CHARLES UNIVERSITY IN PRAGUE

FACULTY OF PHARMACY IN HRADEC KRÁLOVÉ

Department of Biophysics and Physical Chemistry

THE SMALL SAMPLE SIZE PROBLEM IN GENE EXPRESSION TASKS

Diploma Thesis

Savvas Athanasiadis

May 2015

Supervisor: Erik Jurjen Duintjer Tebbens, Ph.D.

,, I declare that this thesis is my original work and I have not received any unauthorized assistance in it completion. All sources of information are named in references and cited properly. This thesis has not been used for obtaining a different or the same degree."

,, Prohlašuji, že tato práce je mým původním autorským dílem. Veškerá literatura a dalšízdroje, z nichž jsem při zpracováníčerpal, jsou uvedeny v seznamu použité literatury a v práci řádně citovány. Práce nebyla využita k získáni jiného nebo stejného titulu."

Hradec Králové

Savvas Athanasiadis

I would like to thank my supervisor Dr. Jurjen Duintjer Tebbens for his immense help and constant support and interest, my friend Zifos Ioannis for his guidance in linear algebra, and all my colleagues for their help and support.

Contents

Abstract

Charles University in Prague
Faculty of Pharmacy in Hradec Králové
Department of Biophysics and Physical Chemistry

Candidate: Savvas Athanasiadis
Supervisor: Jurjen Duintjer Tebbens
Title of diploma thesis: The small sample size problem in gene expression tasks

The thesis addresses classification of genes to tumor types based on their gene expression signatures. The number of variables (amino acids) to be investigated is typically very high (in the thousands) while it is expensive and time-consuming to analyze a high number of genes; usually at most tens of them are available. The combination of a small sample size with a large number of variables makes standard statistical classification methods inappropriate.

The thesis focuses on a modification of a standard classification method, Fisher's linear discriminant analysis, for the case where the number of samples is smaller than the number of variables. It proposes an improved strategy to test this modified method with leave-one-out cross validation. Using so-called low rank updates of the involved covariance matrices, the computational costs of the cross validation process can be reduced by an order of magnitude. Memory demands are reduced as well.

Abstrakt

Univerzita Karlova v Praze
Farmaceutická fakulta v Hradci Králové
Katedra biofyziky a fyzikální chemie

Kandidát: Savvas Athanasiadis
Školitel: Jurjen Duintjer Tebbens
Název diplomové práce: The small sample size problem in gene expression tasks

Práce se zabývá klasifikací genů do nádorových typů na základě jejich genových expresí. Počet proměnných (aminokyselin), které mají být zkoumány, je typicky velmi vysoký (v tisících), zatímco je drahé a časově náročné analyzovat velký počet genů; obvykle maximálně desítky z nich jsou k dispozici. Kombinace malého počtu vzorku s velkým počtem proměnných činí standardní metody statistické klasifikace nevhodnými.

Práce se zaměřuje na modifikaci klasické metody klasifikace, Fisherova lineární diskriminační analýza, pro případ, kde počet vzorků je menší než počet proměnných. Navrhuje zlepšenou strategii pro testování této modifikace a to metodou křížové validace typu leave-one-out. Pomocí aktualizace zúčastněných kovariančních matic s maticemi nízké hodností, lze dosáhnout řádové snížení výpočetních nákladů v metodě křížové validace. Požadavky na paměť jsou též sníženy.

Introduction

Cancer research is constantly one of the most prominent fields of modern medical research. The accurate prediction of tumor types is of crucial importance to medical science and can be of great profit to the pharmaceutical industry. It can contribute significantly to the evolution of rational pharmacotherapy, that is, maximizing the effect of pharmaceuticals while minimizing the risk on patients. Despite the colossal effort and investment in this field of research, there are many limitations and constraints regarding technological and many other sectors of this type of research.

Cancer classification techniques are often morphological and clinical based and have several drawbacks regarding their diagnostic ability. The huge genetic diversity and complexity within tumors requires a lot of effort, time and expensive equipment to map and analyze, and is moreover, threatening efforts to create personalized cancer treatments. However an abstract statistical approach can overcome some of these drawbacks by minimizing the expenses of laboratory equipment and in many cases limit the time consumption.

The purpose of this thesis is to improve on an efficient way of testing a certain popular type of statistical classification of gene expression signatures. The signatures belong to specific types of cancers, and are to be classified into specific diagnostic categories, using artificial neural networks. When working on gene expression tasks, one has to face problems such as the large variable number - small sample size problem. In such tasks it is hard to find sufficient gene expression data. The number of samples capable of working with is very small, while the number of variables of each gene much larger than the number of samples. From a mathematical point of view this leads to singular matrices and makes the whole process of classification much harder to deal with. In

order to confront these problems we use an intuitively reasonable modification of the process used in the usual situation when matrices are non-singular. It tries to make this rather demanding case of classification as efficient as possible.

This leads to further technological problems and constraints such as high computational and storage costs and limitations concerning stability of the numerical methods used. These problems can be time-consuming and lead to financial costs. In particular the testing phase is expensive. Therefore, we developed a fast and efficient way of testing the classification procedure, which will be thoroughly described in this thesis.

The first chapter concerns basic linear algebra and statistics. Mentioned are some fundamental definitions and theorems that will be essential in order to fully understand the concept and the mathematical methods used for classification, as well as a reference to some important numerical methods. The concept of eigenvalues and eigenvectors is of great importance and will be extensively described.

The second chapter of this thesis introduces the specific classification method we will consider, namely Fisher's linear discriminant analysis. We will demonstrate it on classification tasks like the popular Iris flower data set, introduced by Fisher too. The studied gene expression test is mentioned as well as validation techniques, like the leave-one-out cross-validation method (LOOCV) used on these tasks and their corresponding algorithms. Lastly methods of dealing with the the large variable number-small sample size problem are a significant part of the second chapter.

The third chapter describes an efficient way of using the leave-one-out cross validation method for testing the FLDA classification procedure, trying to minimize computational and storage costs and maximize its speed and efficacy. We created a pattern which, for every new iteration of the leave-one-out method uses information from the previous iteration. For this to be done we exploit the fact that the two iterations are close in the sense that the data matrices differ by a matrix of rank one. This can be used to cheaply update most of the matrices involved from one iteration to another. If n is the sample size and p, $p > n$, the number of variables, we show that in this way we can

avoid order pn^2 operations during the process. At the expense of a single order pn^2 matrix-matrix product at the beginning of the entire validation process, all iterations can be performed with order n^2 arithmetic operations except for the solution of an eigenproblem. The eigenproblems can be obtained with order $n^2 \log n$ costs instead of n^3 costs using small rank updates and multiplication with Cauchy-like matrices. Numerical experiments demonstrate the improved timings.

We conclude this introduction with the official, original scope of the thesis (before working on it was started).

The scope of the thesis

Modern gene expression tasks require the evaluation of a very high number of variables (at least in the thousands) whereas it is difficult or often impossible to gather the same number of samples. This so-called small sample size problem causes the underlying numerical methods to become hard to use. The involved matrices will be singular and some of the needed eigenvectors cannot even be defined properly. The purpose of the thesis is to develop new or to further develop existing strategies in order to cope with these difficulties. Any improvement is highly desirable in this very active area of research.

Chapter 1

Basic linear algebra and statistics

In this chapter we describe some of the basic facts of numerical linear algebra and statistics that we will need in the thesis. We used G. Strang's book [38] and Saad's book [37] as the main sources for information concerning numerical linear algebra, and Rencher's book [35] for information concerning statistics.

1.1 Basic matrix definitions

Definition 1.1.1 *A* matrix *is a rectangular array of numbers or other mathematical objects, for which operations such as addition and multiplication are defined. A* square *matrix is a matrix for which the horizontal and vertical dimensions are the same.*

Unless otherwise stated, we will use arrays of numbers.

Definition 1.1.2 *A* real matrix *is a matrix whose elements consist entirely of real numbers. The set of real matrices with n rows and p columns is denoted $\mathbb{R}^{n\times p}$.*

Definition 1.1.3 *A* complex matrix *is a matrix whose elements contain complex numbers. The set of complex matrices with n rows and p columns is denoted $\mathbb{C}^{n\times p}$.*

Definition 1.1.4 *The* transpose *of a matrix A is a matrix denoted as A^T, whose rows are the columns of the original, and whose columns are the rows of the original matrix A.*

Definition 1.1.5 *A* symmetric matrix *is a real square matrix that is equal to its transpose. Formally, the matrix A is symmetric if and only if*

$$A = A^T.$$

Definition 1.1.6 *The* conjugate transpose *of the $n \times p$ matrix A is the $p \times n$ matrix B with $B_{i,j} = \bar{A}_{j,i}$, where $B_{i,j}$ denotes the entry on position i, j. It is denoted A^*.*

Definition 1.1.7 *The entries of the* sum C of two matrices $A \in \mathbb{C}^{m \times n}$ *and $B \in \mathbb{C}^{n \times p}$ are defined as*

$$c_{ij} = a_{ij} + b_{ij},$$

where $i = 1, \ldots, m$ and $j = 1, \ldots, n$. In order for the matrix addition $A + B$ to be defined, the number of columns and rows of A must be equal to the number of columns and rows of B.

Definition 1.1.8 *The entries of the* product D of two matrices $A \in \mathbb{C}^{m \times n}$ *and $B \in \mathbb{C}^{n \times p}$ are defined as*

$$d_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk},$$

where $i = 1, \ldots, m$ and $k = 1, \ldots, p$. In order for the matrix multiplication AB to be defined, the number of columns of A must be equal to the number of rows of B.

Definition 1.1.9 *The* identity matrix *of size n is the $n \times n$ square matrix with ones on the main diagonal and zeros elsewhere. It is usually denoted as I or I_n.*

The identity matrix is the neutral element of matrix multiplication. The inverse element is defined as follows.

Definition 1.1.10 *The* inverse of a square matrix A *is a matrix* A^{-1} *such that*

$$A^{-1}A = AA^{-1} = I.$$

Not every square matrix has an inverse. But if it does, the inverse is unique.

Definition 1.1.11 *A square matrix* Q *is* unitary *if*

$$Q^*Q = I.$$

Thus if Q is unitary, its inverse is Q^*. Also, $Q^*Q = QQ^*$.

Definition 1.1.12 *An* orthogonal matrix *is a square matrix* Q *with real entries for which*

$$Q^TQ = I.$$

We remark that the columns of a unitary or orthogonal matrix have unit length because if q_j is the jth column, then

$$\|q_j\|^2 = {q_j}^*q_j = {e_j}^TQ^*Qe_j = {e_j}^TIe_j = 1,$$

where e_j denotes the jth column of the identity matrix I. Similarly, all columns are orthogonal to each other, i.e

$${q_i}^*q_j = 0, \qquad i \neq j.$$

Definition 1.1.13 *For any rectangular matrix* $A \in \mathbb{C}^{n\times p}$ *with* $n \geq p$ *there exists a factorization of the form*

$$A = QR,$$

called QR factorization*, where* $Q \in \mathbb{C}^{n\times p}$ *has orthogonal columns and* $R \in \mathbb{C}^{p\times p}$ *is a square upper triangular matrix.*

For the existence of the QR-factorization, see, e.g., [8, Theorem 10.5].

Definition 1.1.14 *The* rank *of a matrix* A *is the maximum number of independent rows, or independent columns of* A*. For every matrix, the column rank is equal to the row rank (see, e.g., [8, Theorem 1.27]).*

Definition 1.1.15 *A* singular matrix *is a square matrix which does not have an inverse.*

We will now use the concept of the determinant. For a definition and details, we refer to [8, Chapter 1].

Theorem 1.1.1 *A square matrix A of size p is singular if and only if its determinant is zero, if and only if the columns of the matrix A are linearly dependent.*

Proof: If A is singular, it is not invertible. I.e., there are two vectors c and d such that $Ac = Ad$ but $c \neq d$. Then $Av = 0$, with $v \equiv c - d$ where $v \neq 0$. This means that the entries of v are the coefficients of a linear combination of the columns of A giving the zero vector. Therefore the columns of A are linearly dependent.

Now, suppose $a_1, a_2, \ldots a_n$ is a linearly dependent set of vectors of a column of matrix A, then $\det(a_1, a_2, \ldots a_n) = 0$. If the vectors are linearly dependent then one of them can be written as a linear combination of the others. Without loss of generality, let us say, that vector is a_1. Then,

$$a_1 = c_2 a_2 + \ldots + c_n a_n,$$

for numbers $c_2, \ldots, c_n$ which are not all zero.

Then using the fact that the determinant is a linear function of one column when the others are held fixed, we have

$$\begin{aligned} \det(a_1, a_2, \ldots, a_n) &= \det(c_2 a_2 + \ldots + c_n a_n, a_2, a_n) \\ &= c_2 \det(a_2, a_2, \ldots, a_n) + c_3 \det(a_3, a_2, \ldots, a_n) + \ldots \\ &+ c_n \det(a_n, a_2, \ldots, a_n), \end{aligned}$$

where every term in the last expression is zero because $\det(a_1, a_2, \ldots a_n) = 0$, if $a_i = a_j$ for some $i \neq j$ [8, Theorem 1.9]. This condition says that if two edges of the parallelepiped are the same, then a parallelepiped is degenerate (i.e. "flat" in $\mathbb{R}^n$) and so should have volume zero.

