Pattern separation via ellipsoids and conic programming

Fr. Glineur

Faculté Polytechnique de Mons, Belgium

Mémoire présenté dans le cadre
du *D.E.A. interuniversitaire en mathématiques*

31 août 1998

Contents

To Deborah

Acknowledgments

This is an excellent opportunity to thank the people to which I am greatly indebted : Professor Jacques Teghem, my promotor, who supported and guided me during this year, Professors Martine Labbé, Michel Goemans, Van Hien Nguyen, Jean-Jacques Strodiot and Philippe Toint, who at some point during this *D.E.A. program* made me discover some of the most interesting topics in optimization and Professor Tamás Terlaky, who invited me to give a talk on the subject of this thesis at the *HPOPT98 conference* in Rotterdam.

I am also very grateful to the entire staff of the *Mathematics and Operations Research* department at the *Faculté Polytechnique de Mons*, for their constant kindness, availability and support. I'll specifically mention Dr. Philippe Fortemps, who devoted a large part of his precious time to the proofreading of this thesis.

Finally, I would like to express my gratitude towards my parents and my future wife, whose love and support are to a large extent responsible for the ideal environment I benefited from during the writing of this thesis.

Introduction

Machine learning is a scientific discipline whose purpose is to design computer procedures that are able to perform classification tasks. For example, given a certain number of medical characteristics about a patient (e.g. age, weight, blood pressure, etc.), we would like to infer automatically whether he or she is healthy or not.

A special case of machine learning problem is the separation problem, which asks to find a way to classify patterns that are known to belong to different well-defined classes. This is equivalent to finding a procedure that is able to recognize to which class each pattern belongs.

The obvious utility of such a procedure is its use on unknown patterns, in order to determine to which one of the classes they are most likely to belong.

From a completely different point of view, *mathematical programming* is a branch of optimization which seeks to minimize or maximize a function of several variables under a set of constraints.

Linear programming is a special case for which a very efficient algorithm is known since a long time, the *simplex method*([Dan63]). Another completely different kind of efficient method has been developed much later to solve the linear programming problem : the so-called *interior-point methods* ([Kar84]).

A certain type of interior-point methods has been very recently generalized to a broader class of problems called the *SQL conic programs.* The objective of this work is to

Solve the pattern separation problem using SQL conic programming.

In addition to that statement, the main idea of this thesis is to use ellipsoids to perform the pattern separation. Here is a short overview of this document.

The first chapter is about mathematical programming. We will start by describing how and why researchers were led to study special types of mathematical programs, namely convex programs and conic programs. We will also provide a detailed discussion about conic duality and give a classification of conic programs. We will then describe what are *self-scaled cones* and why they are so useful in conic programming. Finally, we will give an overview of what can be modelled using a SQL conic program, keeping in mind our pattern separation problem. Since most of the material in the chapter is standard, many of the proofs are omitted.

The second chapter will concentrate on pattern separation. After a short description of the problem, we will successively describe four different separation methods using SQL conic programming. For each method, various properties are investigated. Each algorithm has in fact been successively designed with the objective of eliminating the drawbacks of the previous one, while keeping its good properties. We conclude this chapter with a small section describing the state of the art in pattern separation with ellipsoids.

The third chapter reports some computational experiments with our four methods, and provides a comparison with other separation procedures.

Finally, we conclude this work by providing a short summary, highlighting the author's personal contribution and giving some interesting perspectives for further research.

Chapter 1

Conic programming

1.1 Introduction

The goal of *Operations Research* is to model real-life situations where there is a decision to take (several alternatives available, numerical parameters to tune, planning, etc.) in order to find the best one(s) to take.

The concept of *best decision* is of course problem-dependent and very difficult to define. Practitioners usually describe a decision as a set of n parameters called *decision variables*, and try to minimize (or maximize) an *objective function* depending on the decision variables. This function may for example compute the cost associated to a decision. *Optimization*, a mathematical discipline which seeks to minimize (or maximize) functions of several variables, is thus closely related to operations research.

Most of the time, not all combinations of parameters are allowed (e.g. physical dimensions must be positive, a system must satisfy some performance requirements, etc.). This means that we have to describe the set $D \subseteq \mathbb{R}^n$ of allowed decisions. We call them *feasible solutions* and D is the *feasible set*. From this we understand that *mathematical programming* is very helpful to operations research practitioners, since it is a branch of optimization whose goal is to minimize an objective function f_0 depending on n decision variables under a set of constraints, which can be mathematically stated as

$$\min_{x \in \mathbb{R}^n} f_0(x) \quad \text{s.t.} \quad x \in D \subseteq \mathbb{R}^n$$

where x is a vector of $\mathbb{R}^n$ containing the n decision variables. Most of the time, a description of D may be given as a set of l inequality and/or m equality constraints defined by $f(x) \leq 0$ and $g(x) = 0$[1], where f and g are nonlinear mappings from $\mathbb{R}^n$ to $\mathbb{R}^l$ and $\mathbb{R}^m$, and the problem becomes[2]

$$\text{(NLP)} \qquad \min_{x \in \mathbb{R}^n} f_0(x) \quad \text{s.t.} \quad \begin{cases} f(x) \leq 0 \\ g(x) = 0 \end{cases}$$

Unfortunately, we are not able to solve this problem efficiently in the general case. Actually, some complexity results provide a *lower* bound on the number of arithmetic operations needed to solve

[1] The $\leq$ sign is to be understood componentwise

[2] It is easy to show that other types of constraints, such as $f(x) \geq 0$ or $x \in \{0, 1\}^n$, can be easily reformulated into one of these two forms.

the general problem. Even for moderate sizes and accuracy requirements, the bound is very high, e.g. 10^{20} operations to solve a problem involving only 10 variables to 1% accuracy, see [Nes96].

This explains why most of the research in the mathematical programming field concentrates on special cases of the problem, i.e. restrict functions f_0, f and g to certain categories.

Linear programming[3] consists in the special case where both the objective function f_0 and the mappings f and g are affine. Using matrix notation, we may formulate the problem as

$$\text{(LP)} \qquad \min_{x \in \mathbb{R}^n} c^T x \quad \text{s.t.} \quad \begin{cases} Cx \leq d \\ Ax = b \end{cases}$$

where matrices A and C have dimensions $l \times n$ and $m \times n$, column vectors b and d have size l and column vectors x and c have both size n.

Linear programming is in fact the greatest success story in mathematical programming, mostly for two reasons

- Many problems from various fields can be formulated as (or efficiently approximated by) linear programs, e.g. production planning, transportation, scheduling, etc.
- Dantzig discovered in 1957 a very efficient algorithm to solve linear programs : the *simplex method*([Dan63]). It has been thoroughly studied and improved since its first appearance, and is now widely used in commercial software to solve a great variety of problems.

However, linear programming has a serious limitation : some problems simply cannot be formulated as linear programs. This is why researchers have been looking for other kinds of formulations

- that are able to model a broader class of problems (than linear programming), and
- for which an efficient algorithm is available

There is of course a tradeoff between generality and algorithmic efficiency : the more general your problem, the less efficient your methods. A first extreme case is linear programming, a very particular (yet useful) type of problem with a very efficient algorithm, while another extreme case is general nonlinear programming, where no method can compare with the simplex algorithm's efficiency, even in trying to find an approximate solution.

In the next paragraphs, we are going to describe such an intermediate formulation, much more powerful than linear programming while being still solvable with efficient algorithms : the interior-point methods.

1.2 Convex programming

An important difference between linear and nonlinear programming is the existence of local minima. In the linear case, any local minimum is also a global minimum for the problem, while in the nonlinear case some true local minimum may exist (and do exist most of the time). An

[3] Some authors prefer to use the term *Linear optimization*, the term *programming* being too related to computer science.

algorithm designed for nonlinear programming may thus converge to a local minimum, delivering a non-optimal solution whose objective value may be arbitrarily worse than the true global optimum.

Some researchers, namely global optimization researchers, accept this drawback and try to circumvent it with algorithmic methods. However, another way of avoiding this problem has been thoroughly studied : restrict the general problem (NLP) to convex programming. We need the following definitions :

Definition 1.1 *A set $C \subseteq \mathbb{R}^n$ is said to be a* convex set *if for any $x, y \in C$ we have*

$$\lambda x + (1-\lambda)y \in C \text{ when } 0 < \lambda < 1$$

As an important example, note that the affine subspaces of $\mathbb{R}^n$ are convex sets. Another example is the positive orthant $\mathbb{R}^n_+ = \{x \in \mathbb{R}^n \mid x \geq 0\}$. We have the following useful properties about convex sets.

Theorem 1.2 *If $A \subseteq \mathbb{R}^n$ and $B \subseteq \mathbb{R}^n$ are two convex sets, their intersection $A \cap B$ is also a convex set in $\mathbb{R}^n$.*

Theorem 1.3 *If $A \subseteq \mathbb{R}^n$ and $B \subseteq \mathbb{R}^m$ are two convex sets, their Cartesian product $A \times B$ is also a convex set in $\mathbb{R}^{m+n}$.*

Definition 1.4 *A function $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$ is said to be a* convex function *if the following set is convex*

$$\text{epi } f = \{(x, \alpha) \in \mathbb{R}^n \times \mathbb{R} \mid f(x) \leq \alpha\}$$

This set is called the epigraph *of f.*

We also note that affine functions are convex. For more information about convex analysis, see the reference book [Roc70]. When functions f_0 and f are convex, the following problem

$$\text{(CVX)} \qquad \min_{x \in \mathbb{R}^n} f_0(x) \quad \text{s.t.} \quad f(x) \leq 0$$

is called a *convex program* (note there are no more equality constraints). Convex programs are important because of the following interesting theorem :

Theorem 1.5 *The feasible set of a convex program*

$$\{x \in \mathbb{R}^n \mid f(x) \leq 0\}$$

is a convex set. Moreover, every solution that is a local minimum is also a global minimum. [4]

This means that an algorithm that converges to any local minimum is guaranteed to give us the global minimum value of the objective function.

Linear programs are in fact convex programs : the objective is linear, thus convex, while the constraints $Ax = b$ and $Cx \leq d$ may be expressed as $Ax - b \leq 0$, $-Ax + b \leq 0$ and $Cx - d \leq 0$,

[4] Note that this does not mean that the local minimum is unique.

which are affine, thus convex, constraints. This explains why, as stated above, any local minimum in a linear program is also a global optimum.

Many algorithmic approaches have been applied to solve general convex programs (e.g. subgradient schemes), but we would like to stress an important fact about convex programs : given an optimization problem with the functions f_0 and f, it might be more difficult to check that the problem is convex than to solve it !

Basically, this means that we have make sure our functions f_0 and f are convex before trying to solve the problem with a convex programming algorithm. This implies we possess some kind of knowledge about the *structure* of the problem. In the following section, we describe the conic optimization setting where our knowledge about the problem is confined in the description of a convex cone.

1.3 Conic optimization

We start with some basic definitions and theorems.

Definition 1.6 *A set C is a* cone *if and only if*

$$x \in C \Rightarrow \lambda x \in C \quad \forall \lambda \geq 0$$

Checking the convexity of a cone is simpler than for a general set :

Theorem 1.7 *A cone C is convex if and only if*

$$x \in C, y \in C \Rightarrow x + y \in C$$

A cone is said to be *pointed* if it doesn't contain any straight line, which can be expressed as

Definition 1.8 *A cone C is pointed if and only if*

$$-C \cap C = \{0\}$$

where $-C$ stands for $\{x \mid -x \in C\}$

Finally, a cone is said to be *solid* if it has a nonempty interior, i.e.

Definition 1.9 *A cone C is solid if and only if*

$$\text{int } C \neq \varnothing$$

For example, the positive orthant is a pointed and solid convex cone. A linear subspace is a convex cone that is neither pointed, nor solid (except $\mathbb{R}^n$ itself).

We are now in position to describe the general conic optimization problem : given a closed, pointed and solid convex cone C, a vector b and a linear subspace L, the following problem

$$\text{(CONE)} \qquad \min_{x \in \mathbb{R}^n} c^T x \quad \text{s.t.} \quad x \in C \cap (b + L)$$

is a *conic program.*

We are thus seeking the minimum value of a linear objective on the intersection of a convex cone and an affine subspace.

This type of problem has been thoroughly studied by Nesterov and Nemirovsky in their book [NN94]. Here are the main results of their theory

- Any convex program (CVX) can be transformed into an equivalent conic program (CONE).
- A ν-self-concordant barrier F for a cone C is a barrier function defined on the interior of C such that
  - $F(x) \to +\infty$ when x approaches the boundary of C
  - The variation of F is bounded by twice the norm of $\nabla^2 F$
  - The variation of $\nabla^2 F$ is bounded by $\sqrt{\nu}$ times the norm of $\nabla^2 F$ (these last two items may be expressed with rather technical inequalities involving ∇F, $\nabla^2 F$ and $\nabla^3 F$)
- Any n-dimensional convex cone admits a $\mathcal{O}(n)$-self-concordant barrier (unfortunately, there is no efficient scheme to compute this barrier explicitly)
- Given a convex cone C and a ν-self-concordant barrier, the authors design a interior-point method that solves the conic program (CONE) up to ϵ relative accuracy in

  $$\mathcal{O}(\sqrt{\nu} \log \frac{1}{\epsilon})$$

  iterations (each iteration is basically a damped Newton step). This implies that the (CONE) program is solvable in polynomial time. We also understand that low values of ν lead to faster methods.

As an illustration, let us consider the case of linear programming. It is well known that any linear program may be reduced to the so called standard form (nonnegative variables with equality constraints) using standard manipulations (splitting free variables into two nonnegative parts, introducing slack variables for each inequality)

$$\min_{x \in \mathbb{R}^n} c^T x \quad \text{s.t.} \quad \begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

This linear program is in fact equivalent to the following conic program

$$\min_{x \in \mathbb{R}^n} c^T x \quad \text{s.t.} \quad x \in C \cap (d + L)$$

where C is the n-dimensional positive orthant $\mathbb{R}^n_+$ (which is thus responsible for $x \geq 0$), $L = \{x \mid Ax = 0\}$ is the null space of A and d is any vector such that $Ad = b$ (if such a vector doesn't exist, the original program is infeasible).

It is straightforward to check that the following function

$$F(x) = -\sum_{i=1}^{n} \log x_i$$

is a n-self-concordant barrier function for $\mathbb{R}^n_+$, which implies that it is possible to solve a linear program using an interior-point methods with

$$\mathcal{O}(\sqrt{n} \log \frac{1}{\epsilon})$$

iterations.

Why is the conic formulation interesting ? We may view in fact the constraint $x \in C$ as a generalization of the traditional nonnegativity constraint $x \geq 0$, which corresponds to the special case $C = \mathbb{R}^n_+$, as we have seen above.

For example, one of the most useful cones in optimization is the *positive semidefinite* cone $\mathbb{S}^n_+$.

Definition 1.10 *The positive semidefinite cone $\mathbb{S}^n_+$ is a subset of $\mathbb{S}^n$, the set of symmetric $n \times n$ matrices. It consists of all positive semidefinite matrices, i.e.*

$$M \in \mathbb{S}^n_+ \Leftrightarrow x^T M x \geq 0 \; \forall x \in \mathbb{R}^n \Leftrightarrow \lambda(M) \geq 0$$

where $\lambda(M)$ denotes the vector of eigenvalues of M.

It is straightforward to check that $\mathbb{S}^n_+$ is a closed, solid, pointed convex cone. A conic program using $\mathbb{S}^n_+$ is called a *semidefinite program.* As we will see later, this cone provides us with the ability to model many more types of constraints than a linear program.

It is also worth noting that $\log\det M$ is a n-self-concordant barrier for $\mathbb{S}^n_+$, with the same parameter as for $\mathbb{R}^n_+$. This means roughly that solving a semidefinite program involving a square $n \times n$ matrix of unknowns requires in theory the same number of iterations as a linear program involving an n-dimensional vector !

Before going on, we dwell a little upon positive semidefinite matrices. Stressing the fact the $\mathbb{S}^n_+$ allows us to generalize the $\geq$ sign, we will denote that a matrix M is positive semidefinite by $M \succeq 0$. Similarly, the notation $A \succeq B$ will mean $A - B \succeq 0$. The notation $M \succ 0$ will mean that M is positive definite, i.e. $M \in \operatorname{int} \mathbb{S}^n_+$. This is equivalent to $\lambda(M) > 0$ or $x^T M x > 0 \; \forall x \neq 0$. Finally, we state a very useful theorem about positive definite matrices :

Theorem 1.11 (Schur's complement theorem) *Let $M = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$ where A and C are symmetric matrices. If $A \succ 0$ we have the following equivalence*

$$M \succeq 0 \Leftrightarrow C \succeq B^T A^{-1} B$$

1.3.1 Conic duality

Conic optimization enjoys the same kind of rich duality theory as linear programming. We give in this section a brief overview of this theory.

