SUPPORT VECTOR MACHINES VIA ADVANCED OPTIMIZATION TECHNIQUES

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Operations Research and System Analysis

Eitan Rubinstein

SUBMITTED TO THE SENATE OF THE TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY

HESHVAN 5766 HAIFA NOVEMBER 2005

The research thesis was done under the supervision of Prof. A. S. Nemirovski and Dr. M. Zibulevsky in the department of Electrical Engineering.
I thank Prof. A. S. Nemirovski for his help and great effort, and I thank Dr. M. Zibulevsky for his help during the final stages of the thesis. The generous financial help of the Technion is gratefully acknowledged.

Abstract

The most basic problem considered in Machine Learning is the *supervised binary data classification*, where one is given a *training sample* – a random sample (x_i, y_i), $i = 1, ..., \ell$ of *examples* – *attribute vectors* x_i equipped with *labels* $y_i = \pm 1$ – drawn from unknown distribution P, and the goal is to build, based on this sample, a *classifier* – a function $f(x)$ taking values ± 1 such that the *generalization error* $\mathrm{Prob}_P\{(x, y) : f(x) \neq y\}$ is as small as possible. This general problem has numerous applications in classification of documents, texts and images, in computerized medical diagnostic systems, etc.

The SVM approach is the most frequently used technique for solving the outlined problem (same as other classification problems arising in Machine Learning). With this approach the classifier is given by an optimal solution to a specific convex optimization problem. While the theoretical background of SVM is given by a well-developed and deep Statistical Learning Theory, the computational tools used traditionally in the SVM context (that is, numerical techniques for solving the resulting convex programs) are not exactly state-of-the-art of modern Convex Optimization, especially when the underlying data sets are very large (tens of thousands of examples with high-dimensional attribute vectors in the training sample). In the large-scale case, one cannot use the most advanced, in terms of the rate of convergence, convex optimization techniques (Interior Point methods), since the cost of an iteration in such a method (which grows nonlinearly with the sizes of the training data) becomes too large to be practical. At the same time, from the purely optimization viewpoint, the "computationally cheap" optimization methods routinely used in the large-scale SVM's seem to be somehow obsolete.

The goal of the thesis is to investigate the potential of recent advances in "computationally cheap" techniques for extremely large-scale convex optimization in the SVM context. Specifically, we intend to utilize here the *Mirror Prox* algorithm (A. Nemirovski, 2004). The research will focus on (a) representing the SVM optimization programs in the specific *saddle point* form required by the algorithm, (b) adjusting the algorithm to the structure of the SVM problems, and (c) software implementation and testing the algorithm on large-scale SVM data, with the ultimate goal to develop a novel theoretically solid and computationally efficient optimization techniques for SVM-based supervised binary classification.

Notation

$\mathbf{R}$	Set of real numbers
S	Training sample
ℓ	Training sample size
x	Attribute vectors
y	Vector of labels
X	A set (usually a subset of a certain R^n)
$f(x), \phi(x)$	Classifier function
$\mathcal{F}$	Family of real valued functions (classifiers)
γ	The margin of a classifier
D	Distribution function
$\langle x, y \rangle$	Inner product
E	Hilbert space
E^*	A space dual to E
$\|x\|, \phi()$	Norm
$\|x\|_*$	Dual norm
ξ	Slack vector
$K(x, y)$	Kernel function
$O()$	Complexity
t	Iteration number
R	radius of the ball containing the data
L	Lipschitz constant
C, ρ	Constant parameters
c_π, r, k, p	Positive reals
N	Dimension of feature space
M	A certain set
Q	Matrix
η	vector of primal variables (α or w)
$\mathcal{N}(\mathcal{F}, \ell, \gamma)$	*ell*-covering number of $\mathcal{F}$
$\text{fat}_\mathcal{F}(\gamma)$	Fat shattering dimension
$\omega(z)$	distance-generating function
$\omega_z(\zeta)$	local distance
$D_{z_0}[Z]$	Bregman diameter
$P_z(\xi)$	Prox mapping
Ω	Complexity parameter
δ	Confidence

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we present a brief overview on Support Vector Machines along with outline and motivation of the goals of our research, and summarize the results of the Thesis.

1.1 Supervised Binary Classification via Support Vector Machines

1.1.1 Supervised Binary Data Classification

Supervised Binary Classification is the most basic problem of Machine Learning. In this problem, one is given a *training sample* S comprised of ℓ *examples* (x_i, y_i), $i = 1, ..., \ell$, where the *feature vectors* x_i are points of a given set X (usually, a subset of certain $\mathbf{R}^n$), and the *labels* y_i are either $+1$, or -1. It is assumed that this sample is picked at random from an *unknown* distribution $\mathcal{D}$ on $X \times \{-1; 1\}$, and the goal is to build a *classifier* capable to predict well the label y of a tentative example via the attribute vector x of this example. Formally speaking, a classifier is a function $\phi(\cdot)$ on X taking values in $\{-1; 1\}$, and the quality of such a classifier is quantified by its *generalization error*

$$\mathrm{err}_{\mathcal{D}}(\phi) = \mathrm{Prob}_{\mathcal{D}}\left\{(x, y) : \phi(x) \neq y\right\};$$

the less is the error, the better.

In applications, the attribute (sometimes called *input*) vectors x usually are collections of numerical or categorical measurements taken from a certain object (e.g., results of a number of medical tests taken from a patient), while the labels mark the fact that the object belongs ($y = 1$) or does not belong ($y = -1$) to certain set (e.g., a patient has or does not have a particular decease). The problem is of also of interest in the situations when there exists some correlation between the attributes and the labels, so that *in principle* it is possible to predict, to some extent, the value of the label via the values of the attributes. At the same time, we do not know in advance a good predictor and should therefore build it by learning the available data.

The goal of the problem we have just outlined, is to seek for *binary* classifier based on a training sample where *the examples are already classified* "by supervisor" (hence the name "supervised binary classification"). Machine Learning considers various modifications of this problem, e.g., supervised classification in more than two classes, or the *regression estimation* problem, where the "labels" may take arbitrary real values rather than to be ± 1, or *unsupervised* classification (called usually *clustering*) where no labels are given, and the classification should

be carried out solely in terms of a given set of feature vectors. In this wide spectrum of problems, supervised binary classification seems to be the most basic and most frequently used one.

Machine Learning in general, and supervised binary classification in particular, has an extremely wide spectrum of applications, including those in Medical Diagnostics, Bio-informatics, Automatized Classification of Texts and Images (see: [1], Chapter 8), etc.

