UNIVERSITY OF SOUTHAMPTON

Linear and Ellipsoidal Pattern Separation: Theoretical Aspects and Experimental Analysis

by

Andriy Kharechko

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

July 2009

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Andriy Kharechko

This thesis deals with a pattern classification problem, which geometrically implies data separation in some Euclidean feature space. The task is to infer a classifier (a separating surface) from a set or sequence of observations. This classifier would later be used to discern observations of different types. In this work, the classification problem is viewed from the perspective of the optimization theory: we suggest an optimization problem for the learning model and adapt optimization algorithms for this problem to solve the learning problem. The aim of this research is twofold, so this thesis can be split into two self-contained parts because it deals with two different type of classifiers each in a different learning setting.

The first part deals with linear classification in the online learning setting and includes analysis of existing polynomial-time algorithms: the ellipsoid algorithm and the perceptron rescaling algorithm. We establish that they are based on different types of the same space dilation technique, and derive the parametric version of the latter algorithm, which allows to improve its complexity bound and exploit some extra information about the problem. We also interpret some results from the information-based complexity theory to the optimization model to suggest tight lower bounds on the learning complexity of this family of problems. To conclude this study, we experimentally test both algorithms on the positive semidefinite constraint satisfaction problem. Numerical results confirm our conjectures on the behaviour of the algorithms when the dimension of the problem grows.

In the second part, we shift our focus from linear to ellipsoidal classifiers, which form a subset of second-order decision surfaces, and tackle a pattern separation problem with two concentric ellipsoids where the inner encloses one class (which is normally our class of interest, if we have one) and the outer excludes inputs of the other class(es). The classification problem leads to semidefinite program, which allows us to harness the efficient interior-point algorithms for solving it. This part includes analysis of the maximal separation ratio algorithm in the machine learning context and its application to a text categorization problem.

Acknowledgements

First of all, I would like to express my gratitude to my former supervisors Prof. John Shawe-Taylor and Dr Ralf Herbrich for their crucial support of my application for this postgraduate project, their advice and help during the course and their comments on the first draft of this thesis. I also would like to thank to Microsoft Research in Cambridge, Universities UK (Overseas Research Student Award Scheme), Royal Holloway College of the University of London and the School of Electronics and Computer Science of the University of Southampton for giving me this opportunity to pursue my Ph.D. in the United Kingdom by providing scholarships and grants for tuition fees and maintenance, as well as to PASCAL Network of Excellence for funding most of my research visits, Mieres City Council, Spain, for accommodation grant to attend I Summer Course Future Directions in Soft Computing and Max-Planck-Institut für Informatik in Saarbrücken, Germany, for travel grants to attend Advanced Course on the Foundations of Computer Science (ADFOCS) in 2007 and 2008.

Then I would like to acknowledge my lecturers at the Ivan Franko National University of L'viv not only for doing their best in providing high-quality education, but also for their help and advice beyond their duty. Especially I need to mention Dr Oksana Kostiv, Dr Anatoliy Muzychuk and Dr Tetyana Yakhontova for our inspiring and motivating conversations on academic and related matters that broadened my personal outlook, my tutor Dr Serhiy Yaroshko for his guidance during my course of study, and the supervisor of my undergraduate thesis Prof. Nikolai Voitovich for introducing me into the world of scientific research. Without contribution of these people, my Ph.D. project would not have been conceived.

Furthermore, I am grateful to Prof. Steve Schneider for his advice and guidance when I was applying for this postgraduate programme and during my year at Royal Holloway College, to Dr Myroslava Dzikovska and Dr Serhii Zhak for their help and advice during the application process and afterwards, to Dr Amiran Ambroladze, Dr Thore Graepel, Dr Jaz Kandola for their both research advice and informal guidance during my study. I also feel indebted to Dr Sándor Szedmák for our numerous discussions, which heavily influenced my current understanding of many research topics of this thesis and beyond, his instant feedback on my work and kind advice on my future.

During the most down times of my writing-up period, I stayed on track, gradually recovered from my mental health problems, and succeeded with my thesis mainly because of the medical advice and care of Dr Sarah Armstrong and Dr Theresa Creagh; encouragement, support and understanding from my supervisor Dr Craig Saunders and my tutor Eric Cooke; and advice and help from Dr Dmytro Kundys.

Finally, I would like to say that I am always grateful to my family for everything they have been doing for me and to my closest friends in Ukraine, Nazar Pyrch and Andriy Vynnyk, for just always being there.

To my parents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Machine learning is a flourishing branch of artificial intelligence which tackles many-real world problems where traditional (non-learning) algorithmics fails. Among them there are image and speech recognition, data mining, gene extraction, etc. The aim of machine learning is to design computer systems which can learn from experience or, in other words, enhance their performance as more information (data) related to the problem to solve is available.

A good introduction to machine learning with an overview of main learning techniques is presented in Mitchell (1997), and more advanced concept-based treatment of the subject is given by Cherkassky and Mulier (1998). In this work we focus on *pattern classification* in the *online* and batch settings. This type of a learning problem subsumes handwritten character recognition, document categorization, face recognition, medical diagnosis and some other problems. The task is to discriminate *instances* (vector representations of the inputs) that belong to different classes, thus to learn a *classification pattern* (a *classifier*) from the data. This thesis is mainly focused on the cases of one and two classes only (*one-class* and *binary classification*), since *multi-class classification* can always be reduced to the latter problem by virtue of discriminating between either each class and the rest (*one-versus-all* strategy) or any two classes (*pairwise classification*). Different statistical, neural and structural approaches to pattern classification are discussed in Fukunaga (1990); Schalkoff (1992); Duda et al. (2001) and other sources.

The aim of this research is twofold, so this thesis can be split into two self-contained parts because it deals with two different type of classifiers each in a different learning setting. The first part deals with the model of online learning linear classification patterns (Littlestone, 1988; Angluin, 1988) and includes analysis of existing polynomial-time algorithms: the ellipsoid algorithm (Yudin and Nemirovski, 1976b; Khachiyan, 1979; Shor

and Gershovich, 1982) in Chapter 2 and the perceptron rescaling algorithm (Dunagan and Vempala, 2008) in Chapter 3, and derivation of a generalized version of the former algorithm (Chapter 4) for this model, comparison of their computational and learning complexities (Chapter 5) and demonstration of their performance in practice for *semidefinite feasibility programs* which is a real-world application of this model (Chapter 6).

In the second part of the thesis, we shift our focus from linear to *ellipsoidal classifiers*, which form a subset of second-order decision surfaces formed as surfaces of uniformly convex bodies, and tackle a pattern separation problem with two concentric ellipsoids where the inner encloses one class (which is normally our class of interest, if we have one) and the outer excludes inputs of the other class(es). The classification problem leads to a *semidefinite program*, which allows us to harness the efficient *interior-point algorithms* (Nesterov and Nemirovskii, 1994) for solving it. This part includes analysis of the *maximal separation ratio algorithm* (Chapter 7) in the machine learning context and its application to a *text categorization problem* (Chapter 8).

1.2 Historical Background of Machine Learning

The first machine learning techniques were developed by analogy with human learning in general and the way the human brain learns in particular (limited by contemporary scientific understanding of the latter, however). One of the first learning algorithms was the *perceptron algorithm* of Rosenblatt (1958) that aims to learn a *linear classification pattern* (a hyperplane). The idea was inspired by the recent developments of that time in neuroscience, particularly by the computational model of a neuron by McCulloch and Pitts (1943), and evoked in the artificial intelligence community much enthusiasm for further research. However, a profound analysis of the limitations of linear classifiers by Minsky and Papert (1969) dampened it until the middle of the eighties when the backpropagation and other algorithms for neural networks were suggested (Rumelhart and McClelland, 1986). A detailed overview and discussion of different applications of neural networks to pattern recognition problems are presented by Bishop (1995).

The *support vector machine (SVM)* algorithm (Cristianini and Shawe-Taylor, 2000) suggested in Boser et al. (1992) has become a real state-of-the-art tool in pattern analysis. This algorithm searches for a separating hyperplane with the maximal *separation margin* in some high-dimensional *feature space*. The *geometrical margin* is the minimal distance between the separating hyperplane and the closest training example. The feature space is implicitly defined by some *kernel function* (generalized inner product), and the choice of kernels is usually determined by the type of problem being solved (Shawe-Taylor and Cristianini, 2004). The *generalization theory* and other aspects of the SVM are discussed in much detail by Vapnik (1998). A profound treatment of both theory and algorithms of kernel classifiers is presented in Herbrich (2002).

1.3 Development of Convex Optimization

Semidefinite programming optimizes a linear function over a convex set defined by positive semidefinite constraints (linear matrix inequalities), therefore it generalizes some other classes of convex programs such as linear and second-order cone programs, while, at the same time, semidefinite programs (of a reasonable size, however) are also efficiently (in polynomial time) solvable by the state-of-the-art interior-point methods.

Overviews of semidefinite programming with applications to combinatorial optimization, system and control, etc. and descriptions of the interior-point methods are given in the seminal articles of Alizadeh (1995); Vandenberghe and Boyd (1996); Todd (2001). The geometry of the cone of positive semidefinite matrices and generalization of the simplex-type algorithms to semidefinite programming are discussed by Pataki (1995); Lasserre (1995) with references to convex analysis in Rockafellar (1970) and analysis of convex polytopes in Brøndsted (1983). Horn and Johnson (1991) provides an excellent survey of matrix analysis, which includes a chapter on positive semidefinite matrices. The tools of convex programming and modelling together with different applications in engineering and computer science are discussed by Boyd and Vandenberghe (2004). Ben-Tal and Nemirovski (2001) gives a detailed overview of linear, second-order cone and semidefinite programming, indicates their modern applications in engineering and combinatorial optimization, and includes a detailed discussion of complexity of the interior-point methods.

Many methods for solving an SDP were developed by analogy with linear programming. Khachiyan (1979) showed, for the first time, that the *ellipsoid algorithm* solves a linear program in rational arithmetics in polynomial time, and in Grötschel et al. (1981) the algorithm was generalized to convex programming. The latter result means that we can solve a problem of optimizing a linear function over a convex set in polynomial time if we are provided with a *separation oracle* for that set, i.e. a procedure that either confirms if a given point belongs to the set or returns a separating hyperplane otherwise. Polynomial time solvability of semidefinite programming follows from this result (subject to some mild limitations, e.g. see Ramana and Goldman (1995)), since we can use either Cholesky decomposition, or eigendecomposition as a separation oracle for a set defined by positive semidefinite constraints. Indeed, both these methods can check feasibility of a given solution $\mathbf{x} \in \mathbb{R}^m$ (in other words, determine if the matrix $\mathbf{F}(\mathbf{x})$ is positive semidefinite for a given $\mathbf{x}$), and return a *counter-example* (a vector $\mathbf{z} \in \mathbb{R}^n$, such that $\mathbf{z}^\top \mathbf{F}(\mathbf{x})\mathbf{z} < 0$) to construct a separating hyperplane in the case that is not feasible. An *a priori* bound on the optimal value of the problem, which allows us to re-state optimization of the objective function as a sequence of feasibility programs, can be found using duality theory for an SDP. Moreover, if we solve a primal-dual problem, then we optimize the duality gap of the problem which is bounded by 0.

A completely different approach to solving linear programs was suggested by Karmarkar (1984) which boosted the field of the *interior point methods*. This algorithm has better worst-case complexity than the ellipsoid algorithm, moreover in practice it is more efficient than the simplex method. It requires $\mathcal{O}(n \log \frac{1}{\epsilon})$ iterations to find a primal feasible point in $\mathbb{R}^n$ such that the value of the objective is within ϵ of its optimal value. The algorithm constructs an *optimizing sequence*, a sequence $\{\mathbf{x}_k\}$ such that

$$f(\mathbf{x}) \leq \ldots \leq f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) \leq \ldots \leq f(\mathbf{x}_2) \leq f(\mathbf{x}_1),$$

in a way that it uses a projective transformation of the primal feasible set that maps the current approximation $\mathbf{x}_k$ to the centre of the set, makes a step in the feasible steepest descent direction in the mapped space. The progress towards the optimum is measured by a logarithmic potential function (Fletcher, 1987; Wright, 1997). Later Karmarkar's algorithm was modified in a way that a different logarithmic barrier function was suggested and Newton's method was used to choose a search direction (Renegar, 1988). These modifications improved the bound on the number of iterations to $\mathcal{O}(\sqrt{n} \log \frac{1}{\epsilon})$.

Nesterov and Nemirovskii extended the interior-point methods to the case of a generic convex program using the notion of a ν-self-concordant barrier function (Nesterov and Nemirovskii, 1990b), and showed that these methods minimize a linear function on a generic convex set provided the set is endowed with a self-concordant barrier function (Nesterov and Nemirovskii, 1994). Since such a barrier function is known for SDPs (Nesterov and Nemirovskii, 1990a), the interior point method is currently one of the most efficient ways of solving SDPs in practice.

Since efficient polynomial time interior-point methods were generalized to solve semidefinite programs (Nesterov and Nemirovskii, 1990a), the number of problems from different areas that can be posed as semidefinite programs has been rapidly increasing. Nowadays semidefinite programming is applied to system and control theory (Boyd et al., 1994), structural optimization (Ben-Tal and Nemirovski, 2001), machine learning (Graepel, 2002; Kandola et al., 2003; Lanckriet et al., 2004; De Bie and Cristianini, 2004; Graepel and Herbrich, 2004) and other fields. At the same time, some machine learning tools were adapted to solve semidefinite programs in real time (Jiang and Wang, 1999).

1.4 Overview

In Chapter 2 we formally state the main notions of machine learning, define online learning setting, suggest a convex optimization model for this setting and consider the *perceptron* learning algorithms (which originate from the computational model of a spiking neuron cell (Rosenblatt, 1958) and are based on the subgradient descent minimization algorithm) which share the same update rule to amend the current hypothesis but differ in learning parameters and pre-processing of the data prior to making an update after

a mistake occurs. The online learning setting implies that there are no learning data available beforehand, but classification and learning are merged and proceed as new examples arrive one by one every time after the learner has made a hypothesis about the classifier. First and foremost we will focus on those algorithms whose learning complexities grow polynomially with the dimension of the input space. This is guaranteed for the *ellipsoid algorithm* (Khachiyan, 1979) and the randomized *perceptron rescaling algorithm* (Dunagan and Vempala, 2008) or the *probabilistic perceptron*. The polynomiality of the former is achieved by applying space dilation (Shor, 1970b) in the direction of the misclassified example after every update. This approach iteratively increases the *margin* of the dataset, which improves its separability.

The perceptron rescaling algorithm, which is analyzed in Chapter 3, harnesses the plain (non-polynomial in the general case) perceptron algorithm for a fixed number of updates if the margin is large enough and applies the *rescaling procedure* to the dataset otherwise. This transformation follows a probabilistic argument to choose a proper direction of rescaling and under its optimal choice may increase the margin arbitrarily close to 1 if the rescaling factor is raised to infinity. However, we show that the latter observation does not hold for the general choice of the rescaling direction employed by the algorithm. We also show that this procedure is a partial case of space dilation under a fixed choice of the dilation factor. In this thesis we will refer to the dilation technique as the *rescaling procedure* for the reasons given in Section 2.6. Finally, we analyze the positive rescaling factor of the perceptron rescaling algorithm, and show that its value is close to the optimum.

Since the probabilistic perceptron has higher computational complexity than the ellipsoid algorithm, we compare the latter with the plain perceptron (first phase of the probabilistic perceptron) with respect to their computational complexities and show that for the datasets with a fixed margin the plain perceptron outperforms the ellipsoid algorithm on problems of sufficiently high dimensions.

A parametric version of the perceptron rescaling algorithm is derived of Chapter 4. It is demonstrated that some fixed parameters of the latter algorithm (the initial estimate of the margin and the relaxation parameter) may be modified and adapted for particular problems. The generalized theoretical framework allows to determine convergence of the algorithm with any chosen set of values of these parameters, and suggests a potential way of decreasing the complexity of the algorithm which remains the subject of current research. We derive conditions sufficient for convergence of the parametric algorithm in polynomial time with high probability. As a part of this study, we also derive a parametric version of the modified perceptron algorithm for learning noisy threshold functions Blum et al. (1998), which is the perceptron improvement phase of the perceptron rescaling algorithm, and prove its convergence.

To conclude our theoretical study of the model of learning a linear classifier in the online framework, we exploit some results from the *information-based theory of convex programming* (Nemirovsky and Yudin, 1983) in Chapter 5 to sketch a derivation of tight lower bounds on the *learning complexity* of this family of online learning problems. These bounds generalize our comparison of computational complexities of the perceptron algorithms and show that for the datasets with a fixed sufficiently large margin the plain perceptron algorithm has learning complexity of the same order of magnitude as the *optimal algorithm* for this family of problems when the problem dimension grows sufficiently large. These results extend the bounds on the learning complexity of hyperplane learning in Maass and Turán (1994) from the case when the *input space* is a subset of $\mathbb{Z}^n$ to $\mathbb{R}^n$, hence the *input sequence* may be infinite.

For experimental analysis of the perceptron learning in Chapter 6, we choose a semidefinite constraint satisfaction problem. This real-world problem, which has many application, is posed a a linear program with infinitely many constraints.We suggest an efficient separation oracle for this problem, based on the incomplete Cholesky decomposition of symmetric matrices. We also amend the modified perceptron algorithm to handle positive semidefinite constraints. We use a semidefinite relaxation of the MAX-CUT problem (Goemans and Williamson, 1995; Helmberg, 2000) provided by SDPLIB collection (Borchers, 1999) as a benchmark problem for the ellipsoid and the parametric perceptron algorithms. Obtained numerical results show that the latter outperforms the former in practice, while its learning complexity (the number of mistakes) gradually approaches that of the ellipsoid algorithm as the dimension of the problem increases (which confirms our theoretical conjectures in the previous chapter).

In Chapter 7 we give some preliminary information on ellipsoids and mention extremal ellipsoid problems, i.e. the problems to approximate different convex sets of some complex structure with ellipsoids. A good overview of extremal ellipsoids can be found in Ben-Tal and Nemirovski (2001) and Boyd and Vandenberghe (2004). An alternative approach which uses combined interior-point and active-set method to solve the resulting optimization problem is presented in Sun and Freund (2004). Afterwards we consider the maximal separation ratio algorithm from Glineur (1998), follow its derivation and discuss its properties. To conclude this chapter, we derive its dual problem and discuss its interpretation in the learning context, which may be interesting for future research.

We demonstrate an application of the maximal separation ratio algorithm from Glineur (1998) to a text categorization problem in Chapter 8 using an approximation of the latent semantic indexing for feature extraction from Cristianini et al. (2002), present the numerical results on the Reuters-21578 document collection, which show that the algorithm's prediction sometimes reaches the level of the state-of-the-art SVM algorithm, and indicate directions of its improvement for future research.

For the rest of the thesis we denote matrices and vectors by bold face upper and lower case letters, e.g. $\mathbf{A}$ and $\mathbf{x}$. We shall use $\overline{\mathbf{x}} := \mathbf{x}/\left\|\mathbf{x}\right\|$ to denote the unit length vector in the direction of $\mathbf{x}$ provided $\mathbf{x} \neq \mathbf{0}$.

1.5 Published Contributions

The work presented in this thesis has contributed in part or full to the following publications:

- T. Graepel, R. Herbrich, A. Kharechko and J. Shawe-Taylor (2004). Semidefinite Programming by Perceptron Learning. In S. Thrun, L. Saul and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Chapter 8, pp. 457-465. MIT Press, 2004.

- A. Kharechko, J. Shawe-Taylor, R. Herbrich and T. Graepel. Text Categorization via Ellipsoid Separation. PASCAL Workshop on Learning Methods for Text Understanding and Mining. Grenoble, France, January, 2004.

- A. Kharechko, J. Shawe-Taylor, R. Herbrich, and T. Graepel. Text Categorization via Ellipsoid Separation. In *Proceedings of Postgraduate Research Conference in Electronics, Photonics, Communications & Networks, and Computing Science (PREP2004)*, pp. 19-20, University of Hertfordshire, Hatfield, UK, 2004.

- A. Kharechko. Parametric Polynomial Time Perceptron Rescaling Algorithm. In H. Broersma, S. Dantchev, M. Johnson and S. Szeider, editors, *Algorithms and Complexity in Durham 2006 - Proceedings of the Second ACiD Workshop, 18-20 September 2006, Durham, UK*, p. 157. Texts in Algorithmics 7, College Publications, King's College, London, 2006.

Chapter 2

Perceptron Learning Algorithm

2.1 Online Learning Model

Consider a (possibly infinite) sequence of unit vectors $\mathbf{a}_1, \mathbf{a}_2, \ldots \in \mathbb{R}^n$. It is assumed that all the vectors belong to an a priori unknown and possibly infinite set $A \subset \mathbb{R}^n$ which is also known to be *linearly separable* from the origin: there exists a (generally non-unique) hyperplane through the origin defined by its unit normal vector $\mathbf{x} \in \mathbb{R}^n$ that keeps A on its positive side (a *separating hyperplane*), i.e. $\mathbf{a}^\top \mathbf{x} \geq 0$ holds for all $\mathbf{a} \in A$. The task is to find some separating hyperplane using a finite (and preferably as small as possible) number of observed vectors from the sequence.

Assuming that every vector in the sequence represents an input observation (an instance) that was mapped to the Euclidean vector space (an *instance* or *feature space*) over $\mathbb{R}^n$, our task is to determine some linear relation between the data (a *pattern*) in order to be able to decide for any vector in $\mathbb{R}^n$ whether it originates from the same source as all our previous observations. In other words, we want to *learn* a *classification pattern* of the input data defined by some unknown hyperplane through the origin (a *decision surface*). Since all our observations are assumed to be from the same source, our problem is a *one-class pattern classification problem*. The linear decision surface defines a linear *classifier* (a function that determines whether an input belongs to a certain class), so this task can be also viewed as learning an unknown linear threshold function.

Since we can access the dataset A only through some sequence of its members, our learning problem is posed in the *online* setting (the learning and classification phases are merged). We consider the following learning model: our algorithm (a *learner*) suggests a hypothesis of the classifier from the hypothesis space $\mathcal{H} \subseteq \mathbb{R}^n$ and after that receives an observation, if the latter contradicts the current hypothesis (is a *counterexample*) then the hypothesis is updated, and this process continues until no counterexamples occur in the input sequence (in other words, the current hypothesis is *consistent* with

A). This learning model was first suggested by Littlestone (1988) (almost at the same time an equivalent query model was described by Angluin (1988)) for concept learning (learning Boolean functions) that is learning a separating hyperplane of a finite subset of $\mathbb{Z}^n$. Classification of a newly observed example is also called a *trial*. In this setting our problem is uniquely defined by a set of vector observations $A \in \mathbb{R}^n$ and an index set $I \subseteq \mathbb{N}$ which defines the sequence of delivering examples from A.

For the purposes of this work we consider a simpler version of this model relaxed in a way that we ignore a trial when the learner receives an example which supports its current hypothesis.[1] Therefore, we do not need to fix the sequence of data vectors (i.e. the index set I) a priori, but assume that every new example is chosen from a subset of misclassified data in A by a *separation oracle* (a *teacher*) defined in the following way.

Definition 2.1 (Separation Oracle)**.** Given a (possibly infinite) set $A \in \mathbb{R}^n$ a *separation oracle* is a mapping $\Omega_A : \mathbb{R}^n \to \mathbb{R}^n \cup \{*\}$ such that

$$\forall \mathbf{x} \in \mathbb{R}^n \quad \Omega_A(\mathbf{x}) \quad = \quad \begin{cases} \mathbf{a} & \text{if} \quad \exists \mathbf{a} \in A : \mathbf{a}^\top \mathbf{x} \leq 0, \\ * & \text{if} \quad \forall \mathbf{a} \in A : \mathbf{a}^\top \mathbf{x} > 0. \end{cases}$$

The name originates from the early approaches to solving convex feasibility programs when the solver needs either a certificate that the suggested point is feasible, or a hyperplane that separates it from the feasible set. The equivalence between a generic convex program and online learning a linear separation pattern over an infinite dataset will be suggested in Chapter 5.

This definition allows some non-determinism in the way the oracle chooses one of the misclassified examples which does not cause a problem for the exposition of our work. However, for the sake of clarity, throughout this monograph we assume that the oracle is *resisting* or *adversarial* (i.e. for a fixed learning algorithm it aims to provide that sequence of counterexamples which forces the learner to make more mistakes and arrive at a consistent hypothesis as late as possible) unless explicitly stated otherwise. Another relaxation of the classical online learning model which we allow is the ability of the oracle to report the successful termination of the learning procedure (returning '$*$') which may not be possible in the general case.

So far we have considered a family of online learning problems (described above), which we denote as $\mathfrak{P}$, whose input and hypothesis spaces coincide and constitute the same Euclidean space over $\mathbb{R}^n$ of a fixed dimension n. This family of online learning problems $\mathfrak{P}$ is parameterized by a linearly separable set $A \subset \mathbb{R}^n$ and a separation oracle $(\Omega_A \in \mathbb{R}^n \cup \{*\})^{\mathbb{R}^n}$, so its single member (an online learning problem) will be denoted by $\mathcal{P}(A, \Omega_A)$ for given A and Ω_A.

[1] Using the terminology of Littlestone (1988) and others, we consider *conservative*, or *failure-bounded*, or *mistake-driven algorithms* only.

2.2 Online Learning Algorithm and Complexity

Now we give a strict definition of an *online learning algorithm* (Maass and Turán, 1994, Section 2) and discuss a natural way of measuring the efficiency of different online algorithms with respect to the amount of information about the unknown set A which they need in order to deliver a consistent classifier.

Definition 2.2 (Online Learning Algorithm)**.** An online learning algorithm $\mathcal{A}$ for a problem $\mathcal{P}(A, \Omega_A)$ from a family of online learning problems $\mathfrak{P}$ is a set of recursive mappings $\mathsf{A}_i : \mathcal{H}^{i-1} \times \mathcal{X}^{i-1} \to \mathcal{H}$, $i \in \mathbb{N}$ that produce a hypothesis

$$h_t^{\mathcal{A}} \quad \leftarrow \quad \mathsf{A}_t(h_1^{\mathcal{A}}, \ldots, h_{t-1}^{\mathcal{A}}, a_1, \ldots, a_{t-1})$$

where $\mathcal{X}$ is the input space and $\mathcal{H}$ is the hypothesis space and which is based on its previous outputs $h_t^{\mathcal{A}}$ and received counterexamples to them $a_t = \Omega_A(h_{t-1}^{\mathcal{A}}), t \in \{1, \ldots, i-1\}$.

This notion is convenient because it encapsulates the aim of learning to improve performance with experience, and stresses the dependency on the units of input data which in the context of mistake-driven online learning are counterexamples to previously generated hypotheses.

As mentioned by Cesa-Bianchi et al. (2005) any learning algorithm for the *batch* setting (when a fixed finite dataset A, *a training set*, is available in advance) can be adapted to the online environment: at every trial a new example is added to the set of the previously observed examples and the algorithm is re-run. However, for the sake of computational efficiency and following the argument of Littlestone (1988), we prefer *incremental* algorithms, which can update the current hypothesis using only the newly observed counterexample, that is $\mathsf{A}_i : \mathcal{H} \times \mathcal{X} \to \mathcal{H}$, $i \in \mathbb{N}$ in Definition 2.2.

Apart from computational complexity, a natural criterion to compare different online learning algorithms is the minimal number of examples they need to learn a pattern over A. Since in Definition 2.2 we do not set any restrictions on the way the algorithm suggests a new hypothesis, this comparison makes sense only if we are able to estimate the bound on the minimal number of examples that the algorithm needs in order to solve *any* online problem from a fixed family because one can always easily suggest an optimal algorithm for any fixed learning problem which just outputs a consistent hypothesis in one step. In the online learning framework, the learner a priori does not know what specific instance of $\mathfrak{P}$ it needs to solve because it does not know the dataset A in advance, so it acquires information of $\mathcal{P}(A, \Omega_A)$ through the resisting oracle, and we are interested in the minimal amount of data it needs to identify a pattern for any $\mathcal{P}(A, \Omega_A) \in \mathfrak{P}$. This argument leads to the following measure of complexity of online learning algorithms.

Definition 2.3 (Learning Complexity of an Online Learning Algorithm)**.** The *learning complexity* of an online learning algorithm $\mathcal{A}$ on the family of online learning problems $\mathfrak{P}$ is defined as

$$LC(\mathcal{A}, \mathfrak{P}) \quad := \quad \max\{i \in \mathbb{N} \,|\, \exists \mathcal{P}(A, \Omega_A) \in \mathfrak{P} : \, \Omega_A(\mathbf{x}_{i-1}^{\mathcal{A}}) \neq *\}$$

where $\mathbf{x}_{i-1}^{\mathcal{A}} = \mathsf{A}_{i-1}(\mathbf{x}_1^{\mathcal{A}}, \ldots, \mathbf{x}_{i-2}^{\mathcal{A}}, \mathbf{a}_1, \ldots, \mathbf{a}_{i-2})$ and $\mathbf{x}_j^{\mathcal{A}}$ are hypotheses generated by $\mathcal{A}$ and $\mathbf{a}_j$ are the counterexamples provided by the oracle Ω_A, $j \in \{1, \ldots, i-2\}$.

This definition coincides with the notion of a *mistake bound* which is the maximal number of mistakes that an online learning algorithm makes in order to learn any linearly separable dataset of a certain class delivered in an arbitrary sequence of examples. The latter notion was first introduced by Littlestone (1988) for learning finite classes of concepts, and is widely used in machine learning to show convergence and compare the worst case performance of different online learning algorithms. The name 'learning complexity' originates from Maass and Turán (1994), and we prefer this term because it emphasizes the connection with the computational complexity in the sense that the learning complexity may serve as a lower bound on the computational complexity of the algorithm when the cost of updating the current hypothesis is taken as one unit of time. This term also reveals the relation between the complexity of the algorithm and the problem complexity being an intrinsic property of the family of problems, hence generalizing the complexity of the algorithm. Following Maass and Turán (1994), we aim to estimate a lower bound on the learning complexities of all algorithms for a fixed family of problems.

Definition 2.4 (Learning Complexity of a Family of Online Problems)**.** The *learning complexity* of the family of online learning problems $\mathfrak{P}$ is defined as

$$LC(\mathfrak{P}) \quad := \quad \min\{LC(\mathcal{A}, \mathfrak{P}) \,|\, \mathcal{A} \text{ is an algorithm for } \mathfrak{P}\}.$$

2.3 Optimization Problem

We start with a linear programming formulation of the one-class pattern separation problem in the *batch setting* which means that the (*training*) dataset is available before the learning procedure starts. We shall introduce the online learning framework when we consider the incremental subgradient minimization algorithm for this problem in the next section.

As mentioned in the introduction, there is an infinite set $A \subset \mathcal{X}$ of vector representations of the observations from the same source in the Euclidean feature space $\mathcal{X} \subseteq \mathbb{R}^n$. Without loss of generality we assume that the vectors in A are normalized.[2] It is known

[2]Note that not only normalization, but any mapping of a linearly separable dataset onto the unit sphere will fit our learning model.

that this set is linearly separable from the origin, which means that there exists some unit vector $\mathbf{x} \in \mathbb{R}^n$ such that

$$\forall \mathbf{a} \in A: \quad \mathbf{a}^\top \mathbf{x} \quad \geq \quad 0. \tag{2.1}$$

Any such vector $\mathbf{x}$ is called a *feasible vector* of constraint (2.1), and is a normal vector to some separating hyperplane centred at the origin. The separability of the dataset can be measured by the following quantity whose definition is taken from Dunagan and Vempala (2008).

Definition 2.5 (Margin)**.** The *margin* of the dataset $A \subset \mathbb{R}^n$ is the radius of the largest ball inscribed into the polyhedron defined by the set of constraints (2.1)

$$\rho_A \quad = \quad \max_{\|\mathbf{x}\|=1} \min_{\mathbf{a} \in A} \mathbf{a}^\top \mathbf{x}$$

The unit vector $\mathbf{z}$ which realizes the margin

$$\forall \mathbf{a} \in A: \quad \mathbf{a}^\top \mathbf{z} \quad \geq \quad \rho_A$$

is called the centre of the dataset A.

Geometrically, ρ_A is the maximal distance such that some separating hyperplane can be shifted by from the origin in the direction of its normal and still keep the dataset A on its positive side.

Our pattern separation problem implies finding some non-zero vector which satisfies the constraint (2.1) and can be posed as *semi-infinite linear feasibility program*

$$\begin{array}{ll} \text{find} & \mathbf{x} \in \mathbb{R}^n \\ \text{subject to} & \forall \mathbf{a} \in A: \mathbf{a}^\top \mathbf{x} \geq 0, \quad \mathbf{x} \neq \mathbf{0}. \end{array} \tag{2.2}$$

The word *semi-infinite* implies that the linear program has a finite number of variables, but an infinite number of constraints. The vector of variables $\mathbf{x} \in \mathbb{R}^n$ (the *weight vector*) and the margin ρ_A are also known in linear programming as a *design vector* and a *radius of the feasible region* of (2.2) respectively. The set of vectors which satisfy constraints of (2.2) is called a *feasible set* which for feasibility problems coincides with an *optimality set* since there is no objective function. The program (2.2) is equivalent to the second-order conic or semidefinite feasibility programs. The latter will be shown in Chapter 6.

Using the definition of the margin we can parameterize our family of online learning problems $\mathfrak{P}$ in an alternative way. Assuming that the unknown dataset A has margin $\rho \in \mathbb{R}_+$ and centre $\mathbf{z} \in \mathbb{R}^n$, we observe that it is a subset of or equal to the following set

$$\widetilde{A} \quad := \quad \{\tilde{\mathbf{a}} \in \mathbb{R}^n \,|\, \tilde{\mathbf{a}}^\top \mathbf{z} \geq \rho, \, \|\tilde{\mathbf{a}}\| = 1\}.$$

Therefore, without loss of generality we can consider the set $\widetilde{A}$ instead of A because of the infinite size of the latter. Since the former is uniquely defined by ρ and $\mathbf{z}$, we can parameterize an online learning problem as $\mathcal{P}(\mathbf{z}, \rho, \Omega_{\mathbf{z},\rho})$ where $\Omega_{\mathbf{z},\rho}$ is the oracle that provides a counterexample from $\widetilde{A}$. In this case our unknown *target function* is the classifier $f_{\mathbf{z},\rho} : \mathbb{R}^n \to \{0, 1\}$ defined as

$$f_{\mathbf{z},\rho}(\mathbf{y}) \quad := \quad \mathrm{sgn}(\mathbf{y}^\top \mathbf{z} - \rho)$$

where the sign function $\mathrm{sgn} : \mathbb{R} \to \{0, 1\}$ is

$$\mathrm{sgn}(\alpha) \quad := \quad \begin{cases} 1 & \alpha \geq 0 \\ 0 & \alpha < 0. \end{cases}$$

In the perceptron learning we aim to find an approximation of this target function, a classifier $\mathrm{sgn}(\mathbf{y}^\top \mathbf{x})$ whose prediction coincides with the target function on $\widetilde{A}$, but may be different on some part of the unit ball outside $\widetilde{A}$. This is the generalization of the concept learning in two ways: allowing real (not only Boolean inputs) and accepting some difference in predictions outside the target set $\widetilde{A}$. From this viewpoint, our problem is to learn an approximation to the linear threshold function $f_{\mathbf{z},\rho}$ that only makes errors on negative examples.

In this work we mainly consider the learning view and terminology of (2.2), however we will endeavour to provide equivalent notions from optimization theory in order to exploit some results from this field in the learning context and avoid confusing readers familiar with only one of these fields.

2.4 Incremental Subgradient Descent Algorithm

Algorithm 1: Normalized Perceptron Learning Algorithm

Input: a (possibly) infinite set of vectors $A \subset \mathbb{R}^n$.
Output: a vector $\mathbf{x} \in \mathbb{R}^n$ such that $\forall \mathbf{a} \in A : \mathbf{x}^\top \mathbf{a} \geq 0$.

set $t \leftarrow 0$;
set $\mathbf{x}_t \leftarrow \mathbf{0}$;
while *exists* $\mathbf{a} \in A$ *such that* $\mathbf{x}_t^\top \mathbf{a} \leq 0$ **do**
    $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \overline{\mathbf{a}}$;
    $t \leftarrow t + 1$;
end
return $\mathbf{x}_t$

Problem (2.2) can be reformulated as minimization of the piece-wise linear function

$$\begin{array}{ll} \min_{\mathbf{x}} & \sum_{\mathbf{a} \in A} h(\mathbf{a}^\top \mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \mathbb{R}^n, \qquad \mathbf{x} \neq \mathbf{0} \end{array} \tag{2.3}$$

where $h : \mathbb{R} \to \mathbb{R}_+$ according to

$$h(\alpha) \quad := \quad \begin{cases} -\alpha & \text{if } \alpha \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

which is an unconstrained convex program[3]. The function $h(\mathbf{a}^\top \mathbf{x})$ is also known as the *hinge loss function*.

The objective of (2.3) is non-smooth[4], therefore we consider a subgradient descent technique to minimize it, or in other words, try to approach the optimal set by moving in the direction opposite to the subgradient of the objective.

Definition 2.6 (Subgradient)**.** For a given convex function $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ a vector $\mathbf{g} \in \mathbb{R}^n$ is a *subgradient* of $f(\mathbf{x})$ at a point $\mathbf{x} \in \mathbb{R}$ if

$$\forall \mathbf{y} \in \mathbb{R}^n : \quad f(\mathbf{y}) \quad \geq \quad f(\mathbf{x}) + \mathbf{g}^\top (\mathbf{y} - \mathbf{x}).$$

If the function $f(\mathbf{x})$ is differentiable at $\mathbf{x}$ then its subgradient is unique and equals its gradient at this point. Otherwise, there may exist several (or even infinitely many) subgradients at a single point. Geometrically, the subgradient defines a tangential hyperplane to the function surface at $f(\mathbf{x})$. The set of all subgradients of $f(\mathbf{x})$ at $\mathbf{x}$ is called the *subdifferential* of $f(\mathbf{x})$ at $\mathbf{x}$ and is denoted by $\partial f(\mathbf{x})$.

In our case, the subgradient of the objective of (2.3) at some point $\mathbf{x}$ is

$$\mathbf{d} \quad = \quad -\textstyle\sum_{\mathbf{a} \in A : \mathbf{a}^\top \mathbf{x} \leq 0} \mathbf{a}$$

which may be unbounded because of the infinity of A.[5] Therefore, strictly speaking, this algorithm cannot be applied to (2.3) directly. Hence, we switch to the *incremental subgradient algorithm* (Kiwiel, 2004) which minimizes one term of the sum of functions in the objective at a time.

This algorithm is directly applicable to online learning where only one example per trial is presented, so at every trial we solve (2.3) for a single example using the current hypothesis as a starting point.

The classical *perceptron algorithm* (Rosenblatt, 1958) (Algorithm 1) is the simplest incremental subgradient method. It implies choosing some initial approximation $\mathbf{x}_0 \in \mathbb{R}^n$ and then if there is some $\mathbf{a}_t \in A$ such that $\mathbf{a}_t^\top \mathbf{x} \leq 0$ updating it incrementally according to the rule

$$\mathbf{x}_t \quad := \quad \mathbf{x}_{t-1} + \eta_t \mathbf{a}_t$$

[3]The requirement that $\mathbf{x}$ is non-zero can also be included in the function, but we omit this for the sake of clarity of our presentation.

[4]$h(\mathbf{a}^\top \mathbf{x})$ is non-differentiable at $\mathbf{x}$ if $\mathbf{a}^\top \mathbf{x} = 0$.

[5]Moreover, the value of the objective may be also unbounded outside of the optimality set.

where η_t is called a *learning rate* (or a *step size*) until the full subgradient of the objective of (2.3) vanishes, i.e. $\mathbf{x}^\top \mathbf{a} > 0$ holds for all $\mathbf{a} \in A$ (all observed examples are classified correctly).

In our discussion of online learning algorithms following Definition 2.2 in Section 2.2 we mentioned the potential computational advantage of perceptron algorithms over generic online learning algorithms since the former need the current misclassified example only (and do not need all the previous examples) to amend the current hypothesis. However, in general the generic perceptron algorithm does not have a guarantee that its current hypothesis correctly predicts on all the examples on which it has been updated so far until the algorithm converges. But this does not hinder the algorithm from converging in a finite number of steps for the infinite dataset provided it is linearly separable. Hence, its convergence depends only on the margin of the dataset (Block, 1962; Novikoff, 1962; Minsky and Papert, 1988).

