise, following the conclusions drawn from the huge classification experiment conducted in (FERNÁNDEZ-DELGADO et al., 2014), the Random Forest algorithm may be considered as a reference ("gold-standard") to compare with new classier proposals in order to assess their performance in general classification tasks.

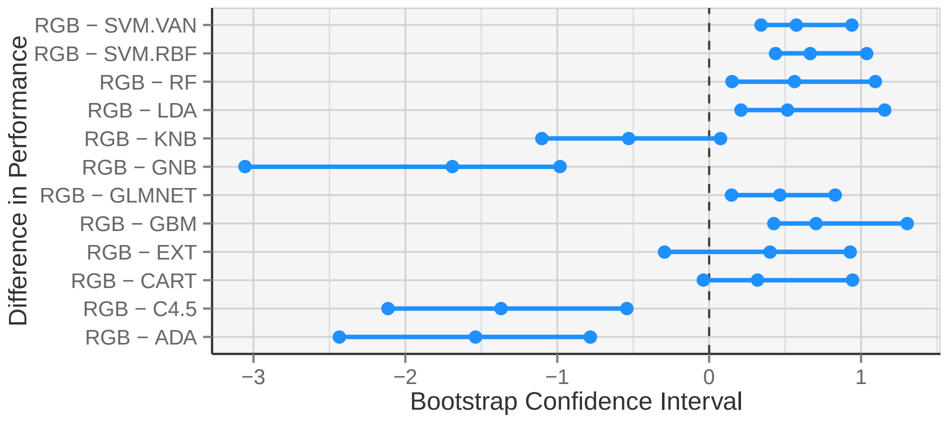
4.3.3 Statistical Analysis

In order to establish what classifiers perform worse/better than RGB, the later has been set as the control algorithm, and a bootstrap analysis similar to the one in the previous section was conducted. The 95% CIs for the differences in average performance between RGB and the other classifiers are shown in Figure 4.4. For each classifier, the CI was calculated excluding the datasets in which it was not executed (denoted as - in the results in Annex E). One can note that RGB is significantly better than GNB and KNB in terms of accuracy, but is clearly worse than ADA, EXT, GBM, RF and SVM.RBF. Also, the SVM.VAN proved to be slightly more accurate than RGB. In contrast, there is not enough evidence to conclude difference between RGB and the tree based learners CART and C4.5, as well as the regularized linear learners LDA and GLMNET. The CIs for the differences in log-loss suggest that in general the RGB classifier gives poorly calibrated posterior probability estimates, although significantly improves the quality of the probabilities predicted by the GNB.

The same analysis was conducted for the BRGB, as shown in Figure 4.5. It is



(a) Comparison of **accuracy** between the control algorithm RGB and the rest of algorithms.

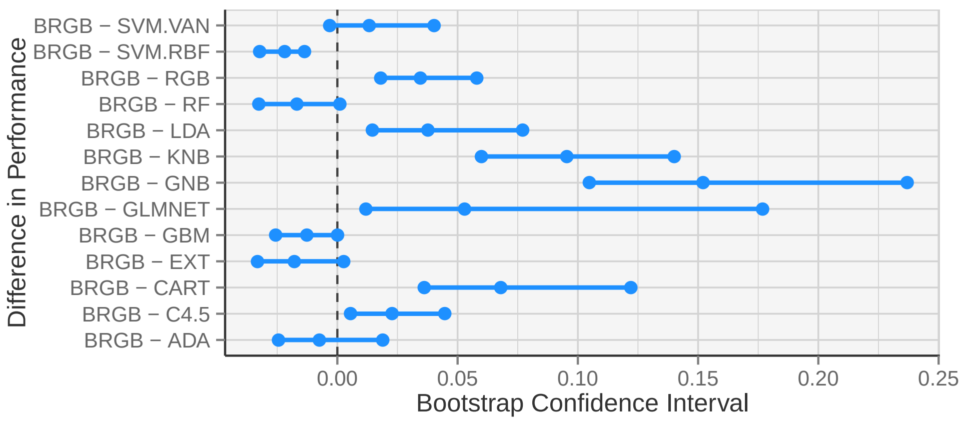


(b) Comparison of **log-loss** between the control algorithm RGB and the rest of algorithms.

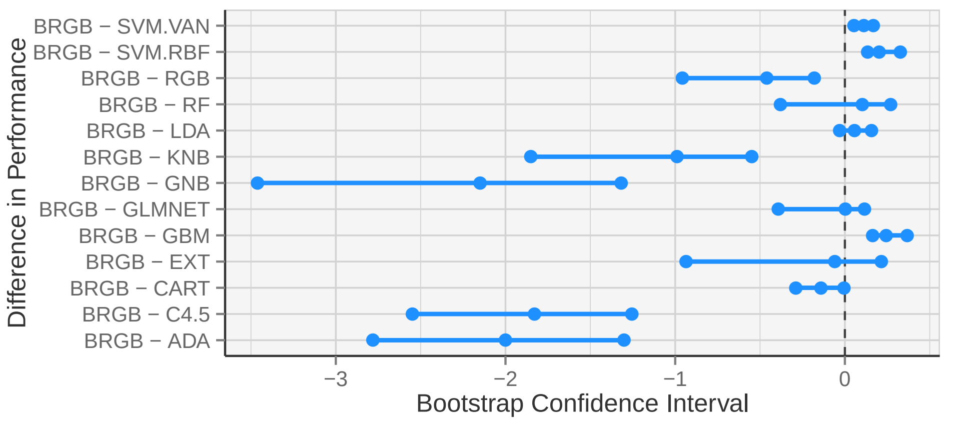
Figure 4.4 – Bootstrapped 95% confidence intervals of the differences in average performance between the control algorithm RGB and the other classifiers.

clear that the BRGB significantly outperforms the RGB on both classification accuracy and probability estimation. Only SVM.RBF proved to be better than BRGB in terms of accuracy. Moreover, there is no significantly difference between BRGB and the best "of-the-shelf" ensemble based architectures (ADA, EXT, GBM and RF) in the average accuracy. GBM in turn gives better probability estimates than BRGB. The SVM models proved to be optimal probability predictors, and significantly outperform the BRGB in terms of average log-loss. In fact, the package `e1071` provides a interface to the **LIBSVM** library (CHANG; LIN, 2011), which uses Platt scaling to convert the SVM output scores into posterior class probabilities.

The BRGB calibrated with Isotonic Regression (BRGB.ISO) was analysed separately, since it was evaluated on a small set of the problems. Assuming the BRGB as the control method, the BRGB.ISO proved to be a better probability predictor on the



(a) Comparison of **accuracy** between the control algorithm BRGB and the rest of algorithms.



(b) Comparison of **log-loss** between the control algorithm BRGB and the rest of algorithms.

Figure 4.5 – Bootstrapped 95% confidence intervals of the differences in average performance between the control algorithm BRGB and the other classifiers.

large-sized datasets (95% CI for the differences in average log-loss: $(-0.578, -0.104)$). On the other hand, it does not improve accuracy significantly (95% CI for the differences in average accuracy: $(-0.009, 0.016)$).

As a final analysis, the following bootstrap sampling was repeated 1000 times, yielding 1000 potentially different rankings of the learning algorithms:

1. Randomly select a bootstrap sample (sampling with replacement) of size 33 from the original 33 problems;

2. Compute the frequency that each method ranks 1st, 2nd, 3rd, etc. on the bootstrap sample, according to the chosen performance measure (accuracy or log-loss).

The algorithms not tested on the large-sized datasets were kept out from this analysis. The average frequencies over the 1000 replicates considering classification accuracy as performance measure are shown in Annex F. They suggest that Extra Trees probably are the top performing method overall, with a 33% chance of ranking first, followed by Random Forests, which would come in 2nd place 36% of the time and 3rd place 27% of the time, with little chance of ranking below third place. If a new problem is selected, there is a 22% chance that BRGB would come in 1st place, but with high chance (57%) for ranking below third place. The bootstrap analysis clearly shows that GNB, KNB, CART, C4.5, LDA and RGB are not competitive on average with the top four models on these problems.

4.4 Experiment with Challenging Classification Datasets

The RGB and BRGB are also evaluated on two hard classification datasets from two famous *Kaggle*'s competitions:

- *Higgs Boson Machine Learning Challenge*: This is a two-class problem. The labeled data contains 250,000 examples on 30 continuous real valued features. It contains missing values which were filled using a ridge regression based imputation method from the `R` package `imputeR` (FENG et al., 2014). Alternatively, the package `mice` (van Buuren; GROOTHUIS-OUDSHOORN, 2011) can be used in the missing value imputation task.

- *Otto Group Product Classification Challenge*: This is a nine-class problem. The labeled data contains 61,878 sparse examples on 93 count features. There are no missing values.

The proposed algorithms are compared with state-of-the-art methods provided by two famous large-scale and distributed machine learning libraries: the *Gradient Boosted Decision Tree* (XGBOOST) from XGBoost[3] (eXtreme Gradient Boosting), and the traditional Naive Bayes (H2O.GNB), Gradient Boosting Machine (H2O.GBM) and Random Forest (H2O.RF) from H2O[4]. In the `R` environment these libraries can be used through the packages `xgboost` (CHEN; HE; BENESTY, 2015) and `h2o` (AIELLO et al., 2015), respectively.

The performance results on test data, averaged over 5 runs of 3-fold CV on each dataset, are shown in Table 4.1. As can be seen, RGB outperforms H2O.GNB on both problems, giving better accuracy rates and better probability estimates. The boosting procedure significantly improves RGB on *Higgs Boson*, but it surprisingly degrades the accuracy rate on *Otto Group*. The probability calibration strategy leads to a better prediction

[3] https://github.com/dmlc/xgboost.
[4] https://github.com/h2oai/h2o-3.

Table 4.1 – Averages and standard errors (in brackets) of performance measures over 5 runs of 3-fold cross-validation.

	Algorithm	**Accuracy**	**Log-Loss**	**Training Time**
Higgs Boson				
	RGB	0.7035 (0.0017)	1.4394 (0.0314)	**0.3432 (0.0321)**
	BRGB	0.7807 (0.0023)	0.4713 (0.0016)	28.2878 (0.9653)
	ISO.BRGB	0.7959 (0.0019)	0.4386 (0.0023)	38.4375 (0.6400)
	XGBOOST	**0.8389 (0.0009)**	**0.3600 (0.0015)**	19.8468 (0.5052)
	H2O.GNB	0.6913 (0.0019)	1.5841 (0.0348)	1.1530 (0.0478)
	H2O.GBM	0.8327 (0.0022)	0.3601 (0.0016)	24.4458 (0.7655)
	H2O.RF	0.8306 (0.0020)	0.3693 (0.0014)	70.6436 (0.7695)
Otto Group				
	RGB	0.6709 (0.0030)	5.9372 (0.0871)	**0.6474 (0.0107)**
	BRGB	0.6445 (0.0505)	1.3189 (0.0529)	74.3039 (0.8991)
	ISO.BRGB	0.7008 (0.0017)	0.7960 (0.0060)	104.1855 (0.5162)
	XGBOOST	**0.8097 (0.0028)**	**0.5003 (0.0056)**	18.1402 (0.4604)
	H2O.GNB	0.5818 (0.0055)	8.3664 (0.1134)	1.2176 (0.1350)
	H2O.GBM	0.8043 (0.0024)	0.5276 (0.0044)	145.4710 (3.5374)
	H2O.RF	0.7839 (0.0020)	0.6578 (0.0020)	86.9567 (1.1190)

performance in terms of both classification accuracy and probability estimation. However, the BRGB.ISO can not perform as well as the H2O.RF, H2O.GBM and XGBOOST algorithms.

4.5 Source Code of the Procedures

The source code written in `R`, that implements all procedures described in this chapter, as well as all datasets used, are available at GitHub[5].

[5] https://github.com/davidnexer/master-thesis.

5 Conclusions and Future Work

This chapter recalls the conclusions and contributions of this dissertation, and finalizes with directions for future work.

5.1 Summary and Implications

The study set out in this dissertation have explored regularization strategies that guarantees positive-definiteness and well-condition of the covariance matrix estimates in high-dimensional data settings. A variety of regularized estimators were introduced as an extension of the classical sample estimator, which proved to be impractical in "small N, large p" problems. These improved covariance estimators have been used to model the correlation structure among attributes in the Gaussian Bayes classifier, yielding non-naive parsimonious class models that proved to perform surprisingly well in high dimensions. Moreover, this study compensated the lack of literature on improving the naive Bayes classifier in problems with continuous-valued attributes.

In general, as demonstrated in Chapter 2, it is not trivial to determine the regularization tuning parameters of the covariance matrix estimators. In many cases, cross-validation procedures seemed to overfit, since they were based on unsupervised learning (unknown true covariance matrix). The analytically-defined estimators in turn, demonstrated to be computationally faster and more accurate alternatives.

The proposed Regularized Gaussian Bayes (RGB) classifier, built upon improved estimates of the class priors, vector of means and covariance matrix, dramatically outperformed the Gaussian Naive Bayes classifier (GNB) in low, medium and high dimensional data settings. In addition, no significantly difference has been found between RGB and the popular learners CART, C4.5, LDA and Regularized GLM. But the former has the advantage of being parameter-free if a covariance estimator with analytically defined shrinkage intensity (like the OAS, SS and PPCA) is selected. This explains the extremely low training time taken by the algorithm in the experiments. However, in general, the RGB shown to predict poorly calibrated posterior probabilities.

Adaptive Boosting led to impressive improvements on classification performance. It significantly improved classification accuracy as well as probability estimation by combining few RGB classifiers and predicting weighted averaged posterior probabilities. In summary, the proposed Boosted Regularized Gaussian Bayes (BRGB) classifier proved to be as accurate as the best of-the-shelf ensemble learners, Extra Trees, Gradient Boosting Machines and Random Forests. Moreover, given enough within-class data examples, calibration of the base learners using the OOB data demonstrated to significantly improve probability estimation of the BRGB. Finally, the learning rate step-size γ shown to be important on avoiding overfitting in some problems.

5.2 Future Work

The work herein can be expanded in a number of directions. Firstly, the use of classification loss functions (classification error rate or log-loss, for example) should be explored to select the regularization tuning parameters of the within-class covariance matrices, using the K-fold CV approach. In this case, the best regularization parameters would be those that lead to the best classification performance on validation data.

The experimental results suggest that, given the p/N ratio, some covariance estimators are more appropriated than others. Thus, the RGB classifier should use a mixture of regularized covariance estimators instead of a single one. More specifically, different estimators should be applied in the same classification problem, selecting the best one to each class according to the within-class p/N ratio. Moreover, Gaussian Mixture Models (GMMs) (BISHOP, 2006) could be used to build the within-class models, allowing represent different subclasses inside one class.

Finally, other boosting approaches should be explored, like *Real AdaBoost* (SCHAPIRE; SINGER, 1999), *Gentle Adaboost* (FRIEDMAN et al., 2000) and *Modest AdaBoost* (VEZHNEVETS; VEZHNEVETS, 2005), which are more robust and stable than the *Discrete Adaboost* and lead to better generalization performance.

Bibliography

ABABOU, R.; BAGTZOGLOU, A.; WOOD, E. On the condition number of covariance matrices in kriging, estimation, and simulation of random fields. *Mathematical Geology*, Kluwer Academic Publishers-Plenum Publishers, v. 26, n. 1, p. 99–133, 1994. Cited on page 32.

AHDESMÄKI, M.; STRIMMER, K. et al. Feature selection in omics prediction problems using cat scores and false nondiscovery rate control. *The Annals of Applied Statistics*, Institute of Mathematical Statistics, v. 4, n. 1, p. 503–519, 2010. Cited on page 66.

AHDESMAKI, M. et al. *sda: Shrinkage Discriminant Analysis and CAT Score Variable Selection*. [S.l.], 2015. R package version 1.3.7. Disponível em: <http://CRAN.R-project.org/package=sda>. Cited 2 times on pages 66 and 85.

AIELLO, S. et al. *h2o: R Interface for H2O*. [S.l.], 2015. R package version 3.0.0.30. Disponível em: <http://CRAN.R-project.org/package=h2o>. Cited on page 90.

ANDERSON, T. *An Introduction to Multivariate Statistical Analysis*. [S.l.]: Wiley, 2003. (Wiley Series in Probability and Statistics). Cited 3 times on pages 25, 26, and 31.

Anestis Touloumis. Nonparametric stein-type shrinkage covariance matrix estimators in high-dimensional settings. *Computational Statistics and Data Analysis*, v. 83, p. 251–261, 2015. Cited on page 39.

AYER, M. et al. An empirical distribution function for sampling with incomplete information. *The annals of mathematical statistics*, Institute of Mathematical Statistics, v. 26, n. 4, p. 641–647, 1955. Cited on page 75.

BANERJEE, O.; GHAOUI, L. E.; D'ASPREMONT, A. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *J. Mach. Learn. Res.*, JMLR.org, v. 9, p. 485–516, jun 2008. Cited on page 28.

BELLMAN, R.; BELLMAN, R. *Adaptive Control Processes: A Guided Tour*. [S.l.]: Princeton University Press, 1961. ('Rand Corporation. Research studies). Cited on page 26.

BENNETT, P. N. *Assessing the calibration of Naive Bayes posterior estimates*. [S.l.], 2000. Cited 2 times on pages 29 and 62.

BICKEL, P. J.; LEVINA, E. Some theory for fisher's linear discriminant function,'naive bayes', and some alternatives when there are many more variables than observations. *Bernoulli*, JSTOR, p. 989–1010, 2004. Cited 2 times on pages 28 and 29.

BICKEL, P. J.; LEVINA, E. Covariance regularization by thresholding. *The Annals of Statistics*, JSTOR, p. 2577–2604, 2008. Cited 4 times on pages 28, 53, 55, and 56.

BICKEL, P. J.; LEVINA, E. Regularized estimation of large covariance matrices. *Ann. Statist.*, The Institute of Mathematical Statistics, v. 36, n. 1, p. 199–227, 02 2008. Cited 8 times on pages 26, 28, 36, 37, 50, 53, 54, and 55.

BIEN, J.; TIBSHIRANI, R. *spcov: Sparse Estimation of a Covariance Matrix.* [S.l.], 2012. R package version 1.01. Disponível em: <http://CRAN.R-project.org/package=spcov>. Cited on page 79.

BISHOP, C. *Pattern Recognition and Machine Learning.* [S.l.]: Springer, 2006. (Information Science and Statistics). Cited 6 times on pages 25, 42, 43, 60, 70, and 94.

BISHOP, C. M. Variational principal components. IET, 1999. Cited on page 47.

BOULLÉ, M. Compression-based averaging of selective naive bayes classifiers. *J. Mach. Learn. Res.*, JMLR.org, v. 8, p. 1659–1685, dec 2007. Cited 2 times on pages 29 and 62.

BREIMAN, L. Bias, variance, and arcing classifiers. Tech. Rep. 460, Statistics Department, University of California, Berkeley, CA, USA, 1996. Cited 2 times on pages 81 and 82.

BREIMAN, L. *Out-of-bag estimation.* [S.l.], 1997. Cited on page 72.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Cited 2 times on pages 29 and 73.

BREIMAN, L. Population theory for boosting ensembles. *Annals of Statistics*, JSTOR, p. 1–11, 2004. Cited on page 68.

BREIMAN, L. et al. *Classification and regression trees.* [S.l.]: CRC press, 1984. Cited on page 82.

BREIMAN, L. et al. Arcing classifier. *The annals of statistics*, Institute of Mathematical Statistics, v. 26, n. 3, p. 801–849, 1998. Cited 2 times on pages 69 and 70.

CAI, T.; LIU, W. Adaptive thresholding for sparse covariance matrix estimation. *Journal of the American Statistical Association*, Taylor & Francis, v. 106, n. 494, p. 672–684, 2011. Cited 3 times on pages 29, 53, and 56.

CAI, T. T. et al. Optimal rates of convergence for covariance matrix estimation. *The Annals of Statistics*, Institute of Mathematical Statistics, v. 38, n. 4, p. 2118–2144, 2010. Cited 2 times on pages 53 and 54.

CANGELOSI, R.; GORIELY, A. Component retention in principal component analysis with application to cdna microarray data. *Biology direct*, BioMed Central Ltd, v. 2, n. 2, p. 1–21, 2007. Cited on page 41.

CANTY, A.; RIPLEY, B. boot: Bootstrap r (s-plus) functions. *R package version*, v. 1, n. 7, 2012. Cited on page 83.

CAPUTO, B. et al. Appearance-based object recognition using svms: Which kernel should i use? In: *Proc of NIPS workshop on Statistical methods for computational experiments in visual processing and computer vision, Whistler.* [S.l.: s.n.], 2002. v. 2002. There is no citation on text.

CARUANA, R.; KARAMPATZIAKIS, N.; YESSENALINA, A. An empirical evaluation of supervised learning in high dimensions. In: ACM. *Proceedings of the 25th international conference on Machine learning.* [S.l.], 2008. p. 96–103. Cited on page 62.

CARUANA, R.; NICULESCU-MIZIL, A. An empirical comparison of supervised learning algorithms. In: ACM. *Proceedings of the 23rd international conference on Machine learning*. [S.l.], 2006. p. 161–168. Cited 2 times on pages 61 and 86.

CHANG, C.-C.; LIN, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, v. 2, p. 1–27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>. Cited on page 88.

CHEN, T.; HE, T.; BENESTY, M. *xgboost: Extreme Gradient Boosting*. [S.l.], 2015. R package version 0.4-1. Disponível em: <http://CRAN.R-project.org/package=xgboost>. Cited on page 90.

CHEN, Y. et al. Shrinkage algorithms for mmse covariance estimation. *Signal Processing, IEEE Transactions on*, p. 5016–5029, 2010. Cited 2 times on pages 26 and 35.

CHRISTENSEN, B. C. et al. Aging and environmental exposures alter tissue-specific dna methylation dependent upon cpg island context. *PLoS Genet*, v. 5, n. 8, 2009. Cited on page 46.

CRESSIE, N. *Statistics for spatial data*. [S.l.]: J. Wiley, 1993. (Wiley series in probability and mathematical statistics: Applied probability and statistics). Cited on page 25.