Now, if the determinant is zero, then the matrix must be singular: Suppose that an inverse does exist. Then

$$\det(A^{-1}A) = \det(I) = 1.$$

But

$$\det(A^{-1}A) = \det(A^{-1})\det(A) = 0$$

[8, Theorem 1.1]. That means that there is no inverse and therefore, the matrix is singular. □

Definition 1.1.16 *Let A be a square matrix of size p and v a non-zero vector such that when multiplied by the matrix A, it yields a multiple of itself:*

$$Av = \lambda v, \tag{1.1}$$

where λ is the scalar multiplier. In every such case the vector v is an eigenvector *and λ is its* eigenvalue. *An* eigenspace *is the set of all eigenvectors with the same eigenvalue, together with the zero vector.*

In geometry, we can imagine the vector v as an arrow which does not change direction when multiplied by A. Its corresponding eigenvalue determines how its length changes, with the sign of the eigenvalue determining whether its direction is reversed or not.

1.2 Eigenvalues and eigenvectors

Let us re-write the equation (1.1) as

$$(A - \lambda I)v = 0.$$

In order for this equation to work for a non-zero vector v, $A - \lambda I$ must be singular. Suppose $A - \lambda I$ is non-singular: Then

$$\begin{aligned} (A - \lambda I)^{-1}(A - \lambda I)v &= (A - \lambda I)^{-1}0 \\ v &= 0 \end{aligned} \tag{1.2}$$

but we are talking about a non-zero vector. Thus, $A - \lambda I$ is singular, which means that

$$\det(A - \lambda I) = 0,$$

see theorem 1.1.1. This equation represents a polynomial in λ of degree p. It has at least one root, therefore every matrix always has at least one eigenvalue-eigenvector pair (eigenpair).

Remark 1 *Even if a matrix is real, the eigenvalues and the eigenvectors can be complex.*

Example: We take the matrix

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

To find the eigenvalues of the matrix A, we use $\det(A - \lambda I) = 0$, which leads to $\lambda^2 = -1$. That means that the eigenvalues of the matrix A are complex, namely $\lambda_1 = i$, $\lambda_2 = -i$.

Now, to find the eigenvectors for each λ use the equation

$$(A - \lambda I)x = 0.$$

For $\lambda = i$ it gives,

$$\begin{bmatrix} -i & 1 \\ -1 & -i \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

where $\begin{bmatrix} y \\ z \end{bmatrix}$ is the eigenvector. Now, using Gaussian elimination for row reduction we get

$$\begin{bmatrix} -i & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

This gives us

$$z = iy,$$

which means

$$E_i = \left\{ \begin{bmatrix} y \\ z \end{bmatrix} = t \begin{bmatrix} 1 \\ i \end{bmatrix}, t \in \mathbb{R} \right\}$$

where E_i is the eigenspace that corresponds to $\lambda = i$. The eigenspace corresponding to $\lambda = -i$ can be found analogously.

Definition 1.2.1 *Suppose the $n \times n$ matrix A has n linearly independent eigenvectors $s_1, \ldots, s_n$. We put them into columns of an eigenvector matrix S. Then*

$$A = S\Lambda S^{-1} = S \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} S^{-1},$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues, is the spectral decomposition *of A.*

Definition 1.2.2 *A* Hermitian matrix *is a square matrix A with complex entries that is equal to its own conjugate transpose. That is, the element in the i-th row and j-th column is equal to the complex conjugate of the element in the j-th row and i-th column,* $A = A^*$.

Theorem 1.2.1 *The eigenvalues of a Hermitian matrix are real.*

Proof: Let λ be the eigenvalue of A and u an associated eigenvector of 2-norm unity. Let (u, v) denote the inner product $(u, v) = v^*u$. Then

$$\lambda = \lambda(u, u) = (Au, u) = (u, A^*u) = (u, Au) = \overline{(Au, u)} = \overline{\lambda},$$

therefore λ must be real. □

Definition 1.2.3 *The square matrix A is* positive definite *if* $x^TAx > 0$ *for every non-zero vector x. The square matrix A is* positive semi-definite *if* $x^TAx \geq 0$ *for every non-zero vector x.*

Definition 1.2.4 *The* LU decomposition *of an* $n \times n$ *matrix A is a product of a lower triangular matrix L and an upper triangular matrix U, such that*

$$A = LU.$$

Definition 1.2.5 *The* Cholesky decomposition *is a decomposition of a Hermitian, positive-definite matrix A into the product of a lower triangular matrix and its conjugate transpose,*

$$A = LL^*,$$

where L is a lower triangular matrix with real and positive diagonal entries.

Theorem 1.2.2 *The eigenvalues of a symmetric, positive definite matrix are positive. The eigenvalues of a symmetric, positive semidefinite matrix are non-negative.*

Proof: We start from equation (1.1) , using it for an eigenvector x

$$Ax = \lambda x.$$

Multiplying both sides by x^T, we get

$$x^T Ax = \lambda x^T x.$$

But the left side is positive according to the definition 1.2.3. That means that $\lambda x^T x$ is also positive and so is the eigenvalue λ. The second claim, for a semidefinite matrix, follows analogously. □

Theorem 1.2.3 *Let $A \in \mathbb{R}^{n\times p}$. Then $A^T A$ is symmetric positive semi-definite. If A has full column rank, then $A^T A$ is symmetric positive definite.*

Proof:

$$(A^T A)^T = A^T (A^T)^T = A^T A,$$

which means that $A^T A$ is symmetric. Furthermore,

$$x^T A^T Ax = (xA)^T (Ax) = \|Ax\|^2 \geq 0,$$

for every nonzero vector x, which means that $A^T A$ is positive semi-definite according to definition 1.2.3. Ax can be zero, but not when A has a full column rank. In that case $A^T A$ is symmetric positive definite. □

Theorem 1.2.4 *Let $A \in \mathbb{R}^{p\times n}$ and $B \in \mathbb{R}^{n\times p}$, let $a_1, \ldots, a_n$ be the columns of A, and let $b^{(1)}, \ldots, b^{(n)}$ be the rows of B. Then $AB = \sum_{j=1}^{n} a_j b^{(j)}$.*

Proof:

$$\begin{aligned}
\sum_{j=1}^{n} a_j b^{(j)} &= a_1 b^{(1)} + \ldots + a_n b^{(n)} \\
&= \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1}b_{11} & a_{p1}b_{12} & \ldots & a_{p1}b_{1p} \end{bmatrix} \\
&+ \begin{bmatrix} a_{12}b_{21} & a_{12}b_{22} & \cdots & a_{12}b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p2}b_{21} & a_{p2}b_{22} & \ldots & a_{p2}b_{2p} \end{bmatrix} + \ldots \\
&+ \begin{bmatrix} a_{1n}b_{n1} & a_{1n}b_{n2} & \cdots & a_{1n}b_{np} \\ \vdots & \vdots & \ddots & \vdots \\ a_{pn}b_{n1} & a_{pn}b_{n2} & \ldots & a_{pn}b_{np} \end{bmatrix} = AB
\end{aligned}$$

because of definition (1.1.8). □

Example: If $x_1, \ldots, x_n \in \mathbb{R}^p$ are n vectors of dimension p and $X = [x_1, \ldots, x_n] \in \mathbb{R}^{p \times n}$, then $\sum_{i=1}^{n} x_i x_i^T = XX^T$.

1.3 The generalized eigenproblem

Definition 1.3.1 *A* generalized eigenproblem *is a problem of the form*

$$\alpha Ax = \beta Bx,$$

where A and B is a pair of given matrices, often referred to as a matrix pencil. We define a generalized eigenvalue *of a matrix pencil as a pair (α, β) of complex numbers, for which there is a vector u, called an* associated generalized eigenvector*, such that*

$$\beta Au = \alpha Bu.$$

In other words, (α, β) is an eigenvalue if and only if

$$\det(\beta A - \alpha B) = 0.$$

There is a set of difficulties arising from this definition of the eigenvalue. First, the pair $(0, 0)$ always satisfies the definition. Also there are many pairs (α, β) that can be termed as 'generalized eigenvalues' to represent the same 'standard eigenvalue' and have the same corresponding eigenvectors. There are three main ways to resolve this issue. The first is to denote with $< \alpha, \beta >$ the set of all pairs that satisfy the definition. A second way is to use pairs but scale them by some norm in $\mathbb{C}^2$ so that for example $\|\alpha\|^2 + \|\beta\|^2 = 1$. Finally, the third way is to take the ratio $\frac{\alpha}{\beta}$ as the eigenvalue. This is the definition we will use in this thesis.

Definition 1.3.2 *A* generalized eigenproblem *is a problem of the form*

$$Ax = \lambda Bx,$$

where A and B are given matrices. If λ and x satisfy the above equation, then λ is the generalized eigenvalue *and x is the* generalized eigenvector.

In the case of the matrix B being non-singular, we can transform the original problem into

$$B^{-1}Ax = \lambda x.$$

This action transforms the pencil (A, B) into $(B^{-1}A, I)$. In case A and B are symmetric and B is positive definite, we can use another alternative. If $B = LL^T$ is the Cholesky decomposition of B, then multiplying by L^{-1} from the left and by L^{-T} from the right we get the standard problem

$$L^{-1}AL^{-T}y = \lambda y.$$

The advantage of this transformation is that when we use it for symmetric matrices, the symmetry is preserved in contrast with the previous transformation.

1.4 Numerical methods

Definition 1.4.1 *A numerical method is a method to compute a mathematical problem on a computer.*

A numerical method's quality is mostly measured by 3 things:

1. Computational costs, i.e. the number of arithmetic operations (addition, subtraction, multiplication, division) needed to solve the underlying mathematical problem. Computational costs are often expressed in orders of magnitude. For example, Gaussian elimination is of the order p^3 where p is the size of the given square matrix. For details on Gaussian elimination, which is used for computing LU and Cholesky decompositions, we refer to [38, Chapter 2].

2. Storage costs, i.e. the amount of memory needed. Storage costs are also expressed in orders of magnitude. For example, Gaussian elimination has storage costs of the order p^2, where p is the size of the given square matrix.

3. Stability. A numerical method's stability concerns how much rounding errors (the computer needs to round most numbers) and other necessary

approximations (often a computer cannot find the exact mathematical solution because it uses only the basic arithmetic operations) have an influence on the accuracy of the result.

We will need the following numerical methods:

1. Gaussian elimination for LU and Cholesky decomposition: The computational and storage costs for this method are mentioned above. Regarding stability, it is conditionally stable, meaning that it is stable only, when the entries in the triangular matrix U stay of the same order of magnitude as those in the original matrix A.

2. The QR method for finding the eigenvalues and eigenvectors of a square matrix:

   Computational and storage costs are of the same order of magnitude as for Gaussian elimination. For symmetric matrices, an estimate for the computational costs to find both eigenvalues and eigenvectors, is $9n^3$ floating point operations [13, Section 8.3]. For symmetric matrices, the QR method is unconditionally stable.

3. The QZ method for finding the generalized eigenvalues and eigenvectors of a matrix pencil:

   Computational and storage costs of the QZ method are of the same order of magnitude as for Gaussian elimination. An estimate for the computational costs to find both generalized eigenvalues and eigenvectors, is $66n^3$ floating point operations [13, Section 7.7.7]. For symmetric positive definite matrices, they are unconditionally stable. They are unstable if the matrices of the pencil have a common nullspace.

1.5 Basic statistics definitions

Definition 1.5.1 *A* random variable*, usually written $\mathcal{X}$, is a variable whose possible values are numerical outcomes of a random phenomenon.*

We can distinguish two types of random variables, discrete and continuous. A discrete random variable can only take a finite or a countable infinite number of distinct values, while a continuous variable takes an uncountable infinite number of values (mostly in an interval or a union of intervals).

Definition 1.5.2 *The* expected value *of a discrete variable $\mathcal{X}$ is defined as*

$$E[\mathcal{X}] = x_1 p_1 + x_2 p_2 + \ldots + x_k p_k,$$

where $x_1, x_2, ..., x_k$ are the values $\mathcal{X}$ can take and $p_1, p_2, ..., p_k$ are the probabilities of the corresponding values. In the case of an infinite number of outcomes, the expected value becomes

$$E[\mathcal{X}] = \sum_{i=1}^{\infty} x_i p_i.$$

Definition 1.5.3 *A continuous random variable $\mathcal{X}$ is* normally distributed *if the probability that an outcome is smaller than a value t is given by the integral*

$$P(\mathcal{X} < t) = \int_{-\infty}^{t} \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)} dx.$$

A normal distribution in the variable $\mathcal{X}$ is fully characterized by its expected value

$$E(\mathcal{X}) = \mu$$

and variance

$$Var(\mathcal{X}) = E[(\mathcal{X} - \mu)^2] = \sigma^2.$$

The so-called standard normal distribution *is given by taking $\mu = 0$ and $Var(\mathcal{X}) = 1$ in a general normal distribution.*

Definition 1.5.4 *If $\mathcal{X} \in \mathbb{R}^p$ is a vector of p normally distributed random variables, then the* variance *of one single variable $\mathcal{X}_j$ is*

$$V(\mathcal{X}_j) = E[(\mathcal{X}_j - \mu_j)^2],$$

while the covariance *between* $\mathcal{X}_j$ *and another variable* $\mathcal{X}_k$ *is*

$$Cov(\mathcal{X}_j, \mathcal{X}_k) = E[(\mathcal{X}_j - \mu_j)(\mathcal{X}_k - \mu_k)],$$

where μ_j *and* μ_k *are the expected values of* $\mathcal{X}_j$ *and* $\mathcal{X}_k$ *respectively.*

Definition 1.5.5 *The population covariance matrix of the p-dimensional random vector* $\mathcal{X}$ *is*

$$\begin{bmatrix} E[(\mathcal{X}_1 - \mu_1)(\mathcal{X}_1 - \mu_1)] & E[(\mathcal{X}_1 - \mu_1)(\mathcal{X}_2 - \mu_2)] & \dots & E[(\mathcal{X}_1 - \mu_1)(\mathcal{X}_n - \mu_n)] \\ E[(\mathcal{X}_2 - \mu_2)(\mathcal{X}_1 - \mu_1)] & E[(\mathcal{X}_2 - \mu_2)(\mathcal{X}_2 - \mu_2)] & \dots & E[(\mathcal{X}_2 - \mu_2)(\mathcal{X}_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(\mathcal{X}_n - \mu_n)(\mathcal{X}_1 - \mu_1)] & E[(\mathcal{X}_n - \mu_n)(\mathcal{X}_2 - \mu_2)] & \dots & E[(\mathcal{X}_n - \mu_n)(\mathcal{X}_n - \mu_n)] \end{bmatrix}.$$

By the definition of the covariance, this matrix is symmetric.