Definition 1.12 *The* dual *of a cone C is the set*

$$C^* = \{x \mid x^T y \geq 0 \; \forall y \in C\}$$

The dual of a cone is always a cone. In fact we have the following

Theorem 1.13 *The dual of a closed, pointed and solid convex cone C is a closed, pointed and solid convex cone C^**

For example, the dual of $\mathbb{R}^n_+$ is $\mathbb{R}^n_+$ itself. We say it is *self-dual.* Another example is the dual of the linear subspace L, which is $L^* = L^\perp$, the linear subspace orthogonal to L.

Theorem 1.14 *If C is a closed, pointed and solid convex cone, $C^{**} = C$*

Using standard Lagrangean theory, it is possible to derive a dual program to the standard conic program.

Theorem 1.15 *The primal conic program*

$$(\mathrm{P}) \qquad \min_{x\in\mathbb{R}^n} c^T x \quad s.t. \quad x \in C \cap (b+L)$$

admits the following dual program

$$(\mathrm{D}) \qquad \min_{y\in\mathbb{R}^n} b^T y \quad s.t. \quad y \in C^* \cap (c+L^\perp)$$

There's a great deal of symmetry between the two programs. In fact, we see that the dual of a conic program based on b, c, C and L is also a conic program, based on the dual objects c, b, C^* and L^*. This implies immediately that the dual of (D) is (P) itself.

Without loss of generality, we may suppose that $b \in L^\perp$ and $c \in L$. With this convention, we have the following

Theorem 1.16 *Let x and y be respectively a feasible solution of (P) and (D). We have a weak duality property expressed as*

$$c^T x + b^T y \geq 0$$

The quantity

$$x^T y = c^T x + b^T y \geq 0$$

is called the duality gap.

Obviously, a pair (x, y) with a zero duality gap must be optimal. It is well known that the converse is true with linear programming (this is the *strong duality* theorem), but this is not the case here. In the following section, we describe all the possible types of conic programs regarding feasibility, attainability of the optimum and duality gap.

1.3.2 Types of conic programs

Given a standard conic program

$$(\mathrm{P}) \qquad \min_{x\in\mathbb{R}^n} c^T x \quad \text{s.t.} \quad x \in C \cap (b+L),$$

let

$$\mathcal{F}_+ = \{x \in \mathbb{R}^n \mid x \in C \cap (b+L)\}$$

be its feasible set and $\delta = \mathrm{dist}(C, b+L)$ the minimum distance between C and the affine subspace $(b+L)$. We also call $\mathcal{F}_{++}$ the set of feasible solutions belonging to the interior of C, i.e.

$$\mathcal{F}_{++} = \{x \in \mathbb{R}^n \mid \ x \in \operatorname{int} C \cap (b+L)\}$$

First of all, the distinction between feasible and infeasible conic programs is not as clear-cut as for linear programming. We have the following cases[5]

[5] In the following, we'll mark with a ‡ the cases which can't happen for linear programs

- A conic program is infeasible. This means the feasible set $\mathcal{F}_+ = \varnothing$. But we distinguish two subcases
  - $\delta = 0$, which means an infinitesimal perturbation of the problem data may transform the program into a feasible one. We call the program *weakly infeasible*‡.
  - $\delta > 0$, which corresponds to the usual infeasibility as for linear programming. We call the program *strongly infeasible*
- A conic program is feasible, which means $\mathcal{F}_+ \neq \varnothing$ (and thus $\delta = 0$). We also distinguish two subcases
  - $\mathcal{F}_{++} = \varnothing$, which means an infinitesimal perturbation of the problem data may transform the program into a infeasible one. We call the program *weakly feasible.*
  - $\mathcal{F}_{++} \neq \varnothing$. We call the program *strongly feasible.* This means there exists at least one feasible solution belonging to the interior of C, and we say the problem satisfies a *Slater* condition (or *interior-point* condition).

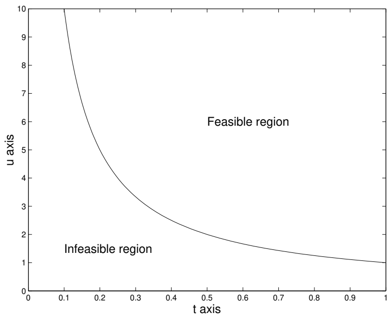
It is possible to characterize these situations by looking at the existence of certain types of directions in the dual problem (level direction, improving direction, improving direction sequence, see [Stu97]).

Examples Let us take $C = \mathbb{S}^2_+$. Let

$$x = \begin{pmatrix} t & v \\ v & u \end{pmatrix}$$

It is easy to see that $x \in C \Leftrightarrow t \geq 0$ and $tu \geq v^2$.

If we choose the affine space $(b + L)$ to enforce the constraint $v = 1$, the feasible set is the epigraph of the positive branch of the hyperbola $tu = 1$, i.e. $\mathcal{F}_+ = \{(t, u) \mid t \geq 0 \text{ and } tu \geq 1\}$ as depicted on the following figure



This problem is strongly feasible. If we add (into the description of the affine subspace) the constraint $t = -1$, we get a strongly infeasible problem (since t must be positive). If we add $t = 0$, we get a weakly infeasible problem (since the distance between the axis $t = 0$ and the hyperbola is zero). If we add $t + u = 2$ we get a weakly feasible problem (because the only feasible point, $t = u = v = 1$, does not belong to the interior of C).

Let us denote $p^* = \inf_{x \in \mathcal{F}_+} c^T x$ the optimal value of (P)[6]. We also denote by $\mathcal{F}^*$ the set of optimal solutions, i.e. feasible solutions with an objective equal to p^*

$$\mathcal{F}^* = \mathcal{F}_+ \cap \{x \in \mathbb{R}^n \mid c^T x = p^*\}$$

[6] Note that we are talking of an infimum, not a minimum, because we are not sure this infimum is attained.

We have the following distinction regarding attainability of the optimum

- A conic program is *solvable* if $\mathcal{F}^* \neq \varnothing$.
- A conic program is *unsolvable* if $\mathcal{F}^* = \varnothing$, but we have two subcases
  - If $p^* = -\infty$, the program is *unbounded*, which is the only possibility for a feasible but unsolvable linear program
  - If p^* is finite, we have a feasible unsolvable bounded program ‡. This situation happens when the infimum defining p^* is not attained, i.e. there exists feasible solution with objective value arbitrarily close to p^* but no optimal solution.

There are of course relations between the primal and the dual. For example, as for linear programming, the weak duality implies that the dual of an unbounded problem is infeasible (see [Stu97] for more information).

Examples With the same strongly feasible problem as above (epigraph of an hyperbola), we choose a linear objective equal to $t + u$. In that case, $\mathcal{F}^*$ is the point $(t, u, v) = (1, 1, 1)$, and the problem is solvable ($p^* = 2$). If we choose an objective equal to $-(t + u)$, $\mathcal{F}^* = \varnothing$ because $p^* = -\infty$, and the problem is unbounded. Finally, choosing t as objective leads to an unsolvable bounded problem : p^* is equal to zero but $\mathcal{F}^* = \varnothing$ because there is no feasible solution with $t = 0$ since $tu \geq 1$.

Finally, we state the various possibilities about the *optimal duality gap* (which is defined as $p^* + d^*$, where d^* is the equivalent of p^* for the dual problem)

- The optimal duality gap is strictly positive ‡
- The optimal duality gap is zero but there is no optimal solution pair. In this case, there exists pairs (x, y) with an arbitrarily small duality gap (which means that the optimum is not attained for at least one of the two programs (P) and (D)) ‡
- An optimal solution pair (x, y) has a zero duality gap, as for linear programming

Fortunately, we may avoid all the trouble caused by these complications if our problem satisfies the Slater condition.

Theorem 1.17 *If the primal program satisfies the Slater condition, the optimal duality gap is zero. Moreover, if p^* is finite (i.e. the primal is not unbounded), we have that the dual is solvable and its optimal solution set is bounded.*

Of course, the dual theorem holds if you switch the words primal and dual. This means that if both the primal and the dual satisfy the Slater condition, the optimal duality gap is zero, both programs are solvable and their optimal solution sets are bounded.

Most of the programs that occur in practice will indeed satisfy the Slater condition.

Examples The strongly feasible program given above has the following data : $C = \mathbb{S}^2_+$,

$$c = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, b = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ and } L = \text{span}\left\{ \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \right\}$$

which gives

$$\min t + u \quad \text{s.t.} \quad v = 1 \text{ and } tu \geq 1$$

Its optimal value is $p^* = 2$. We easily compute that

$$L^* = \operatorname{span} \left\{ \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \right\}$$

which implies that the dual may be stated as

$$\min 2g \quad \text{s.t.} \quad e = 1, f = 1 \text{ and } ef \geq g^2$$

(letting $y = \begin{pmatrix} e & g \\ g & f \end{pmatrix}$).

The optimal value d^* is equal to -2 (because the last constraint is equivalent to $g^2 \leq 1$) , and the optimal duality gap $p^* + d^*$ is zero as expected.[7]

Changing the objective to $c = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, we get an unsolvable bounded problem

$$\min t \quad \text{s.t.} \quad v = 1 \text{ and } tu \geq 1$$

whose optimal value is $p^* = 0$. The dual becomes

$$\min 2g \quad \text{s.t.} \quad e = 1, f = 0 \text{ and } ef \geq g^2$$

which has only one feasible solution, namely $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, and an optimal value $d^* = 0$. In this case, the optimal duality gap is zero but is not attained (because the primal problem is unsolvable).

Finally, we give here an example where the optimal duality gap is nonzero. Let $C = \mathbb{S}^3_+$,

$$b = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix} \quad c = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad L = \operatorname{span} \left\{ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right\}$$

and

$$L^* = \operatorname{span} \left\{ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix} \right\}$$

It is possible to show that the only feasible solution for the primal problem is

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

giving $p^* = 3$, while the dual also admits the same unique feasible solution leading to $d^* = 6$. The optimal duality gap is equal to $p^* + d^* = 9 > 0$. Note that in this case, as expected from the theory, none of the two problems satisfies the Slater condition since their unique feasible solution doesn't belong to the interior of $\mathbb{S}^3_+$.

[7] The reader may check that we have chosen b and c such that $b \in L^\perp$ and $c \in L$ in order to have a zero optimal duality gap.

1.4 Self-scaled cones

The results of Nesterov and Nemirovsky ([NN94]) imply that *any* conic program can be solved by a polynomial-time interior-point algorithm. Since we know good self-concordant barriers (i.e. with low ν parameter) for all cones useful in applications, one could think we can solve efficiently all our conic problems with their algorithm.

But the practical performance of these methods is somewhat disappointing : although the number of iterations is bounded by a polynomial in the size of the problem, this number is very high in practice.

This is in fact a major drawback in complexity theory. An implicit hypothesis in that field is that methods with a low complexity bound perform well in practice, but this is unfortunately not true. For example, considering linear programming, the simplex method has an exponential complexity while the so-called ellipsoid method has polynomial complexity (see e.g. [Sch86]). However, the simplex method performs much better in practice, the ellipsoid method being hopelessly slow on any practical problem. The reason behind this apparent discrepancy is that these complexity bounds actually describe the worst case, i.e. suppose the problem data is chosen such that the algorithm has the slowest possible execution. While the simplex method algorithm very rarely attains its worst-case bound (we need in fact *ad hoc* examples to demonstrate exponential complexity, which never occurs in practice), the ellipsoid method practical complexity always matches its worst-case bound, which explains its bad performance in practice.

Similarly, Nesterov and Nemirovsky's general interior-point method for conic programming takes very short steps, which makes the iteration count much too high in practice and practically equal to the worst-case bound. However, it is known that some interior-point methods are very efficient in practice for linear programming (these are the so-called *large step primal-dual* methods, see e.g. the textbook [Wri97]).

This is why some researchers have become interested in a special category of conic programs for which they could design a large step primal-dual algorithm similar to the efficient linear programming algorithm : these programs are using *self-scaled cones.*

In a seminal paper ([NT97]), Nesterov and Todd describe a polynomial primal-dual method for conic programs involving self-scaled cones. This method takes large steps, i.e. steps that go typically a large fraction of the way to the boundary of the feasible set (instead of the short steps computed by the general method, that are confined to a ball of unit radius in the local norm defined by the Hessian of the barrier).

The definition of a self-scaled cone was originally stated by Nesterov and Todd using a special type of self-concordant barrier. But it was later observed that their definition was in fact equivalent to some concepts well-known to mathematicians. Such cones may in fact be characterized as

- Self-scaled cones
- Homogeneous and self-dual cones
- Cones for which there exists a v-space construction (see [Tun96])

We give here the second characterization because of its geometrical meaning. We already know that a cone is self-dual when $(C^*)^* = C$.

Definition 1.18 *A cone* $C \subseteq \mathbb{R}^n$ *is called a* homogeneous cone *if for any pair of points* $x, y \in$

int C *there exists an invertible linear mapping* $A : \mathbb{R}^n \mapsto \mathbb{R}^n$ *leaving* C *invariant such that* $A(x) = y$

This property is in fact crucial in the development of an efficient primal-dual method, because it allows some kind of efficient scaling of the current primal-dual iterate.

Nesterov and Todd give three important examples of self-scaled cones : the positive orthant $\mathbb{R}^n_+$, the cone of positive semidefinite matrices $\mathbb{S}^n_+$ and the second order cone $\mathbb{L}^n_+$ [8]

Definition 1.19 *The second order cone* $\mathbb{L}^n_+$ *is a set of* $(n+1)$*-dimensional vectors defined as*

$$\mathbb{L}^n_+ = \{(r, x) \in \mathbb{R} \times \mathbb{R}^n \mid r \geq \|x\|\}$$

where $\|\cdot\|$ *denotes the usual Euclidean norm on* $\mathbb{R}^n$.

It is indeed straightforward to check that this is a closed, pointed and solid convex cone (it is in fact the epigraph of the Euclidean norm).

In fact, these three examples are a little more than just examples. Using a classification theorem about Euclidean Jordan algebra, it has been shown ([PJW34]) that any self-scaled cone may be expressed as the Cartesian product of *primitive* self-scaled cones. There are only six primitive self-scaled cones, namely

1. The nonnegative reals $\mathbb{R}_+$
2. The second order cone $\mathbb{L}^n_+$
3. The cone of positive semidefinite matrices with real entries $\mathbb{S}^n_+$ (for any $n > 2$)
4. The cone of positive semidefinite matrices with complex entries (for any $n > 2$)
5. The cone of positive semidefinite matrices with quaternion entries (for any $n > 2$)
6. The cone of 3×3 positive semidefinite matrices with octonion entries

For example, the positive orthant $\mathbb{R}^n_+$ is the Cartesian product of n cones $\mathbb{R}_+$. There is no other self-scaled cone with real entries than the three examples mentioned above.

One may find a bit disappointing the fact that the class of self-scaled cones is not broader. However, as we'll see in the next section, those cones allow us to model much more problems than linear programming.

1.5 Modelling with self-scaled cones

Since we are now able to solve efficiently in practice conic programs involving self-scaled cones, we are now going to investigate what kind of problems we can model with that formulation.

We'll limit ourselves to the real case, i.e. will only use the three real self-scaled cones $\mathbb{R}^n_+$, $\mathbb{L}^n_+$ and $\mathbb{S}^n_+$.[9]

[8] This cone is also known as Lorentz cone or ice-cream cone.

[9] The interiors of these cones will be respectively denoted by $\mathbb{R}^n_{++}$, $\mathbb{L}^n_{++}$ and $\mathbb{S}^n_{++}$.

Since conic programming is a special case of convex programming, there is no hope of modelling a nonconvex problem. This is of course an important restriction, probably the most severe limitation to the applicability of the formulation. For example, the set

$$S = \{(a, b, c) \in \mathbb{R}_+ \times \mathbb{R}_{++} \times \mathbb{R}_+ \mid \frac{a}{b} \leq c\}$$

will be impossible to model because it is nonconvex. An easy way to prove this fact is to notice that S is the epigraph of the function $f\ : (x, y) \mapsto \frac{x}{y}$ restricted to the domain $\mathbb{R}_+ \times \mathbb{R}_{++}$. Since this function is smooth, one may check its convexity by computing its Hessian (see [Roc70, p.27])

$$\nabla^2 f(x, y) = \begin{pmatrix} 0 & -\frac{1}{y^2} \\ -\frac{1}{y^2} & 2\frac{x}{y^3} \end{pmatrix}$$

and seeing it is indefinite on f's domain, proving nonconvexity of f and S.