DEGROOT, M. H.; FIENBERG, S. E. The comparison and evaluation of forecasters. *The statistician*, JSTOR, p. 12–22, 1983. Cited on page 76.

DEMIGUEL, V.; MARTIN-UTRERA, A.; NOGALES, F. J. Size matters: Optimal calibration of shrinkage estimators for portfolio selection. *Journal of Banking & Finance*, Elsevier, v. 37, n. 8, p. 3018–3034, 2013. Cited 2 times on pages 63 and 66.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, v. 39, n. 1, p. 1–38, 1977. Cited 2 times on pages 29 and 61.

DONOHO, D.; GAVISH, M.; JOHNSTONE, I. Optimal shrinkage of eigenvalues in the spiked covariance model. *ArXiv e-prints*, nov 2013. Cited on page 39.

DUDA, R.; HART, P.; STORK, D. *Pattern classification*. [S.l.]: Wiley, 2001. (Pattern Classification and Scene Analysis: Pattern Classification). Cited 2 times on pages 25 and 60.

EFRON, B. Better bootstrap confidence intervals. *Journal of the American statistical Association*, Taylor & Francis, v. 82, n. 397, p. 171–185, 1987. Cited on page 83.

EFRON, B. et al. Least angle regression. *The Annals of statistics*, Institute of Mathematical Statistics, v. 32, n. 2, p. 407–499, 2004. Cited on page 119.

EFRON, B.; TIBSHIRANI, R. J. *An introduction to the bootstrap*. [S.l.]: CRC press, 1994. Cited on page 83.

FAN, J.; FAN, Y.; LV, J. High dimensional covariance matrix estimation using a factor model. *Journal of Econometrics*, v. 147, n. 1, p. 186–197, 2008. Econometric modelling in finance and risk management: An overview. Cited 2 times on pages 27 and 40.

FAN, L.; POH, K. L. A comparative study of pca, ica and class-conditional ica for naïve bayes classifier. In: *Computational and Ambient Intelligence*. [S.l.]: Springer, 2007. p. 16–22. Cited on page 62.

FANG, Y.; WANG, B.; FENG, Y. Tuning-parameter selection in regularized estimations of large covariance matrices. *Journal of Statistical Computation and Simulation*, Taylor & Francis, p. 1–16, 2015. Cited on page 57.

FASIOLO, M. *An introduction to mvnfast. R package version 0.1.2.* [S.l.], 2014. Disponível em: <http://cran.r-project.org/web/packages/mvnfast/vignettes/mvnfast.html>. Cited on page 79.

FENG, L. et al. *imputeR: A General Imputation Framework in R.* [S.l.], 2014. R package version 1.0.0. Disponível em: <http://CRAN.R-project.org/package=imputeR>. Cited on page 90.

FERNÁNDEZ-DELGADO, M. et al. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, JMLR. org, v. 15, n. 1, p. 3133–3181, 2014. Cited on page 87.

FERREIRA, A. J.; FIGUEIREDO, M. A. Boosting algorithms: A review of methods, theory, and applications. In: *Ensemble Machine Learning*. [S.l.]: Springer, 2012. p. 35–85. Cited 2 times on pages 67 and 72.

FERREIRA, J.; DENISON, D.; HAND, D. Weighted naive bayes modelling for data mining. 2001. Cited 2 times on pages 29 and 62.

FIGUEIREDO, M. A. Lecture notes on bayesian estimation and classification. *Instituto de Telecomunicacoes-Instituto Superior Tecnico*, 2004. Cited on page 60.

FILOSI, M.; VISINTAINER, R.; ALBANESE, D. *minerva: minerva: Maximal Information-Based Nonparametric Exploration R package for Variable Analysis*. [S.l.], 2014. R package version 1.4.1. Disponível em: <http://CRAN.R-project.org/package=minerva>. Cited on page 67.

FISHER, T. J.; SUN, X. Improved stein-type shrinkage estimators for the high-dimensional multivariate normal covariance matrix. *Computational Statistics & Data Analysis*, Elsevier, v. 55, n. 5, p. 1909–1918, 2011. Cited on page 37.

FOYGEL, R.; DRTON, M. Extended bayesian information criteria for gaussian graphical models. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2010. p. 604–612. Cited on page 52.

FRANK, E.; HALL, M.; PFAHRINGER, B. Locally weighted naive bayes. In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. [S.l.]: Morgan Kaufmann, 2003. p. 249–256. Cited 2 times on pages 29 and 62.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, Elsevier, v. 55, n. 1, p. 119–139, 1997. Cited 2 times on pages 29 and 68.

FREUND, Y.; SCHAPIRE, R. E. et al. Experiments with a new boosting algorithm. In: *ICML*. [S.l.: s.n.], 1996. v. 96, p. 148–156. Cited 4 times on pages 29, 68, 69, and 70.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. *The elements of statistical learning*. [S.l.]: Springer series in statistics Springer, Berlin, 2001. Cited on page 36.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, v. 9, n. 3, p. 432–441, 2008. Cited 2 times on pages 28 and 52.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, NIH Public Access, v. 33, n. 1, p. 1, 2010. Cited on page 85.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. *glasso: Graphical lasso- estimation of Gaussian graphical models*. [S.l.], 2014. R package version 1.8. Disponível em: <http://CRAN.R-project.org/package=glasso>. Cited on page 52.

FRIEDMAN, J. et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, Institute of Mathematical Statistics, v. 28, n. 2, p. 337–407, 2000. Cited 4 times on pages 29, 70, 77, and 94.

FRIEDMAN, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, JSTOR, p. 1189–1232, 2001. Cited 2 times on pages 29 and 70.

FRIEDMAN, J. H. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, Elsevier, v. 38, n. 4, p. 367–378, 2002. Cited on page 29.

FRIEDMAN, N.; GEIGER, D.; GOLDSZMIDT, M. Bayesian network classifiers. *Mach. Learn.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 29, n. 2-3, p. 131–163, nov 1997. Cited 2 times on pages 29 and 62.

FURRER, R.; BENGTSSON, T. Estimation of high-dimensional prior and posterior covariance matrices in kalman filter variants. *Journal of Multivariate Analysis*, Elsevier, v. 98, n. 2, p. 227–255, 2007. Cited 2 times on pages 28 and 54.

GARCÍA, S. et al. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, Elsevier, v. 180, n. 10, p. 2044–2064, 2010. Cited on page 83.

GEURTS, P.; ERNST, D.; WEHENKEL, L. Extremely randomized trees. *Machine learning*, Springer, v. 63, n. 1, p. 3–42, 2006. Cited on page 29.

GOLDFARB, D.; IYENGAR, G. Robust portfolio selection problems. *Mathematics of Operations Research*, v. 28, n. 1, p. 1–38, 2003. Cited on page 25.

GOLUB, G. H.; LOAN, C. F. V. *Matrix computations*. [S.l.]: JHU Press, 2012. Cited 2 times on pages 114 and 115.

GOLUB, T. R. et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, v. 286, n. 5439, p. 531–537, oct 1999. Cited on page 66.

GUAN, Y.; DY, J. G. Sparse probabilistic principal component analysis. In: *International Conference on Artificial Intelligence and Statistics*. [S.l.: s.n.], 2009. p. 185–192. Cited on page 47.

GUYON, I. et al. Result analysis of the nips 2003 feature selection challenge. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2004. p. 545–552. Cited on page 66.

HAFF, L. R. Empirical bayes estimation of the multivariate normal covariance matrix. *Ann. Statist.*, The Institute of Mathematical Statistics, v. 8, p. 586–597, 1980. Cited on page 33.

HALL, M. A decision tree-based attribute weighting filter for naive bayes. *Knowledge-Based Systems*, Elsevier, v. 20, n. 2, p. 120–126, 2007. Cited 2 times on pages 29 and 62.

HAMILTON, J. *Time Series Analysis*. [S.l.]: Princeton University Press, 1994. Cited on page 25.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. Boosting and additive trees. In: *The Elements of Statistical Learning*. [S.l.]: Springer New York, 2009, (Springer Series in Statistics). p. 337–384. Cited 2 times on pages 67 and 70.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. Linear methods for regression. In: *The Elements of Statistical Learning*. [S.l.]: Springer New York, 2009, (Springer Series in Statistics). p. 43–99. Cited 3 times on pages 117, 118, and 119.

HASTIE, T.; TIBSHIRANI, R.; WAINWRIGHT, M. *Statistical Learning with Sparsity: The Lasso and Generalizations*. [S.l.]: Taylor & Francis, 2015. (Chapman & Hall/CRC Monographs on Statistics & Applied Probability). Cited 3 times on pages 44, 46, and 52.

HAUSSER, J.; STRIMMER, K. Entropy inference and the james-stein estimator, with application to nonlinear gene association networks. *J. Mach. Learn. Res.*, JMLR.org, v. 10, p. 1469–1484, dec 2009. ISSN 1532-4435. Cited 2 times on pages 63 and 66.

HAUSSER, J.; STRIMMER, K. *entropy: Estimation of Entropy, Mutual Information and Related Quantities*. [S.l.], 2014. R package version 1.2.1. Disponível em: <http://CRAN.R-project.org/package=entropy>. Cited on page 66.

HAYKIN, S. *Kalman Filtering and Neural Networks*. [S.l.]: Wiley, 2004. (Adaptive and Cognitive Dynamic Systems: Signal Processing, Learning, Communications and Control). Cited on page 25.

HIGHAM, N. J. Computing a nearest symmetric positive semidefinite matrix. *Linear algebra and its applications*, Elsevier, v. 103, p. 103–118, 1988. Cited 3 times on pages 32, 67, and 80.

HOERL, A. E.; KENNARD, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, v. 12, n. 1, p. 55–67, 1970. Cited 2 times on pages 26 and 117.

HORNIK, K.; BUCHTA, C.; ZEILEIS, A. Open-source machine learning: R meets weka. *Computational Statistics*, Springer, v. 24, n. 2, p. 225–232, 2009. Cited on page 85.

HSIEH, C.-J. et al. Sparse inverse covariance matrix estimation using quadratic approximation. In: SHAWE-TAYLOR, J. et al. (Ed.). *Advances in Neural Information Processing Systems 24*. NIPS, 2011. p. 2330–2338. Disponível em: <http://www.cs.utexas.edu/users/sustik/QUIC>. Cited on page 52.

HSIEH, C.-J. et al. Quic: quadratic approximation for sparse inverse covariance estimation. *The Journal of Machine Learning Research*, JMLR. org, v. 15, n. 1, p. 2911–2947, 2014. Cited on page 52.

HUANG, J. Z. et al. Covariance matrix selection and estimation via penalised normal likelihood. *Biometrika*, Biometrika Trust, v. 93, n. 1, p. 85–98, 2006. Cited 3 times on pages 28, 50, and 53.

J. Hartlap; P. Simon; P. Schneider. Why your model parameter confidences might be too optimistic. unbiased estimation of the inverse covariance matrix. *A&A*, v. 464, n. 1, p. 399–404, 2007. Cited on page 25.

JACKSON, J. E. *A user's guide to principal components*. [S.l.]: John Wiley & Sons, 2005. Cited 2 times on pages 40 and 42.

JAMES, W.; STEIN, C. Estimation with quadratic loss. In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. Berkeley, Calif.: University of California Press, 1961. p. 361–379. Cited 3 times on pages 26, 33, and 63.

JIANG, L.; ZHANG, H.; CAI, Z. A novel bayes model: Hidden naive bayes. *Knowledge and Data Engineering, IEEE Transactions on*, IEEE, v. 21, n. 10, p. 1361–1371, 2009. Cited on page 62.

JOE, H. Generating random correlation matrices based on partial correlations. *Journal of Multivariate Analysis*, Elsevier, v. 97, n. 10, p. 2177–2189, 2006. Cited 2 times on pages 35 and 79.

JOHN, G. H.; LANGLEY, P. Estimating continuous distributions in bayesian classifiers. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. [S.l.], 1995. p. 338–345. Cited on page 85.

JOHNSTONE, I. M. On the distribution of the largest eigenvalue in principal components analysis. *Ann. Statist.*, The Institute of Mathematical Statistics, v. 29, n. 2, p. 295–327, 04 2001. Cited on page 25.

JOHNSTONE, I. M.; LU, A. Y. On consistency and sparsity for principal components analysis in high dimensions [with comments]. *Journal of the American Statistical Association*, Taylor & Francis, Ltd. on behalf of the American Statistical Association, v. 104, n. 486, p. 682–703, 2009. Cited 4 times on pages 27, 44, 54, and 114.

JOLLIFFE, I. *Principal Component Analysis*. [S.l.]: Springer, 2002. (Springer Series in Statistics). Cited 5 times on pages 25, 27, 40, 41, and 42.

JOLLIFFE, I. T.; TRENDAFILOV, N.; UDDIN, M. A modified principal component technique based on the lasso. *Journal of Computational and Graphical Statistics*, v. 12, p. 531–547, 2003. Cited 2 times on pages 27 and 44.

KAO, Y.-H.; ROY, B. Learning a factor model via regularized pca. *Mach. Learn.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 91, n. 3, p. 279–303, jun. 2013. Cited 2 times on pages 27 and 40.

KAROUI, N. E. Operator norm consistent estimation of large-dimensional sparse covariance matrices. *The Annals of Statistics*, JSTOR, p. 2717–2756, 2008. Cited 4 times on pages 28, 51, 53, and 55.

KEOGH, E. J.; PAZZANI, M. J. Learning augmented bayesian classifiers: A comparison of distribution-based and classification-based approaches. In: *In Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*. [S.l.: s.n.], 1999. p. 225–230. Cited 2 times on pages 29 and 62.

KNOTT, M.; BARTHOLOMEW, D. J. *Latent variable models and factor analysis*. [S.l.]: Edward Arnold, 1999. Cited on page 39.

KOHAVI, R. Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. In: *Knowledge Discovery and Data Mining*. [S.l.: s.n.], 1996. p. 202–207. Cited 2 times on pages 29 and 62.

KONONENKO, I. Semi-naive bayesian classifier. In: *Proceedings of the European Working Session on Learning on Machine Learning*. New York, NY, USA: Springer-Verlag New York, Inc., 1991. p. 206–219. Cited 2 times on pages 29 and 62.

LANGLEY, P.; SAGE, S. Induction of selective bayesian classifiers. In: *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994. p. 399–406. Cited 2 times on pages 29 and 62.

LEDOIT, O.; WOLF, M. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance*, v. 10, p. 603–621, 2003. Cited 3 times on pages 25, 26, and 58.

LEDOIT, O.; WOLF, M. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, v. 88, n. 2, p. 365–411, 2004. Cited 5 times on pages 25, 26, 33, 35, and 51.

LEE, C.-H.; GUTIERREZ, F.; DOU, D. Calculating feature weights in naive bayes with kullback-leibler measure. In: *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. [S.l.: s.n.], 2011. p. 1146–1151. Cited 2 times on pages 29 and 62.

LEISCH, F.; DIMITRIADOU, E. mlbench: Machine learning benchmark problems. *R package version*, v. 2, 2010. Cited on page 63.

LI, X. et al. *flare: Family of Lasso Regression*. [S.l.], 2014. R package version 1.5.0. Disponível em: <http://CRAN.R-project.org/package=flare>. Cited on page 53.

LI, X. et al. The flare package for high dimensional linear regression and precision matrix estimation in r. *Journal of Machine Learning Research*, v. 16, p. 553–557, 2015. Cited on page 53.

LIAW, A.; WIENER, M. Classification and regression by randomforest. *R News*, v. 2, n. 3, p. 18–22, 2002. Disponível em: <http://CRAN.R-project.org/doc/Rnews/>. Cited on page 85.

LICHMAN, M. *UCI Machine Learning Repository*. 2013. Disponível em: <http://archive.ics.uci.edu/ml>. Cited 5 times on pages 66, 76, 82, 84, and 129.

LINDSEY, J. *Modelling Frequency and Count Data*. [S.l.]: Clarendon Press, 1995. (Oxford science publications). Cited on page 25.

LIU, H.; ROEDER, K.; WASSERMAN, L. Stability approach to regularization selection (stars) for high dimensional graphical models. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2010. p. 1432–1440. Cited on page 52.

LIU, H.; WANG, L. Tiger: A tuning-insensitive approach for optimally estimating gaussian graphical models. *arXiv preprint arXiv:1209.2437*, 2012. Cited on page 53.

LYNCH, M.; WALSH, B. *Genetics and Analysis of Quantitative Traits*. [S.l.]: Sinauer, 1998. Cited on page 25.

MCNICHOLAS, P. D. et al. *pgmm: Parsimonious Gaussian Mixture Models*. [S.l.], 2015. R package version 1.2. Disponível em: <http://CRAN.R-project.org/package=pgmm>. Cited on page 40.

MCNICHOLAS, P. D.; MURPHY, T. B. Parsimonious gaussian mixture models. *Statistics and Computing*, Springer US, v. 18, n. 3, p. 285–296, 2008. Cited 2 times on pages 27 and 40.

MEINSHAUSEN, N.; BüHLMANN, P. High-dimensional graphs and variable selection with the lasso. *Ann. Statist.*, The Institute of Mathematical Statistics, v. 34, n. 3, p. 1436–1462, 2006. Cited 2 times on pages 28 and 52.

MEYER, D. et al. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. [S.l.], 2015. R package version 1.6-7. Disponível em: <http://CRAN.R-project.org/package=e1071>. Cited on page 85.

MICHIE, D. et al. (Ed.). *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, NJ, USA: Ellis Horwood, 1994. Cited on page 61.

MINKA, T. P. Automatic choice of dimensionality for pca. In: *NIPS*. [S.l.: s.n.], 2000. v. 13, p. 598–604. Cited 3 times on pages 41, 42, and 44.

MITCHELL, T. M. *Machine Learning*. 1. ed. [S.l.]: McGraw-Hill, Inc., 1997. Cited 2 times on pages 47 and 61.

MITCHELL, T. M. Generative and discriminative classifiers: Naive bayes and logistic regression. jan 2015. Cited on page 59.

MONTGOMERY, D. C. *Introduction to statistical quality control*. [S.l.]: John Wiley & Sons, 2007. Cited 2 times on pages 43 and 111.

MOORE, A. W. Gaussian bayes classifiers. Lecture Notes. September 2001. Cited 2 times on pages 29 and 60.

MURPHY, A. H.; WINKLER, R. L. Reliability of subjective probability forecasts of precipitation and temperature. *Applied Statistics*, JSTOR, p. 41–47, 1977. Cited on page 74.

NG, A. Y.; JORDAN, M. I. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In: DIETTERICH, T.; BECKER, S.; GHAHRAMANI, Z. (Ed.). *Advances in Neural Information Processing Systems 14*. [S.l.]: MIT Press, 2002. p. 841–848. Cited on page 59.

NICULESCU-MIZIL, A.; CARUANA, R. Predicting good probabilities with supervised learning. In: ACM. *Proceedings of the 22nd international conference on Machine learning.* [S.l.], 2005. p. 625–632. Cited on page 74.

OPGEN-RHEIN, R.; STRIMMER, K. Accurate ranking of differentially expressed genes by a distribution-free shrinkage approach. *Statistical Applications in Genetics and Molecular Biology*, v. 6, n. 1, 2007. Cited on page 38.

OTHERS, G. R. with contributions from. *gbm: Generalized Boosted Regression Models.* [S.l.], 2015. R package version 2.1.1. Disponível em: <http://CRAN.R-project.org/package=gbm>. Cited on page 85.

PARK, T.; CASELLA, G. The bayesian lasso. *Journal of the American Statistical Association*, Taylor & Francis, v. 103, n. 482, p. 681–686, 2008. Cited on page 47.

PAUL, D. Asymptotics of sample eigenstructure for a large dimensional spiked covariance model. *Statistica Sinica*, v. 17, p. 1617–1642, 2007. Cited on page 27.

PAZZANI, M. J. Constructive induction of cartesian product attributes. In: *Feature Extraction, Construction and Selection.* [S.l.]: Springer US, 1998, (The Springer International Series in Engineering and Computer Science, v. 453). p. 341–354. Cited 2 times on pages 29 and 62.

PLATT, J. et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, Cambridge, MA, v. 10, n. 3, p. 61–74, 1999. Cited on page 74.

POURAHMADI, M. Covariance estimation: The glm and regularization perspectives. *Statist. Sci.*, The Institute of Mathematical Statistics, v. 26, n. 3, p. 369–387, 08 2011. Cited 5 times on pages 26, 33, 50, 52, and 53.

POURAHMADI, M. *High-Dimensional Covariance Estimation: With High-Dimensional Data.* [S.l.]: Wiley, 2013. (Wiley Series in Probability and Statistics). Cited 3 times on pages 26, 33, and 114.

POURAHMADI, M.; WU, W. B. Covariance matrix estimation (high dimensional). *Encyclopedia of Environmetrics*, Wiley Online Library, 2013. Cited 2 times on pages 51 and 55.

QIU, W.; JOE., H. *clusterGeneration: Random Cluster Generation (with Specified Degree of Separation).* [S.l.], 2015. R package version 1.3.4. Disponível em: <http://CRAN.R-project.org/package=clusterGeneration>. Cited 2 times on pages 35 and 79.

R Core Team. *R: A Language and Environment for Statistical Computing.* Vienna, Austria, 2015. Disponível em: <https://www.R-project.org/>. Cited on page 30.

RAICHE, G. *an R package for parallel analysis and non graphical solutions to the Cattell scree test.* [S.l.], 2010. R package version 2.3.3. Disponível em: <http://CRAN.R-project.org/package=nFactors>. Cited on page 40.

RAMEY, J. A. *datamicroarray: Collection of Data Sets for Classification.* [S.l.], 2013. R package version 0.2.2. Disponível em: <https://github.com/ramey/datamicroarray>. Cited 2 times on pages 47 and 66.

RAO, C. R.; MITRA, S. K. *Generalized inverse of matrices and its applications*. [S.l.]: Wiley New York, 1971. Cited on page 115.

RATANAMAHATANA, C. a.; GUNOPULOS, D. Feature selection for the naive bayesian classifier using decision trees. *Applied artificial intelligence*, Taylor & Francis, v. 17, n. 5-6, p. 475–487, 2003. Cited on page 62.