Theorem 2.7 (Novikoff (1962))**.** *The perceptron algorithm (Algorithm 1) needs at most* $\frac{1}{\rho_A^2}$ *updates to output some feasible solution of (2.2).*

The perceptron algorithm also has another sensible shortcoming. Observe that convergence of the subgradient algorithm (Algorithm 1) depends not only on the size (or full-dimensional volume) of the feasible region, but first and foremost on its shape. In other words, the method is slow if there is at least one direction in which the optimal set has very small volume and may need even exponential (in terms of the dimension of the problem and number of different examples presented from A) number of updates in the worst case (for an example see Anthony and Shawe-Taylor (1993)).

These problem was tackled independently by Shor (1970b) (for the generic subgradient descent algorithm) and Dunagan and Vempala (2008) (for the perceptron algorithm for solving linear programs). Both these approaches involve some mapping of the instance space to increase the margin. Therefore, in order to proceed with a discussion of these techniques, we need to derive the perceptron update rule in the mapped space.

There is an alternative way one can pose the optimization problem of (2.3) that avoids an unbounded objective function. This can be done by modifying our problem in the following way

$$\begin{array}{ll} \min_{\mathbf{x},\, \|\mathbf{x}\| \leq 1} & \max_{\mathbf{a} \in A} h(\mathbf{a}^\top \mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \mathbb{R}^n, \qquad \mathbf{x} \neq \mathbf{0}. \end{array}$$

However, in this case the set of the subgradients at any fixed $\mathbf{x}$ consists of the most misclassified data points (those which have the lowest inner product with the current weight vector $\mathbf{x}$), which imposes an extra requirement on the oracle and in the general case is not the case in our learning model since we may receive any misclassified example regardless of the amount of loss of our classifier on it. However, we shall come back to this model in Section 3.7 when we derive the lower bounds on the complexity of online perceptron learning.

2.5 Perceptron Update Rule in the Mapped Space

First we consider some very general linear invertible operator $D : \mathbb{R}^n \to \mathbb{R}^n$ defined by some invertible matrix $\mathbf{D} \in \mathbb{R}^n$ and applied to the input space such that $\forall \mathbf{x} \in \mathbb{R}^n$: $D(\mathbf{x}) = \mathbf{D}\mathbf{x}$. It is clear that this mapping preserves the convexity of any set in $\mathbb{R}^n$.

Assume that we apply the subgradient minimization algorithm in the mapped space, so after t updates we obtain the design vector $\mathbf{y}_t = \mathbf{D}\mathbf{x}_t$ and receive $\mathbf{a}_t$ such that $\mathbf{a}_t^\top \mathbf{x}_t < 0$. Now from the invertibility of D we get

$$\mathbf{a}^\top \mathbf{x} \quad = \quad \mathbf{a}^\top \mathbf{D}^{-1} \mathbf{y} \quad = \quad ((\mathbf{D}^{-1})^\top \mathbf{a})^\top \mathbf{y} \tag{2.4}$$

which gives us the expression for the counterexample with which to update in the mapped space, $\mathbf{b} = (\mathbf{D}^{-1})^\top \mathbf{a}$.

It may seem counter-intuitive that we need the inverted operator to map the data to the space which is the image of the input space under the direct mapping although the explanation is quite simple. Geometrically, mapping D corresponds to switching to a new coordinate system in the input space whose axes are the eigenvectors scaled by the eigenvalues of $\mathbf{D}$. Hence, to compute coordinates of a vector in the new coordinate system we need to squash its components by the value of the eigenvalues of $\mathbf{D}$.

Now if we apply some mapping D_{t-1} onto the input space after $(t-1)$'st update of the subgradient algorithm the t-th update in the mapped space becomes

$$\mathbf{D}_{t-1}\mathbf{x}_t \quad := \quad \mathbf{D}_{t-1}\mathbf{x}_{t-1} \quad + \quad \eta_t \overline{\mathbf{b}}_t$$

where $\mathbf{y}_{t-1}$ is the design vector, $\mathbf{D}_{t-1}$ is the mapping matrix after $t-1$ updates, η_t is the step size and $\overline{\mathbf{b}}_t$ is the normalized update vector in the mapped space for the t-th update. This corresponds to the following update rule in the primal space

$$\mathbf{x}_t \quad := \quad \mathbf{x}_{t-1} \quad + \quad \eta \mathbf{D}_{t-1}^{-1} \overline{\mathbf{b}}_t. \tag{2.5}$$

Considering this equation together with the formula for the data points in the mapped space (2.4), we conclude that we can use the mapping of the input space implicitly provided the operator matrix is easy to invert.

2.6 Rescaling Operator

In order to improve convergence of the subgradient algorithm Shor suggested the space *dilation* (also known as *dilatation* or *extension*) procedure[6](Shor, 1970b,a) which is based on non-orthogonal space transformations and aims at reducing the angle between the current subgradient and the direction to the optimum. It stretches the instance space in the direction of the subgradient after making an update on it, and thus squashes the set of subgradients in this direction (recall (2.4)) in the new space. For example, if there are subgradients with components of the opposite signs in some coordinate system, this modification tends to gradually reduce the influence of those components (increases volume in this dimension) and thus prevents the algorithm from unnecessary wiggling around the optimum. In other words, space dilation decreases the angles between different rays in the cone spanned by the data points. In machine learning terms, it gradually suppresses the features that are already learned, and thus reduces the possibility that successive updates deteriorate correct classification of the examples learned earlier.

Alternatively, the dilation can be justified in the following way. Imagine the cone centred at the origin and spanned by the convex hull of the data points. By assumption of the linear separability, this cone must be pointed, i.e. must not contain any line (only rays). The intersection of this cone with the surface of the unit ball is a polygon which may have an infinite number of sides (e.g. a circle) when the dataset is infinite. In this context, the margin is the distance between the hyperplane which contains this polygon (or the maximal distance if there are several hyperplanes which happens when the polygon is not full-dimensional) and the origin. Suppressing the dataset in one of its directions (which are the rays of the cone) squashes the cone in the mapped space, hence the latter intersects the unit ball farther from the origin (the margin becomes larger).

Recently Dunagan and Vempala suggested a *rescaling* procedure to tackle the case of a small margin by the perceptron algorithm (Dunagan and Vempala, 2008). Their approach is based on the same space transformation as the one suggested by Shor, but (as we shall see in Chapter 3) with dilation factor $\lambda = \frac{1}{2}$ which actually corresponds to contraction (not dilation) of the instance space. In this thesis we shall also use their term 'rescaling' for the Shor's operator as we believe that it more accurately describes its effect.[7]

We proceed with a formal definition of the rescaling operator.

[6]Different names exist because the technique was first presented in several articles in Russian and Ukrainian journals which were translated into English independently of one another and sometimes in a different way.

[7]As one could see further on in this section, actual space dilation takes place only when the factor is chosen from a certain range of values (greater than 1). Otherwise, the space may be deflated, contracted, or remain unchanged.

Definition 2.8 (Rescaling Operator, Shor (1970b))**.** Given some unit vector $\boldsymbol{\zeta} \in \mathbb{R}^n$ and some $\lambda \in \mathbb{R}_+$, the mapping $R_\lambda : \mathbb{R}^n \to \mathbb{R}^n$ such that for any $\mathbf{x} \in \mathbb{R}^n$

$$R_\lambda(\boldsymbol{\zeta})\mathbf{x} \quad = \quad \left(\mathbf{I} + (\lambda - 1)\boldsymbol{\zeta}\boldsymbol{\zeta}^\top\right)\mathbf{x} \quad = \quad \mathbf{x} + (\lambda - 1)(\boldsymbol{\zeta}^\top\mathbf{x})\boldsymbol{\zeta}$$

is called the *rescaling operator* in the direction of $\boldsymbol{\zeta}$ with coefficient λ.

In other words, for every vector the rescaling operator multiplies its component in the direction of rescaling by λ. Therefore, when applied to the space, in the case of re-normalization of its output, the operator rotates its coordinate system to move one of the axes closer to the rescaling direction and stretch by λ. Remember that the data in the initial space are then rescaled in the same direction by $\frac{1}{\lambda}$ (recall (2.4)). Depending on the value of λ we distinguish the following types of rescaling operators

- *deflation* if $\lambda = 0$,
- *contraction* if $0 < \lambda < 1$,
- *identity* if $\lambda = 1$,
- *dilation* if $\lambda > 1$.

It is easy to check the following properties of the rescaling operator:

1. $R_{\lambda\mu}(\boldsymbol{\zeta}) = R_\lambda(\boldsymbol{\zeta})R_\mu(\boldsymbol{\zeta}) = R_\mu(\boldsymbol{\zeta})R_\lambda(\boldsymbol{\zeta})$.
2. $R_\lambda(\boldsymbol{\zeta})R_{\frac{1}{\lambda}}(\boldsymbol{\zeta}) = R_1(\boldsymbol{\zeta}) = \mathbf{I}$.

For more details on the operator the reader is referred to Section 3.2 of Shor (1985). Now we are ready to derive the perceptron algorithm with rescaling following Shor (1970b).

2.7 Perceptron Learning with Rescaling

The main modification of the perceptron learning in Shor (1970b) implies applying to the input space the rescaling operator D_t with matrix

$$\mathbf{D}_t \quad := \quad \left(\mathbf{I} + (\lambda - 1)\overline{\mathbf{b}}_t\overline{\mathbf{b}}_t^\top\right)\mathbf{D}_{t-1}$$

(where $\lambda > 1$ and $\mathbf{D}_0 = \mathbf{I}$) after the t'th update made on the normalized counterexample in the dilated space (by operator D_{t-1}) $\mathbf{b}_t$ provided the algorithm has not converged yet. The idea is that the application of D_t to the dataset according to (2.4) contracts it in the direction of $\mathbf{b}_t$. This makes it more difficult for the successive updates to decrease the component of the weight vector corresponding to $\mathbf{b}_t$ or, in other words, to

undo the correct classification of the example $\mathbf{b}_t$. In the extreme case when $\lambda = +\infty$, the rescaling operator projects the whole dataset onto the orthogonal complement of the current misclassified example (deflates the space in its direction). That choice of the rescaling coefficient, however, is impractical because the data space may degenerate before the algorithm converges to the separating weight vector.

According to (2.5), we can make the dilation implicitly without constructing the direct operator D_t after every update since $\mathbf{D}_{t-1}^{-1}$ is as easy to compute iteratively as the matrix of the direct operator. Indeed,

$$\mathbf{D}_t^{-1} \quad = \quad \mathbf{D}_{t-1}^{-1} \left(\mathbf{I} + \left(\tfrac{1}{\lambda} - 1 \right) \overline{\mathbf{b}}_t \overline{\mathbf{b}}_t^\top \right)$$

which is our recursive formula for the matrix of the rescaling operator. By (2.5) the update rule in the primal space is

$$\mathbf{x}_t \quad := \quad \mathbf{x}_{t-1} \quad + \quad \eta \mathbf{D}_{t-1}^{-1} \overline{\mathbf{b}}_t.$$

2.8 Perceptron Learning by Ellipsoid Algorithm

Algorithm 2: Normalized Perceptron Learning by Ellipsoid Algorithm

Input: a (possibly) infinite set A of vectors $\mathbf{a} \in \mathbb{R}^n$ and a parameter $\varrho \in (0, 1)$.
Output: a vector $\mathbf{x} \in \mathbb{R}^n$ such that $\forall \mathbf{a} \in A : \mathbf{x}^\top \mathbf{a} \geq 0$.

set $t \leftarrow 0$;
set $\mathbf{x}_t \leftarrow \mathbf{0}$;
set $\eta_t \leftarrow \frac{1}{n+1}$;
set $\mathbf{B}_t \leftarrow \mathbf{I} \in \mathbb{R}^{n \times n}$;
set $\varrho_t \leftarrow 1$;

while *exists* $\mathbf{a} \in A$ *such that* $\mathbf{x}_t^\top \mathbf{a} \leq 0$ **do**
  if $\varrho_t < \varrho$ **then**
    return *unseparable*
  end
  $\mathbf{b}_{t+1} \leftarrow \mathbf{B}_t^\top \mathbf{a}$;
  $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \eta_t \mathbf{B}_t \overline{\mathbf{b}}_{t+1}$;
  $\eta_{t+1} \leftarrow \frac{n}{\sqrt{n^2-1}} \eta_t$;
  $\mathbf{B}_{t+1} \leftarrow \mathbf{B}_t \left(\mathbf{I} + \left(\frac{\sqrt{n^2-1}}{n+1} - 1 \right) \overline{\mathbf{b}}_{t+1} \overline{\mathbf{b}}_{t+1}^\top \right)$;
  $\varrho_{t+1} \leftarrow \left(\frac{n}{\sqrt{n^2-1}} \right)^{\frac{n-1}{n}} \left(\frac{n}{n+1} \right)^{\frac{1}{n}} \varrho_t$;
  $t \leftarrow t + 1$;
end

return $\mathbf{x}_t$

The implicit application of the rescaling suggests a dual view of this technique: under some certain choice of the rescaling factor it can be interpreted as a cut-off scheme applied to the unit ball that encloses the feasible set of (2.2) (Shor, 1977). Indeed,

one can construct an optimization method which iteratively checks if the centre of the enclosing ball is a feasible point, and otherwise cuts some part of the ball in the direction of the violated constraint and encloses the remaining part into the ellipsoid of the smallest volume (*Löwner-John ellipsoid* (John, 1948)) which is the unit ball in the rescaled space. In this algorithm the update rule corresponds to re-computing the centre of the new bounding ellipsoid containing the feasible region (Shor and Gershovich, 1982).

From this viewpoint the dilation can be seen as changing the metric of the primal space from spherical to ellipsoidal (i.e. scaling and rotating the axes of the coordinate system) that maps the enclosing ellipsoid into a sphere in the dilated space, and thus locks the process. In the primal space, the volume of the enclosing ellipsoid decreases with a constant factor, thus the algorithm converges. Geometrically, this approach 'rounds' the feasible region in the mapped space, hence performance of the algorithm becomes less and less dependent on its structure or shape (the radius of the largest inscribed ball), but depends on its volume. This scheme under rescaling factor $\lambda = \sqrt{\frac{n+1}{n-1}}$ and decreasing step size $\eta_{t+1} = \frac{n}{\sqrt{n^2-1}}\eta_t$ with $\eta_0 = \frac{1}{n+1}$ made it possible to show polynomial solvability of linear programming (Khachiyan, 1979), and the resulting algorithm is known as the *ellipsoid algorithm* (Algorithm 2).[8] This view of rescaling originates from the information-based complexity theory of convex programming (Nemirovsky and Yudin, 1983) where the ellipsoid algorithm was independently derived by Yudin and Nemirovski (1976a,b) as a realizable algorithm whose complexity tends to approximate the optimal information-based complexity of a generic convex program (Nemirovsky and Yudin, 1983).

The perceptron learning by the ellipsoid algorithm is shown as Algorithm 2.[9] The factor ϱ_t is used to compute the ratio of the volumes of the current ellipsoidal approximation of the feasible set, and the assumed maximal ball of radius ρ which is contained in the feasible region. In the one-dimensional case, the ellipsoid algorithm is equivalent to a simple bisection. We have the following result for its mistake bound.

Theorem 2.9 (Ben-Tal and Nemirovski (2001), Theorem 5.2.1)**.** *Algorithm 2 applied to the one class classification problem (2.2) linearly separable with margin $\rho > 0$ makes at most $2n^2 \log \frac{1}{\rho}$ mistakes.*

[8] Although the ellipsoid algorithm is polynomial in the rational number model, it is shown that it is not polynomial in the real number model (Traub and Wożniakowski, 1982). The question of polynomial solvability of linear programming over the field of real numbers still remains open.

[9] In order to handle possible rounding errors, it is advised to use a slightly larger value of the rescaling coefficient $\lambda = \frac{(n+1)\sqrt{2n^2+3}}{n^2\sqrt{2}}$ and update the step size according to

$$\eta_{t+1} \leftarrow \frac{\sqrt{2n^2+3}}{n^2\sqrt{2}}\eta_t$$

for $n \geq 2$ (Grötschel et al., 1988).

The computational complexity of this algorithm is $\mathcal{O}(n^4 \log \frac{1}{\rho})$ (since the computational cost of one update is a factor of n^2 elementary computations) which still can be further optimized (Nemirovski and Nenakhov, 1991).

2.9 Modified Perceptron Algorithm

Algorithm 3: Modified Perceptron Algorithm

Input: a (possibly) infinite set A of vectors $\mathbf{a} \in \mathbb{R}^n$ and a parameter $\sigma \in \mathbb{R}_+$.
Output: a vector $\mathbf{x} \in \mathbb{R}^n$ such that $\forall \mathbf{a} \in A : \overline{\mathbf{x}}^\top \overline{\mathbf{a}} \geq -\sigma$.

repeat
  set $\mathbf{x}$ uniformly at random in $\{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y}\| = 1\}$;
  set $t \leftarrow 0$;
  while *exists* $\mathbf{a} \in A$ *such that* $\overline{\mathbf{x}}^\top \overline{\mathbf{a}} < -\sigma$ **do**
    $t \leftarrow t + 1$;
    $\mathbf{x} \leftarrow (\mathbf{I} - \overline{\mathbf{a}}\overline{\mathbf{a}}^\top)\mathbf{x}$;
*   **if** $\|\mathbf{x}\| = 0$ **then**
      break
    end
    if $t > \frac{\ln n}{\sigma^2}$ **then**
      break
    end
  end
until $\forall \mathbf{a} \in A : \overline{\mathbf{x}}^\top \overline{\mathbf{a}} \geq -\sigma$;

return x

Before discussing the probabilistic perceptron learning algorithm we need to mention the *perceptron algorithm for learning noisy linear threshold functions* (also known as *wiggling* or *relaxed perceptron*) (Blum et al., 1998) because it is a compound part of the former algorithm.

This algorithm (Algorithm 3) is another attempt to tackle classification of the datasets with very small margin. Here our initial goal to find any classifier which *strictly* separates the data is relaxed to a simpler task - to learn an *almost* separating classifier. The word 'almost' means that we allow some relaxation of the constraint (2.1) up to some value $\sigma > 0$, and require that the found unit weight vector satisfies

$$\forall \mathbf{a} \in A : \quad \mathbf{a}^\top \mathbf{x} \quad \geq \quad -\sigma.$$

The algorithm is based on the probabilistic argument and proceeds in a simple way: we start with a random unit weight vector and every time we meet a misclassified example, the component of this example is subtracted from the weight vector. In other words, the deflation operator in the direction of the unit misclassified example $\mathbf{a}_{t-1}$ is applied to

the weight vector (according to our classification of rescaling operators in Section 2.6)

$$\mathbf{x}_t \quad \leftarrow \quad (\mathbf{I} - \overline{\mathbf{a}}_{t-1}\overline{\mathbf{a}}_{t-1}^{\top})\mathbf{x}_{t-1}.$$

Convergence of this procedure is guaranteed by the following theorem.

Theorem 2.10 (Blum et al. (1998),Theorem 3)**.** *If the set A is linearly separable then with some positive probability $1-\delta$ after at most $\frac{\ln n \ln \frac{1}{\delta}}{\sigma^2}$ the Relaxed Perceptron Algorithm (Algorithm 3) returns a vector $\mathbf{x}$ such that*

$$\forall \mathbf{a} \in A : \mathbf{x}^{\top}\overline{\mathbf{a}} \quad \geq \quad -\sigma.$$

Success of the algorithm greatly depends on the random starting vector $\mathbf{x}_0$, and is justified by the observation that if we start with a unit vector that satisfies

$$\mathbf{x}_*^{\top}\mathbf{x}_0 \quad \geq \quad \frac{1}{\sqrt{n}} \tag{2.6}$$

where $\mathbf{x}_*$ is any (not necessarily optimal) separating weight vector (a feasible solution) then the algorithm converges after at most $\frac{\log n}{\sigma^2}$ updates. The probability of this event is positive, hence the algorithm is polynomial in the dimension of the input space and some separation parameter σ.

Chapter 3

Probabilistic Polynomial-time Perceptron Rescaling Algorithm

3.1 Preliminaries on Randomized Algorithms

First we need to recall some basic notions from the theory of randomized algorithms. Unlike deterministic algorithms, their behaviour depends not only on their input, but also on some random choices determined by the output of a random generator which brings certain advantages like avoiding adversarial inputs (sorting and online algorithms) and deadlocks (network routing and distributed systems), improving load balancing (data structures and parallel algorithms), etc. Instead of average case and worst case complexity analysis used for comparison of deterministic algorithms, complexity estimates of randomized algorithms can be based on their *worst case expected* running time which is the maximum of the expected running time for an input taken over all admissible inputs. However, to allow more realistic comparison with *guaranteed* performance of deterministic algorithms, a stronger bound of complexity *with high probability* is used whose notion we adapt from Sanders (1998).

Definition 3.1 (High Probability)**.** A parameterized event $\Upsilon(n)$ occurs *with high probability* if for any $\alpha > 0$ there exists $n_0 \in \mathbb{N}$ such that

$$\forall n \geq n_0 : \quad \mathbf{P}[\Upsilon(n)] \quad \geq \quad 1 - n^{-\alpha}.$$

In other words, a random event happens with high probability if for any fixed positive α, the probability of its complement $\overline{\Upsilon}(n)$ can be bounded by a polynomial $n^{-\alpha}$ starting from some n_0 (i.e. can be made almost zero).

3.2 Probabilistic Perceptron Algorithm with Rescaling

Algorithm 4: Probabilistic Perceptron Algorithm with Rescaling

Input: a (possibly) infinite set A of vectors $\mathbf{a} \in \mathbb{R}^n$.
Output: a nonzero vector $\mathbf{x} \in \mathbb{R}^n$ such that $\forall \mathbf{a} \in A : \mathbf{a}^\top \mathbf{x} \geq 0$.

set $\mathbf{x} \leftarrow \mathbf{0} \in \mathbb{R}^n$;
set $t \leftarrow 0$;
set $\mathbf{H}_t \leftarrow \mathbf{I} \in \mathbb{R}^{n \times n}$;

while *exists* $\mathbf{a} \in A$ *such that* $\mathbf{a}^\top \mathbf{H}_t^\top \mathbf{x} \leq 0$ **do**
  call Algorithm 1 with the set $\mathbf{H}_t A$ for at most $\frac{1}{\varrho^2}$ updates with $\varrho = \frac{1}{32n}$;
  if $\forall \mathbf{a} \in A : \mathbf{a}^\top \mathbf{H}_t^\top \mathbf{x} \geq 0$ **then**
    return $\mathbf{H}_t^\top \mathbf{x}$
  end
  call Algorithm 3 with the set $\mathbf{H}_t A$ and $\sigma = \frac{1}{32n}$;
  if $\forall \mathbf{a} \in A : \mathbf{a}^\top \mathbf{H}_t^\top \mathbf{x} \geq 0$ **then**
    return $\mathbf{H}_t^\top \mathbf{x}$
  end
  $t \leftarrow t + 1$;
  $\mathbf{H}_t \leftarrow (\mathbf{I} + \overline{\mathbf{x}}\overline{\mathbf{x}}^\top)\mathbf{H}_{t-1}$;
end

An alternative way of making the perceptron algorithm polynomial time was introduced by Dunagan and Vempala (2008) (Algorithm 4). They assume that the margin of the dataset is at least ϱ (thus they bound the number of the perceptron updates by the fixed value $\frac{1}{\varrho^2}$), and if it is not, they rescale the dataset (with a fixed factor $\lambda = 2$) to get

$$A' \quad := \quad \{(\mathbf{I} + \overline{\mathbf{x}}\overline{\mathbf{x}}^\top)\mathbf{a} \,|\, \mathbf{a} \in A\},$$

in order to increase the margin, and re-apply the same procedure again. This procedure is justified by the probabilistic argument which proves that the algorithm converges with *high probability*.

Unlike the ellipsoid algorithm, where classification of new examples and learning a separating hyperplane take place in the primal space and rescaling is only implicit (according to (2.5)), in the probabilistic perceptron the weight vector is learned in the rescaled space, and is mapped back only when a feasible point is found. In the computational sense, this implies that we do not need to invert the operator matrix.

Indeed, after the t'th cycle of Algorithm 4 the dataset is rescaled in some direction $\overline{\mathbf{x}}_t$ by the operator with a matrix

$$\mathbf{H}_t \quad = \quad (\mathbf{I} + \overline{\mathbf{x}}_t\overline{\mathbf{x}}_t^\top)\mathbf{H}_{t-1} \quad = \quad \textstyle\prod_{i=0}^{t-1} \left(\mathbf{I} + \overline{\mathbf{x}}_{t-i}\overline{\mathbf{x}}_{t-i}^\top\right).$$

Assume that afterwards we found the feasible solution $\mathbf{y}$ for the rescaled dataset. Then for all $\mathbf{a} \in A$ we have

$$(\mathbf{H}_t\mathbf{a})^\top\mathbf{y} \;>\; 0$$

which can be re-written as

$$(\mathbf{H}_t\mathbf{a})^\top\mathbf{y} \;=\; \mathbf{a}^\top\mathbf{H}_t^\top\mathbf{y} \;=\; \mathbf{a}^\top\mathbf{x} \;>\; 0$$

where $\mathbf{x} = \mathbf{H}_t^\top\mathbf{y}$ (cf. (2.4)). Therefore, the initial input space was implicitly rescaled according to $\mathbf{y} = \left(\mathbf{H}_t^\top\right)^{-1}\mathbf{x}$ which implies that the rescaling operator after t iterations in this case has the following matrix

$$\mathbf{D}_t \;=\; (\mathbf{H}_t^\top)^{-1} \;=\; ((\textstyle\prod_{i=0}^{t-1}\left(\mathbf{I} + (2-1)\overline{\mathbf{x}}_{t-i}\overline{\mathbf{x}}_{t-i}^\top\right))^\top)^{-1} \;=\; \textstyle\prod_{i=0}^{t-1}\left(\mathbf{I} + (\frac{1}{2}-1)\overline{\mathbf{x}}_{t-i}\overline{\mathbf{x}}_{t-i}^\top\right)$$

In contrast to the space dilation in the ellipsoid algorithm, in the case of the probabilistic perceptron the space is implicitly contracted (since $\lambda = \frac{1}{2} < 1$ - recall the types of rescaling in Section 2.6) which means that in the mapped space the component of the rescaling direction in the dataset is stretched (not squashed as in the ellipsoid algorithm). Geometrically, those data vectors which have positive inner product with the rescaling direction are moved closer to it, while all the others are pushed farther (Figure 3.1). The impact of this transformation, therefore, entirely depends on the choice of the direction, hence not every direction in the convex cone of the dataset suits this purpose unless all of them are sufficiently close to the data centre. As shown by the authors of the algorithm, this procedure increases the margin if the inner product of the rescaling direction with the centre of the dataset $\mathbf{z}$ satisfies (2.6). Therefore, the problem of finding a proper direction of rescaling, which is guaranteed to increase the margin, is almost as hard as solving the initial problem (2.2),[1] so they suggest to solve it approximately with the aid of the relaxed perceptron (Algorithm 3). For this purpose they prove a modified version of Theorem 2.10.

Theorem 3.2 (Blum et al. (1998), Theorem 3)**.** *If the set A is linearly separable then with some positive probability $\eta > 0$ after at most $\frac{\ln n}{\sigma^2}$ the Modified Perceptron Algorithm (Algorithm 3) returns a unit vector $\mathbf{x} \in \mathbb{R}^n$ such that*

1. *for all $\mathbf{a} \in A$: $\mathbf{x}^\top\overline{\mathbf{a}} \geq -\sigma$,*
2. *$\mathbf{x}^\top\mathbf{z} \geq \frac{1}{\sqrt{n}}$.*

The proof of the theorem is based on a similar argument to that of Theorem 2.10 although with a stricter initial assumption: it is shown that if the condition (2.6) is met for the random starting point and the unknown centre of the dataset then the algorithm converges.

[1] In fact, in some sense it is even harder than the latter because we can not check if a given vector satisfies this condition since the dataset centre is unknown.

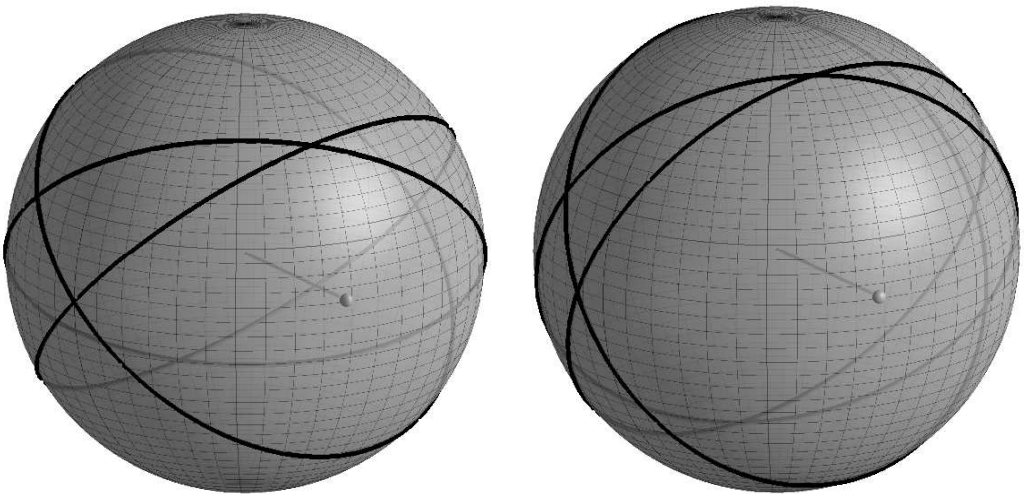


FIGURE 3.1: Illustration of the rescaling procedure. Shown is the feasible region and one feasible point before (**left**) and after (**right**) rescaling with the feasible point.

3.3 Probabilistic Rescaling Procedure

Note that the mere convergence of the Algorithm 3 is not sufficient for the polynomial probabilistic perceptron because not all its outputs satisfy $\mathbf{x}^\top \mathbf{z} \geq \frac{1}{\sqrt{n}}$, and thus can be a suitable direction for rescaling to increase the margin.[2] Since the centre of the dataset is unknown, we cannot strictly determine if this condition is met for a non-zero output of the modified perceptron algorithm. The behaviour of the rescaling procedure is explained by the following lemma.

Lemma 3.3 (Dunagan and Vempala (2008), Lemma 3.3)**.** *Suppose that* $\rho \leq \frac{1}{32n}$ *and* $\sigma \leq \frac{1}{32n}$*. Let* A' *be obtained from* A *by application of the rescaling procedure of one iteration of Algorithm 4. Let* ρ *and* ρ' *be the radii of* A *and* A' *respectively. Then*

1. $\rho' \geq \left(1 - \frac{1}{32n} - \frac{1}{512n^2}\right) \rho$.
2. *With probability at least* $\frac{1}{8}$, $\rho' \geq \left(1 + \frac{1}{3n}\right) \rho$.

Proof. According to Theorem 2.10, with probability at least $\frac{1}{8}$, Algorithm 3 terminates and its output $\mathbf{x}$ satisfies

$$\forall \mathbf{a} \in A: \quad \overline{\mathbf{x}}^\top \overline{\mathbf{a}} \quad \geq \quad -\sigma \quad \text{and} \quad \mathbf{x}^\top \mathbf{z} \quad \geq \quad \frac{1}{\sqrt{n}},$$

[2] Since now we are only interested in the output that meets the second condition of Theorem 3.2, the condition in line (*) of the Algorithm 3 can be modified to checking if $\|\mathbf{x}\| \leq \frac{1}{\sqrt{n}}$.

where $\mathbf{z}$ is the centre of A, although it may also converge and return a vector that does not meet the latter condition.

Assume that A' is the dataset obtained after rescaling all vectors from A in the direction of $\mathbf{x}$ after the first complete iteration of the probabilistic perceptron. We define

$$\mathbf{z}' := \mathbf{z} + \alpha(\mathbf{z}^\top \overline{\mathbf{x}})\overline{\mathbf{x}}, \tag{3.1}$$

even though it may be not the centre of A', we can still use it to bound the radius ρ' of A' from below where α will be specified in the sequel. Therefore,

$$\rho' \;\geq\; \min_{\mathbf{a}' \in A'} \overline{\mathbf{z}}'^\top \overline{\mathbf{a}}' \;=\; \min_{\mathbf{a}' \in A'} \frac{\mathbf{z}'^\top \overline{\mathbf{a}}'}{\|\mathbf{z}'\|}.$$

First we show that the inner product $\mathbf{z}'\overline{\mathbf{a}}'$ is bounded from below. With this aim we expand

$$\begin{aligned} \mathbf{z}'^\top \overline{\mathbf{a}}' &= \left(\frac{\mathbf{a}+(\mathbf{a}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}}{\|\mathbf{a}+(\mathbf{a}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}\|}\right)^\top \mathbf{z}' &= \left(\frac{\overline{\mathbf{a}}+(\overline{\mathbf{a}}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}}{\|\overline{\mathbf{a}}+(\overline{\mathbf{a}}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}\|}\right)^\top \mathbf{z}' \\ &= \frac{[\overline{\mathbf{a}}+(\overline{\mathbf{a}}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}]^\top[\mathbf{z}+\alpha(\mathbf{z}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}]}{\sqrt{1+3(\overline{\mathbf{x}}^\top\overline{\mathbf{a}})^2}} &\geq \frac{\rho+\sigma_{\mathbf{a}}(\mathbf{z}^\top\overline{\mathbf{x}})(1+2\alpha)}{\sqrt{1+3\sigma_{\mathbf{a}}^2}} \end{aligned}$$

where $\sigma_{\mathbf{a}} := \overline{\mathbf{x}}^\top\overline{\mathbf{a}}$. If we choose $\alpha = \frac{1}{2}\left(\frac{\rho}{\mathbf{z}^\top\overline{\mathbf{x}}} - 1\right)$ (although we do not have guarantees that $\mathbf{z}^\top\overline{\mathbf{x}} \neq 0$, but after substituting α in the equality (3.1) above we do not have this problem anymore) then we have

$$\mathbf{z}'^\top \overline{\mathbf{a}}' \;\geq\; \rho\frac{1+\sigma_{\mathbf{a}}}{\sqrt{1+3\sigma_{\mathbf{a}}^2}}.$$

It is possible to show that the function $f(\xi) = \frac{1+\xi}{\sqrt{1+3\xi^2}}$ has a unique point of maximum at $\xi = \frac{1}{3}$ over $[-\sigma, 1]$ where $\sigma \in (0,1)$. The function is also monotonic on $[-\sigma, \frac{1}{3}]$ and $[\frac{1}{3}, 1]$, and $f(-\sigma) = \frac{1-\sigma}{\sqrt{1+3\sigma^2}} \leq 1 = f(1)$ so we proceed to

$$\mathbf{z}'^\top \overline{\mathbf{a}}' \;\geq\; \rho\frac{1-\sigma}{\sqrt{1+3\sigma^2}}.$$

Next we expand the norm of the rescaled centre with α defined as above

$$\|\mathbf{z}'\|^2 \;=\; 1 + (\alpha^2 + 2\alpha)(\mathbf{z}^\top\overline{\mathbf{x}})^2 \;=\; 1 + \frac{\rho^2}{4} + (\mathbf{z}^\top\overline{\mathbf{x}})\left(\frac{\rho}{2} - \frac{3}{4}(\mathbf{z}^\top\overline{\mathbf{x}})\right).$$

We consider two cases:

1. $|\mathbf{z}^\top\overline{\mathbf{x}}| < \frac{1}{\sqrt{n}}$. This may happen with probability at most $\frac{7}{8}$. It is easy to show that the function $g(\omega) = 1 + \frac{\rho^2}{4} + \omega\left(\frac{\rho}{2} - \frac{3}{4}\omega\right)$ has a global maximum at $\omega = \frac{\rho}{3}$ for $\omega \in \mathbb{R}^n$. So we can state

$$\|\mathbf{z}'\|^2 \;\leq\; g(\tfrac{\rho}{3}) \;=\; 1 + \frac{\rho^2}{3}.$$

By virtue of the elementary inequality $\frac{1}{\sqrt{1+\beta}} \geq 1 - \frac{\beta}{2}$ for $\beta \in (-1, 1)$ we have

$$\rho' \;\geq\; \rho\tfrac{1-\sigma}{\sqrt{1+3\sigma^2}\|\mathbf{z}'\|} \;\geq\; \rho(1-\sigma)\left(1-\tfrac{3\sigma^2}{2}\right)\left(1-\tfrac{\rho^2}{6}\right).$$

With the aid of the inequality $(1-a)(1-b)(1-c) \geq (1-a-b-c)$ for $a, b, c \in [-1, 1]$ we obtain

$$\rho' \;\geq\; \rho\left(1-\sigma-\tfrac{3\sigma^2}{2}-\tfrac{\rho^2}{6}\right) \;\geq\; \rho\left(1-\tfrac{1}{32n}-\tfrac{1}{512n^2}\right)$$

for $\rho, \sigma \leq \frac{1}{32n}$.

2. $|\mathbf{z}^\top \overline{\mathbf{x}}| \geq \frac{1}{\sqrt{n}}$ which happens with probability at least $\frac{1}{8}$. Since $\frac{\rho}{3} \leq |\mathbf{z}^\top \overline{\mathbf{x}}|$ we get

$$\|\mathbf{z}'\|^2 \;=\; 1 + \tfrac{\rho^2}{4} + (\mathbf{z}^\top \overline{\mathbf{x}})\tfrac{\rho}{2} - \tfrac{3}{4}(\mathbf{z}^\top \overline{\mathbf{x}})^2 \;\leq\; 1 + \tfrac{\rho^2}{4} + \tfrac{\rho}{2\sqrt{n}} - \tfrac{3}{4n}.$$

Consequently, applying the same elementary inequalities again for $\rho, \sigma \leq \frac{1}{32n}$ we arrive at

$$\begin{aligned}\rho' \;\geq\; \rho(1-\sigma)\left(1-\tfrac{3\sigma^2}{2}\right)\left(1-\tfrac{\rho^2}{8}-\tfrac{\rho}{4\sqrt{n}}+\tfrac{3}{8n}\right) &\geq \rho\left(1-\sigma-\tfrac{3\sigma^2}{2}-\tfrac{\rho^2}{8}-\tfrac{\rho}{4\sqrt{n}}+\tfrac{3}{8n}\right)\\ &\geq \rho\left(1+\tfrac{1}{3n}\right).\end{aligned}$$

□

3.4 Convergence of the Probabilistic Perceptron

Following Lemma 3.3 we know that after one iteration with rescaling of the probabilistic perceptron the margin can either increase by at least $\left(1+\frac{1}{3n}\right)$ (with probability at least $\frac{1}{8}$), or decrease by at most $\left(1-\frac{1}{32n}-\frac{1}{512n^2}\right)$ (with probability at most $\frac{7}{8}$). One can easily see that the expectation of the factor by which the margin is changed after one complete iteration is greater than one, however this is not sufficient for the convergence of the probabilistic perceptron in polynomial time.

The purpose of the rescaling procedure is to make the margin of the dataset at least ϱ in a polynomially bounded number of iterations with *high probability*. Therefore, the sole fact that the expectation of the rescaling factor is greater than 1 does not suit this purpose because it does not necessarily imply that the factor converges to (or does not fall much below, which suffices for this case) its expected value[3]. Therefore, in Dunagan and Vempala (2008) the latter fact is shown to hold with *high probability* by virtue of Chernoff bound (Chernoff, 1952). For the case when the rescaling factor is not less than its expectation by more than some positive ϵ, the margin is proved to grow to at least ϱ

[3]One can construct a counterexample using random walks.

in $\mathcal{O}\left(n \ln \frac{1}{\rho}\right)$ number of iterations of Algorithm 4 which guarantees convergence of the latter (see Theorem 4.1 in the next section).

Theorem 3.4 ((Dunagan and Vempala, 2008, Theorem 3.5))**.** *Algorithm 4 returns a solution to (2.2) in* $\mathcal{O}\left(n \ln \frac{1}{\rho}\right)$ *iterations* $\mathcal{O}\left(n^3 \ln n \ln \frac{1}{\rho}\right)$ *elementary computational operations).*[4]

The number of elementary computations is estimated from the following argument: the complexity of updating the weight vector is $\mathcal{O}(n)$ and we need at most $\frac{1}{\rho^2} = 32^2 n^2$ updates by the plain perceptron, and $\frac{\ln n}{\sigma^2} = 32^2 n^2 \ln n$ by the relaxed perceptron algorithm.