Let us suppose we want to model the following convex problem

$$\text{(CVX)} \qquad \min_{x \in \mathbb{R}^n} f_0(x) \quad \text{s.t.} \quad x \in D$$

where f_0 is a convex function and D a convex set. First of all, we have to make the objective linear since the objective of a conic program is always linear. We simply add a new scalar variable t and solve

$$\text{(CVXT)} \qquad \min_{(x,t) \in \mathbb{R}^n \times \mathbb{R}} t \quad \text{s.t.} \quad (x, t) \in E$$

where $E = (D \times \mathbb{R}) \cap \operatorname{epi} f_0$. Any feasible solution (x, t) will satisfy both $(x, t) \in D \times \mathbb{R}$ (which is equivalent to $x \in D$) and $(x, t) \in \operatorname{epi} f_0$ (which is equivalent to $f_0(x) \leq t$). From this we understand that the minimum value of t in (CVXT) will be equal to the minimum value of $f_0(x)$ in (CVX).

The important thing to notice is that E is still a convex set. D and $\mathbb{R}$ are convex, which implies $(D \times \mathbb{R})$ is convex. $\operatorname{epi} f_0$, being the epigraph of a convex function, is a convex set and E, the intersection of two convex sets, is again a convex set. This shows that we can suppose without any loss of generality that the objective of our convex problem is linear, as in

$$\text{(CVXL)} \qquad \min_{x \in \mathbb{R}^n} c^T x \quad \text{s.t.} \quad x \in D$$

Our next task is to identify which are convex sets we are able to model with conic programming involving real self-scaled cones. Such sets will be called *SQL-representable*[10]. We won't give a complete characterization of these sets but will proceed constructively. We will provide a set of *basic* SQL-representable sets as well as *operations* allowing us to build more complex SQL-representable sets.

Definition 1.20 *A convex set $D \subseteq \mathbb{R}^n$ is SQL-representable if there exists an integer $m \geq 0$, a real self-scaled cone $C \subseteq \mathbb{R}^{n+m}$ and an $n + m$-dimensional affine subspace $(b + L)$ such that*[11]

$$x \in D \Leftrightarrow \exists y \in \mathbb{R}^m \mid (x, y) \in C \cap (b + L)$$

(this condition will be denoted as $D \cong C \cap (b + L)$)

[10] SQL stands for Semidefinite – Quadratic – Linear, in reference to $\mathbb{R}^n_+$, $\mathbb{L}^n_+$ and $\mathbb{S}^n_+$

[11] Strictly speaking, a projection of C on $\mathbb{R}^{n+m}$ and/or a permutation of the coordinates of (x, y) may be necessary, but we'll ignore these formal details in order to keep our notations as clear as possible

The role of the y part is to allow the introduction of m additional real variables that are not present in the original problem but that may help to model it as a conic program. For example, the epigraph of the positive branch of an hyperbola $\{(t,u) \mid tu \geq 1\}$ we described above is a 2-dimensional SQL-representable set using a 3-dimensional cone ($\mathbb{S}^2_+$) and an affine set ($v = 1$ with our previous notations). We have indeed that

$$\mathbb{S}^2_+ \cap \left\{ \begin{pmatrix} t & v \\ v & u \end{pmatrix} \mid v = 1 \right\} = \left\{ \begin{pmatrix} t & 1 \\ 1 & u \end{pmatrix} \mid tu \geq 1 \right\} \cong \{(t,u,1) \mid tu \geq 1\}$$

which satisfies the above definition.

Theorem 1.21 *If D is SQL-representable with cone C and affine subspace $(b + L)$, problem (CVXL) is equivalent the following SQL conic program*

$$\text{(SQL)} \qquad \min_{(x,y)\in\mathbb{R}^{n+m}} (c,0)^T(x,y) \quad s.t. \quad (x,y) \in C \cap (b + L)$$

Proof Indeed, using the definition of an SQL-representable set, to each feasible solution of (CVXL) corresponds at least one feasible solution of (SQL) with the same objective value (left-to-right implication), while to each feasible solution of (SQL) corresponds a feasible solution of (CVXL) with the same objective value (right-to-left implication), which proves the equivalence. To recover the optimal solutions of (CVXL) one simply has to select the first n components of the optimal solutions of (SQL).

1.5.1 Operations on SQL-representable sets

The main three operations preserving SQL-representability are Cartesian product, affine restriction and intersection (see [BTN98] for a related discussion).

Theorem 1.22 *If $A \subseteq \mathbb{R}^n$ and $B \subseteq \mathbb{R}^m$ are two SQL-representable sets, their Cartesian product $A \times B$ is also a SQL-representable set.*

Proof Let $A \cong C_A \cap (b_A + L_A)$ and $B \cong C_B \cap (b_B + L_B)$. If we choose $C = C_A \times C_B$ and $(b + L) = \{(x_A, x_B) \mid x_A \in (b_A + L_A) \text{ and } x_B \in (b_B + L_B)\}$ (this is an affine subspace), we have that $A \cap B \cong C \cap (b + L)$.

Theorem 1.23 *If $A \subseteq \mathbb{R}^n$ is a SQL-representable set and $(c + M)$ an n-dimensional affine subspace, the restriction of A to $(c + M)$, i.e. their intersection $A \cap (c + M)$ is also a SQL-representable set.*

Proof Let $A \cong C_A \cap (b_A + L_A)$. If we choose $(b + L) = \{x \mid x \in (b_A + L_A) \text{ and } x \in (c + M)\}$ (this is an affine subspace), we have that $A \cap (c + M) \cong C_A \cap (b + L)$.

Theorem 1.24 *If $A \subseteq \mathbb{R}^n$ and $B \subseteq \mathbb{R}^n$ are two SQL-representable sets, their intersection $A \cap B$ is also a SQL-representable set.*

Proof Since $A \cap B$ may be viewed as the first n coordinates of the intersection of $A \times B$ with the affine subspace $\{(x_A, x_B) \mid x_A = x_B\}$, we have that $A \cap B$ is SQL-representable by virtue of Theorem 1.23 and Theorem 1.22.

1.5.2 Basic SQL-representable sets

The positive orthant itself $\mathbb{R}^n_+$ is of course SQL-representable. This implies that the set $\{x \in \mathbb{R}^n_+ \mid Ax = b\}$ (the feasible region of a standard form linear program) is SQL-representable by virtue of Theorem 1.23 (use the subspace $(c + M) = \{x \in \mathbb{R}^n \mid Ax = b\}$)

The second order cone itself $\mathbb{L}^n_+$ is of course SQL-representable. A recurrent problem with the linear programming formulation is the impossibility of introducing free variables in an elegant way, because all variables have to be nonnegative (one usually has to *split* a free variable x_i into a difference of nonnegative variables $x_i^+ - x_i^-$, which doubles the number of variables and can introduce numerical difficulties[12]. This problem is solved with SQL conic programming, since $\mathbb{L}^n_+ \cong \mathbb{R}^n$ (simply ignore the first coordinate in $\mathbb{L}^n_+$). This appears as a promising way to model free variables.

Before giving basic SQL-representable sets using $\mathbb{S}^n_+$, let us turn back to the question of the objective in our SQL conic program.

1.5.3 Modelling a nonlinear objective

Definition 1.25 *We say that f is a SQL-representable function if and only if its epigraph* $\operatorname{epi} f$ *is a SQL-representable set*

This definition is very useful since the next theorem shows that we can minimize any SQL-representable function on a SQL-representable set using SQL conic programming.

Theorem 1.26 *Let f_0 be a SQL-representable function and D a SQL-representable set. The following program*

$$\min_{x \in \mathbb{R}^n} f_0(x) \quad s.t. \quad x \in D$$

is equivalent to the following SQL conic program with linear objective

$$\min_{(x,t) \in \mathbb{R}^n \times \mathbb{R}} t \quad s.t. \quad (x,t) \in E$$

Proof As mentioned above, taking $E = (D \times \mathbb{R}) \cap \operatorname{epi} f_0$ makes both programs equivalent. We just have to prove that E is a SQL-representable set. Because of Theorem 1.24 and Definition 1.25, it is sufficient to show that $D \times \mathbb{R}$ is SQL-representable. But $\mathbb{R}$ is SQL-representable (using $\mathbb{L}^1_+$), which implies (Theorem 1.22) that $D \times \mathbb{R}$ and thus E are SQL-representable sets[13].

It is also possible to minimize a positive sum of SQL-representable functions.

Theorem 1.27 *If f and g are two SQL-representable functions on $\mathbb{R}^n$, $\alpha f + \beta g$ is also SQL-representable where $\alpha, \beta \in \mathbb{R}_+$*

[12] These difficulties are due to the fact that the feasible and the optimal solution sets are now unbounded, since any constant offset can be added to both x_i^+ and x_i^- without changing the objective value

[13] In the case of a positive function f, it is possible to choose $E = (D \times \mathbb{R}_+) \cap \operatorname{epi} f$ which is even easier to represent

Proof Let A be equal to the following cartesian product[14]

$$\text{epi } f \times \text{epi } g \times \mathbb{R} \ \subseteq (\mathbb{R}^n \times \mathbb{R}) \times (\mathbb{R}^n \times \mathbb{R}) \times \mathbb{R}$$

We have $(x,t,y,u,v) \in A \Leftrightarrow f(x) \leq t$ and $g(y) \leq u$. Adding two new linear equality constraints $v = \alpha t + \beta u$ and $x = y$, it is easy to check that $(x,t,y,u,v) \in A \Leftrightarrow \alpha f(x) + \beta g(x) \leq v$, which is precisely the epigraph of $\alpha f + \beta g$ (selecting coordinates x and v). Since Theorem 1.22 and Theorem 1.23 imply that A is SQL-representable, the theorem is proved.

1.5.4 More basic SQL-representable sets and functions

The epigraph of the Euclidean norm $\|x\|$ is SQL-representable, since it is precisely the Lorentz cone. Another interesting function is the squared Euclidean norm $\|x\|^2 = x^T x$, whose epigraph is representable via

$$\begin{aligned} x^T x \leq t &\Leftrightarrow x^T x + \frac{(t-1)^2}{4} \leq \frac{(t+1)^2}{4} \\ &\Leftrightarrow (\frac{t+1}{2}, x, \frac{t-1}{2}) \in \mathbb{L}^{n+1}_+ \\ &\Leftrightarrow (r,x,s,t) \in \mathbb{L}^{n+1}_+ \times \mathbb{R}_+ \text{ and } 2r-1=t, 2s+1=t \end{aligned}$$

because of Theorem 1.22 and Theorem 1.23.

We give, without proof, another SQL-representable sets using the second order cone : $\{x \in \mathbb{R}^n \mid x^T A x + b^T x + c\}$ where A is positive semidefinite, which is useful to model convex quadratic constraints.

The positive semidefinite cone itself $\mathbb{S}^n_+$ is of course SQL-representable. But there are many other matrix-oriented SQL-representable sets and functions.

For example, the largest eigenvalue of a matrix $\lambda_{max}(M)$, considered as a function of M, is SQL-representable. Its epigraph may be described as

$$\begin{aligned} \lambda_{max}(M) \leq t &\Leftrightarrow \lambda_{max}(M - tI) \leq 0 \quad \text{where } I \text{ is the identity matrix} \\ &\Leftrightarrow M - tI \preceq 0 \\ &\Leftrightarrow N \succeq 0 \text{ and } N = tI - M \end{aligned}$$

where the last line proves SQL-representability ($N = tI - M$ being a linear constraint). This may be useful to bound from above the maximal eigenvalue of a matrix.

Finally, we enumerate (without proof) other SQL-representable functions using the positive semidefinite cone : the spectral norm of a matrix, the sum of the k largest eigenvalues of a matrix, the sum of the k largest singular values of a rectangular matrix, fractional-quadratic constraints, nonnegative polynomials, etc.

We refer the reader to the course by Ben-Tal and Nemirovsky (see [BTN98]) for more information[15].

[14] The parentheses are present only for the sake of clarity

[15] It is worth noting the similarity between SQL-representability preserving operations and convexity preserving operations. We may think in fact that SQL-representable sets (resp. functions) are simply the convex sets (resp. functions) that are *easy to describe*, just as polyhedrons are for linear programming.

Chapter 2

Pattern separation

2.1 Problem description

The objective of this chapter is to use the SQL conic programming formulation from the previous chapter for pattern separation.

Let us suppose we are faced with a set of objects. Each of these objects is completely described by an n-dimensional vector. We call this vector a *pattern*. Each component in this vector is in fact a numerical characteristic of the objects. We assume that the only knowledge we have about an object is its pattern vector.

Let us imagine there is a natural way to group those objects into c classes. The pattern separation problem is simply the problem of separating these classes, i.e. finding a partition of the whole pattern space $\mathbb{R}^n$ into c disjoint components such that the patterns associated to each class belong to the corresponding component of the partition.

The main use for such a partition is of course classification : suppose we have some well-known objects that we are able to group into classes and some other objects for which we don't know the correct class. Our classification process will take place as follows

1. Separate the patterns of well-known objects. This is called the *learning* phase[1]
2. Use the partition found above to classify the unknown objects. This is called the *generalization* phase.

What is a good separation ? Of course a good algorithm should be able to separate correctly the well-known objects, but is only really useful if it classifies correctly the unknown patterns. The generalization capability is thus the ultimate criteria to judge a separation algorithm.

Here are a few examples of common classification tasks

- Medical diagnosis. This is one of the most important applications. The pattern vector represent various measures of a patient's condition (e.g. age, temperature, blood pressure, etc.). We'd like the separate the class of ill people from the class of healthy people.

[1] Some authors refer to it as *supervised* learning phase. In fact, one may want to separate patterns without knowing *a priori* the classes they belong to, which is then called *unsupervised* learning. This is in fact a *clustering* problem, completely different from ours, and won't be discussed further in this work.

- Species identification. The pattern vector represent various characteristics (color, dimensions, etc.) of a plant or animal. Our objective is to classify them into different species.
- Credit screening. A company is trying to evaluate applicants for a credit card. The pattern contains information about the customer (kind of job, monthly income, owns a house, etc.) and the goal is to identify the applicants for which it is safe to give a credit card from the others.

Patterns will be represented by column vectors belonging to $\mathbb{R}^n$. A set of m patterns will be represented by an $n \times m$ matrix.

2.1.1 Two-class separation versus multi-class separation

The general separation problem may involves more than 2 classes. However, it is possible to reduce the problem of separating c classes to c independent separation problems involving 2 classes each. This is done as follows : let $\mathcal{C} = \{C_i, i = 1 \ldots c\}$ be a set of c classes. For each class C_k, group all the remaining classes into one single class $D_k = \mathcal{C} \setminus C_k$ and separate class C_k from class D_k. The c partitions we find will allow us to test the membership to each class C_k separately.

To classify an unknown pattern, test its appartenance to each of the classes using the c partitions found. If the separation is correct, the pattern should be found to belong to only one class. If it belongs to zero or more than one classes, this means the generalization is not perfect and that the algorithm "hesitates" between two or more classes.

In the rest of this work, we will only consider two-class pattern separation problems.

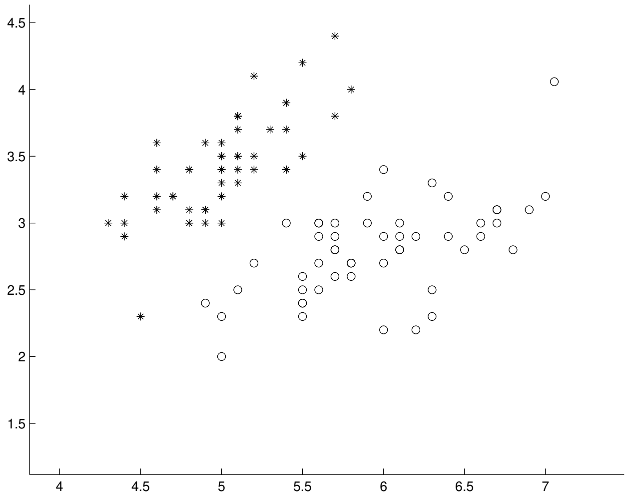
2.1.2 Computing the pattern vectors

In a classification task, the first step is to identify which characteristics are going to form the pattern vector. We distinguish various cases

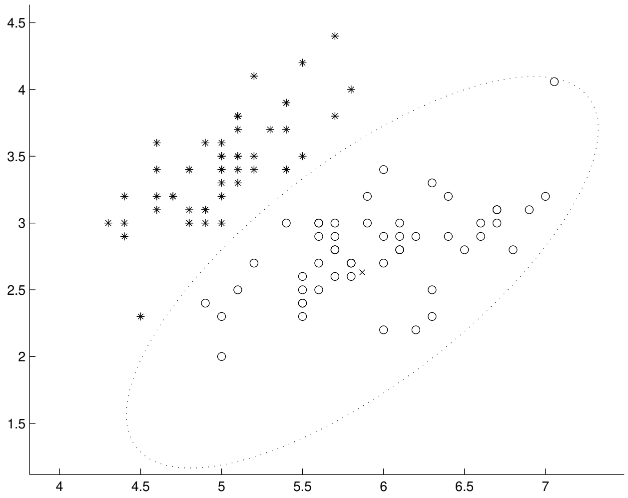
- A numerical continuous characteristic may be directly incorporated into the pattern vector.
- A binary characteristic (yes/no, true/false) may also be directly incorporated into the pattern vector if we choose two scalar values to represent the two possibilities (0/1 or $-1/+1$ for example). The effect of this apparently arbitrary choice will be investigated later (in section 2.2.3).
- A characteristic may be discrete, i.e. constrained to belong to a set of possible values. Here we distinguish two possibilities
  - As for the binary characteristic, we assign a scalar value to each possible value. This may only be done if we assume the values are ordered. A set of values like { 'Excellent', 'Good', 'Average', 'Weak', 'Hopeless' } may be represented by the values { 10, 8, 5, 3, 0 }. This choice is of course totally arbitrary and may have an impact on the separation process.
  - We may also decide to create a binary variable indicating each possible value taken by the characteristic. With the previous example, we would have a 5-dimensional vector whose first coordinate would be 1 only if the original characteristic was 'Excellent', 0 otherwise, This method has the advantage of being less arbitrary, but may increase the size of the pattern vector a lot.