Definition 1.5.6 *A sample of n data (realizations)* $x = [x_1, \dots, x_n]^T \in \mathbb{R}^n$ *from a normally distributed variable, has* sample mean

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n},$$

and sample standard deviation

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}}.$$

In the case where the sample is centered ($\overline{x} = 0$) then the standard deviation becomes

$$\frac{\|x\|}{\sqrt{n-1}},$$

where the numerator is the Euclidean norm of the sample vector x.

Definition 1.5.7 *Consider a sample of n data* $X = [x_1, \dots, x_n]^T \in \mathbb{R}^{n \times p}$ *for p variables, i.e* $x_i \in \mathbb{R}^p$ *for all i, and we denote with* $x_{(1)}, \dots, x_{(p)}$ *the columns of X. The* sample covariance *between the jth and the kth variable is*

$$s_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ij} - \overline{x}_{(j)})(x_{ik} - \overline{x}_{(k)}),$$

where $\overline{x}_{(j)}, \overline{x}_{(k)}$ *are the mean values of the corresponding row.*

In the case where the sample is centered the covariance becomes

$$\frac{{x_{(j)}}^T x_{(k)}}{n-1},$$

where the numerator is the inner product of $x_{(j)}$ and $x_{(k)}$.

Chapter 2

Fisher's Linear Discriminant Analysis

2.1 Classification

A class is a collection of sets or objects that can be defined by a property all its members share. *Classification* is the process of categorization of sets or objects into classes. Sometimes if classes are unknown, a method called cluster analysis can be used. Cluster analysis is the task of categorizing a set of objects in groups in such a way that objects in the same group (called a cluster) share more similar characteristics than those in other groups (clusters).

There are many different ways of classification, among them the ones described in the following lines.

K nearest neighbors (KNN) is a type of classification using a simple algorithm that classifies cases based on a similarity measure (e.g., distance functions). Classification is done by a majority vote of its neighbors, and the case is assigned to the class most common amongst its K nearest neighbors measured by a distance function [1]. If $K = 1$, then the case is simply assigned to the class of its nearest neighbor.

Linear discriminant analysis (LDA) is a method used to find a linear combination of features that characterizes or separates two or more classes of objects or events [35, Chapter 8]. LDA is useful in cases where reduction of dimensionality while preserving class discriminatory information, is essential.

In this chapter we focus on Fisher's LDA, which is a special formulation of LDA.

Quadratic discriminant analysis (QDA) is closely related to linear discriminant analysis (LDA), however, in QDA there is no assumption that the measurements from each class are normally distributed and the covariances of each of the classes are identical. A quadratic classifier is used to separate measurements of two or more classes of objects or events by a quadric surface. We used Cover's paper for information about this method [4].

Support vector machines (SVMs) build learning models with associated learning algorithms. Given a set of training examples, a SVM training algorithm builds a model that assigns the objects into their corresponding categories. A SVM model represents the objects as points in space and divides them in different classes by a gap as wide as possible. New examples are then mapped by prediction and categorized according to which side of the gap they fall on. We used Press's book for information about this method [33].

2.2 Cross-Validation

Cross-validation is often used to predict and estimate the accuracy of a model. Usually a model is given by a dataset of known data on which training is run, called the training dataset, and a dataset of unknown data against which the model is tested, called the testing dataset. Information was taken from [11].

Cross-validation is used:

1. To estimate performance of the learned model from available data using one algorithm. In other words, to estimate the generalizability of an algorithm.

2. To compare the performance of two or more different algorithms and find out the best algorithm for our data.

One round of cross-validation divides a sample of data into complementary subsets, performing the analysis on the training set, and validating the analysis on the testing set). In order to reduce variability, multiple rounds of

cross-validation are performed using different partitions, and the average of the validation results is found.

There are many types of cross-validation. One of them is called Leave-One-Out Cross-Validation. For information about this method we used [29]. Leave-one-out cross-validation (LOOCV) is a special case of k-fold cross-validation where k equals the number of instances in the data used for testing. In LOOCV, in each iteration nearly all the data except for a *single* observation are used for training and the model is tested on that single observation. An accuracy estimate obtained using LOOCV is known to be almost unbiased but it has high variance, leading to unreliable estimates. It is nevertheless widely used when the available data are very rare.

2.3 Classification using Fisher's linear discriminant analysis

Fisher's linear discriminant analysis (FLDA) is a method used in statistics to find a linear combination of features which characterizes or separates two or more classes of objects and events. The resulting combination can be used for dimensionality reduction before later classification. We used J. Duintjer Tebbens's and P. Schlesinger's paper [7] for the information in this section.

Consider a classification task with g groups, $g \geq 2$, and assume that n training objects (x_i, y_i) with $x_i \in \mathbb{R}^p$ and $y_i \in (1, \ldots, g)$ are available. Using the mean vector $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and denoting by N_j the index set of objects in group j, by n_j the size of the group j and by $\hat{x}_j = \frac{1}{n_j}\sum_{i \in N_j} x_i$ the corresponding group's mean vector, the between and within-group covariance matrix B and W, respectively, are defined by

$$B = \frac{1}{g-1}\sum_{j=1}^{g} n_j(\hat{x}_j - \overline{x})(\hat{x}_j - \overline{x})^T, \tag{2.1}$$

$$W = \frac{1}{n-g}\sum_{j=1}^{g}\sum_{i \in N_j}(x_i - \hat{x}_j)(x_i - \hat{x}_j)^T. \tag{2.2}$$

The rank of B is at most $\min(g-1, p)$, while the rank of W is at most $\min(n, p)$. If we assume that $p < n$, then *Fisher's criterion* is to find at most

$g-1$ transformation vectors c that have maximal separation ratio by solving the maximization problem

$$\max_{c\in\mathbb{R}^p, c\neq 0} \frac{c^T B c}{c^T W c}, \tag{2.3}$$

which can be translated to finding the largest eigenpairs of the generalized eigenproblem

$$(B-\lambda W)c = 0, \tag{2.4}$$

which in turn, can be transformed to a standard eigenproblem such as $(W^{-1}B-\lambda I)c = 0$. Then the FLDA-reduced space of dimension i, where $i < g$ is spanned by the eigenvectors corresponding to the i largest eigenvalues. They are ordered decreasingly according to the eigenvalues and are orthogonal to each other. Many applications just aim at dimension reduction and stop after mapping onto FLDA-reduced space. In the original classification process, the simplest and most frequent way to classify is by assigning to the group j of the transformed group mean vector $(c_1,\ldots,c_i)^T\hat{x}_j$ which is closest in the L_2-norm.

Theorem 2.3.1 *The matrices B and W are both symmetric positive semi-definite.*

Proof: According to the example after theorem 1.2.4,

$$B^T = \frac{1}{g-1}(XX^T)^T = \frac{1}{g-1}XX^T = B,$$

where $X = [\sqrt{n_1}(\hat{x}_1-\overline{x}),\ldots,\sqrt{n_g}(\hat{x}_g-\overline{x})]$. Thus B is symmetric. It is also positive semi-definite according to theorem 1.2.3. For the matrix W the claim follows analogously.

2.3.1 A small example where $n > p$

Let us illustrate the whole process with an academic example of a very simple form, where $n > p$. Suppose we have $n = 6$ samples with $p = 3$ variables, each belonging to some gene expression values. They belong to $g = 3$ groups of tumor type.

Let the samples be

$$x_1 = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} \quad x_2 = \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} \quad x_3 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \quad x_4 = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix} \quad x_5 = \begin{bmatrix} -2 \\ -2 \\ -3 \end{bmatrix} \quad x_6 = \begin{bmatrix} -2 \\ -2 \\ -5 \end{bmatrix}$$

Samples 1 and 2 belong to the first tumor type, samples 3 and 4 belong to the second tumor type and samples 5 and 6 belong to the third tumor type.

The corresponding total and class means are:

$$\overline{x} = \begin{bmatrix} 0 \\ \frac{1}{3} \\ -\frac{4}{3} \end{bmatrix}, \qquad \hat{x}_1 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \qquad \hat{x}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \qquad \hat{x}_3 = \begin{bmatrix} -2 \\ -2 \\ -4 \end{bmatrix}.$$

Then the between-covariance matrix is

$$\begin{aligned} B &= \frac{1}{g-1} \sum_{j=1}^{g} n_j (\hat{x}_j - \overline{x})(\hat{x}_j - \overline{x})^T, \\ &= \begin{bmatrix} 6 & 7 & 8 \\ 7 & 8.6667 & 10.3333 \\ 8 & 10.3333 & 12.6667 \end{bmatrix} \end{aligned}$$

and the within-covariance matrix is

$$\begin{aligned} W &= \frac{1}{n-g} \sum_{j=1}^{g} \sum_{i \in N_j} (x_i - \hat{x}_j)(x_i - \hat{x}_j)^T \\ &= \begin{bmatrix} 0.6667 & 0 & 0 \\ 0 & 0.6667 & 0 \\ 0 & 0 & 0.6667 \end{bmatrix}. \end{aligned}$$

We attempt to classify the test vector

$$t = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}.$$

To do that with FLDA we need to solve the generalized eigenproblem

$$(B - \lambda W)c = 0,$$

which can be transformed to the standard eigenproblem $(W^{-1}B - \lambda I)c = 0$ since the matrix W is not singular.

The resulting eigenvectors are:

$$c_1 = \begin{bmatrix} 0.457455 \\ 0.569992 \\ 0.682528 \end{bmatrix} \qquad c_2 = \begin{bmatrix} -0.789980 \\ -0.091886 \\ 0.606209 \end{bmatrix} \qquad c_3 = \begin{bmatrix} -0.408248 \\ 0.816497 \\ -0.408248 \end{bmatrix},$$

while the corresponding eigenvalues are:

$$\lambda_1 = 39.987 \qquad \lambda_2 = 1 \qquad \lambda_3 = 0.$$

We now project the data on the space spanned by the two leading eigenvectors, because they contribute to the separation (whereas $\lambda_3 = 0$ and the corresponding ratio is zero).

We calculated the projections P_1, P_2, P_3 of the class means onto the 2 largest eigenvectors c_1 and c_2 to be

$$P_1 = \begin{bmatrix} 2.27997 \\ -0.36754 \end{bmatrix} \qquad P_2 = \begin{bmatrix} 0.34492 \\ -1.48807 \end{bmatrix} \qquad P_3 = \begin{bmatrix} -4.78501 \\ -0.66110 \end{bmatrix}$$

respectively, and the projection of the test vector

$$P_t = \begin{bmatrix} 3.41995 \\ -0.55131 \end{bmatrix}.$$

Measuring the distances between each projection gives:

$$\|P_1 - P_t\| = 1.1547 \qquad \|P_2 - P_t\| = 3.2146 \qquad \|P_3 - P_t\| = 8.2057.$$

We find that the smallest one is the distance of P_t to P_1, which means that the test vector is assigned to the first group, which is what is to be expected by looking at the samples and t.

2.3.2 The Iris flower data set

Now let us consider a less academic example, namely a famous data set first studied by R.A Fisher [9].

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by Sir Ronald Fisher (1936) which can be used as an example for testing and performing discriminant analysis. The data set consists of 50 ($n = 50$) samples from each of three species of Iris: Iris Setosa (figure 2.2, Iris Versicolor (figure 2.3) and Iris Virginica (figure 2.4) (hence, we classify them into $g = 3$ groups). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters (hence, the number of variables is $p = 4$), see figure 2.1.

Based on Fisher's linear discriminant model, this data set became a typical test case for many classification techniques in machine learning such as support vector machines. The use of this data set in cluster analysis however is uncommon, since the data set only contains two clusters with rather obvious separation. One of the clusters contains Iris Setosa, while the other cluster contains both Iris Virginica and Iris Versicolor and is not separable without the species information Fisher used. This makes the data set a good example to explain the difference between supervised and unsupervised techniques in data mining: Fisher's linear discriminant model can only be obtained when the object species are known, class labels and clusters are not necessarily the same.