RESHEF, D. N. et al. Detecting novel associations in large data sets. *science*, American Association for the Advancement of Science, v. 334, n. 6062, p. 1518–1524, 2011. Cited on page 67.

RIFKIN, R.; KLAUTAU, A. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, JMLR. org, v. 5, p. 101–141, 2004. Cited on page 77.

ROBERTSON, T. et al. *Order restricted statistical inference*. [S.l.]: Wiley New York, 1988. Cited on page 74.

ROTHMAN, A. J. et al. Sparse permutation invariant covariance estimation. *Electron. J. Statist.*, The Institute of Mathematical Statistics and the Bernoulli Society, v. 2, p. 494–515, 2008. Cited on page 28.

ROTHMAN, A. J.; LEVINA, E.; ZHU, J. Generalized thresholding of large covariance matrices. *Journal of the American Statistical Association*, Taylor & Francis, v. 104, n. 485, p. 177–186, 2009. Cited 3 times on pages 28, 53, and 56.

ROTHMAN, A. J.; LEVINA, E.; ZHU, J. A new approach to cholesky-based covariance regularization in high dimensions. *Biometrika*, p. 539–550, 2010. Cited 4 times on pages 28, 50, 53, and 54.

SCHÄFER, J. et al. *corpcor: Efficient Estimation of Covariance and (Partial) Correlation*. [S.l.], 2014. R package version 1.6.7. Disponível em: <http://CRAN.R-project.org/package=corpcor>. Cited on page 38.

SCHÄFER, J.; STRIMMER, K. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical applications in genetics and molecular biology*, v. 4, n. 1, 2005. Cited 2 times on pages 32 and 38.

SCHAPIRE, R. E. Explaining adaboost. In: *Empirical inference*. [S.l.]: Springer, 2013. p. 37–52. Cited on page 70.

SCHAPIRE, R. E.; FREUND, Y. *Boosting: Foundations and algorithms*. [S.l.]: MIT press, 2012. Cited 2 times on pages 67 and 70.

SCHAPIRE, R. E.; SINGER, Y. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, Springer, v. 37, n. 3, p. 297–336, 1999. Cited 2 times on pages 77 and 94.

SCHNABEL, R. B.; ESKOW, E. A revised modified cholesky factorization algorithm. *SIAM J. on Optimization*, Society for Industrial and Applied Mathematics, v. 9, n. 4, p. 1135–1148, apr 1999. Cited on page 28.

SHEN, H.; HUANG, J. Z. Sparse principal component analysis via regularized low rank matrix approximation. *J. Multivar. Anal.*, Academic Press, Inc., v. 99, n. 6, p. 1015–1034, jul 2008. Cited on page 27.

SIMM, J.; de Abril, I. M.; SUGIYAMA, M. *Tree-Based Ensemble Multi-Task Learning Method for Classification and Regression*. [S.l.], 2014. v. 97, n. 6, 1677–1681 p. Disponível em: <http://CRAN.R-project.org/package=extraTrees>. Cited on page 85.

SØRLIE, T. et al. Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proceedings of the National Academy of Sciences*, v. 98, p. 10869–10874, sep 2001. Cited on page 66.

SPEED, T. A correlation for the 21st century. *Science*, Citeseer, v. 334, n. 6062, p. 1502–1503, 2011. Cited on page 67.

STACKLIES, W. et al. pcamethods – a bioconductor package providing pca methods for incomplete data. *Bioinformatics*, v. 23, p. 1164–1167, 2007. Cited on page 40.

STEIN, C. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In: *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. Berkeley, Calif.: University of California Press, 1956. p. 197–206. Cited 4 times on pages 25, 26, 31, and 33.

THERNEAU, T.; ATKINSON, B.; RIPLEY, B. *rpart: Recursive Partitioning and Regression Trees*. [S.l.], 2015. R package version 4.1-10. Disponível em: <http://CRAN.R-project.org/package=rpart>. Cited on page 85.

TIBSHIRANI, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, v. 58, p. 267–288, 1996. Cited 3 times on pages 27, 118, and 119.

TIPPING, M. E.; BISHOP, C. M. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Wiley Online Library, v. 61, n. 3, p. 611–622, 1999. Cited 3 times on pages 27, 39, and 41.

TREES, H. L. V. *Detection, estimation, and modulation theory*. [S.l.]: John Wiley & Sons, 2004. Cited on page 35.

van Buuren, S.; GROOTHUIS-OUDSHOORN, K. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, v. 45, n. 3, p. 1–67, 2011. Disponível em: <http://www.jstatsoft.org/v45/i03/>. Cited on page 90.

VEZHNEVETS, A.; VEZHNEVETS, V. Modest adaboost-teaching adaboost to generalize better. In: *Graphicon*. [S.l.: s.n.], 2005. v. 12, n. 5, p. 987–997. Cited on page 94.

WAGAMAN, A. S.; LEVINA, E. Discovering sparse covariance structures with the isomap. *Journal of Computational and Graphical Statistics*, v. 18, n. 3, p. 551–572, 2009. Cited 2 times on pages 28 and 51.

WANG, B. *CVTuningCov: Regularized Estimators of Covariance Matrices with CV Tuning*. [S.l.], 2014. R package version 1.0. Disponível em: <http://CRAN.R-project.org/package=CVTuningCov>. Cited 2 times on pages 37 and 57.

WANG, Y. et al. *mvabund: Statistical Methods for Analysing Multivariate Abundance Data*. [S.l.], 2015. R package version 3.10.4. Disponível em: <http://CRAN.R-project.org/package=mvabund>. Cited on page 36.

WARTON, D. I. Penalized normal likelihood and ridge regularization of correlation and covariance matrices. *Journal of the American Statistical Association*, v. 103, n. 481, p. 340–349, 2008. Cited 2 times on pages 33 and 35.

WASSERMAN, L.; ROEDER, K. High dimensional variable selection. *Annals of statistics*, NIH Public Access, v. 37, n. 5A, p. 2178, 2009. Cited on page 37.

WEBB, G. I.; BOUGHTON, J. R.; WANG, Z. Not so naive bayes: aggregating one-dependence estimators. *Machine learning*, Springer, v. 58, n. 1, p. 5–24, 2005. Cited on page 62.

WEBB, G. I. et al. Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive bayesian classification. *Machine Learning*, Springer, v. 86, n. 2, p. 233–272, 2012. Cited on page 62.

WEIHS, C. et al. klar analyzing german business cycles. In: BAIER, D.; DECKER, R.; SCHMIDT-THIEME, L. (Ed.). *Data Analysis and Decision Support*. Berlin: Springer-Verlag, 2005. p. 335–343. Cited on page 85.

WITTEN, D. et al. *PMA: Penalized Multivariate Analysis*. [S.l.], 2013. R package version 1.0.9. Disponível em: <http://CRAN.R-project.org/package=PMA>. Cited on page 46.

WITTEN, D. M.; FRIEDMAN, J. H.; SIMON, N. New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, Taylor & Francis, v. 20, n. 4, p. 892–900, 2011. Cited on page 52.

WITTEN, D. M.; TIBSHIRANI, R. Covariance-regularized regression and classification for high dimensional problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Blackwell Publishing Ltd, v. 71, n. 3, p. 615–636, 2009. Cited on page 27.

WITTEN, D. M.; TIBSHIRANI, R.; HASTIE, T. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, Oxford University Press, 2009. Cited 2 times on pages 27 and 46.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, IEEE, v. 1, n. 1, p. 67–82, 1997. Cited on page 87.

WONG, F.; CARTER, C. K.; KOHN, R. Efficient estimation of covariance selection models. *Biometrika*, v. 90, n. 4, p. 809–830, 2003. Cited on page 28.

WU, W. B.; POURAHMADI, M. Nonparametric estimation of large covariance matrices of longitudinal data. *Biometrika*, Biometrika Trust, v. 90, n. 4, p. 831–844, 2003. Cited 5 times on pages 27, 28, 47, 50, and 53.

XUE, L.; MA, S.; ZOU, H. Positive-definite l1-penalized estimation of large covariance matrices. *Journal of the American Statistical Association*, Taylor & Francis Group, v. 107, n. 500, p. 1480–1491, 2012. Cited on page 56.

YAN, Y. *FinCovRegularization: Covariance Matrix Estimation and Regularization for Finance*. [S.l.], 2015. R package version 1.0.0. Disponível em: <http://CRAN.R-project.org/package=FinCovRegularization>. Cited on page 57.

YANG, R.; BERGER, J. O. Estimation of a covariance matrix using the reference prior. *The Annals of Statistics*, JSTOR, p. 1195–1211, 1994. Cited on page 32.

YUAN, M.; LIN, Y. Model selection and estimation in the gaussian graphical model. *Biometrika*, v. 94, n. 1, p. 19–35, 2007. Cited 3 times on pages 25, 28, and 51.

ZADROZNY, B.; ELKAN, C. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In: *ICML*. [S.l.: s.n.], 2001. v. 1, p. 609–616. Cited 4 times on pages 62, 74, 75, and 86.

ZAIDI, N. A. et al. Alleviating naive bayes attribute independence assumption by attribute weighting. *Journal of Machine Learning Research*, v. 14, p. 1947–1988, 2013. Cited 2 times on pages 29 and 62.

ZHANG, M.-L.; nA, J. M. P.; ROBLES, V. Feature selection for multi-label naive bayes classification. *Information Sciences*, v. 179, n. 19, p. 3218–3229, 2009. Cited 2 times on pages 29 and 62.

ZHAO, T. et al. The huge package for high-dimensional undirected graph estimation in r. *The Journal of Machine Learning Research*, JMLR. org, v. 13, n. 1, p. 1059–1062, 2012. Cited on page 52.

ZHAO, T. et al. *huge: High-dimensional Undirected Graph Estimation*. [S.l.], 2014. R package version 1.2.6. Disponível em: <http://CRAN.R-project.org/package=huge>. Cited on page 52.

ZHENG, F. et al. Subsumption resolution: an efficient and effective technique for semi-naive bayesian learning. *Machine learning*, Springer, v. 87, n. 1, p. 93–125, 2012. Cited on page 62.

ZHENG, Z.; WEBB, G. Lazy learning of bayesian rules. *Machine Learning*, Kluwer Academic Publishers, v. 41, n. 1, p. 53–84, 2000. Cited 2 times on pages 29 and 62.

ZHU, J. et al. Multi-class adaboost. *Statistics and its Interface*, v. 2, n. 3, p. 349–360, 2009. Cited on page 77.

ZOU, H.; HASTIE, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Blackwell Publishing Ltd, v. 67, n. 2, p. 301–320, 2005. Cited 3 times on pages 27, 45, and 119.

ZOU, H.; HASTIE, T. *elasticnet: Elastic-Net for Sparse Estimation and Sparse PCA*. [S.l.], 2012. R package version 1.1. Disponível em: <http://CRAN.R-project.org/package=elasticnet>. Cited on page 46.

ZOU, H.; HASTIE, T.; TIBSHIRANI, R. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, v. 15, n. 2, p. 265–286, 2006. Cited 4 times on pages 27, 44, 45, and 47.

Appendix

APPENDIX A – The Multivariate Normal Distribution

A continuous random variable $x \in (-\infty, \infty)$ follows a normal distribution, $x \sim \mathcal{N}(x|\mu, \sigma^2)$, if it has the following probability density function (p.d.f.):

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right), \tag{A.1}$$

which is governed by two parameters, the mean $\mu = \boldsymbol{E}(x)$ and the variance $\sigma^2 = \mathrm{var}(x)$. The inverse of the variance $\mathrm{inv}(\sigma^2) = 1/\sigma^2$ is called the precision, and the square root of the variance $\mathrm{std}(x) = \sigma$ is called the standard deviation. The term in the exponent of (A.1) can be written as follows:

$$d(x, \mu) = (x-\mu)\left(\sigma^2\right)^{-1}(x-\mu) \tag{A.2}$$

This quantity measures the squared standardized distance from x to the mean μ.

For a p-dimensional vector $\boldsymbol{x} = [x_1, \ldots, x_p]$, the normal distribution is governed by a p-dimensional mean vector $\boldsymbol{\mu} = \boldsymbol{E}(\boldsymbol{x})$ and a $p \times p$ covariance matrix $\boldsymbol{\Sigma} = \mathrm{cov}(\boldsymbol{x})$ that must be symmetric and positive-definite. The main diagonal elements of $\boldsymbol{\Sigma}$ are the variances of the x's and the off-diagonal elements are the covariances. Now the squared standardized distance from $\boldsymbol{x}$ to $\boldsymbol{\mu}$ is

$$D(\boldsymbol{x}, \boldsymbol{\mu}) = (\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu}) \tag{A.3}$$

The multivariate normal density function is obtained simply by replacing the standardized distance in (A.2) by the multivariate generalized distance in (A.3), commonly referred to as the squared Mahalanobis distance, and changing the constant term $1/\sqrt{2\pi\sigma^2}$ to a more general form that makes the area under the probability density function unity regardless of the value of p (MONTGOMERY, 2007). Therefore, the multivariate normal probability density function is

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right\} \tag{A.4}$$

where $-\infty < x_j < \infty$, $j = 1, 2, \ldots, p$. The inverse of the covariance matrix $\mathrm{inv}\left(\boldsymbol{\Sigma}\right) = \boldsymbol{\Sigma}^{-1}$ is the precision matrix, which is also symmetric and positive-definite. Given a random sample of size N, say $\boldsymbol{X} = \{\boldsymbol{x}_n\}_{n=1}^{N}$, from a multivariate normal distribution, the sample mean vector and sample covariance matrix are unbiased estimators of the corresponding population quantities, that is

$$\boldsymbol{E}\left(\overline{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n\right) = \boldsymbol{\mu} \text{ and } \boldsymbol{E}\left(\boldsymbol{S} = \frac{1}{N-1}\sum_{n=1}^{N}(\boldsymbol{x}_n - \overline{\boldsymbol{x}})(\boldsymbol{x}_n - \overline{\boldsymbol{x}})^T\right) = \boldsymbol{\Sigma}. \tag{A.5}$$

APPENDIX B – Covariance Matrix Decompositions

This section provides a brief review of the most common matrix decompositions applied to the covariance matrix.

B.1 Variance-Correlation Decomposition

The variance-correlation decomposition expresses the covariance matrix $\boldsymbol{\Sigma}_{p\times p}$ in terms of the standard deviations and correlations of the random variables $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_p]$:

$$\boldsymbol{\Sigma} = \boldsymbol{DRD} \tag{B.1}$$

where $\boldsymbol{D} = \operatorname{diag}(\sigma_{11}, \dots, \sigma_{pp})$ is the diagonal matrix of standard deviations and $R = (r_{ij})$ is the correlation matrix of $\boldsymbol{X}$, where

$$r_{ij} = \operatorname{corr}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sigma_{ij}^2}{\sigma_{ii}\sigma_{jj}}. \tag{B.2}$$

Often, it is useful to interpret the matrix $\boldsymbol{R}$ as the covariance matrix of the standardized random variables $(\boldsymbol{x}_i - \mu_i)/\sigma_{ii}$ with $i = 1, \dots, p$.

B.2 Spectral Decomposition

The spectral or eigenvalue-eigenvector decomposition of a covariance matrix $\boldsymbol{\Sigma}_{p\times p}$ is given by

$$\boldsymbol{\Sigma} = \boldsymbol{P\Lambda P}^T = \sum_{i=1}^{p} \lambda_i \boldsymbol{e}_i \boldsymbol{e}_i^T, \tag{B.3}$$

where $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \dots, \lambda_p)$ is a diagonal matrix of the ordered eigenvalues of $\boldsymbol{\Sigma}$ and $\boldsymbol{P} = [\boldsymbol{e}_1, \dots, \boldsymbol{e}_p]$ is a orthogonal matrix ($\boldsymbol{PP}^T = \boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{I}$) with the corresponding normalized eigenvectors as its columns. For rectangular matrices, a similar role is played by the *singular value decomposition* (SVD). The entries of $\boldsymbol{\Lambda}$ and $\boldsymbol{P}$ have interpretations as variances and coefficients of the principal components (PCs) in principal component analysis (PCA).

The matrix $\boldsymbol{\Sigma}$ is nonnegative definite, if and only if all its eigenvalues are nonnegative, and is positive definite, if and only if all its eigenvalues are positive. The *condition number* of a nonnegative definite matrix is defined as λ_1/λ_p, that is, the ratio of its largest and smallest eigenvalues. It will be infinity when the latter is zero.

The *operator* or *spectral* norm of $\boldsymbol{\Sigma}$ is defined by

$$\|\boldsymbol{\Sigma}\|^2 = \lambda_{\max}^2 \tag{B.4}$$

where $\lambda_{\max}$ is the largest eigenvalue of $\boldsymbol{\Sigma}$. Two matrices that are close in some sense, for example, in the operator norm, have eigenvalues and eigenvectors that are also close (POURAHMADI, 2013). In particular, consistency or convergence of a covariance matrix estimator in the operator norm guarantees the convergence of their PCs loadings (JOHNSTONE; LU, 2009).

B.3 Cholesky and Modified-Cholesky Decompositions

The standard Cholesky decomposition of a symmetric positive definite square matrix is of the form (GOLUB; LOAN, 2012)

$$\boldsymbol{\Sigma} = \boldsymbol{C}\boldsymbol{C}^T, \tag{B.5}$$

where $\boldsymbol{C} = (c_{ij})$ is a unique lower-triangular matrix with positive diagonal entries. Statistical interpretation of the entries of $\boldsymbol{C}$ is difficult in its present form for a given covariance matrix. However, $\boldsymbol{C}$ can be reduced to an unit lower triangular matrix through multiplication by the inverse of $\boldsymbol{D} = \text{diag}(c_{11}, \ldots, c_{pp})$, as follows:

$$\boldsymbol{\Sigma} = \boldsymbol{C}\boldsymbol{D}^{-1}\boldsymbol{D}\boldsymbol{D}\boldsymbol{D}^{-1}\boldsymbol{C}^T = \boldsymbol{L}\boldsymbol{D}^2\boldsymbol{L}^T, \tag{B.6}$$

where $\boldsymbol{L} = \boldsymbol{C}\boldsymbol{D}^{-1}$, referred to as the modified Cholesky factor, is obtained from $\boldsymbol{C}$ by dividing the entries of its i^{th} column by c_{ii}. This is usually called the modified Cholesky decomposition of $\boldsymbol{\Sigma}$, which makes the task of statistical interpretation of the diagonal entries of $\boldsymbol{C}$ and the resulting unit lower triangular matrix much easier (POURAHMADI, 2013). The factorization (B.6) can also be written in the forms

$$\boldsymbol{P}\boldsymbol{\Sigma}\boldsymbol{P}^T = \boldsymbol{D}^2,\ \boldsymbol{\Sigma}^{-1} = \boldsymbol{P}^T\boldsymbol{D}^{-2}\boldsymbol{P} \tag{B.7}$$

where $\boldsymbol{P} = \boldsymbol{L}^{-1}$ is a unit triangular matrix. The first identity in (B.7) is similar to the spectral decomposition, in which $\boldsymbol{\Sigma}$ is diagonalized by a lower triangular matrix. The second identity is, in fact, the modified Cholesky decomposition of the precision matrix $\boldsymbol{\Sigma}^{-1}$.

B.4 Singular Value Decomposition

Given an $N \times p$ data matrix $\boldsymbol{X}$, the singular value decomposition (SVD) factorizes $\boldsymbol{X}$ as (GOLUB; LOAN, 2012):

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T \tag{B.8}$$

where $\boldsymbol{U} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{V} \in \mathbb{R}^{p\times p}$ are orthogonal matrices and $\boldsymbol{D} \in \mathbb{R}^{N\times p}$ is a diagonal matrix with diagonal entries in decreasing order. The non-zero elements d_i's of $\boldsymbol{D}$ are the singular values of $\boldsymbol{X}$, the columns $\boldsymbol{u}_i$'s of $\boldsymbol{U}$ are the left singular vectors of $\boldsymbol{X}$, and the columns $\boldsymbol{v}_i$'s of $\boldsymbol{V}$ are the right singular vectors of $\boldsymbol{X}$. The best rank-q approximation to $\boldsymbol{X}$, $\hat{\boldsymbol{X}}_q$, in both Frobenius norm and operator norm is given by

$$\hat{\boldsymbol{X}}_q = \arg\min_{\boldsymbol{X}_q} \|\boldsymbol{X} - \boldsymbol{X}_q\|_F = \sum_{i=1}^{q} d_i \boldsymbol{u}_i \boldsymbol{v}_i^T \tag{B.9}$$

and $\|\boldsymbol{X} - \hat{\boldsymbol{X}}_q\|_2 = d_{q+1}$. This means that the smallest singular value of $\boldsymbol{X}$ is the 2-norm distance of $\boldsymbol{X}$ to the set of all rank-deficient matrices (GOLUB; LOAN, 2012). The SVD of $\boldsymbol{X}$ is widely used in the context of linear regression to compute the pseudo-inverse or Moore-Penrose inverse of $\boldsymbol{X}$

$$\boldsymbol{X}^+ = \boldsymbol{V}^T\boldsymbol{D}^{-1}\boldsymbol{U} \tag{B.10}$$

which gives the unique least-squares solution to the regression coefficients when $N > p$, and the minimum-norm least-squares solution when $p > N$ (RAO; MITRA, 1971). The inverse $\boldsymbol{D}^{-1} = \text{diag}(1/d_1, \dots, 1/d_p)$ is simply the inverse elements of $\boldsymbol{D}$ along the diagonal.