2.1.3 Ellipsoid representation

Up to now, we do not have specified yet how we are going to describe our separating partitions. The main idea of this thesis is to use *ellipsoids* to separate our classes. This means that we would like to compute a separating ellipsoid such that the points from one class belong to the interior of the ellipsoid while the points from the other class lie outside this ellipsoid. Let us explain this idea on the following figures



This example is an easy bidimensional separation problem taken from a species classification data set (known as Fisher's Iris test set), using only the first two characteristics. The patterns from the first class appear as small circles, while the other class appear as small crosses. Computing a separation ellipsoid using the maximum separation ratio method (see Section 2.2) gives



We have decided to use ellipsoids for the following reasons :

- We expect patterns from the same class to be *close* to each other. This suggests enclosing them in some kind of hull, possibly a ball. But we also want our procedure to be scaling invariant. This is why we use the affine deformations of balls, which are the ellipsoids.

- Ellipsoids are the simplest convex sets (besides affine sets, which obviously do not fit our purpose)
- The set of points lying between two parallel hyperplanes is a (degenerate) ellipsoid. This means our separation procedures will generalize procedures using a hyperplane to separate patterns.
- We know that some geometrical problems involving ellipsoids can be modelled as SQL conic programs.

We now state the formal definition of an ellipsoid.

Definition 2.1 *An ellipsoid $\mathcal{E} \subseteq \mathbb{R}^n$ is a set described by a center $c \in \mathbb{R}^n$ and an $n \times n$ symmetric positive semidefinite matrix E such that*

$$\mathcal{E} = \{x \in \mathbb{R}^n \mid (x-c)^T E(x-c) \leq 1\}$$

The positive semidefiniteness of E is an important condition[2]. We already understand at this point that the positive semidefinite cone $\mathbb{S}^n_+$ will help us to model such an object.

2.1.4 Separating patterns

Our short presentation has avoided two difficulties that may arise with a pattern separation algorithm using ellipsoids, namely

1. Most of the time, the separating ellipsoid is not unique. How do we choose one ?
2. It may happen that there exists no separating ellipsoid.

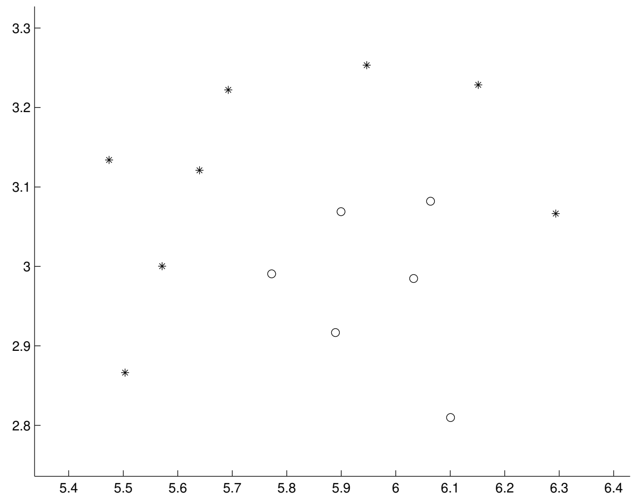
The way we solve these problems is to use optimization. Each ellipsoid is *a priori* a feasible solution. The objective function of our program will measure how well this ellipsoid separates our points. Ideally, non separating ellipsoids should have a high objective value (recall we always *minimize* our objective), while separating ellipsoids should have a lower objective value. With this kind of formulation, the conic program will always give us a solution, even when there is no separating ellipsoid.

The problem is thus to find such an objective function that adequately represents the quality of the ellipsoid separation. In the rest of this chapter, we present various attempts to meet this goal.

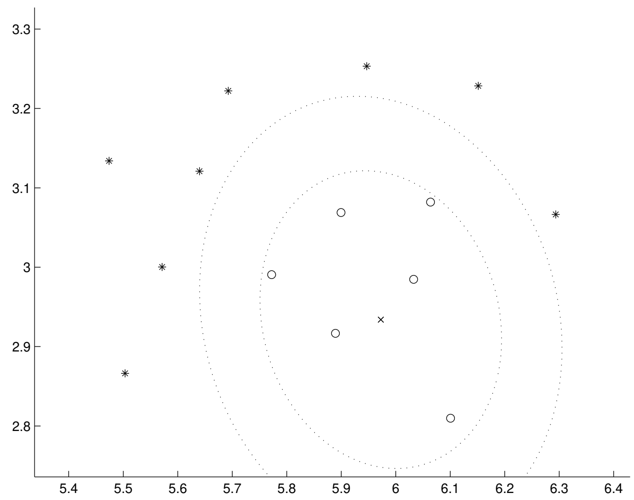
2.2 The maximal separation ratio method

Let us consider a very simple example

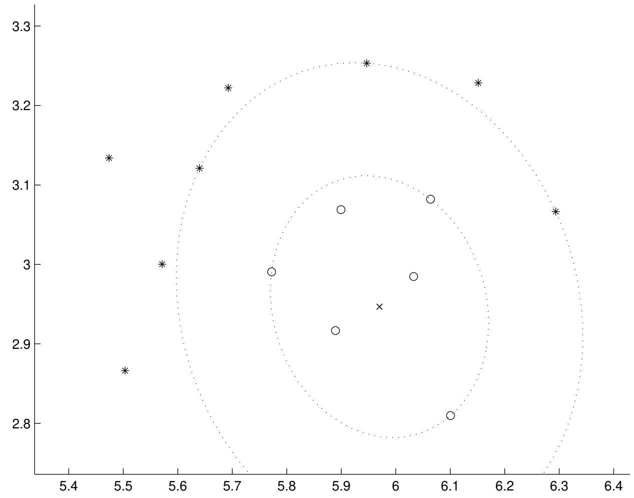
[2]When $E \not\succeq 0$, the equation above may describe any quadratic set



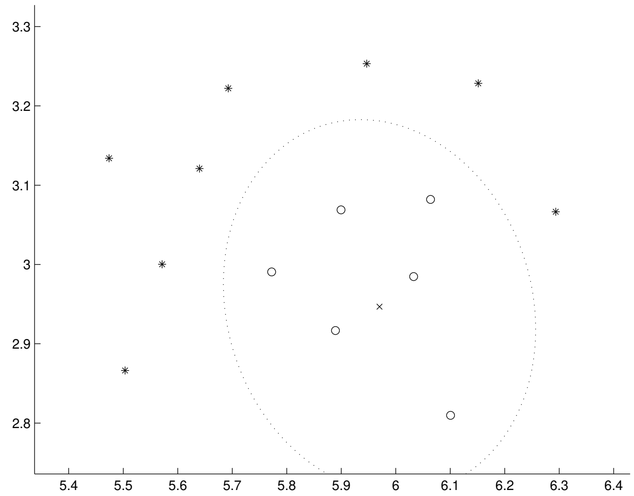
We want to include the small circles in an ellipsoid in order obtain the best separation from the small crosses. A way to express this is to ask for two different separating ellipsoids. We want these ellipsoids to share the same center and axis directions (i.e. we want them to be geometrically similar), but the second one will be larger by a factor ρ, which we will subsequently call the *separation ratio*. The next figure shows such a pair of ellipsoids with a ρ equal to $\frac{3}{2}$.



We now use the separation ratio to assess the quality of the separation : the higher the value of ρ, the better the separation. Our goal will be to maximize ρ over the set of separating ellipsoids. The next figure shows the optimal pair of ellipsoids, with the maximal ρ equal to 1.863.



Finally, we don't need these two ellipsoids, so we finally partition the pattern space using an intermediate ellipsoid whose size is the mean size of our two ellipsoids.



Let us now describe the conic program we use to optimize the separation ratio.

The variables we want to optimize are those describing the ellipsoid $\mathcal{E}$, i.e. the center c and the matrix E. Let $\{a_i\}$ be the n_a points to be included and $\{b_j\}$ be the n_b points to be excluded. We need to express the fact that our ellipsoid is separating the patterns, i.e. state that $a_i \in \mathcal{E}\ \forall i$ and $b_j \notin \mathcal{E}\ \forall j$, which is equivalent to

$$(a_i - c)^T E(a_i - c) \leq 1\ \forall i \text{ and } (b_j - c)^T E(b_j - c) > 1\ \forall j$$

In fact, because of our separation ratio objective, we want the points $\{b_j\}$ to be outside our ellipsoid enlarged with a factor ρ. Using the fact that this enlarged ellipsoid is simply described by

$$\mathcal{E}' = \{x \in \mathbb{R}^n \mid (x - c)^T E(x - c) \leq \rho^2\}$$

we get the following program

$$\max \rho \quad \text{s.t.} \quad \begin{cases} (a_i - c)^T E(a_i - c) \leq 1 \ \forall i \\ (b_j - c)^T E(b_j - c) \geq \rho^2 \ \forall j \\ E \in \mathbb{S}^n_+ \end{cases}$$

Unfortunately, it is not possible to formulate this program using SQL conic programming, since it is in general nonconvex ! In fact, the first set of constraints is already nonconvex, hence not SQL-representable.

To see that, just take the first constraint with $n = 1$ and $a_i = 0$. We get $c^2 E \leq 1$. The points $X_1 = (c_1, E_1) = (1, 1)$ and $X_2 = (c_2, E_2) = (\frac{1}{2}, 4)$ satisfy the constraint, but the point $X = \frac{X_1 + X_2}{2} = (\frac{3}{4}, \frac{5}{2})$ does not, which proves that this constraint doesn't define a convex set. How do we cope with that difficulty ?

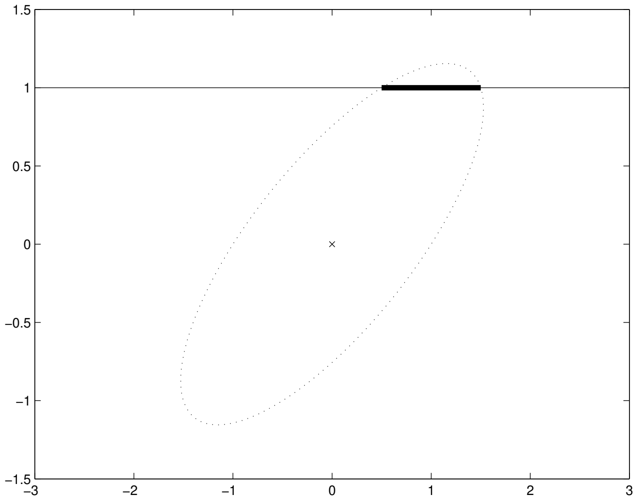
1. We may fix the center c. In that case, our constraint simply become linear in the components of E (because the a_i's are constants), and are of course SQL-representable. However, this solution raises the delicate question of the choice of c. We would indeed have to compute c with some kind of heuristics (e.g. centroid). Instead, we prefer to optimize c in order to get the best possible separation. This is why we developed another approach :

2. Use an homogeneous description of our ellipsoid.

2.2.1 Homogeneous description

We use the following simple fact : any n-dimensional ellipsoid $\mathcal{E}$ may be viewed as the projection onto $\mathbb{R}^n$ of the intersection of a $(n + 1)$-dimensional ellipsoid $\tilde{\mathcal{E}}$ centered at the origin with a correctly chosen hyperplane.

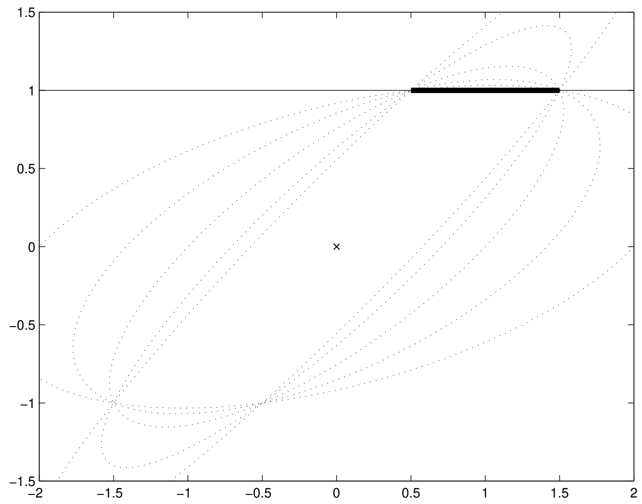
Let us illustrate these concepts for $n = 1$. A 1-dimensional ellipsoid is just a segment. If we consider the case where $c = 1$ and $E = 4$, i.e. $\mathcal{E} = \{x \in \mathbb{R} \mid 4(x - 1)^2) \leq 1\}$ we get the segment $[\frac{1}{2}, \frac{3}{2}]$. We see that this segment is the intersection of a centered 2-dimensional ellipsoid with the $y = 1$ hyperplane :



The idea is thus to optimize this centered $(n + 1)$-dimensional ellipsoid instead of E and c. As mentioned above, since its center is now fixed, our constraints become linear and hence are SQL-representable. Before going further, let us note two drawbacks of our formulation :

1. The concept of separation ratio does not smoothly extend to the $(n+1)$-dimensional ellipsoids. This means that a pair of n-dimensional ellipsoids with $\rho = 2$ does not lead to a pair of centered $(n+1)$-dimensional ellipsoids with the same separation ratio (this is straightforward to check on a small example).

2. There are infinitely many $(n+1)$-dimensional centered ellipsoids corresponding to each n-dimensional ellipsoid, as shown by the following figure



The whole family of ellipsoids may be parameterized by $\tilde{E} = \begin{pmatrix} \frac{3}{4}\lambda + 1 & -\lambda \\ -\lambda & \lambda \end{pmatrix}$.

However, as we will see later, our formulation will overcome both these difficulties. Let us go through the mathematical details. Assuming we know $\tilde{E}$, how can we get back to E and c ? First of all, note that $\tilde{E}$ has to be positive semidefinite in order to define an ellipsoid. Let us decompose $\tilde{E}$ as

$$\tilde{E} = \begin{pmatrix} s & v^T \\ v & F \end{pmatrix}$$

where F is a $n \times n$ matrix, v a vector of $\mathbb{R}^n$ and s is a scalar. We also compute d such that $v = -Fd$.

$$\begin{aligned} (1,x)^T \tilde{E}(1,x) &= s + 2x^T v + x^T F x \\ &= s - 2x^T F d + x^T F x \quad \text{(using the definition of } d) \\ &= (x-d)^T F (x-d) + s - d^T F d \\ &= (x-d)^T F (x-d) + \delta \end{aligned}$$

where we have defined $\delta = s - d^T F d$. We successively have

$$\begin{aligned} (1,x) \in \tilde{\mathcal{E}} &\Leftrightarrow (1,x)^T \tilde{E}(1,x) \leq 1 \\ &\Leftrightarrow (x-d)^T F (x-d) + \delta \leq 1 \\ &\Leftrightarrow (x-d)^T F (x-d) \leq 1 - \delta \\ &\Leftrightarrow (x-d)^T \frac{F}{1-\delta}(x-d) \leq 1 \end{aligned} \tag{2.1}$$

This proves that $\tilde{E}$ defines a n-dimensional ellipsoid described by a center c and a matrix E such that

$$c = d \text{ and } E = \frac{F}{1-\delta}$$

We also have the following useful equality

$$(1,x)^T\tilde{E}(1,x) = (1-\delta)(x-c)^T E(x-c) + \delta \tag{2.2}$$

This means that this number δ measures the difference between the ideal quantity $(x-c)^T E(x-c)$ and the one we are able to model $(1,x)^T\tilde{E}(1,x)$.