1.1.2 Supervised Binary Classification via Support Vector Machines

Representing a classifier

The standard way to specify a binary classifier $\phi(x)$ is to represent it in the form

$$\phi(x) = \mathrm{sign}(f(x)), \tag{1.1.1}$$

where $f(x)$ is a real-valued function. For the sake of brevity, we use the word "classifier" for both ϕ and f, speak about the generalization error $\mathrm{err}_{\mathcal{D}}(f)$ of $f(\cdot)$, etc.; this slight ambiguity is not dangerous, since it will be always clear from the context which interpretation of the word "classifier" is meant.

Generic scheme for building a classifier

The most general scheme for building classifiers is as follows. Given in advance a set X of possible values of attribute vectors, we fix a family of tentative classifiers f, that is, a family $\mathcal{F}$ of real-valued functions on X. Next, given a training sample $S = \{(x_i, y_i)\}_{i=1}^{\ell}$, we find in this family a classifier f which exhibits the "best possible classification abilities" on the training sample, and this is the classifier yielded by the sample. The major "degrees of freedom" of this conceptual scheme are

- *the underlying family $\mathcal{F}$ of tentative classifiers*,

and

- *the way in which the "classification abilities" of a tentative classifier $f \in \mathcal{F}$ are quantified.*

Specifying these "degrees of freedom", referred to as a *setup*, we specify the outlined conceptual classification scheme and can investigate its quality in terms of the resulting generalization error.

Margin and Soft Margin-based affine classifiers

We are about to present two setups of primary importance for modern Machine Learning; these are the setups we intend to deal with. In both cases, it is assumed that X is a bounded subset of Euclidean (or real Hilbert) space E with inner product $\langle \cdot, \cdot \rangle$ and the associated norm $\|x\| = \sqrt{\langle x, x \rangle}$.

Example 1: Linear margin-based classification. This setup is as follows:

- $\mathcal{F}$ is the set of all continuous *affine* functions $f(x) = \langle w, x \rangle + b$ on $E \supset X$, or, which is the same, of functions of the indicated form associated with all possible *weights* $w \in E$ and all possible *shifts* $b \in \mathbf{R}$.

- A tentative classifier $f(x) = \langle w, x \rangle + b$ is called *feasible* on a given training sample $S = \{(x_i, y_i)\}_{i=1}^{\ell}$, if

$$\min_i y_i f(x_i) \geq 1, \ i = 1, ..., \ell; \tag{1.1.2}$$

the "classification ability" exhibited by such a classifier on S is $1/\|w\|$, the larger is this quantity, the better. Thus, the optimal classifier is given by the optimal solution to the optimization problem

$$\min_{w,b}\left\{\frac{1}{2}\langle w,w\rangle : 1-y_i[\langle w,x_i\rangle+b]\le 0,\ i=1,...,\ell\right\}. \tag{1.1.3}$$

The outlined setup admits a transparent geometric interpretation. A classifier $f(x)=\langle w,x\rangle+b\in\mathcal{F}$ is feasible on the training sample S if and only if the stripe Π between two parallel hyperplanes $\Pi_+=\{x: f(x)=1\}$ and $\Pi_-=\{x: f(x)=-1\}$ orthogonal to w separates the set $S_+=\{x_i: y_i=1\}$ of (attribute vectors of) "positive" examples from S and the set $S_-=\{x_i: y_i=-1\}$ of "negative" examples, so that all positive points $x_i\in S_+$ are on and above Π_+, and all points $x_i\in S_-$ are on and below Π_-, the "above" given by the direction of w. The geometric width of Π (called the *geometric margin* of the classifier) clearly is $1/\|w\|$, which is exactly the "classification ability" of the classifier as defined above. It follows that the classifier yielded by the setup corresponds to the widest possible strip separating S_- and S_+; such a classifier does exist and is unique, provided that both S_+ and S_- are nonempty and can be separated by a strip of positive width.

A drawback of the margin-based linear classification is that its does not work at all when the sets of positive and negative examples S_+ and S_- cannot be separated by a stripe of positive width, which may be caused by just few "outliers". The second setup we are about to consider is aimed at overcoming this drawback.

Example 2: Linear classification based on soft margin. Here, as above, the family $\mathcal{F}$ of tentative classifiers is comprised of affine functions $f(x)=\langle w,x\rangle+b$ with $w\in E$ and $b\in\mathbf{R}$. With such a classifier and a training sample $S=\{(x_i,y_i)\}_{i=1}^{\ell}$, we associate the *slack margin vector* $\xi=(\xi_1,...,\xi_\ell)$ defined as

$$\xi_i=\xi_i(w,S)\equiv\max[0,1-y_if(x_i)],\ i=1,...,\ell.$$

The "classification ability" of f on S is quantified by the quantity $1/\sqrt{\|w\|+C^2\|\xi\|_2^2}$, where $\|\xi\|_2=\sqrt{\xi^T\xi}$ is the standard Euclidean norm on $\mathbf{R}^\ell$, and $C>0$ is a positive constant. Thus, the optimal classifier is given by the optimal solution to the optimization problem

$$\begin{array}{c}\min\limits_{w,b,\xi}\left\{\frac{1}{2}\left[\langle w,w\rangle+C^2\xi^T\xi\right] : 1-y_i[\langle w,x_i\rangle+b]\le\xi_i,\ i=1,...,\ell,\xi\ge 0\right\}\\ \Updownarrow\\ \min\limits_{w,b}\left\{\frac{1}{2}\left[\langle w,w\rangle+C^2\xi^T(w,S)\xi(w,S)\right] : \xi_i(w,S)=\max[0,1-y_i[\langle w,x_i\rangle+b]],\ i=1,...,\ell\right\}.\end{array} \tag{1.1.4}$$

This setup admits a geometric interpretation similar to the one in Example 1. Same as in this Example, we associate with the (w,b)-part of a feasible solution to (1.1.4) a stripe Π between the parallel hyperplanes $\Pi_\pm=\{x:\langle w,x\rangle+b=\pm1\}$, but now we do not require from this stripe to separate the sets S_+ of "positive" and S_- of "negative" attribute vectors from the sample. Instead, we quantify the "mis-placements" of x_i's with respect to the stripe Π by the quantities $\xi_i(w,S)$ and penalize the original objective in (1.1.3) by the squared norm of the resulting residual vector $\xi(w,S)$. It is well known that the setup in Example 2 can, essentially, be reduced to the one of Example 1 as applied to properly transformed attribute vectors; we will consider this issue in more details in Chapter 2.