Performing the FLDA classification procedure described above and using the leave-one-out cross-validation method we found 98 percent success rate. More precisely, using the following pseudo-code:

Algorithm 2.3.1 *Testing FLDA with leave-one-out cross validation.*

For $j = 1$ *till* n*:*

1. *select the* j*th sample as test vector*

2. *compute* B *and* W *from the remaining* $n - 1$ *samples*

3. *assign the test vector to a class using the FLDA-projection vectors obtained from solving a generalized eigenproblem with* B *and* W

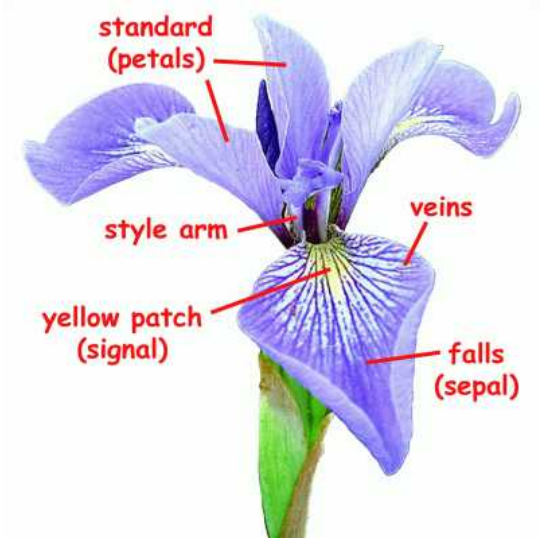

Figure 2.1: A figure of the general characterictics of the flowers, some of which are used as variables.

4. *if the assignment is correct, then* $success_j = 1$*, otherwise* $success_j = 0$
   end

and running this procedure, we found that

$$\frac{\sum_{j=1}^{n} success_j}{n} = 0.98.$$

Picture for figure 2.1 was taken from [23]. Pictures for the species of Iris Setosa were taken from [17] and [18]. Pictures for the species of Iris Vesicolor were taken from [19] and [20]. Pictures for the species of Iris Virginica were taken from [21] and [22].

Figure 2.2: Flowers from the species of Iris Setosa

Figure 2.3: Flowers from the species of Iris Versicolor.

Figure 2.4: Flowers from the species of Iris Virginica.

2.4 Fisher's criterion for the high dimension/small sample size problem

This section focuses on the use of Fisher's linear discriminant analysis when the number of variables largely exceeds the number of given samples, or the otherwise stated, "$p \gg n$" problem. In such case, classification based on FLDA is challenging and the original FLDA needs to be modified, and with high dimensionality implementation, issues like reduction of computational and storage costs and improving numerical stability are of crucial importance. For an overview of classification methods with high-dimensional data, we refer to [24], for an overview in the general LDA framework to [10],[25] and for

applications to microarray data to [14].

When $p > n$ the matrix W is singular. In that case the generalized eigenproblem cannot be transformed to a standard eigenproblem and that makes the process hard to perform. Furthermore, in cases where $p \gg n$ there are problems arising concerning storage and computational costs, which makes the problem even more challenging to solve.

When the covariance matrices are singular, the generalized eigenproblem is ill-posed. If equation (2.4) is satisfied, then there are eigenvectors c which satisfy the equation for some value λ. If c lies in the null space of B but not of W, then λ is a zero eigenvalue. On the other hand, if c lies in the null space of W but not of B, then we say λ is an infinite eigenvalue. If c does not lie in the null space of B and neither in the null space of W, then λ must be finite and nonzero. If c lies in the common null space of B and W, any value λ is an eigenvalue! The presence of a common null space will make solving the eigenproblem (2.4) very challenging. The QZ-algorithm may solve generalized eigenproblems with singular matrices but when it comes to a common null space it suffers from numerical instability [31]. B and W have a common nullspace as soon as $n + g - 1 < p$. Lastly, apart from all the difficulties mentioned, Fisher's criterion loses it's meaning to some extent when it comes to a singular covariance matrix W because the transformation vectors c in the nullspace of W would lead to division by zero in (2.3).

To deal with these problems many modifications have been proposed. A very simple and popular one is to regularize W by adding a small multiple of the identity matrix,

$$\hat{W} = W + \varepsilon I, \qquad \varepsilon > 0 \tag{2.5}$$

and to perform classical FLDA with B and $\hat{W}$. Approaches of this type have been called perturbation methods [3]. Other approaches include

1. Methods exploiting the Moore-Penrose pseudo-inverse implemented, e.g. in the R statistical programming language [36]

2. A method based on the GSVD (generalized singular value decomposition) [27]

3. The null space method [3]

4. An intuitively reasonable criterion [39]

The last two methods are more extensively described in the lines below.

2.4.1 The nullspace method

The nullspace method fully concentrates on the nullspace of W. This is reasonable because vectors in this nullspace do not contribute to separation. It modifies (2.4) as

$$\max_{c\in\mathbb{R}^p, Wc=0} c^T Bc.$$

This criterion leads to a standard eigenproblem in the nullspace of W.

2.4.2 An intuitively reasonable criterion

Here we start with the criterion from the null space method,

$$\max_{c\in\mathbb{R}^p, Wc=0} c^T Bc.$$

However, transformation vectors for which the maximum in this equation is zero are not interesting anymore; their between- group variance is minimal, hence they do not contribute to discrimination. Therefore, we first select only transformation vectors with nonzero between-group variance. If this does not yield enough (mutually orthogonal) transformation vectors, we leave the null space of W and select the next transformation vectors in the complement of the null space of W. Here of course, the ratio c^TBc/c^TWc is always finite and we can use the original criterion

$$\max_{c\in\mathbb{R}^p, Wc\neq 0} \frac{c^T Bc}{c^T Wc}.$$

2.4.3 Elimination of the common nullspace

Eliminating the common nullspace of the matrices B and W has many advantages in FLDA-based computations, such as improving stability, lowering

computational and storage costs and the avoidance of issues related to non-defined eigenvectors. The vectors c in the common nullspace do not contribute to discrimination because $c^T Bc = 0 = c^T Wc$. The common nullspace can be eliminated by considering the *total* covariance matrix

$$T = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(x_i - \overline{x})^T = \frac{1}{n-1}(X^T - \overline{x}\mathbf{1}_n{}^T)(X - \mathbf{1}_n\overline{x}^T),$$

where $X \in \mathbb{R}^{n\times p}$ is the sample matrix whose ith row contains the ith training object and $\mathbf{1}_n = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$. For the covariance matrices the following is true:

Theorem 2.4.1 *The common nullspace of B and W is the nullspace of T.*

Proof: [7, Lemma 1].

This means that the complement of the common nullspace of B and W is spanned by the eigenvectors for nonzero eigenvalues of the total covariance matrix. These eigenvectors can be computed inexpensively. If the total covariance matrix T has rank q where $q \leq n$, this preprocessing reduces the original p-dimensional problem to dimension q. And since $p \gg n$ the benefit is considerable.

The eigenvectors of T corresponding to nonzero eigenvalues can be computed with the following theorem.

Theorem 2.4.2 *Let $Z \in \mathbb{R}^{n\times p}$ with $n < p$, let the diagonal matrix D_1 contain the nonzero eigenvalues of $ZZ^T \in \mathbb{R}^{n\times n}$ and let the columns of V_1 contain the corresponding eigenvectors. Then the normalized eigenvectors for nonzero eigenvalues of $Z^T Z \in \mathbb{R}^{p\times p}$ are given by the columns of $Z^T V_1 D_1^{-\frac{1}{2}}$.*

Proof: [7, Lemma 2].

Using the above theorems, we come to a procedure which can be summarized in the following pseudo-code:

Algorithm 2.4.1 *FLDA for $p \gg n$ using the modified criterion in subsection 2.4.2 and elimination of the common nullspace.*

1. *Compute* $X - \mathbf{1}_n\overline{x}^T \in \mathbb{R}^{n\times p}$

2. *Compute* $T = (X - \mathbf{1}_n\overline{x}^T)(X - \mathbf{1}_n\overline{x}^T)^T \in \mathbb{R}^{n\times n}$

3. *Compute the eigenvectors* $v_1, \ldots, v_q$, $q \leq n$, *belonging to the nonzero eigenvalues* $d_1, \ldots, d_q$ *of* T

4. *Put* $w_i = \frac{(X-\mathbf{1}_n\overline{x}^T)v_i}{\sqrt{d_i}} \in \mathbb{R}^p, \quad i = 1, \ldots, q$

5. *Project the data onto the space spanned by* $w_1, \ldots, w_q$ *and perform FLDA with the modified criterion in subsection 2.4.2 in this space.*

2.5 Classification of cancers using gene expression profiling

The example with $p \gg n$ that we consider in detail in the thesis concerns classification of cancers to specific diagnostic categories depending on their gene expression signatures using artificial neural networks (ANNs). The ANNs were trained using the small, round blue-cell tumors (SRBCTs) that belong to four diagnostic categories (hence they are classified into $g = 4$ groups) as a model. The results of the classification indicates the potential applications that can be used for tumor diagnosis and the identification of candidate targets for therapy. We received information concerning the genes used in this section from [26].

The small, round blue-cell tumors of childhood consist of the four groups neuroblastoma (NB) (figure 2.5), rhabdomyosarcoma (RMS) (figure 2.6), non-Hodgkin lymphoma (NHL) (figure 2.7) and the Ewing family of tumors (EWS). These cancers are difficult to distinguish by light microscopy, or by a single test. Artificial neural networks (ANNs) are computer-based algorithms which are modeled on the structure and behavior of neurons in the human brain and can be trained to recognize and categorize complex patterns. This is accomplished by adjusting parameters of the ANN by a process that minimizes errors through learning from experience. Their calibration is achieved by using various types of input data, such as gene-expression levels generated by cDNA

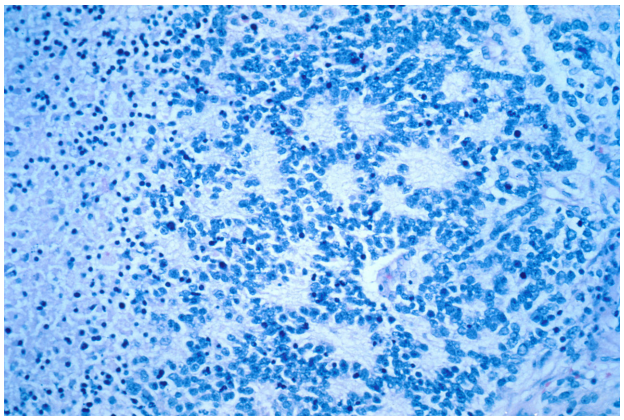

Figure 2.5: Microscopic view of a typical neuroblastoma with rosette formation.

Figure 2.6: Non-contrast computed tomography of head showing a large mass without any intracranial extension.

(complementary DNA) microarrays, and the output can be grouped into any given number of categories.

In this experiment cDNA microarrays containing 6567 genes initially were used. There are 63 ($n = 63$) training samples that include both tumor biopsy material and cell lines. For two particular samples, ST486 (BL-C2 and C4) and GICAN (NB-C2 and C7) that were treated as separate samples, two independent microarray experiments were performed to test the reproducibility of the experiments. By filtering for a minimal level of expression, the number of genes was markedly reduced. The genes were filtered by requiring that a gene should have red intensity greater than 20 across all experiments. The number of genes that passed this filter was 2308 (hence, $p = 2308$). Each slide was normalized across all experiments such that the relative (or normalized) red

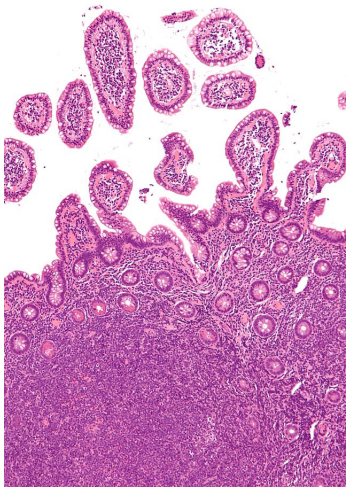

Figure 2.7: Micrograph of mantle cell lymphoma, a type of non-Hodgkin lymphoma.

intensity (RRI) for each gene was defined as: RRI = mean intensity of that spot/mean intensity of filtered genes. The natural logarithm (ln) of RRI was used as a measure of the expression levels.

Summarizing, we end up with a dataset of $n = 63$ samples containing $p = 2308$ genes belonging to $g = 4$ groups. We will use this training set to assess the performance of our classification method for $p \gg n$ based on FLDA. For completeness, we describe the classification procedure used in the original study in the next paragraph, see also figure 2.5, (a). All figures were taken from [26].

Hierarchical clustering and MDS (Multidimensional scaling) plots were performed. The diagnostic classification capabilities of these ANN models were tested on a set of 25 blinded test samples. A sample is classified to a diagnostic category if it receives the highest vote for that category and since the classifier has only four possible outputs, all samples will be classified to one of the four categories. This enables the rejection on a diagnosis of a sample classified to a given category. A sample's diagnosis is rejected if it falls outside the 95th percentile of the probability distribution of distances between samples and their ideal output (for example, for EWS it is EWS = 1, RMS = NB = BL = 0).