When $\delta = 0$, we will say that $\tilde{E}$ is the *canonical* homogeneous form of the ellipsoid described by c and E. In that case, the correspondance is much simpler, since we just have $c = d$, $E = F$ and

$$(1,x)^T\tilde{E}(1,x) = (x-c)^T E(x-c)$$

which would allow us to model the constraints $a_i \in \mathcal{E}$ and $b_j \notin \mathcal{E}$ exactly.

It is easy to see that each n-dimensional ellipsoid with center c and matrix E admits a unique canonical homogeneous form given by

$$\begin{pmatrix} c^T Ec & -Ec \\ -Ec & E \end{pmatrix}$$

Combining with the results of the previous paragraph, this implies the existence of a one-to-one correspondence between n-dimensional ellipsoids and canonical centered $(n+1)$-dimensional ellipsoids.

An interesting idea would be to work only with canonical homogeneous ellipsoids. In that case, we would be able to model our constraints exactly, including the definition of the separation ratio that would be equal for an n-dimensional ellipsoid and its $(n+1)$-dimensional canonical homogeneous counterpart.

In order to do that, we would like to add a constraint forcing δ to be zero. Unfortunately, this constraint is again nonconvex. To see it, check that both

$$M_1 = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \text{ and } M_2 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

are canonical while

$$M = \frac{M1 + M2}{2} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

is not. This means we are going to have to optimize over all homogeneous ellipsoids, not only canonical ones.

Before going further, let us investigate a little more the quantity δ defined earlier. The fact that $E \succeq 0$ implies that $F \succeq 0$ (because of the well-known theorem stating that the principal minors of a positive semidefinite matrix are also positive semidefinite). Applying Schur's Theorem 1.11 to $\tilde{E}$ we have that $s \geq v^T F^{-1} v$. Since $v = -Fd$ this implies

$$s \geq d^T F F^{-1} F d \Leftrightarrow s \geq d^T F d \Leftrightarrow \delta \geq 0$$

Using the fact that $F \succeq 0$ and inequality (2.1) above we also get

$$0 \leq (x-d)^T F(x-d) \leq 1 - \delta \Rightarrow \delta \leq 1$$

To conclude, we'll just remember that we always have $0 \leq \delta \leq 1$. Let us now turn back to our separation problem.

2.2.2 Conic formulation

With the homogeneous formulation, the constraints expressing separation become

$$(1,a_i)^T\tilde{E}(1,a_i) \leq 1 \ \forall i \text{ and } (1,b_j)^T\tilde{E}(1,b_j) \geq \rho^2 \ \forall j$$

which gives the following program

$$\max \rho \quad \text{s.t.} \quad \begin{cases} (1,a_i)^T\tilde{E}(1,a_i) \leq 1 \ \forall i \\ (1,b_j)^T\tilde{E}(1,b_j) \geq \rho^2 \ \forall j \\ \tilde{E} \in \mathbb{S}^{n+1}_+ \end{cases}$$

This is not exactly a conic program since the second set of constraints isn't linear because of the ρ^2 term. However, letting $k = \rho^2$ and noting that maximizing ρ is the same as minimizing $-k$ we get the final form

$$\text{(MAXSEP)} \qquad \min -k \quad \text{s.t.} \quad \begin{cases} (1,a_i)^T\tilde{E}(1,a_i) \leq 1 \ \forall i \\ (1,b_j)^T\tilde{E}(1,b_j) \geq k \ \forall j \\ \tilde{E} \in \mathbb{S}^{n+1}_+ \end{cases}$$

A priori this program doesn't really maximize the separation ratio as defined above because ρ is defined on the $(n+1)$-dimensional homogeneous ellipsoid $\tilde{\mathcal{E}}$ instead of the original n-dimensional ellipsoid $\mathcal{E}$. This difference becomes more apparent if we use Equation 2.2 to obtain an equivalent formulation of the linear constraints involving E, c and δ

$$(1,a_i)^T\tilde{E}(1,a_i) \leq 1 \ \Leftrightarrow (1-\delta)(a_i-c)^T E(a_i-c)+\delta \leq 1 \Leftrightarrow (a_i-c)^T E(a_i-c) \leq 1$$

and

$$(1,b_j)^T\tilde{E}(1,b_j) \geq k \Leftrightarrow (1-\delta)(b_j-c)^T E(b_j-c)+\delta \geq k \Leftrightarrow (b_j-c)^T E(b_j-c) \geq \frac{k-\delta}{1-\delta}$$

we restate (MAXSEP) as

$$\min -k \quad \text{s.t.} \quad \begin{cases} (a_i-c)^T E(a_i-c) \leq 1 \ \forall i \\ (b_j-c)^T E(b_j-c) \geq \frac{k-\delta}{1-\delta} \ \forall j \\ \tilde{E} \in \mathbb{S}^{n+1}_+ \end{cases}$$

This means that the value of the square of the separation ratio associated to the optimal ellipsoid is not k but $\frac{k-\delta}{1-\delta}$. However, the following theorem will give us some insight on the situation

Theorem 2.2 *If there exists a separating ellipsoid, the optimal solution of (MAXSEP) has $k^* > 1$ and the optimal homogeneous ellipsoid is canonical. If there is no separating ellipsoid, the optimal solution of (MAXSEP) has $k^* = 1$.*

This result is rather paradoxical. Although we are apparently not able to model the constraint $\delta = 0$ because of its nonconvexity, the optimal solution will always verify $\delta = 0$ and be canonical, except when there is no separating ellipsoid.

Proof If there exists a separating ellipsoid E, c such that

$$(a_i - c)^T E(a_i - c) \leq 1 \ \forall i \text{ and } (b_j - c)^T E(b_j - c) > 1 \ \forall j$$

we let $\tilde{E}$ be equal to the canonical homogeneous ellipsoid associated to E and c and

$$k = \min_j (b_j - c)^T E(b_j - c) > 1$$

and get a feasible solution with $k > 1$. This implies that the optimal value k^* is greater than 1 (recall we are maximizing k). Let us suppose that the optimal d^* is different from 0, i.e. the optimal ellipsoid $\tilde{E}^*$ is not canonical. Using the fact that $0 \leq d^* \leq 1$ and $k^* > 1$, we find

$$\frac{k^* - \delta^*}{1 - \delta^*} = \frac{k^*(1 - \delta^*) + \delta^*(k^* - 1)}{1 - \delta^*} = k^* + \frac{\delta^*(k^* - 1)}{1 - \delta^*} > k^*$$

This means that the canonical ellipsoid associated to E^* and c^* would be a feasible solution with a k greater than k^*, i.e. with an objective value lower than the optimum. This is impossible, and $\tilde{E}$ must be canonical.

Let us suppose now there exists no separating ellipsoid. If we had $k^* > 1$, applying the previous reasoning would prove that the canonical ellipsoid associated to E^* and c^* would be separating, which is impossible. Since the solution

$$\tilde{E} = \begin{pmatrix} 1 & 0_{1\times n} \\ 0_{n\times 1} & 0_{n\times n} \end{pmatrix}$$

is always feasible with $k = 1$, this proves that $k^* = 1$ (note that in that case δ is equal to 1).

Corollary 2.3 *If there exists a separating ellipsoid, the SQL conic program (MAXSEP) provides an ellipsoid with the highest possible separation ratio.*

Proof Since any separating ellipsoid with separation ratio $\rho > 1$ provides us with a feasible solution of (MAXSEP) having $k = \rho^2$ (via its canonical homogeneous representation), we have that $\rho^2 \leq k^* \Leftrightarrow \rho \leq \sqrt{k^*}$ for all separating ellipsoids. However, the optimal solution has a $\sqrt{k^*}$ separation ratio (since it is canonical), which proves that it has indeed the highest possible value.

2.2.3 Independence from the coordinate system

In the introduction, we said we chose ellipsoids because we wanted our separation process to be independent from the coordinate system of the pattern space. This means that if we transform the patterns using an invertible linear transformation, we want to find the same optimal separating ellipsoid, i.e. the original one transposed to the transformed space.

Affine-scaling invariance is a special case of this property. It is indeed very useful since the scale of the continuous characteristics is somewhat arbitrary. A separation procedure that would give different results with different scalings is probably less trustworthy than one which always gives the same results.

However, this property is stronger that affine-scaling invariance, since we allow any linear transformation (e.g. rotations, homothetic transformations, etc.).

Another fine consequence of this independence property is the fact that we can model a binary value with arbitrary scalar values. Indeed, if a binary characteristic takes the values a and b instead of the classical 0 and 1, the linear invertible transformation

$$x \mapsto \frac{x-a}{b-a}$$

establishes the equivalence between both possibilities.

A last consequence of this property is the possibility to model a discrete characteristic taking v different values with only $v-1$ binary variables.

Let us denote the v original binary variables by b_i, $i = 1 \ldots v$, each one corresponding to a possible value of the discrete characteristic. Using 0-1 variables for simplicity, we notice that

$$b_v = 1 - \sum_{i=1}^{v-1} b_i$$

Applying the following invertible linear transformation

$$b_v \mapsto b_v + \sum_{i=1}^{v-1} b_i$$

to our formulation transforms b_v to a variable always equal to 1. It is then clear that we may drop this variable for our separation process, since it provides no real information about the patterns, and we end up with $v-1$ binary variables.

We now prove that this property is true for our maximum separation ratio method.

Theorem 2.4 *Given two sets of points to be separated, the maximal separation ratio and the ellipsoid(s) that achieve it are independent from the coordinate system.*

Proof Although this may be intuitively understandable, we provide a mathematical proof. Let A be the $n \times n$ matrix associated to an invertible linear transformation on the pattern space. As always, let us denote by $\{a_i\}$ the points to be included and by $\{b_j\}$ the points to be excluded. Let center c and matrix E describe a separating ellipsoid with separation ratio ρ. We have the following

$$\begin{aligned}
& (a_i - c)^T E (a_i - c) \leq 1 \ \forall i \text{ and } (b_j - c)^T E (b_j - c) \geq \rho^2 \ \forall j \\
\Leftrightarrow \quad & (a_i - c)^T A^T A^{-T} E A^{-1} A (a_i - c) \leq 1 \ \forall i \text{ and } (b_j - c)^T A^T A^{-T} E A^{-1} A (b_j - c) \geq \rho^2 \ \forall j \\
\Leftrightarrow \quad & (Aa_i - Ac)^T (A^{-T} E A^{-1}) (Aa_i - Ac) \leq 1 \ \forall i \text{ and } (Ab_j - Ac)^T (A^{-T} E A^{-1}) (Ab_j - Ac) \geq \rho^2 \ \forall j
\end{aligned}$$

Denoting by $a'_i = Aa_i$, $b'_j = Ab_j$, $c' = Ac$ and $E' = A^{-T} E A^{-1}$, this is equivalent to

$$(a'_i - c')^T E' (a'_i - c') \leq 1 \ \forall i \text{ and } (b'_j - c')^T E' (b'_j - c') \geq \rho^2 \ \forall j$$

which states that the ellipsoid with center c' and matrix E' has at least a ρ separation ratio in the transformed pattern space[3]. Since the argument is reversible (going from E' to E), this proves that the maximal value of ρ must be equal for the original and for the transformed space.

[3] We should check that E' really defines an ellipsoid, i.e. $E' \succeq 0$. This is the case since $E \succeq 0 \Leftrightarrow x^T E x \geq 0 \ \forall x \Leftrightarrow x^T A^T A^{-T} E A^{-1} A x \geq 0 \ \forall x \Leftrightarrow (Ax)^T E' (Ax) \geq 0 \ \forall x \Leftrightarrow y^T E' y \geq 0 \ \forall y$

2.3 The minimum volume method

When there exists no separating ellipsoid, the maximum separation algorithm described above will return the following solution

$$\tilde{E} = \begin{pmatrix} 1 & 0_{1\times n} \\ 0_{n\times 1} & 0_{n\times n} \end{pmatrix}$$

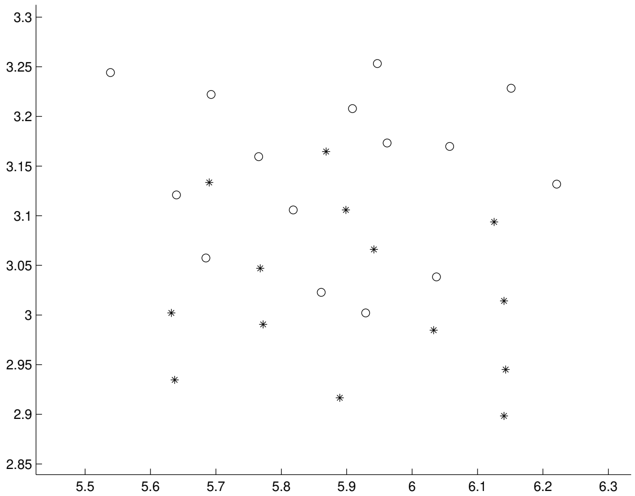
This ellipsoid is in fact highly degenerate, since it corresponds to the whole pattern space.

In that case, solving the SQL conic program doesn't provide us with any useful information. We could have expected something like a nearly separating ellipsoid. The purpose of the rest of this section is to design a procedure that won't suffer from this drawback and will provide us a usable ellipsoid, even when a separating ellipsoid doesn't exist. We first review what are the situations where the complete separation by an ellipsoid is not possible.

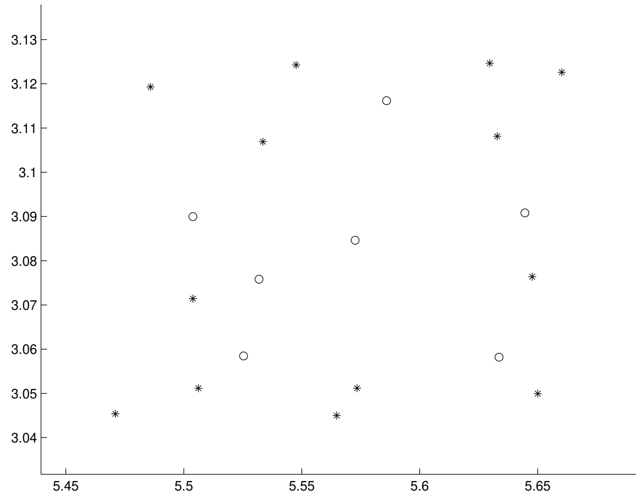
2.3.1 No separating ellipsoid

The absence of separating ellipsoid may happen in two different situations

1. One (or more) of the points that have to lie outside the ellipsoid (the b_j's) belongs to the convex hull of the points that have to lie inside the ellipsoid (the a_i's). If an ellipsoid contains all the a_i's, it will contain their convex hull (since a convex set contains the convex hull of any subset of its points, see [Roc70, p.11]) and thus won't be able to separate all the b_j's. This case is depicted on the following figure



2. Even when the situation described above doesn't occur, it might happen that there is no separating ellipsoid due to a special configuration of the patterns. Such an example is provided on the following figure



In this particular example, a separating set would have to approximate the pentagonal shape of the convex hull of the small circles, which is impossible with an ellipsoid.

A natural idea to handle such situations is to ignore completely the points from the outside class and simply try to find the smallest possible ellipsoid. Based on this idea, we developed the *minimum volume method.*

2.3.2 Modelling the volume of an ellipsoid

The volume of an n-dimensional ellipsoid described by a matrix E is proportional to $(\det E)^{-1/2}$. However, there is no simple way to model this function with an SQL conic program[4], so we tried to find an alternate way of modelling the size of our ellipsoid.

It is well known that the lengths of the semi-axes of an ellipsoid are equal to the square root of the inverse of the eigenvalues of E. We propose the following measure of the size of an ellipsoid

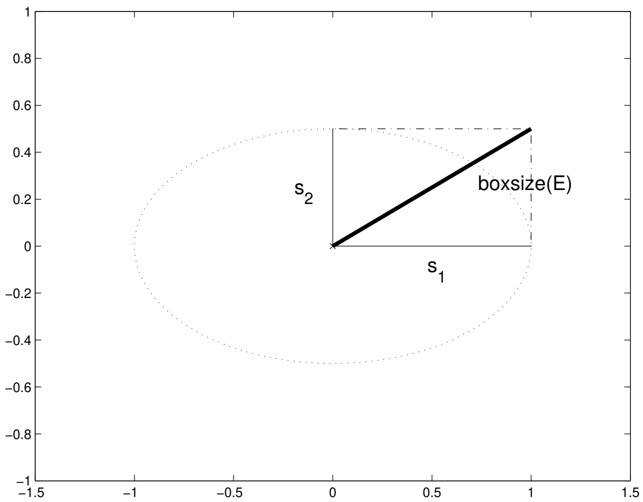
$$\text{boxsize(E)} = \sqrt{\sum_{i=1}^{n} \lambda_i(E)^{-1}}$$

Let us first explain the geometrical meaning of this measure. Using the above mentioned fact about the lengths of the semi-axes s_i, we find that

$$\text{boxsize(E)} = \sqrt{\sum_{i=1}^{n} s_i^2}$$

We believe the following figure is sufficient to grasp the meaning of boxsize(E)

[4]It is indeed possible to model the function $(\det E)^{-\frac{1}{2n}}$, whose minimization would fit our purpose, but this needs a rather awkward construction involving many additional variables.