Support Vector Machines

The techniques for building classifiers presented in Examples 1 and 2 and in their natural modifications to be considered in the main body of this Thesis are commonly referred to as *Support Vector Machines* (SVMs)[1)]. SVMs originate from the pioneering work of Boser, Guyon and Vapnik [**?**] and Cortes and Vapnik [**?**] and are now considered as a classic techniques for processing problems of Supervised Binary Classification. The associated body of knowledge consists, essentially, of 3 parts:

1. *Statistical Learning Theory* – a deep and highly nontrivial theory capable to quantify the quality (the generalization error) of a classifier yielded by an SVM (and by similar techniques associated with more general families $\mathcal{F}$ of tentative classifiers). We shall review this theory in more details in Chapter 2; for the time being, it suffices to say that this theory provides theoretical justification of the SVM approach along with general guidelines on its implementation;

2. *Computational SVM machinery*. An extremely attractive feature of SVMs, not shared by other techniques for solving classification problems of Machine Learning, is that building the associated "theoretically justified" classifiers reduces to solving *well-structured convex* optimization problems like (1.1.3), (1.1.4), and as such can be carried out in a computationally efficient manner. This is a key to real-world applications which more often than not require processing really large-scale data sets (samples with many hundreds/thousands of examples with the dimensions of the attribute vectors up to tens and hundreds of thousands), which makes computational efficiency of the approaches a real *must*. Over the years, the computational aspects of SVMs were subject of intensive research and testing, and the corresponding algorithms and software include a wide spectrum of convex optimization tools adjusted to SVM optimization models;

3. *Methodology of adjusting SVM models to specific data sets and experience gained in academic and real world applications*. In fact, the SVM machinery is *by far* more flexible than it is suggested by Examples 1, 2. A crucial new "flexibility dimension" is added by a possibility to pre-process the models presented in these examples by an appropriate *nonlinear* embedding $x \mapsto \phi(x)$ of the set X of possible values of attribute vectors into a Euclidean/Hilbert *feature space* F with inner product $\langle \cdot, \cdot \rangle_F$. Given such an embedding, we can identify the attribute vectors x with their images $\phi(x)$ in the feature space; with this identification, *affine classifiers* $f(\cdot) = \langle w, \cdot \rangle_F + b$ on F induce *nonlinear* classifiers $\mathrm{sign}(f(\phi(x)))$ on the original attribute space, thus bringing into the scope of linear SVMs wide families of highly nonlinear classifiers. A nice feature of such a construction is that its implementation usually does *not* require explicit specification of the embedding $x \mapsto \phi(x)$; all which matters is the associated *kernel* $K(x', x'') \equiv \langle \phi(x'), \phi(x'') \rangle_F$. Such a kernel can be an arbitrary real-valued function on $X \times X$ satisfying the well-known *Merser* conditions (see, e.g., [1], Chapter 3) and can be specified without explicit reference to the underlying embedding (as it is the case with popular *Gaussian kernels* $K(x', x'') = \exp\{-\|x' - x''\|_2^2/\sigma^2\}$ on subsets X of $\mathbf{R}^n$). There are numerous real world

[1)] The name comes from the fact that the w-component w_* of the unique optimal solution to (1.1.3) is a linear combination, with nonnegative coefficients, of the vectors $y_i x_i$ associated with *support* feature vectors x_i from the training sample – those belonging to the boundary hyperplanes $\Pi_\pm$ of the widest stripe Π separating S_+ and S_-. These support vectors are fully responsible for the resulting classifier – the latter remains unchanged when one removes from the training sample "non-support" examples.

situations where affine classification of reasonable quality is impossible in the original attribute space but is possible in a properly defined feature space (that is, with properly chosen kernel), so that the "flexibility dimension" becomes really vital for the success. At the same time, utilizing the just outlined option requires various techniques for choosing and adjusting kernels (and other elements of of SVM models, like the parameter C in (1.1.4)). These techniques and related experience form the concluding part of the SVM body of knowledge.

1.2 Computational Machinery of SVMs and Positioning of the research

Our research is focused on the second major component of the SVM machinery, that is, on computational tools for processing SVM models. Our major goal is *to investigate the potential in the SVM context of novel "computationally cheap" techniques for extremely large-scale optimization, specifically, the Mirror Prox (MP) algorithm originating from [3].* To motivate this goal, we start with brief outline of the state-of-the-art in modern Convex Optimization.

1.2.1 Modern Convex Optimization and SVMs

Interior Point Methods – advantages and limitations

During the last two decades, the major emphasis in convex optimization was on *Interior Point Methods* (IPM) – theoretically efficient polynomial time algorithms capable to obtain high-accuracy solutions of various families of well-structured convex optimization problems (like those of Linear, Quadratic and Semidefinite Programming) at a relatively low iteration count. As a result, today essentially all Convex Programming is "within the reach" of IPMs; in particular, these methods were used successfully to process SVM models of medium size. However, it eventually became clear that aside of Linear Programming, the scope of practical applications of IMP methods is restricted by problems with at most few thousands of decision variables. This restriction comes from the fact that IPMs have "computationally demanding" iterations with the number of arithmetic operations growing nonlinearly with the design dimension n of the problem, namely, as $O(n^3)$, unless problem's data possess favourable sparsity structure. As a matter of fact, the latter is often the case with Linear Programming programs of real-life origin and is *not* the case with nonlinear problems. Fast – cubic – growth of computational effort per iteration makes it practically too expensive or even impossible to handle problems with tens and hundreds thousands of design variables – fast, in terms of iteration count, convergence does not help much when the very first iteration "lasts forever". In situations like this, high accuracy turns out to be too computationally expensive; what one would like to do, is "to offer reduced accuracy at reduced price", but this is impossible when already a single iteration is too computationally costly. Note that problems of sizes prohibitively large for IPMs arise in many applications, including those in SVM.