3.5 Optimal Rescaling for the Probabilistic Perceptron

As mentioned by Dunagan and Vempala (2008) if the margin ρ of the dataset A is positive and the centre $\mathbf{z}$ is known, then rescaling in its direction with some positive factor $\nu > 0$ according to

$$\forall \mathbf{a} \in A : \mathbf{a}' = \left(\mathbf{I} + \nu \mathbf{z}\mathbf{z}^\top\right) \mathbf{a} \tag{3.2}$$

increases the margin arbitrarily close to 1 when $\nu \to \infty$.[5] We state this observation as a lemma and provide a formal proof.

Lemma 3.5. *Rescaling the feasible region of (2.2) in the direction of its centre* $\mathbf{z}$ *with some positive factor* ν *according to (3.2) increases its margin arbitrarily close to* 1 *if* $\nu \to \infty$*.*

Proof. We mainly follow the logic of the proof of Lemma 3.3 just using $\nu \geq 1$ and $\mathbf{z}$ instead of $\overline{\mathbf{x}}$ as a direction of rescaling.

Rescaling operator maps the centre $\mathbf{z}$ to

$$\mathbf{z}' = \mathbf{z} + \nu(\mathbf{z}^\top \mathbf{z})\mathbf{z} = (1 + \nu)\mathbf{z}$$

whose norm is equal to $1 + \nu$ so $\overline{\mathbf{z}}' = \mathbf{z}$. In other words, the centre does not change since we rescale the feasible region in its direction.

[4]The number of operations we state is of factor nm lower than in Theorem 3.5 of Dunagan and Vempala (2008) where m is the size of the dataset because we do not count the complexity of obtaining an update vector (a counterexample) which we assume to be provided by the oracle whose complexity we neglect.

[5]This corresponds to rescaling the input space with factor $\frac{1}{1+\nu}$ following the same argument as in Section 3.2.

Now we have

$$\begin{array}{rcccccc} \rho' & \geq & \min_{\mathbf{a}'\in A'} \overline{\mathbf{z}}'^{\top}\overline{\mathbf{a}}' & = & \min_{\mathbf{a}\in A} \frac{\mathbf{z}^{\top}(\mathbf{a}+\nu(\mathbf{a}^{\top}\mathbf{z})\mathbf{z})}{\|\mathbf{a}+\nu(\mathbf{a}^{\top}\mathbf{z})\mathbf{z}\|} & = & \min_{\mathbf{a}\in A} \frac{\mathbf{z}^{\top}(\overline{\mathbf{a}}+\nu(\overline{\mathbf{a}}^{\top}\mathbf{z})\mathbf{z})}{\|\overline{\mathbf{a}}+\nu(\overline{\mathbf{a}}^{\top}\mathbf{z})\mathbf{z}\|} \\ & = & \min_{\mathbf{a}\in A} \frac{\mathbf{z}^{\top}\overline{\mathbf{a}}+\nu\mathbf{z}^{\top}\overline{\mathbf{a}}}{\sqrt{1+(2\nu+\nu^2)(\mathbf{z}^{\top}\overline{\mathbf{a}})^2}} & = & \min_{\mathbf{a}\in A} \frac{\rho_{\mathbf{a}}(1+\nu)}{\sqrt{1+(2\nu+\nu^2)\rho_{\mathbf{a}}^2}} & & \end{array}$$

where $\rho_{\mathbf{a}} = \mathbf{z}^{\top}\overline{\mathbf{a}}$.

It is easy to see that the function $f_\nu(\xi) = \frac{\xi(1+\nu)}{\sqrt{1+(2\nu+\nu^2)\xi^2}}$ with nonnegative ν is strictly monotonically increasing on $\mathbb{R}$ (and bounded with $\frac{1+\nu}{\sqrt{\nu^2+2\nu}}$ from above) so we proceed to

$$\overline{\mathbf{z}}'^{\top}\overline{\mathbf{a}}' \geq \frac{\rho(1+\nu)}{\sqrt{1+(2\nu+\nu^2)\rho^2}}$$

since $\rho_{\mathbf{a}} \in [\rho, 1]$ for all $\mathbf{a} \in A$.

Now for any positive ρ we have the following

$$\lim_{\nu\to\infty} \frac{\rho(1+\nu)}{\sqrt{1+(2\nu+\nu^2)\rho^2}} = \lim_{\nu\to\infty} \frac{\rho(\frac{1}{\nu}+1)}{\sqrt{\frac{1}{\nu^2}+(\frac{2}{\nu}+1)\rho^2}} = 1$$

which proves the statement of the lemma. □

This result is not surprising because when $\nu \to +\infty$ the rescaling factor $\lambda = \frac{1}{\nu+1} \to 0$ therefore the operator that maps the input space converges to the deflation operator. Since the dataset is mapped by the inverse operator this means that the component in the direction of the centre becomes infinitely large comparing to the rest of the space basis, so the perceptron algorithm in the mapped space converges in one update.

3.6 Multiple Rescaling for the Probabilistic Perceptron

However, as mentioned earlier, the centre of the feasible region remains unknown to us in the general case, and to find it is in some sense even more challenging task than to solve a feasibility problem (2.2) which is our initial goal.

Now we are going to demonstrate that increasing the factor ν does not help in the general case when the rescaling direction is just an output of the modified perceptron algorithm. We will follow the argument similar to that of Lemma 3.3 to determine if we can extend the statement of Lemma 3.5 to the generic choice of the rescaling direction used by the probabilistic perceptron and improve the factor $\left(1 + \frac{1}{3n}\right)$ of margin increase (guaranteed by the same lemma) by choosing an appropriate value of ν.

3.6.1 General Expression of the Rescaling Factor

Consider the mapped centre of the feasibility domain

$$\mathbf{z}' := \mathbf{z} + \alpha\nu(\mathbf{z}^\top\overline{\mathbf{x}})\overline{\mathbf{x}} \tag{3.3}$$

(with α to be chosen later) which may not be the centre of the rescaled region, nevertheless can still be used to lower bound the radius. By analogy with the proof of Lemma 3.3 we have

$$\rho' \;\geq\; \min_{\mathbf{a}'\in A'} \overline{\mathbf{z}}'^\top\overline{\mathbf{a}}' \;=\; \min_{\mathbf{a}'\in A'} \frac{\mathbf{z}'^\top\overline{\mathbf{a}}'}{\|\mathbf{z}'\|}.$$

First we need to bound the inner product $\mathbf{z}'^\top\overline{\mathbf{a}}'$ from below for all $\mathbf{a}\in A$, so we expand

$$\begin{aligned}\mathbf{z}'^\top\overline{\mathbf{a}}' &= \mathbf{z}'^\top\left(\frac{\mathbf{a}+\nu(\mathbf{a}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}}{\|\mathbf{a}+\nu(\mathbf{a}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}\|}\right) = \mathbf{z}'^\top\left(\frac{\overline{\mathbf{a}}+\nu(\overline{\mathbf{a}}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}}{\|\overline{\mathbf{a}}+\nu(\overline{\mathbf{a}}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}\|}\right)\\ &= \frac{[\mathbf{z}+\alpha\nu(\mathbf{z}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}]^\top[\overline{\mathbf{a}}+\nu(\overline{\mathbf{a}}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}]}{\sqrt{1+2\nu(\overline{\mathbf{x}}^\top\overline{\mathbf{a}})^2+\nu^2(\overline{\mathbf{x}}^\top\overline{\mathbf{a}})^2}} \;\geq\; \frac{\rho+\sigma_{\mathbf{a}}(\mathbf{z}^\top\overline{\mathbf{x}})(\nu+\alpha\nu(\nu+1))}{\sqrt{1+\nu(\nu+2)\sigma_{\mathbf{a}}^2}}\end{aligned}$$

where we use the fact that $\mathbf{z}^\top\overline{\mathbf{a}} \geq \rho$ and denote $\sigma_{\mathbf{a}} := \overline{\mathbf{x}}^\top\overline{\mathbf{a}}$. Now we choose $\alpha = \frac{1}{\nu(\nu+1)}\left(\frac{\rho}{\mathbf{z}^\top\overline{\mathbf{x}}} - \nu\right)$ (the case $\mathbf{z}^\top\overline{\mathbf{x}} = 0$ does not cause any problem after we substitute α in equation (3.3)) to obtain

$$\mathbf{z}'^\top\overline{\mathbf{a}}' \;\geq\; \rho\frac{1+\sigma_{\mathbf{a}}}{\sqrt{1+\nu(\nu+2)\sigma_{\mathbf{a}}^2}}$$

which allows us to expand our lower bound on ρ' to

$$\rho' \;\geq\; \min_{\mathbf{a}\in A} \rho\frac{1+\sigma_{\mathbf{a}}}{\sqrt{1+\nu(\nu+2)\sigma_{\mathbf{a}}^2}}\frac{1}{\|\mathbf{z}'\|}. \tag{3.4}$$

In order to estimate the expected margin of the rescaled dataset, we need to lower bound the factor multiplying ρ in the left hand side of (3.4) (from now on we shall refer to it as a *rescaling factor*) where we need to consider two cases: when the inner product of the rescaling direction $\overline{\mathbf{x}}$ with the dataset centre $\mathbf{z}$ satisfies the condition of margin increase

$$\overline{\mathbf{x}}^\top\mathbf{z} \;\geq\; \frac{1}{\sqrt{n}} \tag{3.5}$$

which happens with some positive probability η for an arbitrary output of the relaxed perceptron algorithm harnessed in the probabilistic perceptron (*positive rescaling*), and otherwise (*negative rescaling*).

3.6.2 Bounding the Inverted Norm of the Rescaled Centre

We start by considering the lower bound on the inverted norm of the mapped dataset centre $\mathbf{z}'$ in both cases. As will be shown later, this component of the rescaling factor solely contributes to the margin increase when condition (3.5) is met.

The norm of the rescaled centre with α defined as above can be expressed as

$$\begin{array}{rcccl} \|\mathbf{z}'\|^2 & = & [\mathbf{z}+\alpha\nu(\mathbf{z}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}]^\top[\mathbf{z}+\alpha\nu(\mathbf{z}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}] & = & 1+\alpha\nu(\alpha\nu+2)(\mathbf{z}^\top\overline{\mathbf{x}})^2 \\ & = & 1+\frac{1}{\nu+1}\left(\frac{\rho}{\mathbf{z}^\top\overline{\mathbf{x}}}-\nu\right)\left(\frac{1}{\nu+1}\left(\frac{\rho}{\mathbf{z}^\top\overline{\mathbf{x}}}-\nu\right)+2\right)\left(\mathbf{z}^\top\overline{\mathbf{x}}\right)^2 & = & 1+\frac{(\rho-\nu\mathbf{z}^\top\overline{\mathbf{x}})(\rho+(\nu+2)\mathbf{z}^\top\overline{\mathbf{x}})}{(1+\nu)^2} \\ & = & 1+\frac{\rho^2+2\rho\mathbf{z}^\top\overline{\mathbf{x}}-\nu(\nu+2)(\mathbf{z}^\top\overline{\mathbf{x}})^2}{(1+\nu)^2}, & & \end{array}$$

and by virtue of elementary inequality $\frac{1}{\sqrt{1+\beta}} \geq 1-\frac{\beta}{2}$ for $\beta \in (-1,1)$ we arrive at

$$\frac{1}{\|\mathbf{z}'\|} \geq 1+\frac{\nu(\nu+2)(\mathbf{z}^\top\overline{\mathbf{x}})^2-2\rho\mathbf{z}^\top\overline{\mathbf{x}}-\rho^2}{2(1+\nu)^2} = 1+\frac{\nu(\nu+2)\omega^2-2\rho\omega-\rho^2}{2(1+\nu)^2}. \tag{3.6}$$

where we denote $\omega = \mathbf{z}^\top\overline{\mathbf{x}}$.

At this point the difference between positive and negative rescaling can be assessed. As we will see later, this is the only component of the rescaling factor which can be made greater than 1, i.e. can contribute to the margin increase. When condition (3.5) holds we can substitute ω with $\frac{1}{\sqrt{n}}$, so it will be a positive term and at least $\mathcal{O}(\sqrt{n})$ times greater than the other terms in this expression since ρ is at most $\mathcal{O}\left(\frac{1}{n}\right)$. Otherwise, the norm of the mapped centre of the dataset as a function of $\mathbf{z}^\top\overline{\mathbf{x}}$ attains its maximum at $\mathbf{z}^\top\overline{\mathbf{x}} = \frac{\rho}{\nu(\nu+1)}$, hence can be upper bounded by

$$1+\frac{1}{(1+\nu)^2}\left(\rho-\frac{\rho}{\nu+1}\right)\left(\rho+\frac{\rho(\nu+2)}{\nu(\nu+1)}\right) = 1+\left(\frac{\rho}{\nu+1}\right)^2\frac{\nu^2+2\nu+2}{(\nu+1)^2},$$

which, by virtue of the same elementary inequality as in the previous case, bounds from below its inverted norm as

$$\frac{1}{\|\mathbf{z}'\|} \geq 1-\frac{\rho^2}{2(\nu+1)^2}\left(1+\frac{1}{(\nu+1)^2}\right). \tag{3.7}$$

3.6.3 Bounding the Inner Product with the Rescaled Centre

It is possible to show that the function $\varphi(\xi) := \frac{1+\xi}{\sqrt{1+\nu(\nu+2)\xi^2}}$ with nonnegative ν has the unique global maximum at $\xi = \frac{1}{\nu(\nu+2)}$ with the value $\frac{\nu+1}{\sqrt{\nu(\nu+2)}}$.[6] This value is greater or equal to 1 for positive ν as the ratio of the arithmetic and geometric means of ν and $\nu+2$ and decreases to 1 as ν goes to $+\infty$. The function is monotonically increasing on $[-\sigma, \frac{1}{\nu(\nu+2)}]$ and monotonically decreasing on $[\frac{1}{\nu(\nu+2)}, 1]$, so in order to lower bound it we need to compare the values of $\varphi(\xi)$ at $-\sigma$ and 1 for some small $\sigma > 0$. In the case when $\nu \in (-1,0)$ this function does not have any extreme points on $[-1;1]$ because it is strictly monotonically increasing.[7]

[6] Indeed, taking the derivative of $f(\xi)$ and equating it to zero leads to the following equation

$$\frac{1}{\sqrt{1+\nu(\nu+2)\xi^2}} - \frac{(1+\xi)\xi\nu(\nu+2)}{(1+\nu(\nu+2)\xi^2)^{\frac{3}{2}}} = 0$$

that has the unique solution $\xi = \frac{1}{\nu(\nu+2)}$. Second order analysis shows that it is a point of maximum.

[7] When $\nu = 0$ we have an identity operator.

Now we check when $\varphi(-\sigma) \leq \varphi(1)$ holds. We have

$$\frac{1-\sigma}{\sqrt{1+\nu(\nu+2)\sigma^2}} \quad = \quad \varphi(-\sigma) \quad \leq \quad \varphi(1) \quad = \quad \frac{2}{\sqrt{1+\nu(\nu+2)}} \quad = \quad \frac{2}{\nu+1} \tag{3.8}$$

which after squaring and multiplying both sides by $(1+\nu(\nu+2)\sigma)(\nu+1)^2$ transforms to

$$(1-\sigma)^2(1+\nu^2+2\nu) \quad \leq \quad 4(1+\nu(\nu+2)\sigma^2),$$

and after expanding both sides can be simplified to

$$\nu(\nu+2)(3\sigma^2+2\sigma-1)+(-\sigma^2+2\sigma+3) \quad \geq \quad 0.$$

Both polynomials of σ in the right hand side of the inequality above have a common root $\sigma=-1$, therefore we can divide both sides of the inequality by $(\sigma+1)$ since $\sigma \in (0,1)$ and obtain

$$\nu(\nu+2)(3\sigma-1)+(3-\sigma) \quad \geq \quad 0.$$

For practical purposes we may assume that $\sigma \in \left(0, \frac{1}{3}\right)$ since in Algorithm 4 $\sigma = \frac{1}{32n}$ for all $n \in \mathbb{N}$, so we have

$$\nu^2+2\nu+\frac{3-\sigma}{3\sigma-1} \quad \leq \quad 0.$$

The strict inequality holds when $\nu \in \left(-1, -1+2\sqrt{\frac{1-\sigma}{1-3\sigma}}\right)$, equality is attained for $\nu = -1+2\sqrt{\frac{1-\sigma}{1-3\sigma}}$, and for $\nu > -1+2\sqrt{\frac{1-\sigma}{1-3\sigma}}$ we have $\varphi(-\sigma) > \varphi(1) = \frac{2}{\nu+1}$.[8]

3.6.4 Optimal Range of the Rescaling Factor Values

To continue our estimates of the optimal rescaling coefficient we need to combine our analysis of (3.8) together with the lower bounds for the inverted norm of the mapped dataset centre in both cases of positive and negative rescaling ((3.6) and (3.7) respectively). Following our argument for (3.8) we distinguish two ranges of possible values of the rescaling factor ν.

1. $\nu \in (-1,0) \cup \left(0, -1+2\sqrt{\frac{1-\sigma}{1-3\sigma}}\right]$. In this case $\varphi(-\sigma) \leq \varphi(1)$ so by virtue of the same elementary inequality we get

   $$\min_{\mathbf{a}\in A} \frac{1+\sigma_{\mathbf{a}}}{\sqrt{1+\nu(\nu+2)\sigma_{\mathbf{a}}^2}} \quad \geq \quad \frac{1-\sigma}{\sqrt{1+\nu(\nu+2)\sigma^2}} \quad \geq \quad (1-\sigma)\left(1-\frac{\nu(\nu+2)\sigma^2}{2}\right).$$

   Hence, according to (3.4) the positive rescaling factor can be re-written as

   $$\rho' \quad \geq \quad \rho(1-\sigma)\left(1-\frac{\nu(\nu+2)\sigma^2}{2}\right)\left(1+\frac{\nu(\nu+2)\omega^2-2\rho\omega-\rho^2}{2(1+\nu)^2}\right).$$

[8] Indeed, the left hand side of the inequality becomes zero with $\nu = -1 \pm 2\sqrt{\frac{1-\sigma}{1-3\sigma}}$ and the inequality holds at $\nu = 0$.

Only the last two terms of the above bound depend on ν so denoting $\beta := \nu(\nu+2)$ we consider the function

$$\phi_1(\beta) \;:=\; \left(1-\tfrac{\beta\sigma^2}{2}\right)\left(1+\tfrac{\beta\omega^2-2\rho\omega-\rho^2}{2(1+\beta)}\right)$$

whose derivative is

$$\phi_1'(\beta) \;=\; -\tfrac{\sigma^2}{2}\left(1+\tfrac{\beta\omega^2-2\rho\omega-\rho^2}{2(1+\beta)}\right)+\left(1-\tfrac{\beta\sigma^2}{2}\right)\left(\tfrac{\omega^2}{2(1+\beta)}-\tfrac{\beta\omega^2-2\rho\omega-\rho^2}{2(1+\beta)^2}\right).$$

To check the condition of extremum of $\psi_1(\beta)$ we equate its first derivative to zero and after cancelling common factors and simplification obtain the following quadratic equation for β

$$\beta^2\sigma^2(\omega^2+2)+2\beta\sigma^2(\omega^2+2)+2\sigma^2-\sigma^2\rho\omega-\sigma^2\rho^2-2\omega^2-4\rho\omega-2\rho \;=\; 0$$

with roots $\beta = -1 \pm \frac{(\omega+\rho)}{\sigma}\sqrt{\frac{\sigma^2+2}{\omega^2+2}}$. The derivative changes its sign from plus to minus at $-1+\frac{(\omega+\rho)}{\sigma}\sqrt{\frac{\sigma^2+2}{\omega^2+2}}$ which is the point of maximum of $\phi_1(\beta)$. Solving equation (to express ν via β)

$$\nu^2+2\nu \;=\; -1+\tfrac{(\omega+\rho)}{\sigma}\sqrt{\tfrac{\sigma^2+2}{\omega^2+2}}$$

and considering the range of admissible values of ν we find out that the optimal value

$$\nu^* \;=\; \min\left\{-1+\sqrt{\tfrac{(\omega+\rho)}{\sigma}\sqrt{\tfrac{\sigma^2+2}{\omega^2+2}}},\; -1+2\sqrt{\tfrac{1-\sigma}{1-3\sigma}}\right\}.$$

Analyzing both possible values for ν^* one can conclude that after some $N_1 \in \mathbb{N}$ for all $n \geq N_1$

$$-1+\sqrt{\tfrac{(\omega+\rho)}{\sigma}\sqrt{\tfrac{\sigma^2+2}{\omega^2+2}}} \;\geq\; -1+2\sqrt{\tfrac{1-\sigma}{1-3\sigma}}$$

since substituting the values for σ, ρ and lower bound for ω it is easy to see that the former goes to $+\infty$ (because $\frac{\omega}{\sigma} \geq \frac{\frac{1}{\sqrt{n}}}{\frac{1}{32n}} = 32\sqrt{n}$) and the latter goes to 1 as $n \to +\infty$.

Now we estimate the negative rescaling factor which according to (3.7) can be re-written as

$$\rho' \;\geq\; \rho(1-\sigma)\left(1-\tfrac{\nu(\nu+2)\sigma^2}{2}\right)\left(1-\tfrac{\rho^2(\nu^2+2\nu+2)}{2(\nu+1)^4}\right).$$

Again only the last two factors of the bound above depend on ν so defining β in the same way as above we consider the function

$$\psi_1(\beta) \;:=\; \left(1-\tfrac{\beta\sigma^2}{2}\right)\left(1-\tfrac{\rho^2(\beta+2)}{2(\beta+1)^2}\right)$$

with a derivative

$$\psi_1'(\beta) \;=\; -\tfrac{\sigma^2}{2}\left(1-\tfrac{\rho^2(\beta+2)}{2(\beta+1)^2}\right)+\left(1-\tfrac{\beta\sigma^2}{2}\right)\left(1+\tfrac{\rho^2(\beta+3)}{2(1+\beta)^3}\right).$$

To check the condition of extremum of $\phi_1(\beta)$ we equate its first derivative to zero and assuming $\sigma=\rho$ (which is the case in Algorithm 4) after cancelling common factors and simplification get the following cubic equation

$$-\beta^3-3\beta^2-2\beta+2+\rho^2 \;=\; 0$$

which has only one real root

$$\beta \;=\; -1+\tfrac{1}{6}(216+108\rho^2+12\sqrt{312+324\rho^2+81\rho^4})^{\frac{1}{3}}+\frac{2}{(216+108\rho^2+12\sqrt{312+324\rho^2+81\rho^4})^{\frac{1}{3}}}.$$

This is the point of maximum of $\psi_1(\beta)$ because the derivative changes its sign from plus to minus. Since the ρ^2 term is negligibly small in the expression above (in fact, with the suggested value $\rho=\frac{1}{32n}$ it influences at most the fourth digit after the decimal point) we can conclude that for $\beta>\frac{1}{2}$ the negative rescaling term is monotonically decreasing. Solving the inequality

$$\nu^2+2\nu \;\geq\; \tfrac{1}{2}$$

and recalling our argument for the positive rescaling factor the considered optimality range for ν is the following line segment $\nu\in\left(-1+\sqrt{\frac{3}{2}},\;\; -1+2\sqrt{\frac{1-\sigma}{1-3\sigma}}\right)$. Indeed, its left end is the point of maximum of the negative factor and the right end maximizes the positive factor therefore the optimal rescaling factor must be achieved with ν somewhere between these two values.

2. $\nu\geq -1+2\sqrt{\frac{1-\sigma}{1-3\sigma}}$. In this case $\varphi(-\sigma)\geq\varphi(1)$, so using the same elementary inequality now we get

$$\min_{\mathbf{a}\in A}\tfrac{1+\sigma_{\mathbf{a}}}{\sqrt{1+\nu(\nu+2)\sigma_{\mathbf{a}}^2}} \;\geq\; \tfrac{2}{\sqrt{1+\nu^2+2\nu}} \;\geq\; \tfrac{2}{1+\nu}.$$

Therefore, (3.4) in the case of the positive rescaling can be re-written as

$$\rho' \;\geq\; \rho\tfrac{2}{1+\nu}\left(1+\tfrac{\nu(\nu+2)\omega^2-2\rho\omega-\rho^2}{2(1+\nu)^2}\right)$$

so we analyze function

$$\phi_2(\nu) \;:=\; \tfrac{2}{1+\nu}\left(1+\tfrac{\nu(\nu+2)\omega^2-2\rho\omega-\rho^2}{2(1+\nu)^2}\right)$$

along with its derivative

$$\phi_2'(\nu) \;:=\; -\tfrac{2}{(\nu+1)^2}\left(1+\tfrac{\nu(\nu+1)\omega^2-2\rho\omega-\rho^2}{2(1+\nu)^2}\right)+\tfrac{2}{1+\nu}\left(\tfrac{\omega^2}{\nu+1}-\tfrac{\nu(\nu+1)\omega^2-\rho\omega-\rho^2}{(\nu+1)^3}\right).$$

Equating the latter to zero gives us the following equation

$$\nu^2(\omega^2+2)+2\nu(\omega^2+2)+2-2\omega^2-6\rho\omega-3\rho^2 \;=\; 0$$

with roots $\nu=-1\pm\frac{(\rho+\omega)\sqrt{3}}{\sqrt{\omega^2+2}}$. Taking into account the range of admissible values of ν, the optimal value can be found as

$$\nu^* \;=\; \max\left\{-1+2\sqrt{\tfrac{1-\sigma}{1-3\sigma}},\;\; -1+\tfrac{(\rho+\omega)\sqrt{3}}{\sqrt{\omega^2+2}}\right\}.\ ^{9}$$

Comparing both possible values for ν^* one can conclude that after some $N_2\in\mathbb{N}$ for all $n\geq N_2$

$$-1+2\sqrt{\tfrac{1-\sigma}{1-3\sigma}} \;\geq\; -1+\tfrac{(\rho+\omega)\sqrt{3}}{\sqrt{\omega^2+2}}$$

since after substituting the values for σ,ρ and the lower bound for ω it is easy to see that the former goes to 1 and the latter goes to -1 as $n\to+\infty$.

Now we estimate the negative rescaling factor which according to (3.7) can be re-written as

$$\rho' \;\geq\; \rho\tfrac{2}{1+\nu}\left(1-\tfrac{\rho^2(\nu^2+2\nu+2)}{2(\nu+1)^4}\right).$$

Again only the last two terms of the bound above depend on ν so defining β in the same way as earlier we consider the function

$$\psi_2(\beta) \;:=\; \tfrac{2}{\sqrt{\beta+1}}\left(1-\tfrac{\rho^2(\beta+2)}{2(\beta+1)^2}\right)$$

with a derivative

$$\psi_2'(\beta) \;=\; -\tfrac{1}{(\beta+1)^{\frac{3}{2}}}\left(1-\tfrac{\rho^2(\beta+2)}{2(\beta+1)^2}\right)+\tfrac{1}{\sqrt{\beta+1}}\tfrac{\rho^2(\beta+3)}{2(\beta+1)^3}.$$

If we multiply both terms of the derivative by $2(\beta+1)^{\frac{7}{2}}$, it is easy to see that for $\beta>3$, which is true for all values of ν from the considered range, the derivative is strictly negative. Indeed after simplification we get

$$-2\beta^2-(4-2\rho^2)\beta+5\rho^2-2 \;<\; 0$$

(we can even substitute 1 which is an upper bound for any possible expected value of the margin) for the values of β from this range. Therefore, the rescaling factor only decreases when ν grows from $-1+2\sqrt{\frac{1-\sigma}{1-3\sigma}}$ to infinity.

To summarize, we have shown that the optimal value of the factor of positive rescaling resides in $\left(-1+\sqrt{\frac{3}{2}},\,-1+2\sqrt{\frac{1-\sigma}{1-3\sigma}}\right)$ which includes 1. More accurate approximation

[9] Alternatively, it can be seen that the factor of ρ is strictly decreasing as ν grows from $-1+2\sqrt{\frac{1-\sigma}{1-3\sigma}}$ to infinity, hence the values of ν from this range do not give better rescaling (greater increase of the margin) than those of the previous case.

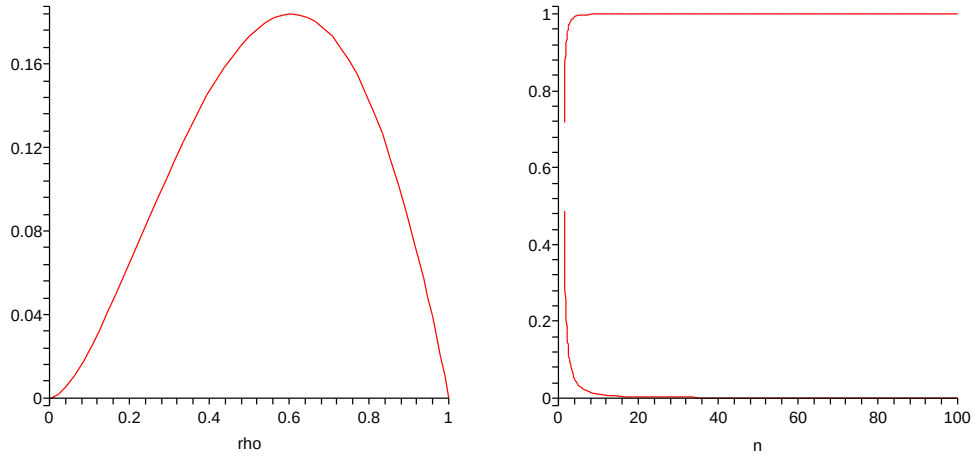
to the optimal ν^* is quite a rather sophisticated task, since it requires solving a transcendent equation (like some in the proof of the convergence theorem for the algorithm in (Dunagan and Vempala, 2008)), and depends on the particular problem (ω, ρ and σ are parameterized by the problem dimension n), thus it is outside of the scope of this work. However, these results have another interesting conclusion: they justify that for the probabilistic perceptron, suggested in Dunagan and Vempala (2008), the contraction operator is indeed the best among the different types of the rescaling operators. This is true because the range of the values of the factor $\nu \in (-1, +\infty)$ we have considered corresponds to the range of the values of rescaling factor $\lambda \in (0, +\infty)$ and we have shown that the optimal λ lies within $(0, 1)$.

3.7 Comparison of Polynomial-time Perceptron Algorithms

As stated in Theorem 2.9, the ellipsoid algorithm needs at most $2n^4 \ln \frac{1}{\rho}$ elementary computations to solve the problem (2.2), and as we discussed in Section 3.4, the probabilistic perceptron algorithm takes at most $\mathcal{O}\left(n^4 \ln n \ln \frac{1}{\rho}\right)$ computations for the same task (as mentioned earlier, in this study we disregard the complexity of the separation oracle). From these estimates it looks like the former clearly outperforms the latter by approximately a factor of $\ln n$ in the worst case. These are, however, the worst case bounds only.

Recall that the probabilistic perceptron is a compound algorithm and the factor $\ln n$ in its complexity bound appears due to its second stage (the relaxed perceptron). This technique is needed in order to supply a proper direction for rescaling the space which satisfies (3.5) and most likely cannot be substituted by a more efficient algorithm because despite its shortcomings it exactly fits the probabilistic argument of the probabilistic perceptron. Therefore, to compare polynomial time perceptron algorithms, we focus our attention on the case when the solution is found before this stage (which usually happens when the actual margin ρ is at least greater than or equal to our lower estimate) and the probabilistic perceptron makes at most $\frac{1}{\rho^2}$ updates using at most $\frac{n}{\rho^2}$ elementary computations (i.e. degenerates to the plain perceptron algorithm). Comparing this bound with the complexity of the ellipsoid algorithm we can determine the range of the actual margins of the datasets in which the former outperforms the latter, and derive an estimate ρ for the probabilistic perceptron. In this chapter we focus on the computational complexity of both algorithms while their comparison with respect to their learning complexities will be presented in the next chapter.

Comparing the computational complexity of both algorithms, we note that the cost of an update in the ellipsoid algorithm is a factor n higher than that in the plain perceptron. We need to establish the relation between ρ and n which satisfies the following inequality



(a) Plot of $\rho^2 \ln \frac{1}{\rho}$ over $(0, 1)$.

(b) The solution of the inequality $\rho^2 \ln \frac{1}{\rho} \geq \frac{1}{2n^3}$, with respect to ρ, is between two curves.

FIGURE 3.2: **Solution of the inequality $\rho^2 \ln \frac{1}{\rho} \geq \frac{1}{2n^3}$.** Figure 3.2(a) shows the plot of the right hand side, and Figure 3.2(b) plots the lower (lower curve) and the upper (upper curve) bounds on ρ with respect to n when the inequality holds.

holds

$$\frac{n}{\rho^2} \quad \leq \quad 2n^4 \ln \frac{1}{\rho}.$$

This transcendent inequality can be simplified to

$$\rho^2 \ln \frac{1}{\rho} \quad \geq \quad \frac{1}{2n^3}.$$

To approximate the solution we plot the left hand side of the inequality above as a function of ρ (Figure 3.2(a)). As one can see, it is concave on $(0, 1)$, and since the right hand side is decreasing when $n \to +\infty$, there exists some $\rho_1, \rho_2 : 0 \leq \rho_1 \leq \rho_2 \leq 1$ such that the inequality holds starting from some value of n for $\rho \in [\rho_1, \rho_2]$. For the case of the equality, the values of ρ_1 and ρ_2 can be obtained analytically. Then we can express

$$\rho_i \quad = \quad \sqrt{-\frac{1}{n^3 W(-\frac{1}{n^3})}}, \quad i \in \{1, 2\}$$

where $W(x)$ is the Lambert function (also known as the *omega function*) which is the inverse of $F(W) = We^W$ (Corless et al., 1996). This function is multivalued in the general case, and does not have real branches for $x < -\frac{1}{e}$ (which corresponds to $n < e^{\frac{1}{3}} < 2$), but has two real branches on $x \in [-\frac{1}{e}, 0)$ which is exactly our case for $n \in [2, +\infty]$ (Figure 3.2(b)). Therefore, for the dataset of dimension $n \geq 2$, the admissible range for the margins which allows the plain perceptron to defeat the ellipsoid algorithm increases and can be seen in the Figure 3.2(b) as the area between two graphs (the

lower is the graph of ρ_1, and the upper depicts the values of ρ_2). This result supports the observation that the ellipsoid algorithm slows down extremely in high-dimensional spaces because the fraction of the iterative volume decrease of the enclosing ellipsoid goes to 1 as the problem dimension n goes to ∞. In this situation, if the margin of the dataset is sufficiently large (at least larger than ρ_1), it is advisable to apply the probabilistic perceptron taking parameter ϱ to be the lower estimate of the margin, and use the generalized convergence framework described in the next chapter (since the value of ϱ is fixed in Dunagan and Vempala (2008) at $\frac{1}{32n}$). Then the probabilistic perceptron can be viewed as a polynomial time framework for the classical perceptron algorithm since it outperforms the ellipsoid algorithm mainly if it finds the solution within its phase equivalent to the plain perceptron. In Chapter 5 we will complete the analysis of the asymptotic behaviour of the perceptron learning in high-dimensional spaces using the results of the information-based complexity theory (Nemirovsky and Yudin, 1983). Experimental comparison of the algorithms will follow in Chapter 6

Chapter 4

Parametric Probabilistic Perceptron Rescaling Algorithm

4.1 Convergence Theorem for Perceptron Rescaling Algorithm

We start with the convergence theorem of the polynomial-time perceptron rescaling algorithm for linear programming (Algorithm 4) in order to expand its proof for our further analysis of the algorithm.

Theorem 4.1 (Dunagan and Vempala (2008), Theorem 3.4)**.** *With high probability, Algorithm 4 finds a feasible solution in* $\mathcal{O}\left(n \ln \frac{1}{\rho_A}\right)$ *iterations where* ρ_A *is the actual margin of the dataset* A*.*

Proof. If the dataset A has a margin $\rho_A \geq \frac{1}{32n}$, then the algorithm will find a solution at the perceptron stage. Otherwise it is sufficient to show that ρ_A will grow to at least $\frac{1}{32n}$ in $\mathcal{O}\left(n \ln \frac{1}{\rho_A}\right)$ iterations. Let X_i be a random variable which becomes 1, if ρ_A grows by a factor of $(1 + 3n)$ or more after the i'th full iteration of the algorithm, and 0 otherwise (*indicator variable*). Let X be the sum of the first T random variables X_i. However random variables X_i are not independently identically distributed (*i.i.d.*), so we define another indicator variable Y_i, which equals 1, if during the i'th iteration of the algorithm, a starting unit vector $\mathbf{x}$ of Algorithm 3 satisfies

$$\mathbf{z}^\top \mathbf{x} \quad \geq \quad \frac{1}{\sqrt{n}},$$

where $\mathbf{z}$ is the centre of A, and 0 otherwise. Variables Y_i are i.i.d., and let Y denote the sum of the first T of Y_i's. Then from Lemma 3.3 we have $X_i \geq Y_i$ for all i (so $X \geq Y$) and

$$\mathbf{E}[Y] \quad \geq \quad \frac{T}{8}.$$

Now Chernoff bound (Chernoff, 1952) gives us

$$\mathbf{P}\left\{Y < (1-\epsilon)\mathbf{E}[Y]\right\} \quad \leq \quad e^{-\frac{\epsilon^2\mathbf{E}[Y]}{2}}.$$

Let ρ_T be the value of ρ_A after T iterations with $T = 4096n\ln\frac{1}{\rho_A}$ and $\epsilon = \frac{1}{16}$. We can expand Chernoff bound to

$$e^{-\frac{\epsilon^2\mathbf{E}[Y]}{2}} \quad \leq \quad e^{-\frac{\epsilon^2 T}{16}} \quad = \quad e^{-\frac{4096n\ln\frac{1}{\rho_A}}{16^2\cdot 16}} \quad = \quad e^{-n\ln\frac{1}{\rho_A}} \quad \leq e^{-n} \tag{4.1}$$

if we recall that $\ln\frac{1}{\rho_A} \geq \ln 32n \geq 5\ln 2 \geq 1$, since $n \in \mathbb{N}$. (4.1) fulfils our definition of high probability (Definition 3.1) because exponential function grows faster than polynomial.

For the case when X does not recede below its expectation by more than ϵ, we have

$$\begin{array}{rcl} \rho_T & \geq & \rho_A\left(1+\frac{1}{3n}\right)^{X}\left(1-\frac{1}{32n}-\frac{1}{512n^2}\right)^{T-X} \\ & \geq & \rho_A\left(1+\frac{1}{3n}\right)^{Y}\left(1-\frac{1}{32n}-\frac{1}{512n^2}\right)^{T-Y} \\ & \geq & \rho_A\left(1+\frac{1}{3n}\right)^{\frac{T}{8}-\epsilon\frac{T}{8}}\left(1-\frac{1}{32n}-\frac{1}{512n^2}\right)^{\frac{7T}{8}+\epsilon\frac{T}{8}} \\ & \geq & \rho_A\left(1+\frac{1}{3n}\right)^{\frac{15T}{128}}\left(1-\frac{1}{32n}-\frac{1}{512n^2}\right)^{\frac{113T}{128}}. \end{array} \tag{4.2}$$

It can be shown that for all $n \in \mathbb{N}$ the right hand side of the latter inequality is bounded by $\rho_A e^{\frac{T}{1024n}}$.[1]

Therefore, we arrive at

$$\rho_T \quad \geq \quad \rho_A e^{\frac{T}{1024n}} \quad = \quad \rho_A e^{\frac{4096n\log\frac{1}{\rho_A}}{1024n}} \quad \geq \quad \rho_A e^{4\log\frac{1}{\rho_A}} \quad = \quad \rho_A e^{\log{\rho_A}^{-4}} \quad = \quad \rho_A^{-3} \quad \geq \quad 32^3 n^3$$

which shows that with probability at least $1-e^{-n}$ in at most T iterations, the radius of the feasible region ρ_A grows to more than $\frac{1}{32n}$. [2] □

[1] For $n \geq 2$ one can use Euler's inequalities

$$\forall x \in \mathbb{R}_+ : \quad \left(1+\tfrac{1}{x}\right)^{x+1} \geq e, \quad \left(1-\tfrac{1}{x}\right)^{x-1} \geq e^{-1}. \tag{4.3}$$

For example, we can transform the second factor in (4.2) to

$$\left(1+\tfrac{1}{3n}\right)^{\frac{15T}{128}} \quad = \quad \left(1+\tfrac{1}{3n}\right)^{\frac{3n+1}{3n+1}\frac{15T}{128}} \quad \geq \quad e^{\frac{15T}{128(3n+1)}}$$

and the third factor to

$$\left(1-\tfrac{1}{32n}-\tfrac{1}{512n^2}\right)^{\frac{113T}{128}} \quad = \quad \left(1-\tfrac{1+16n}{512n^2}\right)^{\left(\frac{512n^2}{16n+1}-1\right)\left(\frac{512n^2}{16n+1}-1\right)^{-1}\frac{113T}{128}} \quad \geq \quad e^{-\frac{16n+1}{512n^2-16n-1}\frac{113T}{128}}$$

Combining two previous inequalities we need to get

$$e^{\frac{15T}{128(3n+1)}-\frac{16n+1}{512n^2-16n-1}\frac{113T}{128}} \quad \geq \quad e^{\frac{T}{1024n}}$$

which (after cancelling $\frac{T}{128}$ from both sides) implies (with the aid of the Maple symbolic processing environment)

$$\frac{16512n^3-19560n^2-1005n+1}{8n(3n+1)(512n^2-16n-1)} \quad \geq \quad 0$$

which holds for $n \geq 2$, but not for $n = 1$ (in this case, it is easy to check the inequality numerically).