We are now going to prove that this function is SQL-representable. Let us consider the following constraint

$$\begin{pmatrix} E & I \\ I & T \end{pmatrix} \succeq 0$$

where I is the $n \times n$ identity matrix. This constraint on matrices E and T is obviously SQL-representable. We have

$$\begin{aligned} \begin{pmatrix} E & I \\ I & T \end{pmatrix} \succeq 0 \quad &\Rightarrow \quad T \succeq E^{-1} \\ &\Leftrightarrow \quad T - E^{-1} \succeq 0 \\ &\Rightarrow \quad \text{trace } (T - E^{-1}) \geq 0 \\ &\Leftrightarrow \quad \text{trace } T - \text{trace } E^{-1} \geq 0 \\ &\Leftrightarrow \quad \text{trace } T \geq \text{trace } E^{-1} \end{aligned}$$

where we used successively Schur complement's theorem, the fact that the trace of a positive semidefinite matrix is nonnegative (since it is the sum of nonnegative eigenvalues) and the linearity of the trace. But we know that the eigenvalues of E^{-1} are simply the inverses of the eigenvalues of E, which implies that trace E^{-1} is precisely equal to boxsize(E)^2. Noting that trace T is a linear function of T, we study the following SQL conic program

$$\text{(BOX)} \qquad \min \ \text{trace } T \quad \text{s.t.} \qquad \begin{pmatrix} E & I \\ I & T \end{pmatrix} \succeq 0$$

According to the previous inequalities, any feasible solution of (BOX) will have an objective trace $T \geq$ trace E^{-1}, which implies that the optimum is also greater or equal to trace E^{-1}. However, since $T = E^{-1}$ is a feasible solution with an objective equal to trace E^{-1}, we see that the optimal value of (BOX) must be exactly trace E^{-1} [5].

[5]Since (BOX) is a convex program, this is in fact an alternate proof of the well-known fact that trace E^{-1} is a convex function of E (see [Bar82, Ros65])

We use this fact in the following conic program

$$\text{(MINVOL)} \qquad \min \operatorname{trace} T \quad \text{s.t.} \quad \begin{cases} (1, a_i)^T \tilde{E} (1, a_i) \le 1 \ \forall i \\ \tilde{E} = \begin{pmatrix} s & v^T \\ v & F \end{pmatrix} \in \mathbb{S}_+^{n+1} \\ \begin{pmatrix} G & I \\ I & T \end{pmatrix} \in \mathbb{S}_+^{2n} \\ G = F \end{cases}$$

This is indeed a SQL conic program since all the constraints are either linear of positive semidefinite. The idea is as follows : we optimize an n-dimensional ellipsoid using the homogeneous form $\tilde{E}$ we already used for our first method. The $(1, a_i)^T \tilde{E}(1, a_i) \le 1$ ensures that the a_i's belong to the ellipsoid, and the rest of the constraints try to reproduce the program (BOX) using this time F in place of E.

2.3.3 Conic formulation

Again we use Equation 2.2 to obtain an equivalent formulation of the linear constraints involving E, c and δ instead of $\tilde{E}$

$$\min \operatorname{trace} T \quad \text{s.t.} \quad \begin{cases} (a_i - c)^T E (a_i - c) \le 1 \ \forall i \\ \tilde{E} \in \mathbb{S}_+^{n+1} \\ \begin{pmatrix} (1-\delta)E & I \\ I & T \end{pmatrix} \in \mathbb{S}_+^{2n} \end{cases}$$

Using our result about the (BOX) program, this implies that the optimal value of this program will be

$$\operatorname{trace} \ ((1-\delta)E)^{-1} = \frac{\operatorname{trace} E^{-1}}{1-\delta}$$

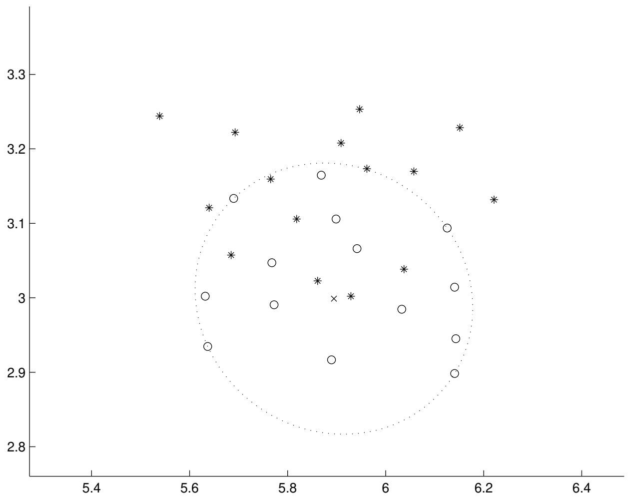
Obviously, δ will be zero in an optimal solution (otherwise decreasing δ, leaving E and c unchanged, would give a better objective value), which will give an optimum equal to

$$\operatorname{trace} E^{-1} = \text{boxsize(E)}^2$$

We finally have proved the following theorem

Theorem 2.5 *The optimal solution of the conic program (MINVOL) provides a canonical homogeneous ellipsoid representing the ellipsoid with minimum* boxsize *among all ellipsoids containing the patterns a_i's.*

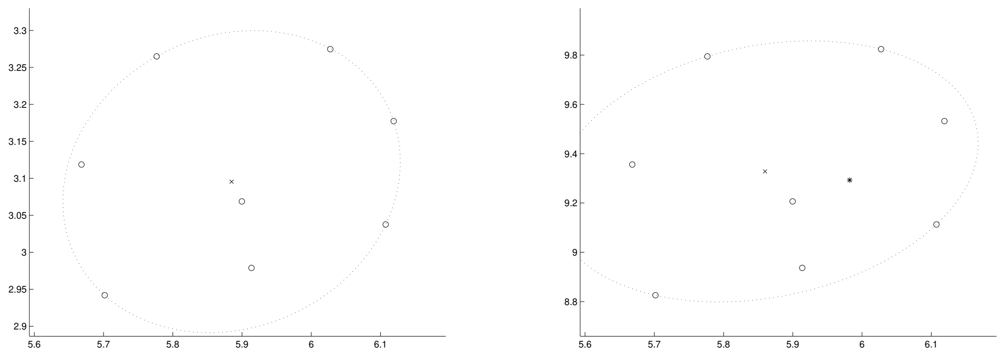
Here is such an ellipsoid with minimum boxsize, on one of the examples where the maximum separation method failed to provide any useful ellipsoid



2.3.4 Remarks

One of the improvements of this method over the maximal separation ratio is that it provides an ellipsoid even when there is no separating ellipsoid, but if suffers from two drawbacks

- The b_j's are not taken into account for the determination of the ellipsoid, which may be thought as rather weird for a separation procedure.
- This procedure is not independent from the coordinate system. In fact, it is even not affine scaling invariant, simply because the boxsize of an ellipsoid depends on the scaling. This can be easily checked on the following example



These two sets of patterns differ only by the scaling of the y coordinate, which is multiplied by 3 in the second figure (look at the scale of the y-axis). One easily understand that the second ellipsoid is wider because its height is more penalized (i.e. has more influence on the boxsize) with the second scaling.

2.3.5 The maximum sum separation method

One of the design goals of this method was to retain the main advantage of the minimum volume method, i.e. providing a useful solution when there is no separating ellipsoid while trying to use the b_j's and be independent from coordinate method.

The reason for the failure of the maximum separation ratio algorithm in the case there is no separating ellipsoid is the requirement that *all* the b_j's must lie outside the ellipsoid. As we already mentioned above, the presence of only one b_j pattern inside the convex hull of the a_i's is sufficient to make the algorithm return the spurious $k = 1$ solution (degenerate ellipsoid).

A simple method to avoid this is to require that *most of* the b_i's lie outside of the ellipsoid. In order to model this kind of constraint, we compute the separation ratios between the a_i's and each b_k separately (call it ρ_k) and require most of them to be as large as possible (as opposed to maximizing the smallest ρ_k in the case of the maximum separation ratio).

Of course, a constraint involving an expression like *most of* has no well-defined mathematical meaning. Our first suggestion to interpret it is to try to maximize the arithmetic mean of these separation ratios. This is of course equivalent to maximizing the sum of the ρ_j's (since there is a constant number of b_j's). However, as we've seen earlier, the square of the separation ratio is easier to model than the separation ratio itself. This is why we are going to maximize the sum of these squared separation ratios. We call this procedure the *maximum sum method*.

2.3.6 Conic formulation

This leads to the following conic program

$$\text{(SUMSEP)} \qquad \min -\sum_j k_j \quad \text{s.t.} \quad \begin{cases} (1,a_i)^T\tilde{E}(1,a_i) \le 1 \ \forall i \\ (1,b_j)^T\tilde{E}(1,b_j) = k_j \ \forall j \\ \tilde{E} \in \mathbb{S}_+^{n+1} \end{cases}$$

From a computational point of view, it is of course possible to simplify (SUMSEP) to a program with less variables constraints

$$\min \ -\sum_j (1,b_j)^T\tilde{E}(1,b_j) \quad \text{s.t.} \quad \begin{cases} (1,a_i)^T\tilde{E}(1,a_i) \le 1 \ \forall i \\ \tilde{E} \in \mathbb{S}_+^{n+1} \end{cases}$$

since $\sum_j (1,b_j)^T\tilde{E}(1,b_j)$ is a linear function of E, but we will stick to the (SUMSEP) formulation for the sake of clarity.

Using again the E, c and δ representation of Equation 2.2, we have

$$\min -\sum_j k_j \quad \text{s.t.} \quad \begin{cases} (a_i-c)^TE(a_i-c) \le 1 \ \forall i \\ (b_j-c)^TE(b_j-c) = \frac{k_j-\delta}{1-\delta} \ \forall j \\ \tilde{E} \in \mathbb{S}_+^{n+1} \end{cases}$$

which is again equivalent to

$$\min -\sum_j \big((1-\delta)k'_j + \delta\big) \quad \text{s.t.} \quad \begin{cases} (a_i-c)^TE(a_i-c) \le 1 \ \forall i \\ (b_j-c)^TE(b_j-c) = k'_j \ \forall j \\ \tilde{E} \in \mathbb{S}_+^{n+1} \end{cases}$$

letting $k'_j = \frac{k_j-\delta}{1-\delta}$.

Theorem 2.6 *The optimal homogeneous ellipsoid of (SUMSEP) is canonical.*

Proof We use exactly the same line of reasoning as for the maximum separation method (cf. Theorem 2.2). First recall that there always exists a feasible (but spurious) solution with

$$\tilde{E} = \begin{pmatrix} 1 & 0_{1\times n} \\ 0_{n\times 1} & 0_{n\times n} \end{pmatrix}$$

such that all k_j's are equal to 1. Since its objective value is equal $-\sum_j 1 = -n_b$, the optimal value of (SUMSEP) is less or equal to $-\sum_j 1 = -n_a$.

Let us suppose there the optimal value of δ is different from zero. This would imply that

$$-\sum_j \left((1-\delta^*)k_j'^* + \delta^*\right) < -\sum_j k_j'^*$$

since the left-hand side is the optimal value of (SUMSEP) and the right-hand side corresponds to the objective value of a feasible solution, namely the canonical homogeneous ellipsoid associated to the optimal E^* and c^*, having $\delta = 0$). This is equivalent to

$$\begin{aligned}
& -\sum_j \left(\delta^*(1-k_j'^*)\right) < 0 \\
\Leftrightarrow\quad & -\sum_j \left(1-k_j'^*\right) < 0 \\
\Leftrightarrow\quad & -\sum_j 1 < -\sum k_j'^* \\
\Leftrightarrow\quad & -\sum_j (1-\delta^*) < -\sum \left((1-\delta^*)k_j'^*\right) \\
\Leftrightarrow\quad & -\sum_j (1-\delta^*) - \sum_j \delta^* < -\sum_j \left((1-\delta^*)k_j'^* + \delta^*\right) \\
\Leftrightarrow\quad & -\sum_j 1 < -\sum_j \left((1-\delta^*)k_j'^* + \delta^*\right)
\end{aligned}$$

This means that the optimal objective value of (SUMSEP) is greater than $-\sum_j 1 = n_b$, which is a contradiction. This proves that $\delta = 0$.

We also have the following corollary

Corollary 2.7 *The SQL conic program (SUMSEP) provides an ellipsoid with the highest possible sum of squared separation ratios.*

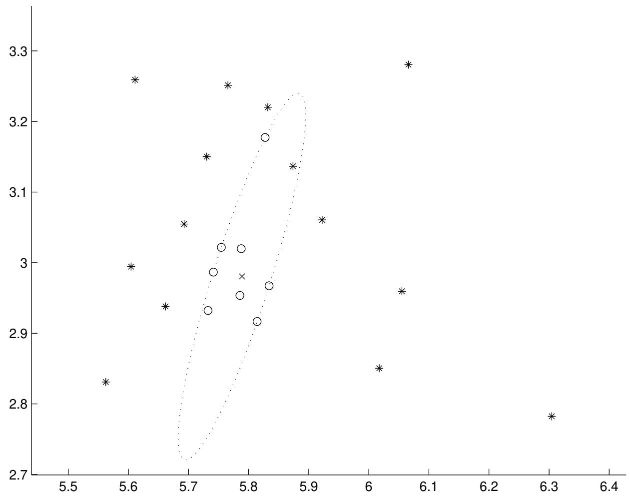
Since its proof completely mimics the one for Corollary 2.3, we do not reproduce it here.

It is also straightforward to check that this method is independent from the coordinate system, since the objective depends only on the separation ratios, which are themselves independent from the coordinate system (the demonstration is completely similar to the one for Theorem 2.4).

Theorem 2.8 *Given two sets of points to be separated, the ellipsoid(s) that achieves the maximum sum of squared separation ratios is independent from the coordinate system.*

2.3.7 Remark on separation

It is worth noting that no constraint forces the ellipsoid provided by this algorithm to be separating. The objective drives the sum of the k_j toward its maximum, but does not guarantee separation, which occurs only if all k_j's are greater than 1. In fact, it is even possible to find a non-separating ellipsoid when separation is indeed possible, as is shown on the following figure



One may analyse this method a bit further and give some insight on why this phenomenon is rather likely to appear. We maximize the sum of the k_j, which is equivalent to maximizing the sum of the squared separation ratios, which is in turn equivalent to maximizing the quadratic mean of the separation ratios

$$\max \sqrt{\frac{\sum_j \rho_j^2}{n_b}}$$

It is in fact well-know that the quadratic mean is much more influenced by its large elements than by the small ones. This is why the optimization process will mainly concentrate on increasing the largest separation ratios, rather than ensuring that the small ones do not become too small. This may be considered as a drawback, since these small ratios may fall under one, and prevent the ellipsoid from separating the patterns. The next and last method we present is an attempt to reduce this undesirable effect.

2.4 The minimum squared sum method

2.4.1 Generalized mean

Let us recall some basic facts about the *generalized mean.*

Definition 2.9 *Let $\{u_i, i = 1 \dots n\}$ be a set of n positive reals. Their q-generalized mean is defined as*

$$M_q(u_1, \dots, u_n) = \left(\frac{\sum_{i=1}^n u_i^q}{n} \right)^{\frac{1}{q}}$$

where q is any real belonging to $\mathbb{R} \cup \{-\infty, +\infty\}$. The cases where q is equal to $-\infty$, 0 and $+\infty$ are to be understood as limits.

This is the generalization of some well-known means. The case $q = 1$ corresponds to the usual arithmetic mean, the case $q = 2$ gives the quadratic mean and the case $q = -1$ is equal to the harmonic mean. The following theorem summarizes the main properties of this mean.