"Cheap" optimization techniques for extremely large-scale convex problems

From practical viewpoint, design dimension n about 10^4 makes inapplicable methods with iteration cost $O(n^3)$; n like 10^5 and more makes a must nearly linear in n arithmetic cost of iteration. At the present level of our knowledge, these requirements rule out IPMs and other

polynomial time methods and leave us with "cheap" first order computational methods like gradient descent, conjugate gradients, quasi-Newton methods with restricted memory, etc. Our options are further restricted by the necessity to handle constraints, even simple ones (which do arise in the SVM context). These limitations imply important theoretical consequences. Indeed, all known "computationally cheap" optimization techniques are *black box oriented* – they are unable to utilize full a priori knowledge of problem's data and structure and progress solely on the basis of local information on the problem (the values and the derivatives of the objective and the constraints) collected along generated sequence of iterates. As a result, these methods obey limits of performance established by Information-Based Complexity Theory (see, e.g., [5]) which state, in particular, that *in the large-scale case, black box oriented methods can exhibit only slow – sublinear – rate of convergence*. The achievable convergence rate depends on the smoothness of the objective and the geometry of the feasible set and is never better than $O(1/t^2)$ (that is, the inaccuracy in terms of the objective goes to 0 with the number t of objective's evaluations not faster than $O(1/t^2)$); in the large-scale nonsmooth case, the best guaranteed rate of convergence is as slow as $O(1/\sqrt{t})$. A practical conclusion of the outlined limits of performance is that *in the large-scale case, one cannot hope to get high-accuracy solutions in reasonable time*. Note, however, that in most practical applications there is no need in high-accuracy solutions, already medium-accuracy ones are sufficient. What is really important in large-scale applications, is *whether medium-accuracy solutions can be obtained at dimension-independent convergence rate, or the corresponding iteration count deteriorates with the problem's dimension*. And in this respect, there are good news: *on problems with favourable geometry* (which is the case for many SVM models), *properly designed computationally cheap methods possess dimension-independent rate of convergence and thus are well-suited for the large-scale case*.

It should be added that *sometimes, optimization to high accuracy is not merely redundant, it is even counter-productive*. This phenomenon usually takes place in situations where the optimization by itself is a tool rather than a goal and, besides this, the data of the optimization problem to be solved are subject to perturbations (e.g., are random). A simplest example here is the Maximum Likelihood parameter estimation. Closer to our subject, this is the situation with SVM models as well – what we are interested in, is a classifier with good generalization error, which is not exactly the same as the classifier with the best possible classification abilities on our (random!) training sample. What happens in numerous situations of the outlined type is as follows: at the beginning of the optimization process, progress in purely optimization terms (that is, progress in the value objective function) goes along with progress in the "actual quality" of the approximate solution (the estimation error in the Maximum Likelihood example or the generalization error in the SVM one). At certain point in time, everything changes – further progress in terms of the objective function is accompanied by deteriorating, sometimes dramatic, of the "actual" quality of the solution. This behaviour, called "overfitting" in the SVM literature, admits informal "phenomenological" explanation as follows: at the beginning of the optimization process, when the "major portion" of initial non-optimality in terms of the objective function is eliminated, this happens via adjusting the solution to the "representative" component of the data. At a certain point, further progress in the objective is caused by adjusting the solution to "individual", statistically meaningless, features of the particular data we are handling, which may be counter-productive as far as our actual goals are concerned. Sometimes we have in our disposal a possibility to quantify the "actual" quality of an intermediate solution (in the SVM context, this can be done by testing an intermediate classifier on the *validation sample*, see Chapter 5). In situations like this, an optimization process with many cheap

steps could be more preferable than a process with few expensive ones even when both the processes require the same overall effort. Indeed, in the first case we can better localize the "turning point" and as a result end up with a solution of better quality (cf., e.g., numerical results in Chapter 5).

The fact that simple first order optimization methods are unavoidable when processing large data sets via SVM is, of course, a common knowledge in the Machine Learning community. What seems to be *not* a common knowledge there, are recent advances in the area of these methods, advances which can make the first order algorithms commonly used in SVMs somehow obsolete.

1.2.2 Recent advances in theory of "cheap" computational methods for extremely large-scale optimization

Traditionally, "cheap" optimization methods deal with convex optimization programs of the form

$$\min_{x\in X} f(x), \tag{1.2.1}$$

where X is a simple convex compact set in $\mathbf{R}^N$ and $f(x)$ is a Lipschitz continuous convex function on X. A sufficient for our purposes sample of basic results on limits of performance of black-box-oriented methods on large-scale problems of this type are as follows (for details, see, e.g., [5, 7, 6, 3])

- *The case of $\|\cdot\|_2$-geometry:* X is a subset of $\|\cdot\|_2$-ball of radius R in $\mathbf{R}^N$.
  - [*Nonsmooth case*] f is Lipschitz continuous, with constant L, convex function on X: $|f(x) - f(y)| \le L\|x - y\|_2$.
    Here the unimprovable in the large-scale case efficiency estimate (that is, upper bound on the residual $\epsilon = f(x) - \min\limits_X f$ in terms of the number t of required steps, with single evaluation of $f(x)$, $f'(x)$ per step) is
    $$\epsilon = O(1)\frac{LR}{\sqrt{t}} \tag{1.2.2}$$
    (from now on, all $O(1)$'s are absolute constants).
  - [*Smooth case*] f possesses Lipschitz continuous, with constant L, gradient on X: $\|f'(x) - f'(y)\|_2 \le L\|x - y\|_2$ for all $x, y \in X$.
    Here the unimprovable in the large-scale case efficiency estimate is
    $$\epsilon = O(1)\frac{LR^2}{t^2} \tag{1.2.3}$$
- *The case of $\|\cdot\|_1$-geometry:* X is a subset of $\|\cdot\|_1$-ball of radius R in $\mathbf{R}^N$.
  - [*Nonsmooth case*] f is Lipschitz continuous, with constant L, convex function on X: $|f(x) - f(y)| \le L\|x - y\|_1$.
    Here the unimprovable in the large-scale case efficiency estimate is
    $$\epsilon = O(1)\frac{LR\sqrt{\log N}}{\sqrt{t}} \tag{1.2.4}$$

– [*Smooth case*] f possesses Lipschitz continuous, with constant L, gradient on X: $\|f'(x) - f'(y)\|_\infty \leq L\|x - y\|_1$ for all $x, y \in X$.

Here the unimprovable in the large-scale case efficiency estimate is

$$\epsilon = O(1)\frac{LR^2 \log N}{t^2}. \tag{1.2.5}$$

Note that the outlined efficiency estimates are (nearly) dimension independent; the corresponding methods are "cheap" (single computation of f, f' per step plus overhead of $O(n)$), provided that X is simple enough. Note that the outlined dimension-independent rate of convergence is impossible when passing from $\|\cdot\|_2$ and $\|\cdot\|_1$-balls X to, e.g., boxes. For example, when X is the $\|\cdot\|_\infty$-ball of radius R and f is convex with Lipschitz continuous gradient: $\|f'(x) - f'(y)\|_2 \leq L\|x - y\|_\infty$, the best known efficiency estimates are proportional to the design dimension N.

As we see, the limits of performance of black-box-oriented methods for solving (1.2.1) heavily depend on the smoothness properties of the objective. Unfortunately, smoothness is a "rare commodity" – the cases when one can reformulate a convex problem in the form of (1.2.1) and end up with both smooth f and "simple" X (the latter is necessary when a cheap optimization method is sought) are pretty rare.