The SVD of $\boldsymbol{X}$ is also closely related to the spectral decomposition of $\boldsymbol{X}^T\boldsymbol{X}$. Substituting the SVD of $\boldsymbol{X}$ into $\boldsymbol{X}^T\boldsymbol{X}$

$$\begin{aligned} \boldsymbol{X}^T\boldsymbol{X} &= \left(\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T\right)^T \left(\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T\right) \\ &= \left(\boldsymbol{V}\boldsymbol{D}^T\boldsymbol{U}^T\right)\left(\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T\right) \\ &= \boldsymbol{V}\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{V}^T \\ &= \boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}^T, \end{aligned} \tag{B.11}$$

then $\boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}^T$ is the spectral decomposition of $\boldsymbol{X}^T\boldsymbol{X}$. Thus, the i^{th} eigenvalue of $\boldsymbol{X}^T\boldsymbol{X}$, $\lambda_i = d_i^2$, is the square of the i^{th} singular value of $\boldsymbol{X}$, and the corresponding i^{th} eigenvector of $\boldsymbol{X}^T\boldsymbol{X}$ is the i^{th} right singular vector of $\boldsymbol{X}$. Assuming, without loss of generality, that the column means of $\boldsymbol{X}$ are all 0, it is easy to show that the spectral decomposition of the empirical covariance estimate

$$\boldsymbol{S} = \frac{1}{N-1}\boldsymbol{X}^T\boldsymbol{X} \tag{B.12}$$

is given by

$$\boldsymbol{S} = \boldsymbol{V}\left(\frac{\boldsymbol{D}^2}{N-1}\right)\boldsymbol{V}^T \tag{B.13}$$

So, the i^{th} eigenvalue of $\boldsymbol{S}$ is $\lambda_i = d_i^2/(N-1)$, and the corresponding normalized eigenvector is $\boldsymbol{e}_i = \boldsymbol{v}_i$.

APPENDIX C – Regularized Linear Regression

This section provides a brief review of the most common regularization techniques applied to linear regression.

C.1 Least Squares

Given p predictors $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_p]$, and considering the usual linear regression model with N observations, the response vector $\boldsymbol{y} = (y_1, \dots, y_N)^T$ is predicted by

$$\hat{\boldsymbol{y}} = \hat{\beta}_0 + \sum_{j=1}^{p} \boldsymbol{x}_j \hat{\beta}_j. \tag{C.1}$$

The most popular model fitting procedure to (C.1) is *ordinary least squares* (OLS), which produces the vector of coefficients $\hat{\boldsymbol{\beta}} = (\hat{\beta}_0, \dots, \hat{\beta}_p)^T$ by minimizing the residual sum of squares (HASTIE; TIBSHIRANI; FRIEDMAN, 2009b)

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2, \tag{C.2}$$

Assuming that $\boldsymbol{X}$ has full column rank, and hence $\boldsymbol{X}^T\boldsymbol{X}$ is positive definite, the unique OLS solution is

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1} \boldsymbol{X}^T\boldsymbol{y}. \tag{C.3}$$

C.2 Ridge Regression

It is well know that OLS does poorly in both prediction and interpretation for high-dimensional data. If the number of predictors p exceeds the number of observations N, the matrix $\boldsymbol{X}^T\boldsymbol{X}$ becomes singular, and the OLS estimates are not uniquely defined. Moreover, in the presence of high correlation among predictors, the OLS estimation may become unstable and exhibit high variance. By imposing a size constraint t on the coefficients estimates, ridge regression (HOERL; KENNARD, 1970) alleviates this problem. The ridge coefficients minimize a penalized residual sum of squares (HASTIE; TIBSHIRANI; FRIEDMAN, 2009b)

$$\hat{\boldsymbol{\beta}}_{\text{ridge}}(t) = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X\beta}\|^2, \text{ subject to } \sum_{j=1}^{p} \beta_j^2 \leq t, \tag{C.4}$$

or equivalently

$$\hat{\boldsymbol{\beta}}_{\text{ridge}}(\alpha) = \arg\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{y} - \boldsymbol{X\beta}\|^2 + \alpha \sum_{j=1}^{p} \beta_j^2 \right\}, \tag{C.5}$$

and their solutions are given by the closed form expression

$$\hat{\boldsymbol{\beta}}_{\text{ridge}}(\alpha) = \left(\boldsymbol{X}^T \boldsymbol{X} + \alpha \boldsymbol{I}^{(0)} \right)^{-1} \boldsymbol{X}^T \boldsymbol{y}. \tag{C.6}$$

where $\boldsymbol{I}^{(0)}$ is the identity matrix with zero on the first diagonal element to suppress penalization over the intercept β_0, and $\alpha \geq 0$ controls the amount of shrinkage toward zero applied to the remaining coefficients. There is a one-to-one correspondence between the parameters α and t.

C.3 Lasso Regression

Despite guaranteeing a unique solution to the regression coefficients by adding a penalty α to the diagonal entries of $\boldsymbol{X}^T\boldsymbol{X}$, ridge regression cannot produce a parsimonious model, because it always keeps all the predictors in the model. For this reason, model interpretation becomes impracticable when the number of predictors is large. The *least absolute shrinkage and selection operator* (LASSO) method (TIBSHIRANI, 1996) for regularization of the OLS estimator allows to set some of the regression coefficients to zero, so that the corresponding variables are essentially removed from the model. The LASSO minimizes the residual sum of squares constraining the sum of the absolute value of the coefficients. Because of the nature of this constraint it tends to produce some coefficients that are exactly zero, doing both continuous shrinkage and automatic variable selection simultaneously. The LASSO estimate is defined by (HASTIE; TIBSHIRANI; FRIEDMAN, 2009b)

$$\hat{\boldsymbol{\beta}}_{\text{LASSO}}(t) = \arg\min_{\boldsymbol{\beta}} (\boldsymbol{y} - \boldsymbol{X\beta})^T (\boldsymbol{y} - \boldsymbol{X\beta}), \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq t, \tag{C.7}$$

and can also be expressed in Lagrangian form

$$\hat{\boldsymbol{\beta}}_{\text{LASSO}}(\alpha) = \arg\min_{\boldsymbol{\beta}} \left\{ (\boldsymbol{y} - \boldsymbol{X\beta})^T (\boldsymbol{y} - \boldsymbol{X\beta}) + \alpha \sum_{j=1}^{p} |\beta_j| \right\} \tag{C.8}$$

where $\alpha \geq 0$ denotes the shrinkage intensity.

The constraint in the form of the ℓ_1 penalty $\sum_{j=1}^{p} |\beta_j|$ makes the solutions nonlinear in the $\boldsymbol{y}$, and there is no closed form expression for $\boldsymbol{\beta}_{\text{LASSO}}$. So, computing the LASSO solution is a quadratic programming problem. In particular, the LARS algorithm (EFRON et al., 2004) allows to determine the complete coefficient paths in one estimation run at the cost of a single matrix inversion.

C.4 Elastic Net

The LASSO simultaneously produces both an accurate and sparse model, which makes it a favorable variable selection method. However, it has some limitations as pointed out by Zou and Hastie (2005). Firstly, in the $p > N$ case, the LASSO selects at most N variables before it saturates, because of the nature of the convex optimization problem. For example, if applied to microarray data, where it is common to have thousands of predictors (genes) with few dozen of observations, selecting at most N genes may be unsatisfactory. Secondly, for usual $N > p$ situations, if there are high correlations between predictors, it has been empirically observed by Tibshirani (1996) that the prediction performance of the LASSO is dominated by ridge regression.

The elastic net (ZOU; HASTIE, 2005) generalizes the LASSO to overcome these drawbacks. For any non-negative α_1 and α_2, the elastic net estimates $\hat{\boldsymbol{\beta}}_{\text{elastic}}$ are given as follows (HASTIE; TIBSHIRANI; FRIEDMAN, 2009b)

$$\hat{\boldsymbol{\beta}}_{\text{elastic}}(\alpha_1, \alpha_2) = \arg\min_{\boldsymbol{\beta}} \left\{ (\boldsymbol{y} - \boldsymbol{X\beta})^T (\boldsymbol{y} - \boldsymbol{X\beta}) + \alpha_1 \sum_{j=1}^{p} \left(\alpha_2 \beta_j^2 + (1 - \alpha_2)|\beta_j| \right) \right\} \tag{C.9}$$

The elastic net penalty is a convex combination of the ridge and lasso penalties. Thus the elastic net selects variables like the lasso, and shrinks together the coefficients of correlated predictors like ridge. When $p > N$, for $\alpha_2 > 0$, the elastic net can potentially include all variables in the fitted model, so this particular limitation of the LASSO is removed. Obviously, the LASSO is a special case of the elastic net when $\alpha_2 = 0$. Given a fixed α_2, the LARS-EN algorithm (ZOU; HASTIE, 2005) efficiently solves the elastic net problem for all α_1 with the computational cost of a single least squares fit.

Annex

ANNEX A – Performance Comparison between the Covariance Estimators

Comparison of seven covariance matrix estimators on simulated multivariate Gaussian data $\boldsymbol{X} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, considering two different covariance target structures: dense and sparse. The sample size is $N = \{1000, 200, 100, 50, 10\}$ and the number of variables is $p = 100$. The results for each combination of covariance target, algorithm and p/N ratio are averaged over 50 simulations.

Dataset	Ratio (p/N)	Estimator	L0 (Mean)	L0 (Std Dev.)	L1 (Mean)	L1 (Std Dev.)	L2 (Mean)	L2 (Std Dev.)	L3 (Mean)	L3 (Std Dev.)	Eig Diff (Mean)	Eig Diff (Std Dev.)	Eig Cos (Mean)	Eig Cos (Std Dev.)	Time (Mean)	Time (Std Dev.)
Nonsparse	0.1	emp	0.0000	0.0000	5.2352	0.1020	0.1114	0.1487	0.0148	0.0002	2.0370	0.3961	0.5231	0.2393	0.0084	0.0008
		diag	0.9900	0.0000	30.6214	0.2102	2387.8594	60.8841	0.0262	0.0000	4.8280	0.3935	0.2364	0.0835	0.0005	0.0005
		oas	0.0000	0.0000	7.2302	0.1146	280.9043	13.9590	0.0136	0.0002	0.3252	0.2340	0.5231	0.2393	0.0049	0.0004
		ss	0.0000	0.0000	5.4867	0.0836	145.1069	9.4420	0.0129	0.0002	0.2708	0.2165	0.5149	0.2466	0.0530	0.0010
		ppca	0.0000	0.0000	5.2461	0.0995	0.1122	0.1498	0.0148	0.0002	2.0370	0.3961	0.5231	0.2393	0.1442	0.0052
		spc	0.0000	0.0000	5.2461	0.0995	0.1122	0.1498	0.0148	0.0002	2.0370	0.3961	0.5231	0.2393	1.4352	0.0203
		quic	0.0181	0.1166	16.8137	6.0711	1315.3915	1072.5932	0.0161	0.0020	0.6336	0.6848	0.5069	0.2415	0.2002	0.0221
	0.5	emp	0.0000	0.0000	31.2689	0.6601	1.2820	1.6111	0.0332	0.0006	7.6901	1.1701	0.3379	0.2064	0.0025	0.0005
		diag	0.9900	0.0000	31.0695	0.7251	2401.6958	208.3323	0.0266	0.0001	4.0729	0.6719	0.2062	0.1005	0.0001	0.0003
		oas	0.0000	0.0000	30.4886	0.8001	2667.3658	205.8698	0.0239	0.0003	0.8506	0.5472	0.3379	0.2064	0.0011	0.0004
		ss	0.0000	0.0000	17.9626	0.4139	1026.0062	102.2525	0.0210	0.0003	1.3095	0.6132	0.3624	0.1889	0.0182	0.0008
		ppca	0.0000	0.0000	29.2532	1.3357	472.6435	183.8806	0.0345	0.0006	7.6901	1.1701	0.3379	0.2064	0.1364	0.0011
		spc	0.0000	0.0000	28.5401	1.1797	332.0929	149.0391	0.0336	0.0006	7.3362	1.1834	0.3433	0.2035	5.7241	0.0508
		quic	0.0000	0.0000	18.4306	1.4225	423.6103	362.0843	0.0291	0.0012	6.5481	1.1052	0.3411	0.2019	0.3189	0.0302
	1	emp	0.0000	0.0000	124.0701	2.6376	2.6084	3.2778	0.0467	0.0012	12.6792	1.7542	0.2732	0.1651	0.0053	0.0007
		diag	0.9900	0.0000	31.2950	0.8508	2314.3210	238.8347	0.0271	0.0002	3.2266	0.8392	0.1916	0.1140	0.0003	0.0005
		oas	0.0000	0.0000	43.8749	1.2486	4707.4781	383.8428	0.0279	0.0003	1.8157	0.8818	0.2732	0.1651	0.0008	0.0004
		ss	0.0000	0.0000	24.0354	0.7414	1622.9284	195.9154	0.0238	0.0002	1.9876	0.6819	0.2344	0.1479	0.0126	0.0006
		ppca	0.0000	0.0000	50.7225	3.2598	3760.4740	974.0532	0.0454	0.0016	12.6792	1.7542	0.2732	0.1651	0.1354	0.0109
		spc	0.2715	0.1052	44.7917	2.2324	1822.1421	570.4777	0.0420	0.0012	9.8540	1.5960	0.2215	0.1615	4.4509	0.0496
		quic	0.0000	0.0000	31.3828	1.0308	209.7720	55.8915	0.0421	0.0012	11.7322	1.7237	0.2759	0.1688	0.5360	0.0663
	2	emp	0.0000	0.0000	Inf	NA	4.6629	6.3308	0.0668	0.0019	22.6582	2.1478	0.2157	0.1534	0.0041	0.0009
		diag	0.9900	0.0000	32.4752	1.1826	2359.2651	330.9748	0.0281	0.0003	2.2743	0.9923	0.1958	0.1079	0.0001	0.0004
		oas	0.0000	0.0000	55.2546	2.2558	6920.6028	801.5507	0.0309	0.0003	2.3706	1.0248	0.2157	0.1534	0.0006	0.0005
		ss	0.0000	0.0000	29.7773	1.4106	2341.5770	394.7752	0.0262	0.0004	2.1589	0.8222	0.2494	0.1529	0.0084	0.0052
		ppca	0.0000	0.0000	64.8093	4.7472	7579.3411	1913.5136	0.0491	0.0048	22.6582	2.1478	0.2157	0.1534	0.0460	0.0008
		spc	0.9461	0.0300	43.0345	0.8542	457.4841	263.2123	0.0426	0.0017	8.3575	2.0531	0.2062	0.1261	1.7597	0.0338
		quic	0.0000	0.0000	61.2376	2.6759	157.6482	68.1426	0.0615	0.0019	21.5838	2.1182	0.2162	0.1540	1.1882	0.1699
	10	emp	0.0000	0.0000	Inf	NA	29.0719	42.2338	0.1549	0.0105	85.3937	11.1217	0.1647	0.1200	0.0031	0.0004
		diag	0.9900	0.0000	42.0673	3.4344	2439.9614	922.5596	0.0359	0.0021	4.4064	3.5366	0.1513	0.1093	0.0001	0.0004
		oas	0.0000	0.0000	66.6441	6.0270	9066.2884	2399.0901	0.0364	0.0017	3.2175	2.9795	0.1647	0.1200	0.0004	0.0005
		ss	0.0000	0.0000	40.8157	4.5225	2803.8756	1372.8393	0.0343	0.0014	2.9081	2.5342	0.1665	0.1294	0.0035	0.0005
		ppca	0.0000	0.0000	69.0361	5.3569	7341.6696	2004.9533	0.0881	0.0087	85.3937	11.1217	0.1647	0.1200	0.0038	0.0038
		spc	0.9893	0.0006	73.6408	6.4628	3196.3626	540.4285	0.0527	0.0014	9.5743	5.7172	0.1280	0.0955	0.1383	0.0325
		quic	0.0000	0.0000	170.4729	15.0094	170.8719	151.7917	0.1460	0.0101	82.2596	10.9366	0.1645	0.1204	4.6494	1.1840
Sparse	0.1	emp	0.0000	0.0000	5.2509	0.1129	0.2058	0.2653	0.0287	0.0004	4.2258	0.4667	0.3311	0.1958	0.0086	0.0009

Dataset	Ratio (p/N)	Estimator	L0 (Mean)	L0 (Std Dev.)	L1 (Mean)	L1 (Std Dev.)	L2 (Mean)	L2 (Std Dev.)	L3 (Mean)	L3 (Std Dev.)	Eig Diff (Mean)	Eig Diff (Std Dev.)	Eig Cos (Mean)	Eig Cos (Std Dev.)	Time (Mean)	Time (Std Dev.)
		diag	0.9900	0.0000	126.9900	0.9665	23223.3860	425.7737	0.0492	0.0000	7.7812	0.1809	0.0193	0.0553	0.0004	0.0005
		oas	0.0000	0.0000	25.5562	0.3312	1489.0352	57.2409	0.0248	0.0003	0.9212	0.3751	0.3311	0.1958	0.0049	0.0002
		ss	0.0000	0.0000	25.2162	0.3528	1457.4679	62.7286	0.0245	0.0003	0.8507	0.3719	0.3385	0.2050	0.0540	0.0076
		ppca	0.0000	0.0000	5.2517	0.1128	0.2054	0.2653	0.0287	0.0004	4.2258	0.4667	0.3311	0.1958	0.1463	0.0123
		spc	0.0000	0.0000	5.3420	0.1174	0.2060	0.2793	0.0288	0.0004	4.2043	0.4622	0.3398	0.2103	14.4037	0.1519
		quic	0.0000	0.0000	13.6093	2.1022	481.2512	210.0899	0.0246	0.0006	3.2006	0.4500	0.3287	0.2019	0.8017	0.0342
	0.5	emp	0.0000	0.0000	31.1940	0.5745	1.1154	1.4095	0.0643	0.0012	14.0162	1.2949	0.1884	0.1285	0.0026	0.0007
		diag	0.9900	0.0000	127.5010	2.1666	23280.6557	1008.4358	0.0499	0.0001	6.5294	0.4203	0.0339	0.0891	0.0002	0.0004
		oas	0.0000	0.0000	75.1492	1.5226	9157.0407	465.2007	0.0390	0.0003	0.5858	0.4648	0.1884	0.1285	0.0014	0.0005
		ss	0.0000	0.0000	74.8250	1.5976	9045.8728	505.8350	0.0389	0.0003	0.6563	0.4623	0.1785	0.1269	0.0192	0.0051
		ppca	0.0000	0.0000	89.7140	10.2232	10158.5253	2989.2304	0.0649	0.0025	14.0162	1.2949	0.1884	0.1285	0.1373	0.0044
		spc	0.1635	0.1361	85.6355	7.1706	7521.3839	1950.3646	0.0609	0.0023	9.4936	1.1015	0.1425	0.1377	5.7855	0.0728
		quic	0.0000	0.0000	28.6013	0.5490	508.5818	67.4560	0.0593	0.0012	13.0512	1.2932	0.1854	0.1274	1.0108	0.0777
	1	emp	0.0000	0.0000	123.3807	2.6563	2.1544	2.9207	0.0909	0.0017	23.5889	1.8636	0.1762	0.1211	0.0052	0.0005
		diag	0.9900	0.0000	127.1994	3.0593	22982.4486	1357.8739	0.0507	0.0002	5.2313	0.6812	0.0289	0.0669	0.0002	0.0004
		oas	0.0000	0.0000	94.5772	2.7102	13676.1066	977.9414	0.0433	0.0003	1.3650	0.8262	0.1762	0.1211	0.0008	0.0004
		ss	0.0000	0.0000	94.4786	2.7007	13574.9135	991.4959	0.0432	0.0003	1.7169	0.8336	0.1886	0.1238	0.0136	0.0076
		ppca	0.0000	0.0000	118.9680	4.9463	19781.9820	2160.9221	0.0603	0.0054	23.5889	1.8636	0.1762	0.1211	0.1332	0.0015
		spc	0.9323	0.0428	90.6591	2.2275	7172.6122	777.9113	0.0580	0.0022	8.6448	1.7643	0.1041	0.1267	4.5352	0.0582
		quic	0.0000	0.0000	48.6410	0.6354	511.0577	93.7888	0.0852	0.0017	22.5789	1.8542	0.1751	0.1198	1.2580	0.1151
	2	emp	0.0000	0.0000	Inf	NA	5.5489	6.9171	0.1298	0.0035	40.2818	3.4064	0.1451	0.1014	0.0041	0.0011
		diag	0.9900	0.0000	128.9976	4.3730	23316.8547	1967.0825	0.0524	0.0005	3.5858	1.1895	0.0268	0.0720	0.0002	0.0004
		oas	0.0000	0.0000	107.0803	4.0036	17103.0096	1675.6609	0.0461	0.0003	1.7844	1.1659	0.1451	0.1014	0.0005	0.0005
		ss	0.0000	0.0000	106.3545	4.5676	16647.0410	1919.2997	0.0461	0.0003	2.4102	1.1359	0.1415	0.1061	0.0078	0.0009
		ppca	0.0000	0.0000	120.3357	3.3678	20082.7689	1632.5651	0.0688	0.0060	40.2818	3.4064	0.1451	0.1014	0.0474	0.0084
		spc	0.9853	0.0037	71.5255	1.9464	712.7910	278.2604	0.0687	0.0010	5.0886	2.6424	0.0676	0.0957	1.7649	0.0367
		quic	0.0000	0.0000	88.1624	2.0243	371.9590	131.4570	0.1229	0.0031	39.0459	3.3402	0.1450	0.0994	2.5659	0.2837
	10	emp	0.0000	0.0000	Inf	NA	19.3262	27.0722	0.3024	0.0180	147.7288	16.5247	0.1200	0.0872	0.0033	0.0005
		diag	0.9900	0.0000	140.1204	10.7399	23952.2873	4925.0302	0.0651	0.0022	5.7284	3.1985	0.0494	0.0794	0.0001	0.0003
		oas	0.0000	0.0000	114.1258	6.6821	18431.8509	3191.6372	0.0561	0.0040	9.2966	5.2071	0.1200	0.0872	0.0003	0.0005
		ss	0.0000	0.0000	106.2841	7.1519	14553.5475	3233.4907	0.0578	0.0036	8.8790	4.5488	0.1230	0.0840	0.0036	0.0005
		ppca	0.0000	0.0000	115.1053	6.3383	15600.3566	2881.2143	0.1621	0.0153	147.7288	16.5247	0.1200	0.0872	0.0031	0.0003
		spc	0.9895	0.0003	108.8799	6.2155	3044.6827	640.2270	0.0932	0.0015	11.7594	5.7824	0.0824	0.1052	0.1125	0.0289
		quic	0.0000	0.0000	206.3005	13.5148	275.0561	286.1633	0.2887	0.0181	143.2306	16.6132	0.1205	0.0869	7.0850	1.7115

ANNEX B – Results for the Synthetic Classification Datasets

Comparison of seven Gaussian Bayes classifiers on artificially generated data. The results correspond to average performance measures on test data from a sample of 30 pseudo-replicate results, obtained via 30 runs of hold-out cross-validation. At each run, 300 cases have been randomly selected for training and the hold-out cases were then used as test data.