If we perform the FLDA classification procedure in section 2.4.2 and

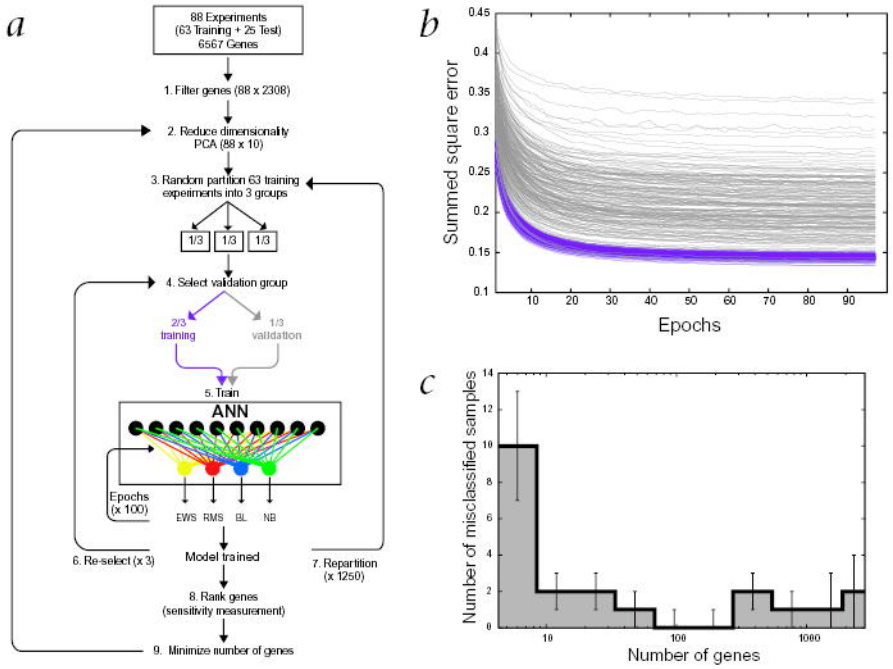

Figure 2.8: *a*: Schematic illustration of the analysis process. b: Monitoring the calibration of the models. A pair of lines, purple (training) and gray (validation), represents one model. The decrease in the classification errors with increasing epochs demonstrates the learning of the models to distinguish these cancers. c: Minimizing the number of genes. The average number of misclassified samples for all 3750 models is plotted against increasing number of used genes. The misclassifications minimized to zero using the 96 highest ranked genes.

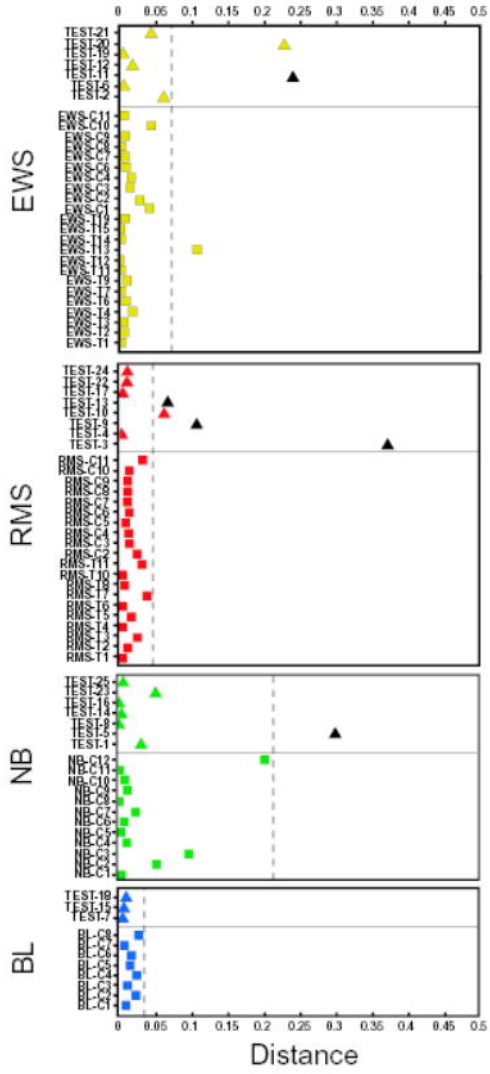

Figure 2.9: Classification and diagnosis of the samples.

use the leave-one-out cross-validation method using algorithm 2.4.1 we find a 100 percent success rate, indicating that for these data, the modified criterion in section 2.4.2 seems to be reasonable for classification purposes. We note however that the usage of PCA (i.e elimination of the null-space of T) before LDA has been criticized in some publications, see, e.g., [5].

Chapter 3

Efficient leave-one-out cross validation for FLDA with $p > n$

In this chapter we propose a cheap way to test Algorithm 2.4.1 (a modified FLDA method for the $p > n$ case) with leave-one-out cross validation (LOOCV). When using standard LOOCV, one runs Algorithm 2.4.1 n times in total, i.e. once for every observation that is left out as test vector. This means that every run of Algorithm 2.4.1 constructs a new set of training data, centers them, extracts appropriate eigenvectors from them for dimension reduction, etc... However, the training data of the individual runs will be very similar. In fact, they differ by one observation only, namely by the current observation that is left out. Our main idea to make LOOCV faster is to exploit the fact that the training data of two consecutive iterations of the LOOCV process are close by using information from the previous iteration for the current iteration.

3.1 The costs of Algorithm 2.4.1

Let us first investigate in detail the computational costs of Algorithm 2.4.1. We assume that we are not working with sparse matrices. Possible savings due to accidentally arising zero entries are incorporated by the fact that we give approximate operations counts. Computational costs are expressed by the number of floating point operations (flops). The multiplication of a matrix of

size $n \times p$ with a size p column vector requires n inner products of p-dimensional vectors. Every such inner product is computed by about p additions and p multiplications, hence the n inner products require ca. $2pn$ floating point operations in total.

The first two steps of Algorithm 2.4.1 are best performed in such a way that XX^T is computed explicitly, because this matrix can be used later in step 5. We compute T (i.e. the first two steps of Algorithm 2.4.1) by using the following:

$$\begin{aligned} T &= (X - \mathbf{1}_n \overline{x}^T)(X - \mathbf{1}_n \overline{x}^T)^T = (X - \mathbf{1}_n \frac{\mathbf{1}_n^T X}{n})(X - \mathbf{1}_n \frac{\mathbf{1}_n^T X}{n})^T \\ &= \left(I - \frac{\mathbf{1}_n \mathbf{1}_n^T}{n}\right) X \left(\left(I - \frac{\mathbf{1}_n \mathbf{1}_n^T}{n}\right) X\right)^T \\ &= \left(I - \frac{\mathbf{1}_n \mathbf{1}_n^T}{n}\right) XX^T \left(I - \frac{\mathbf{1}_n \mathbf{1}_n^T}{n}\right). \end{aligned} \tag{3.1}$$

Note that the multiplication of $XX^T \in \mathbb{R}^{n \times n}$ with $I - \frac{\mathbf{1}_n \mathbf{1}_n^T}{n}$ runs fully in $n-$dimensional spaces. It requires to multiply XX^T with the vector of ones ($n(n-1)$ flops), divide with n (n flops), put the resulting vector on n rows (no flops) and add the result to XX^T (n^2 flops). Thus we have for steps one and two:

$$\begin{array}{rrrl} & XX^T & \ldots & 2pn^2 \quad \text{flops} \\ \text{two multiplications with} & I - \frac{\mathbf{1}_n \mathbf{1}_n^T}{n} & \ldots & 4n^2 \quad \text{flops} \\ & \text{steps 1 and 2 in total ca.} & \ldots & 2(p+2)n^2 \quad \text{flops} \end{array}$$

In step 3, we solve a symmetric eigenproblem of size n. If we need both eigenvalues and eigenvectors, which we do, the computational costs using the standard (stable) QR algorithm are

$$\text{step 3 in total ca.} \quad \ldots \quad 9n^3 \quad \text{flops}$$

For later use, we denote the diagonal matrix containing the nonzero eigenvalues of T with D and the matrix with the corresponding eigenvectors with V.

Step 4 needs not be performed explicitly, but it can be incorporated into the computations of step 5.

3.1.1 Step five of Algorithm 2.4.1

Before we can address its computational costs, we have to describe in more detail the way step five of Algorithm 2.4.1 is implemented. We will use an implementation closely following the one proposed in [7, Algorithm 1]. In step five we have to:

- Compute the between- and within-covariance matrices in the projected space (we recall that the goal of the projection was to eliminate the common nullspace of these two matrices)
- Solve the corresponding generalized eigenproblem
- The modified Fisher criterion in subsection 2.4.2 next requires to select first the eigenvectors in the nullspace of the within-covariance matrix (they correspond to infinite generalized eigenvalues) and then those corresponding to finite generalized eigenvalues (i.e. in the nullspace of neither the between- nor the within-covariance matrix).
- Finally, assign to a class by comparing the distances of the test vector projected on the space generated by the selected generalized eigenvectors to the class means projected on the same space.

Let $\tilde{B}$ and $\tilde{W}$ denote the between- and within-covariance matrices projected onto the complement of the common nullspace of B and W, i.e.

$$\begin{aligned} \tilde{B} &= \left((X - \mathbf{1}_n\overline{x})^T V D^{-\frac{1}{2}}\right)^T B \left((X - \mathbf{1}_n\overline{x})^T V D^{-\frac{1}{2}}\right), \\ \tilde{W} &= \left((X - \mathbf{1}_n\overline{x})^T V D^{-\frac{1}{2}}\right)^T W \left((X - \mathbf{1}_n\overline{x})^T V D^{-\frac{1}{2}}\right), \end{aligned}$$

where we used Theorem 2.4.2. In [7, Section 3.3, Lemma 3] it was shown that the generalized eigenproblem in the projected space,

$$\tilde{B}c = \lambda \tilde{W} c$$

can be transformed to

$$\tilde{B}c = \lambda T c,$$

where $\tilde{T}$ denotes the (total) covariance matrix projected onto the complement of the nullspace of T. There are two important advantages in doing so: First,

$\tilde{W}$ needs not be computed and second, T is nothing but the diagonal matrix D [7, Section 3.3, Lemma 4]. Therefore, $\tilde{B}c = \lambda Tc$ can in turn be transformed to a standard eigenproblem with system matrix $D^{-\frac{1}{2}}\tilde{B}D^{-\frac{1}{2}}$ [7, Section 3.3, equation (23)].

To compute $\tilde{B}$ we use that

$$B = \frac{(N_g(G^TX - \mathbf{1}_g\overline{x}^T))^T N_g(G^TX - \mathbf{1}_g\overline{x}^T)}{g-1},$$

where $N_g \in \mathbb{R}^{g\times g}$ is the diagonal matrix containing on its diagonal the roots of the class sizes $n_1, \ldots, n_g$. Then

$$\tilde{B} = \tilde{B}_1^T\tilde{B}_1, \quad \tilde{B}_1 = \frac{N_g(G^TX - \mathbf{1}_g\overline{x}^T)(X - \mathbf{1}_n\overline{x})^TVD^{-\frac{1}{2}}}{\sqrt{g-1}} \in \mathbb{R}^{g\times n}, \tag{3.2}$$

and the only matrix that needs to be computed is $\tilde{B}_1$.

An elegant way to extract the generalized eigenvalues corresponding to the modified criterion of subsection 2.4.2 in the complement of the common nullspace of B and W (i.e. the eigenvectors of $D^{-\frac{1}{2}}\tilde{B}D^{-\frac{1}{2}}$) is described in [7, Section 3.3]. As all computations to extract these eigenvectors (the FLDA transformation vectors in the complement of the common nullspace) live in n- and g-dimensional spaces, we omit the details. In particular, no p-dimensional computations are involved in this phase and no order n^3 computations either (the eigenproblem for $D^{-\frac{1}{2}}\tilde{B}D^{-\frac{1}{2}}$ can be solved efficiently using the SVD of $\tilde{B}_1D^{-\frac{1}{2}}$). Let us store the obtained FLDA transformation vectors in a matrix $E \in \mathbb{R}^{n\times g-1}$.

To perform the final class assignment, we need to project the class means and the test vector first onto the complement of the common nullspace of B and W and then onto the FLDA transformation vectors stored in E. Hence, the resulting projection is onto the space spanned by the columns of

$$(X - \mathbf{1}_n\overline{x}^T)^TVD^{-\frac{1}{2}}E. \tag{3.3}$$

More precisely, we compute the projected class means matrix $\tilde{\tilde{M}}$ and test vector $\tilde{\tilde{x}}_t$,

$$\tilde{\tilde{M}} = N_g^{-2}G^TX(X - \mathbf{1}_n\overline{x}^T)^TVD^{-\frac{1}{2}}E \in \mathbb{R}^{g\times n}, \tag{3.4}$$

$$\tilde{\tilde{x}}_t^T = x_t^T(X - \mathbf{1}_n\overline{x}^T)^TVD^{-\frac{1}{2}}E \in \mathbb{R}^{1\times n}, \tag{3.5}$$

and assign to the projected class mean (i.e. the row of $\tilde{\tilde{M}}$) which is closest to $\tilde{\tilde{x}}_t$ in the Euclidean norm.

Let us now summarize the main computational costs of step five of Algorithm 2.4.1. To compute $\tilde{B}_1$ in (3.2), we can use (leaving aside the division with $\sqrt{g-1}$ which does not influence the eigenproblem to be solved with $D^{-\frac{1}{2}}\tilde{B}_1^T\tilde{B}_1D^{-\frac{1}{2}}$)

$$\tilde{B}_1 = N_g G^T[X(X-\mathbf{1}_n\overline{x}^T)^T]VD^{-\frac{1}{2}} - N_g\mathbf{1}_g[\overline{x}^T(X-\mathbf{1}_n\overline{x}^T)^T]VD^{-\frac{1}{2}}, \tag{3.6}$$

where only the factors between the [ ]-type brackets involve p-dimensional computations. But

$$X(X-\mathbf{1}_n\overline{x}^T)^T = XX^T - X\frac{X^T\mathbf{1}_n}{n}\mathbf{1}_n^T = XX^T\left(I - \frac{\mathbf{1}_n\mathbf{1}_n^T}{n}\right)$$

and XX^T having been computed in step 1 and 2 (see (3.1)) we can compute the first term in (3.6) as

$$\left(\left(\left(N_gG^TXX^T\right)_1\left(I-\frac{\mathbf{1}_n\mathbf{1}_n^T}{n}\right)\right)_2 VD^{-\frac{1}{2}}\right)_3,$$

where the lower indices denote the order of multiplication. With this order, all three multiplications require ca. $2gn^2$ floating point operations. Similarly, for the second term in (3.6), we compute

$$\left(\left(\left(N_g\mathbf{1}_g\frac{\mathbf{1}_n^TX}{n}X^T\right)_1\left(I-\frac{\mathbf{1}_n\mathbf{1}_n^T}{n}\right)\right)_2 VD^{-\frac{1}{2}}\right)_3$$

in $2gn^2$ flops.