[2] Note that for our purpose it is enough to show that

$$\rho_T \quad \geq \quad \rho_A e^{\log\frac{1}{\rho_A}} \quad = \quad \rho_A^0 \quad = \quad 1$$

4.2 Analysis of Convergence

Next we will summarize the main ideas behind the proof of Theorem 4.1 in order to state conditions of convergence of the algorithm parameterized by our lower estimate ϱ of the actual margin of the feasible region ρ_A, and some other values which will be explained in the sequel.

In brief, the algorithm converges if it either finds a feasible point in at most $\frac{1}{\varrho^2}$ updates of its plain perceptron phase (which is guaranteed to take place when the margin ρ_A is at least ϱ), or increases the margin up to ϱ in a finite number of its full iterations (with rescaling) polynomial in the problem dimension and the logarithm of the inverted actual margin which allows the plain perceptron to converge to a solution. The main 'bottleneck' of the algorithm is the perceptron improvement phase when the direction of rescaling is generated: there are only probabilistic guarantees that the size of the feasible region increases when the latter is rescaled in this direction, and may decrease otherwise.

Let us assume that after one complete iteration of the probabilistic perceptron algorithm the margin ρ is either increased by some factor of q_+ with probability at least η, or decreased by the factor of q_- with probability at most $1-\eta$.[3] Then like in (4.2) we need to guarantee that after T iterations the margin will be at least ϱ which can be simplified to

$$\rho_T \quad \geq \quad \rho_A q_+^{T\eta(1-\epsilon)} q_-^{T(1-\eta(1-\epsilon))} \quad \geq \quad \varrho$$

for some $\epsilon \in \mathbb{R}_{++}$ and is equivalent to

$$q_+^{\eta(1-\epsilon)} q_-^{(1-\eta(1-\epsilon))} \quad \geq \quad \sqrt[T]{\frac{\varrho}{\rho_A}}.$$

This allows us to estimate the number of steps by taking the natural logarithm of both sides

$$\frac{\ln \frac{1}{\rho_A} + \ln \varrho}{\eta(1-\epsilon)\ln q_+ + (1-\eta(1-\epsilon))\ln q_-} \quad \leq \quad \frac{\ln \frac{1}{\rho_A}}{\eta(1-\epsilon)\ln q_+ + (1-\eta(1-\epsilon))\ln q_-} \quad \leq \quad T, \tag{4.4}$$

where the last inequality follows from the fact that $\ln \varrho < 0$ because $\varrho < 1$. Although dropping the term $\ln \varrho$ is equivalent to requiring the margin to grow from ρ_A to 1 in T iterations (check the previous inequality), but since we need to estimate T for the sole purpose of showing convergence of the algorithm, we allow this overestimate.

which proves that the statement holds and may allow us to further lower the bound on the right hand side of (4.2), e.g. in the given case just show that

$$\rho_T \quad \geq \quad e^{\frac{1}{4096n}}.$$

[3]The factors q_+ and q_- depend on the dimension n and some other parameters of the algorithm, but for the meantime, in order to avoid cluttering our formulae, we shall drop any indication of these dependencies.

Now we can state the following sufficient condition of convergence of the algorithm

$$\eta(1-\epsilon)\ln q_+ \quad + \quad (1-\eta(1-\epsilon))\ln q_- \quad > \quad 0.$$

Observe that

$$q_+^{\eta(1-\epsilon)} q_-^{(1-\eta(1-\epsilon))} \quad \leq \quad q_+^{\eta} q_-^{1-\eta} \quad \leq \quad \eta q_+ + (1-\eta) q_- \quad = \quad \mathbf{E}(q),$$

where the first inequality holds since $q_+ \geq q_-$ and the second follows from the convexity of the function $f(\alpha) = q_+^{\alpha} q_-^{1-\alpha}$ (as the exponential function with a base $\frac{q_+}{q_-}$ and a constant factor q_-). This transformation shows that convergence with high probability is a stronger property than convergence by expectation (since the latter follows from the former, but not vice versa).

Although the sufficient condition of convergence implies that the margin gradually increases, it still does not guarantee convergence in polynomial time since its right hand side can be infinitesimal. In order to suffice polynomiality of the algorithm we need to bound it by some polynomial, e.g.

$$\eta(1-\epsilon)\ln q_+ \quad + \quad (1-\eta(1-\epsilon))\ln q_- \quad \geq \quad \tfrac{1}{bn^{\beta}} \tag{4.5}$$

where b and β are some positive real numbers. Now we can strengthen (4.4) to

$$T \quad \geq \quad bn^{\beta} \ln \tfrac{1}{\rho_A}, \tag{4.6}$$

where b and β can be obtained by maximizing the right hand side of (4.5).

So far we have considered the condition on the rescaling factors q_+ and q_- which enables the algorithm to increase the margin to at least ϱ and derived the lower estimate on the number of iterations T needed when the positive rescaling happens within ϵ of its expectation. Now we need to prove that the latter event happens with high probability.

To satisfy Definition 3.1, similarly to (4.1) we use the Chernoff's inequality to lower bound the probability of the opposite event by $e^{-n^{\beta}}$.

Like in the proof of Theorem 4.1 consider the indicator variable X_i which becomes 1, if the margin increases by at least q_+ during the i'th iteration of the probabilistic perceptron algorithm, and 0 otherwise, and let us denote the sum of such random variables after T consecutive iterations by X. Since these variables are not i.i.d., we define another sequence of indicator variables, Y_i, which equals 1, if during the i'th iteration of the algorithm, a starting unit vector $\mathbf{x}$ of Algorithm 3 satisfies

$$\mathbf{z}^{\top}\mathbf{x} \quad \geq \quad \tfrac{1}{n^{\frac{\beta}{2}}},$$

where $\mathbf{z}$ is the centre of A and β is the same as in the condition on T, and 0 otherwise. Variables Y_i are i.i.d., and let Y denote the sum of the first T of Y_i's. Then from a modified Lemma 3.3 (which we shall prove in the next section) we have $X_i \geq Y_i$ for all i (so $X \geq Y$) and

$$\mathbf{E}[Y] \quad \geq \quad T\eta.$$

Therefore, the expectation of X is at least $T\eta$, so according to Chernoff bound, we require that the probability that X will fall below its expectation by more than ϵ is

$$e^{-\frac{\epsilon^2 T\eta}{2}} \quad \geq \quad e^{-n^\beta},$$

since we need it to be low in the sense of Definition 3.1. From this inequality we obtain the other lower bound on the number of iterations

$$T \quad \geq \quad \tfrac{2n^\beta}{\epsilon^2\eta}. \tag{4.7}$$

Combining this bound with the estimate (4.6) we can

finalize our formula for numbers of iterations T the parametric algorithm needs to converge

$$\max\left\{\tfrac{2n^\beta}{\epsilon^2\eta},\, bn^\beta \ln\tfrac{1}{\rho_A}\right\} \quad \leq \quad \max\left\{\tfrac{2}{\epsilon^2\eta},\, b\right\} n^\beta \ln\tfrac{1}{\rho_A} \quad \leq \quad T \tag{4.8}$$

with b and β obtained from (4.5).

Let us check this formula for the perceptron rescaling algorithm from Chapter 3, where $\epsilon = \frac{1}{16}$ and $\eta \geq \frac{1}{8}$, and according to (4.2) $\beta = 1$ and $b = 1024$. Having substituted these values into (4.8), we get

$$T \quad \geq \quad \max\left\{2^{12},\, 2^{10}\right\} n \ln\tfrac{1}{\rho_A},$$

we choose $T = 2^{12} n \ln\frac{1}{\rho_A} = 4096 n \ln\frac{1}{\rho_A}$, which is exactly the number used in the proof of Theorem 4.1.

4.3 Parametric Modified Perceptron Algorithm

First we derive a parametric version of the modified perceptron algorithm (Algorithm 3), which we state as Algorithm 5.

For a set of unit vectors (constraints) $A \subset \mathbb{R}^n$ linearly separable from the origin with the margin ρ_A and its centre $\mathbf{z} \in \mathbb{R}^n$, and for a random unit vector $\mathbf{x} \in \mathbb{R}^n$, let us define the following probability

$$\eta_\theta \quad := \quad \mathbf{P}\left\{\mathbf{x}^\top \mathbf{z} \geq \theta\right\}, \tag{4.9}$$

Algorithm 5: Parametric Modified Perceptron Algorithm

Input: a (possibly) infinite set A of vectors $\mathbf{a} \in \mathbb{R}^n$ and parameters σ and $\theta \in \mathbb{R}_{++}$.
Output: a unit vector $\mathbf{x} \in \mathbb{R}^n$ such that $\forall \mathbf{a} \in A : \mathbf{x}^\top \overline{\mathbf{a}} \geq -\sigma$.

repeat
  set $\mathbf{x}$ uniformly at random in $\{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y}\| = 1\}$;
  set $t \leftarrow 0$;
  while *exists* $\mathbf{a} \in A$ *such that* $\overline{\mathbf{x}}^\top \overline{\mathbf{a}} < -\sigma$ **do**
    $t \leftarrow t + 1$;
    $\mathbf{x} \leftarrow (\mathbf{I} - \overline{\mathbf{a}}\overline{\mathbf{a}}^\top)\mathbf{x}$;
*   **if** $\|\mathbf{x}\| = 0$ **then**
      break
    end
    if $t > \frac{2}{\sigma^2} \ln \frac{1}{\theta}$ **then**
      break
    end
  end
until $\forall \mathbf{a} \in A : \overline{\mathbf{x}}^\top \overline{\mathbf{a}} \geq -\sigma$;
return $\overline{\mathbf{x}}$

where $0 < \theta \leq 1$ ($\theta = \frac{1}{\sqrt{n}}$ in Algorithm 3). We assume that η_θ can be either computed or at least non-trivially estimated from below. In order to get $\mathbf{x}$ which meets

$$\mathbf{z}^\top \mathbf{x} \quad \geq \quad \theta, \tag{4.10}$$

we can harness the same modified perceptron algorithm for learning linear threshold functions (Algorithm 3) described in Chapter 2. Now we can extend Theorem 2.10 to a more general form of Algorithm 3 (Algorithm 5), where we seek an output $\mathbf{x}$ that satisfies (4.10).

Theorem 4.2. *For a given set of unit vectors $A \subset \mathbb{R}^n$, linearly separable from the origin, let $\mathbf{y} \in \mathbb{R}^n$ be any unit vector such that for all $\mathbf{a} \in A : \mathbf{a}^\top \mathbf{y} \geq 0$. Then with probability at least η_θ, in at most $\frac{2}{\sigma^2} \ln \frac{1}{\theta}$ updates the parametric modified perceptron algorithm (Algorithm 5) returns a unit vector $\mathbf{x} \in \mathbb{R}^n$ such that*

1. $\forall \mathbf{a} \in A : \mathbf{a}^\top \mathbf{x} \geq -\sigma$,
2. $\mathbf{y}^\top \mathbf{x} \geq \theta$.

Proof. With probability at least η_θ, the random unit vector $\mathbf{x}$ at the start of the algorithm satisfies $\mathbf{y}^\top \mathbf{x} \geq \theta$ for some unit separating weight vector $\mathbf{y}$ for A. The inner product $\mathbf{y}^\top \mathbf{x}$ cannot decrease since

$$(\mathbf{x} - (\mathbf{a}^\top \mathbf{x})\mathbf{a})^\top \mathbf{y} \quad = \quad \mathbf{x}^\top \mathbf{y} - (\mathbf{x}^\top \mathbf{a})(\mathbf{a}^\top \mathbf{y}) \quad \geq \quad \mathbf{x}^\top \mathbf{y}$$

because $\mathbf{x}^\top\mathbf{a}$ has to be negative in order for $\mathbf{a}$ to be an update vector and $\mathbf{a}^\top\mathbf{y}$ is positive by the definition of the separating vector.[4]

On the other hand, the norm of the weight vector $\mathbf{x}$ decreases with every update according to

$$\begin{aligned}(\mathbf{x}-(\mathbf{a}^\top\mathbf{x})\mathbf{a})^\top(\mathbf{x}-(\mathbf{a}^\top\mathbf{x})\mathbf{a}) &= \mathbf{x}^\top\mathbf{x}-2(\mathbf{a}^\top\mathbf{x})^2+(\mathbf{a}^\top\mathbf{x})^2 &= \mathbf{x}^\top\mathbf{x}-(\mathbf{a}^\top\mathbf{x})^2\\ &&\leq \mathbf{x}^\top\mathbf{x}(1-\sigma^2),\end{aligned}$$

since we make an update on $\mathbf{a}$ only if $\mathbf{a}^\top\overline{\mathbf{x}}<-\sigma$.

After t iterations the norm of the weight vector $\mathbf{x}_t$ falls to at most $(1-\sigma^2)^{\frac{t}{2}}$. If $t>\frac{2}{\sigma^2}\ln\frac{1}{\theta}$, then after taking the natural logarithm of the previous estimate of $\|\mathbf{x}_t\|$, we get

$$\ln\|\mathbf{x}_t\| \leq \tfrac{t}{2}\ln(1-\sigma^2) < \tfrac{t}{2}(-\sigma^2) < -\tfrac{2\sigma^2}{2\sigma^2}\ln\tfrac{1}{\theta} = \ln\theta$$

where the second inequality follows from $1+\alpha\leq e^\alpha$ that holds for all $\alpha\in\mathbb{R}$. So after that number of steps the norm of the weight vector $\mathbf{x}$ reduces to less than $\frac{1}{\theta}$. If the starting vector satisfies $\mathbf{y}^\top\mathbf{x}\geq\theta$, then after t updates of the algorithm we have

$$\mathbf{y}^\top\overline{\mathbf{x}}_t = \tfrac{\mathbf{y}^\top\mathbf{x}_t}{\|\mathbf{x}_t\|} > \tfrac{\theta}{\theta} = 1$$

which is impossible. Therefore, if the starting vector satisfies this condition (which happens with probability at least η_θ), then the modified perceptron algorithm converges after at most $\frac{2}{\sigma^2}\ln\frac{1}{\theta}$ updates and its output fulfils this condition, too. □

According to Theorem 4.2 the worst-case number of updates of the parametric modified perceptron algorithm decreases if we increase θ, but at the same time the probability η_θ of generating a suitable starting vector, which is crucial for successful termination of the algorithm, also decreases. It would be very helpful for further analysis of the rescaling procedure to compute some estimates of the dependency of η_θ on θ in (4.9).

Observe that not every output of the modified perceptron algorithm satisfies (4.10), since according to Theorem 4.2 the algorithm converges if this condition is met for any feasible vector, not necessarily optimal. At the same time, the algorithm may converge even if (4.10) does not hold for its starting vector. That is why we also need to consider the case when the probabilistic perceptron algorithm rescales the dataset in the direction

[4]Note that this holds for the case when the inner product $\mathbf{y}^\top\mathbf{x}$ is non-negative. Otherwise when

$$\mathbf{y}^\top\mathbf{x} < \tfrac{(\mathbf{a}^\top\mathbf{x})(\mathbf{y}^\top\mathbf{a})}{1-\sqrt{1-(\mathbf{a}^\top\mathbf{x})^2}} < 0$$

consider the difference

$$\mathbf{y}^\top\mathbf{x}-\tfrac{\mathbf{y}^\top(\mathbf{x}-(\mathbf{a}^\top\mathbf{x})\mathbf{a})}{\|\mathbf{x}-(\mathbf{a}^\top\mathbf{x})\mathbf{a}\|} = \mathbf{y}^\top\mathbf{x}-\tfrac{\mathbf{y}^\top\mathbf{x}-(\mathbf{a}^\top\mathbf{x})(\mathbf{y}^\top\mathbf{a})}{\sqrt{1-2(\mathbf{y}^\top\mathbf{a})^2+(\mathbf{y}^\top\mathbf{a})^2}} = (\mathbf{y}^\top\mathbf{x})\left(\tfrac{\sqrt{1-(\mathbf{y}^\top\mathbf{a})^2}-1}{\sqrt{1-(\mathbf{y}^\top\mathbf{a})^2}}\right)+\tfrac{(\mathbf{a}^\top\mathbf{x})(\mathbf{y}^\top\mathbf{a})}{\sqrt{1-(\mathbf{y}^\top\mathbf{a})^2}} < 0$$

where the last inequality follows from the previous condition.

Many thanks to Olof Barr for pointing this out.

which does not meet (4.10) (negative rescaling) and to lower bound the negative rescaling factor q_- to know the maximal possible decrease of the margin in the worst case.

4.4 Parametric Rescaling Procedure

Now we derive the rescaling procedure for the case when ϱ, σ and θ are not fixed like in the perceptron rescaling algorithm (Algorithm 4), but may vary to exploit extra information about a specific problem to solve. For this purpose we generalize the rescaling procedure of the probabilistic perceptron rescaling algorithm (described by Lemma 3.3) to get a parametric algorithm (Algorithm 6), and state the following lemma.

Algorithm 6: Parametric Probabilistic Perceptron Rescaling Algorithm

Input: a (possibly) infinite set A of vectors $\mathbf{a} \in \mathbb{R}^n$, parameters θ, ϱ and $\sigma \in \mathbb{R}_{++}$.
Output: a unit vector $\mathbf{x} \in \mathbb{R}^n$ such that $\forall \mathbf{a} \in A : \overline{\mathbf{a}}^\top \mathbf{x} \geq 0$.

set $\mathbf{x} \leftarrow \mathbf{0} \in \mathbb{R}^n$;
set $t \leftarrow 0$;
set $\mathbf{H}_t \leftarrow \mathbf{I} \in \mathbb{R}^{n \times n}$;
while *exists* $\mathbf{a} \in A$ *such that* $\overline{\mathbf{a}}^\top \mathbf{H}_t^\top \mathbf{x} \leq 0$ **do**
  call Algorithm 1 with the set $\mathbf{H}_t A$ for at most $\frac{1}{\varrho^2}$ updates;
  if $\forall \mathbf{a} \in A : \overline{\mathbf{a}}^\top \mathbf{H}_t^\top \mathbf{x} \geq 0$ **then**
    return $\frac{\mathbf{H}_t^\top \mathbf{x}}{\|\mathbf{H}_t^\top \mathbf{x}\|}$
  end
  call Algorithm 5 with the set $\mathbf{H}_t A$ and parameters θ and σ;
  if $\forall \mathbf{a} \in A : \overline{\mathbf{a}}^\top \mathbf{H}_t^\top \mathbf{x} \geq 0$ **then**
    return $\frac{\mathbf{H}_t^\top \mathbf{x}}{\|\mathbf{H}_t^\top \mathbf{x}\|}$
  end
  $t \leftarrow t + 1$;
  $\mathbf{H}_t \leftarrow (\mathbf{I} + \overline{\mathbf{x}\mathbf{x}}^\top)\mathbf{H}_{t-1}$;
end

Lemma 4.3. *Suppose that $\rho_A \leq \varrho, \sigma < 1$ and $0 < \frac{\varrho}{3} \leq \theta < 1$. Let A' be obtained from A by application of the rescaling procedure of Algorithm 6. Let ρ_A and ρ'_A be the radii of A and A' respectively. Then*

1. $\rho'_A \geq (1-\sigma)\left(1 - \frac{3\sigma^2}{2}\right)\left(1 - \frac{\varrho^2}{6}\right)\rho_A$.
2. *With probability at least η_θ defined in (4.9),* $\rho'_A \geq (1-\sigma)\left(1 - \frac{3\sigma^2}{2}\right)\left(1 + \frac{3\theta^2}{8} - \frac{\varrho\theta}{4} - \frac{\varrho^2}{8}\right)\rho_A$

Proof. According to Theorem 4.2, with probability at least η_θ Algorithm 5 terminates and its output $\mathbf{x}$ satisfies

$$\forall \mathbf{a} \in A: \quad \mathbf{x}^\top \overline{\mathbf{a}} \;\geq\; -\sigma \quad \text{and} \quad \mathbf{x}^\top \mathbf{z} \;\geq\; \theta,$$

where $\mathbf{z}$ is the centre of A, although it may also return a vector that does not meet the latter condition.

Assume that A' is the dataset obtained after rescaling all vectors from A in the direction of $\mathbf{x}$ after the first complete iteration of the parametric perceptron rescaling algorithm. We define

$$\mathbf{z}' := \mathbf{z} + \alpha(\mathbf{z}^\top \overline{\mathbf{x}})\overline{\mathbf{x}}$$

in order to bound the radius ρ'_A of A' from below where α will be specified in the sequel. Therefore,

$$\rho'_A \;\geq\; \min_{\mathbf{a}' \in A'} \overline{\mathbf{z}}'^\top \overline{\mathbf{a}}' \;=\; \min_{\mathbf{a}' \in A'} \tfrac{\mathbf{z}'^\top \overline{\mathbf{a}}'}{\|\mathbf{z}'\|}.$$

First we show that the inner product $\mathbf{z}'\overline{\mathbf{a}}'$ is bounded from below. With this aim we expand

$$\begin{aligned} \mathbf{z}'^\top \overline{\mathbf{a}}' &= \left(\tfrac{\mathbf{a}+(\mathbf{a}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}}{\|\mathbf{a}+(\mathbf{a}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}\|}\right)^\top \mathbf{z}' = \left(\tfrac{\overline{\mathbf{a}}+(\overline{\mathbf{a}}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}}{\|\overline{\mathbf{a}}+(\overline{\mathbf{a}}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}\|}\right)^\top \mathbf{z}' \\ &= \tfrac{[\overline{\mathbf{a}}+(\overline{\mathbf{a}}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}]^\top[\mathbf{z}+\alpha(\mathbf{z}^\top\overline{\mathbf{x}})\overline{\mathbf{x}}]}{\sqrt{1+3(\overline{\mathbf{x}}^\top\overline{\mathbf{a}})^2}} \;\geq\; \tfrac{\rho_A+\sigma_{\mathbf{a}}(\mathbf{z}^\top\overline{\mathbf{x}})(1+2\alpha)}{\sqrt{1+3\sigma_{\mathbf{a}}^2}} \end{aligned}$$

where $\sigma_{\mathbf{a}} := \overline{\mathbf{x}}^\top\overline{\mathbf{a}}$. If we choose $\alpha = \frac{1}{2}\left(\frac{\rho_A}{\mathbf{z}^\top\overline{\mathbf{x}}} - 1\right)$ (although we do not have guarantees that $\mathbf{z}^\top\overline{\mathbf{x}} \neq 0$, but after substituting α in the expression of $\mathbf{z}'$ above we do not have this problem anymore), then we have

$$\mathbf{z}'^\top \overline{\mathbf{a}}' \;\geq\; \rho_A \tfrac{1+\sigma_{\mathbf{a}}}{\sqrt{1+3\sigma_{\mathbf{a}}^2}}.$$

It is possible to show that the function $f(\xi) = \frac{1+\xi}{\sqrt{1+3\xi^2}}$ has a unique point of maximum at $\xi = \frac{1}{3}$ over $[-\sigma, 1]$ where $\sigma \in (0,1)$. The function is also monotonic on $[-\sigma, \frac{1}{3}]$ and $[\frac{1}{3}, 1]$, and $f(-\sigma) = \frac{1-\sigma}{\sqrt{1+3\sigma^2}} \leq 1 = f(1)$ so we proceed to

$$\mathbf{z}'^\top \overline{\mathbf{a}}' \;\geq\; \rho_A \tfrac{1-\sigma}{\sqrt{1+3\sigma^2}}.$$

Next we expand the norm of the rescaled centre with α defined as above

$$\|\mathbf{z}'\|^2 \;=\; 1 + (\alpha^2 + 2\alpha)(\mathbf{z}^\top\overline{\mathbf{x}})^2 \;=\; 1 + \tfrac{\rho_A^2}{4} + (\mathbf{z}^\top\overline{\mathbf{x}})\left(\tfrac{\rho_A}{2} - \tfrac{3}{4}(\mathbf{z}^\top\overline{\mathbf{x}})\right).$$

We consider two cases:

1. $|\mathbf{z}^\top\overline{\mathbf{x}}| < \theta$. This may happen with probability at most η_θ. It is easy to show that the function $g(\omega) = 1 + \frac{\rho_A^2}{4} + \omega\left(\frac{\rho_A}{2} - \frac{3}{4}\omega\right)$ has a global maximum at $\omega = \frac{\rho_A}{3}$ for $\omega \in \mathbb{R}^n$. So we can state

$$\|\mathbf{z}'\|^2 \;\leq\; g(\tfrac{\rho_A}{3}) \;=\; 1 + \tfrac{\rho_A^2}{3} \;\leq\; 1 + \tfrac{\varrho^2}{3}.$$

By virtue of the elementary inequality $\frac{1}{\sqrt{1+\tau}} \geq 1 - \frac{\tau}{2}$ for $\tau \in (-1, 1)$, we have

$$\rho'_A \;\geq\; \rho_A \tfrac{1-\sigma}{\sqrt{1+3\sigma^2}\|\mathbf{z}'\|} \;\geq\; \rho_A(1-\sigma)\left(1-\tfrac{3\sigma^2}{2}\right)\left(1-\tfrac{\varrho^2}{6}\right).$$

2. $|\mathbf{z}^\top \overline{\mathbf{x}}| \geq \theta$, which happens with probability at least η_θ. Since $\frac{\rho_A}{3} \leq \frac{\varrho}{3} \leq \theta \leq |\mathbf{z}^\top \overline{\mathbf{x}}|$ we get

$$\|\mathbf{z}'\|^2 \;=\; 1 + \tfrac{\rho_A^2}{4} + (\mathbf{z}^\top \overline{\mathbf{x}})\tfrac{\rho_A}{2} - \tfrac{3}{4}(\mathbf{z}^\top \overline{\mathbf{x}})^2 \;\leq\; 1 + \tfrac{\varrho^2}{4} + \tfrac{\varrho\theta}{2} - \tfrac{3\theta^2}{4}.$$

Consequently, applying the same elementary inequality again for $\theta\varrho$ and $\sigma \leq 1$, we arrive at

$$\rho'_A \;\geq\; \rho_A(1-\sigma)\left(1-\tfrac{3\sigma^2}{2}\right)\left(1-\tfrac{\varrho^2}{8}-\tfrac{\rho\theta}{4}+\tfrac{3\theta^2}{8}\right).$$

□

It is easy to see that substituting $\theta = \frac{1}{\sqrt{n}}$, $\varrho = \sigma = \frac{1}{32n}$, and $\eta_\theta = \frac{1}{8}$ one gets Lemma 3.3.

Notice that parameter σ influences both stretching and squashing the feasible region in the same way which demonstrates the trade-off between the number of updates we need to make by the modified perceptron algorithm (at most $\frac{2}{\sigma^2}\ln\frac{1}{\theta}$) and the quality of rescaling: if σ becomes smaller, then the vector we aim to find is closer to the feasible region, thus rescaling will be more sensible, but we need to make more updates in order to find it, and vice versa, if we relax the constraints too much (large σ) then we are able to find the required point faster, but the rescaling will be weaker, so we will need to rescale more times then in the previous case to increase the margin to the same value. The quality of rescaling also directly depends on the value of θ.

Following Lemma 4.3 we express the parametric rescaling factors q_+ and q_- (first introduced in Section 4.2) in terms of our parameters θ, ϱ and σ:

$$q_+(\theta, \varrho, \sigma) \;=\; (1-\sigma)\left(1-\tfrac{3\sigma^2}{2}\right)\left(1-\tfrac{\varrho^2}{8}-\tfrac{\rho\theta}{4}+\tfrac{3\theta^2}{8}\right) \tag{4.11}$$

and

$$q_-(\varrho, \sigma) \;=\; (1-\sigma)\left(1-\tfrac{3\sigma^2}{2}\right)\left(1-\tfrac{\varrho^2}{6}\right). \tag{4.12}$$

In the next section we express our parameters θ, ϱ and σ in terms of the dimension n to calculate the complexity of the parametric perceptron rescaling algorithm.

4.5 Convergence and Complexity

Let us set

$$\varrho_{\gamma,c} \quad = \quad \frac{1}{cn^{\gamma}} \tag{4.13}$$

for some $c \in \mathbb{R}_{++}$,

$$\sigma_{\beta,d} \quad = \quad \frac{1}{dn^{\beta}} \tag{4.14}$$

for some $d \in \mathbb{R}_{++}$, and

$$\theta_{\beta,\kappa} = \frac{\kappa}{n^{\frac{\beta}{2}}} \tag{4.15}$$

for some $\kappa \in \mathbb{R}_{++}$, where β is the same as in Section 4.2 and $\frac{\beta}{2} \leq \gamma \leq \beta$. Now we can define the following probability

$$\eta_{\beta,\kappa} \quad := \quad \mathbf{P}\left\{\mathbf{x}^{\top}\mathbf{z} \geq \theta_{\beta,\kappa}\right\}. \tag{4.16}$$

Substituting the above expressions for θ, ϱ and σ into the formulae for rescaling factors (4.11-4.12) we notice that both d and κ have more influence on rescaling since they are coefficients of the terms of the highest order in these expressions which means that they are also more rigid comparing to c. Considering (4.16) we see that the role of κ will be negligible as n grows, so it is arguable if we can improve much the algorithm by trying to find a unit rescaling direction which satisfies (4.10) with $\theta_{\beta,\kappa}$ where κ is other than 1.

Now we are ready to state the convergence theorem for our parametric algorithm in analogy to Theorem 4.1 and derive its complexity bounds like in Theorem 3.4.

Theorem 4.4. *Assume that a set $A \subset \mathbb{R}^n$ of unit vectors is linearly separable from the origin with margin ρ_A and there exist $\beta, \gamma, b, c, d, \kappa \in \mathbb{R}_{++}$ and some fixed ϵ $(0 < \epsilon < 1)$ such that $\theta_{\beta,\kappa}, \varrho_{\gamma,c}$ and $\sigma_{\beta,d}$ defined in (4.13-4.15) satisfy Lemma 4.3 and*

$$\eta_{\beta,\kappa}(1-\epsilon)\ln q_{+}(\theta_{\beta,\kappa}, \varrho_{\gamma,c}, \sigma_{\beta,d}) \quad + \quad (1-\eta_{\beta,\kappa}(1-\epsilon))\ln q_{-}(\varrho_{\gamma,c}, \sigma_{\beta,d}) \quad \geq \quad \frac{1}{bn^{\beta}} \tag{4.17}$$

for $\eta_{\beta,\kappa}$ as in (4.16) and $q_{+}(\theta_{\beta,\kappa}, \varrho_{\gamma,c}, \sigma_{\beta,d})$ and $q_{-}(\varrho_{\gamma,c}, \sigma_{\beta,d})$ defined in (4.11) and (4.12) respectively.

Then the parametric perceptron rescaling algorithm (Algorithm 6) with $\theta_{\beta,\kappa}, \varrho_{\gamma,c}$ and $\sigma_{\beta,d}$ outputs a feasible solution for A after at most $\mathcal{O}\left(n^{\beta}\ln\frac{1}{\rho_A}\right)$ iterations (at most $\mathcal{O}\left(n^{3\beta}\ln n \ln\frac{1}{\rho_A}\right)$ queries to the oracle).

Proof. As we showed in Section 4.2, since the sufficient condition of polynomial-time convergence (4.5) holds, after at most $T = \max\left\{\frac{2}{\epsilon^2\eta_{\beta,\kappa}}, b\right\}n^{\beta}\ln\frac{1}{\rho_A}$ steps (according to (4.8)) the parametric perceptron rescaling algorithm with high probability returns a feasible solution.

During its single iteration the algorithm uses the perceptron learning algorithm (Algorithm 1) for at most $\frac{1}{\varrho_{\gamma,c}^2} = c^2 n^{2\gamma}$ updates and the parametric modified perceptron algorithm (Algorithm 5) for at most $\frac{2}{\sigma_{\beta,d}^2} \ln \frac{1}{\theta_{\beta,\kappa}} = \beta d^2 n^{2\beta} \ln \frac{n}{\kappa^{\frac{2}{\beta}}}$ updates, so the total number of queries to the oracle during one iteration is at most $\mathcal{O}\left(n^{2\beta} \ln n\right)$. □

In some cases the rescaling factors $q_+(\theta_{\beta,\kappa}, \varrho_{\gamma,c}, \sigma_{\beta,d})$ and $q_-(\varrho_{\gamma,c}, \sigma_{\beta,d})$ allow the following representation:

$$q_+(\theta_{\beta,\kappa}, \varrho_{\gamma,c}, \sigma_{\beta,d}) \quad = \quad \left(1 + \frac{1}{p_+(\beta,n)}\right) \tag{4.18}$$

and

$$q_-(\varrho_{\gamma,c}, \sigma_{\beta,d}) \quad = \quad \left(1 - \frac{1}{p_-(\beta,n)}\right), \tag{4.19}$$

where $p_+(\beta, n)$ and $p_-(\beta, n)$ are some polynomials in n of degree β. Then we can formulate an alternative to (4.17) condition on polynomial-time convergence of the perceptron rescaling algorithm.

Corollary 4.5. *Assume that a set $A \subset \mathbb{R}^n$ of unit vectors is linearly separable from the origin with margin ρ_A and there exist $\beta, \gamma, b, c, d, \kappa \in \mathbb{R}_{++}$ and some fixed ϵ $(0 < \epsilon < 1)$ such that $\theta_{\beta,\kappa}, \varrho_{\gamma,c}$ and $\sigma_{\beta,d}$ defined in (4.13-4.15) satisfy Lemma 4.3, rescaling factors $q_+(\theta_{\beta,\kappa}, \varrho_{\gamma,c}, \sigma_{\beta,d})$ and $q_-(\varrho_{\gamma,c}, \sigma_{\beta,d})$ defined in (4.11) and (4.12) admit representations (4.18) and (4.19) respectively and*

$$\frac{\eta_{\beta,\kappa}(1-\epsilon)}{p_+(\beta,n)+1} \quad - \quad \frac{1-\eta_{\beta,\kappa}(1-\epsilon)}{p_-(\beta,n)-1} \quad \geq \quad \frac{1}{bn^\beta} \tag{4.20}$$

for $\eta_{\beta,\kappa}$ as in (4.16).

Then the parametric perceptron rescaling algorithm (Algorithm 6) with $\theta_{\beta,\kappa}, \varrho_{\gamma,c}$ and $\sigma_{\beta,d}$ outputs a feasible solution for A after at most $\mathcal{O}\left(n^\beta \ln \frac{1}{\rho_A}\right)$ iterations (at most $\mathcal{O}\left(n^{3\beta} \ln n \ln \frac{1}{\rho_A}\right)$ queries to the oracle).

Proof. Using Euler's inequalities (4.3) we can bound the rescaling factors $q_+(\theta_{\beta,\kappa}, \varrho_{\gamma,c}, \sigma_{\beta,d})$ and $q_-(\varrho_{\gamma,c}, \sigma_{\beta,d})$ in (4.18-4.19) as

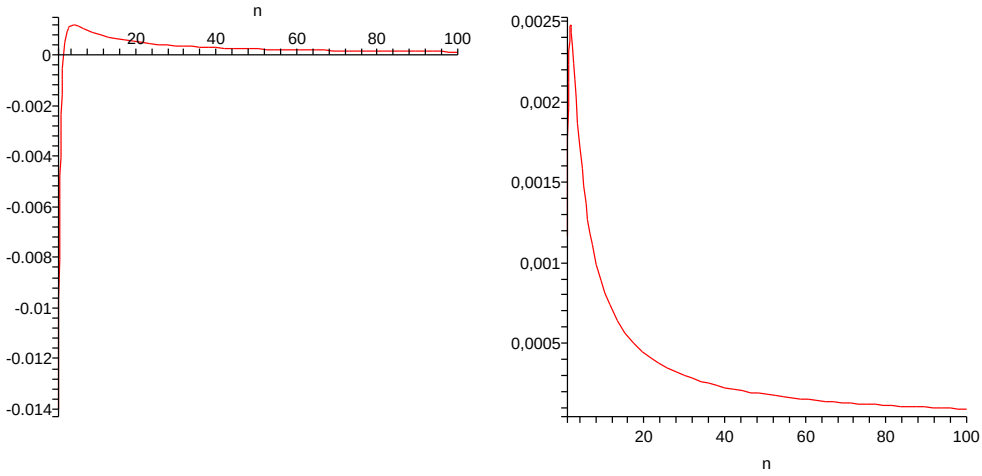
$$\left(1 + \frac{1}{p_+(\beta,n)}\right) \quad = \quad \left(1 + \frac{1}{p_+(\beta,n)}\right)^{\frac{p_+(\beta,n)+1}{p_+(\beta,n)+1}} \quad \geq \quad e^{\frac{1}{p_+(\beta,n)+1}}$$

and

$$\left(1 - \frac{1}{p_-(\beta,n)}\right) \quad = \quad \left(1 - \frac{1}{p_-(\beta,n)}\right)^{\frac{p_-(\beta,n)-1}{p_-(\beta,n)-1}} \quad \geq \quad e^{-\frac{1}{p_-(\beta,n)-1}}.$$

Substituting these lower bounds instead of $q_+(\theta_{\beta,\kappa}, \varrho_{\gamma,c}, \sigma_{\beta,d})$ and $q_-(\varrho_{\gamma,c}, \sigma_{\beta,d})$ into (4.17) and using (4.20) we get Theorem 4.4. □

Condition (4.20) should be more convenient to check than (4.17), since it does not contain logarithmic or exponential functions, only polynomials. However, it is easy to



(a) $\varrho = \frac{1}{4n}$. (b) $\varrho = \frac{1}{16\sqrt{n}}$.

FIGURE 4.1: **Convergence of the parametric perceptron rescaling algorithm.** The function (of the problem dimension n), which corresponds to the sufficient condition of convergence (the right hand side of 4.21), is plotted for the case when our lower estimate of the margin is set as $\varrho = \frac{1}{4n}$ (Figure 4.1(a)) and $\varrho = \frac{1}{16\sqrt{n}}$ (Figure 4.1(b)).

see that (4.20) is a stronger requirement than (4.17) (it implies (4.17), but not vice versa), so there are cases when (4.17) holds while (4.20) does not.

In the parametric perceptron rescaling algorithm we can tune $\varrho_{\gamma,c}$ and $\sigma_{\beta,d}$ to minimize the number of iterations T for any fixed problem (by satisfying (4.17) for its dimension n). By tuning the constants we re-distribute the computational effort between searching for a feasible vector (the perceptron phase) and searching for the rescaling direction (the modified perceptron phase) to find the optimal equilibrium. However, to reduce the complexity of the algorithm by an order of magnitude, we need to lower β, which is a much more complicated task because for this purpose we need to estimate the probability $\eta_{\beta,\kappa}$ for every $\theta_{\beta,\kappa}$. Because of these computational difficulties needed to estimate the required probability, this question has not been studied thoroughly by the author yet.

There is also a theoretical limitation on the amount of potential gain from optimizing the set of parameters of the algorithm, which results from the information-based complexity theory (Nemirovsky and Yudin, 1983), whose results possibly may be adapted to our problem with infinitely many constraints (an infinite dataset A), and state how close is the complexity of the algorithm to the lower bound on the complexity for this class of problems $\mathfrak{P}$ (defined in Section 2.1). We will move to this problem in the next chapter.