> For example, SVM models (1.1.3), (1.1.4) "as they are" possess smooth convex objectives but complicated feasible sets. It is not difficult to convert these problems to the form of (1.2.1) with simple X. For example, assuming w.l.o.g. that E is $\mathbf{R}^N$ with the standard inner product, (1.1.3) can be rewritten as
>
> $$\min_{\|w\|_2 \leq 1, |\beta| \leq 1} f(w, \beta) = \max_i -y_i[w^T x_i + R\beta], \quad R = \max_i \|x_i\|_2 \tag{1.2.6}$$
>
> (the optimal value in this problem is minus the geometric margin of the best possible affine classifier on the training sample). The feasible sets of the resulting problem is a simple subset of Euclidean ball of radius R, $R = \max_i \|x\|_i$, and the objective is convex Lipschitz continuous, with constant $O(1)$, function, although a nonsmooth one. Thus, the problem can be solved with dimension-independent efficiency estimate $O(1)Rt^{-1/2}$.

A recent (2003) breakthrough in this area, due to Yu. Nesterov [6], is that *exploiting a priori knowledge of the structure of the objective* f (in Convex Optimization, this knowledge is a rule rather than exception), *one usually can convert a nonsmooth minimization problem* (1.2.1) *into a saddle point problem*

$$\min_{x \in X} \max_{y \in Y} \phi(x, y) \tag{1.2.7}$$

with <u>*smooth*</u> (Lipschitz continuous gradient) convex-concave cost function ϕ. As a result, *in the case of favourable geometry of* X, Y*, the dimension-independent efficiency estimate achievable for "cheap" optimization methods improves from* $O(1)t^{-1/2}$ (see (1.2.2), (1.2.4)) *to* $O(1)t^{-1}$. Note that this acceleration, quite important from the practical viewpoint, does not contradict the "ultimacy" of the limits of performance of black-box-oriented methods as established by the Information-Based Complexity Theory, since passing from nonsmooth optimization problem (1.2.1) to smooth saddle point problem (1.2.7) utilizes a priori knowledge of the structure of the original objective and thus is *not* black-box-based.

A simple and important observation of Nesterov made it possible to introduce cheap computational techniques for solving extremely large-scale convex programs by utilizing a priori

knowledge of problem's structure – an option which was previously available only in the context of Interior Point methods (and thus impossible to use in the really large-scale case). During years 2003 – 2004, three essentially different "cheap" optimization algorithms were developed, utilizing saddle point reformulation of (1.2.1) were introduced in order to ensure in favourable circumstances dimension-independent $O(1)t^{-1}$-rate of convergence. They are called:*smoothening* algorithm of Yu. Nesterov [6], *Mirror Prox* algorithm of A. Nemirovski [3] and *Dual extrapolation* algorithm of Yu. Nesterov [9].

1.2.3 Goals of the Thesis

The primary goal of our research is to investigate the potential of the outlined recent progress in "computationally cheap" techniques for extremely large-scale convex optimization in the SVM context. Specifically, we intend to utilize here the Mirror Prox algorithm. The research focuses on

(a) representing the SVM optimization programs in the specific saddle point form required by the algorithm,

(b) adjusting the algorithm to the structure of the SVM problems, and

(c) software implementation and testing the Mirror Prox algorithm on large-scale SVM data, with the ultimate goal to develop a novel theoretically solid and computationally efficient optimization techniques for SVM-based large-scale Supervised Binary Classification.

1.3 Structure of Thesis and Overview of results

In this section, we describe the structure of the main body of the Thesis and outline our major results.

The main body of Thesis consists of four chapters:

- Chapter 2 reviews the basics of the Statistical Learning Theory;
- Chapter 3 presents the necessary background on the Mirror Prox algorithm, which is the main computational tool we intend to exploit in the SVM context;
- Chapter 4 converts the basic SVM models to saddle point problems to be solved by the Mirror Prox algorithm;
- Chapter 5 describes our implementation of the MP algorithm as applied to SVM-originating saddle point problems, discusses and motivates the methodology of our numerical experimentation, present a detailed summary of our numerical results and makes recommendations and conclusions on processing SVM models via MP.

In more details, the contents of the Thesis can be described as follows.

1.3.1 Statistical Learning Theory

In this section, we primarily overview some well-known SVM-related results on Statistical Learning Theory; our exposition mainly follows the one in [1], with some extensions which, to the best of our knowledge, are new. These results form a theoretical justification of the SVM machinery; roughly speaking, they bound from above the generalization error of a classifier taken from a given class $\mathcal{F}$ in terms of

(a) the length of the training sample,
(b) the behaviour exhibited by the classifier on this sample, and
(c) the "richness" of the family $\mathcal{F}$ of tentative classifiers. When applied to the SVM-based classifiers, these bounds are as follows.

A. Bounding generalization error via margin. The corresponding result (Theorem 2.3 – an extension of Theorem 4.18 in [1]) is as follows:

> [Theorem 2.3] *Let $(E, \|\cdot\|$ be a κ-regular normed space, let X be a subset of the centered at the origin $\|\cdot\|$-ball of radius R, and let $\gamma \in (0, R]$. Further, let $\mathcal{D}$ be a probability distribution on $X \times \{-1, 1\}$, ℓ be a positive integer and $\delta \in (0, 1)$. Then the following is true, up to $\mathcal{D}$-probability of bad sampling $\leq \delta$:*
>
> *If $S = \{(x_1, y_1), ..., (x_\ell, y_\ell)\}$ is an ℓ-element training sample drawn from $\mathcal{D}$ and an affine function*
>
> $$f(x) = \langle w, x\rangle + b$$
>
> *with $\|w\|_* \leq 1$ and $|b| \leq R$ has margin γ on S:*
>
> $$y_i f(x_i) \geq \gamma \; i = 1, ..., \ell,$$
>
> *then the generalization error of the classifier* $\mathrm{sign}(f(x))$ *admits upper bound as follows:*
>
> $$\mathrm{err}_{\mathcal{D}}(f) \leq \frac{O(1)}{\ell}\left[\kappa\frac{R^2}{\gamma^2}\log^2\left(\ell\frac{R}{\gamma}\right) + \log\frac{2}{\delta}\right]. \qquad (1.3.1)$$