For the final projection space (3.3) we cannot exploit the fact that XX^T was computed earlier. Here we need to multiply the $p\times n$ matrix $(X-\mathbf{1}_n\overline{x}^T)^T$ with an $n\times(g-1)$ matrix $(VD^{-\frac{1}{2}}E)$. This requires $2pn(g-1)$ flops. The class assignment, finally, requires to multiply a $g\times p$ matrix $(N_g^{-2}G^TX)$ with a $p\times(g-1)$ matrix for $\tilde{\tilde{M}}$ and a $1\times p$ matrix (x_t^T) with a $p\times(g-1)$ matrix for $\tilde{\tilde{x}}_t$. These costs are of about $2(g-1)(g+1)p$ flops. As we assume that $p\gg n$ and naturally $n\gg g$, the dominant costs of the entire step five are

$$\text{step 5 in total ca.} \quad \ldots \quad 2pn(g-1) \quad \text{flops.}$$

3.2 The difference between the training data of two consecutive iterations of LOOCV

The previous section showed that the main computational bottlenecks of Algorithm 2.4.1 are steps one and two (order pn^2 flops) and also step 3 (order n^3 flops). If we test this classification algorithm using LOOCV, we run it in total n times, hence the total costs are of order pn^3+n^4, which will be prohibitive in many applications. For example, a realistic situation in gene expression tasks is say $p = 1000$ and $n = 40$. If we assume for simplicity that we need exactly $pn^3 + n^4$ flops, then the total flop count for the LOOCV testing process is $1000(40)^3 + 40^4 = 66,56 \cdot 10^6$ flops. However, if we *double* the sample size to 80, then the flop count will be $1000(80)^3 + 80^4 = 552,96 \cdot 10^6$ flops, that is about *nine* times as much.

In this section, we present the main tools for lowering the costs of steps 1,2, and 3 of Algorithm 2.4.1 when testing it with LOOCV. As mentioned, the main idea is to exploit the fact that the training data of two consecutive iterations of the LOOCV process are very similar.

Let us denote the training data matrix in the jth iteration of the LOOCV process, where we have left out the jth observation, by $X_j \in \mathbb{R}^{(n-1)\times p}$ and let us denote the corresponding centered data matrix with $X_j^c \in \mathbb{R}^{(n-1)\times p}$. Similarly, $\overline{x}_j$ and $\overline{x}_{j-1}$ denote the overall mean of the training data in the jth and $(j-1)$st iteration, respectively. To exploit information from the $(j-1)$st iteration we will use the following relations.

Lemma 3.2.1 *Let*

$$d_j = x_j - x_{j-1} \in \mathbb{R}^p$$

denote the difference between the jth and the $(j-1)$st observation, let e_j denote the jth unit vector and let

$$f_j = e_{j-1} - \frac{\mathbf{1}_{n-1}}{n-1} \in \mathbb{R}^{n-1}.$$

Then

$$X_j = X_{j-1} - e_{j-1}d_j^T, \qquad \overline{x}_j = \overline{x}_{j-1} - \frac{d_j}{n-1}, \qquad X_j^c = X_{j-1}^c - f_j d_j^T.$$

Proof: We can write $X_j = [x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n]^T$ where the x_i are the individual observations and similarly $X_{j-1} = [x_1, \ldots, x_{j-2}, x_j, \ldots, x_n]^T$. Hence, X_j and X_{j-1} differ only in their $(j-1)$st row and the difference is that X_j contains in that row x_{j-1} while X_{j-1} contains x_j in that row. This can be expressed as

$$X_j = X_{j-1} - e_{j-1}(x_j - x_{j-1})^T,$$

which gives the first equality of the claim. Furthermore,

$$\overline{x}_j = \frac{X_j^T \mathbf{1}_{n-1}}{n-1} = \frac{(X_{j-1}^T - d_j e_{j-1}^T)\mathbf{1}_{n-1}}{n-1} = \overline{x}_{j-1} - \frac{d_j}{n-1}.$$

Finally,

$$\begin{aligned} X_j^c &= X_j - \mathbf{1}_{n-1}\overline{x}_j^T = X_{j-1} - e_{j-1}d_j^T - \mathbf{1}_{n-1}(\overline{x}_{j-1} - \frac{d_j}{n-1})^T \\ &= X_{j-1} - \mathbf{1}_{n-1}\overline{x}_{j-1}^T - (e_{j-1} - \frac{\mathbf{1}_{n-1}}{n-1})d_j^T = X_{j-1}^c - f_j d_j^T. \end{aligned}$$

□

Thus the data matrix and the centered data matrix of one iteration in LOOCV can be obtained from respectively the data matrix and the centered data matrix of the previous iteration with a modification matrix of rank one. Such matrices are cheap to store (its suffices to store the two vectors which define them instead of the whole matrix) and to work with, as we will see.

In steps one and two of Algorithm 2.4.1 we need to compute the matrix $X_j^c(X_j^c)^T$ if we are at the jth iteration of the LOOCV process. In step three we need its spectral decomposition. The matrix $X_j^c(X_j^c)^T$ can be obtained cheaply from $X_{j-1}^c(X_{j-1}^c)^T$ using the following theorem, which is the basis for the improvement we propose.

Theorem 3.2.2 *With the notation used in Lemma 3.2.1, there holds*

$$X_j^c(X_j^c)^T = X_{j-1}^c(X_{j-1}^c)^T - f_j(X_{j-1}^c d_j)^T - (X_{j-1}^c d_j)f_j^T + \|d_j\|^2 f_j f_j^T.$$

Proof: We have, with Lemma 3.2.1,

$$\begin{aligned} X_j^c(X_j^c)^T &= (X_{j-1}^c - f_j d_j^T)(X_{j-1}^c - f_j d_j^T)^T \\ &= X_{j-1}^c(X_{j-1}^c)^T - f_j(X_{j-1}^c d_j)^T - (X_{j-1}^c d_j)f_j^T + f_j d_j^T d_j f_j^T. \end{aligned}$$

□

We see that $X_j^c(X_j^c)^T$ and $X_{j-1}^c(X_{j-1}^c)^T$ differ by a symmetric matrix of rank three. Please note that the rank-two matrix

$$-f_j(X_{j-1}^c d_j)^T - (X_{j-1}^c d_j)f_j^T$$

is symmetric. With the notation $U_j = [f_j, X_{j-1}^c d_j]$ it can also be written as

$$-f_j(X_{j-1}^c d_j)^T - (X_{j-1}^c d_j)f_j^T = -U_j U_j^T.$$

From Lemma 3.2.1 we also obtain an updating formula for $X_j(X_j^c)^T$:

Corollary 3.2.3 *With the notation used in Lemma 3.2.1, there holds*

$$X_j(X_j^c)^T = X_{j-1}(X_{j-1}^c)^T - e_{j-1}(X_{j-1}^c d_j)^T - (X_{j-1}d_j)f_j^T + \|d_j\|^2 e_{j-1}f_j^T.$$

Proof: We have, with Lemma 3.2.1,

$$\begin{aligned} X_j(X_j^c)^T &= (X_{j-1} - e_{j-1}d_j^T)(X_{j-1}^c - f_j d_j^T)^T \\ &= X_{j-1}(X_{j-1}^c)^T - e_{j-1}(X_{j-1}^c d_j)^T - (X_{j-1}d_j)f_j^T + e_{j-1}d_j^T d_j f_j^T. \end{aligned}$$

□

This update is not symmetric (and neither is $X_j(X_j^c)^T$).

3.3 Eliminating order pn^3 costs

We now describe how the results of the previous section can be used to reduce the order pn^3 costs arising when testing Algorithm 2.4.1 with LOOCV. More precisely, we here focus on improving steps 1 and 2 of the algorithm, but we will also improve step 5. The order n^4 costs of step 3 will be addressed in the next section.

With Theorem 3.2.2, the current (total) covariance matrix $T_j = X_j^c(X_j^c)^T$ of the jth iteration of the LOOCV process can be computed as

$$T_j = X_{j-1}^c(X_{j-1}^c)^T - f_j(X_{j-1}^c d_j)^T - (X_{j-1}^c d_j)f_j^T + \|d_j\|^2 f_j f_j^T. \tag{3.7}$$

This requires, with $T_{j-1} = X_{j-1}^c(X_{j-1}^c)^T$ available from the previous LOOCV iteration, to

1. compute $X_{j-1}^c d_j$, $\|d_j\|^2$ and f_j

2. compute the three rank-one matrices in (3.7)

3. add them to T_{j-1}.

Clearly, the last two items can be done both in order $(n-1)^2$ flops; there are $p-$dimensional computations in the first item only. To avoid these $p-$dimensional computations at every iteration of the LOOCV process, we propose to compute *once*, before LOOCV is launched, the matrix

$$S = XX^T \in \mathbb{R}^{n\times n}$$

containing *all* observations, at about $2pn^2$ costs. Then

$$\begin{aligned} X_{j-1}^c d_j &= \left(X_{j-1} - \mathbf{1}_{n-1}\overline{x}^T\right)(x_j - x_{j-1}) \\ &= \left(X_{j-1} - \mathbf{1}_{n-1}\frac{\mathbf{1}_{n-1}^T X_{j-1}}{n-1}\right)(x_j - x_{j-1}) \\ &= \left(I - \mathbf{1}_{n-1}\frac{\mathbf{1}_{n-1}^T}{n-1}\right) X_{j-1}(x_j - x_{j-1}) \\ &= \left(I - \mathbf{1}_{n-1}\frac{\mathbf{1}_{n-1}^T}{n-1}\right)(S_{j-1}e_j - S_{j-1}e_{j-1}), \end{aligned}$$

where $S_j \in \mathbb{R}^{(n-1)\times n}$ denotes S without its jth row. Hence, $X_{j-1}^c d_j$ can in every iteration be obtained with about $2n$ flops. For $\|d_j\|^2$ we have

$$\|d_j\|^2 = (x_j - x_{j-1})^T(x_j - x_{j-1}) = S_{j,j} + S_{j-1,j-1} - 2S_{j,j-1}.$$

We see that steps 1 and 2 of Algorithm 2.4.1 can be computed, in every iteration of LOOCV, with order $(n-1)^2$ operations instead of order $p(n-1)^2$ flops. This is a significant improvement; the dependency on p (p is assumed to be much larger than n) has fully disappeared.

The same independence can be achieved for step 5: A look at (3.6) and (3.4), which are the cost dominating operations of the step, reveals that both contain the matrix

$$X(X - \mathbf{1}_n\overline{x}^T)^T.$$

To form this matrix would imply performing $p-$dimensional computations, but fortunately this matrix can also be updated from the previous LOOCV

iterations, see Corollary 3.2.3. The update formula is

$$X_j(X_j^c)^T = X_{j-1}(X_{j-1}^c)^T - e_{j-1}(X_{j-1}^c d_j)^T - (X_{j-1}d_j)f_j^T + \|d_j\|^2 e_{j-1}f_j^T$$

and requires basically the same operations as the update formula (3.7) we just discussed. The only difference is that we need $X_{j-1}d_j$ instead of $X_{j-1}^c d_j$. But it is clear from the above that $X_{j-1}d_j$ equals simply $S_{j-1}e_j - S_{j-1}e_{j-1}$. Hence again, we are able to avoid all $p-$dimensional operations and in every iteration of the LOOCV process, steps 4 and 5 of Algorithm 2.4.1 require costs of order $(n-1)^2$ only.

We remark that during the LOOCV process, the group coding matrix does not remain constant. In fact, it can also be updated similarly to the matrices in Lemma 3.2.1 with a matrix of rank one. We do not discuss the details as they do not significantly influence computational costs.

3.4 Eliminating order n^4 costs

The improvements of the previous section are of an order of magnitude, from pn^3 to pn^2. If $p \gg n$, the order n^4 costs created by running step 3 n times are often negligible compared to pn^3 or even pn^2. However, when $p > n$ but not $p \gg n$, it may be beneficial to reduce the order n^4 costs as well.

3.4.1 A diagonal-plus-small-rank eigenproblem

In step three of Algorithm 2.4.1, we can solve the eigenproblem with computational costs of order $(n-1)^2\log(n-1)$ instead of $(n-1)^3$ as follows. Assume the spectral decomposition of T_{j-1} is

$$T_{j-1} = V_{j-1}D_{j-1}V_{j-1}^T$$

with a diagonal eigenvalue matrix D_{j-1} and an orthogonal eigenvector matrix V_{j-1}. Then, using again Theorem 3.2.2,

$$\begin{aligned} T_j &= T_{j-1} - U_jU_j^T + \|d_j\|^2 f_jf_j^T \\ &= V_{j-1}D_{j-1}V_{j-1}^T - U_jU_j^T + \|d_j\|^2 f_jf_j^T \\ &= V_{j-1}\left[D_{j-1} - V_{j-1}^TU_jU_j^TV_{j-1} + \|d_j\|^2 V_{j-1}^Tf_j(V_{j-1}^Tf_j)^T\right]V_{j-1}^T. \end{aligned}$$

If we can compute in a cheap way the spectral decomposition

$$D_{j-1} - V_{j-1}^T U_j U_j^T V_{j-1} + \|d_j\|^2 V_{j-1}^T f_j (V_{j-1}^T f_j)^T = V_j D_j V_j^T, \tag{3.8}$$

with a diagonal eigenvalue matrix D_j and an orthogonal eigenvector matrix V_j, then it follows that

$$T_j = V_{j-1} V_j D_j V_j^T V_{j-1}^T \tag{3.9}$$

is the spectral decomposition of T_j needed at the jth iteration of the LOOCV process.