4.6 Two Examples

In order to illustrate our parametric probabilistic perceptron algorithm we suggest two sets of possible choice of the parameters of the algorithm which satisfy Theorem 4.4. To check the latter we evaluate and plot the left hand side of the following inequality (which ensures convergence of Algorithm 6)

$$\eta_{\beta,\kappa}(1-\epsilon)\ln q_+(\theta_{\beta,\kappa},\varrho_{\gamma,c},\sigma_{\beta,d}) \quad + \quad (1-\eta_{\beta,\kappa}(1-\epsilon))\ln q_-(\varrho_{\gamma,c},\sigma_{\beta,d}) \quad - \quad \tfrac{1}{bn^\beta} \quad \geq \quad 0 \tag{4.21}$$

as a function of n obtained from (4.17) with β, γ, b, c, d and κ replaced by some fixed values.

The best way to reveal some advantages of the parametric algorithm is to show that we can set γ to other values than just the same as β as in Algorithm 4, which means that we could run the perceptron phase of the algorithm for smaller number of updates, but put more effort in rescaling to increase the margin to larger $\varrho_{\gamma,c}$. Therefore, following the limits $\frac{\beta}{2} \leq \gamma \leq \beta$ set in (4.13) we keep $\gamma = \beta$ in the first case and set $\gamma = \frac{\beta}{2}$ in the second example. In both case we then minimize c and choose b to satisfy (4.21). Note that it is possible to reduce γ to $\frac{\beta}{2}$ because this modification does not change the order of magnitude of the rescaling factors (4.11-4.12).

In both cases we keep $\beta = 1$ and $\kappa = 1$ as in Algorithm 4 because of the aforementioned difficulties with estimating the probability $\eta_{\beta,\kappa}$ for other values of β and κ. We also keep $\epsilon = \frac{1}{16}$ to maintain the same probabilistic framework as in Theorem 4.1 and leave $d = 32$ unchanged as one of the terms of the highest order of magnitude present in both positive and negative rescaling factors.

To maintain inequality (4.21) true for all $n \in \mathbb{N}$, we choose $\gamma = 1$ and $c = 4$ in the first example (Figure 4.1(a)) and $\gamma = \frac{1}{2}$ and $c = 16$ in the second one (Figure 4.1(b)). In both cases it suffices for (4.21) to keep $b = 1024$. The right hand side of the inequality is plotted in Figure 4.1. In Chapter 6 we will test the parametric algorithm with the second choice of the parameters on the positive semidefinite constraint satisfaction problems.

4.7 Summary

In this chapter we analyzed the convergence theorem for the perceptron rescaling algorithm (Algorithm 4) and derived a formula for the number of its iterations, which depends on some parameters of the algorithm. This enabled us to define a more general probabilistic rescaling framework (than that of Lemma 3.3) and suggest a parametric versions of the perceptron rescaling algorithm and the modified perceptron (Algorithm 3) as its compound part. We proved convergence of both algorithms and discussed two examples of a possible choice of the parameters which illustrate one of the advantages of

the parametric algorithm - a possibility to re-distribute its computational effort between its two phases, the deterministic perceptron and the probabilistic rescaling. In Chapter 5 we will try to establish how the complexity of the perceptron rescaling algorithm relates to the complexity of this class of online learning problems. In Chapter 6 we will apply and test the parametric perceptron rescaling algorithm on the positive semidefinite constraint satisfaction problems.

Chapter 5

Learning Complexity of Perceptron Learning

5.1 Basic Notions of Information-based Complexity Theory

First we give a short summary of main notions of the information-based complexity (IBC) theory of convex programming (Nemirovsky and Yudin, 1983).

Consider the following convex programming problem $\mathcal{P}(f) \in \mathfrak{C}$

$$\begin{array}{ll} \min_{\mathbf{x}\in\mathbb{R}^n} & f(\mathbf{x}), \\ \text{subject to} & \|\mathbf{x}\| \leq 1, \end{array} \tag{5.1}$$

where the function $f(\mathbf{x})$ is continuous and convex on $\mathbb{R}^n$. For the purposes of this work we also assume that $f(\mathbf{x})$ is Lipschitz continuous on the unit ball ($\|\mathbf{x}\| \leq 1$)[1] with Lipschitz constant L equal to 1, that is for any $\mathbf{x}_1$ and $\mathbf{x}_2$ that belong to the unit ball,

$$|f(\mathbf{x}_2) - f(\mathbf{x}_1)| \quad \leq \quad \|\mathbf{x}_2 - \mathbf{x}_1\|.$$

The set of functions satisfying these conditions will be denoted by $\mathcal{F}_{\mathbb{R}^n}$. This problem belongs to the family of convex programs $\mathfrak{C}$ over the unit ball parameterized by $f : \mathbb{R}^n \to \mathbb{R}$. Hence, for each $f \in \mathcal{F}_{\mathbb{R}^n}$ we have an instance $\mathcal{P}(f) \in \mathfrak{C}$.

The information-based complexity theory views an optimization problem in a way similar to the approach of experimental science: the problem is a complex object of some fixed type, and the algorithm estimates some property of the object in a series of steps (or experiments) and each of them delivers some fixed amount of information about the

[1] From now on in this chapter under the notion of the "unit ball" we understand the ball centred at the origin unless otherwise stated explicitly.

object from a fixed source. It is assumed that all the required information cannot be obtained in one experiment due to the complexity of the object. For the case of a convex program as an object of investigation this source of information is a *first order oracle*.

Definition 5.1 (First Order Oracle)**.** Given a convex continuous function $f : \mathbb{R}^n \to \mathbb{R}$ ($f \in \mathcal{F}_{\mathbb{R}^n}$) a *first order oracle* is a mapping $\Psi_f : \mathbb{R}^n \to \mathbb{R} \times \mathbb{R}^n$ such that

$$\forall \mathbf{x} \in \mathbb{R}^n : \quad \Psi_f(\mathbf{x}) \quad = \quad (f(\mathbf{x}), f'(\mathbf{x}))$$

where $f'(\mathbf{x})$ is any subgradient of $f(\mathbf{x})$ at $\mathbf{x} \in \mathbb{R}^n$.[2]

It is also assumed that the oracle is *local*.

Definition 5.2 (Local Oracle)**.** An oracle $\Psi_f(\mathbf{x})$ is *local* if for every $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ from $\mathcal{F}_{\mathbb{R}^n}$ and for any $\mathbf{x} \in \mathbb{R}^n$ such that $f_1(\mathbf{x}) \equiv f_2(\mathbf{x})$ in some neighbourhood of $\mathbf{x}$, in the unit ball we have

$$\Psi_{f_1}(\mathbf{x}) \quad = \quad \Psi_{f_2}(\mathbf{x}).$$

Again, similarly to the definition of the separation oracle (Definition 2.1) we allow some non-determinism in the way the oracle chooses one of the subgradients at a fixed $\mathbf{x}$, but for the sake of clarity, we again assume that the oracle is adversarial (i.e. for a fixed algorithm for $\mathcal{P}(f)$ it aims to provide those subgradients which force the algorithm to make as many steps as possible to arrive to the solution of $\mathcal{P}(f)$).

Unlike the case of the online learning problem, which we discussed in Chapter 2, parameterized by an infinite dataset and a separation oracle, in convex programming we consider a first order oracle to be an external object towards the problem determined by a fixed $f \in \mathcal{F}_{\mathbb{R}^n}$.[3]

The cost of one step of an algorithm (or experiment) is considered to be unit which is one of the main differences between the computational and information-based complexities. Therefore, we define a *black box algorithm* for a convex program from $\mathfrak{C}$. The name 'black box' is used to stress the fact that we do not need to know how the algorithm works, but we just view it as a tool which processes limited information about the problem: after a limited number of queries to the oracle, it reports its solution up to the required accuracy ϵ.

Definition 5.3 (Black Box Algorithm)**.** A *black box algorithm* $\mathcal{B}$ for the convex program $\mathcal{P}(f)$ from a family of convex continuous programs $\mathfrak{C}$ is a set of recursive mappings

[2]Since we do not require $f(\mathbf{x})$ to be differentiable.

[3]To be exact, we allow some relaxation here because in the general case the output of the oracle is approximate only, therefore strictly speaking we need to fix the accuracy of the oracle which may influence the accuracy of our solution. As mentioned earlier we also have to define a probabilistic distribution over its possible outputs for the case when $f \in \mathcal{F}_{\mathbb{R}^n}$ is not differentiable at some $\mathbf{x} \in \mathbb{R}^n$ and a subset of subgradients exist. However, for the purpose of this article these simplifications are admissible.

$\mathsf{B}_i : (\mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n)^{i-1} \to \mathbb{R}^n$, $i \in \mathbb{N}$

$$\mathbf{x}_t^{\mathcal{B}} \quad \leftarrow \quad \mathsf{B}_t(\mathbf{x}_1^{\mathcal{B}}, \ldots, \mathbf{x}_{t-1}^{\mathcal{B}}, f(\mathbf{x}_1^{\mathcal{B}}), \ldots, f(\mathbf{x}_{t-1}^{\mathcal{B}}), f'(\mathbf{x}_1^{\mathcal{B}}), \ldots, f'(\mathbf{x}_{t-1}^{\mathcal{B}}))$$

with $(f(\mathbf{x}_i^{\mathcal{B}}), f'(\mathbf{x}_i^{\mathcal{B}})) = \Psi(\mathbf{x}_i^{\mathcal{B}})$, $i \in \{1, \ldots, t-1\}$ such that

$$\forall \epsilon \in \mathbb{R}_{++}, \forall f \in \mathcal{F}_{\mathbb{R}^n}, \exists T = T_{\mathcal{B}}(f, \epsilon) \in \mathbb{N} : \quad f(\mathbf{x}_T^{\mathcal{B}}) - f^* \leq \epsilon, \quad \|\mathbf{x}_T^{\mathcal{B}}\| \leq 1$$

where $f^* = \min_{\mathbf{x} \in \mathbb{R}^n, \|\mathbf{x}\| \leq 1} f(\mathbf{x})$.

In the light of the above definition we need to emphasize that the main property of the black box algorithm $\mathcal{B}$ we are interested in is the function $T_{\mathcal{B}}(f, \epsilon)$. It defines the following measure of the algorithm's performance with respect to the information it needs to solve a fixed problem in $\mathfrak{C}$.

Definition 5.4 (Information-based Complexity of an Algorithm)**.** The *information-based complexity* of the algorithm $\mathcal{B}$ on the family of convex continuous programs $\mathfrak{C}$ is defined as

$$IBC(\mathcal{B}, \mathfrak{C}, \epsilon) \quad = \quad \max_{f \in \mathcal{F}_{\mathbb{R}^n}} T_{\mathcal{B}}(f, \epsilon)$$

which is the minimal number of steps that the black box algorithm $\mathcal{B}$ needs to solve any problem from $\mathfrak{C}$ to the accuracy of at least $\epsilon \in \mathbb{R}_{++}$.

Information-based complexity is also known as *analytical complexity* due to Nesterov (2004). Similarly to learning algorithms we shall focus on the algorithms whose information-based complexity grows polynomially with respect to the dimension of the problem and the required accuracy. This definition also allows us to define the complexity of the family $\mathfrak{C}$ in a similar way.

Definition 5.5 (Information-based Complexity of the Family of Convex Programs)**.** The *information-based complexity* of the family of convex programs $\mathfrak{C}$ over a unit ball is defined as

$$IBC(\mathfrak{C}, \epsilon) \quad = \quad \min_{\mathcal{B}} IBC(\mathcal{B}, \mathfrak{C}, \epsilon).$$

Again, like in the definition of learning complexity (Definition 2.4), the notion of the information-based complexity is also algorithm-oriented, i.e. any upper bounds on the information-based complexity of some family of problems imply that there exists an algorithm to solve any problem of that family in that number of steps. Because of our assumption of the unit cost of a single step of a black box algorithm, the information-based complexity of a family of problems may serve as a lower bound on the computational effort needed to solve any instance of that family. For some families of problems these results may be trivial. For example, if we have a linear problem with a fixed number of constraints, then we can define an oracle which returns the constraint vectors and a cost function in a single response, there exists a black box algorithm with information-based

complexity 1 which is the simplex algorithm (despite its exponential complexity in the worst case).

In the next section we present main results on the upper and lower estimates of the information-based complexity of $\mathfrak{C}$.

5.2 Information-based Complexity of Convex Programming

For the sake of simplicity of the exposition we need to make one more assumption about the family of convex programs $\mathfrak{C}$. We assume that the functions $f \in \mathcal{F}_{\mathbb{R}^n}$ are normalized according to

$$\max_{\mathbf{x}\in\mathbb{R}^n,\, \|\mathbf{x}\|\leq 1} f(\mathbf{x}) - \min_{\mathbf{x}\in\mathbb{R}^n,\, \|\mathbf{x}\|\leq 1} f(\mathbf{x}) \quad \leq \quad 1. \tag{5.2}$$

It is easy to make $\mathcal{F}_{\mathbb{R}^n}$ meet this condition. Indeed, for every $f \in \mathcal{F}_{\mathbb{R}^n}$ which does not satisfy it we can define

$$V_f \quad = \quad \max_{\mathbf{x}\in\mathbb{R}^n,\, \|\mathbf{x}\|\leq 1} f(\mathbf{x}) - \min_{\mathbf{x}\in\mathbb{R}^n,\, \|\mathbf{x}\|\leq 1} f(\mathbf{x}).$$

Now mapping every $f \in \mathcal{F}_{\mathbb{R}^n} \to \frac{f}{V_f}$ we satisfy the normalization requirement so without losing generality we assume that it is met for all $f \in \mathcal{F}_{\mathbb{R}^n}$.

Now we are ready to discuss the results of the information-based complexity theory of $\mathfrak{C}$ from (Nemirovsky and Yudin, 1983, Section 1.4.1.3) (a more specific exposition for this case is given in (Nemirovski, 2002, Chapter 5)) which we state in the form of the theorem.

Theorem 5.6 (Nemirovsky and Yudin (1983), Theorem 4.1.2)**.** *The following estimates of the information-based complexity of $\mathfrak{C}$ under normalization assumption (5.2) with respect to the problem dimension n and required accuracy ϵ hold*

1. *If $\epsilon \leq \frac{1}{n}$:*

$$\Omega(1)n\ln\left(2+\tfrac{1}{\epsilon}\right) \quad \leq \quad IBC(\mathfrak{C})(\epsilon) \quad \leq \quad \mathcal{O}(1)n\ln\left(2+\tfrac{1}{\epsilon}\right).$$

2. *If $\frac{1}{n} < \epsilon < \frac{1}{\sqrt{n}}$:*

$$\Omega(1)n \quad \leq \quad IBC(\mathfrak{C})(\epsilon) \quad \leq \quad \mathcal{O}(1)n\ln\left(2+\tfrac{1}{\epsilon}\right).$$

3. *If $\epsilon \geq \frac{1}{\sqrt{n}}$:*

$$\tfrac{\Omega(1)}{\epsilon^2} \quad \leq \quad IBC(\mathfrak{C})(\epsilon) \quad \leq \quad \tfrac{\mathcal{O}(1)}{\epsilon^2}.$$

Observe that for a fixed accuracy $\epsilon < \frac{1}{n}$ the complexity of the class of problems depends on its dimensionality, and the estimate $\mathcal{O}(1)n\ln\left(2+\frac{1}{\epsilon}\right)$ is attained by the centres of

gravity method (Yudin and Nemirovski, 1976a) (first outlined independently by Levin (1965) and Newman (1965)), which is designed to optimize the information-based complexity of $\mathfrak{C}$. However, this method is impractical in the computational sense and one of its computationally tractable approximations is the ellipsoid algorithm with IBC $\mathcal{O}(1)n^2 \ln\left(2+\frac{1}{\epsilon}\right)$. When the dimension of the family of problems grows, its influence on the complexity decreases. For large scale problems every new accuracy digit increases the number of steps by a factor of 100 thus high accuracy becomes computationally prohibitive. In the next section we will establish the relation between the information-based and learning complexities and derive the estimates of the learning complexity of the online learning problem using the results of Theorem 5.6.

5.3 Piecewise Linear Formulation of a Convex Program

In order to exploit the results of the information-based theory of convex programming in online learning we have to establish the equivalence between those two types of optimization models. In this section we shall demonstrate how any convex program (5.1) with the objective being convex and continuous over $\mathbb{R}^n$ which satisfies the normalization requirement (5.2) and is Lipschitz continuous over the unit ball with Lipschitz constant 1 can be formulated as a minimax problem with piecewise linear cost function.

Let us construct the set of subgradients of $f(\mathbf{x})$ over the unit ball

$$G \;:=\; \{\mathbf{g} \in \mathbb{R}^n \,|\, \exists \mathbf{x_g}, \|\mathbf{x_g}\| \le 1 : \forall \mathbf{x} \in \mathbb{R}^n \, f(\mathbf{x}) \ge f(\mathbf{x_g}) + \mathbf{g}^\top(\mathbf{x}-\mathbf{x_g})\}.$$

Theorem 5.7 (Hiriart-Urruty and Lemaréchal (1993), Proposition 6.2.2)**.** *The set G of subgradients of $f(\mathbf{x})$ over the unit ball is bounded, furthermore*

$$\forall \mathbf{g} \;\in\; G: \;\; \|\mathbf{g}\| \;\le\; L$$

where L is the Lipschitz constant of $f(\mathbf{x})$ over the unit ball.

Therefore, in our case the lengths of the vectors in G are bounded by 1.

Following the definition of a subgradient and the set G, we can replace $f(\mathbf{x})$ over the unit ball by

$$\max_{\mathbf{g}\in G} \quad f(\mathbf{x_g}) + \mathbf{g}^\top(\mathbf{x}-\mathbf{x_g}).$$

Now we separate the constant and linear parts of the function above. Using the set G we can define the following set of pairs

$$B \;:=\; \{(\mathbf{b},\beta) \in \mathbb{R}^n \times \mathbb{R} \,|\, \mathbf{b} = -\mathbf{g},\, \beta = f(\mathbf{x_g}) - \mathbf{g}^\top \mathbf{x_g},\, \mathbf{g} \in G\}.$$

Observe that the values $\beta \in B$ are bounded from above by the value of $f(\mathbf{x})$ at the origin and from below according to

$$\beta \;\geq\; \min_{\mathbf{g}\in G} \beta_{\mathbf{g}} \;=\; \min_{\mathbf{g}\in G} f(\mathbf{x}_{\mathbf{g}}) - \mathbf{g}^{\top}\mathbf{x}_{\mathbf{g}} \;\geq\; f_* - \max_{\mathbf{g}\in G} \mathbf{g}^{\top}\mathbf{x}_{\mathbf{g}} \;\geq\; f_* - 1$$

where f_* is the optimal value of (5.1) over the unit ball and the last transformation is the result of the Cauchy-Schwartz inequality and Theorem 5.7.

Now we can state (5.1) as

$$\begin{array}{cc} \min_{\mathbf{x}\in\mathbb{R}^n} & \max_{(\mathbf{b},\beta)\in B} \beta - \mathbf{b}^{\top}\mathbf{x} \\ \text{subject to} & \|\mathbf{x}\| \leq 1. \end{array} \tag{5.3}$$

Theorem 5.8 (Problem equivalence)**.** *The family of convex problems (5.1) is equivalent to the family of minimax problems (5.3).*

Proof. By construction, as shown earlier in this section, the objectives of both problems are identical over the unit ball, so the equivalence is straightforward. □

In the next section we shall establish the relation between the information-based and learning complexities and derive the estimates of the learning complexity of the online learning problem using the results of Theorem 5.6.

5.4 Learning Complexity of Online Learning Problem

According to our results of comparison of the polynomial time perceptron algorithms with respect to their computational complexities in Section 3.7, if the margin of the dataset is relatively large compared with the cubic inverse of the space dimension, then the classical perceptron algorithm performs better than the ellipsoid algorithm.

In this section we shall focus on the learning complexity of online perceptron learning and address probably the most interesting question of this chapter: is there any lower bound on the learning complexity of the problem (2.2) in the online formulation for the linearly separable set A with a fixed margin ρ such that it is an intrinsic property of the problem itself thus cannot be improved by any algorithm? Answering this question will allow us to see how close are both ellipsoid and perceptron algorithms to the optimal technique, and whether the former methods can be improved. We shall endeavour to answer this question, applying the results of the information-based complexity theory of convex programming (Nemirovsky and Yudin, 1983) to online learning and our target in this section is to estimate from below the learning complexity of the online learning of a linear threshold function.

Solving problem (2.2) we ideally are interested in learning the optimal classifier which is finding the weight vector that achieves the actual margin of the dataset. For a fixed margin ρ of the dataset A we then require the predictions of our solution to coincide with the output of the target function which is equivalent to the following problem

$$\begin{array}{ll} \min_{\mathbf{x}\in\mathbb{R}^n} & \max_{\mathbf{a}\in A}\, \rho - \mathbf{a}^\top \mathbf{x} \\ \text{subject to} & \|\mathbf{x}\| \leq 1 \end{array} \tag{5.4}$$

although the term ρ may be dropped since it is constant for the whole set A. In the online framework the main difference with (2.2) consists in the stricter assumption on the oracle for (5.4) which has to provide one of the *most misclassified* examples (those on which the current hypothesis has the maximal loss) unlike the separation oracle (Definition 2.1) which delivers *any* counterexample.

Since the values of the objective of (5.4) are in $[0, 1+\rho]$ in order to satisfy the normalization requirement (5.2) we need to divide the objective of (5.4) by $1+\rho$ and get an instance of (5.3)

$$\begin{array}{ll} \min_{\mathbf{x}\in\mathbb{R}^n} & \max_{\mathbf{a}\in A}\, \frac{\rho}{1+\rho} - \frac{\mathbf{a}^\top \mathbf{x}}{1+\rho} \\ \text{subject to} & \|\mathbf{x}\| \leq 1. \end{array} \tag{5.5}$$

Note that the solution of (5.5) with the accuracy at least $\frac{\rho}{1+\rho}$ (i.e. $\epsilon \leq \frac{\rho}{1+\rho}$) is at the same time the solution of problem (2.2) which is an optimization model of our online learning problem. Applying Theorem 5.6 to (5.5) we obtain the estimates of the information-based complexity of (5.5) with respect to the dimension n of A and the margin ρ. Observe that the lower bounds on the information-based complexity of (5.5) is the lower estimate of the learning complexity of (2.2) since both oracles Ω_A and Ψ_f are adversarial, but the range of subgradients returned by the latter at any given point $\mathbf{x} \in \mathbb{R}^n$ is a subset of the range of the former (i.e. the number of steps of the "black box" algorithm when it uses outputs of Ψ_f cannot be greater than if it uses Ω_A). Furthermore we can state the following theorem.

Theorem 5.9 (Lower Bounds on Learning Complexity of Online Learning)**.** *The following estimates of the learning complexity of (2.2) in online learning framework with respect to the problem dimension n and the margin ρ hold*

1. *If $\rho \leq \frac{1}{n}$:*
$$\Omega(1) n \ln\left(2 + \tfrac{1}{\rho}\right) \;\leq\; LC(\mathfrak{P}).$$
2. *If $\frac{1}{n} < \rho < \frac{1}{\sqrt{n}}$:*
$$\Omega(1) n \;\leq\; LC(\mathfrak{P}).$$
3. *If $\rho \geq \frac{1}{\sqrt{n}}$:*
$$\tfrac{\Omega(1)}{\rho^2} \;\leq\; LC(\mathfrak{P}).$$

Proof. In this section we have already explained that Theorem 5.6 can be applied to (5.5) and $IBC(\mathfrak{C})\left(\frac{\rho}{1+\rho}\right) \leq LC(\mathfrak{P})$.

It is easy to see that $IBC(\mathfrak{C})(\rho) \leq IBC(\mathfrak{C})\left(\frac{\rho}{1+\rho}\right)$ since $\rho > \frac{\rho}{1+\rho}$ and we do not need more steps to achieve lower accuracy. Therefore we have this chain

$$IBC(\mathfrak{C})(\rho) \quad \leq \quad IBC(\mathfrak{C})\left(\frac{\rho}{1+\rho}\right) \quad \leq \quad LC(\mathfrak{P})$$

Applying Theorem 5.6 to (5.5) with $\epsilon = \rho$ and using the chain of inequalities above we get the statements of the theorem. □

It is possible to show the upper bounds stated in Theorem 5.6 also hold for the learning complexity of $\mathfrak{P}$. Although the argument we have used so far does not work in this case, we can just indicate the algorithms which attain those learning complexities and thus show the upper bounds.

5.5 Analysis of Learning Complexity of Online Learning

Now we need to make sure that the bounds given in Theorem 5.9 are tight, i.e. there exist algorithms that attain them. Let us fix the dimension n and consider the same three cases.

1. $\rho \leq \frac{1}{n}$. The lower complexity $\mathcal{O}(1)n \ln\left(2 + \frac{1}{\rho}\right)$ is achieved by the centres of gravity method which is the optimal for solving a generic convex program of fixed dimension (Yudin and Nemirovski, 1976a). The algorithm implies computing the gravity centre of a convex body which contains the feasible region (e.g. the ball at the first iteration), cutting the part which is either not feasible or corresponds to increase of the objective, and repeating this procedure for the remaining part of the body until a termination criterion is met. This algorithm is based on the observation of exponential decrease of the volume of the enclosing body which guarantees polynomial time convergence of the method and optimizes the information-based complexity of $\mathfrak{C}$. However this algorithm is not practical due to the high cost of computation of the gravity centre of an arbitrary convex body. This idea of *central cuts* dates back to (Levin, 1965; Newman, 1965) and its modifications suggest to use a n-dimensional simplex as a circumscribing body (Yamnitsky and Levin, 1982), thus simplifies the computation of the centre of gravity. Another implementation of this idea implies approximation of a *volumetric centre* of the feasible region (Vaidya, 1996) and the resulting algorithm has the same information-based complexity, but its single step involves explicit matrix inversion which makes it prohibitive for high dimensions. It requires by a factor $\mathcal{O}(n)$ more computations

than a step of the ellipsoid algorithm (Algorithm 2) which is a computationally tractable version of the approach of central cuts. The ellipsoid algorithm has a factor n higher information-based complexity, but its single iteration is cheaper than that of the above mentioned techniques. The complexity $\mathcal{O}(1)n\ln\left(2+\frac{1}{\rho}\right)$ is also an upper bound on the learning complexity of the online learning problem (2.2) although for some values of the margin ρ it is not tight.

2. $\frac{1}{n} < \rho < \frac{1}{\sqrt{n}}$. In this case as the margin grows the complexity bound $\frac{\mathcal{O}(1)}{\rho^2}$ (the perceptron algorithm) gradually becomes lower than $\mathcal{O}(1)n\ln\left(2+\frac{1}{\rho}\right)$. Indeed, the estimate $\mathcal{O}(1)n$ approaches $\frac{\mathcal{O}(1)}{\rho^2}$ from above when ρ goes to $\frac{1}{\sqrt{n}}$.

3. $\rho \geq \frac{1}{\sqrt{n}}$. In this case the learning complexity does not depend on the dimension anymore and the plain perceptron algorithm (Algorithm 1) attains the optimal learning complexity of the family of online learning problems $\mathfrak{P}$.[4]

Alternatively, we can consider the learning complexity of online learning for a fixed margin ρ when the problem dimension goes to infinity. In this case one can see that when n becomes larger than $\frac{1}{\rho^2}$ the perceptron algorithm cannot be outperformed.

[4]Note that the learning complexity implies that the algorithm needs at most $\frac{\mathcal{O}(1)}{\rho^2}$ counterexamples to generate a solution while in the case of the perceptron algorithm it may happen that several updates on the same counterexamples are needed for some sequences of examples. However from the Block-Novikoff theorem (Theorem 2.7) we know that the number of updates does not exceed $\frac{1}{\rho^2}$ thus we conclude that the perceptron algorithm becomes optimal in this case.

Chapter 6

Learning Approach to Semidefinite Programming

6.1 Objectives

As a real-world instance of the model of online learning with an infinite dataset we consider a semidefinite constraint satisfaction problem. In this chapter we show that a generic semidefinite feasibility program is equivalent to a linear program with infinitely many constraints, derive a separation oracle based on incomplete Cholesky decomposition of a symmetric matrix, adapt the ellipsoid and probabilistic perceptron algorithms to solve this problem, and present some numerical comparisons of these techniques against one another and discuss their relation to the state-of-the-art interior point algorithms.

6.2 Semidefinite Constraint Satisfaction Problem

The basic notions of semidefinite programming are collected in Sections A.3-A.5 of Appendix, so in this chapter we just informally introduce the key terms while referring the reader to that part of the thesis for further particulars.

Semidefinite programming optimizes a linear function over the intersection of the cone of $n \times n$ positive semidefinite matrices $\mathbb{S}^n_+$ and an affine set

$$F \quad := \quad \{\mathbf{F}(\mathbf{x}) := \mathbf{F}_0 + \textstyle\sum_{i=1}^m x_i \mathbf{F}_i \,|\, \mathbf{F}_i \in \mathbb{S}^n,\, i \in \{0, 1, \ldots, m\},\, \mathbf{x} \in \mathbb{R}^m\}$$

To see the affinity of the set F one needs to represent any its member in the form

$$\mathbf{G}(\mathbf{y}) \quad = \quad \textstyle\sum_{j=0}^m y_j \mathbf{G}_j, \quad \sum_{j=0}^m y_j \quad = \quad 1$$

with $\mathbf{y} \in \mathbb{R}^{m+1}$ and $\mathbf{G}_j \in \mathbb{S}^n, j \in \{0, 1, \ldots, m\}$ which can be obtained by mapping

$$\begin{array}{lll} \forall i \in \{1, \ldots, m\} & \mathbf{G}_i := \mathbf{F}_0 - \mathbf{F}_i, & y_i := -x_i \\ & \mathbf{G}_0 := \mathbf{F}_0 & y_0 := 1 + \sum_{j=1}^m x_i. \end{array}$$

The first representation is usually preferred in the literature since it does not impose any constraint on the design vector $\mathbf{x}$, so like in Definition A.23 from now on we shall consider a semidefinite program (SDP) in the form

$$\begin{array}{ll} \min_{\mathbf{x} \in \mathbb{R}^m} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & \mathbf{F}(\mathbf{x}) \in F, \quad \mathbf{F}(\mathbf{x}) \in \mathbb{S}^n_+, \end{array} \tag{6.1}$$

where $\mathbf{c} \in \mathbb{R}^m$ is a given cost vector and F and $\mathbf{F}(\mathbf{x})$ are defined as above.

In perceptron learning we aim for finding a feasible (not necessarily optimal) solution, so in this chapter we shall mainly focus on a feasibility subproblem of SDP (SDFP) also known as a *semidefinite constraint satisfaction problems* (SCSP)

$$\begin{array}{ll} \text{find} & \mathbf{x} \in \mathbb{R}^m \\ \text{subject to} & \mathbf{F}_0 + \sum_{i=1}^m \mathbf{F}_i x_i \succeq \mathbf{0}\,. \end{array} \tag{6.2}$$

In order to apply the perceptron learning algorithms to semidefinte programming we need to convert an SDP to the online learning model of linear pattern separation (Section 2.1), and for this purpose we have to figure out two main technicalities:

1. Indicate a dataset A for the online learning problem, which implies equivalence of a positive semidefinite constraint to some set of linear constraints.
2. Construct a separation oracle for a positive semidefinite constraint.

These problems will be tackled in two subsequent sections.

6.3 Linear Programming View of an SDP

The following proposition shows that a semidefinite program is a direct generalization of a linear program (Definition A.10).

Proposition 6.1. *Every semidefinite program is a linear program with infinitely many linear constraints (semi-infinite program).*

Proof. Obviously, the objective function in (6.1) is linear in $\mathbf{x}$. For any $\mathbf{u} \in \mathbb{R}^n$, define the vector $\mathbf{a}_\mathbf{u} := \big(\mathbf{u}^\top \mathbf{F}_1 \mathbf{u}, \ldots, \mathbf{u}^\top \mathbf{F}_m \mathbf{u}\big)$. Then, the constraints in (6.1) can be written

as

$$\forall \mathbf{u} \in \mathbb{R}^n : \quad \mathbf{u}^\top \mathbf{F}(\mathbf{x})\, \mathbf{u} \geq 0 \qquad \Leftrightarrow \qquad \forall \mathbf{u} \in \mathbb{R}^n : \quad \mathbf{x}^\top \mathbf{a}_\mathbf{u} \geq -\mathbf{u}^\top \mathbf{F}_0 \mathbf{u}\,. \tag{6.3}$$

This is a linear constraint in $\mathbf{x}$ for all $\mathbf{u} \in \mathbb{R}^n$ (of which there are infinitely many). □

In the view of this preposition, the dataset A can be defined by virtue of

$$A(\mathbf{F}_0, \ldots, \mathbf{F}_m) := \left\{ \mathbf{a}_\mathbf{u} := \left(\mathbf{u}^\top \mathbf{F}_0 \mathbf{u}, \ldots, \mathbf{u}^\top \mathbf{F}_m \mathbf{u} \right) \mid \mathbf{u} \in \mathbb{R}^n \right\}.$$

Without losing generality and for convenience of our exposition we can assume that the vectors $\mathbf{u}$ are normalized, but in order to avoid too much complication we do not normalize $\mathbf{a}_\mathbf{u}$ and the reason for this will be provided further on. So far we have resolved the first of two technical issues.

Since the objective function is linear in $\mathbf{x}$, we may solve an SDP itself by a sequence of semidefinite feasibility problems (6.2) introducing the additional constraint $\mathbf{c}^\top \mathbf{x} \leq c_0$ and iteratively decreasing $c_0 \in \mathbb{R}$ by replacing it with the value of the objective at every newly found feasible point. Although this algorithm is impractical in a general case, we describe it as extension of our approach and state the following proposition.

Proposition 6.2. *Any SDP can be solved by a sequence of homogenized SCSPs of the following form*

$$\text{find} \quad \mathbf{x} \in \mathbb{R}^{m+1} \quad \text{subject to} \quad \mathbf{G}(\mathbf{x}) := \sum_{i=0}^{m} x_i \mathbf{G}_i \succ \mathbf{0}$$

where $\mathbf{G}_i \in \mathbb{S}^{n+2}$, $i \in \{0, 1, \ldots, m\}$.

Proof. In order to make $\mathbf{F}_0$ and c_0 dependent on the optimization variables, we introduce an auxiliary variable $x_0 > 0$; the solution to the original problem is given by $x_0^{-1} \cdot \mathbf{x}$. Moreover, we can repose two linear constraints $c_0 x_0 - \mathbf{c}^\top \mathbf{x} \geq 0$ and $x_0 > 0$ as an LMI using the fact that a block-diagonal matrix is positive (semi)definite if and only if every block is positive (semi)definite. Thus, the following matrices are sufficient:

$$\mathbf{G}_0 = \begin{pmatrix} \mathbf{F}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^\top & c_0 & 0 \\ \mathbf{0}^\top & 0 & 1 \end{pmatrix}, \qquad \mathbf{G}_i = \begin{pmatrix} \mathbf{F}_i & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -c_i & 0 \\ \mathbf{0} & 0 & 0 \end{pmatrix}.$$

Given an upper and a lower bound on the objective function, repeated bisection can be used to determine the solution in $\mathcal{O}\left(\log \frac{1}{\varepsilon}\right)$ steps to accuracy ε provided the employed algorithm is able to detect infeasibility. □

6.4 Cholesky Separation Oracle

Having constructed the dataset A, finding a vector $\mathbf{a_u} \in A$ such that $\mathbf{x}^\top \mathbf{a_u} \leq 0$ is equivalent to identifying a vector $\mathbf{u} \in \mathbb{R}^n$ such that

$$\sum_{i=0}^{m} x_i \mathbf{u}^\top \mathbf{F}_i \mathbf{u} \;=\; \mathbf{u}^\top \mathbf{F}(\mathbf{x})\, \mathbf{u} \;\leq\; 0$$

that is after t perceptron updates we need to check if $\mathbf{F}(\mathbf{x}_t) \succeq \mathbf{0}$ and generate a counterexample otherwise. For this purpose consider two of the possible criteria of positive semidefiniteness of a symmetric matrix stated in the next theorem.

Theorem 6.3. *Let* $\mathbf{A} \in \mathbb{S}^n$. *Then* $\mathbf{A} \succ \mathbf{0}$ *if and only if any of the following holds*

- *all eigenvalues of* $\mathbf{A}$ *are positive,*
- *Cholesky factorization* $\mathbf{A} = \mathbf{L}^\top \mathbf{L}$ *exists with* $l_{ii} > 0$, $i \in \{1, \ldots, n\}$.

Proof. Basic properties of a positive semidefinite matrix (e.g. see Horn and Johnson (1991)). □

Therefore, one possible way of finding a counterexample $\mathbf{u}$ (and consequently $\mathbf{a_u}$) for the current solution $\mathbf{x}_t$ is to compute the eigenvector corresponding to the smallest eigenvalue of $\mathbf{F}(\mathbf{x}_t)$; if this eigenvalue is positive, the algorithm stops and outputs $\mathbf{x}_t$. However since we do not necessarily need the eigenvector which corresponds to the minimal eigenvalue of the matrix, but could utilize any counterexample, we choose a computationally easier procedure to find a suitable $\mathbf{u} \in \mathbb{R}^n$ based on the incomplete Cholesky decomposition.

The Cholesky factorization $\mathbf{A} = \mathbf{R}^\top \mathbf{R}$ (where $\mathbf{R}$ is an upper triangular matrix), which is a dual implementation of the Gram-Schmidt orthogonalization, is an oracle for the positive definiteness of the square symmetric matrix $\mathbf{A} \in \mathbb{S}^n$ because $\mathbf{A}$ is positive definite if and only if all diagonal entries of its Cholesky factor $\mathbf{R}$ are positive as stated in Theorem 6.3.

We apply Cholesky factorization to check if a squared symmetric matrix $\mathbf{A} \in \mathbb{S}^n$ is positive semidefinite, and if it is not, then to find a vector $\mathbf{v} \in \mathbb{R}^n$ such that $\mathbf{v}^\top \mathbf{A} \mathbf{v} < 0$. The presented procedure is a modified incomplete Cholesky factorization of a kernel matrix (Shawe-Taylor and Cristianini, 2004, Section 5.2) where at every stage we check if there is a row of $\mathbf{A}$ with negative norm residual, and if not, then we compute the next row of $\mathbf{R}$ using the one which has the largest norm residual.

When at the j-th iteration of the Cholesky decomposition we find the negative norm residual ($d_{i_{min}} < 0$), we apply the permutation of matrices and $\mathbf{A}$ and $\mathbf{R}$ to move it to the the j-th position on the diagonal of $\mathbf{R}$). Now we look for the negative example in the form $\mathbf{v} = (v_1, \ldots, v_{j-1}, 1, 0, \ldots, 0)^\top$ for which we have

$$\mathbf{v}^\top \mathbf{R}^\top \mathbf{R}\mathbf{v} = (\sum_{i=1}^{j} v_i r_{1i})^2 + (\sum_{i=2}^{j} v_i r_{2i})^2 + \ldots + (v_{j-1} r_{j-1j-1} + r_{j-1j})^2 + \underbrace{r_{jj}^2}_{<0}$$

Observe that we can make all except the last squared terms equal to zero using suitable values for $v_1, \ldots, v_{j-1}$ which can be easily found via back-substitution starting from v_{j-1}. The existence and uniqueness of the solution is guaranteed by the positivity of the values $r_{11}, \ldots, r_{j-1j-1}$. Therefore, the product above becomes just equal to $r_{jj}^2 < 0$. The full procedure is given in Algorithm 7.

Theorem 6.4. *Matrix* $\mathbf{A} \in \mathbb{S}_+^n$ *if and only if Algorithm 7 outputs* positive semidefinite *when* $\mathbf{A}$ *is passed as an input.*

Proof. By construction as described above in this section. □

6.5 Limitations of the Positive Semidefinite Oracle

If we combine Propositions 6.1 and the convergence theorem of the perceptron learning algorithm (Theorem 2.7) together with Algorithm 7, we obtain a perceptron algorithm to solve semidefinite feasibility programs (and using Proposition 6.2 we can get the algorithm that solves SDPs sequentially approximating the minimum from above).

In fact, any of the perceptron learning algorithms discussed so far which deliver a feasible solution in a finite number of steps can be applied to solve (6.2). Like in the previous chapters, we mainly focus on the algorithms which converge in polynomial time, and in this family of algorithms those are the ellipsoid algorithm (Algorithm 2) and the probabilistic perceptron in its parametric formulation (Algorithm 6).

Now the application of both these algorithms to CSDP is straightforward, but before proceeding to experiments we need to mention an issue concerned with normalization of A.