Dataset	Estimator	Accuracy (Mean)	Accuracy (Std Dev.)	AUC (Mean)	AUC (Std Dev.)	Log-Loss (Mean)	Log-Loss (Std Dev.)	Time (Mean)	Time (Std Dev.)
twonorm	diag	0.9761	0.0012	0.9976	0.0001	0.0623	0.0019	0.0046	0.0010
	emp	0.9629	0.0038	0.9942	0.0009	0.1068	0.0113	0.0044	0.0007
	oas	**0.9772**	0.0008	**0.9978**	0.0001	**0.0597**	0.0009	**0.0041**	0.0010
	ss	0.9771	0.0010	0.9978	0.0001	0.0598	0.0011	0.0129	0.0015
	ppca	0.9761	0.0011	0.9977	0.0001	0.0617	0.0014	0.0125	0.0009
	spc	0.9767	0.0009	**0.9978**	0.0001	0.0605	0.0015	0.5946	0.1718
	quic	0.9707	0.0030	0.9964	0.0005	0.0786	0.0085	0.0102	0.0021
threenorm	diag	0.8327	0.0069	0.9144	0.0061	0.3733	0.0122	0.0183	0.0696
	emp	0.8145	0.0095	0.8958	0.0100	0.4732	0.0293	0.0043	0.0008
	oas	0.8622	0.0059	0.9385	0.0045	0.3181	0.0125	**0.0036**	0.0006
	ss	0.8613	0.0072	0.9377	0.0049	0.3205	0.0137	0.0110	0.0008
	ppca	**0.8799**	0.0028	**0.9536**	0.0017	**0.2813**	0.0039	0.0124	0.0013
	spc	0.8697	0.0043	0.9455	0.0028	0.3013	0.0071	0.5305	0.1796
	quic	0.8459	0.0163	0.9268	0.0116	0.3593	0.0415	0.0124	0.0027
ringnorm	diag	0.9860	0.0007	0.9987	0.0001	0.0387	0.0012	0.0046	0.0009
	emp	0.9688	0.0040	0.9972	0.0003	0.0921	0.0137	**0.0044**	0.0011
	oas	**0.9865**	0.0004	**0.9988**	0.0000	**0.0370**	0.0005	**0.0044**	0.0030
	ss	0.9864	0.0005	**0.9988**	0.0001	0.0371	0.0007	0.0142	0.0059
	ppca	0.9857	0.0007	0.9987	0.0001	0.0389	0.0009	0.0156	0.0153
	spc	0.9853	0.0010	**0.9988**	0.0001	0.0403	0.0025	0.6021	0.1953
	quic	0.9815	0.0024	0.9982	0.0003	0.0518	0.0072	0.0098	0.0014
waveform	diag	0.8054	0.0064	0.9533	0.0064	0.8103	0.0957	0.0067	0.0013
	emp	0.7811	0.0099	0.9377	0.0049	0.6744	0.0459	0.0059	0.0005
	oas	0.7995	0.0068	0.9473	0.0033	0.4986	0.0264	**0.0054**	0.0007
	ss	0.7974	0.0075	0.9456	0.0036	0.5056	0.0258	0.0169	0.0031
	ppca	0.8015	0.0060	0.9482	0.0027	0.4841	0.0281	0.0196	0.0037
	spc	0.8015	0.0060	0.9482	0.0027	0.4841	0.0281	0.0758	0.0237
	quic	**0.8202**	0.0097	**0.9575**	0.0043	**0.3842**	0.0375	0.0491	0.0045
waveform noise	diag	**0.7966**	0.0077	**0.9488**	0.0048	0.8092	0.0875	0.0096	0.0006
	emp	0.6911	0.0135	0.8805	0.0084	2.1906	0.2012	0.0101	0.0014
	oas	0.7589	0.0094	0.9214	0.0051	0.7022	0.0494	**0.0075**	0.0016
	ss	0.7576	0.0086	0.9201	0.0051	0.7507	0.0455	0.0260	0.0007
	ppca	0.7872	0.0136	0.9383	0.0071	0.4715	0.0506	0.0529	0.0008
	spc	0.7888	0.0137	0.9396	0.0072	**0.4603**	0.0492	2.3477	0.1931
	quic	0.7784	0.0225	0.9368	0.0112	0.5939	0.1767	0.2262	0.0235
corelearn	diag	0.7700	0.0093	0.8356	0.0071	0.7023	0.0564	0.0042	0.0010
	emp	0.7685	0.0083	0.8331	0.0073	0.8147	0.0677	0.0040	0.0009
	oas	0.7540	0.0054	0.8199	0.0055	**0.5813**	0.0241	**0.0035**	0.0006
	ss	0.7655	0.0070	0.8317	0.0064	0.6268	0.0411	0.0115	0.0016
	ppca	0.7532	0.0180	0.8180	0.0169	0.8082	0.0675	0.0095	0.0007
	spc	0.7531	0.0172	0.8183	0.0168	0.8111	0.0622	0.6451	0.1277
	quic	**0.7711**	0.0080	**0.8366**	0.0067	0.6863	0.0554	0.0084	0.0012
xor	diag	0.2194	0.0378	0.5123	0.0084	2.1169	0.0206	0.0080	0.0007
	emp	0.8600	0.0132	0.9876	0.0023	0.5178	0.0240	0.0083	0.0014
	oas	0.8766	0.0122	0.9882	0.0015	0.6135	0.0195	**0.0076**	0.0010
	ss	0.8776	0.0129	0.9897	0.0016	0.5242	0.0198	0.0227	0.0029
	ppca	**0.8951**	0.0141	**0.9916**	0.0017	**0.4854**	0.0209	0.0118	0.0028
	spc	0.7517	0.0179	0.9668	0.0049	0.7308	0.0404	0.1745	0.0208
	quic	0.8597	0.0125	0.9851	0.0019	0.7033	0.0469	0.0167	0.0024

ANNEX C – Description of the Real World Classification Datasets

Description of the 33 real world problems applied in the classification study of Chapter 4. These datasets are all publicly available as part of the UCI Machine Learning Repository (LICHMAN, 2013), and all of them are valid for classification tasks. For each dataset, the name, number of attributes (original and after transformation of the categorical variables into dummy variables), number of examples (without missing values), number of classes, and within-class p/N ratio (minimum, median and maximum) are given.

Dataset	# Attributes	# Observations	# Classes	Min Ratio (p/N)	Med Ratio (p/N)	Max Ratio (p/N)
australian	14 - 34	690	2	0.0888	0.0998	0.1107
balance	4	625	3	0.0139	0.0139	0.0816
bands	19	365	2	0.0826	0.1117	0.1407
breast	9	277	2	0.0203	0.0290	0.0377
bupa	6	345	2	0.0300	0.0357	0.0414
car	6 - 15	1728	4	0.0124	0.1282	0.2308
chess	36 - 37	3196	2	0.0222	0.0232	0.0242
ecoli	7	336	8	0.0420	0.1154	0.3000
german	20 - 24	1000	2	0.0343	0.0571	0.0800
heart	13 - 17	270	2	0.1133	0.1275	0.1417
ionosphere	33	351	2	0.1422	0.1981	0.2540
iris	4	150	3	0.0800	0.0800	0.0800
pageblocks	10	5472	5	0.0020	0.0870	0.3571
phoneme	5	5404	2	0.0013	0.0022	0.0032
pima	8	768	2	0.0160	0.0229	0.0299
segmentation	19	2310	7	0.0485	0.0485	0.0485
soldat	72	4844	2	0.0241	0.0315	0.0388
sonar	60	208	2	0.5405	0.5795	0.6186
soybean	35 - 62	683	15	0.6739	3.1000	3.1000
spambase	57	4597	2	0.0204	0.0259	0.0314
thyroid	21	7200	3	0.0011	0.0190	0.0422
titanic	3 - 5	2201	2	0.0034	0.0052	0.0070
vehicle	18	846	4	0.0826	0.0839	0.0905
vowel	13	990	11	0.1000	0.1000	0.1000
wine	13	178	3	0.1831	0.2203	0.2708
adult	14 - 95	45222	2	0.0028	0.0056	0.0085
dna	180	3186	3	0.1088	0.2347	0.2353
letter	16	20000	26	0.0197	0.0209	0.0218
optdigits	64	5620	10	0.1084	0.1107	0.1119
penbased	16	10992	10	0.0140	0.0146	0.0152
satellite	36	6435	7	0.0235	0.0387	0.0575
shuttle	9	58000	7	0.0002	0.0526	0.9000
texture	40	5500	11	0.0800	0.0800	0.0800

ANNEX D – Parameter specification for the classification algorithms

Parameter specification for the classification algorithms employed in the experimentation of Chapter 4.

Algorithm	Parameter	Description	Value
GaussNaiveBayes *R package*: e1071 *Function*: naiveBayes	none	none	none
KernelNaiveBayes *R package*: klaR Function: NaiveBayes	none	none	none
LinearDiscriminant *R package*: sda *Function*: sda	lambda	Shrinkage intensity for the correlation matrix.	Estimated from the data (SCHÄFER; STRIMMER, 2005).
	lambda.var	Shrinkage intensity for the variances.	Estimated from the data (OPGEN-RHEIN; STRIMMER, 2007).
	lambda.freqs	Shrinkage intensity for the frequencies.	Estimated from the data (HAUSSER; STRIMMER, 2009).
GlmNet R package: glmnet Function: cv.glmnet	lambda	Regularization parameter.	Defined via 5-fold cross-valiation.
Cart R package: rpart Function: rpart	split	Split criteria.	"gini"
	cp	Complexity parameter to which the tree will be pruned.	Defined via 5-fold cross-valiation.
C4.5 R package: RWeka Function: J48	C	Confidence threshold for pruning the tree.	0.25
LinearSvm R package: e1071 Function: svm	kernel	The kernel function to be used.	"linear"
	cost	Cost of constraints violation.	Defined via 5-fold cross-valiation.
NonlinearRbfSvm R package: e1071 Function: svm	kernel	The kernel function to be used.	"radial"
	cost	Cost of constraints violation.	Defined via 5-fold cross-valiation.
	gamma	Inverse kernel width for the Radial Basis kernel function.	Estimated from the data (CAPUTO et al., 2002).
ExtraTrees R package: extraTrees Function: extraTrees	ntree	The number of trees.	150
Adaboost R package: RWeka Function: AdaBoostM1	W	Base classifier.	"J48" (C4.5 Decision Tree).
	I	Number of iterations.	50
GradientBoosting R package: gbm Function: gbm	n.trees	The total number of trees to fit.	150
	shrinkage	Learning rate or step-size reduction.	5e-2

Algorithm	Parameter	Description	Value
	bag.fraction	The fraction of the training set observations randomly selected to fit the tree in the next iteration.	0.5
	interaction.depth	The maximum depth of variable interactions.	5
	n.minobsinnode	Minimum number of observations in the trees terminal nodes.	10
RandomForest R package: randomForest Function: randomForest	ntree	Number of trees to grow.	150
	mtry	Number of variables randomly sampled as candidates at each split.	sqrt(p). The square root of the total number of variables.

ANNEX E – Results for the Real World Classification Datasets

Comparison of 19 classifiers on 33 real world datasets. The results correspond to average performance measures on test data from a sample of 30 pseudo-replicate results, obtained via 6 runs of 5-fold cross-validation.

Dataset	Algorithm	Accuracy (Mean)	Accuracy (Std Dev.)	AUC (Mean)	AUC (Std Dev.)	Log-Loss (Mean)	Log-Loss (Std Dev.)	Time (Mean)	Time (Std Dev.)
australian	GaussNaiveBayes	0.7290	0.0379	0.8185	0.0319	3.6459	1.1724	0.0146	0.0008
	KernelNaiveBayes	0.6952	0.0459	0.7964	0.0293	2.5168	0.6427	0.0406	0.0025
	LinearDiscriminant	0.8522	0.0296	0.9215	0.0159	0.3570	0.0413	0.0253	0.0077
	GlmNet	0.8549	0.0273	0.9142	0.0284	0.3855	0.0780	0.2479	0.0428
	Cart	0.8522	0.0269	0.8962	0.0246	0.3971	0.0688	0.0188	0.0013
	C4.5DecisionTree	0.8461	0.0269	0.8578	0.0440	1.5678	0.8055	0.0613	0.0392
	LinearSvm	0.8522	0.0318	0.9215	0.0215	0.3709	0.0568	3.2460	0.4957
	RegGaussBayes-diag	0.7747	0.0273	0.8422	0.0308	1.8601	0.5601	0.0022	0.0005
	RegGaussBayes-lowran	0.8065	0.0299	0.8598	0.0350	0.9961	0.4226	0.0114	0.0030
	RegGaussBayes	0.7662	0.0292	0.8390	0.0290	1.8940	0.5400	0.0019	0.0003
	BoostedRgb-diag	0.8282	0.0278	0.9039	0.0243	0.4585	0.0884	0.5683	0.0646
	BoostedRgb-lowrank	0.8452	0.0278	0.9100	0.0237	0.4844	0.0199	0.7895	0.0975
	BoostedRgb	0.8241	0.0271	0.8860	0.0287	0.4730	0.0257	0.5393	0.0664
	NonlinearRbfSvm	0.8406	0.0294	0.9154	0.0231	0.3705	0.0521	2.1152	0.0688
	ExtraTrees	0.8519	0.0278	0.9116	0.0178	0.5196	0.1910	0.0340	0.0150
	Adaboost	0.8628	0.0288	0.9152	0.0188	3.4961	0.7840	0.6862	0.0540
	GradientBoosting	0.8648	0.0249	0.9339	0.0158	0.3402	0.0549	0.3290	0.0114
	RandomForest	0.8708	0.0216	0.9351	0.0150	0.3649	0.0896	0.1460	0.0037
balance	GaussNaiveBayes	0.9045	0.0100	0.9570	0.0054	0.4766	0.0162	0.0023	0.0005
	KernelNaiveBayes	0.9125	0.0089	0.9468	0.0055	0.5082	0.0231	0.0076	0.0006
	LinearDiscriminant	0.8696	0.0110	0.9764	0.0043	0.3338	0.0166	0.0066	0.0012
	GlmNet	0.8883	0.0214	0.9765	0.0111	0.2927	0.0869	0.2172	0.0035
	Cart	0.7806	0.0304	0.8670	0.0238	1.0408	0.3523	0.0055	0.0006
	C4.5DecisionTree	0.7843	0.0251	0.8442	0.0245	4.0970	0.9459	0.0296	0.0007
	LinearSvm	0.9168	0.0159	0.9845	0.0040	0.2430	0.0223	0.3227	0.0232
	RegGaussBayes-diag	0.9032	0.0118	0.9572	0.0054	0.4781	0.0156	0.0021	0.0003
	RegGaussBayes-lowran	0.8992	0.0178	0.9609	0.0067	0.4401	0.0344	0.0055	0.0017
	RegGaussBayes	0.9069	0.0107	0.9776	0.0035	0.3483	0.0170	0.0020	0.0000
	BoostedRgb-diag	0.9163	0.0065	0.9359	0.0050	0.7228	0.0235	0.2598	0.0449
	BoostedRgb-lowrank	0.9179	0.0089	0.9760	0.0042	0.6907	0.0158	0.6130	0.0674
	BoostedRgb	0.9440	0.0171	0.9901	0.0031	0.5448	0.0210	0.2373	0.0510
	NonlinearRbfSvm	0.9680	0.0153	0.9981	0.0012	0.1094	0.0181	0.6068	0.0071
	ExtraTrees	0.7904	0.0204	0.9191	0.0074	2.7241	0.1938	0.0157	0.0021
	Adaboost	0.7667	0.0258	0.9348	0.0116	5.2620	0.7964	0.1618	0.0094
	GradientBoosting	0.9107	0.0117	0.9801	0.0038	0.2917	0.0279	0.0996	0.0016
	RandomForest	0.8493	0.0218	0.9468	0.0074	0.5486	0.1545	0.0393	0.0014
bands	GaussNaiveBayes	0.4032	0.0193	0.4635	0.0318	3.4264	0.6844	0.0076	0.0005
	KernelNaiveBayes	0.4973	0.0607	0.5484	0.0638	2.0198	0.6305	0.0219	0.0005
	LinearDiscriminant	0.6726	0.0528	0.7285	0.0487	0.6201	0.0524	0.0110	0.0019
	GlmNet	0.6712	0.0372	0.7223	0.0414	0.6186	0.0369	0.1674	0.0136
	Cart	0.6219	0.0537	0.6568	0.0517	0.8475	0.3094	0.0111	0.0007
	C4.5DecisionTree	0.6324	0.0591	0.6377	0.0622	6.7502	2.3559	0.0389	0.0021
	LinearSvm	0.6772	0.0382	0.7187	0.0363	0.6222	0.0243	1.6346	0.2613
	RegGaussBayes-diag	0.6384	0.0527	0.6872	0.0550	0.9824	0.2958	0.0017	0.0005
	RegGaussBayes-lowran	0.6717	0.0447	0.7102	0.0566	0.8734	0.2520	0.0082	0.0028
	RegGaussBayes	0.6753	0.0378	0.7334	0.0403	0.9533	0.2718	0.0017	0.0005
	BoostedRgb-diag	0.6658	0.0486	0.7253	0.0495	0.6295	0.0856	0.4169	0.0577
	BoostedRgb-lowrank	0.6680	0.0389	0.7236	0.0400	0.6184	0.0173	0.4908	0.0590
	BoostedRgb	0.6767	0.0454	0.7440	0.0490	0.5989	0.0290	0.3729	0.0491
	NonlinearRbfSvm	0.7100	0.0389	0.7620	0.0408	0.5829	0.0341	0.7185	0.0249
	ExtraTrees	0.7598	0.0394	0.8355	0.0310	0.5119	0.0291	0.0183	0.0025
	Adaboost	0.7315	0.0395	0.8041	0.0449	5.6901	1.2557	0.2928	0.0103
	GradientBoosting	0.7000	0.0463	0.7875	0.0442	0.6104	0.0902	0.1233	0.0022
	RandomForest	0.7489	0.0400	0.8180	0.0398	0.5355	0.0352	0.0539	0.0010
breast	GaussNaiveBayes	0.9624	0.0166	0.9669	0.0164	0.8458	0.4042	0.0044	0.0005
	KernelNaiveBayes	0.9642	0.0139	0.9910	0.0101	0.3259	0.2212	0.0113	0.0005
	LinearDiscriminant	0.9600	0.0170	0.9952	0.0028	0.1618	0.0781	0.0101	0.0021
	GlmNet	0.9641	0.0173	0.9942	0.0045	0.0940	0.0325	0.1820	0.0041
	Cart	0.9427	0.0166	0.9585	0.0157	0.2280	0.0888	0.0070	0.0007
	C4.5DecisionTree	0.9495	0.0199	0.9622	0.0214	0.6979	0.4685	0.0317	0.0011