The prototype of this statement – Theorem 4.18 in [1] – deals with the case when $(E, \|\cdot\|)$ is a Hilbert space with the inner product norm and $\langle\cdot,\cdot\rangle$ is the inner product on E; to the best of our knowledge, this is the only situation considered in the literature so far. The novelty is that we allow for $(E, \|\cdot\|)$ to be a *κ-regular normed space*, where the latter notion, taken from [4]) is defined as follows: Let E be a normed space with a norm $\|\cdot\|$, and let $\pi(x)$ be an equivalent norm on E, that is, $\pi(\cdot)$ is another norm on E satisfying the relation

$$\forall x \in E : \|x\|^2 \leq \pi^2(x) \leq c_\pi\|x\|^2 \qquad (*)$$

with a certain constant $c_\pi < \infty$. The notion of a *κ-regular* normed space $(E, \|\cdot\|)$ is defined as follows (see [4]):

> **Definition 2.4***Let $(E, \|\cdot\|)$ be a normed space.*
>
> (i) *Let $\pi(\cdot)$ be a norm on E which is equivalent to $\|\cdot\|$, and r be a positive real. We say that $\pi(\cdot)$ is r-smooth, if the function $P(x) = \pi^2(X)$ is continuously Frechet differentiable and satisfies the inequality*
>
> $$\forall (x, z \in E) : P(x + z) \leq P(x) + \langle P'(x), z\rangle + rP(z).$$
>
> (ii) *Let κ be a positive real. We say that $\|\cdot\|$ is κ-regular, if there exists r and r-smooth norm $\pi(\cdot)$ on E such that $rc_\pi \leq \kappa$, where c_π is the "compatibility constant" of $\pi(\cdot)$ and $\|\cdot\|$, see $(*)$.*

Basic examples of regular normed spaces include, e.g.,

1. *Hilbert space.* The inner product norm $\|x\|_E = \sqrt{\langle x, x\rangle_E}$ on a Hilbert space E with inner product $\langle \cdot, \cdot \rangle_E$ clearly is 1-smooth. Consequently, a Hilbert space is 1-regular;

2. *Spaces $\ell_p(M)$, $2 \le p < \infty$.* Let M be a set, and let $1 \le p < \infty$. The space $\ell_p(M)$ is comprised of all real-valued functions $\phi(\cdot)$ on M with countable supports $\{\mu \in M : \phi(\mu) \neq 0\}$ and finite $\|\cdot\|_p$-norms

$$\|\phi(\cdot)\|_p = \left(\sum_{\mu \in M} |\phi(\mu)|^p \right)^{1/p};$$

the linear operations on $\ell_p(M)$ are defined in the standard point-wise fashion. The space $\ell_\infty(M)$ is defined similarly, up to the fact that now we define the norm (and require it to be finite) as

$$\|\phi(\cdot)\|_\infty = \sup_{\mu \in M} |\phi(\mu)|.$$

It is easily seen (see [4]) that *when $2 \le p < \infty$, the norm $\|\cdot\|_p$ is r-smooth with $r = p - 1$.* Thus,

The spaces $\ell_p(M)$, $2 \le p < \infty$, are $(p-1)$-regular.

The *upper bounds on the* "level of regularity" κ in the above example deteriorates when $p \to \infty$. This indeed happens in the case of an infinite-dimensional space; however, in the *finite-dimensional* situation κ can be bounded, uniformly in $2 \le p \le \infty$, solely in terms of the dimension, and this bound extremely slowly – logarithmically - deteriorates with the dimension, as can be seen from the following example:

Space $\ell_p(M)$, $2 \le p \le \infty$, associated with a <u>finite</u> set M, is κ-regular with

$$\kappa \le O(1) \min\{p - 1, \log(\mathrm{Card}(M) + 1)\},$$

where $O(1)$ is an appropriate absolute constant.

For more examples of regular normed spaces, see Section 2.3.1

In Theorem 2.3, $\langle \xi, x \rangle$ is the value of a continuous linear functional ξ on a vector $x \in E$, and $\|\xi\|_* = sup|\langle \xi, x\rangle|, x \in E : \|x\| \le 1$ is the conjugate (to norm $\|\cdot\|$) norm on the dual to $(E, \|\cdot\|)$ space E^* of continuous linear functionals on E; in the case of Hilbert space E with the inner product norm $\|\cdot\|$, E^* is canonically identified with E; with this identification, $\|\cdot\|_*$ becomes $\|\cdot\|$, and Theorem 2.3 becomes a well-known standard result of Machine Learning theory. The power of the latter result in the SVM context comes from the fact that the associated upper bound on the generalization error is "dimension-independent" – what matters is solely the ratio R/γ of the Hilbert ball containing the set of tentative feature vectors to the geometric margin γ exhibited by the (normalized) affine classifier on the training sample. The dimension-independent nature of the result allows to work with extremely high-dimensional, and even infinite-dimensional, Hilbert feature spaces. Theorem 2.3 extends this option to a much wider family of feature spaces. As an extreme example, consider the case where E is the space $\ell_\infty(M)$ associate with a *finite* set M, that is, E is the space $\mathbf{R}^N$ of the dimension $N = \mathrm{Card}(M)$ equipped with the uniform norm $\|\cdot\|_\infty$. Here E^* can be naturally identified with $E = \mathbf{R}^N$; with this identification, $\langle \xi, x \rangle$ becomes just $\xi^T x$, and $\|\cdot\|_*$ becomes the norm $\|\cdot\|_1$ on $E^* = E = \mathbf{R}^N$. The problem of finding a normalized classifier with the largest possible geometric margin on the training sample

becomes now the popular SVM model with $\|\cdot\|_1$-normalization of tentative classifiers. Theorem 2.3 states that the "generalization abilities" of this model are, up to factor $\log N$, the same as for the standard "Euclidean" model with $\|\cdot\|_2$ in the roles of $\|\cdot\|$ and $\|\cdot\|_*$. Thus, unless the dimension of the feature space becomes astronomically large, $\|\cdot\|_1$-SVMs (that is, SVM models obtained from (1.1.3) by replacing the objective $\|w\|_2^2 \equiv \langle w, w\rangle$ with $\|w\|_1^2$) are nearly as theoretically valid as the standard $\|\cdot\|_2$-SVMs (1.1.3).