Consider the following CSDP already in a homogeneous form[1]

$$\begin{array}{ll} \text{find} & \mathbf{x} \in \mathbb{R}^3 \\ \text{subject to} & x_1 \begin{bmatrix} -\lambda_1 & 0 \\ 0 & -\lambda_2 \end{bmatrix} + x_2 \begin{bmatrix} 0 & 0 \\ 0 & \mu \end{bmatrix} + x_3 \begin{bmatrix} \delta & 0 \\ 0 & 0 \end{bmatrix} \succeq \mathbf{0} \end{array} \tag{6.4}$$

where $0 < \lambda_1 < \lambda_2$, $\lambda_2 \ll \mu$ (at least $\mu > \lambda_2 \sqrt{\frac{1-\sigma^2}{\sigma^2}}$), $\delta \ll \lambda_1$ (at least $\delta < \lambda_1 \sqrt{\frac{1-\sigma^2}{\sigma^2}}$) and $0 < \sigma < 1$. This problem is strictly feasible, and one of its solutions is $(0, 1, 1)^\top$.

[1]This example was suggested by Amiran Ambroladze.

Algorithm 7: Incomplete Cholesky Decomposition for Generating Counterexamples

Input: symmetric matrix $\mathbf{A} \in \mathbb{S}^n$
Output: certify that $\mathbf{A} \succeq 0$, otherwise find $\mathbf{v} \in \mathbb{R}^n : \mathbf{v}^\top \mathbf{A}\mathbf{v} < 0$

$\mathbf{R} \leftarrow \mathbf{0}, \quad \mathbf{P} \leftarrow \mathbf{I}_n, \quad \mathbf{d} \leftarrow \texttt{diag(A)}, \quad j \leftarrow 1;$
$a_{min} \leftarrow \texttt{min}(d_i), \quad a_{max} \leftarrow \texttt{max}(d_i);$

while $a_{min} \geq 0$ **and** $a_{max} > 0$ **do**
  $i_{max} \leftarrow \texttt{argmax}(d_i);$
  if $i_{max} \neq j$ **then**
    swap i_{max}-th and j-th rows and columns of matrix $\mathbf{P}$;
    $\mathbf{A} \leftarrow \mathbf{PAP}, \quad \mathbf{R} \leftarrow \mathbf{PRP}, \quad \mathbf{d} \leftarrow \mathbf{Pd};$
  end
  $r_{jj} \leftarrow \sqrt{a_{max}};$
  $d_j \leftarrow 0;$
  for $i = j + 1$ **to** n **do**
    $r_{ji} \leftarrow (a_{ji} - \sum_{k=1}^{j-1} r_{kj} r_{ki})/r_{jj};$
    $d_i \leftarrow d_i - r_{ji}^2;$
  end
  $a_{min} \leftarrow \texttt{min}(d_i), \quad a_{max} \leftarrow \texttt{max}(d_i);$
  $j \leftarrow j + 1;$
end

if $a_{min} < 0$ **then**
  $i_{min} \leftarrow \texttt{argmin}(d_i);$
  if $i_{min} \neq j$ **then**
    swap i_{min}-th and j-th rows and columns of matrix $\mathbf{P}$;
    $\mathbf{A} \leftarrow \mathbf{PAP}, \quad \mathbf{R} \leftarrow \mathbf{PRP}, \quad \mathbf{d} \leftarrow \mathbf{Pd};$
  end
  $\mathbf{v} \leftarrow \mathbf{0}, \quad v_j \leftarrow 1;$
  for $i = j - 1$ **down to** 1 **do**
    $v_i \leftarrow -\sum_{k=i+1}^{j} r_{ik} v_k / r_{ii};$
  end
  $\mathbf{v} \leftarrow \mathbf{Pv};$
  return $\mathbf{v}$;
else
  return *positive semidefinite*;
end

Let us assume that we use the modified perceptron algorithm (Algorithm 3), e.g. as a part of the probabilistic perceptron rescaling algorithm, to find an almost feasible vector $\mathbf{x}$ such that for all $\mathbf{a} \in A_{\mathbf{F}}$

$$\frac{\mathbf{x}^\top \mathbf{a}}{\|\mathbf{x}\|\|\mathbf{a}\|} \quad \geq \quad -\sigma. \tag{6.5}$$

We also assume that at some point after t steps of the algorithm (or at the start and $t = 0$) we have $\mathbf{x}_t = (1, 0, 0)^\top$. In this case

$$\mathbf{F}(\mathbf{x}_t) \quad = \quad \begin{bmatrix} -\lambda_1 & 0 \\ 0 & -\lambda_2 \end{bmatrix} \quad \prec \quad \mathbf{0}.$$

$\mathbf{F}(\mathbf{x}_t)$ has two eigenvectors $\mathbf{u}_1 = (1,0)^\top$ and $\mathbf{u}_2 = (0,1)^\top$ that induce two counterexamples $\mathbf{a}_{\mathbf{u}_1} = (-\lambda_1, 0, \delta)^\top$ and $\mathbf{a}_{\mathbf{u}_2} = (-\lambda_2, \mu, 0)^\top$. Notice that

$$\frac{\mathbf{x}_t \mathbf{a}_{\mathbf{u}_1}}{\|\mathbf{x}_t\| \|\mathbf{a}_{\mathbf{u}_1}\|} = \frac{-\lambda_1}{\sqrt{\lambda_1^2 + \delta^2}} < -\sigma$$

and

$$\frac{\mathbf{x}_t \mathbf{a}_{\mathbf{u}_2}}{\|\mathbf{x}_t\| \|\mathbf{a}_{\mathbf{u}_2}\|} = \frac{-\lambda_2}{\sqrt{\lambda_2^2 + \mu^2}} > -\sigma$$

while

$$-\lambda_2 = \mathbf{u}_2^\top \mathbf{F}(\mathbf{x}_t)\mathbf{u}_2 < \mathbf{u}_1^\top \mathbf{F}(\mathbf{x}_t)\mathbf{u}_1 = -\lambda_1.$$

Therefore, the output of the oracle in the general case does not allow us to check the condition (6.5) for non-zero σ.

6.6 Unnormalized Parametric Modified Perceptron Algorithm

Algorithm 8: Unnormalized Parametric Modified Perceptron Algorithm

Input: a (possibly) infinite set A of vectors $\mathbf{a} \in \mathbb{R}^n$ and parameters σ and $\theta \in \mathbb{R}_{++}$.
Output: a unit vector $\mathbf{x} \in \mathbb{R}^n$ such that $\forall \mathbf{a} \in A : \mathbf{x}^\top \mathbf{a} \geq -\sigma$.

repeat
  set $\mathbf{x}$ uniformly at random in $\{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y}\| = 1\}$;
  set $t \leftarrow 0$;
  set $s \leftarrow 0$;
  while *exists* $\mathbf{a} \in A$ *such that* $\overline{\mathbf{x}}^\top \mathbf{a} < -\sigma$ **do**
    $t \leftarrow t + 1$;
    $\mathbf{x} \leftarrow (\mathbf{I} - \overline{\mathbf{a}}\overline{\mathbf{a}}^\top)\mathbf{x}$;
    $s \leftarrow s + \frac{1}{\|\mathbf{a}\|^2}$;
*   **if** $\|\mathbf{x}\| = 0$ **then**
      break
    end
    if $s > \frac{2}{\sigma^2} \ln \frac{1}{\theta}$ **then**
      break
    end
  end
until $\forall \mathbf{a} \in A : \overline{\mathbf{x}}^\top \mathbf{a} \geq -\sigma$;
return $\overline{\mathbf{x}}$

In order to avoid the problem described in the previous section, we amend the condition (6.5) in the modified perceptron algorithm (Algorithm 3) on making an update if there exists a unit vector $\mathbf{u}$ such that

$$\mathbf{u}^\top (\mathbf{F}(\overline{\mathbf{x}}) + \sigma \mathbf{I})\mathbf{u} < 0$$

where $\mathbf{I}$ is the identity matrix in $\mathbb{R}^{n\times n}$ and which is equivalent to

$$\mathbf{u}^\top \mathbf{F}(\overline{\mathbf{x}})\mathbf{u} \;=\; (\mathbf{u}^\top \mathbf{F}_1\mathbf{u},\ldots,\mathbf{u}^\top \mathbf{F}_m\mathbf{u},\mathbf{u}^\top \mathbf{F}_0\mathbf{u})\overline{\mathbf{x}} \;=\; \mathbf{a}_{\mathbf{u}}^\top\overline{\mathbf{x}} \;<\; -\sigma\mathbf{u}^\top\mathbf{I}\mathbf{u} \;=\; -\sigma.$$

Note that in this case we drop normalization of $\mathbf{a}_u$ at all.

We state the resulting algorithm in its parametric version (following the argument of Chapter 4) as Algorithm 8, and prove its convergence in the following Theorem.[2]

Theorem 6.5. *For a given set of vectors $A \subset \mathbb{R}^n$, linearly separable from the origin, let $\mathbf{y} \in \mathbb{R}^n$ be any unit vector such that for all $\mathbf{a} \in A:\ \mathbf{a}^\top\mathbf{y} \geq 0$. Then with probability at least η_θ, after at most t updates such that*

$$\textstyle\sum_{i=1}^{t} \frac{1}{\|\mathbf{a}_i\|^2} \;\geq\; \frac{2}{\sigma^2}\ln\frac{1}{\theta} \tag{6.6}$$

where $\mathbf{a}_i$ is the vector that i'th update was made on and $i \in \{1,\ldots,t\}$, the parametric modified perceptron algorithm (Algorithm 8) returns a unit vector $\mathbf{x} \in \mathbb{R}^n$ such that

1. $\forall \mathbf{a} \in A:\ \mathbf{a}^\top\mathbf{x} \geq -\sigma$,
2. $\mathbf{y}^\top\mathbf{x} \geq \theta$.

Proof. With probability at least η_θ, the random unit vector $\mathbf{x}$ at the start of the algorithm satisfies $\mathbf{y}^\top\mathbf{x} \geq \theta$ for some unit separating weight vector $\mathbf{y}$ for A. The inner product $\mathbf{y}^\top\mathbf{x}$ cannot decrease since

$$(\mathbf{x} - (\overline{\mathbf{a}}^\top\mathbf{x})\overline{\mathbf{a}})^\top\mathbf{y} \;=\; \mathbf{x}^\top\mathbf{y} - (\mathbf{x}^\top\overline{\mathbf{a}})(\overline{\mathbf{a}}^\top\mathbf{y}) \;\geq\; \mathbf{x}^\top\mathbf{y}$$

because $\mathbf{x}^\top\mathbf{a}$ has to be negative in order for $\mathbf{a}$ to be an update vector and $\mathbf{a}^\top\mathbf{y}$ is positive by the definition of the separating vector.

On the other hand, the norm of the weight vector $\mathbf{x}$ decreases with every update according to

$$\begin{aligned}(\mathbf{x} - (\overline{\mathbf{a}}^\top\mathbf{x})\overline{\mathbf{a}})^\top(\mathbf{x} - (\overline{\mathbf{a}}^\top\mathbf{x})\overline{\mathbf{a}}) \;&=\; \mathbf{x}^\top\mathbf{x} - 2(\overline{\mathbf{a}}^\top\mathbf{x})^2 + (\overline{\mathbf{a}}^\top\mathbf{x})^2 \;=\; \mathbf{x}^\top\mathbf{x} - (\overline{\mathbf{a}}^\top\mathbf{x})^2\\ &=\; \mathbf{x}^\top\mathbf{x} - \tfrac{(\mathbf{a}^\top\mathbf{x})^2}{\|\mathbf{a}\|^2} \;\leq\; \mathbf{x}^\top\mathbf{x}(1-\sigma^2),\end{aligned}$$

since we make an update on $\mathbf{a}$ only if $\mathbf{a}^\top\overline{\mathbf{x}} < -\sigma$.

After t iterations the norm of the weight vector $\mathbf{x}_t$ falls to at most $\sqrt{\prod_{i=1}^{t}\left(1-\frac{\sigma^2}{\|\mathbf{a}_i\|^2}\right)}$ where $\mathbf{a}_i$ is the counterexample we update on during the i'th iteration of the algorithm and $i \in \{1,\ldots,t\}$. If (6.6) holds, then after taking the natural logarithm of the previous

[2]This algorithm was suggested by Amiran Ambroladze as a remedy against the problem described in the previous section.

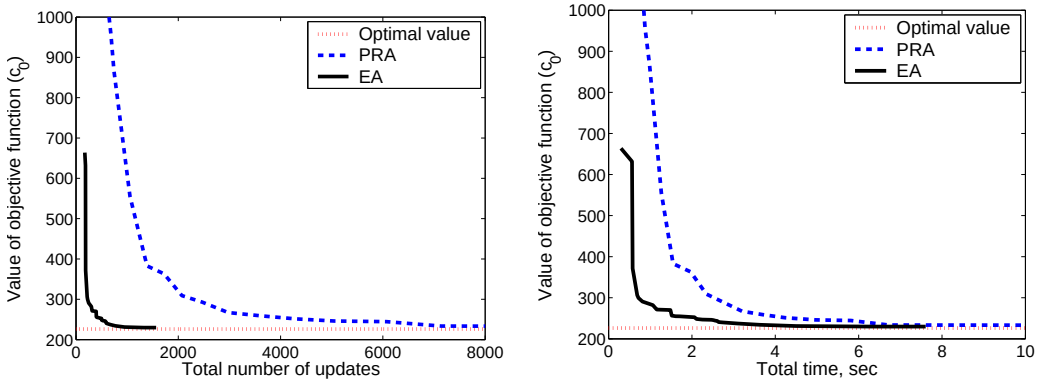


FIGURE 6.1: The plots show the decay of the attained objective function value of 'mcp100' problem against the number of updates (**Left**) and time (**Right**) needed by the ellipsoid and perceptron rescaling algorithms to approach the optimum.

estimate of $\|\mathbf{x}_t\|$, we get

$$\ln \|\mathbf{x}_t\| \leq \frac{1}{2} \ln \prod_{i=1}^{t} \left(1 - \frac{\sigma^2}{\|\mathbf{a}_i\|^2}\right) < \frac{1}{2} \sum_{i=1}^{t} \frac{-\sigma^2}{\|\mathbf{a}_i\|^2} < -\frac{2\sigma^2}{2\sigma^2} \ln \frac{1}{\theta} = \ln \theta$$

where the second inequality follows from $1 + \alpha \leq e^{\alpha}$ that holds for all $\alpha \in \mathbb{R}$. So after that number of steps the norm of the weight vector $\mathbf{x}$ reduces to less than $\frac{1}{\theta}$. If the starting vector satisfies $\mathbf{y}^\top \mathbf{x} \geq \theta$, then after t updates of the algorithm we have

$$\mathbf{y}^\top \overline{\mathbf{x}}_t = \frac{\mathbf{y}^\top \mathbf{x}_t}{\|\mathbf{x}_t\|} > \frac{\theta}{\theta} = 1$$

which is impossible. Therefore, if the starting vector satisfies this condition (which happens with probability at least η_θ), then the modified perceptron algorithm converges after a finite number of updates and its output also satisfies this condition. □

6.7 Experimental Results

In this section our task is to compare two polynomial-time algorithms based on the subgradient descent method:+ + the ellipsoid algorithm (Algorithm 2) and the probabilistic parametric perceptron rescaling algorithm (Algorithm 6). From theoretical estimates of their complexities (Theorem 2.9 and Theorem 4.1) the former has lower learning and computational complexities (by a factor of $\mathcal{O}(n \ln n)$) in the worst case than the latter. However, in Chapter 5 we have conjectured that when the dimension of the problem n grows and the margin ρ is fixed, then the optimal complexity for this class of problems approaches $\frac{1}{\rho^2}$, so the advantage of the lower learning complexity of the ellipsoid algorithm vanishes.

We considered benchmark problems arising from semidefinite relaxations to the MAX-CUT problems of weighted graphs, which is posed as finding a maximum weight bisection

of a graph. The benchmark MAXCUT problems have the following relaxed SDP form (see Helmberg (2000)):

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad \mathbf{1}^\top\mathbf{x} \quad \text{subject to} \quad \underbrace{-\frac{1}{4}\left(\operatorname{diag}(\mathbf{C}\mathbf{1})-\mathbf{C}\right)}_{\mathbf{F}_0} + \underbrace{\operatorname{diag}(\mathbf{x})}_{\sum_i x_i\mathbf{F}_i} \succeq \mathbf{0}\,, \tag{6.7}$$

where $\mathbf{C}\in\mathbb{R}^{n\times n}$ is the adjacency matrix of the graph with n vertices. One property of the semidefinite relaxation for the MAXCUT problem is that dimension of the square constraint matrices n equals to the dimension of the design vector m. This problems also has nice special structure that allows us to evaluate the constraint matrix $\mathbf{F}(\mathbf{x})$ very fast: all its component matrices except $\mathbf{F}_0$ have only one non-zero entry, located on the diagonal.

The experiments reported in this section fall into two parts. Our initial aim was to demonstrate that the perceptron rescaling algorithm works in practice on infinite datasets of infinite size and to assess its efficacy on a benchmark example from graph bisection provided by SDPLIB 1.2 Borchers (1999). We chose to compare the algorithm against the ellipsoid method (whose complexity is only by factor $\mathcal{O}(n)$ higher than the optimal algorithm for this class of problems, as we conjectured in the previous chapter) in order to see how much our probabilistic algorithm looses in practice in terms of the number of mistakes (learning complexity), and whether its simplicity and lower cost of a single update can compensate this loss in terms of the total computational effort used.

The ellipsoid algorithm was implemented as in Algorithm 2. As for the parametric perceptron rescaling algorithm, we set all values of its parameters to those used in Algorithm 4, except the estimate of the margin ϱ, which was increased to $\frac{1}{16\sqrt{n}}$. The convergence of the algorithm with this set of the parameters was shown in Section 4.6. Both algorithms were implemented in MatLab (with the random generator set to 0 at the start of each run of the probabilistic perceptron), only the code of the Cholesky separation oracle was in C++. All the experiments described in this section were run on a 2.6GHz machine.

First we ran both algorithm on 'mcp100' (the smallest MAXCUT relaxation problem in the collection, $m, n = 100$), which we re-stated as a sequence of the semidefinite constraint satisfaction programs (as shown in Proposition 6.2). After solving the first SCSP (the problem itself without the objective) and finding a feasible point we introduced a new constraint which kept the objective function to below its value at this point, and repeated this process iteratively the relative distance between two consecutive points was less than 0.05.

For this test we have updated our implementation in two ways. Firstly, we used the perceptron algorithm with margins (also known as τ-perceptron; see Li et al. (2002)) instead of the plain perceptron algorithm. This was done to force the algorithm find

Problem		PRA		EA	
name	m, n	updates	time, sec	updates	time, sec
mcp100	100	313	**0.4**	**173**	0.5
mcp124-1	124	209	**0.4**	**167**	0.9
mcp124-2	124	381	**0.5**	**212**	1.1
mcp124-3	124	748	**0.9**	**255**	1.3
mcp124-4	124	1305	**1.5**	**294**	**1.5**
mcp250-1	250	429	**4.2**	**350**	11.6
mcp250-2	250	696	**4.8**	**420**	14.1
mcp250-3	250	1340	**8.1**	**513**	16.6
mcp250-4	250	2504	**14.1**	**598**	19.8
mcp500-1	500	844	**38.7**	**683**	147.1
mcp500-2	500	1518	**39.7**	**856**	177.6
mcp500-3	500	2529	**64.2**	**1005**	192.5
mcp500-4	500	5166	**119.3**	**1195**	223.1
maxG11	800	721	**165.7**	**700**	570.4

TABLE 6.1: Comparison of the perceptron rescaling algorithm against the ellipsoid algorithm on 14 semidefinite relaxations of MAXCUT problems from SDPLIB 1.2 collection. For each problems, a constraint satisfaction problems was solved (to find a feasible point), and the objective function was ignored. The table contains brief problem description and time and the number of updates each algorithm needed to converge (the best values for each problem are highlighted in bold font).

points deeper in the interior of the feasible region (hence, to approach the optimum faster). Therefore, the maximal number of allowed mistakes at the perceptron phase was changed to the bound from Li et al. (2002).

The decay of the objective with time is plotted in the Figure 6.1 (left). The right plot shows how many updates both algorithms needed to reach every approximation to the optimum. In this experiment the ellipsoid algorithm completely outperformed the probabilistic perceptron. Furthermore, we do realize that both algorithms cannot compete in solving an SDP with the state-of-the-art interior-point methods exploits the structure of the positive semidefinite cone, so is not based on the same 'black box' concept as these perceptron learning techniques.

The second set experiment implied just solving feasibility problems, so the objective function was completely ignored. These results are shown in Table 6.1, and now we see that although the perceptron rescaling algorithm still makes more updates to converge, it needs less computational effort for this task, than the ellipsoid algorithm. This makes the probabilistic perceptron potentially interesting for high-dimensional learning problems due to its simplicity and low cost of making an update.

Chapter 7

Pattern Separation via Ellipsoids

7.1 Basics on Ellipsoids

In chapter 2 we made a short overview of hyperplane pattern separation. Now we consider a pattern separation algorithm that uses ellipsoids for binary separation (suggested in (Glineur, 1998)), and in the next chapter we discuss its application to a document categorization problem.

First we give a formal definition of an ellipsoid and show its alternative representation, then describe the main ideas of the maximal separation ratio algorithm, a geometrical interpretation, proceed to the SDP formulation of the algorithm, and conclude with a discussion of its properties.

Definition 7.1 (Ellipsoid)**.** A *(full-dimensional) ellipsoid* $E \subset \mathbb{R}^n$ is a set of points given by its centre $\mathbf{c}$ and an $n \times n$ positive definite matrix $\mathbf{A}$ such that

$$E := \{\mathbf{A}\mathbf{u} + \mathbf{c} \,|\, \|\mathbf{u}\| \leq 1,\, \mathbf{u} \in \mathbb{R}^n,\, \mathbf{A} \in \mathbb{S}^n_{++}\}. \tag{7.1}$$

The eigenvectors and eigenvalues of $\mathbf{A}$ define the directions and half-lengths of the axes of the ellipsoid E (figure 7.1) respectively. If we substitute matrix $\mathbf{A}$ with an identity matrix of the same size then we obtain a unit ball centred at the point $\mathbf{c}$. Therefore a generic ellipsoid is simply an image of some ball under the affine mapping given by the positive definite matrix $\mathbf{A}$. If we relax the positive definite constraint on matrix $\mathbf{A}$ and allow it to be either positive semidefinite, or indefinite symmetric, or even a rectangular matrix then the convex set E is a *degenerate zero volume* ellipsoid because it has zero volume in some dimensions (which correspond to the basis of the kernel of $\mathbf{A}$). Any point, or any line segment in $\mathbb{R}^n$ is an example of a degenerate ellipsoid. From now on we will deal only with cases when $\mathbf{A}$ is either positive semidefinite or positive definite.

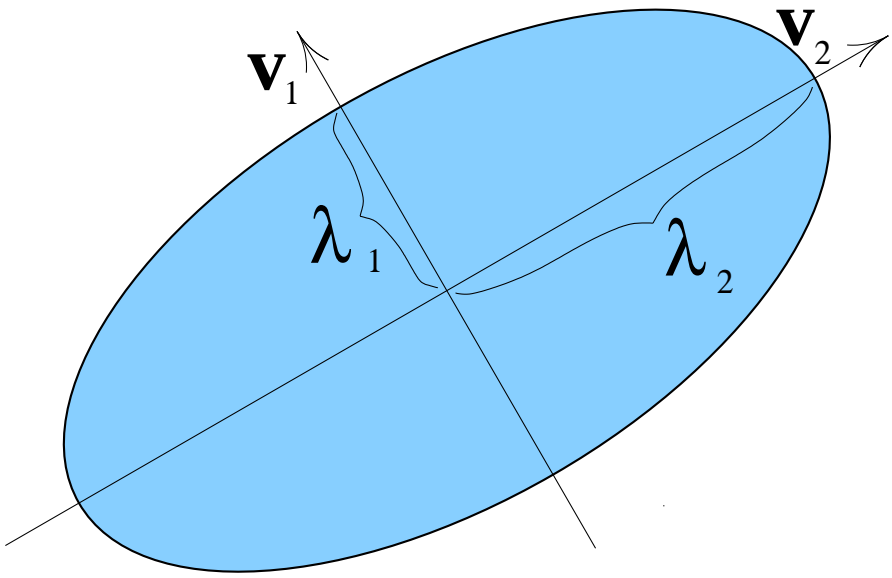


FIGURE 7.1: Example of a 2-dimensional ellipsoid. Eigenvalues λ_1 and λ_2 and eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$ of the matrix $\mathbf{A}$ constitute the lengths and directions of the axes of the ellipsoid respectively.

Alternatively an ellipsoid can be represented as

$$E := \{\mathbf{x} \in \mathbb{R}^n \,|\, (\mathbf{x}-\mathbf{c})^\top \mathbf{E}(\mathbf{x}-\mathbf{c}) \leq 1,\, \mathbf{E} \succ \mathbf{0}\}, \tag{7.2}$$

and by setting $\mathbf{E} = \mathbf{A}^{-2}$ one can see the equivalence of the definitions. Note that if the matrix $\mathbf{E}$ is positive semidefinite, i. e. has some eigenvalues equal to zero then corresponding eigenvalues of matrix $\mathbf{A}$ and the corresponding axes of ellipsoid E are infinitely large. We call such ellipsoids *degenerate infinite volume* ellipsoids. For instance, for $\mathbf{E} \in \mathbb{R}^3$ if $\mathbf{E}$ has one zero eigenvalue then E is an elliptic cylinder of infinite height, if two eigenvalues of $\mathbf{E}$ are equal to zero then E consists of all points that belong to and lie between two parallel planes.

In addition equation (7.2) also allows us to introduce a new metric in $\mathbb{R}^n$. Observe that if we substitute matrix $\mathbf{E}$ with an identity matrix in $\mathbb{R}^n$ in (7.2), then $(\mathbf{x}-\mathbf{c})^\top \mathbf{E}(\mathbf{x}-\mathbf{c})$ is a squared Euclidean distance between points $\mathbf{x}$ and $\mathbf{c}$. Nevertheless even if $\mathbf{E}$ is just a positive definite (not necessarily identity) matrix, it is easy to see that the function $\text{dist}(\mathbf{x},\mathbf{c}) = \|\mathbf{x}-\mathbf{c}\|_{\mathbf{E}} := \sqrt{(\mathbf{x}-\mathbf{c})^\top \mathbf{E}(\mathbf{x}-\mathbf{c})}$ satisfies all distance axioms in $\mathbb{R}^n$.

Furthermore we can also describe an ellipsoid as a set

$$E := \{\mathbf{x} \in \mathbb{R}^n \,|\, f(\mathbf{x}) \leq 0\} \tag{7.3}$$

where

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{C}\mathbf{x} + 2\mathbf{d}^\top \mathbf{x} + e = \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}^\top \begin{pmatrix} \mathbf{C} & \mathbf{d} \\ \mathbf{d}^\top & e \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$
$$= \left(\mathbf{x}^\top 1\right) \widehat{\mathbf{E}} \left(\mathbf{x}^\top 1\right)^\top, \qquad \mathbf{C} \succ \mathbf{0}, \qquad e - \mathbf{d}^\top \mathbf{C}^{-1} \mathbf{d} < 0\,.$$

If we set $\mathbf{C} := \mathbf{E}$, $\mathbf{d} := -\mathbf{E}\mathbf{c}$ and $e := \mathbf{c}^\top \mathbf{E}\mathbf{c} - 1$ and add 1 to both sides of the inequality, then this description is simply (7.2). In fact, (7.3) also defines a degenerate ellipsoid centred at the origin in the augmented space $\mathbb{R}^{n+1}$ (although the matrix $\widehat{\mathbf{E}}$ is not positive semidefinite because of the condition $e - \mathbf{d}^\top \mathbf{C}^{-1} \mathbf{d} < 0$) which has zero volume since it entirely belongs to the hyperplane $x_{n+1} = 1$. In this case the requirements $\mathbf{C} \succ \mathbf{0}$, $e - \mathbf{d}^\top \mathbf{C}^{-1} \mathbf{d} < 0$ are equivalent to the requirements to the projection of this ellipsoid onto the plane $x_{n+1} = 1$ to be a non-degenerate ellipsoid. A similar mapping will be used in the sequel in order to remove an unknown ellipsoid centre $\mathbf{c}$ from the optimization problem while searching for the optimal separating ellipsoid and thus avoid the products of variables in the constraint set.

In the rest of this manuscript for the sake of clarity of our derivations we will refer to each of these definitions depending on the problem setting. In order to avoid confusion we will attempt to state explicitly which description of an ellipsoid we use if that is not clear from the context.

Since the affine mapping specified in (7.1) is invertible (if E is a full-dimensional ellipsoid), the volume of the ellipsoid E is proportional to the factor $\det \mathbf{A}$ to the volume of the unit ball in $\mathbb{R}^n$. Therefore we can express the volume of a non-degenerate ellipsoid $E \in \mathbb{R}^n$ as

$$\operatorname{vol} E = b_n \det \mathbf{A} = b_n (\det \mathbf{E})^{-\frac{1}{2}}, \tag{7.4}$$

where b_n is the volume of a unit ball in $\mathbb{R}^n$. Since the value b_n is constant for a given n, the determinant of the ellipsoid matrix can be used in volume optimization problems for ellipsoids.

7.2 Motivation and Geometrical Interpretation

As seen from the definitions, ellipsoids are convex sets that are easy to describe whose volumes are easy to compute and which generalize many common types of sets and geometrical objects (like points, segments, etc.) as degenerate cases. Moreover since an ellipsoid is an affine set itself and its image under an affine mapping is again an ellipsoid, the family of ellipsoids is closed with respect to affine transformations. These features of ellipsoids imply their wide use in optimization to approximate sets with more complex structures.

The most common ellipsoidal approximation problems are inner and outer approximations of some given or unknown convex set.

Definition 7.2 (Inner and Outer Ellipsoidal Approximations)**.** Given some arbitrary set S an ellipsoid E is called

- *inner approximation* of the set S if it is the largest (maximal volume) ellipsoid contained in S.
- *outer approximation* of the set S if it is the smallest (minimal volume) ellipsoid containing S.

In the literature on convex optimization inner and outer ellipsoidal approximations are sometimes also called *extremal ellipsoids*.

It is known from Löwner-John theorem (John, 1948) that inner and outer approximations always exist and are unique if set S is closed, bounded and solid. However finding such an ellipsoid in some cases may be computationally intractable problem depending on the type and description of the set S we use (Ben-Tal and Nemirovski, 2001).

We use ellipsoids in the framework of the binary pattern classification problems. Assume that we have a vector representation in $\mathbb{R}^n$ of training and test sets which contains examples of two classes, e.g. positive and negative. Using the approach from (Glineur, 1998) we construct two co-centred ellipsoids with the same directions and proportional lengths of the axes while one of them is of minimal size to include all examples of one class, and the other is of maximal size to exclude all examples of the other class. Therefore the optimization criterion is to maximize the squared separation ratio between the corresponding half-axes of outer and inner ellipsoids. This leads to an SDP, and if the data are separable then the SDP is feasible and bounded so its solution exists and is unique. Finally, we construct a third ellipsoid with the same center $\mathbf{c}$ and axis directions $\mathbf{E}$, but its half-axes are means of the half-axes of the previous two. We use the latter ellipsoid as a classification function which assigns a label to the point depending on whether the point belongs to its interior or exterior (see Figure 7.2).

7.3 Semidefinite Programming Formulation

Let us assume that we have m vector representations of observations together with their labels, $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in (\mathbb{R}^n \times \{-1, +1\})^m$, where $\mathbf{x}_i$ are the vector representations of the inputs and y_i are the classes (either positive or negative). We search for two separating ellipsoids with common matrix $\mathbf{E}$ and centre $\mathbf{c}$, and maximal separation ratio ρ. Using the definition of an ellipsoid (7.1) we can formulate the following optimization

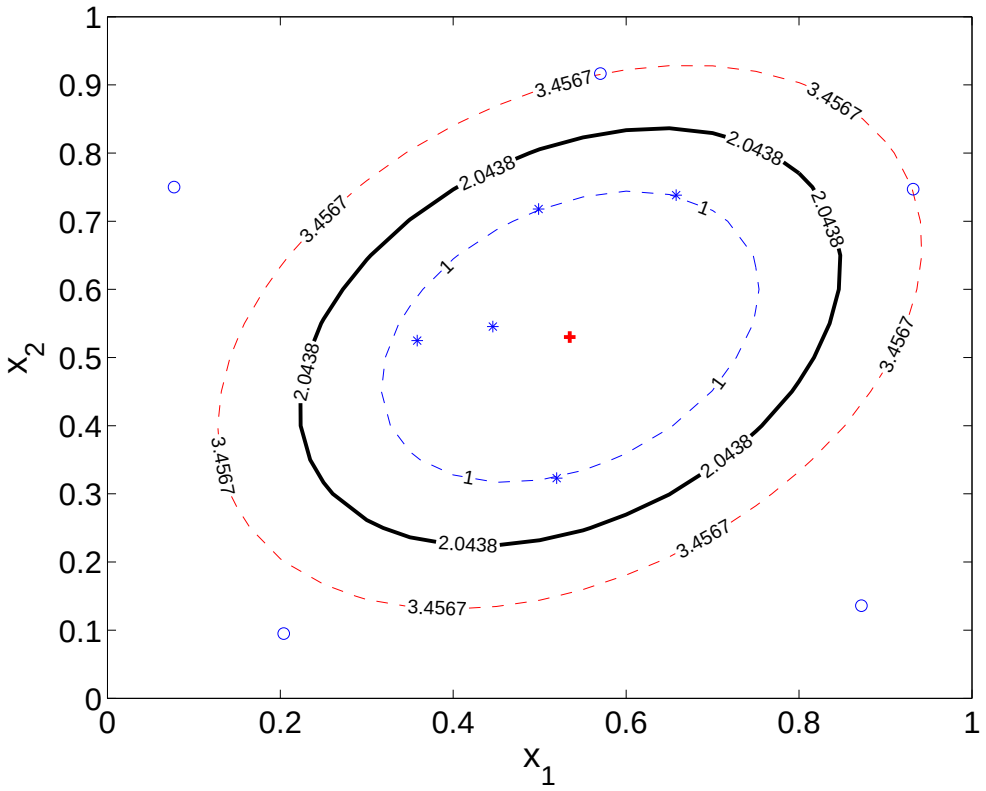


FIGURE 7.2: Example of ellipsoid separation. The maximal margin ellipsoid is plotted as a solid curve; two separating ellipsoids are plotted as dashed curves. The positive points are marked with asterisks and the negative ones with circles.

problem

$$\begin{aligned} \text{max} \quad & \rho^2 \\ \text{subject to} \quad & (\mathbf{x}_i - \mathbf{c})^\top \mathbf{E} (\mathbf{x}_i - \mathbf{c}) \leq 1\,, \qquad i \in I_+, \\ & (\mathbf{x}_i - \mathbf{c})^\top \mathbf{E} (\mathbf{x}_i - \mathbf{c}) \geq \rho^2\,, \qquad i \in I_-, \\ & \mathbf{E} \succ \mathbf{0}, \end{aligned} \tag{7.5}$$

where $I_+ := \{i \in \{1, \ldots, m\} \,|\, y_i = +1\}$ and $I_- := \{i \in \{1, \ldots, m\} \,|\, y_i = -1\}$.

However, it is easy to see that solving (7.5) is not a straightforward task since neither its objective, nor its constraints are linear in terms of the unknown variables. In order to overcome the first difficulty we just denote the squared separation ratio as a new variable $k := \rho^2$ since we never use the actual (non-squared) separation ratio in the problem. For coping with the other obstacle we exploit the trick described in the alternative description of an ellipsoid (7.3): we map the problem into the $(n+1)$-dimensional space where we attempt to separate the augmented dataset (with the $(n+1)$-th components equal to 1) with an ellipsoid centred at the origin, and then project the solution back to the original space. Before making the latter re-formulation of problem (7.5) we need

to check whether maximizing the separation ratio in the augmented space we will also maximize it in the initial space, and if the solution in the augmented space is unique.

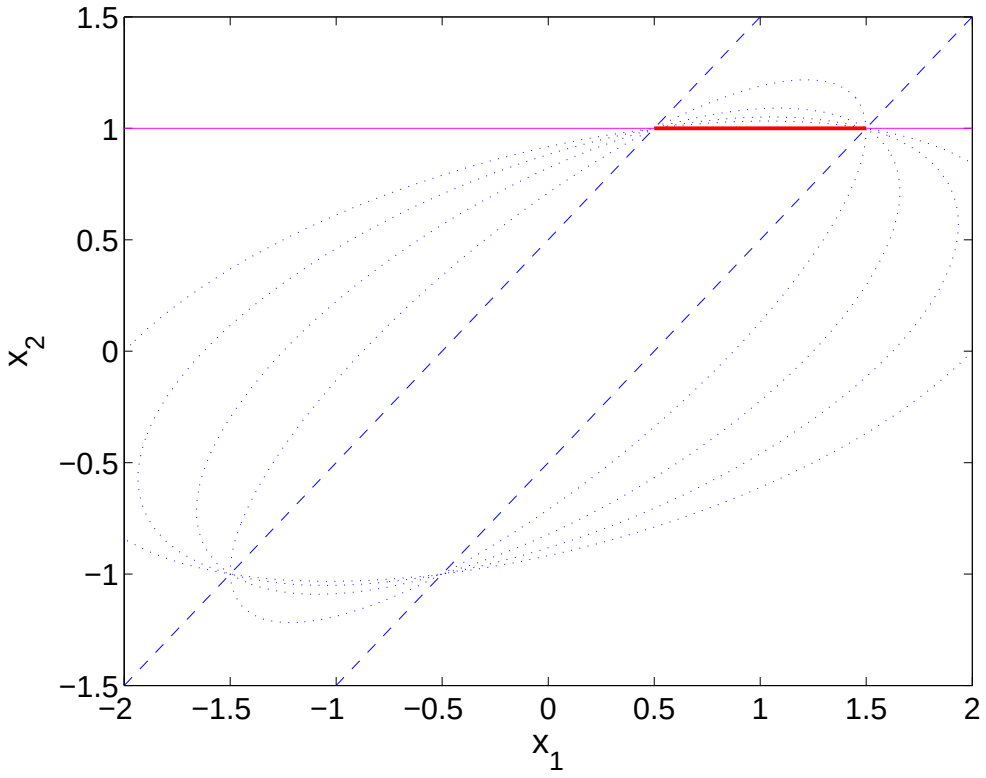


FIGURE 7.3: The number of ellipsoids in the augmented space centred at the origin whose projection to the initial space corresponds to the optimal solution is infinite. In order to resolve this problem we require for the ellipsoid in the augmented space to have infinite length in one direction (degenerate infinite volume ellipsoid).

We start with resolving the problem of the solution uniqueness. At first glance, it becomes clear that indeed there is an infinite number of $(n+1)$-dimensional ellipsoids centred at the origin corresponding to some chosen ellipsoid in the original space (c.f. Figure 7.3). Therefore we have to ensure the uniqueness of the solution of our problem by imposing some extra requirements. Generally speaking, we have to choose only one from all those ellipsoids so we fix our solution to be the one with an infinite volume in the added dimension, i.e. an elliptic cylinder of an infinite height (in the 2-dimensional case - the set of points which belong to or lie between two lines, e.g. two dashed lines in Figure 7.3). Now we proceed to a mathematical formulation of the problem.