Dataset	Algorithm	Accuracy (Mean)	Accuracy (Std Dev.)	AUC (Mean)	AUC (Std Dev.)	Log-Loss (Mean)	Log-Loss (Std Dev.)	Time (Mean)	Time (Std Dev.)
	LinearSvm	0.9702	0.0133	0.9949	0.0043	0.0873	0.0315	0.3588	0.0319
	RegGaussBayes-diag	0.9627	0.0161	0.9670	0.0164	0.8417	0.4012	0.0017	0.0005
	RegGaussBayes-lowran	0.9520	0.0202	0.9676	0.0167	0.7752	0.3740	0.0055	0.0017
	RegGaussBayes	0.9532	0.0184	0.9654	0.0169	0.8074	0.3845	0.0014	0.0005
	BoostedRgb-diag	0.9634	0.0160	0.9864	0.0102	0.2219	0.0393	0.4207	0.0603
	BoostedRgb-lowrank	0.9644	0.0174	0.9887	0.0071	0.2901	0.0187	0.4734	0.0642
	BoostedRgb	0.9561	0.0189	0.9858	0.0091	0.2820	0.0173	0.4052	0.0649
	NonlinearRbfSvm	0.9693	0.0154	0.9932	0.0052	0.0961	0.0350	0.5180	0.0236
	ExtraTrees	0.9715	0.0129	0.9960	0.0027	0.0813	0.0247	0.0106	0.0019
	Adaboost	0.9688	0.0155	0.9858	0.0080	0.9149	0.4850	0.1506	0.0095
	GradientBoosting	0.9676	0.0157	0.9944	0.0037	0.1248	0.0701	0.1188	0.0025
	RandomForest	0.9727	0.0132	0.9948	0.0040	0.0922	0.0316	0.0386	0.0008
	GaussNaiveBayes	0.5565	0.0427	0.6044	0.0590	0.8083	0.1926	0.0029	0.0003
	KernelNaiveBayes	0.6575	0.0464	0.7043	0.0533	0.7172	0.1169	0.0075	0.0005
	LinearDiscriminant	0.6662	0.0535	0.7120	0.0557	0.6232	0.0311	0.0063	0.0012
	GlmNet	0.6807	0.0563	0.7245	0.0601	0.6224	0.0448	0.1544	0.0022
	Cart	0.6546	0.0664	0.6689	0.0746	0.7179	0.2045	0.0072	0.0005
	C4.5DecisionTree	0.6671	0.0595	0.6707	0.0641	3.5972	1.8561	0.0301	0.0017
	LinearSvm	0.6870	0.0504	0.7290	0.0557	0.6229	0.0378	0.4139	0.0340
	RegGaussBayes-diag	0.5546	0.0501	0.6010	0.0591	0.7968	0.1763	0.0017	0.0005
bupa	RegGaussBayes-lowran	0.6048	0.0449	0.6481	0.0514	0.7886	0.1836	0.0056	0.0026
	RegGaussBayes	0.5870	0.0435	0.6588	0.0526	0.7331	0.1399	0.0012	0.0004
	BoostedRgb-diag	0.6527	0.0481	0.6948	0.0527	0.6548	0.0153	0.2316	0.0597
	BoostedRgb-lowrank	0.6662	0.0482	0.7164	0.0559	0.6399	0.0214	0.3726	0.0726
	BoostedRgb	0.6700	0.0493	0.7235	0.0515	0.6362	0.0202	0.2341	0.0624
	NonlinearRbfSvm	0.7024	0.0562	0.7590	0.0569	0.5966	0.0471	0.4338	0.0200
	ExtraTrees	0.7164	0.0523	0.7831	0.0574	0.5668	0.0492	0.0125	0.0023
	Adaboost	0.7024	0.0556	0.7490	0.0584	2.8526	2.4555	0.0610	0.0337
	GradientBoosting	0.7145	0.0482	0.7827	0.0540	0.6355	0.1202	0.0535	0.0013
	RandomForest	0.7367	0.0530	0.7892	0.0570	0.5668	0.0550	0.0281	0.0010
	GaussNaiveBayes	0.1009	0.0124	0.3917	0.0105	12.5553	0.4467	0.0094	0.0005
	KernelNaiveBayes	0.7002	0.0013	0.9396	0.0028	1.4158	0.1039	0.0349	0.0011
	LinearDiscriminant	0.8922	0.0163	0.9871	0.0026	0.3107	0.0159	0.0267	0.0053
	GlmNet	0.9322	0.0103	0.9955	0.0010	0.1547	0.0218	1.5478	0.1382
	Cart	0.8636	0.0168	0.9778	0.0041	0.4066	0.0896	0.0158	0.0008
	C4.5DecisionTree	0.9339	0.0156	0.9805	0.0073	0.9455	0.3470	0.0438	0.0023
	LinearSvm	0.9350	0.0112	0.9948	0.0015	0.1673	0.0195	3.1298	0.1079
	RegGaussBayes-diag	0.6811	0.0191	0.9249	0.0073	0.6413	0.0409	0.0033	0.0005
car	RegGaussBayes-lowran	0.8264	0.0195	0.9654	0.0051	0.4456	0.0286	0.0105	0.0018
	RegGaussBayes	0.9010	0.0138	0.9909	0.0017	0.2247	0.0248	0.0031	0.0003
	BoostedRgb-diag	0.8736	0.0189	0.9603	0.0032	0.5333	0.0207	1.2928	0.0782
	BoostedRgb-lowrank	0.9086	0.0168	0.9784	0.0023	0.6657	0.0152	1.5664	0.0962
	BoostedRgb	0.9806	0.0082	0.9989	0.0005	0.3547	0.0543	1.1993	0.0760
	NonlinearRbfSvm	0.9930	0.0041	0.9999	0.0001	0.0295	0.0061	5.9611	0.0753
	ExtraTrees	0.9152	0.0142	0.9870	0.0030	0.5082	0.1194	0.0299	0.0033
	Adaboost	0.9605	0.0132	0.9964	0.0019	0.8191	0.3409	0.6105	0.0164
	GradientBoosting	0.9557	0.0092	0.9973	0.0008	0.1620	0.0103	0.7095	0.0081
	RandomForest	0.8656	0.0126	0.9822	0.0019	0.3497	0.0110	0.1847	0.0298
	GaussNaiveBayes	0.6174	0.0129	0.7619	0.0165	5.1773	1.0950	0.0260	0.0012
	KernelNaiveBayes	0.5223	0.0007	0.7444	0.0038	4.9966	0.9231	0.0514	0.0007
	LinearDiscriminant	0.9383	0.0103	0.9840	0.0036	0.1838	0.0138	0.0979	0.0325
	GlmNet	0.9727	0.0082	0.9955	0.0018	0.0857	0.0202	0.8698	0.0516
	Cart	0.9670	0.0103	0.9827	0.0068	0.1258	0.0283	0.0466	0.0009
chess	C4.5DecisionTree	0.9926	0.0028	0.9975	0.0018	0.0691	0.0475	0.0826	0.0033
	LinearSvm	0.9721	0.0077	0.9945	0.0023	0.0951	0.0225	12.8221	0.9631
	RegGaussBayes-diag	0.6633	0.0165	0.8270	0.0108	2.0374	0.3035	0.0049	0.0004
	RegGaussBayes-lowran	0.7299	0.0204	0.8385	0.0185	0.9318	0.1333	0.0168	0.0016
	RegGaussBayes	0.7775	0.0201	0.9049	0.0141	1.2195	0.2314	0.0049	0.0005
	BoostedRgb-diag	0.9422	0.0134	0.9882	0.0041	0.2884	0.0138	1.2623	0.1090
	BoostedRgb-lowrank	0.9428	0.0119	0.9869	0.0042	0.4397	0.0064	1.6894	0.1391

Dataset	Algorithm	**Accuracy** (Mean)	**Accuracy** (Std Dev.)	**AUC** (Mean)	**AUC** (Std Dev.)	**Log-Loss** (Mean)	**Log-Loss** (Std Dev.)	**Time** (Mean)	**Time** (Std Dev.)
	BoostedRgb	0.9928	0.0032	0.9993	0.0007	0.1750	0.0108	1.2589	0.0804
	NonlinearRbfSvm	0.9926	0.0028	0.9997	0.0003	0.0208	0.0063	23.3639	0.4086
	ExtraTrees	0.9957	0.0017	0.9993	0.0011	0.0350	0.0309	0.0625	0.0056
	Adaboost	0.9958	0.0026	0.9986	0.0011	0.1114	0.0758	2.4090	0.1420
	GradientBoosting	0.9880	0.0039	0.9993	0.0004	0.0514	0.0052	1.4843	0.0109
	RandomForest	0.9855	0.0049	0.9987	0.0008	0.0965	0.0063	0.7502	0.0799
ecoli	GaussNaiveBayes	0.6541	0.0678	0.8803	0.0204	1.8660	0.4294	0.0037	0.0005
	KernelNaiveBayes	0.8315	0.0430	0.9632	0.0117	0.7891	0.2920	0.0169	0.0005
	LinearDiscriminant	0.7704	0.0475	0.9373	0.0183	0.7381	0.1495	0.0064	0.0010
	GlmNet	0.7786	0.0496	0.9320	0.0241	0.7394	0.1343	0.3156	0.0685
	Cart	0.7941	0.0361	0.9290	0.0203	1.2305	0.6391	0.0066	0.0008
	C4.5DecisionTree	0.8028	0.0487	0.9056	0.0367	3.4652	1.6156	0.0308	0.0006
	LinearSvm	0.7867	0.0477	0.9331	0.0197	0.7454	0.0857	0.3401	0.0160
	RegGaussBayes-diag	0.7846	0.0393	0.9405	0.0190	0.9879	0.3864	0.0033	0.0005
	RegGaussBayes-lowran	0.7947	0.0343	0.9406	0.0197	1.0775	0.4176	0.0075	0.0023
	RegGaussBayes	0.8024	0.0382	0.9443	0.0197	0.9555	0.3998	0.0026	0.0005
	BoostedRgb-diag	0.8044	0.0355	0.9415	0.0162	1.0813	0.3154	0.5752	0.0535
	BoostedRgb-lowrank	0.8119	0.0476	0.9458	0.0194	0.9739	0.0388	0.8938	0.0951
	BoostedRgb	0.8260	0.0473	0.9492	0.0190	0.9029	0.1174	0.5389	0.0520
	NonlinearRbfSvm	0.8359	0.0400	0.9647	0.0128	0.5193	0.0729	0.3771	0.0055
	ExtraTrees	0.8533	0.0375	0.9714	0.0120	0.5454	0.2497	0.0105	0.0018
	Adaboost	0.8350	0.0387	0.9569	0.0169	4.4160	1.2814	0.1290	0.0082
	GradientBoosting	0.8461	0.0394	0.9662	0.0114	0.5693	0.1518	0.1206	0.0022
	RandomForest	0.8528	0.0356	0.9710	0.0119	0.5627	0.2992	0.0252	0.0007
german	GaussNaiveBayes	0.7113	0.0741	0.7683	0.0709	0.9851	0.4834	0.0114	0.0005
	KernelNaiveBayes	0.7200	0.0241	0.8169	0.0297	0.6800	0.0964	0.0288	0.0005
	LinearDiscriminant	0.7633	0.0294	0.8404	0.0219	0.4966	0.0328	0.0249	0.0055
	GlmNet	0.7652	0.0295	0.8402	0.0218	0.4968	0.0312	0.2246	0.0101
	Cart	0.7430	0.0245	0.8019	0.0211	0.5779	0.1203	0.0241	0.0011
	C4.5DecisionTree	0.7227	0.0339	0.7226	0.0406	3.7423	0.9475	0.0519	0.0021
	LinearSvm	0.7638	0.0285	0.8397	0.0207	0.4963	0.0251	6.9104	0.6713
	RegGaussBayes-diag	0.7292	0.0242	0.7890	0.0249	0.8225	0.1953	0.0021	0.0003
	RegGaussBayes-lowran	0.7273	0.0267	0.7985	0.0263	0.6806	0.0815	0.0091	0.0022
	RegGaussBayes	0.7390	0.0234	0.8019	0.0254	0.8210	0.1916	0.0018	0.0004
	BoostedRgb-diag	0.7453	0.0293	0.8257	0.0227	0.5737	0.1819	0.5435	0.0627
	BoostedRgb-lowrank	0.7525	0.0260	0.8262	0.0219	0.5587	0.0122	0.7364	0.0836
	BoostedRgb	0.7217	0.0265	0.7997	0.0199	0.5629	0.0136	0.5429	0.0537
	NonlinearRbfSvm	0.7595	0.0301	0.8351	0.0197	0.5034	0.0257	3.6384	0.0703
	ExtraTrees	0.7578	0.0209	0.8347	0.0197	0.5094	0.0416	0.0375	0.0038
	Adaboost	0.7483	0.0220	0.8160	0.0216	5.6784	0.7639	0.8601	0.0146
	GradientBoosting	0.7698	0.0207	0.8427	0.0219	0.5070	0.0441	0.3653	0.0072
	RandomForest	0.7650	0.0211	0.8449	0.0192	0.4929	0.0228	0.1970	0.0031
heart	GaussNaiveBayes	0.7765	0.0842	0.8475	0.0744	1.3258	0.8642	0.0071	0.0004
	KernelNaiveBayes	0.8309	0.0514	0.8969	0.0450	0.5857	0.2529	0.0196	0.0007
	LinearDiscriminant	0.8426	0.0394	0.9083	0.0350	0.4034	0.0972	0.0107	0.0046
	GlmNet	0.8377	0.0437	0.9024	0.0352	0.4082	0.0689	0.1760	0.0031
	Cart	0.8025	0.0576	0.8238	0.0635	0.5053	0.1036	0.0086	0.0007
	C4.5DecisionTree	0.7790	0.0737	0.7918	0.0753	3.3782	1.6900	0.0358	0.0015
	LinearSvm	0.8247	0.0519	0.9045	0.0326	0.3920	0.0643	0.5995	0.0924
	RegGaussBayes-diag	0.8191	0.0556	0.8753	0.0554	0.9661	0.4879	0.0019	0.0003
	RegGaussBayes-lowran	0.8037	0.0485	0.8666	0.0507	0.7186	0.2994	0.0068	0.0023
	RegGaussBayes	0.8031	0.0495	0.8573	0.0583	0.9477	0.4708	0.0013	0.0005
	BoostedRgb-diag	0.8216	0.0499	0.8799	0.0436	0.4810	0.0464	0.3742	0.0696
	BoostedRgb-lowrank	0.8025	0.0477	0.8775	0.0442	0.4985	0.0363	0.4683	0.0785
	BoostedRgb	0.7494	0.0556	0.8234	0.0495	0.5307	0.0410	0.3694	0.0814
	NonlinearRbfSvm	0.8160	0.0491	0.8926	0.0415	0.4149	0.0790	0.3994	0.0108
	ExtraTrees	0.8006	0.0431	0.8877	0.0430	0.4589	0.1682	0.0117	0.0024
	Adaboost	0.7864	0.0426	0.8606	0.0437	5.2892	1.4906	0.1473	0.0039
	GradientBoosting	0.8006	0.0397	0.8779	0.0363	0.5469	0.1327	0.0842	0.0016
	RandomForest	0.8185	0.0444	0.9007	0.0381	0.4220	0.1132	0.0328	0.0009

Dataset	Algorithm	Accuracy (Mean)	Accuracy (Std Dev.)	AUC (Mean)	AUC (Std Dev.)	Log-Loss (Mean)	Log-Loss (Std Dev.)	Time (Mean)	Time (Std Dev.)
ionosphere	GaussNaiveBayes	0.8172	0.0538	0.9180	0.0352	1.1625	0.5372	0.0123	0.0004
	KernelNaiveBayes	0.9112	0.0302	0.9649	0.0163	0.7280	0.2809	0.0367	0.0007
	LinearDiscriminant	0.8551	0.0415	0.9030	0.0367	0.4509	0.1414	0.0151	0.0036
	GlmNet	0.8490	0.0384	0.9055	0.0394	0.4176	0.1313	0.2250	0.0481
	Cart	0.8975	0.0331	0.9219	0.0271	0.3739	0.2218	0.0149	0.0007
	C4.5DecisionTree	0.9051	0.0330	0.9206	0.0358	1.4537	0.9412	0.0473	0.0029
	LinearSvm	0.8480	0.0365	0.8992	0.0369	0.4043	0.0799	1.2361	0.2888
	RegGaussBayes-diag	0.8148	0.0549	0.9153	0.0356	1.1898	0.5474	0.0024	0.0005
	RegGaussBayes-lowran	0.9003	0.0365	0.9473	0.0342	1.5234	0.8312	0.0097	0.0025
	RegGaussBayes	0.9145	0.0324	0.9674	0.0196	1.7235	0.7388	0.0015	0.0005
	BoostedRgb-diag	0.9331	0.0249	0.9700	0.0193	0.3921	0.0210	0.4252	0.0721
	BoostedRgb-lowrank	0.9535	0.0236	0.9800	0.0140	0.3243	0.0292	0.5601	0.0700
	BoostedRgb	0.9464	0.0278	0.9768	0.0164	0.2472	0.0635	0.4109	0.0598
	NonlinearRbfSvm	0.9478	0.0265	0.9847	0.0122	0.1482	0.0559	0.6255	0.0217
	ExtraTrees	0.9449	0.0250	0.9872	0.0099	0.1791	0.0323	0.0127	0.0018
	Adaboost	0.9397	0.0274	0.9685	0.0174	1.7672	0.7861	0.4673	0.0139
	GradientBoosting	0.9312	0.0250	0.9749	0.0122	0.2498	0.0865	0.1960	0.0041
	RandomForest	0.9345	0.0281	0.9811	0.0122	0.2046	0.0904	0.0540	0.0015
iris	GaussNaiveBayes	0.9522	0.0388	0.9952	0.0050	0.1360	0.1003	0.0024	0.0005
	KernelNaiveBayes	0.9589	0.0347	0.9941	0.0059	0.1664	0.1396	0.0072	0.0004
	LinearDiscriminant	0.9778	0.0237	0.9989	0.0016	0.0570	0.0386	0.0040	0.0006
	GlmNet	0.9533	0.0367	0.9964	0.0045	0.1546	0.0969	0.2182	0.0069
	Cart	0.9356	0.0471	0.9756	0.0261	0.6705	0.9745	0.0048	0.0006
	C4.5DecisionTree	0.9456	0.0475	0.9650	0.0312	1.2379	1.3280	0.0284	0.0018
	LinearSvm	0.9556	0.0449	0.9974	0.0043	0.1149	0.0464	0.1440	0.0024
	RegGaussBayes-diag	0.9467	0.0397	0.9941	0.0057	0.1437	0.0967	0.0024	0.0005
	RegGaussBayes-lowran	0.9644	0.0302	0.9981	0.0021	0.0966	0.0814	0.0057	0.0028
	RegGaussBayes	0.9711	0.0300	0.9988	0.0017	0.0634	0.0319	0.0017	0.0005
	BoostedRgb-diag	0.9600	0.0332	0.9961	0.0051	0.3109	0.0371	0.2301	0.0594
	BoostedRgb-lowrank	0.9667	0.0277	0.9957	0.0063	0.2820	0.0432	0.4797	0.0582
	BoostedRgb	0.9633	0.0253	0.9964	0.0041	0.2867	0.0328	0.1994	0.0516
	NonlinearRbfSvm	0.9622	0.0312	0.9980	0.0026	0.1086	0.0397	0.1786	0.0032
	ExtraTrees	0.9544	0.0386	0.9966	0.0039	0.1052	0.0546	0.0049	0.0007
	Adaboost	0.9389	0.0496	0.9746	0.0233	1.2566	1.1001	0.0295	0.0060
	GradientBoosting	0.9422	0.0446	0.9932	0.0079	0.2854	0.2900	0.0260	0.0009
	RandomForest	0.9444	0.0404	0.9957	0.0053	0.1192	0.0851	0.0078	0.0005
pageblocks	GaussNaiveBayes	0.8992	0.0280	0.9773	0.0052	0.8847	0.1422	0.0114	0.0006
	KernelNaiveBayes	0.9357	0.0059	0.9862	0.0026	0.6537	0.0939	0.0330	0.0007
	LinearDiscriminant	0.9461	0.0056	0.9915	0.0020	0.3510	0.0629	0.0636	0.0160
	GlmNet	0.9479	0.0169	0.9931	0.0065	0.1801	0.0659	16.6559	7.1161
	Cart	0.9670	0.0039	0.9925	0.0018	0.1791	0.0431	0.0681	0.0039
	C4.5DecisionTree	0.9702	0.0047	0.9894	0.0037	0.4086	0.1255	0.0844	0.0042
	LinearSvm	0.9646	0.0036	0.9966	0.0013	0.1349	0.0144	10.7514	0.5315
	RegGaussBayes-diag	0.9116	0.0088	0.9740	0.0043	0.9097	0.1276	0.0063	0.0005
	RegGaussBayes-lowran	0.9031	0.0068	0.9773	0.0039	1.0780	0.1390	0.0147	0.0021
	RegGaussBayes	0.9291	0.0057	0.9822	0.0033	0.8323	0.1217	0.0059	0.0003
	BoostedRgb-diag	0.9303	0.0069	0.9886	0.0033	0.4995	0.1156	3.2507	0.1340
	BoostedRgb-lowrank	0.9367	0.0081	0.9900	0.0020	0.4603	0.0796	3.6723	0.1516
	BoostedRgb	0.9402	0.0065	0.9906	0.0041	0.4425	0.1006	3.2488	0.1084
	NonlinearRbfSvm	0.9652	0.0060	0.9971	0.0009	0.1240	0.0158	16.6983	0.3071
	ExtraTrees	0.9723	0.0050	0.9960	0.0011	0.2364	0.0560	0.0918	0.0093
	Adaboost	0.9708	0.0042	0.9981	0.0005	0.2562	0.0487	1.8213	0.7652
	GradientBoosting	0.9730	0.0039	0.9985	0.0005	0.0886	0.0131	2.5217	0.0219
	RandomForest	0.9743	0.0045	0.9978	0.0008	0.1526	0.0416	0.8136	0.1473
phoneme	GaussNaiveBayes	0.7599	0.0138	0.8356	0.0102	0.6375	0.0375	0.0060	0.0007
	KernelNaiveBayes	0.7799	0.0129	0.8683	0.0088	0.5369	0.0296	0.0100	0.0006
	LinearDiscriminant	0.7569	0.0094	0.8537	0.0075	0.4748	0.0124	0.0328	0.0074
	GlmNet	0.7501	0.0084	0.8513	0.0074	0.4718	0.0108	0.4577	0.0304
	Cart	0.8157	0.0125	0.8680	0.0220	0.4513	0.0343	0.0376	0.0012
	C4.5DecisionTree	0.8621	0.0143	0.9075	0.0114	1.1572	0.2504	0.0632	0.0034