In this respect, it is worthy to mention that in a $\|\cdot\|_1$-SVM we are seeking to separate the sets $S_+ = \{x_i : y_i = 1\}$ and $S_- = \{x_i : y_i = -1\}$ of the positive and the negative training input vectors by a stripe with the largest possible $\|\cdot\|_\infty$-width, while in the $\|\cdot\|_2$-model the goal is to find the separating strip of the largest $\|\cdot\|_2$-width. One could think that since the $\|\cdot\|_\infty$-width of a strip always is $\le$ the $\|\cdot\|_2$-width, the $\|\cdot\|_1$-SVM always is inferior as compared to the $\|\cdot\|_2$-one – the numerical value γ_1 of the geometric margin in the former model always is $\le$ the value γ_2 of the margin in the latter model. This conclusion, however, overlooks the fact that what matters is not the numerical value of the geometric margin itself, but the ratio of this value to the radius R of the smallest ball containing the set X of tentative input vectors. For $\|\cdot\|_1$-SVM, $R = R_1$ should be taken w.r.t. the norm $\|\cdot\|_\infty$, and for $\|\cdot\|_2$-SVM, $R = R_2$ should be taken w.r.t. the norm $\|\cdot\|_2$. It is clear that $R_1 \le R_2$, so that in the ratios γ_1/R_1 and γ_2/R_2 to be compared *both* numerators and denominators are linked by similar inequalities. As a result, we cannot say in advance which one of the ratios is larger (that is, which one of the SVM models is better from the viewpoint of the generalization error); the answer depends on the particular geometry of X, and there are cases when $\|\cdot\|_1$-SVM is much better than the $\|\cdot\|_2$-one. This observation, we believe, makes it clear that our extension of bounds on generalization error from the case of a Hilbert feature space to the case of a regular feature space is more than just an academic exercise.

B. Bounding generalization error via margin and margin slacks. The corresponding result (Theorem 2.4 – an extension of Theorems 4.22 and 4.24 in [1]) is as follows:

[Theorem 2.4] *Let $(E, \|\cdot\|)$ be a κ-regular normed space, and let X be a subset of the centered at the origin $\|\cdot\|$-ball of radius R. Let us fix $p \in [2, \infty]$ with $p < \infty$ when X is infinite, and real $\gamma \in (0, \sqrt{2}R]$.*

Further, let $\mathcal{D}$ be a probability distribution on $X \times \{-1, 1\}$ such that

$$\mathrm{Prob}_{((x,y),(x',y'))\sim\mathcal{D}\times\mathcal{D}}\{x = x', y \neq y'\} = 0,$$

ℓ be a positive integer and $\delta \in (0, 1)$. Then the following is true, up to $\mathcal{D}$-probability of bad sampling $\le \delta$:

If an affine function

$$f(x) = \langle w, x\rangle + b$$

with $|b| \le \sqrt{2}R$ has on ℓ-element training sample $S = \{(x_1, y_1), ..., (x_\ell, y_\ell)\}$ drawn from $\mathcal{D}$ slack variable vector ξ w.r.t. γ, S:

$$\xi_i = \max[0, \gamma - y_i f(x_i)],\ i = 1, ..., \ell,$$

such that

$$\sqrt{\|w\|_*^2 + R^{-2}\|\xi\|_{p_*}^2} \le 1,\ p_* = \frac{p}{p-1},$$

then f induces a classifier with generalization error satisfying the bound

$$\mathrm{err}_{\mathcal{D}} \leq \frac{O(1)}{\ell}\left[\widehat{\kappa}(p)\frac{R^2}{\gamma^2}\log^2\left(\ell\frac{R}{\gamma}\right)+\log\frac{2}{\delta}\right],$$

where, for all regular spaces listed so far,

$$\widehat{\kappa} \leq O(1)\left[\kappa+\min[p-1,\log\mathrm{Card}(X)]\right].$$

Here again we manage to extend the theoretical results known for the case of Hilbert feature space E with the inner product norm $\|\cdot\| = \|\cdot\|_E$ and for the Euclidean norm $\|\cdot\|_2$ of the slack vector ($p = 2$) to the case of regular normed spaces $(E, \|\cdot\|)$ and $\|\cdot\|_{\frac{p}{p-1}}$-norm of the slack vector, with $p \in [2, \infty)$ for general X and $p \in [2, \infty]$ for finite X.

1.3.2 Background on the Mirror Prox algorithm

In Chapter 3, we present the detailed description of the construction and the convergence properties of the Mirror Prox algorithm developed in [3]. For the time being, it suffices to indicate that the performance of this algorithm depends on the choice of a *distance-generating function*; this choice allows to adjust, to some extent, the MP to the geometry of the saddle point problem. Our presentation, as compared to [3], contains few minor novelties aimed at "good" choice of the distance-generating function for a couple of geometries (p-balls with $1 < p \leq 2$ and "extended simplexes"), which were not considered in [3]; these novelties are summarized in Proposition 3.3.

1.3.3 Saddle Point reformulations of the SVM models

In Chapter 4, we make the major step towards our ultimate goal of processing SVM models via the Mirror Prox algorithm. Specifically, we derive here the saddle point reformulations of SVM models (recall that this is problem's format required by MP). We present the required reformulations (see (4.2.4), (4.3.4), Theorem 4.1)) for a wide variety of SVM's, both *kernel-generated* and *plain* ones. In both cases, the SVM models build an affine classifier in the feature space. For the kernel-generated models, the final form of this classifier is

$$f(x) = \sum_{i=1}^{\ell} K(x, x_i)y_j\alpha_i + b, \tag{1.3.2}$$

where $K(x', x'') : X \times X \to \mathbf{R}$ is a given Merser kernel on the set X of tentative values of attribute vectors, x_i are the attribute vectors from the training sample, y_i are their labels, and α_i are the "weights" finally yielded by the optimal solution to an SVM-type optimization problem (which we further convert to a saddle point problem); in the SVM slang, (1.3.2) represents the desired classifier as a combination of support input vectors. In contrast to this, in a *plain* SVM model we work directly in the feature space (which we always assume to be certain $\mathbf{R}^N$) and look for an affine classifier in the form of

$$f(x) = w^T\phi(x) + b$$

where $x \to \phi(x)$ is a given "lifting" of attribute vectors into the feature space.

We demonstrate that in all cases, the saddle point reformulations of various SVM "soft margin" models are of the generic form

$$\min_{\zeta:\|\zeta\|\le\rho}\max_{\lambda\in B_p^+(1):\sum_i y_i\lambda_i=0}\left[\sum_i\lambda_i+\lambda^TQ\zeta\right],\quad\left[B_p^+(1)=\{\lambda\ge 0:\|\lambda\|_p\le 1\}\right] \tag{1.3.3}$$

where

- ζ stands for the vector of "primal variables" (either α, or w, depending on whether we are working with a kernel-generated or a plain SVM model);
- $\|\zeta\|$ is an appropriate norm (which is either the kernel-generated 2-norm $\|\zeta\|_K^2 = \sum_{i,j}\zeta_i\zeta_j y_i y_j K(x_i,x_j)$, or $\|\cdot\|_r$ with $1\le r\le 2$),
- parameter $p\in[2,\infty]$ is responsible for how we measure the norm of the margin slack vector ξ (the latter norm is $\|\xi\|_{\frac{p}{p-1}}$);
- parameter $\rho>0$ is a setup parameter of the SVM model (cf. parameter C in (1.1.4))
- Q is a given matrix.