Let us recall the description of a degenerate zero volume ellipsoid in $\mathbb{R}^{n+1}$ in the form (7.3), i.e. described by the inequality $f(\mathbf{x}) = \left(\mathbf{x}^\top 1\right) \widehat{\mathbf{E}} \left(\mathbf{x}^\top 1\right)^\top \leq 0$ and some extra requirements. It is not convenient to describe our optimal classifier this way since the matrix $\widehat{\mathbf{E}}$ is not positive semidefinite. In order to cast our problem in the framework of semidefinite programming we change the main inequality in (7.3) to $f(\mathbf{x}) \leq 1$, and also replace the condition $e - \mathbf{d}^\top \mathbf{C}^{-1} \mathbf{d} < 0$ with the one that requires positive semidefiniteness

of $\widehat{\mathbf{E}}$ that is $e - \mathbf{d}^\top \mathbf{C}^{-1}\mathbf{d} \geq 0$ (Schur complement of $\widehat{\mathbf{E}}$). Now we expand the function $f(\mathbf{x})$ for some positive example $\mathbf{x}_+$ applying the transformation $\mathbf{d} := -\mathbf{C}\mathbf{c}$ as

$$\begin{aligned} f(\mathbf{x}_+) &= \begin{pmatrix} \mathbf{x}_+ \\ 1 \end{pmatrix}^\top \begin{pmatrix} \mathbf{C} & \mathbf{d} \\ \mathbf{d}^\top & e \end{pmatrix} \begin{pmatrix} \mathbf{x}_+ \\ 1 \end{pmatrix} \\ &= \mathbf{x}_+^\top \mathbf{C}\mathbf{x}_+ + 2\mathbf{d}^\top \mathbf{x}_+ + e \\ &= \mathbf{x}_+^\top \mathbf{C}\mathbf{x}_+ - 2\mathbf{c}^\top \mathbf{C}\mathbf{x}_+ + \mathbf{c}^\top \mathbf{C}\mathbf{c} + e - \mathbf{c}^\top \mathbf{C}\mathbf{c} \\ &= (\mathbf{x}_+ - \mathbf{c})^\top \mathbf{C}(\mathbf{x}_+ - \mathbf{c}) + e - \mathbf{c}^\top \mathbf{C}\mathbf{c} \quad \leq \quad 1, \end{aligned}$$

which can be re-written as

$$(\mathbf{x}_+ - \mathbf{c})^\top \mathbf{C}(\mathbf{x}_+ - \mathbf{c}) \quad \leq \quad \eta \quad \Longleftrightarrow \quad (\mathbf{x}_+ - \mathbf{c})^\top \frac{\mathbf{C}}{\eta}(\mathbf{x}_+ - \mathbf{c}) \quad \leq \quad 1 \qquad (7.6)$$

with $\eta := 1 - e + \mathbf{c}^\top \mathbf{C}\mathbf{c}$[1]. Equation (7.6) is the definition of an ellipsoid in the form (7.2). Therefore it has been shown that the matrix $\widehat{\mathbf{E}} \in \mathbb{S}_+^{n+1}$ defines an ellipsoid in $\mathbb{R}^n$ described by its centre $\mathbf{c}$ and the matrix $\mathbf{E} := \frac{\mathbf{C}}{\eta}$ under the conditions

$$\mathbf{C} \succ \mathbf{0} \quad \text{and} \quad e - \mathbf{d}^\top \mathbf{C}^{-1}\mathbf{d} \geq 0 \qquad \Longleftrightarrow \qquad \widehat{\mathbf{E}} \succeq 0. \qquad (7.7)$$

Next we consider the value η and its role in the description (7.6). Certainly we need it to be positive (otherwise (7.6) does not hold). Fortunately this is guaranteed by the positive definiteness of $\mathbf{C}$ since $\eta \geq (\mathbf{x}_i - \mathbf{c})^\top \mathbf{C}(\mathbf{x}_i - \mathbf{c}) > 0$, $\forall i \in I_+$, $\mathbf{x}_i \neq \mathbf{c}$ which holds if the training data are separable. On the other hand,

$$0 \leq e - \mathbf{d}^\top \mathbf{C}^{-1}\mathbf{d} = e - \mathbf{c}^\top \mathbf{C}\mathbf{c} = 1 - \eta \quad \Longleftrightarrow \quad \eta \leq 1.$$

Consequently, $0 \leq \eta \leq 1$. We prefer η to be equal to 1 because then all transformations are much simpler since $\mathbf{E} = \mathbf{C}$, and in this case we will call $\widehat{\mathbf{E}}$ a *canonical* homogeneous form of the ellipsoid described by $\mathbf{E}$ and $\mathbf{c}$. Note that also every ellipsoid in $\mathbb{R}^n$ has a unique canonical homogeneous representation

$$\begin{pmatrix} \mathbf{E} & -\mathbf{E}\mathbf{c} \\ -\mathbf{E}\mathbf{c} & \mathbf{c}^\top \mathbf{E}\mathbf{c} \end{pmatrix}. \qquad (7.8)$$

Therefore there is a bijective mapping between ellipsoids in $\mathbb{R}^n$ and their canonical homogeneous representations in $\mathbb{R}^{n+1}$. Moreover, since for the canonical homogeneous representation $e - \mathbf{d}^\top \mathbf{C}^{-1}\mathbf{d} = 1 - \eta = 0$ (i.e. matrix $\hat{\mathbf{E}}$ has exactly one zero eigenvalue), every canonical homogeneous ellipsoid is degenerate with zero volume (c.f. Figure 7.3).

Now we check if maximizing the separation ratio in the augmented space we will maximize it in the initial space. In the separable case, for every negative example $\mathbf{x}_-$, we

[1] In (Glineur, 1998) the value $\delta := 1 - \eta$ is used to characterize the solution in $\mathbb{R}^{n+1}$. However we prefer η since we consider that it clarifies the derivation of the method.

have

$$k \leq \left(\mathbf{x}_-^\top 1\right) \widehat{\mathbf{E}} \left(\mathbf{x}_-^\top 1\right)^\top = \eta(\mathbf{x}_- - \mathbf{c})^\top \mathbf{E}(\mathbf{x}_- - \mathbf{c}) + 1 - \eta \quad \Longleftrightarrow \quad (\mathbf{x}_- - \mathbf{c})^\top \mathbf{E}(\mathbf{x}_- - \mathbf{c}) \geq \frac{t+\eta}{\eta}$$

where $t := k - 1 = \rho^2 - 1$. Hence, we conclude that the canonical homogeneous representation preserves the optimal separation ratio achieved in the original space.

For convenience we denote by $\hat{\mathbf{x}}$ the result of the mapping $\mathbb{R}^n \to \mathbb{R}^{n+1} : \mathbf{x} \to \left(\mathbf{x}^\top, 1\right)^\top$. Now we can re-state (7.1) as the semidefinite program

$$\begin{array}{lll} \min\limits_{t,\widehat{\mathbf{E}}} & -t & \\ \text{subject to} & y_i(1 - \hat{\mathbf{x}}_i^\top \widehat{\mathbf{E}} \hat{\mathbf{x}}_i) \geq t\tau_{y_i}\,, \qquad i = 1, \ldots, m\,, & (7.9) \\ & \widehat{\mathbf{E}} \succeq \mathbf{0}\,, & \end{array}$$

where $\tau_{+1} = 0$ and $\tau_{-1} = 1$.

7.4 Analysis and Properties of the Algorithm

In this section we quote and discuss some properties of the semidefinite program (7.9) proved in (Glineur, 1998).

First of all, as we made clear in the previous section we are looking for a canonical solution of (7.9). Although we cannot add this requirement ($\eta = 1$) explicitly to the problem formulation because it is not convex, it turns out that if the data are separable then the solution of (7.9) possesses this feature.

Theorem 7.3 (Glineur (1998))**.** *If there exists a separating ellipsoid, the optimal solution of (7.9) has $t > 0$ and the optimal homogeneous ellipsoid is canonical. If there is no separating ellipsoid, the optimal solution of (7.9) has $t = 0$.*

Note that in the unseparable case the algorithm returns the default feasible solution

$$\widehat{\mathbf{E}} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \qquad \text{and} \qquad t = 0 \tag{7.10}$$

which corresponds to the set of points that belong to and lie between two parallel hyperplanes $x_{n+1} = +1$ and $x_{n+1} = -1$. In this case $\eta = 0$ so there is no solution in the original space.

Corollary 7.4 (Glineur (1998))**.** *If there exists a separating ellipsoid, the semidefinite program (7.9) provides an ellipsoid with the highest possible separation ratio.*

The other nice feature of the maximal separation ratio algorithm (MAXSEP) is its independence of invertible affine transformations of the coordinate system.

Theorem 7.5 (Glineur (1998))**.** *Given two sets of points to be separated, the maximal separation ratio and the ellipsoids that achieve it are independent of the coordinate system.*

We compared coordinate system invariance under invertible affine transformations of the maximal separation ratio algorithm and the support vector machine in $\mathbb{R}^2$. We used a polynomial kernel of degree 2 ($k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + 1)^2$ - which is actually not invariant)[2] to make the comparison fair because an ellipsoid is described by the polynomial of the order 2 (c.f. (7.3)). However even in this case the SVM algorithm still has an advantage over ellipsoid separation because its hypothesis space is much larger (it contains *any* separation curve that can be represented by a polynomial of order 2). The data were generated according to Gaussian distribution centred at the origin with standard deviation chosen to allow a unit ball to contain 97.8 per cent of points (points outside the ball were ignored). Positive examples were chosen to be points inside the ball, and negative ones were obtained by mapping the points from the interior of the ball in the polar coordinate system via inverting the distance from the centre. Generated datasets were mapped to spaces with new metric defined by some randomly generated positive definite matrix and shifted origin. The results were averaged over 500 tests and error rate is plotted at Figure 7.4 against the exponent of the values of the rescaling coefficient.

As one can see from Figure 7.4 the error rate of the maximal separation ratio algorithm remains the same after all rescalings of the dataset. SVM with polynomial kernel is not coordinate system invariant so one has to rescale the dataset to make values of all dimensions be of approximately the same magnitude prior to the learning phase. Otherwise some dimensions will influence classification more than others. The error rate of the SVM grows fast up to the ratio of negative examples in the test set when the rescaling factor goes to 0 faster when all dimensions are rescaled then if we rescale only one of the dimensions because the margin vanishes faster in the former case. If we rescale only one dimension then polynomial kernel of degree 2 degenerates first into the polynomial kernel of degree 1 and later into the constant kernel.

7.5 Dual Problem

Let us introduce a Lagrangian for (7.9) in the following form:

$$\mathcal{L}\left(t, \widehat{\mathbf{E}}, \boldsymbol{\lambda}, \mathbf{Z}\right) := -t + \sum_{i=1}^{m} \lambda_i \left(t\tau_{y_i} - y_i \left(1 - \mathbf{x}_i^\top \widehat{\mathbf{E}} \mathbf{x}_i\right)\right) - \operatorname{tr} \mathbf{Z}\widehat{\mathbf{E}}. \tag{7.11}$$

[2]We used the MATLAB implementation of the support vector machine algorithm made by Ralf Herbrich from Microsoft Research in Cambridge.

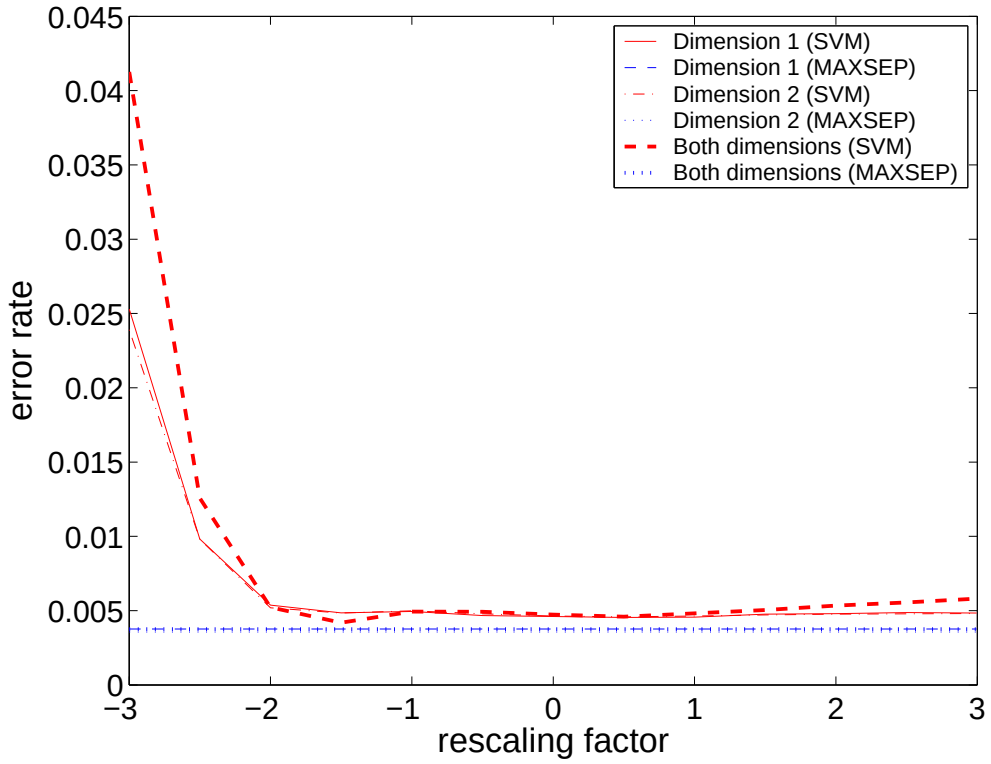


FIGURE 7.4: Comparison of coordinate rescaling invariance of the maximal separation ratio algorithm and the SVM with a polynomial kernel of degree 2. The exponent of the rescaling coefficient is plotted along the x-axis.

Indeed, equation (7.11) is a barrier function for (7.9) since

$$\phi(t,\widehat{\mathbf{E}}) = \max_{\boldsymbol{\lambda}\succeq 0,\mathbf{Z}\succeq 0} \mathcal{L}\left(t,\widehat{\mathbf{E}},\boldsymbol{\lambda},\mathbf{Z}\right) = \begin{cases} -t & y_i(1-\mathbf{x}_i^\top\widehat{\mathbf{E}}\mathbf{x}_i) \geq t\tau_{y_i}\,, i=1,\ldots,m\,, \widehat{\mathbf{E}}\succeq 0, \\ +\infty & \text{otherwise.} \end{cases} \tag{7.12}$$

Therefore we can cast (7.9) as an unconstrained optimization problem, and after applying the weak duality theorem get

$$p^\star = \min_{t,\widehat{\mathbf{E}}\succeq 0} \max_{\boldsymbol{\lambda}\succeq 0,\mathbf{Z}\succeq 0} \mathcal{L}\left(t,\widehat{\mathbf{E}},\boldsymbol{\lambda},\mathbf{Z}\right) \geq \max_{\boldsymbol{\lambda}\succeq 0,\mathbf{Z}\succeq 0} \min_{t,\widehat{\mathbf{E}}\succeq 0} \mathcal{L}\left(t,\widehat{\mathbf{E}},\boldsymbol{\lambda},\mathbf{Z}\right) = d^\star. \tag{7.13}$$

The minimization problem on the right hand side of the inequality (7.13) can be simplified to

$$\begin{aligned} \min_{t,\widehat{\mathbf{E}}\succeq 0} \mathcal{L}\left(t,\widehat{\mathbf{E}},\boldsymbol{\lambda},\mathbf{Z}\right) &= -t + \sum_{i=1}^m \lambda_i \left(t\tau_{y_i} - y_i\left(1-\mathbf{x}_i^\top\widehat{\mathbf{E}}\mathbf{x}_i\right)\right) - \operatorname{tr}\mathbf{Z}\widehat{\mathbf{E}} \\ = \quad t\left(\sum_{i=1}^m \lambda_i\tau_{y_i} - 1\right) &- \sum_{i=1}^m \lambda_i y_i + \operatorname{tr}\left(\sum_{i=1}^m y_i\lambda_i\mathbf{x}_i\mathbf{x}_i^\top - \mathbf{Z}\right)\widehat{\mathbf{E}}, \end{aligned} \tag{7.14}$$

and after taking partial derivatives with respect to the primal variables t and $\widehat{\mathbf{E}}$ and equating them to zero we get

$$\sum_{i=1}^{m} \lambda_i \tau_{y_i} = 1, \tag{7.15}$$

$$\mathbf{Z} = \sum_{y=1}^{m} \lambda_i y_i \mathbf{x}_i \mathbf{x}_i^\top. \tag{7.16}$$

Also from Karush-Kuhn-Tucker complementarity conditions of Fletcher (Fletcher, 1987) we obtain

$$\lambda_i \left(t\tau_{y_i} - y_i \left(1 - \mathbf{x}_i^\top \widehat{\mathbf{E}} \mathbf{x} \right) \right) = 0, \quad i \in \{1, \ldots, m\}, \tag{7.17}$$

$$\operatorname{tr} \mathbf{Z}\widehat{\mathbf{E}} = 0. \tag{7.18}$$

Now the dual problem to (7.9) can be posed as

$$\begin{array}{rc} \max\limits_{\boldsymbol{\lambda}} & -\sum_{i=1}^{m} y_i \lambda_i \\ \text{subject to} & \sum_{i=1}^{m} y_i \lambda_i \mathbf{x}_i \mathbf{x}_i^\top \succeq 0, \\ & \sum_{i=1}^{m} \lambda_i \tau_{y_i} = 1, \quad \lambda_i \geq 0, \quad i \in \{1, \ldots, m\}. \end{array} \tag{7.19}$$

Now if we recall that $\tau_{+1} = 0$ and $\tau_{-1} = 1$ then using condition (7.15) we can re-write the objective function of the dual in the form

$$\max_{\boldsymbol{\lambda} \geq 0} -\sum_{i=1}^{m} y_i \lambda_i = \min_{\boldsymbol{\lambda} \geq 0} \left(\sum_{i \in I_-} \lambda_i - \sum_{i \in I_+} \lambda_i \right) = 1 - \max_{\boldsymbol{\lambda} \geq 0} \sum_{i \in I_+} \lambda_i. \tag{7.20}$$

Now we can re-write the dual problem as

$$\begin{array}{rc} \min\limits_{\boldsymbol{\lambda} \geq 0} & \sum_{i \in I_+} \lambda_i \\ \text{subject to} & \sum_{i \in I_+} \lambda_i \mathbf{x}_i \mathbf{x}_i^\top - \sum_{i \in I_-} \lambda_i \mathbf{x}_i \mathbf{x}_i^\top \succeq 0, \\ & \sum_{i \in I_-} \lambda_i = 1. \end{array} \tag{7.21}$$

From formulation (7.21) we conclude that the solution of the dual problem has a sparse representation, and moreover we can measure the contribution of every example and extract those which correspond to positive λ_i ("support examples"). The dual problem (7.21) also explicitly requires the presence of at least one negative example in the dataset which is an obvious requirement since otherwise the separation problem does not make any sense. Now we consider complementarity conditions. From (7.17) we conclude that $\lambda_i > 0$ if and only if $\mathbf{x}_i$ belongs to the boundary of either inner or outer ellipsoid. Therefore we can introduce two new sets of indices $I_+^\star := \{i = 1, \ldots, m \,|\, y_i = +1,\ \mathbf{x}_i^\top \widehat{\mathbf{E}} \mathbf{x}_i = 1\}$

and $I_-^\star := \{i = 1, \ldots, m \,|\, y_i = -1,\, \mathbf{x}_i^\top \widehat{\mathbf{E}} \mathbf{x}_i = t + 1\}$ and use them instead of I_+ and I_- in the formulation of the dual problem (7.21). Using (7.16) we can re-write (7.18) as

$$\begin{aligned}
\operatorname{tr} \mathbf{Z}\widehat{\mathbf{E}} &= \operatorname{tr} \left(\sum_{i \in I_+} \lambda_i \mathbf{x}_i \mathbf{x}_i^\top - \sum_{i \in I_-} \lambda_i \mathbf{x}_i \mathbf{x}_i^\top \right) \widehat{\mathbf{E}} \\
&= \operatorname{tr} \left(\sum_{i \in I_+^\star} \lambda_i \mathbf{x}_i \mathbf{x}_i^\top - \sum_{i \in I_-^\star} \lambda_i \mathbf{x}_i \mathbf{x}_i^\top \right) \widehat{\mathbf{E}} \\
&= \sum_{i \in I_+^\star} \lambda_i \mathbf{x}_i^\top \widehat{\mathbf{E}} \mathbf{x}_i - \sum_{i \in I_-^\star} \lambda_i \mathbf{x}_i^\top \widehat{\mathbf{E}} \mathbf{x}_i \\
&= \sum_{i \in I_+^\star} \lambda_i - (t+1) \sum_{i \in I_-^\star} \lambda_i \;=\; \sum_{i \in I_+^\star} \lambda_i - t - 1 \;=\; 0, \qquad (7.22)
\end{aligned}$$

which is equivalent to the zero duality gap optimality condition (if we consider the dual problem in the formulation (7.19)). From (7.22) we also notice the following interesting property of the dual solution: if the problem is unseparable (and the algorithm returns the default solution (7.10) with t equal 0 and $\widehat{\mathbf{E}}$ be a rank-one matrix with all zero elements except the last one in the last row) then the dual solution $\mathbf{Z}$ is a difference between convex combinations of Kronecker products of the positive and negative examples respectively (since $\sum_{I_+^\star} \lambda_i = 1$). However, if separation is possible, then a conic combination (since $\sum_{I_+^\star} \lambda_i = 1$ is no longer met because $\sum_{I_+^\star} \lambda_i > 1$) of tensor products of positive examples is used. In the unseparable case all λ_i becomes non-zero since the whole training set belongs to the one of two hyperplanes which are boundaries of the default solution in the augmented space. This also implies that unless we have separation in the training set we cannot use the chunking approach (similar to the one presented in (Mangasarian and Musicant, 1999) for SVM) to learning the optimal ellipsoid classifier.

Condition (7.18) also implies that $\mathbf{Z}\widehat{\mathbf{E}} = \widehat{\mathbf{E}}\mathbf{Z} = 0$ which means that the matrices $\mathbf{Z}$ and $\widehat{\mathbf{E}}$ share a common eigenbasis, but for every common eigenvector either the eigenvalue of $\mathbf{Z}$, or the eigenvalue of $\widehat{\mathbf{E}}$, or both are equal to 0 (similar to complementary slackness in linear programming) (Ben-Tal and Nemirovski, 2001). Considering the fact that two co-centred separating ellipsoids exist if and only if $\widehat{\mathbf{E}} \succeq \mathbf{0}$ we conclude that in the separable case the matrix $\mathbf{Z}$ has exactly one non-zero eigenvalue. However the constraint $\mathbf{Z} = \mathbf{v}\mathbf{v}^\top$ is not convex so we cannot incorporate it into the dual problem in order to search for $\mathbf{Z}$ directly in this representation.

Chapter 8

Text Categorization via Ellipsoid Separation

8.1 Problem Formulation and Features

Ellipsoid separation would appear to be most effectively applicable to such types of binary classification problems where a set of examples with one label (e.g. positive set) is much smaller and single-clustered than the other set because in this case we can find a mapping to some feature space where we can separate the two classes by enclosing the smaller class inside the inner ellipsoid, and keeping the larger one outside the outer ellipsoid. One such area is document classification based on semantic content (text categorization) since the class of relevant documents is usually of much smaller size than the set of all available documents.

The problem of document categorization may arise when the documents from some set have to be ranked according to their relevance to some usually predefined set of topics (i.e. classification of news articles based on their dealing with business topics). In this chapter we discuss a new batch learning algorithm for text classification that applies non-linear ellipsoid separation discussed in the previous chapter to the vector space representation of text documents. We use the bag-of-words vector representation of text documents, the maximal separation ratio method for pattern separation via ellipsoids (Glineur, 1998), and the approximation of the latent semantic feature extraction technique with Gram-Schmidt orthogonalization (GSK algorithm) (Cristianini et al., 2002). We present numerical results which indicate some potential for the given approach. The rest of the chapter is organized as follows: we describe the general formulation of the algorithm in this section, the specific problems of applying it to text documents in Section 8.2, and show how latent semantic feature extraction can help dealing with some of the resulting problems in Section 8.3, followed by numerical results in Section 8.4.

Let us assume we have computed vector representations of the set of m labeled documents $((\hat{\mathbf{d}}_1, y_1), \ldots, (\hat{\mathbf{d}}_m, y_m)) \in (\mathbb{R}^n \times \{-1, +1\})^m$ where $\hat{\mathbf{d}}_i$ are the feature vectors and y_i are the document labels for all $i \in \{1, \ldots, m\}$. Moreover, we shall use a mapping $\boldsymbol{\phi} : \hat{\mathbf{d}} \to (\hat{\mathbf{d}}^\top, 1)^\top$ in order to search for the separation ellipsoid in its canonical homogeneous form. We denote the mapped inputs by $\hat{\mathbf{x}}_i := \boldsymbol{\phi}(\hat{\mathbf{d}}_i)$. Now we apply the maximal separation ratio algorithm discussed in the previous chapter to separate the positive from the negative vector representations of the text documents, i.e. aim to enclose all positive examples inside an inner ellipsoid, and then look for the co-centered ellipsoid of maximal size with the same axis directions to keep all negative examples outside. In other words we require $\hat{\mathbf{x}}_i^\top \widehat{\mathbf{E}} \hat{\mathbf{x}}_i \leq 1$ for all $i \in \{1, \ldots, m : y_i = +1\}$ (for positive examples) and $\hat{\mathbf{x}}_i^\top \widehat{\mathbf{E}} \hat{\mathbf{x}}_i \geq 1 + t$ for all $i \in \{1, \ldots, m : y_i = -1\}$ (for negative examples) where t is a squared distance between two separating ellipsoids in a metric space with the norm $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^\top \mathbf{E} \mathbf{x}}$. This implies that $t + 1$ is the squared separation ratio, i.e. the ratio between corresponding half-axes of outer and inner ellipsoids. Combining these two sets of inequalities we get the same semidefinite minimization program as problem (7.9)

$$\begin{aligned} \min_{t, \widehat{\mathbf{E}}} \quad & -t \\ \text{subject to} \quad & y_i(1 - \hat{\mathbf{x}}_i^\top \widehat{\mathbf{E}} \hat{\mathbf{x}}_i) \geq t \tau_{y_i}\,, \qquad i = 1, \ldots, m\,, \\ & \widehat{\mathbf{E}} \succeq \mathbf{0} \end{aligned} \tag{8.1}$$

where $\tau_{+1} = 0$ and $\tau_{-1} = 1$. Using the solution of (8.1) we define a class of ellipsoid classifiers $h_{t,\widehat{\mathbf{E}}} : \mathbb{R}^n \to \mathbb{R}$ parametrized by a symmetric positive semidefinite matrix $\widehat{\mathbf{E}} \in \mathbb{S}_+^{n+1}$ and some positive value $t \in \mathbb{R}_+$ such that

$$h_{t,\widehat{\mathbf{E}}}(\hat{\mathbf{x}}) := \operatorname{sign}(f_{t,\widehat{\mathbf{E}}}(\hat{\mathbf{x}}))\,, \qquad f_{t,\widehat{\mathbf{E}}}(\hat{\mathbf{x}}) := \left(\frac{1 + \sqrt{t+1}}{2}\right)^2 - \hat{\mathbf{x}}^\top \widehat{\mathbf{E}} \hat{\mathbf{x}}, \tag{8.2}$$

where the latter equation corresponds to checking if the point $\hat{\mathbf{x}}$ belongs to the interior of the intermediate ellipsoid (see Figure 7.2) since $\left(\frac{1+\sqrt{t+1}}{2}\right)^2$ is a squared mean of 1 and optimal separation ratio $\sqrt{t+1}$. In this way we generalize the maximal separation ratio algorithm for classification problems.

8.2 Vector Representation of Text Documents

The most common approach used in learning for text categorization problems is mapping the set of documents to some linear metric space (feature space), and then applying learning classification techniques based on distance functions in that space. For this purpose the documents are often mapped using the so-called bag-of-words approach when the occurrence of every distinct word from the set of documents (excluding stop-words, articles, and prepositions) is counted as a separate dimension, and thus every document

Algorithm 1 Gram-Schmidt Kernel (GSK) Feature Extraction Algorithm

Require: A training set $((\mathbf{d}_i, y_i), \ldots, (\mathbf{d}_m, y_m)) \in \left(\mathbb{R}^l \times \{-1, +1\}\right)^m$, bias $B \in \mathbb{R}_+$ and a dimension of the subspace, $k \in \mathbb{N}$
  for $i = 1, \ldots, m$ **do**
    $n_i = \mathbf{d}_i^\top \mathbf{d}_i$
  end for
  for $j = 1, \ldots, k$ **do**
    for $i = 1, \ldots, m$ **do**
      $b_i = B^{\frac{y_i+1}{2}} \cdot n_i$
    end for
    $i_j = \operatorname{argmax}_i b_i$
    for $i = 1, \ldots, m$ **do**
      $\hat{\mathbf{D}}_{i,j} \leftarrow \frac{1}{\sqrt{\mathbf{n}_{i_j}}} \cdot \left(\mathbf{d}_i^\top \mathbf{d}_{i_j} - \sum_{t=1}^{j-1} \hat{\mathbf{D}}_{i,t} \hat{\mathbf{D}}_{i_j,t}\right)$
      $n_i \leftarrow n_i - \hat{\mathbf{D}}_{i_j}^2$
    end for
  end for
  return matrix $\hat{\mathbf{D}}$ with the training set mapped into the feature subspace stored in its rows

is represented as a vector with word frequencies as its components. Furthermore, the vectors are often normalized.

The obvious drawback of the bag-of-words vector representation of the document set is the high dimensionality of the vector space because of the high number of distinct terms in the text. As a result the SDP will be computationally impractical. Moreover the dimensionality of the space is usually greater than the size of the dataset so even if a solution of the SDP could be computed it would be degenerate since we need at least $n + 1$ points in order to define an ellipsoid in an n-dimensional space.

In order to solve this problem we need to look for some subspace of the bag-of-words vector representation of the text documents. This subspace must have much lower dimensionality while preserving the ellipsoid separability of the original space. For this purpose we need to use some feature extraction algorithm. In the next section a variant of latent semantic feature extraction will be described that meets both requirements.

8.3 Latent Semantic Feature Extraction

The weakness of the bag-of-words approach is its total ignorance of the occurrence of semantically similar words, e.g. synonyms. Ideally, semantically similar documents are mapped to the same directions in the feature space. Although the explicit computation of the co-occurrence of semantically similar words is a rather expensive procedure, the latent semantic indexing approach from information retrieval constructs a feature space based on semantic similarity between different words.

Algorithm 2 Gram-Schmidt Kernel (GSK) Algorithm for New Examples

Require: A new example $\mathbf{d} \in \mathbb{R}^l$
  for $j = 1, \ldots, k$ **do**
    $\tilde{d}_j = \frac{1}{\sqrt{n_{i_j}}} \cdot \left(\mathbf{d}^\top \mathbf{d}_{i_j} - \sum_{t=1}^{j-1} \mathbf{d}_t \hat{\mathbf{D}}_{i_j,t} \right)$
  end for
  return the image $\tilde{\mathbf{d}}$ of $\mathbf{d}$ in the feature subspace

The Gram-Schmidt kernel (GSK) feature extraction algorithm (Cristianini et al., 2002) is based on the latent semantic indexing approach and it projects document feature vectors onto a subspace spanned by k representations of training examples in the feature space. The subspace is selected by applying the Gram-Schmidt orthogonalization procedure to documents in feature space. This subspace has lower dimension than the feature space, and at the same time it incorporates some semantic similarity between documents. In our algorithm, in order to construct the subspace we use a generalized GSK algorithm that has a bias towards positive examples (see Algorithms 1 and 2). The pseudo-code is taken from (Cristianini et al., 2002). Note that the dual of this algorithm is the partial Cholesky decomposition of the kernel or inner product matrix with elements $\mathbf{d}_i^\top \mathbf{d}_j$.

8.4 Numerical Results

We have implemented the above strategy for ranking a set of test documents with respect to their relevance to the training topic. The documents were ranked according to the value of $f_{t,\mathbf{E}}(\mathbf{d})$. The quality of the ranking was assessed using microaverage precision[1]. We used a MATLAB implementation of the algorithm based on SDPT3 semidefinite program solver package (Toh et al., 1996). The value of the bias B for the GSK algorithm was chosen to be equal to either the inverse fraction of positive examples in the category, or 10 whichever is smaller.

As benchmark data we used the 'Mod-Apte' split of the Reuters-21578 document collection available from the home page of David D. Lewis. The Mod-Apte sample contains 9603 training and 3299 test documents, and 90 categories.[2] We used the 10 most popular categories for which we computed SVM classification with a second order polynomial kernel using SVM$^{\text{light}}$ (Joachims, 1999) to compare our results with.

In Figure 8.1 we present our results for a different numbers of dimensions of the feature subspace for the category `acq`. We stopped increasing the value of dimensions n at $n = 30$ because of the computational complexity of the algorithm. The average CPU

[1]The microaverage precision is defined as the average of all precisions computed at the threshold $f_{t,\mathbf{E}}(\mathbf{d})$ where $\mathbf{d}$ ranges over all positive documents only. The precision at some threshold u is the fraction of positive documents among all documents for which $f_{t,\mathbf{E}}(\mathbf{d}) \geq u$.

[2]Bag-of-words vector respresentation of the document collection was done by Huma Lodhi.

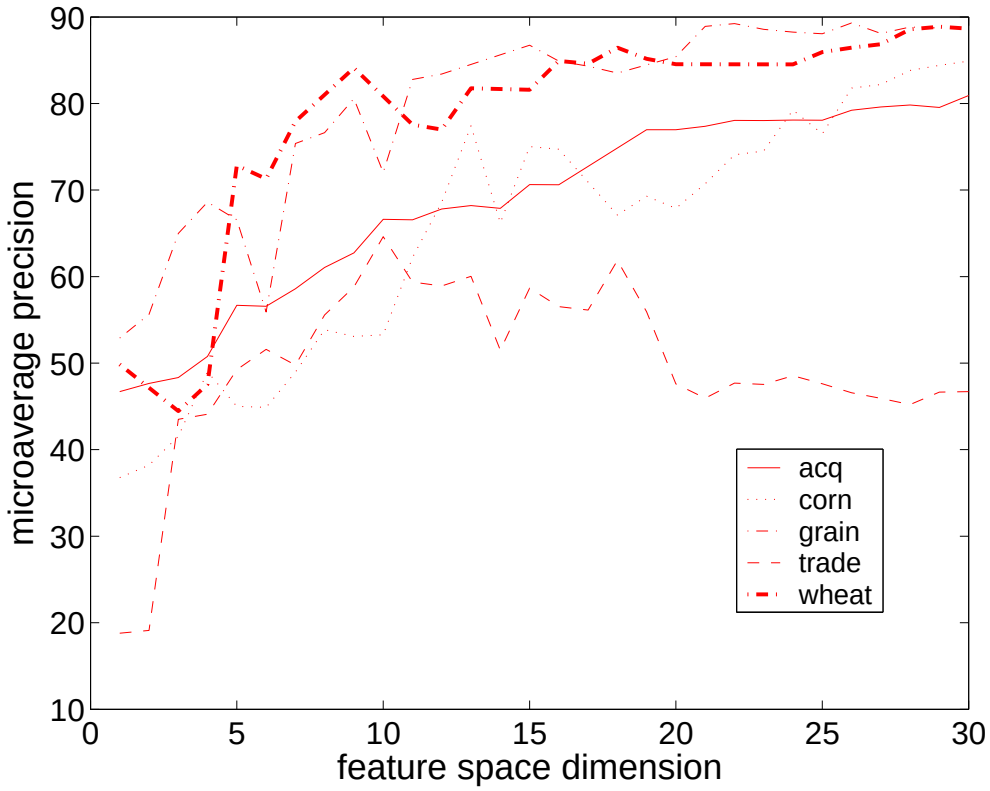


FIGURE 8.1: Choosing dimension of the feature space. The value of microaverage precision achieved with ellipsoid separation is plotted for different dimensions of the feature space for the categories `acq`, `corn`, `grain`, `trade` and `wheat`.

time of the algorithm performance on one category on a Pentium 4, 2.8GHz machine with 1 GB RAM was less than 15 minutes.

In Table 8.1 shows our results obtained with $n = 30$ dimensions of feature space in comparison to the best ones (after tuning the parameter to control the trade-off between the accuracy and the margin) obtained by soft-margin SVM classification with a second order polynomial kernel on the vector representation of the documents in the same space obtained by the GSK algorithm. As one can see we reached the accuracy of the SVM in one case, performed slightly worse in 6 cases and much worse on the `trade`, `interest` and `money-fx` categories. This is explained by the fact that our algorithm does not use slack variables to deal with noise in the data, and is not kernelized.

Figure 8.1 indicates that increasing the dimension a little has the potential to further improve the performance. This is not practical using our current implementation for complexity reasons. We anticipate, however, that using a chunking approach similar to the one adopted for the SVM, we should be able to scale the algorithm to higher dimensional feature spaces. The figure also shows that in some cases adding an extra dimension to the feature space may deteriorate the classification accuracy of the algorithm, therefore choosing the dimensionality of the feature space for the Gram-Schmidt

	Ellipsoid ($k = 30$)	SVM ($k = 30$)	SVM ($n = 20494$)
earn	**97.6**	**97.6**	99.7
acq	80.9	**85.0**	98.6
money-fx	58.9	**75.1**	80.9
grain	89.0	**94.8**	98.4
crude	82.1	**90.4**	96.1
trade	46.7	**80.3**	84.3
interest	56.0	**77.1**	87.1
ship	79.6	**84.5**	92.8
wheat	88.7	**93.9**	93.1
corn	84.9	**90.9**	92.2

TABLE 8.1: Comparison of the ellipsoid separation against SVM on ten categories of the Reuters-21578 dataset using microaverage precision (in percent) as a performance measure. The ellipsoid separation was done on $n = 30$ features extracted using the GSK algorithm (second column), and the SVM classification was performed on both: the same set of 30 features (third column) and the whole set of 20494 features (fourth column).

approximation of the latent semantic feature extraction is a separate task which needs some heuristics to be solved.

Chapter 9

Conclusions and Future Research

The problem of online learning a separating hyperplane was addressed in the first six chapters of this thesis. We considered the perceptron learning algorithms based on the subgradient descent method which provide a feasible solution in polynomial time (mainly, the ellipsoid algorithm and the probabilistic perceptron rescaling algorithm) and showed that their polynomiality is based on the same technique - the rescaling of the instance space. The ellipsoid algorithm uses this approach to dilate the space, while the perceptron algorithm with rescaling harnesses space contraction. In the ellipsoid algorithm the rescaling factor is chosen to optimize its performance, and we showed that the factor used by the probabilistic perceptron is also close to its optimal value. Comparing their computational complexities we conclude that the ellipsoid algorithm outperforms the probabilistic perceptron in the general case, but the latter algorithm is more efficient on large scale problems when the margin is large enough relatively to the cubic inverse of the dimension.

The generalized version of the probabilistic perceptron algorithm with rescaling (Dunagan and Vempala, 2008) was proposed further. It was suggested that the parameters of the algorithm may be varied to tailor the needs of particular problems. The convergence theorem of the algorithm was modified to allow to determine if the algorithm converges under any given set of the parameters. Two possible sets of parameters were suggested and tested in order to demonstrate the applicability of the modified theoretical framework. A potential way of decreasing the complexity of the algorithm via changing the powers of the dimension in the parameters of the algorithm was indicated. A parametric version of the modified perceptron algorithm (Blum et al., 1998) was also derived as a part of this research.

Aftertwards we turned our attention to the learning complexity of the online learning of a separating hyperplane. We conjectured the equivalence between this problem and a generic convex program over the unit ball with a Lipschitz continuous objective, and using some results from the information-based complexity theory of convex programming

suggested tight lower estimates of the learning complexity of our problem. It turned out that for the case of a fixed dimension when the margin is large enough the complexity of the generic perceptron algorithm is of the same order of magnitude as the complexity of the problem, therefore in this case its complexity cannot be improved.

It is an open question if the rescaling procedure can be utilized in the second order perceptron Cesa-Bianchi et al. (2005), which exploits spectral information of the sequence of vector inputs but is not polynomial-time in the general case, or in perceptron-based active learning Dasgupta et al. (2005). We are also interested in the comparison of our results on the learning complexity and the similar results on problem complexity of the probabilistic PAC online learning model Valiant (1984); Kearns and Vazirani (1994).

Following the recent extension of the perceptron rescaling algorithm to generic conic systems in Belloni et al. (2007), it would be promising to consider our parametric version of the algorithm in this framework, too. A deterministic version of the perceptron rescaling algorithm Barr (2007) would be an interesting baseline to compare performance the parametric algorithm against.

Semidefinite programming has interesting applications in machine learning. In turn, we have shown how a simple learning algorithm can be modified to solve higher order convex optimization problems such as semidefinite programs. Although the experimental results given here suggest the approach is far from computationally competitive, the insights gained may lead to effective algorithms in concrete applications in the same way that, for example, SMO is a competitive algorithm for solving quadratic programming problems arising from support vector machines. The positive definite perceptron algorithm excels at solving positive definite CSPs as found, e.g., in problems of transformation invariant pattern recognition as solved by Semidefinite Programming Machines Graepel and Herbrich (2004).