3.4.2 Eigenvalues

The spectral decomposition (3.8) can be obtained with computational costs of order $(n-1)^2$ because the matrix to decompose is of the form diagonal plus small rank. For a proof of this fact for the simplest case where the small rank matrix is a symmetric rank-one matrix we refer to [13, Section 8.4.3]. For the general case, a detailed proof is given in the paper [2]. We will here only present the main results.

Theorem 3.4.1 *The eigenvalues of a symmetric rank-k updated diagonal matrix* $\hat{D} + \hat{U}_k \hat{U}_k^T$, $\hat{U}_k \in \mathbb{R}^{(n-1)\times k}$, *are the zeros of the function*

$$f(\xi) = \det\left(I_k - \hat{U}_k^T (\xi I - \hat{D})^{-1} \hat{U}_k\right).$$

The function $f(\xi)$ in the theorem is the so-called modified Weinstein determinant. The computation of its zeros can be done with a function minimizer like Newton's method; the main costs are in multiplication with $(\xi I - \hat{D})^{-1}$, which is a diagonal matrix of order $n-1$. Thus the costs to obtain all $n-1$ zeros are of order $(n-1)^2$.

3.4.3 Eigenvectors

For eigenvectors, the situation is a little more complicated. For simplicity, we will assume that the original diagonal matrix and its small rank update have

no common eigenvalues. If ξ is an eigenvalue, then the modified Weinstein determinant is zero for that ξ and consequently, the matrix

$$I_k - \hat{U}_k^T(\xi I - \hat{D})^{-1}\hat{U}_k \tag{3.10}$$

is singular. The non-empty nullspace of this matrix can be used to obtain the corresponding eigenvector.

Theorem 3.4.2 *Let ξ be an eigenvalue of a symmetric rank-k updated diagonal matrix $\hat{D} + \hat{U}_k\hat{U}_k^T$, $\hat{U}_k \in \mathbb{R}^{(n-1)\times k}$, distinct from all diagonal entries of $\hat{D}$. Moreover, let $y_k \in \mathbb{R}^k$ be a nonzero vector in the nullspace of (3.10). Then*

$$(\xi I - \hat{D})^{-1}\hat{U}_k y_k$$

is an eigenvector corresponding to the eigenvalue ξ.

Consequently, every eigenvector can be obtained with the diagonal scaling of a linear combination of k vectors of size $(n-1)$. The costs are of order $(n-1)$, and to compute all eigenvectors we need $(n-1)^2$ flops. To have an orthogonal matrix, the eigenvectors should in addition be normalized, requiring just another $(n-1)^2$ flops.

Our assumption that the original diagonal matrix and its small rank update have no common eigenvalues was always satisfied in the experiments, except for one common zero eigenvalue. In fact, for all matrices $T_j = X_j^c(X_j^c)^T$ in the LOOCV process there holds

$$T_j = \left(I - \frac{\mathbf{1}_{n-1}\mathbf{1}_{n-1}^T}{n-1}\right) X_j X_j^T \left(I - \frac{\mathbf{1}_{n-1}\mathbf{1}_{n-1}^T}{n-1}\right),$$

see (3.1), hence

$$T_j\mathbf{1}_{n-1} = 0\,\mathbf{1}_{n-1},$$

showing that $\mathbf{1}_{n-1}$ is always an eigenvector for the eigenvalue zero. Fortunately, this eigenvector is not needed to perform the elimination of the common nullspace (see Theorem 2.4.2).

Summarizing, both eigenvalues and eigenvectors of the diagonal plus rank-three matrix in (3.8) can be computed in order $(n-1)^2$ flops. However, if we need the eigenvectors of $X_j^c(X_j^c)^T$, which we do, they are obtained

as the product of the two orthogonal matrices V_{j-1} and V_j in (3.9). This multiplication requires about $2(n-1)^3$ flops in general, making all improvements so far useless.

3.4.4 Fast multiplication of eigenvector matrices

To perform fast multiplication of the eigenvector matrices V_{j-1} and V_j in (3.9) we exploit the special structure of the eigenvectors described in Theorem 3.4.2. This will lead us to so-called Cauchy and Cauchy-like matrices, which are examples of matrices with a small displacement rank. To introduce these concepts as clearly as possible, we start with the situation where a diagonal matrix is updated with a symmetric rank-one update. After that, we describe rank-two updates. In fact, in an implementation we propose to use precisely this order: First we compute the eigenvectors of

$$D_{j-1} + \|d_j\|^2 V_{j-1}^T f_j (V_{j-1}^T f_j)^T \tag{3.11}$$

in (3.8) and then of its rank-two update

$$\left(D_{j-1} + \|d_j\|^2 V_{j-1}^T f_j (V_{j-1}^T f_j)^T\right) - V_{j-1}^T U_j U_j^T V_{j-1}. \tag{3.12}$$

Let us introduce the notation $r = \|d_j\| V_{j-1}^T f_j$, let us denote the eigenvalues of

$$D_{j-1} + \|d_j\|^2 V_{j-1}^T f_j (V_{j-1}^T f_j)^T = D_{j-1} + rr^T$$

with $\xi_1, \ldots, \xi_{n-1}$ and let Ξ denote the diagonal matrix containing these eigenvalues. Let the nonzero entries of the diagonal matrix D_{j-1} be denoted as $d_1, \ldots, d_{n-1}$.

Lemma 3.4.3 *The eigenvector matrix of (3.11) whose kth column is*

$$(\xi_k I - D_{j-1})^{-1} r$$

can be written as

$$\mathrm{diag}(r) C(D_{j-1}, \Xi), \tag{3.13}$$

where $\mathrm{diag}(r)$ *is the diagonal matrix whose diagonal entries are the entries of the vector* r *and* $C(D_{j-1}, \Xi)$ *is the Cauchy matrix with entry on position* i, j

defined as

$$C(D_{j-1}, \Xi)_{i,j} = \frac{1}{\xi_j - d_i}.$$

Proof: The kth column of the eigenvector matrix, $(\xi_k I - D_{j-1})^{-1} r$, can be written as

$$(\xi_k I - D_{j-1})^{-1} r = \begin{pmatrix} \frac{r_1}{\xi_k - d_1} \\ \vdots \\ \frac{r_n}{\xi_k - d_{n-1}}, \end{pmatrix}$$

hence the entire eigenvector matrix is

$$\begin{pmatrix} \frac{r_1}{\xi_1 - d_1} & \cdots & \frac{r_1}{\xi_{n-1} - d_1} \\ \vdots & & \vdots \\ \frac{r_{n-1}}{\xi_1 - d_{n-1}} & \cdots & \frac{r_{n-1}}{\xi_{n-1} - d_{n-1}}, \end{pmatrix}$$

which is the Cauchy matrix multiplied with $\mathrm{diag}(r)$ from the left. □

We remark that in order for (3.13) to be an orthogonal matrix, its columns need to be normalized.

For the rank-two update (3.12), let us introduce the notation $[s_1, s_2] = V_{j-1}^T U_j \in \mathbb{R}^{(n-1)\times 2}$, let us denote the eigenvalues of

$$D_{j-1} + rr^T - [s_1, s_2][s_1, s_2]^T$$

with $\lambda_1, \ldots, \lambda_{n-1}$ and let Λ denote the diagonal matrix containing these eigenvalues. Also, let for λ_k the vector $y^{(k)} \in \mathbb{R}^2$ denote a vector in the non-trivial nullspace of

$$I_2 - [s_1, s_2]^T (\lambda_k I - \Xi)^{-1} [s_1, s_2].$$

Lemma 3.4.4 *The eigenvector matrix of (3.12) whose kth column is*

$$(\lambda_k I - \Xi)^{-1} [s_1, s_2] y^{(k)}$$

can be written as

$$\mathrm{diag}(s_1) C(\Xi, \Lambda) \mathrm{diag}(y_1 + y_2)$$

where $\mathrm{diag}(y_1)$ *is the diagonal matrix whose diagonal entries are* $y_1^{(1)}, \ldots, y_1^{(n-1)}$, $\mathrm{diag}(y_2)$ *is the diagonal matrix whose diagonal entries are* $y_2^{(1)}, \ldots, y_2^{(n-1)}$, *and* $C(\Xi, \Lambda)$ *is the Cauchy matrix with entry on position* i, j *defined as*

$$C(\Xi, \Lambda)_{i,j} = \frac{1}{\lambda_j - \xi_i}.$$

Proof: As in Lemma 3.4.3, the matrix whose kth column is $(\lambda_k I - \Xi)^{-1} s_1$, can be written as

$$\begin{pmatrix} \frac{s_1}{\lambda_1 - \xi_1} & \cdots & \frac{s_1}{\lambda_{n-1} - \xi_1} \\ \vdots & & \vdots \\ \frac{s_{n-1}}{\lambda_1 - \xi_{n-1}} & \cdots & \frac{s_{n-1}}{\lambda_{n-1} - \xi_{n-1}}, \end{pmatrix}$$

which is the Cauchy matrix $C(\Xi, \Lambda)$ multiplied with $\mathrm{diag}(s_1)$ from the left. Similarly, the matrix whose kth column is $(\lambda_k I - \Xi)^{-1} s_2$, is the Cauchy matrix $C(\Xi, \Lambda)$ multiplied with $\mathrm{diag}(s_2)$ from the left. The matrix whose kth column is

$$(\lambda_k I - \Xi)^{-1} [s_1, s_2] y^{(k)}$$

has then as kth column

$$y_1^{(k)} \mathrm{diag}(s_1) C(\Xi, \Lambda) e_k + y_2^{(k)} \mathrm{diag}(s_2) C(\Xi, \Lambda) e_k$$

and the whole matrix can be written as

$$\mathrm{diag}(s_1) C(\Xi, \Lambda) \mathrm{diag}(y_1) + \mathrm{diag}(s_2) C(\Xi, \Lambda) \mathrm{diag}(y_2).$$

□

We have been using the following types of matrices.

Definition 3.4.5 *Consider two sequences of real numbers* $a_1, \ldots, a_n$ *and* $b_1, \ldots, b_n$ *where*

$$a_i \neq b_j, \quad i, j = 1, \ldots, n.$$

The Cauchy matrix $C(a, b)$ *for these sequences is the matrix whose entry on position* i, j *is*

$$\frac{1}{a_i - b_j}.$$

Consider two more sequences of nonzero real numbers $q_1, \ldots, q_n$ *and* $t_1, \ldots, t_n$. *The* Cauchy-like matrix $C(a, b)$ *with respect to these sequences is the matrix whose entry on position* i, j *is*

$$\frac{q_i t_j}{a_i - b_j}. \tag{3.14}$$

Cauchy and Cauchy-like matrices are examples of matrices with a so-called *low displacement rank* [32]. Their n^2 entries are fully defined with a

low number of scalars (in the case of a Cauchy-matrix with the $2n$ scalars $a_1, \ldots, a_n, b_1, \ldots, b_n$). The crucial fact for our implementation is, that matrices with a low displacement rank allow matrix-vector multiplication at costs lower than the usual $2n^2$ flops. For matrices with a low displacement rank the flop count is of the order $n \log n$, which can be achieved using techniques like the fast multipole method [12].

In our situation, we have to multiply the eigenvector matrix of the previous LOOCV iteration, V_{j-1}, first with the (normalized) Cauchy-like matrix given by Lemma 3.4.3. This requires $n-1$ matrix-vector products, giving in total order $(n-1)^2 \log(n-1)$ flops. Then we have to multiply the result with the (normalized) Cauchy-like matrix in Lemma 3.4.4. This will require another $(n-1)^2 \log(n-1)$ flops.

Thus the entire third step of Algorithm 2.4.1 requires in every iteration of the LOOCV process order $(n-1)^2 \log(n-1) + (n-1)^2$ flops in total.

3.5 Numerical experiments

In this section we compare the computing costs of leave-one-out cross validation applied to four classification methods: (1) The perturbation method described in section 2.4, see (2.5); (2) the nullspace method in subsection 2.4.1; (3) the intuitively reasonable criterion introduced in subsection 2.4.2 and forming the basis for Algorithm 2.4.1; (4) its fast implementation proposed in this chapter. We implemented all experiments in GNU Octave [16], a freely available programming language mainly intended for numerical computations and very similar to Matlab [30].

The costs for the individual methods are reported in terms of CPU-timings. However, these timings are in general not very representative for the actual computational and storage costs of the methods. This is due to the fact that some functions in Octave are internally linked to packages programmed in C++ or similar languages, which are significantly faster than Octave and Matlab (the priorities of Octave are rather numerical stability and user-friendliness). This holds only for particular functions (for instance, solving an eigenproblem), whereas others are programmed in Octave, thus making the overall timing misleading. In our situation, this causes the order of magnitude reductions in computational costs that we described (and proved) not to be reflected in the CPU-timings. A doubling of the number of genes (n) does not correspond to a 16 times longer solution process for classical LOOCV (corresponding to n^4 costs) nor to an 8 times longer solution process for our improved LOOCV (corresponding to n^3 costs) as it should. Also, we did not implement fast solution of eigenproblems using Cauchy matrices, as we were not able to find an Octave implementation of the multipole method or a different method for order $n \log n$ matrix-vector multiplication. Nevertheless, the presented timings show a clear improvement with the help of the techniques described in this chapter.