Theorem 2.10 *The q-generalized mean of a fixed set of reals is an increasing monotonic function of q* [6]*. Moreover, it is equal to the minimum of the reals when $q = -\infty$, to their maximum when $q = +\infty$ and to their geometric mean when $q = 0$.*

2.4.2 Conic formulation

From the previous discussion about the drawback of the maximum sum method, we understand that we would like to maximize a quantity depending on the separation ratios that would be more influenced by the small values than by the large ones. A generalized mean with a low (negative) parameter q will fit that purpose[7]. It is rather straightforward to model the case $q = -2$ with an SQL conic program, but we were able to model the even better case where $q = -4$. We call this last method the *minimum squared sum method.*

Maximizing the -4-generalized mean of the separation ratios ρ_k is equivalent to

$$\begin{aligned} \max M_{-4}(\rho_1, \ldots, \rho_{n_b}) &\Leftrightarrow \max \left(\sum_j \rho_j^{-4} \right)^{-\frac{1}{4}} \\ &\Leftrightarrow \min \sum_j \rho_j^{-4} \\ &\Leftrightarrow \min \sum_j \left(\frac{1}{\rho_j^2} \right)^2 \end{aligned}$$

We already know how to model a quantity like ρ_j^2 : we use the following pair of constraints

$$\left\{ \begin{array}{l} (1, a_i)^T \tilde{E}(1, a_i) \leq 1 \ \forall i \\ (1, b_j)^T \tilde{E}(1, b_j) = k_j \ \forall j \end{array} \right.$$

Modelling the inverse quantity, i.e. ρ_j^{-2}, is relatively easy. Permuting some of the variables we get

$$\left\{ \begin{array}{l} (1, a_i)^T \tilde{E}(1, a_i) = k_i \ \forall i \\ (1, b_j)^T \tilde{E}(1, b_j) \geq 1 \ \forall j \end{array} \right.$$

It is rather clear that this set constraints implies that each a_i will be separated from the b_j's with a squared separation ratio equal to $1/k_i$, which is equivalent to $k_i = \rho_i^{-2}$. Maximizing the -4-generalized mean of the separation ratios leads thus to the following program

$$\min \sum_i k_i^2 \quad \text{s.t.} \quad \left\{ \begin{array}{l} (1, a_i)^T \tilde{E}(1, a_i) = k_i \ \forall i \\ (1, b_j)^T \tilde{E}(1, b_j) \geq 1 \ \forall j \\ \tilde{E} \in \mathbb{S}_+^{n+1} \end{array} \right.$$

[6] It is even strictly monotonic provided the reals are not all equal

[7] The ideal value would be of course $q = -\infty$. This corresponds in fact to maximizing the minimum separation ratio, which is exactly what does the first method we described. Recall that one drawback of this method is that the constraints are very stringent, since *all* the b_j's must lie outside the ellipsoid

This is not yet a SQL conic program, because the objective is not linear. However, using the second order cone will do the trick. We first incorporate the objective into the constraints

$$\min t^2 \quad \text{s.t.} \quad \begin{cases} (1,a_i)^T\tilde{E}(1,a_i) = k_i \ \forall i \\ (1,b_j)^T\tilde{E}(1,b_j) \geq 1 \ \forall j \\ \tilde{E} \in \mathbb{S}_+^{n+1} \\ \sum_i k_i^2 \leq t^2 \end{cases}$$

Finally, noting that minimizing t^2 is the same as minimizing t and that the last constraint is equivalent to $(t, k_1, \ldots, k_{n_a}) \in \mathbb{L}_+^{n_a+1}$, we get the final SQL conic program

$$\text{(SQSEP)} \qquad \min t \quad \text{s.t.} \quad \begin{cases} (1,a_i)^T\tilde{E}(1,a_i) = k_i \ \forall i \\ (1,b_j)^T\tilde{E}(1,b_j) \geq 1 \ \forall j \\ \tilde{E} \in \mathbb{S}_+^{n+1} \\ (t, k_1, \ldots, k_{n_a}) \in \mathbb{L}_+^{n_a+1} \end{cases}$$

As usual, we introduce the alternate variables E, c and δ. We also denote $\frac{k_i-\delta}{1-\delta}$ by k_i' to get the following equivalent program

$$\min t \quad \text{s.t.} \quad \begin{cases} (a_i-c)^T E(a_i-c) = k_i' \ \forall i \\ (b_j-c)^T E(b_j-c) \geq 1 \ \forall j \\ \tilde{E} \in \mathbb{S}_+^{n+1} \\ (t, k_1, \ldots, k_{n_a}) \in \mathbb{L}_+^{n_a+1} \end{cases}$$

The optimal t will obviously satisfy

$$t^{*2} = \sum_i k_i^{*2} = \sum_i \left((1-\delta^*)k_j'^{\,*} + \delta^*\right)^2$$

Theorem 2.11 *The optimal homogeneous ellipsoid of (SQSEP) is canonical.*

We skip the proof of this theorem, since it is rather technical, involving the same kind of ideas and inequalities as the corresponding proof for the (SUMSEP) program.

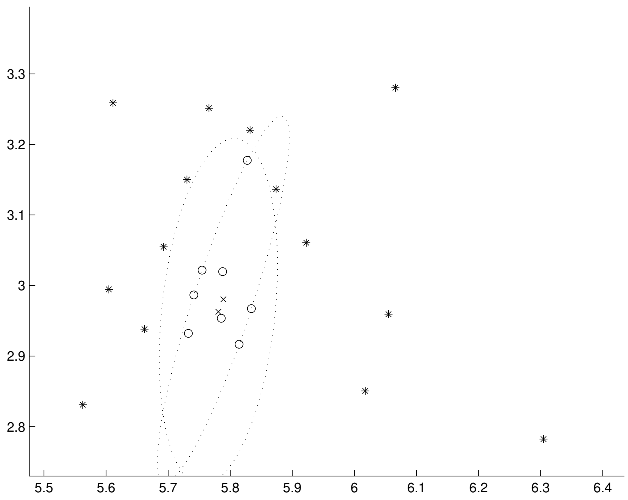
We also have the following corollary

Corollary 2.12 *The SQL conic program (SQSEP) provides an ellipsoid with the highest possible -4-generalized mean of separation ratios.*

Since its proof completely mimics the one for Corollary 2.3, we don't reproduce it here either. It is also straightforward to check that this method is independent from the coordinate system, since the objective depends only on the separation ratios, which are themselves independent from the coordinate system.

Theorem 2.13 *Given two sets of points to be separated, the ellipsoid(s) that achieves the maximum -4-generalized mean of separation ratios is independent from the coordinate system.*

To conclude the description of this method, we provide an example backing our claim that this algorithm reduces the drawback of the previous method.



On this figure, the thinnest ellipsoid was computed with the maximum sum separation algorithm and includes a misclassified cross, while the other ellipsoid was computed with the minimum squared sum separation method and is perfectly separating the patterns.

2.5 Exploitation

2.5.1 Basic strategy

The four methods we have discussed above provide us with a mean of computing an ellipsoid that separates the a_i's from the b_i's.

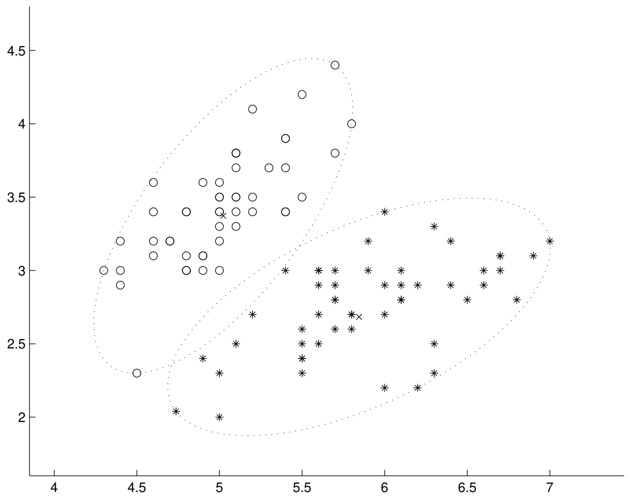
In order to use this ellipsoid $\mathcal{E}$, i.e. to try to classify an unknown pattern p, we simply test whether p belongs to $\mathcal{E}$, i.e. compute the quantity

$$p_{\mathcal{E}} = (p - c)^T E(p - c)$$

If it is less or equal to one, our procedure assigns p to the class of the a_i's, otherwise it assigns p to the class of the b_j's.

We may also introduce some kind threshold $\alpha > 1$ in order to leave the unknown pattern unclassified when it lies near the boundary of the ellipsoid (i.e. the algorithm "hesitates" between the two classes). For example, we may assign p to the class of the a_i's when $p_{\mathcal{E}} \leq \frac{1}{\alpha}$, assign it to the class of the b_j's when $p_{\mathcal{E}} \geq \alpha$ and leave it unclassified otherwise.

A obvious characteristic of our separation problem is that it is not symmetric between the a_i's and the b_i's. This suggests that we may compute a second separating ellipsoid $\mathcal{E}'$ that would include the b_i's and leave the a_i's outside. These two ellipsoids are illustrated on the following figure (using the minimum volume method)



The results of the classification of unknown patterns may of course vary according to the ellipsoid we choose. In our numerical experiments, we computed both ellipsoids $\mathcal{E}$ and $\mathcal{E}'$ for each classification tasks and evaluated the classification error for each of the ellipsoids.

2.5.2 Mixed strategy

We also tested a third approach using both ellipsoids. When examining the previous figure, we see that it is possible that a test pattern belongs to either both or none of the two ellipsoids. This just means that the two ellipsoids contradict each other, i.e. predict a different class for the test pattern.

Our third method goes as follows. First compute both $p_{\mathcal{E}}$ and $p_{\mathcal{E}'}$, then assign the test pattern to the class of a_i's if $p_{\mathcal{E}} < p_{\mathcal{E}'}$ or the class of b_j's otherwise.

It is worth noting that when the first two methods agree, this third method also gives the same result. Indeed, we have in that case one of the quantities $p_{\mathcal{E}}$ and $p_{\mathcal{E}'}$ less than one and the other one greater than one, which implies that our third decision rule will also predict the same class for the test pattern.

When the two ellipsoid disagree, for example when the test pattern belongs to both $\mathcal{E}$ and $\mathcal{E}'$, this third method will classify it in the ellipsoid to which it belongs the most, in the sense of p being "more inside" (in a relative scale depending from each ellipsoid).

We tested the three approaches with each data set, and the results are reported in Section 3.2.1.

2.6 State of the art

We investigated the existing literature in order to find whether ellipsoids had already been used in pattern separation and found only two relevant references :

- Rosen ([Ros65]) develops a method similar to our minimum volume algorithm (i.e. he uses boxsize(E) as a measure of the size of his ellipsoids). However, his formulation doesn't include any constraint on the positive semidefiniteness of E, which is only checked *a posteriori*. He mentions a computational experiment involving the classification of normal and abnormal vectorcardiograms (33 patterns belonging to $\mathbb{R}^3$) but does't report any accuracy results.

- Barnes ([Bar82] builds on Rosen's formulation and explicitly adds a positive semidefiniteness constraint on E. He only reports a very small computational experiment involving 7 patterns in $\mathbb{R}^2$.

Both these authors develop specialized algorithms for solve their problem, as opposed to our methods that use standard SQL conic programming. Their numerical experiments only report the efficiency of their algorithm in finding a minimum volume ellipsoid, and do not test the validity of the ellipsoid they compute to classify patterns.

Chapter 3

Computational experiments

3.1 Description of the tests

3.1.1 Implementation

As mentioned above, the real utility of a pattern separation algorithm is related to its generalization capability. In order to test our methods, we implemented them and ran them on various test sets. The purpose of this chapter is to report these results, comment on them and compare our methods to other existing classification procedures.

We decided to program our methods using the MATLAB environment, mainly for two reasons

1. MATLAB programming allows very short development cycles, which allows us to test new ideas rather quickly.

2. A MATLAB package called SDPPack (see [FA$^+$97]) has been designed by F. Alizadeh, J.-P. A. Haeberly, M. V. Nayakkankuppann, M. L. Overton and S. Schmieta to solve SQL conic programs. We decided to use that package rather than reinventing the wheel.

All the computational experiments reported here were conducted on a standard PC (200 MHz Pentium running Windows NT). The source code is available from the author.

3.1.2 Test data

We report here the results we obtained with four different data sets. Here is a short description for each of them

1. **Fisher's Iris**. Each pattern vector contains the petal length, sepal length, petal width and sepal width of an iris plant. There are three different classes representing three different types of iris plant. Each class contains 50 patterns. Since our methods only apply to problems with two classes, we first separated class 1 and class 2, then class 2 and class 3.

2. **Wisconsin Breast Cancer**. Each pattern vector is specified by 9 numerical attributes describing a cytological test carried out on a breast tumor. The objective is to predict the benign or malignant nature of the tumor. This data set consists in 683 patterns.

3. **Boston Housing**. This data set contains 596 patterns describing housing values in the suburbs of Boston, depending on various observations consisting of one binary and 12 continuous characteristics. The objective is to predict whether the housing value is above or below the median.

4. **Pima Indians Diabetes**. This data set consists of 768 patterns corresponding to patients. All patients are women of Pima Indian heritage. Each one is described by 8 numerical attributes. The objective is to predict whether a patient is showing signs of diabetes.

These four sets were obtained via the Repository of Machine Learning Databases and Domain Theories maintained by the University of California at Irvine ([MM98]). They were selected mainly because of the availability of testing results for other procedures.

3.1.3 Testing methodology

To evaluate the accuracy of a method we proceeded as follows. To conduct an *experiment*, we divide a data set into two parts : a *learning set* and a *validation set*. Our methods are then applied to the learning set. The algorithm has absolutely no knowledge about the patterns from the validation set, which will be used to test its accuracy. This is called *cross-validation*.

The division between learning set and validation set is made in a completely random fashion, the only parameter being the size of the learning set we want, given as a percentage of the total size of the data set. However, this selection was done such that the learning set and the validation set have roughly the same distribution of patterns between the two classes.

Once the ellipsoids have been computed, they are used to classify the patterns from the learning set. The learning error percentage we report is the percentage of misclassified learning patterns. The next step is to use the ellipsoids to classify the patterns from the validation set. The testing error percentage we report is the percentage of misclassified validation patterns.

Of course these percentages can vary very much from one experiment to another, depending on the choice of the learning and validation sets. In order to report representative results, we ran several experiments using the same data set and give the average figures. We also report the standard deviation of these error percentages.

3.2 Results

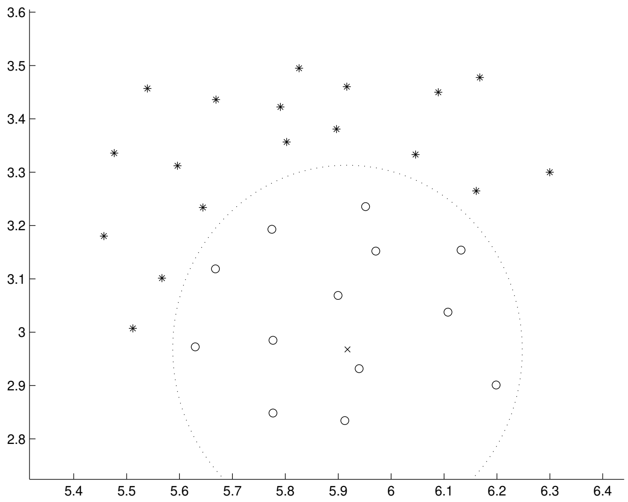
The main objective of these computational experiments is to discover whether one of the proposed methods dominates the other ones, and to compare their performance with other classification procedures.

3.2.1 Exploitation method

For nearly all our experiments, the mixed strategy (i.e. using both ellipsoids to classify a pattern) proved to be much better than using only one ellipsoid. For example, here are the complete results for separation on the Wisconsin Breast Cancer data set with a 20 % learning set (average on 10 experiments). For each category (*Learning* and *Testing*), the first two columns report the single ellipsoid strategies results, while the third column reports the mixed strategy results.

	Learning			Testing		
Method	Ellip. 1	Ellip. 2	Mixed	Ellip. 1	Ellip. 2	Mixed
Max. separation ratio	0.00 %	0.00 %	0.00 %	6.83 %	8.13 %	5.60 %
Min. volume	1.61 %	9.16 %	1.90 %	8.23 %	15.45 %	5.05 %
Max. sum	6.64 %	1.17 %	2.20 %	12.34 %	9.61 %	5.58 %
Min. squared	0.07 %	0.29 %	0.15 %	7.32 %	5.07 %	5.42 %

If we look at the learning errors, we see that one of the single ellipsoid strategies usually performs better than the mixed strategy, but not much, while the other single ellipsoid strategy performs a lot worse. This can be understood as one of the classes has usually a natural inclination to be included in an ellipsoid, while the other class is often some kind of *complement*, which has not always the ellipsoidal shape. This kind of situation is schematically depicted on the following figure, where the circles possess a natural ellipsoidal shape while the crosses don't.



However, if we examine the testing errors, which reflect the true performance of the methods, we see that the mixed strategy nearly always outperforms the single ellipsoid strategies (the only exception is on the last line, where Ellip. 2 classified marginally better than the mixed strategy). We have therefore decided only to report learning and testing results with the mixed strategy.

3.2.2 Comparison of the four methods

We tested each of the four methods on each of the five classification tasks described above, both with a 20 % learning set (average of 10 experiments) and with a 50 % learning set (average of 4 experiments). Here are the results of our experiments, featuring the error rates and their standard deviations both for the learning set and the testing set. In order to make the tables more readable, we highlighted the lowest error percentage in each column.