Dataset	Algorithm	Accuracy (Mean)	Accuracy (Std Dev.)	AUC (Mean)	AUC (Std Dev.)	Log-Loss (Mean)	Log-Loss (Std Dev.)	Time (Mean)	Time (Std Dev.)
	LinearSvm	0.7745	0.0086	0.8501	0.0069	0.4837	0.0104	34.9151	1.3047
	RegGaussBayes-diag	0.7618	0.0133	0.8362	0.0101	0.6326	0.0369	0.0029	0.0003
	RegGaussBayes-lowran	0.7895	0.0101	0.8475	0.0097	0.6341	0.0429	0.0083	0.0039
	RegGaussBayes	0.7850	0.0078	0.8558	0.0072	0.5986	0.0324	0.0027	0.0005
	BoostedRgb-diag	0.7694	0.0112	0.8431	0.0099	0.4878	0.0206	0.6219	0.0660
	BoostedRgb-lowrank	0.7811	0.0132	0.8624	0.0096	0.5047	0.0148	1.2309	0.0866
	BoostedRgb	0.7888	0.0065	0.8777	0.0081	0.4409	0.0129	0.6416	0.0595
	NonlinearRbfSvm	0.8513	0.0084	0.9323	0.0058	0.3327	0.0136	39.9249	0.5459
	ExtraTrees	0.9158	0.0075	0.9744	0.0038	0.2388	0.0316	0.1147	0.0247
	Adaboost	0.9101	0.0082	0.9592	0.0057	2.2027	0.2259	3.9940	0.1783
	GradientBoosting	0.8662	0.0088	0.9433	0.0051	0.3064	0.0120	0.7002	0.0080
	RandomForest	0.9111	0.0074	0.9698	0.0041	0.2584	0.0285	0.4949	0.1052
pima	GaussNaiveBayes	0.7543	0.0301	0.8289	0.0263	0.6201	0.1088	0.0045	0.0005
	KernelNaiveBayes	0.7550	0.0308	0.8415	0.0278	0.5557	0.0755	0.0105	0.0006
	LinearDiscriminant	0.7678	0.0318	0.8502	0.0238	0.4835	0.0331	0.0135	0.0043
	GlmNet	0.7704	0.0343	0.8506	0.0258	0.4855	0.0376	0.2393	0.0045
	Cart	0.7528	0.0211	0.8013	0.0251	0.5728	0.1006	0.0115	0.0008
	C4.5DecisionTree	0.7367	0.0304	0.7812	0.0343	1.8937	0.8477	0.0334	0.0011
	LinearSvm	0.7697	0.0315	0.8515	0.0240	0.4855	0.0357	1.3237	0.0641
	RegGaussBayes-diag	0.7539	0.0294	0.8286	0.0261	0.6145	0.1042	0.0018	0.0004
	RegGaussBayes-lowran	0.7515	0.0294	0.8220	0.0234	0.6152	0.1033	0.0049	0.0014
	RegGaussBayes	0.7452	0.0262	0.8259	0.0250	0.6009	0.0857	0.0016	0.0005
	BoostedRgb-diag	0.7654	0.0265	0.8510	0.0264	0.5468	0.0150	0.3639	0.0584
	BoostedRgb-lowrank	0.7606	0.0315	0.8479	0.0256	0.5403	0.0171	0.4726	0.0898
	BoostedRgb	0.7606	0.0336	0.8428	0.0277	0.5432	0.0193	0.3463	0.0595
	NonlinearRbfSvm	0.7624	0.0338	0.8505	0.0252	0.4832	0.0371	1.3033	0.0184
	ExtraTrees	0.7621	0.0312	0.8417	0.0216	0.4895	0.0268	0.0218	0.0061
	Adaboost	0.7370	0.0196	0.8119	0.0247	4.8052	1.8972	0.3059	0.1180
	GradientBoosting	0.7446	0.0311	0.8261	0.0229	0.5577	0.0549	0.1356	0.0018
	RandomForest	0.7674	0.0330	0.8460	0.0221	0.4917	0.0519	0.0698	0.0012
segmentation	GaussNaiveBayes	0.7955	0.0128	0.9788	0.0028	1.5825	0.1984	0.0124	0.0006
	KernelNaiveBayes	0.8984	0.0108	0.9799	0.0046	1.0770	0.1986	0.0636	0.0006
	LinearDiscriminant	0.9147	0.0078	0.9931	0.0013	0.3424	0.0577	0.0363	0.0038
	GlmNet	0.9496	0.0097	0.9979	0.0009	0.1546	0.0407	4.2433	0.9199
	Cart	0.9205	0.0112	0.9907	0.0027	0.3433	0.0995	0.0488	0.0009
	C4.5DecisionTree	0.9662	0.0085	0.9858	0.0045	0.8090	0.2459	0.0594	0.0036
	LinearSvm	0.9571	0.0082	0.9985	0.0006	0.1392	0.0140	4.2207	0.0890
	RegGaussBayes-diag	0.8056	0.0127	0.9806	0.0029	1.4065	0.2045	0.0059	0.0004
	RegGaussBayes-lowran	0.8317	0.0140	0.9861	0.0021	1.0817	0.1769	0.0167	0.0026
	RegGaussBayes	0.8714	0.0120	0.9921	0.0018	0.8093	0.1669	0.0055	0.0005
	BoostedRgb-diag	0.6346	0.0579	0.9549	0.0067	1.0066	0.1862	2.4919	0.0940
	BoostedRgb-lowrank	0.9113	0.0140	0.9927	0.0024	0.6390	0.0478	3.1743	0.1237
	BoostedRgb	0.9620	0.0108	0.9976	0.0009	0.3500	0.0420	2.5000	0.0918
	NonlinearRbfSvm	0.9706	0.0070	0.9990	0.0006	0.1015	0.0141	8.6298	0.0789
	ExtraTrees	0.9792	0.0068	0.9996	0.0002	0.0903	0.0097	0.0418	0.0041
	Adaboost	0.9840	0.0070	0.9990	0.0006	0.4447	0.1983	1.9053	0.0306
	GradientBoosting	0.9782	0.0060	0.9995	0.0003	0.0667	0.0163	2.2842	0.0129
	RandomForest	0.9782	0.0064	0.9996	0.0002	0.0885	0.0096	0.2834	0.0229
soldat	GaussNaiveBayes	0.7005	0.0175	0.7705	0.0163	3.1914	0.2397	0.0659	0.0010
	KernelNaiveBayes	0.6980	0.0195	0.7851	0.0185	2.4690	0.2222	0.1164	0.0020
	LinearDiscriminant	0.7660	0.0151	0.8415	0.0132	0.5047	0.0262	0.2065	0.0245
	GlmNet	0.7762	0.0156	0.8486	0.0137	0.4882	0.0213	21.5928	17.5295
	Cart	0.7373	0.0209	0.7782	0.0177	0.5613	0.0225	0.4308	0.0174
	C4.5DecisionTree	0.7322	0.0133	0.7305	0.0193	5.4241	0.7314	0.4884	0.0184
	LinearSvm	0.7822	0.0136	0.8492	0.0128	0.4874	0.0193	373.8409	39.6586
	RegGaussBayes-diag	0.7007	0.0173	0.7707	0.0161	3.1769	0.2358	0.0153	0.0006
	RegGaussBayes-lowran	0.6882	0.0173	0.7449	0.0150	2.2578	0.1812	0.0425	0.0031
	RegGaussBayes	0.7077	0.0126	0.8161	0.0120	2.8035	0.3618	0.0151	0.0004
	BoostedRgb-diag	0.7300	0.0175	0.8078	0.0154	0.5431	0.0140	3.0874	0.1154
	BoostedRgb-lowrank	0.7414	0.0168	0.8175	0.0138	0.5594	0.0080	4.0400	0.1946

Dataset	Algorithm	Accuracy (Mean)	Accuracy (Std Dev.)	AUC (Mean)	AUC (Std Dev.)	Log-Loss (Mean)	Log-Loss (Std Dev.)	Time (Mean)	Time (Std Dev.)
	BoostedRgb	0.7947	0.0128	0.8670	0.0115	0.5236	0.0092	3.0655	0.1162
	NonlinearRbfSvm	0.8098	0.0115	0.8817	0.0117	0.4332	0.0194	152.4475	1.6798
	ExtraTrees	0.8179	0.0136	0.8978	0.0096	0.4368	0.0309	0.2821	0.0211
	Adaboost	0.7906	0.0194	0.8627	0.0143	2.8301	1.3979	26.3659	24.0383
	GradientBoosting	0.7917	0.0131	0.8702	0.0103	0.4525	0.0152	5.4077	0.0916
	RandomForest	0.8070	0.0131	0.8889	0.0099	0.4410	0.0258	2.9764	0.0828
sonar	GaussNaiveBayes	0.6823	0.0697	0.7759	0.0598	2.4657	0.8062	0.0214	0.0006
	KernelNaiveBayes	0.7522	0.0739	0.8394	0.0564	1.4532	0.6685	0.0667	0.0011
	LinearDiscriminant	0.7427	0.0580	0.8258	0.0553	0.5781	0.1321	0.0236	0.0163
	GlmNet	0.7322	0.0596	0.8246	0.0541	0.7420	0.5421	0.3975	0.0403
	Cart	0.7092	0.0554	0.7313	0.0687	0.9253	0.5777	0.0179	0.0008
	C4.5DecisionTree	0.7389	0.0472	0.7433	0.0666	6.3210	2.2377	0.0586	0.0018
	LinearSvm	0.7526	0.0689	0.8342	0.0576	0.5066	0.0757	0.6274	0.0313
	RegGaussBayes-diag	0.6816	0.0661	0.7746	0.0600	2.3561	0.7754	0.0028	0.0005
	RegGaussBayes-lowran	0.7538	0.0661	0.8636	0.0515	1.0584	0.5268	0.0115	0.0025
	RegGaussBayes	0.7885	0.0620	0.8668	0.0511	1.3836	0.6222	0.0018	0.0004
	BoostedRgb-diag	0.8272	0.0474	0.8920	0.0499	0.4976	0.0418	0.4548	0.0729
	BoostedRgb-lowrank	0.8424	0.0612	0.9256	0.0487	0.4357	0.0512	0.6529	0.0776
	BoostedRgb	0.8032	0.0639	0.8961	0.0424	0.4271	0.0558	0.4427	0.0584
	NonlinearRbfSvm	0.8640	0.0495	0.9325	0.0397	0.3297	0.0897	0.7156	0.0371
	ExtraTrees	0.8759	0.0458	0.9501	0.0290	0.3938	0.0357	0.0118	0.0019
	Adaboost	0.8306	0.0539	0.9040	0.0540	3.7407	1.7678	0.4996	0.0854
	GradientBoosting	0.8542	0.0569	0.9313	0.0367	0.3700	0.1445	0.2190	0.0034
	RandomForest	0.8271	0.0538	0.9253	0.0346	0.4155	0.0352	0.0536	0.0014
soybean	GaussNaiveBayes	0.5314	0.0398	0.9169	0.0078	11.1328	0.7397	0.0464	0.0021
	KernelNaiveBayes	0.7196	0.0330	0.9783	0.0033	2.4726	0.3102	0.4950	0.0060
	LinearDiscriminant	0.9218	0.0263	0.9940	0.0054	0.6747	0.3747	0.0375	0.0081
	GlmNet	0.1628	0.0038	0.6844	0.0019	2.4720	0.0049	2.7225	1.1734
	Cart	0.8752	0.0267	0.9912	0.0058	0.5821	0.3347	0.0283	0.0011
	C4.5DecisionTree	0.9063	0.0256	0.9797	0.0096	1.3473	0.6005	0.0753	0.0031
	LinearSvm	0.9119	0.0241	0.9969	0.0022	0.4087	0.0502	2.6872	0.1108
	RegGaussBayes-diag	0.9226	0.0263	0.9955	0.0050	1.4024	0.6685	0.0108	0.0006
	RegGaussBayes-lowran	0.8959	0.0301	0.9965	0.0023	0.9676	0.3985	0.0324	0.0024
	RegGaussBayes	0.9341	0.0238	0.9965	0.0045	0.8957	0.4983	0.0092	0.0004
	BoostedRgb-diag	0.8702	0.0182	0.9880	0.0015	0.8038	0.1126	4.5080	0.1213
	BoostedRgb-lowrank	0.9037	0.0292	0.9926	0.0044	0.8408	0.2444	6.8065	0.4379
	BoostedRgb	0.9327	0.0258	0.9974	0.0019	0.2392	0.1186	4.4065	0.1133
	NonlinearRbfSvm	0.9182	0.0203	0.9970	0.0027	0.4017	0.0401	3.7747	0.1261
	ExtraTrees	0.9187	0.0239	0.9965	0.0031	0.3101	0.1775	0.0245	0.0030
	Adaboost	0.9077	0.0275	0.9942	0.0051	1.4485	1.0093	0.5504	0.1864
	GradientBoosting	0.9178	0.0287	0.9984	0.0009	0.2141	0.0741	2.8212	0.0503
	RandomForest	0.9241	0.0240	0.9985	0.0008	0.2694	0.0353	0.1850	0.0026
spam	GaussNaiveBayes	0.7122	0.0146	0.8303	0.0138	8.1549	0.5950	0.0502	0.0019
	KernelNaiveBayes	0.6091	0.0203	0.7605	0.0179	8.8350	0.6016	0.0845	0.0013
	LinearDiscriminant	0.8873	0.0123	0.9503	0.0070	0.2897	0.0257	0.1723	0.0064
	GlmNet	0.9266	0.0088	0.9718	0.0050	0.2330	0.0267	4.6668	4.7177
	Cart	0.8924	0.0120	0.9096	0.0125	0.3421	0.0426	0.1905	0.0054
	C4.5DecisionTree	0.9243	0.0098	0.9386	0.0108	0.9430	0.1904	0.2592	0.0093
	LinearSvm	0.9283	0.0082	0.9720	0.0043	0.2382	0.0219	103.3359	13.5562
	RegGaussBayes-diag	0.7769	0.0128	0.8983	0.0086	4.0132	0.3390	0.0103	0.0006
	RegGaussBayes-lowran	0.8381	0.0669	0.9154	0.0405	0.5241	0.2385	0.0320	0.0028
	RegGaussBayes	0.7935	0.0126	0.9121	0.0085	3.0446	0.3180	0.0102	0.0006
	BoostedRgb-diag	0.8395	0.0313	0.9149	0.0152	1.3056	0.3251	2.3553	0.1388
	BoostedRgb-lowrank	0.9170	0.0089	0.9635	0.0067	0.3907	0.0177	3.0237	0.1997
	BoostedRgb	0.8708	0.0209	0.9437	0.0090	0.6553	0.1715	2.3269	0.1046
	NonlinearRbfSvm	0.9345	0.0071	0.9785	0.0045	0.1866	0.0200	66.0356	1.0406
	ExtraTrees	0.9569	0.0063	0.9880	0.0039	0.1988	0.0528	0.2639	0.0213
	Adaboost	0.9510	0.0067	0.9845	0.0043	0.5290	0.2475	5.1649	4.9506
	GradientBoosting	0.9491	0.0065	0.9859	0.0035	0.1430	0.0154	3.6536	0.1013
	RandomForest	0.9523	0.0068	0.9873	0.0038	0.1776	0.0392	2.1290	0.1324

Dataset	Algorithm	Accuracy (Mean)	Accuracy (Std Dev.)	AUC (Mean)	AUC (Std Dev.)	Log-Loss (Mean)	Log-Loss (Std Dev.)	Time (Mean)	Time (Std Dev.)
thyroid	GaussNaiveBayes	0.9503	0.0032	0.9872	0.0019	0.2501	0.0326	0.0098	0.0006
	KernelNaiveBayes	0.9684	0.0043	0.9956	0.0016	0.1468	0.0421	0.0186	0.0006
	LinearDiscriminant	0.9367	0.0019	0.9843	0.0015	0.2768	0.0257	0.0605	0.0094
	GlmNet	0.9452	0.0027	0.9944	0.0010	0.1709	0.0181	1.2684	0.1360
	Cart	0.9856	0.0036	0.9993	0.0003	0.0436	0.0119	0.0198	0.0009
	C4.5DecisionTree	0.9830	0.0043	0.9958	0.0019	0.2018	0.0833	0.0449	0.0018
	LinearSvm	0.9440	0.0024	0.9956	0.0011	0.1620	0.0077	48.4140	26.2523
	RegGaussBayes-diag	0.9500	0.0030	0.9871	0.0020	0.2501	0.0346	0.0049	0.0004
	RegGaussBayes-lowran	0.9322	0.0031	0.9829	0.0013	0.2439	0.0195	0.0126	0.0039
	RegGaussBayes	0.9464	0.0034	0.9869	0.0023	0.2765	0.0465	0.0046	0.0005
	BoostedRgb-diag	0.9485	0.0032	0.9882	0.0014	0.3977	0.0261	1.6006	0.0780
	BoostedRgb-lowrank	0.9424	0.0020	0.9867	0.0012	0.4685	0.0236	2.4989	0.1167
	BoostedRgb	0.9469	0.0036	0.9870	0.0018	0.3946	0.0455	1.6128	0.0858
	NonlinearRbfSvm	0.9644	0.0043	0.9958	0.0014	0.1100	0.0147	22.5138	0.5126
	ExtraTrees	0.9723	0.0047	0.9989	0.0002	0.0686	0.0047	0.0887	0.0075
	Adaboost	0.9840	0.0043	0.9970	0.0010	0.4508	0.1315	2.0595	0.1078
	GradientBoosting	0.9837	0.0036	0.9995	0.0002	0.0412	0.0085	1.4582	0.0113
	RandomForest	0.9842	0.0042	0.9995	0.0002	0.0416	0.0055	0.8033	0.0565
titanic	GaussNaiveBayes	0.7707	0.0189	0.7561	0.0231	0.8017	0.0849	0.0040	0.0005
	KernelNaiveBayes	0.7072	0.0229	0.7704	0.0134	0.7151	0.0534	0.0079	0.0005
	LinearDiscriminant	0.7783	0.0142	0.8230	0.0177	0.5094	0.0245	0.0160	0.0034
	GlmNet	0.7781	0.0144	0.8250	0.0175	0.5057	0.0191	0.3931	0.0088
	Cart	0.7815	0.0155	0.8125	0.0238	0.5067	0.0282	0.0098	0.0007
	C4.5DecisionTree	0.7902	0.0135	0.8278	0.0152	0.4899	0.0206	0.0330	0.0014
	LinearSvm	0.7760	0.0144	0.7903	0.0203	0.5310	0.0170	3.3133	0.1426
	RegGaussBayes-diag	0.7707	0.0189	0.7556	0.0229	0.7978	0.0842	0.0023	0.0005
	RegGaussBayes-lowran	0.7733	0.0139	0.8108	0.0193	0.8785	0.1204	0.0060	0.0026
	RegGaussBayes	0.7456	0.0171	0.8050	0.0213	0.7500	0.0811	0.0019	0.0003
	BoostedRgb-diag	0.7729	0.0140	0.8034	0.0180	0.5595	0.0136	0.3675	0.0481
	BoostedRgb-lowrank	0.7815	0.0143	0.8434	0.0153	0.5420	0.0106	0.6829	0.0634
	BoostedRgb	0.7805	0.0146	0.8377	0.0181	0.5393	0.0107	0.3727	0.0688
	NonlinearRbfSvm	0.7873	0.0148	0.8242	0.0147	0.5042	0.0185	6.9175	0.0988
	ExtraTrees	0.7893	0.0135	0.7892	0.0135	7.2787	0.4646	0.0141	0.0025
	Adaboost	0.7881	0.0131	0.8457	0.0150	0.4819	0.0216	0.1816	0.0137
	GradientBoosting	0.7890	0.0138	0.8456	0.0152	0.4819	0.0214	0.2654	0.0049
	RandomForest	0.7905	0.0128	0.8038	0.0149	4.4174	0.4383	0.0901	0.0014
vehicle	GaussNaiveBayes	0.4492	0.0309	0.7424	0.0210	2.5792	0.3559	0.0097	0.0005
	KernelNaiveBayes	0.6212	0.0333	0.8551	0.0197	2.2125	0.4032	0.0416	0.0018
	LinearDiscriminant	0.7812	0.0208	0.9561	0.0069	0.4860	0.0394	0.0182	0.0033
	GlmNet	0.7977	0.0252	0.9659	0.0055	0.4174	0.0365	2.1725	0.4775
	Cart	0.6750	0.0323	0.8938	0.0202	1.0589	0.4868	0.0184	0.0009
	C4.5DecisionTree	0.7161	0.0361	0.8683	0.0221	4.8899	1.0118	0.0454	0.0011
	LinearSvm	0.7993	0.0248	0.9664	0.0065	0.4246	0.0337	2.3697	0.1516
	RegGaussBayes-diag	0.4488	0.0310	0.7412	0.0207	2.5784	0.3532	0.0033	0.0004
	RegGaussBayes-lowran	0.6056	0.0396	0.8524	0.0231	1.3475	0.2239	0.0103	0.0022
	RegGaussBayes	0.8188	0.0279	0.9680	0.0075	0.4712	0.1287	0.0027	0.0004
	BoostedRgb-diag	0.6996	0.0437	0.9016	0.0198	1.0494	0.0192	0.9297	0.0592
	BoostedRgb-lowrank	0.7502	0.0243	0.9375	0.0089	0.9163	0.0265	1.2097	0.0818
	BoostedRgb	0.8387	0.0256	0.9739	0.0052	0.6273	0.0235	0.9174	0.0665
	NonlinearRbfSvm	0.8417	0.0300	0.9766	0.0065	0.3501	0.0391	2.5912	0.0171
	ExtraTrees	0.7398	0.0267	0.9534	0.0076	0.5270	0.0309	0.0303	0.0044
	Adaboost	0.7681	0.0303	0.9499	0.0094	5.3659	0.9008	0.8032	0.0139
	GradientBoosting	0.7668	0.0248	0.9591	0.0070	0.4622	0.0452	0.5145	0.0111
	RandomForest	0.7455	0.0253	0.9534	0.0070	0.5153	0.0284	0.1275	0.0025
vowel	GaussNaiveBayes	0.5736	0.0362	0.9355	0.0077	1.1352	0.0728	0.0074	0.0005
	KernelNaiveBayes	0.6684	0.0349	0.9599	0.0064	0.8961	0.0672	0.0541	0.0005
	LinearDiscriminant	0.5409	0.0375	0.9219	0.0092	1.2634	0.0825	0.0182	0.0059
	GlmNet	0.5921	0.0369	0.9349	0.0086	1.1649	0.0899	1.3989	0.0639
	Cart	0.5271	0.0327	0.8924	0.0165	2.4466	0.5865	0.0329	0.0011
	C4.5DecisionTree	0.7470	0.0336	0.8905	0.0200	6.6400	1.2020	0.0463	0.0025