All experiments were performed on a notebook with a quad core Intel(R) Core(TM) i7 CPU at 1.60 GHz with 8 GB of RAM memory.

We first present the results when using the perturbation method de-

scribed in section 2.4, see (2.5) and the nullspace method in subsection 2.4.1; both are modifications of FLDA for the $p > n$ case. We performed LOOCV with these methods for the complete data set with $n = 63$ genes, but also with smaller data sets. More precisely, in subsequent experiments, 1 sample from each of the four gene groups was deleted, meaning that the total number of samples was decreased by four, and we measured the timings for this new, smaller sample size. This was done until the sample size was too small to contain $g = 4$ groups, which was the case with 35 samples in total. We illustrate the results in Figure 3.1 below, where the x-axis gives the number of samples and the y-axis the timing in seconds. In the perturbation method, ε in (2.5) was chosen as 10^{-5}.

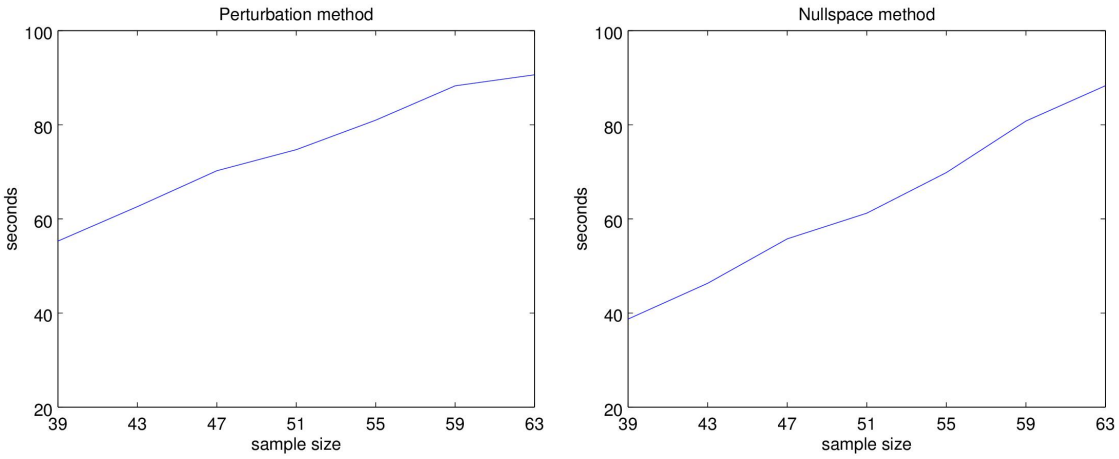

Figure 3.1: Timings of the perturbation method (left) and the nullspace method (right)

We see that both methods take about one and a half minute to perform the entire LOOCV process with all gene samples. With a smaller number of samples, the perturbation method is slower. This can be explained by the fact that this method solves eigenproblems of dimension p (order p^3 costs) whereas the nullspace method solves eigenproblems of dimension $p - n$ (order $(p - n + 1)^3$ costs) because the nullspace of W of dimension $n - 1$ has been projected out. The successful classification rates for the perturbation method are, from the smallest sample size to the full sample size $n = 63$: 100%, 100%, 100%, 96.078%, 96.364%, 96.610% and 96.825%. For the nullspace method the

successful classification rates are 100%, 100%, 100%, 100%, 100%, 98.305%, 100%.

The poorer performance of the perturbation method may be due to the sensitivity to the choice of ε in (2.5); our choice $\varepsilon = 10^{-5}$ may lead to a problem too far from original FLDA.

Now we turn our attention to the intuitively reasonable criterion introduced in subsection 2.4.2. The successful classification rate was for this method always 100%, indicating that the criterion is suited for the studied gene expression data. If we use its highly optimized implementation presented in [7], but not the special techniques for LOOCV of this chapter, we obtain the left part of Figure 3.2. The timings for the total LOOCV process are clearly below one second for the full sample size – a very substantial improvement as compared to the previous methods (note the different scaling of the y-axes). For the smallest tested sample size (39 samples), the total time is about 0.2 seconds. But if we use the techniques of this chapter designed for LOOCV testing, the full sample size requires about as much CPU-time as the smallest sample size without using our techniques. This is displayed in the right part of Figure 3.2. Even more important, the *slope* of the right curve is much smaller than the slope of the left curve, indicating a lower computational complexity (although it does not, as explained above, correspond to the order of magnitude improvement which we proved theoretically).

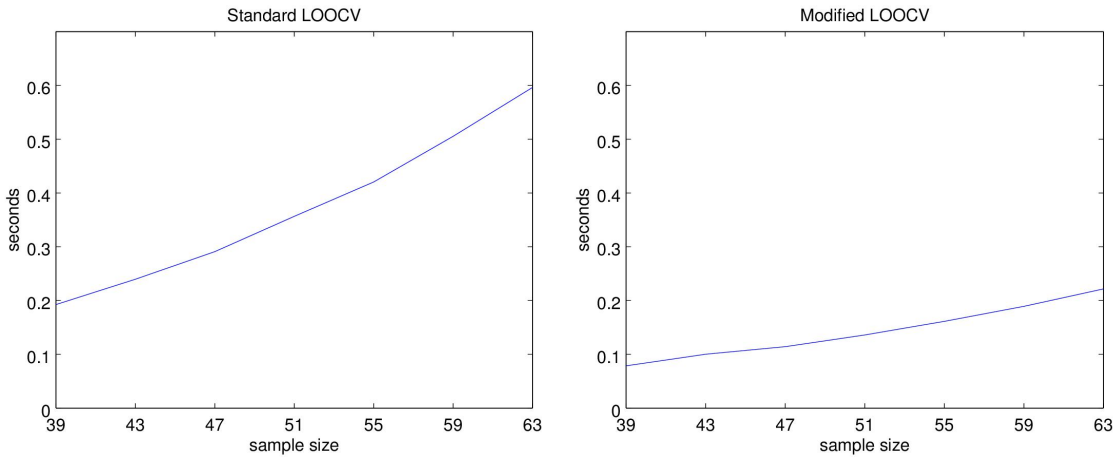

Figure 3.2: Timings of the standard LOOCV method (left) compared to the improved one (right) applied to FLDA with the criterion from subsection 2.4.2.

Conclusion

Our contribution to improve classification of gene expression signatures to tumor types does not concern the classification method itself (linear discriminant analysis) but rather its efficient testing. With typically a very high number of amino acids but a moderate number of gene samples, cross-validation testing of the classification procedure involves high computational and computer storage costs with potentially unreliable results due to finite arithmetics. Depending on the computer's available processor and memory, the testing phase can become useless for practice or simply uncomputable. The thesis shows that using techniques from linear algebra like low rank updates and Cauchy matrices, the overall complexity of cross-validation testing can be reduced by an order of magnitude: If p denotes the number of amino acids and n the number of genes, the computational costs are reduced from order $pn^3 + n^4$ floating point operations to order $pn^2 + n^3 \log n$ operations.

Bibliography

[1] ALTMAN N. S. : An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 1992, p.175–185.

[2] ARBENZ P. , GANDER W. , GOLUB G. H. : Restricted rank modification of the symmetric eigenvalue problem: Theoretical considerations. *Linear Algebra Appl.* 104, 1988, p.75-95.

[3] CHENG Y.G. , ZHUANG Y.M. , YANG J.Y. : Optimal Fisher discriminant analysis using the rank decomposition. *Pattern Recognition* 25, 1992, p.101-111.

[4] COVER TM : Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. *IEEE Transactions on Electronic Computers*, 1965, p. 326–334.

[5] DAI J. J. , LIEU L. , ROCKE D. : Dimension reduction for classification with gene expression microarray data. *Stat. Appl. Genet. Mol. Biol.* 5, 2006, p. 1544-6115.

[6] DEVIJNER P. A. , KITTLER J. : *Pattern Recognition: A Statistical Approach*. 1^{st} Ed. Prentice Hall, 1982, 480 p. ISBN-10: 0136542360. ISBN-13: 978-0136542360.

[7] DUINTJER TEBBENS J. , SCHLESINGER P. : Improving implementation of linear discriminant analysis for the high dimension/small sample size problem. *Computational Statistics and Data Analysis* 52, 2007, p.423-437.

[8] FIEDLER M. : *Special matrices and their applications in numerical mathematics*. 2^{nd} Ed. Dover Publications, 2008, 384p. ISBN-10: 0486466752. ISBN-13: 978-0486466750.

[9] FISHER R. A. : The use of multiple measurements in taxonomic problems. *Annals of Eugenics* 7, 1936, p.179–188.

[10] FRIEDMAN J.H. : Regularized discriminant analysis. *J. Amer Statist. Assoc.* 84, 1989, p.165-175.

[11] GEISSER S. : *Predictive Inference*. 1^{st} Ed. Chapman and Hall/CRC, 1993, 240 p. ISBN 0-412-03471-9.

[12] GERASOULIS A. : A fast algorithm for the multiplication of generalized Hilbert matrices with vectors. *Math. Comp.* 50, 1988, p.179-188.

[13] GOLUB G. H. , VAN LOAN C. F. : *Matrix computations*. 4^th Ed. Johns Hopkins University Press, Baltimore, MD, 2013, 756p. ISBN: 978-1-4214-0794-4

[14] GUO Y. , HASTIE T. , TIBSHIRANI R. : Regularized discriminant analysis and its application in microarrays. *Biostatistics* 8, 2007, p.86-100

[15] HONG Z.Q. , YANG J.Y. : Optimal discriminant plane for a small number of samples and design method of classifier on the plane. *Pattern Recognition* 24, 1991, p.317-324.

[16] http : //www.gnu.org/software/octave/

[17] http : //www.signa.org/index.pl?Display + Iris − setosa + 23, 2.5.2015

[18] http : //commons.wikimedia.org/wiki/File : Kosaciec_szczecinkowaty_Iris_setosa.jpg, 2.5.2015

[19] http : //www.desirableplants.com/g_galega_ixia_plants_by_mail_order.html,, 2.5.2015

[20] http : //www.ouellette001.com/PapiersPeints/Iris_versicolor.htm,, 2.5.2015

[21] http : //commons.wikimedia.org/wiki/File : Iris_virginica.jpg, 2.5.2015

[22] https : //www.prairiemoon.com/seeds/wildflowers − forbs/iris − virginica −shrevei − southern − blue − flag − iris.html, 2.5.2015

[23] http : //w3.uniroma1.it/chemo/metrics/chemom/mslide4.html, 2.5.2015

[24] KALINA J. : Classification methods for high-dimensional data. *Biocybernetics and Biomedical Engineering* 34, 2014, p.10-18.

[25] KALINA J. , VALENTA Z. , DUINTJER TEBBENS J. : Computation of Regularized Linear Discriminant Analysis. *Proceedings of COMPSTAT 2014 M. Gilli, G. Gonzalez-Rodrigues, A. Nieto-Reyes (Eds.), Universite de Geneve*, 2014, p.1-8.

[26] KHAN J. , WEI J. S. , RINGER M. , SAAL L. H. , LADANYI M. , WESTERMANN F. , BERTHOLD F. , SCHWAB M. , ANTONESCU C. R. , PETERSON C. , MELTZER P. S. : Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nat Med.*, 2001, p.673–679.

[27] KIM H. , HOWLAND P. , PARK P. : Dimension reduction in text classification with support vector machines. *J. Mach. Learn. Res* 6, 2005, p.37-53.

[28] KRZANOWSKI W.J. ,JONATHAN P. ,MCCARTHY W.V. , THOMAS M.R. : Discriminant analysis with singular covariance matrices: methods and applications to spectroscopic data. *Appl. Statist.* 44, 1999, p.101-115.

[29] LING L. , TAMER ÖZSU M. : *Encyclopedia of Database Systems*. Springer, 2009, 3748 p. ISBN: 978-0387355443

[30] Mathworks, Inc. , MATLAB 8.5 , 2015 , http : //www.mathworks.com/products/matlab/

[31] MOLER C.B. , STEWART G.W. : An algorithm for generalized matrix eigenvalue problems. *SIAM J. Numer. Anal.* 10, 1973, p.241-256.

[32] PAN V. Y. , WANG X. : Inversion of displacement operators. *SIAM J. Matrix Anal. Appl.* 24, 2003, p.660-677.

[33] PRESS W. H., TEUKOLSKY S.A.; VETTERLING W. T., FLANNERY B. P. : *The Art of Scientific Computing* 3^{rd} Ed. Cambridge University Press, 2007. ISBN: 978-0-521-88068-8.

[34] RAO, R. C. : The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society* 2, 1948, p.159–203.

[35] RENCHER A. C. : *Methods of Multivariate Analysis*. 2^{nd} Ed. Wiley-Interscience, 2002, 738 p. ISBN: 978-0-471-46172-2.

[36] R. Development Core Team : *A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, 2005.

[37] SAAD Y. : *Numerical Methods for Large Eigenvalue Problems*. SIAM - Society for Industrial and Applied Mathematics, 2011, 292 p. ISBN-10: 1611970725. ISBN-13: 978-1611970722.

[38] STRANG G. : *Introduction to linear algebra*. 3^{rd} Ed. Wellesley-Cambridge Press, 2003, 568 p. ISBN-10: 0961408898. ISBN-13: 978-0961408893.

[39] YANG J. , YANG J.Y. : Why can LDA be performed in PCA transformed space? *Pattern Recognition* 36, 2003, p.563-566.