Fisher's Iris (class 1 from class 2)

This data set is known to be easily linearly separable. As our methods generalize separating hyperplane methods (cf. Subsection 2.1.3), we expected very good results, which is confirmed by the following table

	20 % learning set				50 % learning set			
	Learning		Testing		Learning		Testing	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Max. sep. ratio	**0.00** %	0.0 %	**0.00** %	0.0 %	**0.00** %	0.0 %	**0.00** %	0.0 %
Min. volume	**0.00** %	0.0 %	**0.00** %	0.0 %	**0.00** %	0.0 %	**0.00** %	0.0 %
Max. sum	**0.00** %	0.0 %	**0.00** %	0.0 %	**0.00** %	0.0 %	**0.00** %	0.0 %
Min. squared sum	**0.00** %	0.0 %	**0.00** %	0.0 %	**0.00** %	0.0 %	**0.00** %	0.0 %

All our experiments separated both the learning and the testing patterns perfectly. This example isn't really representative of the difficulty of real-life classification tasks.

Fisher's Iris (class 2 from class 3)

This data set is more challenging than the previous one, since it is known that there exists no separating hyperplane.

	20 % learning set				50 % learning set			
	Learning		Testing		Learning		Testing	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Max. sep. ratio	**0.00** %	0.0 %	7.50 %	3.5 %	**0.00** %	0.0 %	7.00 %	3.5 %
Min. volume	1.00 %	3.2 %	**5.25** %	1.3 %	2.00 %	1.6 %	**4.00** %	1.6 %
Max. sum	**0.00** %	0.0 %	8.00 %	3.2 %	2.00 %	2.8 %	7.50 %	4.1 %
Min. squared sum	**0.00** %	0.0 %	7.25 %	3.2 %	0.50 %	1.0 %	7.00 %	3.8 %

Our methods often perform very well on the learning set, with error percentages ranging from 0 % to 2 %, which corresponds to a single or no misclassified pattern. The results are better for the 20 % learning set, since there are less constraints to satisfy. The *maximum separation ratio* method performed perfectly on the learning set, which is easy to understand if one recalls that it concentrates on finding a separating ellipsoid.

The error rates for the test patterns range from 4 % to 8 %. The best method is without any discussion the *minimum volume* algorithm. It also has a very low standard deviation, which may be understood as a good robustness property, i.e. the accuracy of the method does not depend too much on which patterns are selected for the learning set.

It is worth noting that the methods with the highest learning accuracy are not necessarily the ones that perform best on the validation set. This is due to the *overlearning* effect, i.e. the fact that the ellipsoid tries to fit the learning data too closely (including some associated noise), which affects its generalization capabilities. Indeed, the method with the best generalization is the only one that was not able to separate perfectly the 20 % learning set.

Wisconsin Breast Cancer

This data set comes from a very useful application, and the procedure which provided its data is actually used for machine-aided medical diagnosis.

	20 % learning set				50 % learning set			
	Learning		Testing		Learning		Testing	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Max. sep. ratio	**0.00** %	0.0 %	5.60 %	1.3 %	**1.54** %	3.1 %	6.37 %	2.6 %
Min. volume	1.90 %	1.2 %	**5.05** %	1.1 %	2.56 %	0.9 %	4.47 %	0.7 %
Max. sum	2.20 %	2.0 %	5.58 %	2.0 %	2.71 %	0.8 %	**4.17** %	0.6 %
Min. squared sum	0.15 %	0.5 %	5.42 %	1.6 %	1.90 %	0.5 %	5.05 %	0.7 %

As usual, the *maximum separation ratio* method has the lowest learning error rates, but is no more able to separate perfectly the 50 % learning set (1.54 % error rate). The testing error rates are fairly good, ranging from 4.17 % to 6.37 %. The best methods are here the *minimum volume* and the *maximum sum* algorithms, which are very close to each other. This data set may also be considered as rather easy to separate.

We may also note another consequence of overlearning : the *maximum separation ratio* method testing results are better with a 20 % learning set than with a 50 % learning set, which means that the additional patterns have decreased its generalization capacity.

Boston Housing

This data set is much more difficult to separate. We didn't find in the literature any reported result with an error rate less than 16 %.

	20 % learning set				50 % learning set			
	Learning		Testing		Learning		Testing	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Max. sep. ratio	**0.0** %	0.0 %	16.5 %	1.3 %	23.7 %	27.4 %	31.7 %	19.4 %
Min. volume	18.1 %	4.2 %	21.9 %	3.2 %	37.1 %	15.1 %	37.2 %	15.1 %
Max. sum	37.4 %	16.4 %	38.9 %	14.4 %	52.1 %	3.3 %	51.4 %	1.8 %
Min. squared sum	1.2 %	1.4 %	**15.8** %	1.3 %	**7.5** %	3.4 %	**12.4** %	0.9 %

The *maximum separation ratio* method only separated the 20 % learning set correctly. On the other learning set, it performed quite poorly with a large 23.7 % error rate. This is due to the fact that the 50 % learning sets are not always separable. This means that the method will get a kind of *all-or-nothing* error rate, i.e. 0 % or 50 % depending on the actual separability of the learning set. Apparently, separation was possible approximately on half of the learning sets. This also explains the large standard deviation observed.

The *minimum volume* and *maximum sum* algorithms do not seem to be well-suited to this classification task, with rather catastrophic learning and testing rates. The best method is without discussion the *minimum squared sum* algorithm, with both very low learning and testing errors. It is again interesting to note that going from a 20 % to 50 % learning set increases the learning error by 6.3 % but decreases the testing error by 3.4 %.

Pima Indians Diabetes

This data set is even more difficult to separate than the previous one. We didn't find in the literature any reported result with an error rate less than 24 %.

	20 % learning set				50 % learning set			
	Learning		Testing		Learning		Testing	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Max. sep. ratio	50.0 %	15.9 %	50.0 %	15.9 %	42.5 %	15.1 %	42.5 %	15.1 %
Min. volume	29.5 %	8.5 %	34.0 %	6.5 %	32.6 %	3.6 %	33.5 %	3.6 %
Max. sum	22.9 %	4.2 %	29.7 %	3.5 %	43.0 %	14.0 %	44.1 %	14.2 %
Min. squared sum	**21.3** %	4.7 %	**28.5** %	2.5 %	**26.1** %	6.9 %	**28.9** %	6.7 %

The results resemble closely those of the Boston Housing data set. The *maximum separation ratio* performs now poorly on both sizes of the learning set, which means that both of them are not separable from time to time. The *maximum sum* performance is decent on the 20 % learning set, but collapses on the 50 % set, another clear sign of overlearning. Finally, the best method is again the *minimum squared sum* algorithm, with practically the same accuracy on the 20% and the 50% test sets. The 20 % testing errors are in fact better because of their very low standard deviation.

Overall comments

Our understanding of these results is that one has to make a distinction between small (or easy) problems (Iris, Breast Cancer) and the more difficult ones.

- For small and/or easy problems, the *maximum separation ratio* is unbeatable on the learning errors. Again, we stress that this is a normal consequence of the fact that this methods strives to separate the classes completely. However, its generalization capacity is quite poor and we do not recommend it. Instead, we think that the *minimum volume* algorithm is often a good choice. This may seem rather surprising since this method does not try to exclude the b_j's out of the ellipsoid, but it looks like this isn't a problem when the classes are clearly (nearly) separated. It might also be worthy to give the *maximum sum* algorithm a try on these problems.

- For larger and/or more difficult problems, the *minimum squared sum* method is a clear winner. In these kind of classification tasks, the absence of a (nearly) separating ellipsoid seems to be a great handicap for the other methods (especially for the *maximum separation ratio* algorithm), while it doesn't bother the *minimum squared sum* too much. We are quite satisfied with this fact, since one of the design goals of this method was to improve the results of the other methods on inseparable sets. Recall we used an objective function that takes into account the smaller values of the separation ratio (as opposed to the *maximum sum* algorithm which is greatly influenced by the large ratios) while being not as stringent as the *maximum separation ratio* algorithm to be able to handle the inseparable sets.

3.2.3 Comparison with other procedures

Our task has been made much easier thanks to the authors of *Logical Analysis of Data (LAD)* classification procedure. In [EB$^+$96], they extensively search the machine learning literature and give the performance results of the best method they found, in order to compare it with their own results. Their method, which is also based on mathematical programming, often beats all the existing procedures.

In the following table, we report the testing errors obtained by the best of our four methods, both on the 20 % and the 50% learning sets, along with the 50 % results for the *LAD* method. The last three columns report the best result other than *LAD* as reported in [EB+96], along with the associated size of the training set and a reference.

	Best ellipsoidal		LAD	Best other method		
Training set size	20 %	50 %	50 %	Variable	% train.	Ref.
Wisconsin Breast Cancer	5.05 %	4.17 %	**3.1** %	3.8 %	80 %	[SKMS94]
Boston Housing	15.77 %	**12.38** %	16.0 %	16.8 %	80 %	[SKMS94]
Pima Indians Diabetes	28.50 %	28.91 %	28.1 %	**24.1** %	75 %	[Smi88]

On the relatively easy *Wisconsin Breast Cancer* data set, the *LAD* method is the best with approximately 3.1 % error (96.9 % accuracy), followed by another method (see [SKMS94]) with 3.8 % error (96.2 % accuracy, using a 80 % training set). Our *minimum volume method* still provides a rather low error rate at about 4.2 % (93.8 % accuracy).

The *minimum squared sum* algorithm is a clear winner on the *Boston Housing* data set, with 12.4 % error (87.6 % accuracy). It is even more surprising to observe that the separation with a 20 % learning set is already better than the other procedures (*LAD* and [SKMS94]) with a 50 % and even a 80 % learning set.

Finally, the *Adap learning* algorithm using a neural network (see [Smi88]) is the best on the *Pima Indians Diabetes* with a 24.1 % error rate (75.9 % accuracy using a 75 % training set). Our *minimum squared sum* algorithm and the *LAD* method follow with approximately 28 % misclassified patterns.

Most of the reported results we found in the literature use cross-validation with 80 %, 90 % or even *all-but-one* pattern learning sets[1]. It is therefore not completely fair to compare these results with ours, which only use 50 % or 20 % for the learning set.

In fact, our methods already produce acceptable results with a very small learning set, which can definitely be viewed as an advantage over other methods (this could be called *quick* generalization). On the other hand, the limited number of computational experiments we conducted with a 80 % learning set did not seem to increase the testing accuracy in a significant manner.

To summarize, we would say our method provides a viable way of performing classification, giving occasionally excellent results (significantly better than any other existing procedure) and competitive error rates on many data sets.

[1]This is called the *leave-one-out* cross-validation technique.

Chapter 4

Conclusions

4.1 Summary

4.1.1 Conic programming

In this chapter, after some general information about convex and conic programming, we explained why *SQL conic programming* is so appealing to practitioners and how it is possible to use it to model a great variety of constraints. This rather new field is now becoming mature, and its very wide applicability shows a great potential for future research.

4.1.2 Pattern separation

We introduced in this chapter four pattern separation methods using ellipsoids. We believe that three of them, those which involve separation ratios, are completely new. The remaining method, namely the *minimum volume* method, had already been proposed but had never been seriously tested. For each methods some properties were investigated (independence from the coordinate system, behaviour when there exists no separating ellipsoid, etc.).

Another interesting contribution is the use of the *homogeneous ellipsoid* representation and the concept of *canonical* homogeneous ellipsoid. This allowed us to optimize both the ellipsoid's shape and its center as well as to prove that our methods are exact, i.e. really optimize the quantity we want to minimize or maximize.

4.1.3 Computational experiments

The main result from this chapter is that our methods provide a viable way to classify patterns. Numerical tests highlighted the fact that the *minimum volume* method is well-suited for small and/or easy pattern separation problems, while the *minimum squared sum* algorithm is the best for more difficult classification tasks. The presence of an *overlearning* effect was demonstrated.

As far as comparison with other classification procedures is concerned, we repeat the conclusion of this chapter, viz. that separating patterns using ellipsoids occasionally delivers excellent results (significantly better than any other existing procedure) and gives competitive error rates on many data sets.

4.2 Perspectives for future research

Among the possible directions for further research, we have the following

- *Successive separations.* The idea behind this expression may be described as follows. We first use of our methods to classify the patterns with a certain degree of certainty (cf. Subsection 2.5.1). The patterns for which the method hesitates are isolated to form a second separation problem. The patterns for which the second separation process hesitates will form a third separation problem, and so on until all patterns are classified with certitude. At each step, the uncertain region would be in our case comprised between a pair of homothetic ellipsoids.

- *Combining ellipsoids.* This idea is twofold

  1. We can try to combine the ellipsoids provided by different methods, i.e. create some sort of *averaged* ellipsoid. This would possibly lead to a more robust separation process.
  2. Another completely different way of proceeding would be to use only a subset of the learning set to compute the ellipsoids. This would be repeated a certain number a times, using different subsets, and the resulting ellipsoids would be *averaged.* This is related to the *jackknife* methods used in statistics. The possible advantage of this procedure would be a lower sensibility to outliers and wrongly classified learning patterns, reducing thus the overlearning effect.

- *Bi-ellipsoid optimization.* As we have seen in our computational experiments, the mixed exploitation strategy has the best generalization capabilities. This means that we have to use two ellipsoids to classify a pattern, but that each of these ellipsoids is optimized completely separately. Moreover, this is done according to an objective function that doesn't take into account the further use of this optimized ellipsoid with another one. An interesting idea would be to combine the determination of these two ellipsoids in a single SQL conic program, where the objective would take into account the mixed exploitation strategy that is going to be used.

We believe that the implementation of one or several of these suggestions would lead to a further improvement of the accuracy and the robustness of our methods.

Bibliography

[Bar82] E. R. Barnes, *An algorithm for separating patterns by ellipsoids*, IBM Journal of Research and Development **26** (1982), no. 6, 759–764.

[BTN98] A. Ben-Tal and A. Nemirovsky, *Convex optimization in engineering: Modelling, analysis, algorithms*, Lecture notes available at `http://ssor.twi.tudelft.nl/~prodanov/frame3.htm`, 1998.

[Dan63] G. B. Dantzig, *Linear programming and extensions*, Princeton University Press, Princeton, N.J., 1963.

[EB+96] T. Ibaraki E. Boros, P. L. Hammer et al., *An implementation of logical analysis of data*, Tech. report, IDIAP, Martigny, Valais, Switzerland, July 1996.

[FA+97] M. V. Nayakkankuppann F. Alizadeh, J. A. Haeberly et al., *SDPPack user's guide*, Tech. report, New York University, New York, USA, 1997.

[Kar84] N. K. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica **4** (1984), 373–395.

[MM98] C.J. Merz and P.M. Murphy, *UCI repository of machine learning databases*, University of California, Irvine, Dept. of Information and Computer Sciences, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1998.

[Nes96] Y. Nesterov, *Nonlinear optimization*, Notes from a lecture given at CORE, UCL, Belgium, 1996.

[NN94] Y. E. Nesterov and A. S. Nemirovsky, *Interior-point polynomial methods in convex programming*, SIAM Studies in Applied Mathematics, SIAM Publications, Philadelphia, 1994.

[NT97] Y. Nesterov and M. J. Todd, *Self-scaled barriers and interior-point methods for convex programming*, Mathematics of Operations Research **22** (1997), no. 1, 1–42.

[PJW34] J. Von Neumann P. Jordan and E. Wigner, *On algebraic generalization of the quantum mechanical formalism*, Annals of Mathematics **36** (1934), 29–64.

[Roc70] R. T. Rockafellar, *Convex analysis*, Princeton University Press, Princeton, N. J., 1970.

[Ros65] J. B. Rosen, *Pattern separation by convex programming*, Journal of Mathematical Analysis and Applications **10** (1965), 123–134.

[Sch86] A. Schrijver, *Theory of linear and integer programming*, Wiley–Interscience series in discrete mathematics, John Wiley & sons, 1986.

[SKMS94] S. Kasif S. K. Murthy and Steven Salzberg, *A system for induction of oblique decision trees*, Journal of Artificial Intelligence Research **2** (1994), 1–32.

[Smi88] J. W. Smith, *Using the Adap learning algorithm to forecast the onset of diabetes mellitus*, Proceedings of the Symposium on Computer Applications and Medical Care, IEEE Computer Society Press, 1988, pp. 261–265.

[Stu97] J. F. Sturm, *Primal-dual interior-point approach to semidefinite programming*, Ph.D. thesis, Erasmus Universiteit Rotterdam, The Netherlands, 1997.

[Tun96] L. Tunçel, *Primal-dual symmetry and scale invariance of interior-point algorithms for convex optimization*, Tech. report, Department of Combinatorial Optimization, University of Waterloo, 1996.

[Wri97] S. J. Wright, *Primal-dual interior-point methods*, SIAM, Society for Industrial and Applied Mathematics, Philadelphia, 1997.