Dataset	Algorithm	Accuracy (Mean)	Accuracy (Std Dev.)	AUC (Mean)	AUC (Std Dev.)	Log-Loss (Mean)	Log-Loss (Std Dev.)	Time (Mean)	Time (Std Dev.)
	LinearSvm	0.7170	0.0397	0.9693	0.0056	0.8915	0.0482	3.0605	0.1347
	RegGaussBayes-diag	0.5660	0.0363	0.9345	0.0079	1.1412	0.0674	0.0069	0.0003
	RegGaussBayes-lowran	0.6776	0.0322	0.9594	0.0054	0.9310	0.0807	0.0155	0.0020
	RegGaussBayes	0.7914	0.0296	0.9847	0.0038	0.5428	0.0718	0.0060	0.0002
	BoostedRgb-diag	0.6401	0.0317	0.9401	0.0066	1.7177	0.0209	2.4126	0.1121
	BoostedRgb-lowrank	0.7830	0.0288	0.9739	0.0035	1.5321	0.0213	2.7674	0.1669
	BoostedRgb	0.9190	0.0185	0.9920	0.0033	1.2200	0.0249	2.3068	0.0898
	NonlinearRbfSvm	0.9726	0.0128	0.9995	0.0004	0.1590	0.0231	3.4557	0.0437
	ExtraTrees	0.9727	0.0097	0.9994	0.0003	0.3888	0.0234	0.0339	0.0033
	Adaboost	0.9377	0.0181	0.9968	0.0019	1.2380	0.4919	1.3302	0.0315
	GradientBoosting	0.8854	0.0280	0.9925	0.0029	0.3889	0.0535	1.0184	0.0239
	RandomForest	0.9429	0.0168	0.9980	0.0009	0.4741	0.0236	0.1378	0.0037
wine	GaussNaiveBayes	0.9720	0.0293	0.9982	0.0022	0.0928	0.0977	0.0061	0.0004
	KernelNaiveBayes	0.9683	0.0298	0.9975	0.0035	0.1257	0.1307	0.0224	0.0013
	LinearDiscriminant	0.9851	0.0217	0.9993	0.0012	0.0404	0.0418	0.0098	0.0052
	GlmNet	0.9757	0.0291	0.9982	0.0027	0.1166	0.0593	0.3614	0.0055
	Cart	0.8659	0.0534	0.9368	0.0327	1.2075	0.9555	0.0073	0.0007
	C4.5DecisionTree	0.9268	0.0426	0.9454	0.0296	2.0164	1.1158	0.0332	0.0009
	LinearSvm	0.9776	0.0212	0.9986	0.0016	0.1066	0.0284	0.2046	0.0044
	RegGaussBayes-diag	0.9767	0.0254	0.9979	0.0028	0.0974	0.1135	0.0036	0.0005
	RegGaussBayes-lowran	0.9766	0.0255	0.9976	0.0048	0.1067	0.1303	0.0075	0.0027
	RegGaussBayes	0.9898	0.0172	0.9996	0.0007	0.0302	0.0406	0.0018	0.0004
	BoostedRgb-diag	0.9795	0.0273	0.9984	0.0037	0.2687	0.0390	0.5364	0.0677
	BoostedRgb-lowrank	0.9789	0.0220	0.9984	0.0028	0.2663	0.0303	0.5718	0.0799
	BoostedRgb	0.9874	0.0150	0.9994	0.0010	0.1559	0.0184	0.5116	0.0825
	NonlinearRbfSvm	0.9832	0.0240	0.9991	0.0014	0.0814	0.0321	0.2582	0.0033
	ExtraTrees	0.9832	0.0215	0.9993	0.0011	0.1445	0.0282	0.0054	0.0009
	Adaboost	0.9665	0.0266	0.9925	0.0132	0.8094	0.7840	0.0663	0.0218
	GradientBoosting	0.9804	0.0234	0.9986	0.0023	0.0723	0.0836	0.0729	0.0011
	RandomForest	0.9822	0.0202	0.9992	0.0013	0.1361	0.0292	0.0144	0.0005
adult	GaussNaiveBayes	0.6117	0.0252	0.7265	0.0317	7.6476	1.6022	0.5404	0.0129
	KernelNaiveBayes	0.8212	0.0317	0.9154	0.0310	0.6681	0.1793	0.4979	0.0109
	LinearDiscriminant	0.8379	0.0031	0.9230	0.0026	0.3525	0.0062	2.4109	0.1756
	GlmNet	-	-	-	-	-	-	-	-
	Cart	0.8442	0.0044	0.9081	0.0041	0.3729	0.0067	3.8932	0.0612
	C4.5DecisionTree	0.8616	0.0027	0.9218	0.0031	0.6560	0.0555	10.8843	0.3059
	LinearSvm	-	-	-	-	-	-	-	-
	RegGaussBayes-diag	0.6372	0.0124	0.7628	0.0097	6.2068	0.4600	0.2185	0.0057
	RegGaussBayes-lowran	0.6978	0.0155	0.7728	0.0181	1.4747	0.1457	0.6461	0.0116
	RegGaussBayes	0.6404	0.0097	0.7652	0.0101	6.2812	0.4673	0.2169	0.0044
	BoostedRgb-diag	0.8062	0.0110	0.8710	0.0196	1.9236	0.6458	18.7086	0.5003
	BoostedRgb-lowrank	0.7121	0.0082	0.8593	0.0051	0.4640	0.0142	36.6319	0.3734
	BoostedRgb	0.8105	0.0073	0.8781	0.0226	1.3672	0.7504	18.6152	0.4807
	BoostedRgb-iso	0.8157	0.0129	0.8726	0.0124	0.4546	0.0388	30.7228	0.3765
	NonlinearRbfSvm	-	-	-	-	-	-	-	-
	ExtraTrees	0.8338	0.0037	0.9164	0.0026	0.5729	0.0309	8.4013	0.1442
	Adaboost	-	-	-	-	-	-	-	-
	GradientBoosting	0.8658	0.0030	0.9470	0.0019	0.2947	0.0048	57.7679	0.7587
	RandomForest	0.8608	0.0034	0.9312	0.0025	0.4758	0.0214	62.3351	1.8925
dna	GaussNaiveBayes	0.9301	0.0254	0.9889	0.0060	0.9450	0.3740	0.1304	0.0040
	KernelNaiveBayes	0.7207	0.0287	0.9161	0.0153	1.8444	0.5374	0.3409	0.0061
	LinearDiscriminant	0.9480	0.0077	0.9926	0.0014	0.1587	0.0182	0.3297	0.0103
	GlmNet	-	-	-	-	-	-	-	-
	Cart	0.9058	0.0093	0.9701	0.0042	0.3318	0.0549	0.2956	0.0058
	C4.5DecisionTree	0.9222	0.0083	0.9443	0.0103	1.5850	0.3161	0.4286	0.0055
	LinearSvm	-	-	-	-	-	-	-	-
	RegGaussBayes-diag	0.9445	0.0093	0.9922	0.0016	0.2961	0.0540	0.0467	0.0013
	RegGaussBayes-lowran	0.9183	0.0123	0.9851	0.0032	0.2683	0.0439	0.1151	0.0048
	RegGaussBayes	0.9416	0.0080	0.9920	0.0018	0.3559	0.0733	0.0451	0.0009
	BoostedRgb-diag	0.9482	0.0077	0.9887	0.0024	0.4476	0.0174	8.1810	0.0845

Dataset	Algorithm	Accuracy (Mean)	Accuracy (Std Dev.)	AUC (Mean)	AUC (Std Dev.)	Log-Loss (Mean)	Log-Loss (Std Dev.)	Time (Mean)	Time (Std Dev.)
	BoostedRgb-lowrank	0.9392	0.0074	0.9880	0.0020	0.4581	0.0107	13.4676	0.2913
	BoostedRgb	0.8722	0.0175	0.9725	0.0086	0.4934	0.0435	8.1674	0.1408
	BoostedRgb-iso	0.9130	0.0113	0.9921	0.0031	0.3094	0.0194	15.6467	0.1837
	NonlinearRbfSvm	-	-	-	-	-	-	-	-
	ExtraTrees	0.9565	0.0062	0.9938	0.0013	0.2749	0.0250	0.1694	0.0080
	Adaboost	-	-	-	-	-	-	-	-
	GradientBoosting	0.9617	0.0052	0.9951	0.0013	0.1195	0.0131	10.8253	0.1659
	RandomForest	0.9546	0.0075	0.9939	0.0011	0.2930	0.0091	3.7014	0.0635
letter	GaussNaiveBayes	0.6428	0.0065	0.9566	0.0016	1.6262	0.0470	0.0522	0.0036
	KernelNaiveBayes	0.7053	0.0075	0.9769	0.0010	1.3615	0.0419	0.2528	0.0154
	LinearDiscriminant	0.7024	0.0071	0.9663	0.0013	1.3588	0.0464	0.3046	0.1465
	GlmNet	-	-	-	-	-	-	-	-
	Cart	0.4737	0.0101	0.9024	0.0033	1.8439	0.0302	0.3136	0.0086
	C4.5DecisionTree	0.8738	0.0053	0.9513	0.0023	3.1950	0.1496	0.7382	0.2186
	LinearSvm	-	-	-	-	-	-	-	-
	RegGaussBayes-diag	0.6412	0.0064	0.9565	0.0015	1.6188	0.0464	0.1746	0.0056
	RegGaussBayes-lowran	0.6818	0.0070	0.9667	0.0013	1.5060	0.0458	0.2042	0.0132
	RegGaussBayes	0.8822	0.0048	0.9961	0.0004	0.5067	0.0307	0.1712	0.0076
	BoostedRgb-diag	0.6778	0.0116	0.9686	0.0014	1.7418	0.0416	68.4690	0.8660
	BoostedRgb-lowrank	0.7061	0.0135	0.9739	0.0016	1.6866	0.0382	78.8411	0.9308
	BoostedRgb	0.9293	0.0034	0.9966	0.0003	1.5663	0.0125	40.8969	0.4413
	BoostedRgb-iso	0.9173	0.0039	0.9973	0.0002	0.7971	0.0141	69.8116	0.6933
	NonlinearRbfSvm	-	-	-	-	-	-	-	-
	ExtraTrees	0.9728	0.0020	0.9997	0.0001	0.2669	0.0077	0.7534	0.1247
	Adaboost	-	-	-	-	-	-	-	-
	GradientBoosting	0.8994	0.0041	0.9967	0.0004	0.4127	0.0104	65.7891	0.7363
	RandomForest	0.9643	0.0030	0.9995	0.0001	0.2759	0.0065	6.5600	0.7564
optdigits	GaussNaiveBayes	0.8085	0.0173	0.9762	0.0033	3.1053	0.3817	0.1941	0.6455
	KernelNaiveBayes	0.8698	0.0091	0.9847	0.0025	1.4904	0.1615	0.3619	0.0132
	LinearDiscriminant	0.9502	0.0061	0.9976	0.0006	0.2357	0.0379	0.2554	0.2182
	GlmNet	-	-	-	-	-	-	-	-
	Cart	0.7648	0.0204	0.9427	0.0066	1.0289	0.0871	0.2441	0.0546
	C4.5DecisionTree	0.9017	0.0101	0.9529	0.0048	2.7110	0.2866	0.3764	0.0973
	LinearSvm	-	-	-	-	-	-	-	-
	RegGaussBayes-diag	0.9035	0.0087	0.9868	0.0021	1.5196	0.1274	0.0236	0.0008
	RegGaussBayes-lowran	0.9230	0.0085	0.9916	0.0021	0.8824	0.1255	0.0807	0.0147
	RegGaussBayes	0.9746	0.0039	0.9952	0.0018	0.4974	0.1014	0.0225	0.0008
	BoostedRgb-diag	0.7564	0.0350	0.7676	0.1150	6.8694	3.9140	12.8413	0.2481
	BoostedRgb-lowrank	0.9651	0.0077	0.9966	0.0017	0.8470	0.0393	18.7437	0.5675
	BoostedRgb	0.9869	0.0036	0.9992	0.0006	0.4369	0.0449	12.7754	0.1817
	BoostedRgb-iso	0.9885	0.0030	0.9997	0.0002	0.1911	0.0102	26.1351	0.4231
	NonlinearRbfSvm	-	-	-	-	-	-	-	-
	ExtraTrees	0.9860	0.0033	0.9998	0.0001	0.2018	0.0073	0.2649	0.2034
	Adaboost	-	-	-	-	-	-	-	-
	GradientBoosting	0.9729	0.0039	0.9994	0.0002	0.0997	0.0107	23.0262	1.3467
	RandomForest	0.9830	0.0030	0.9997	0.0001	0.2225	0.0077	2.7606	0.1462
penbased	GaussNaiveBayes	0.8577	0.0065	0.9842	0.0012	1.3209	0.0974	0.0304	0.0010
	KernelNaiveBayes	0.8641	0.0051	0.9880	0.0012	0.9869	0.0842	0.1054	0.0024
	LinearDiscriminant	0.8760	0.0067	0.9913	0.0008	0.5450	0.0401	0.1631	0.0621
	GlmNet	-	-	-	-	-	-	-	-
	Cart	0.8155	0.0117	0.9709	0.0024	0.7045	0.0342	0.1317	0.0043
	C4.5DecisionTree	0.9617	0.0040	0.9830	0.0023	0.9707	0.1339	0.2453	0.1572
	LinearSvm	-	-	-	-	-	-	-	-
	RegGaussBayes-diag	0.8579	0.0062	0.9842	0.0012	1.2977	0.0940	0.0200	0.0009
	RegGaussBayes-lowran	0.9456	0.0042	0.9973	0.0005	0.3672	0.0437	0.0488	0.0019
	RegGaussBayes	0.9829	0.0031	0.9990	0.0004	0.1681	0.0385	0.0188	0.0006
	BoostedRgb-diag	0.9143	0.0071	0.9906	0.0013	1.1023	0.0342	9.8933	0.1530
	BoostedRgb-lowrank	0.9850	0.0026	0.9990	0.0004	0.9829	0.0340	12.8496	0.2612
	BoostedRgb	0.9952	0.0015	0.9996	0.0003	0.5347	0.0266	9.8765	0.1597
	BoostedRgb-iso	0.9951	0.0015	0.9997	0.0002	0.2558	0.0105	19.5584	0.2724

Dataset	Algorithm	Accuracy (Mean)	Accuracy (Std Dev.)	AUC (Mean)	AUC (Std Dev.)	Log-Loss (Mean)	Log-Loss (Std Dev.)	Time (Mean)	Time (Std Dev.)
	NonlinearRbfSvm	-	-	-	-	-	-	-	-
	ExtraTrees	0.9936	0.0016	0.9997	0.0002	0.0810	0.0106	0.1983	0.0317
	Adaboost	-	-	-	-	-	-	-	-
	GradientBoosting	0.9862	0.0024	0.9996	0.0002	0.0580	0.0060	14.2015	0.1739
	RandomForest	0.9913	0.0019	0.9997	0.0002	0.0869	0.0101	2.4053	0.1006
	GaussNaiveBayes	0.7961	0.0102	0.9619	0.0023	3.9450	0.1996	0.0414	0.0025
	KernelNaiveBayes	0.8198	0.0103	0.9719	0.0020	3.2370	0.1748	0.1372	0.0041
	LinearDiscriminant	0.8396	0.0077	0.9774	0.0019	0.6362	0.0677	0.1546	0.0056
	GlmNet	-	-	-	-	-	-	-	-
	Cart	0.8108	0.0153	0.9555	0.0082	0.6556	0.0650	0.1414	0.0036
	C4.5DecisionTree	0.8627	0.0093	0.9316	0.0060	3.2091	0.2894	0.2662	0.0122
	LinearSvm	-	-	-	-	-	-	-	-
	RegGaussBayes-diag	0.7960	0.0102	0.9619	0.0023	3.9440	0.1996	0.0128	0.0005
	RegGaussBayes-lowran	0.7960	0.0107	0.9592	0.0029	1.8906	0.1312	0.0395	0.0019
satellite	RegGaussBayes	0.8567	0.0063	0.9783	0.0018	0.8565	0.0736	0.0123	0.0005
	BoostedRgb-diag	0.8429	0.0083	0.9627	0.0030	0.5040	0.0329	10.3129	0.3159
	BoostedRgb-lowrank	0.8428	0.0071	0.9736	0.0023	0.4590	0.0130	12.2525	0.2494
	BoostedRgb	0.8673	0.0081	0.9747	0.0018	0.4291	0.0229	7.4613	0.2411
	BoostedRgb-iso	0.8400	0.0051	0.9741	0.0023	0.5408	0.0094	11.0054	0.2548
	NonlinearRbfSvm	-	-	-	-	-	-	-	-
	ExtraTrees	0.9183	0.0064	0.9949	0.0005	0.2488	0.0095	0.1854	0.0238
	Adaboost	-	-	-	-	-	-	-	-
	GradientBoosting	0.9057	0.0059	0.9933	0.0007	0.2534	0.0114	9.7771	0.4548
	RandomForest	0.9172	0.0067	0.9944	0.0007	0.2531	0.0124	2.1746	0.0921
	GaussNaiveBayes	0.8485	0.0085	0.9846	0.0011	0.5785	0.0256	0.0764	0.0411
	KernelNaiveBayes	0.9904	0.0026	0.9988	0.0003	0.0720	0.0115	0.0986	0.0068
	LinearDiscriminant	0.8469	0.0442	0.9791	0.0111	1.1992	0.7149	0.4338	0.1562
	GlmNet	-	-	-	-	-	-	-	-
	Cart	0.9947	0.0003	0.9987	0.0001	0.0389	0.0031	0.1961	0.0039
	C4.5DecisionTree	0.9996	0.0002	0.9999	0.0001	0.0067	0.0044	0.5423	0.2645
	LinearSvm	-	-	-	-	-	-	-	-
	RegGaussBayes-diag	0.9228	0.0053	0.9944	0.0004	0.2961	0.0200	0.0543	0.0009
	RegGaussBayes-lowran	0.9282	0.0249	0.9913	0.0035	0.3671	0.1211	0.1188	0.0135
shuttle	RegGaussBayes	0.9625	0.0074	0.9980	0.0003	0.1507	0.0269	0.0535	0.0012
	BoostedRgb-diag	0.9890	0.0054	0.9996	0.0001	0.4459	0.0730	15.9592	0.1744
	BoostedRgb-lowrank	0.9598	0.0076	0.9985	0.0001	0.6822	0.0696	21.7860	0.2970
	BoostedRgb	0.9875	0.0065	0.9997	0.0002	0.4035	0.0432	16.0104	0.1429
	BoostedRgb-iso	0.9853	0.0042	0.9868	0.0096	0.2243	0.0519	32.5853	0.3561
	NonlinearRbfSvm	-	-	-	-	-	-	-	-
	ExtraTrees	0.9997	0.0001	1.0000	0.0000	0.0011	0.0005	0.6973	0.0766
	Adaboost	-	-	-	-	-	-	-	-
	GradientBoosting	0.9998	0.0001	1.0000	0.0000	0.0007	0.0005	35.8609	0.3060
	RandomForest	0.9998	0.0001	1.0000	0.0000	0.0009	0.0002	8.1312	5.1291
	GaussNaiveBayes	0.7750	0.0100	0.9781	0.0013	3.2204	0.1555	0.0480	0.0020
	KernelNaiveBayes	0.8016	0.0095	0.9816	0.0013	2.7484	0.1493	0.2580	0.0045
	LinearDiscriminant	0.9946	0.0017	1.0000	0.0000	0.0166	0.0068	0.1572	0.0373
	GlmNet	-	-	-	-	-	-	-	-
	Cart	0.8102	0.0133	0.9765	0.0022	0.7132	0.0639	0.3518	0.0110
	C4.5DecisionTree	0.9281	0.0089	0.9697	0.0044	1.8626	0.2630	0.2801	0.0321
	LinearSvm	-	-	-	-	-	-	-	-
	RegGaussBayes-diag	0.7741	0.0096	0.9780	0.0013	3.2252	0.1556	0.0175	0.0006
texture	RegGaussBayes-lowran	0.9602	0.0045	0.9988	0.0003	0.4649	0.0837	0.0591	0.0057
	RegGaussBayes	0.9983	0.0011	1.0000	0.0000	0.0097	0.0090	0.0165	0.0005
	BoostedRgb-diag	0.8231	0.0081	0.9777	0.0021	0.6214	0.0402	18.4564	0.3182
	BoostedRgb-lowrank	0.9739	0.0050	0.9990	0.0006	0.1635	0.0317	21.9618	0.3674
	BoostedRgb	0.9992	0.0008	1.0000	0.0000	0.0063	0.0020	13.8867	0.2360
	BoostedRgb-iso	0.9989	0.0011	1.0000	0.0000	0.0545	0.0086	19.2060	0.3763
	NonlinearRbfSvm	-	-	-	-	-	-	-	-
	ExtraTrees	0.9848	0.0032	0.9999	0.0001	0.1366	0.0058	0.1469	0.0187
	Adaboost	-	-	-	-	-	-	-	-

Dataset	Algorithm	Accuracy (Mean)	Accuracy (Std Dev.)	AUC (Mean)	AUC (Std Dev.)	Log-Loss (Mean)	Log-Loss (Std Dev.)	Time (Mean)	Time (Std Dev.)
	GradientBoosting	0.9782	0.0041	0.9995	0.0003	0.0806	0.0127	17.6259	0.1690
	RandomForest	0.9770	0.0038	0.9997	0.0001	0.1438	0.0071	1.7533	0.1321

ANNEX F – Overall Rank of the Classifiers

Frequency that each method ranks 1st, 2nd, 3rd, etc. It was obtained via bootstrap analysis on 1×10^3 bootstrap replicates of the problems. The algorithms are ranked by average accuracy. The 0.3304 entry for Extra Trees in the column for 1st place suggests that if a new problem is selected, there is a 33.04% chance that Extra Trees ranks 1st in terms of classification accuracy.

Algorithm	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
EXT	0.3304	0.1805	0.1201	0.1258	0.1198	0.0935	0.0000	0.0299	0.0000	0.0000
BRGB	0.2164	0.0904	0.1224	0.1498	0.0594	0.0596	0.1510	0.0915	0.0293	0.0302
RF	0.1500	0.3623	0.2712	0.0000	0.0925	0.0323	0.0610	0.0307	0.0000	0.0000
GBM	0.1227	0.1508	0.2704	0.2432	0.0621	0.0619	0.0302	0.0000	0.0587	0.0000
LDA	0.0889	0.0000	0.1563	0.0592	0.0609	0.2432	0.1776	0.0314	0.0911	0.0914
RGB	0.0619	0.0950	0.0000	0.0899	0.2449	0.0594	0.0601	0.2065	0.1822	0.0000
CART	0.0298	0.0000	0.0000	0.0607	0.2133	0.0874	0.0280	0.2776	0.0592	0.2439
GNB	0.0000	0.0000	0.0000	0.0000	0.0585	0.0881	0.0618	0.0603	0.3022	0.4292
KNB	0.0000	0.0601	0.0000	0.0894	0.0280	0.1221	0.1248	0.2116	0.1868	0.1773
C4.5	0.0000	0.0609	0.0596	0.1820	0.0605	0.1525	0.3055	0.0605	0.0905	0.0280