Among the algorithms for large-scale non-smooth convex optimization whose information-based complexity is close to the optimal we want to mention the bundle mirror approach Ben-Tal and Nemirovski (2005) and the smoothing technique Nesterov (2005) which can improve the solvability of a numerous machine learning problems and are awaiting for their new potential applications in machine learning.

Finally, we adapt the maximal separation ratio algorithm of Glineur (1998) for text categorization problem. The technique proposed is attractive theoretically in that it attempts to place an ellipsoid in feature space. Preliminary experiments are encouraging since they demonstrate that the algorithm can perform document classification up to the level of the state-of-the-art SVM algorithm. Further research is needed to investigate the method and its strengths and weaknesses, in particular using non-linear inner product functions (i.e., kernels) and introducing soft-margins similarly to SVMs (see Cortes and Vapnik (1995)). It will be of particular interest to test its performance on the categories

with very few positive examples. The low dimensionality and ellipsoid approach would appear to be suited to this type of problem.

Appendix A

Preliminaries on Semidefinite Programming

A.1 Brief Overview of Convex Analysis

A semidefinite program[1] (SDP) is a special case of a generic convex program (CP). Therefore we first take a quick glance onto convex programming on the whole. We need the following definitions of a convex set, a convex function, and a convex program, and some other notions associated with convex programming.

Definition A.1 (Convex Set)**.** A set $X \subseteq \mathbb{R}^n$ is called a *convex set* if

$$\forall x_1, x_2 \in X, \forall \lambda \in [0,1] : \lambda x_1 + (1-\lambda) x_2 \in X. \tag{A.1}$$

In other words, a set is convex if any two points of it can be connected with a straight line segment whose interior entirely belongs to this set. A single point, a ball, a line (and its segment), a hyperplane, a half-space, an ellipsoid, etc, are examples of convex sets. The theorem below suggests a possible way of constructing compound convex sets.

Theorem A.2 (Intersection of Convex Sets)**.** *If* $\forall i \in \{1, \ldots, m\} : X_i$ *is a convex set, then* $\bigcap_{i=1}^{m} X_i$ *is also a convex set.*

Proof. Consider any $x_1, x_2 \in X$ and any $\lambda \in [0,1]$. $\forall i \in \{1, \ldots, m\} : x_1, x_2 \in X_i$ since X is an intersection of X_i. Then $\forall i \in \{1, \ldots, m\} : \lambda x_1 + (1-\lambda) x_2 \in X_i$ since X_i are convex sets. Therefore $\forall i \in \{1, \ldots, m\} : \lambda x_1 + (1-\lambda) x_2 \in X$ as an intersection of X_i. □

[1] In the context of convex optimization and its subcases we use the term *programming* in the meaning *optimization* and the term *program* as a synonym to *problem* because of our respect to traditional mathematical terminology. Though we try to avoid ambiguity and to make meanings of those terms be clear from the context.

From the Theorem A.2 we conclude that a *polyhedron*

$$\begin{aligned} P &= \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{a}_i^\top \mathbf{x} \leq b_i,\, \mathbf{a}_i \in \mathbb{R}^n,\, b_i \in \mathbb{R},\, i = 1, \ldots, m\} \\ &= \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{A}\mathbf{x} \leq \mathbf{b},\, \mathbf{A} \in \mathbb{R}^{m \times n},\, \mathbf{b} \in \mathbb{R}^m\} \end{aligned} \tag{A.2}$$

is a convex set as an intersection of half-spaces. Moreover it is possible to prove that *every* convex set can be represented as an intersection of some (usually infinite) number of half-spaces (Boyd and Vandenberghe, 2004).

A particular class of convex sets that we will deal a lot with is a convex cone.

Definition A.3 (Convex Cone)**.** A set $X \subseteq \mathbb{R}^n$ is called a *convex cone* if

$$\forall \mathbf{x}, \mathbf{y} \in X \; \forall \lambda, \mu \geq 0 : \lambda\mathbf{x} + \mu\mathbf{y} \in X. \tag{A.3}$$

A cone C is called *solid* if its interior is not empty ($\operatorname{int} C \neq \emptyset$).

Geometrically a section of a disk of an infinite radius is an example of a convex cone. Among other examples there are a non-negative orthant $\mathbb{R}^n_+$, a set of all symmetric matrices in $\mathbb{R}^{m \times m}$ ($\mathbb{S}^m := \{\mathbf{A} \in \mathbb{R}^{m \times m} \,|\, \mathbf{A} = \mathbf{A}^\top\}$) and any linear subspace.

Definition A.4 (Convex Function)**.** A function $f(\mathbf{x}) : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ is called *convex function* if

$$\forall x_1, x_2 \in X, \forall \lambda \in [0, 1] : \lambda f(x_1) + (1 - \lambda) f(x_2) \geq f(\lambda x_1 + (1 - \lambda) x_2) \tag{A.4}$$

where $X \subseteq \mathbb{R}^n$ is a convex set.

A function $f(\mathbf{x}) : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ is *concave* if $-f(\mathbf{x})$ is convex.

Any linear function $\mathbf{c}^\top \mathbf{x}$, any affine function $\mathbf{a}^\top \mathbf{x} + b$ and any norm $\|\mathbf{x}\|$ (e.g. Euclidean norm $\|\mathbf{x}\| = \sqrt{\mathbf{x}^\top \mathbf{x}}$) are examples of convex functions. Furthermore, all linear and all affine functions are both convex and concave.

We define two types of sets associated with convex functions which we will need to introduce two criteria of convex functions.

Definition A.5 (Epigraph)**.** *Epigraph* of a function $f(\mathbf{x}) : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ is a set

$$\mathbf{epi} f = \{(\mathbf{x}, t) \in X \times \mathbb{R} \,|\, f(\mathbf{x}) \leq t\}. \tag{A.5}$$

In the 2-dimensional case epigraph is a set of points which lie above the plot of the function.

Theorem A.6 (Epigraph of convex function)**.** *Function $f(\mathbf{x}) : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ is convex if and only if* $\mathbf{epi} f$ *is a convex set.*

Definition A.7 (α-sublevel Set)**.** *α-sublevel set* of a function $f(\mathbf{x}) : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ is a set

$$C(\alpha) := \{\mathbf{x} \in X \,|\, f(\mathbf{x}) \leq \alpha\} \tag{A.6}$$

In other words, α-sublevel set is a set on which the function has values less, or equal to α.

Theorem A.8 (Sublevel Sets of Convex Function)**.** *If function $f(\mathbf{x}) : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ is convex then all its sublevel sets are convex sets.*

The inverse statement does not hold. For example, all sublevel sets of $f(x) = -e^x$ are convex, but the function is not (in fact, it is strictly concave)(Boyd and Vandenberghe, 2004). Now we are able to define a generic convex program.

Definition A.9 (Convex Program)**.** An optimization problem

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f_0(\mathbf{x}) \\ \text{subject to} \quad & f_i(\mathbf{x}) \leq 0, \quad i \in \{1,\dots,m\} \\ & \mathbf{a}_j^\top \mathbf{x} - b_j = 0, \quad j \in \{1,\dots,s\} \end{aligned} \tag{A.7}$$

where $\mathbf{a}_i \in \mathbb{R}^n, b_i \in \mathbb{R}, i \in \{1,\dots,s\}$ is called *convex program* if $f_i(\mathbf{x}), i \in \{0,\dots,m\}$ are convex functions.

A set $X := \{\mathbf{x} \in \mathbb{R}^n \,|\, f_i(\mathbf{x}) \leq 0,\, i = 1,\dots,m,\, \mathbf{a}_j^\top \mathbf{x} - b_j = 0,\, j = 1,\dots,s\}$ is called a *feasible set*, or a *solution set* of the convex program (A.7).

A generic convex program is a problem to minimize a convex function over a convex set. Indeed, a feasible set of problem (A.7) is a convex set as an intersection of m 0-sublevel sets of convex functions (defined by the inequality constraints), and s hyperplanes (defined by the equality constraints).

We call a convex program

- *feasible* if its feasible set is non-empty;
- *bounded* if it is either infeasible (in this case we consider its optimal value to be equal to $+\infty$), or its objective function is bounded below over all points from its feasible set.

A program which is both bounded and feasible is *solvable*, and therefore every point from its feasible set X is called *feasible solution*.

A program is *strictly feasible* if the interior of its feasible set is non-empty, i.e. at least one point $\mathbf{x} \in \mathbb{R}^n$ exists such that $\forall i \in \{1,\dots,m\} :\, f_i(\mathbf{x}) < 0$, $\forall j \in \{1,\dots,s\} :\, \mathbf{a}_j^\top \mathbf{x} - b_j = 0$.

Convex programs are classified according to the types of their objective and constraint functions $f_i(\mathbf{x}), i = \{0, \ldots, m\}$ which define their feasible sets. A typical example is a linear program (LP).

Definition A.10 (Linear Program)**.** A convex program in the form

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n_+} \quad & \mathbf{c}_0^\top \mathbf{x} - d_0 \\ \text{subject to} \quad & \mathbf{c}_i^\top \mathbf{x} - d_i \geq 0, \quad i \in \{1, \ldots, m\} \\ & \mathbf{a}_j^\top \mathbf{x} - b_j = 0, \quad j \in \{1, \ldots, s\} \end{aligned} \tag{A.8}$$

where functions $f_i(\mathbf{x}), i = \{0, \ldots, m\}$ are affine functions and the feasible set is a subset of non-negative orthant $\mathbb{R}^n_+ = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{x} \geq 0\}$ is a *linear program.*

Next we introduce a concept of local and global optimality in order to show the most attractive property of convex programming. Without losing generality we consider a definition of local and global minimum only since we can easily convert any maximization problem into minimization taking a negative of the objective function, and changing the direction of optimization from maximization to minimization.

Definition A.11 (Local and Global Minimum Points)**.** Given a function $f(\mathbf{x}) : X \subseteq \mathbb{R}^n \mapsto \mathbb{R}$ a point $\mathbf{x} \in \mathbb{R}^n$ is called

- a *local minimum point* of the function $f(\mathbf{x})$ on the set X if $\exists \epsilon > 0$ such that $\forall \mathbf{y} \in \{\mathbf{y} \in X \,|\, \|y - x\| \leq \epsilon\} : f(\mathbf{x}) \leq f(\mathbf{y})$;
- a *global minimum point* of the function $f(\mathbf{x})$ on the set X if $\forall \mathbf{y} \in X : f(\mathbf{x}) \leq f(\mathbf{y})$;

Theorem A.12. *Every local minimum solution to a convex program (A.7) is already its global minimum solution.*

Proof. Let us suppose that a point $\mathbf{x}_1 \in X \subseteq \mathbb{R}^n$ is a local minimum point of convex program (A.7) for some $\epsilon > 0$ where X is a feasible set of (A.7). Now assume that the point $\mathbf{x}_1$ is not a point of global minimum of (A.7), i.e. there is a point $\mathbf{y} \in X$: such that $\forall \mathbf{z} \in X : f_0(\mathbf{y}) \leq f_0(\mathbf{z})$ and $f_0(\mathbf{y}) < f_0(\mathbf{x}_1)$. Let us choose λ such that for a point $\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{y}$ holds $\|\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{y} - \mathbf{x}_1\| = \|(1-\lambda)\mathbf{y} - (1-\lambda)\mathbf{x}_1\| < \epsilon$ (i.e. $\lambda \geq 1 - \frac{\epsilon}{\|\mathbf{y}-\mathbf{x}_1\|}$). Then from the convexity of the function $f_0(\mathbf{x})$ we have

$$f_0(\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{y}) \leq \lambda f_0(\mathbf{x}_1) + (1-\lambda) f_0(\mathbf{y}) < \lambda f_0(\mathbf{x}_1) + (1-\lambda) f_0(\mathbf{x}_1) = f_0(\mathbf{x}_1)$$

which contradicts to the local minimality of $\mathbf{x}_1$. Therefore our assumption that there is a point $\mathbf{y} \in X$ such that $f_0(\mathbf{y}) < f_0(\mathbf{x})$ is false, hence the point $\mathbf{x}_1$ is a point of global minimum of (A.7). □

Note that the theorem does not state the uniqueness of the solution.

A.2 Convex Duality

One of the most important concepts in convex programming is *duality*.

Definition A.13 (Dual Cone)**.** Let C be a cone. A set

$$C^* = \{y \,|\, x^\top y \geq 0,\, \forall x \in C\}$$

is called the *dual cone* of C.

A cone C is called *self-dual* if its dual is C itself ($C = C^*$).

Now we introduce an unconstrained objective function for (A.7) called *Lagrange function*.

Definition A.14 (Lagrange Function, or Lagrangian)**.** A function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m_+ \times \mathbb{R}^s \mapsto \mathbb{R}$ defined as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) := f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{j=1}^{s} \nu_j (\mathbf{a}_j^\top \mathbf{x} - b_j) \tag{A.9}$$

is called the *Lagrange function*, or the *Lagrangian* corresponding to the problem (A.7), and $(\boldsymbol{\lambda}^\top, \nu^\top)^\top \in \mathbb{R}^m_+ \times \mathbb{R}^s$ is called a vector of *Lagrange multipliers*, or *dual variables*.

We maximize the Lagrangian (A.9) and define

$$g(\mathbf{x}) := \max_{\boldsymbol{\lambda} \in \mathbb{R}^m_+, \boldsymbol{\nu} \in \mathbb{R}^s} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \begin{cases} f_0(\mathbf{x}) & f_i(\mathbf{x}) \leq 0,\, i = 1, \ldots, m, \\ & \mathbf{a}_j^\top \mathbf{x} - b_j = 0,\, j = 1, \ldots, s \\ +\infty & \text{otherwise} \end{cases} \tag{A.10}$$

which is equal to the objective function of the primal convex problem (A.7) on the feasible set of the latter, and $+\infty$ outside therefore (A.10) is also called the *barrier function* of the problem (A.7). Let us denote the optimal value of (A.7) by $p^\star$,

$$p^\star := \inf_{\mathbf{x} \in \mathbb{R}^n} \{f_0(\mathbf{x}) \,|\, f_i(\mathbf{x}) \leq 0,\, i = 1, \ldots, m,\, \mathbf{a}_j^\top \mathbf{x} - b_j = 0,\, j = 1, \ldots, s\}. \tag{A.11}$$

In other words $p^\star$ is a lower bound on (A.7), and if we pose the latter as an *unconstrained* optimization problem using its Lagrangian then we get

$$p^\star \leq \min_{\mathbf{x} \in \mathbb{R}^n} \max_{\boldsymbol{\lambda} \in \mathbb{R}^m_+, \boldsymbol{\nu} \in \mathbb{R}^s} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}). \tag{A.12}$$

Now by just swapping minimization and maximization in (A.12) we can formulate a *dual problem* to (A.7)

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^m_+, \boldsymbol{\nu} \in \mathbb{R}^s} \min_{\mathbf{x} \in \mathbb{R}^n} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}). \tag{A.13}$$

From now we refer to (A.7) as the *primal problem* to (A.13).

Convex duality theory tells us the lower bound on $p^\star$. The weak duality theorem states that it can be found via optimizing (A.13). If we denote the optimal value of (A.13) with $d^\star$

$$d^\star := \sup_{\boldsymbol{\lambda}\in\mathbb{R}^m_+,\boldsymbol{\nu}\in\mathbb{R}^s} \min_{\mathbf{x}\in\mathbb{R}^n} \mathcal{L}(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\nu}), \tag{A.14}$$

then the following theorem holds.

Theorem A.15 (Weak Duality)**.** *The optimal value of the dual problem (A.13) always gives a lower bound on the optimal value of the primal problem (A.7), i. e.* $d^\star \leq p^\star$.

The objective function ($\min_{\mathbf{x}\in\mathbb{R}^n} \mathcal{L}(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\nu})$) of the dual problem (A.13) is concave, and if the primal problem is unbounded, then the optimal value of the dual problem is equal to $-\infty$. Moreover, weak duality also holds if the primal problem (A.7) is not convex, and even in this case the objective function of the dual problem is still concave.

The weak duality theorem also allows us to introduce a certificate of optimality of possible solutions to (A.7) and (A.13) called a duality gap.

Definition A.16 (Duality Gap)**.** If we have some solution $\hat{\mathbf{x}}$ to the primal problem (A.7) and some solution $(\hat{\boldsymbol{\lambda}}^\top, \hat{\boldsymbol{\nu}}^\top)^\top$ to the dual problem (A.13), and corresponding values of (A.7) and (A.13) are $\hat{p}$ and $\hat{d}$ respectively, then the following difference

$$\eta(\hat{\mathbf{x}}, (\hat{\boldsymbol{\lambda}}^\top, \hat{\boldsymbol{\nu}}^\top)^\top) := f_0(\hat{\mathbf{x}}) - \min_{\mathbf{x}\in\mathbb{R}^n} \mathcal{L}(\mathbf{x},\hat{\boldsymbol{\lambda}},\hat{\boldsymbol{\nu}}) = \hat{p}^\star - \hat{d}^\star \geq 0 \tag{A.15}$$

is called the *duality gap* associated with $\hat{\mathbf{x}}$ and $(\hat{\boldsymbol{\lambda}}^\top, \hat{\boldsymbol{\nu}}^\top)^\top$.

If $p^\star = d^\star$ then it is said that *strong duality* holds. The following theorem states a sufficient condition of strong duality for convex programs.

Theorem A.17 (Strong Convex Duality)**.** *If the convex program (A.7) is strictly feasible then strong duality holds, i. e.* $p^\star = d^\star$.

The condition of this theorem is often referred to as a *Slater condition*. Note that the Slater condition is not sufficient for a convex program to attain its optimal value. In order to achieve the latter it either also has to be bounded (and therefore solvable), or the Slater condition has to hold for its dual as well, so it has to be strictly feasible, too (Boyd and Vandenberghe, 2004). In other words, either $p^\star \in \mathbb{R}$, or $d^\star \in \mathbb{R}$ must hold.

A.3 A Generic Semidefinite Program

First we recall some definitions and facts from Linear Algebra.

Definition A.18 (Positive Definite and Semidefinite Matrix)**.** A symmetric matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is called

- *positive definite* ($\mathbf{A} \succ 0$) if $\forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{0} : \mathbf{x}^\top \mathbf{A}\mathbf{x} > 0$
- *positive semidefinite* ($\mathbf{A} \succeq 0$) if $\forall \mathbf{x} \in \mathbb{R}^n : \mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0$
- *negative definite* ($\mathbf{A} \prec 0$) if $\forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{0} : \mathbf{x}^\top \mathbf{A}\mathbf{x} < 0$ (i. e. $-\mathbf{A} \succ 0$)
- *negative semidefinite* ($\mathbf{A} \preceq 0$) if $\forall \mathbf{x} \in \mathbb{R}^n : \mathbf{x}^\top \mathbf{A}\mathbf{x} \leq 0$ (i. e. $-\mathbf{A} \succeq 0$).

Let us denote a set of all positive semidefinite matrices in $\mathbb{R}^{n\times n}$ with $\mathbb{S}^n_+$, and a set of all positive definite matrices in $\mathbb{R}^{n\times n}$ with $\mathbb{S}^n_{++}$. It is easy to see that $\mathbb{S}^n_+$ is a convex self-dual cone in $\mathbb{R}^{n\times n}$ and $\mathbb{S}^n_{++}$ is its interior ($\mathbb{S}^n_{++}$ is not a cone itself because it does not contain a zero element).

We define an inner product in $\mathbb{R}^{n\times n}$ as Frobenius (component-wise) inner product

$$\langle \mathbf{X}, \mathbf{Y} \rangle_F = \mathbf{X} \bullet \mathbf{Y} = \operatorname{tr}(\mathbf{X}^\top \mathbf{Y}) = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{X}_{ij}\mathbf{Y}_{ij}, \tag{A.16}$$

where a trace of a matrix is a sum of its diagonal elements ($\operatorname{tr} \mathbf{A} := \sum_{i=1}^n a_{ii}$). The inner product (A.16) induces a norm in $\mathbb{R}^{n\times n}$

Definition A.19 (Frobenius norm)**.**

$$\|\mathbf{X}\|_F = \sqrt{\mathbf{X} \bullet \mathbf{X}} = \sqrt{\operatorname{tr}(\mathbf{X}^\top \mathbf{X})} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{X}_{ij}^2}. \tag{A.17}$$

It is trivial to check that the norm (A.17) satisfies all axioms of norm.

Using Frobenius inner product we can state the following criterion of a positive definite (semidefinite) matrix.

Theorem A.20. *A matrix* $\mathbf{A}$ *is positive definite (semidefinite) if and only if*

$$\forall x \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{0} : \mathbf{x}\mathbf{x}^\top \bullet \mathbf{A} > 0 \, (\mathbf{x}\mathbf{x}^\top \bullet \mathbf{A} \geq 0).$$

Proof.

$$\mathbf{x}\mathbf{x}^\top \bullet \mathbf{A} = \operatorname{tr}(\mathbf{x}\mathbf{x}^\top \mathbf{A}) = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{x}_i \mathbf{x}_j \mathbf{A}_{ij} = \mathbf{x}^\top \mathbf{A}\mathbf{x}$$

□

Theorem A.21 (Eigenvalues of Positive Definite (Semidefinite) Matrix)**.** *Let* $\lambda_1, \ldots, \lambda_n$ *be the eigenvalues of a symmetric matrix* $\mathbf{A} \in \mathbb{S}^n$*. Then* $\mathbf{A} \succ 0$ *(*$\mathbf{A} \succeq 0$*) if and only if* $\forall i \in \{1, \ldots, n\} : \lambda_i > 0$ *(*$\lambda_i \geq 0$*).*

We denote $\mathbf{A} \succeq \mathbf{B}$ ($\mathbf{A} \succ \mathbf{B}$) if $\mathbf{A} - \mathbf{B} \succeq 0$ ($\mathbf{A} - \mathbf{B} \succ 0$) so one can see that the binary relation $\succ$ is a partial order relation over $\mathbb{S}^n_+$.

A practical criterion of positive (semi-)definiteness of a compound matrix is given in the following theorem

Theorem A.22 (Schur complement)**.**

$$\textit{Let} \quad \mathbf{D} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{pmatrix} \quad \textit{with} \quad \mathbf{A} \in \mathbb{S}^m_{++} \quad \textit{and} \quad \mathbf{C} \in \mathbb{S}^n \quad \textit{then}$$

$$\mathbf{D} \succ 0 \quad (\mathbf{D} \succeq 0) \qquad \Longleftrightarrow \qquad \mathbf{C} - \mathbf{B}^\top \mathbf{A}^{-1} \mathbf{B} \succ 0 \quad (\mathbf{C} - \mathbf{B}^\top \mathbf{A}^{-1} \mathbf{B} \succeq 0).$$

The matrix $\mathbf{C} - \mathbf{B}^\top \mathbf{A}^{-1} \mathbf{B}$ is called the *Schur complement* of $\mathbf{A}$ in $\mathbf{D}$.

Note that given some set of symmetric matrices $\mathbf{F}_0, \mathbf{F}_1, \ldots, \mathbf{F}_n \in \mathbb{S}^n$ we can define some subspace in $\mathbb{S}^n_+$ as an affine combination of these matrices via *linear matrix inequality* (LMI)

$$\mathbf{F}(\mathbf{x}) := \mathbf{F}_0 + x_1 \mathbf{F}_1 + \ldots + x_m \mathbf{F}_m \succeq 0, \tag{A.18}$$

where $\mathbf{x} \in \mathbb{R}^m$.

Now we are able to define a generic semidefinite program. We consider a semidefinite program in the following form (Vandenberghe and Boyd, 1996):

Definition A.23 (Semidefinite Program)**.** A *semidefinite program* is given by

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^m} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{F}(\mathbf{x}) = \mathbf{F}_0 + \sum_{i=1}^m \mathbf{F}_i x_i \succeq 0\,, \end{aligned} \tag{A.19}$$

with $\mathbf{c} \in \mathbb{R}^m$ and for all $i \in \{0, \ldots, m\}$ we have $\mathbf{F}_i \in \mathbb{S}^n$.

As one can see from the definition, semidefinite programming is optimization of a linear function under a positive semidefinite constraint. In the sequel we will show that this constraint can be transformed into an infinite number of linear constraints.

A.4 Primal-Dual Formulation of an SDP

A Lagrangian for our primal SDP (A.19) is

$$\mathcal{L}(\mathbf{x}, \mathbf{Z}) := \mathbf{c}^\top \mathbf{x} - \operatorname{tr} \mathbf{Z}\mathbf{F}(\mathbf{x}). \tag{A.20}$$

where the matrix $\mathbf{Z} \in \mathbb{S}^n_+$ is a dual variable since $\mathbb{S}^n_+$ is self-dual cone. Indeed, if we maximize the Lagrangian (A.20) over $\mathbb{S}^n_+$ then we get a barrier function for the primal SDP (A.19)

$$g(\mathbf{x}) := \max_{\mathbf{Z} \in \mathbb{S}^m_+} \mathcal{L}(\mathbf{x}, \mathbf{Z}) = \begin{cases} \mathbf{c}^\top \mathbf{x} & \mathbf{F}(\mathbf{x}) \succeq 0, \\ +\infty & \text{otherwise} \end{cases} \tag{A.21}$$

In analogy to generic convex duality we can pose (A.19) as an unconstrained optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} \max_{\mathbf{Z} \in \mathbb{S}^m_+} \mathcal{L}(\mathbf{x}, \mathbf{Z}) = \min_{\mathbf{x} \in \mathbb{R}^n} \max_{\mathbf{Z} \in \mathbb{S}^m_+} \mathbf{c}^\top \mathbf{x} - \operatorname{tr} \mathbf{F}(\mathbf{x}) \mathbf{Z}, \tag{A.22}$$

and also obtain the dual problem

$$\max_{\mathbf{Z} \in \mathbb{S}^m_+} \min_{\mathbf{x} \in \mathbb{R}^n} \mathbf{c}^\top \mathbf{x} - \operatorname{tr} \mathbf{F} \mathbf{Z}(\mathbf{x}). \tag{A.23}$$

Expanding the dual problem to

$$\min_{\mathbf{x} \in \mathbb{R}^n} \max_{\mathbf{Z} \in \mathbb{S}^m_+} \mathbf{c}^\top \mathbf{x} - \operatorname{tr} \mathbf{F}_0 \mathbf{Z} - \sum_{i=1}^n x_i \operatorname{tr} \mathbf{F}_i \mathbf{Z},$$

we get its barrier function

$$\begin{aligned} g(\mathbf{Z}) := \min_{\mathbf{x} \in \mathbb{R}^n} \mathcal{L}(\mathbf{x}, \mathbf{Z}) &= \min_{\mathbf{x} \in \mathbb{R}^n} (\mathbf{c}^\top \mathbf{x} - \operatorname{tr} \mathbf{F}_0 \mathbf{Z} - \sum_{i=1}^n x_i \operatorname{tr} \mathbf{F}_i \mathbf{Z}) \\ &= \begin{cases} \operatorname{tr} \mathbf{F}_0 \mathbf{Z} & c_i = \operatorname{tr} \mathbf{F}_i \mathbf{Z},\ i = 1, \ldots, n \\ -\infty & \text{otherwise} \end{cases}, \end{aligned} \tag{A.24}$$

then we can re-write the dual problem to the SDP (A.19) in its standard form

$$\begin{aligned} \max_{\mathbf{Z} \in \mathbb{S}^m_+} \quad & -\operatorname{tr} \mathbf{F}_0 \mathbf{Z} \\ \text{subject to} \quad & \operatorname{tr} \mathbf{F}_i \mathbf{Z} = c_i, \quad i = 1, \ldots, n, \end{aligned} \tag{A.25}$$

It can be shown that the dual SDP can be represented in the same form as primal (Vandenberghe and Boyd, 1996) since semidefinite programming duality is symmetric because a cone of positive semidefinite matrices $\mathbb{S}^m_+$ is self-dual.

Any solution $\mathbf{Z}$ to the dual SDP (A.25) gives a lower bound on the optimal value of the primal problem (A.19). Therefore for every pair of feasible solutions $\mathbf{x}$, $\mathbf{Z}$ to primal and dual SDP respectively we can define the duality gap corresponding to these solutions as

$$\eta(\mathbf{x}, \mathbf{Z}) := \mathbf{c}^\top \mathbf{x} + \operatorname{tr} \mathbf{F}_0 \mathbf{Z} = \operatorname{tr} \mathbf{F}(\mathbf{x}) \mathbf{Z} \geq 0. \tag{A.26}$$

For the sake of algorithmic purposes it is often more efficient to optimize a duality gap of an SDP, in other words to solve a *primal-dual semidefinite program*

$$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^n, \mathbf{Z} \in \mathbb{S}^m_+} \quad & \mathbf{c}^\top \mathbf{x} + \operatorname{tr} \mathbf{F}_0 \mathbf{Z} \\ \text{subject to} \quad & \mathbf{F}(\mathbf{x}) \succeq 0, \\ & \operatorname{tr} \mathbf{F}_i \mathbf{Z} = c_i, \quad i = 1, \ldots, n. \end{aligned} \tag{A.27}$$

A.5 Duality Theory for Semidefinite Programming

Let us denote the optimal value of the SDP (A.19) with p_*, i.e.

$$p_* := \inf\{\mathbf{c}^\top \mathbf{x} \,|\, \mathbf{F}(\mathbf{x}) \succeq 0,\, \mathbf{x} \in \mathbb{R}^n\} \tag{A.28}$$

and the optimal value of the dual SDP with d_* (A.25)

$$d_* := \sup\{-\operatorname{tr} \mathbf{F}_0 \mathbf{Z} \,|\, \operatorname{tr} \mathbf{F}_i \mathbf{Z} = c_i,\, i = 1, \ldots, n\}. \tag{A.29}$$

Then the following theorem holds

Theorem A.24 (Nesterov and Nemirovskii (1994))**.** *$p_* = d_*$ if either of the following conditions hold:*

1. *The primal SDP (A.19) is strictly feasible.*
2. *The dual SDP (A.25) is strictly feasible.*

If both conditions hold than the optimal values of both primal and dual programs are attained.

Bibliography

Farid Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

Martin Anthony and John Shawe-Taylor. Using the perceptron algorithm to find consistent hypothesis. *Combinatorics, Probability, and Computing*, 2:385–387, 1993.

Olof Barr. *Discrete methods used in graph theory and linear programming.* PhD thesis, Centre for Mathematical Sciences, Lund University, Lund, Sweden, 2007.

Alexandre Belloni, Robert M. Freund, and Santosh S. Vempala. An efficient re-scaled perceptron algorithm for conic systems. In Nader H. Bshouty and Claudio Gentile, editors, *Learning Theory, 20th Annual Conference on Learning Theory, COLT 2007, San Diego, CA, USA; June 13-15, 2007. Proceedings*, volume 4539 of *Lecture Notes in Computer Science*, pages 393–408. Springer, 2007.

Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, volume 1 of *MPS-SIAM Series on Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, 2001.

Aharon Ben-Tal and Arkadi Nemirovski. Non-euclidean restricted memory level method for large-scale convex optimization. *Mathematical Programming*, 102(3):407–456, 2005.

Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

Herbert D. Block. The perceptron: A model for brain functioning. *Reviews of Modern Physics*, 34:123–135, 1962.

Avrim Blum, Alan Frieze, Ravi Kannan, and Santosh Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1-2):35–52, 1998.

Brian Borchers. SDPLIB 1.2, A library of semidefinite programming test problems. *Optimization Methods and Software*, 11(1):683–690, 1999.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on the Computational Learning Theory*, pages 144–152. ACM Press, 1992.

Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics, 1994.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Arne Brøndsted. *An Introduction to Convex Polytopes*, volume 90 of *Graduate Texts in Mathematics*. Springer-Verlag, 1983.

Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. A second-order perceptron algorithm. *SIAM Journal of Computing*, 34(3):640–668, 2005.

Vladimir Cherkassky and Filip Mulier. *Learning from Data: Concepts, Theory and Methods*. Wiley-Interscience, 1998.

Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, December 1952.

Robert M. Corless, Gaston H. Gonnet, D. E. G. Hare, David J. Jeffrey, and Donald E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5: 329–359, 1996.

Corinna Cortes and Vladimir Vapnik. Support vector networks. *Machine Learning*, 20: 273–297, 1995.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Methods*. Cambridge University Press, 2000.

Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. Latent semantic kernels. *Journal of Intelligent Information Systems*, 18(2/3):127–152, 2002.

Sanjoy Dasgupta, Adam T. Kalai, and Claire Monteleoni. Analysis of perceptron-based active learning. In Peter Auer and Ron Meir, editors, *Learning Theory, 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005, Proceedings*, volume 3559 of *Lecture Notes in Computer Science*, pages 249–263. Springer, 2005.

Tijl De Bie and Nello Cristianini. Convex methods for transduction. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, New York, second edition, 2001.

John Dunagan and Santosh Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. *Mathematical Programming, Series A*, 114(1):101–114, July 2008.

Roger Fletcher. *Practical Methods of Optimization*. Wiley-Interscience, Chichester, second edition, 1987. ISBN 0-471-91547-4.

Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, 1990.

François Glineur. Pattern separation via ellipsoids and conic programming. Mémoire de D.E.A., Faculté Polytechnique de Mons, Mons, Belgium, September 1998.

Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of Association for Computing Machinery*, 42(6):1115–1145, 1995.

Thore Graepel. Kernel matrix completion by semidefinite programming. In José R. Dorronsoro, editor, *Proceedings of the International Conference on Neural Networks, ICANN2002*, Lecture Notes in Computer Science, pages 694–699. Springer, 2002.

Thore Graepel and Ralf Herbrich. Invariant pattern recognition by Semidefinite Programming Machines. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, second corrected edition, 1988.

Christoph Helmberg. Semidefinite programming for combinatorial optimization. Technical Report ZR-00-34, Konrad-Zuse-Zentrum für Informationstechnik Berlin, October 2000.

Ralf Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, 2002.

Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms I*, volume 305 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1993.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1991.

Danchi Jiang and Jun Wang. A recurrent neural network for real-time semidefinite programming. *IEEE Transactions on Neural Networks*, 10(1):81–93, January 1999.

Thorsten Joachims. Making large-scale support vector machine learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alex J. Smola, editors, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1999.

Fritz John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to Richard Courant on his 60th Birthday, January 8, 1948*, pages 187–204. Interscience Publishers, New York, 1948.

Jaz Kandola, Thore Graepel, and John Shawe-Taylor. Reducing kernel matrix diagonal dominance using semi-definite programming. In *Proceedings of 16th Annual Conference on Learning Theory*, volume 2777 of *Lecture Notes in Artificial Intelligence*, pages 288–302. Springer-Verlag, 2003.

Narendra Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. The MIT Press, 1994.

Leonid G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

Krzysztof C. Kiwiel. Convergence of approximate and incremental subgradient methods for convex optimization. *SIAM Journal on Optimization*, 14:807–840, 2004.

Gert R. G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael Jordan. Learning the kernel matrix with positive semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

Jean B. Lasserre. Linear programming with positive semi-definite matrices. Technical Report LAAS-94099, Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS, 1995.

A. Yu. Levin. On an algorithm for minimization of convex functions. *Soviet Mathematics Doklady*, 6:286–290, 1965.

Yaoyong Li, Hugo Zaragoza, Ralf Herbrich, John Shawe-Taylor, and Jaz Kandola. The perceptron algorithm with uneven margins. In *Proceedings of the International Conference of Machine Learning (ICML'2002)*, pages 379–386, 2002.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

Wolfgang Maass and György Turán. How fast can a threshold gate learn? In Stephen J. Hanson, George A. Drastal, and Ronald L. Rivest, editors, *Computational Learning Theory and Natural Learning Systems - Volume I: Constraints and Prospects*, pages 381–414. MIT Press, 1994. ISBN 0-262-58126-4.

Olvi L. Mangasarian and David R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10(5):1032, September 1999.

Warren S. McCulloch and Walter H. Pitts. A logical claculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

Marvin L. Minsky and Seymour A. Papert. *Perceptrons*. MIT Press, 1969.

Marvin L. Minsky and Seymour A. Papert. *Perceptrons: an Introduction to Computational Geometry*. MIT Press, expanded edition, 1988.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

A. S. Nemirovski and E. I. Nenakhov. "Multistep" method of ellipsoids. *Ekonomika i Matematicheskie Metody*, 27(6):1115–1120, 1991. (In Russian). English translation: *Matekon: Translations of Russian and East European Mathematical Economics*.

Arkadi Nemirovski. Five lectures on modern convex optimization, 2002. CORE Summer School on Modern Convex Optimization, August 26-30, 2002.

Arkadii S. Nemirovsky and David B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons, 1983. ISBN 0-471-10345-4.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, 2004.

Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

Yurii E. Nesterov and Arkadii S. Nemirovskii. Optimization over positive semidefinite matrices. Technical report, Central Economic and Mathematical Institute of USSR Academy of Sciences, Moscow, 1990a.

Yurii E. Nesterov and Arkadii S. Nemirovskii. Self-concordant functions and polynomial timemethods in convex programming. Technical report, Central Economic and Mathematical Institute of USSR Academy of Sciences, Moscow, 1990b.

Yurii E. Nesterov and Arkadii S. Nemirovskii. *Interior Point Methods in Convex Programming – Theory and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, 1994.

Donald J. Newman. Location of the maximum on unimodal surfaces. *Journal of Association for Computing Machinery*, 12(3):395–398, July 1965.

A. B. J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622. Polytechnic Institute of Brooklyn, 1962.

Gábor Pataki. Cone-LP's and semi-definite programs: facial structure, basic solutions, and the symplex method. Technical Report GSIA, Carnegie Mellon University, 1995.

Motakuri Ramana and A. J. Goldman. Some geometric results in semidefinite programming. *Journal of Global Optimization*, 7(1):33–50, 1995.

James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40(1):59–93, 1988.

Ralph T. Rockafellar. *Convex Analysis*. Princeton University Press, New Jersey, 1970.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

David E. Rumelhart and James L. McClelland. *Parallel Distributed Processing*. MIT Press, 1986.

Peter Sanders. Randomized priority queues for fast parallel access. *Journal of Parallel and Distributed Computing*, 49(1):86–97, 1998.

Robert J. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches*. John Wiley & Sons, 1992.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods and Pattern Analysis*. Cambridge University Press, 2004. ISBN 0-521-81397-2.

Naum Z. Shor. Convergence rate of the gradient descent method with dilatation of the space. *Cybernetics*, 6(2):102–108, 1970a.

Naum Z. Shor. Utilization of the operation of space dilatation in the minimization of convex functions. *Cybernetics*, 6(1):7–15, 1970b.

Naum Z. Shor. Cut-off method with space extension in convex programming problems. *Cybernetics*, 13(1):94–96, 1977.

Naum Z. Shor. *Minimization Methods for Non-Differentiable Functions*, volume 3 of *Springer Series in Computational Mathematics*. Springer-Verlag, 1985.

Naum Z. Shor and Vladimir I. Gershovich. Method of ellipsoids, its generalizations and applications. *Cybernetics*, 18(5):606–617, 1982.

Peng Sun and Robert M. Freund. Computation of minimum volume covering ellipsoids. *Operations Research Letters*, 52(5):690–706, 2004.

Michael J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001.

Kim-Chuan Toh, Michael J. Todd, and Reha H. Tütüncü. SDPT3 – a MATLAB software package for semidefinite programming. Technical Report TR1177, Cornell University, 1996.

Joseph F. Traub and Henryk Wożniakowski. Complexity of linear programming. *Operations Research Letters*, 1(2):59–62, 1982.

Pravin M. Vaidya. A new algorithm for minimizing convex functions over convex sets. *Mathematical Programming*, 73(3):291–341, 1996.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 17(11): 1134–1142, 1984.

Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38 (1):49–95, 1996.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

Stephen J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

Boris Yamnitsky and Leonid A. Levin. An old linear programming algorithm runs in polynomial time. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 327–328, New York, 1982. IEEE.

David B. Yudin and Arkadii S. Nemirovski. Evaluation of the informational complexity of mathematical programming problems. *Ekonomika i Matematicheskie Metody*, 12(1): 128–142, 1976a. (In Russian). English translation: *Matekon: Translations of Russian and East European Mathematical Economics*, 13(2):3-25, 1976-7.

David B. Yudin and Arkadii S. Nemirovski. Informational complexity and efficient methods for the solution of convex extremal problems. *Ekonomika i Matematicheskie Metody*, 12(2):357–369, 1976b. (In Russian). English translation: *Matekon: Translations of Russian and East European Mathematical Economics*, 13(3):25-45, 1977.