1.3.4 Implementation, experimentation, numerical results and conclusions

In concluding Chapter 5, we

1. Work out in full details algorithmic implementation of the MP algorithm as applied to (1.3.3) (Section 5.1).
2. Present our experimentation methodology and describe the data sets used in our experiments (Section 5.2).
3. Describe the numerical results yielded by our experiments (Section 5.3) and formulate recommendations and conclusions suggested by these results (Section 5.4).

A brief overview of our experimentation methodology is as follows. Since our goal is to investigate the potential of the Mirror Prox algorithm in the SVM context rather than to process specific classification problems, we focused *solely on plain SVM models*; in our opinion, this restriction should not cause much bias in answering the question we are actually interested in, while allowing at the same time to avoid time-consuming (and essentially irrelevant in our context) issues of choosing appropriate kernels and adjusting their parameters for various data sets. In our MATLAB-based experimentation, we use 17 data sets, mainly found in Internet; 9 of these 17 data sets are qualified as "large-scale ones", with attribute(≡feature) dimensions varying from 500 to 100,000 and sizes of training samples varying ftom 40 to 15,000, while the remaining 8 data sets were medium- and low-dimensional (attribute dimension from 8 to 166). Every one of these data sets was processed several times, with different parameters ρ, p, r of the SVM model and in relevant cases – with several competitive MP setups, with the total number of experiments amounting to 316. Our goal was to evaluate the performance characteristics of MP *in the particular SVM context* we are interested in rather than to characterize the performance of MP from purely optimization perspective. In particular, we paid systematic attention to

what happens with the generalization error, as evaluated on dedicated *validation sample*, of intermediate classifiers generated in course of running the method.

Detailed statistical data on various performance characteristics of different versions of MP as recorded in our experiments are presented in Section 5.3; here we restrict ourselves with just few highlights:

- MP yielded classifiers of quite respectable quality with average generalization error, as evaluated on the validation samples, about 19% in all experiments, and about 18% in large-scale experiments. In about 50% of the latter experiments, the validation error was below 10%;

- The quality of the classifiers yielded by MP was better than the one for classifiers yielded by "precise" Interior Point methods (see the above comments on "overfitting"), and these classifiers were obtained much faster than with the IPM's. For example, the *total* CPU time used to build "good" classifiers for all 9 of our large-scale data sets by the recommended version of MP (the one with early termination and $p = r = 2$, see Section 5.3.2) was as small as 455 sec, which is less than 15% of the CPU time required to process *just one* of these data sets by a high-performance commercial IPM solver `mosekopt`. It should be added that the IPM solver failed to process one of the large-scale data sets ("out of memory" abnormal termination) and failed to meet its built-in convergence criteria on two other large-scale data sets. In contrast to this, computational behaviour exhibited in our experiments by MP was quite reliable.

The bottom line, as suggested by our experimental results, is as follows:

The Mirror Prox algorithm seems to be a highly attractive computational tool for processing large-scale SVM models; it allows to get reasonably good classifiers at an essentially lower computational cost than the Interior Point polynomial time methods. In addition, *the quality of MP-originating classifiers in all our experiments has never been worse, and in many cases – essentially better than the quality of classifiers associated with the precise Interior Point solutions to the corresponding optimization models.*

Chapter 2

Basics from Statistical Learning Theory

In this Chapter, we outline some basic facts of Statistical Learning Theory, one of the two major components of our research. The presentation follows, with certain modifications (which might be of interest by their own right), the one in [1], Section 4.3.

2.1 The setup

In what follows, we focus on the basic model of *supervised learning with binary classification.*

2.1.1 The model

The basic model of supervised binary classification is as follows. There exists an *unknown* probability distribution $\mathcal{D}$ supported on a given set $X \times \{-1, 1\}$, so that realizations of the corresponding random variable, called *examples*, are pairs (x, y), where $x \in X$ and $y = \pm 1$ (y is called the *label* of the example). If not otherwise stated, we assume from now on, that X is a finite, although perhaps a very large, set. We are given a *training sample* $S = \{(x_1, y_1), ..., (x_\ell, y_\ell)\}$ of ℓ examples drawn, independently of each other, from the distribution $\mathcal{D}$, and our goal is to build, based on this data, a *classifier* – a function $f(z) : X \to \mathbf{R}$ which we intend to use to "predict" the label y of a new example (z, y) given the z-part of the example. The predicted label, by definition, is

$$\widehat{y} = \mathrm{sign}(f(z)), \tag{2.1.1}$$

where for a real a $\mathrm{sign}(a)$ equals to 1 if $a \geq 0$ and equals to 0 otherwise.

The quality of a classifier f is measured by its *generalization error*

$$\mathrm{err}_{\mathcal{D}}(f) \equiv \mathrm{Prob}_{\mathcal{D}} \{(x, y) : \mathrm{sign}(f(x)) \neq y\} . \tag{2.1.2}$$

Our ideal goal would be *to find for a given S a classifier belonging to a pre-specified class $\mathcal{F}$ of functions $f : X \to \mathbf{R}$ with as small generalization error as possible.*

Based on the results of the Statistical Learning Theory we are interested in providing an upper bounds on the generalization error in terms of:

(a) The capacity ("richness") of the family $\mathcal{F}$ of potential classifiers.

(b) The cardinality ℓ of the training sample.

(c) The behaviour of a particular classifier $f \in \mathcal{F}$ on the training sample.

The role of results of this type in building classification algorithms stems from the fact that they, essentially, say what kind of behaviour on a given training sample S should we require from a classifier $f \in \mathcal{F}$ in order to have an appropriate generalization error. With this understanding at hand, we can choose a classifier with the best possible, over $f \in \mathcal{F}$, behaviour on S.

> In fact, the existing results of Statistical Learning Theory are too rough to yield really accurate bounds on the generalization error of candidate classifiers. Besides this, there are situations when it is too difficult computationally to implement straightforwardly theoretical recommendations (to optimize the corresponding criterion over $f \in \mathcal{F}$). As a result, the theory, essentially, suggests the structure of the criterion to be optimized, while "fine tuning" of the criterion is upon the researcher working on a particular application.

2.1.2 Reliability of error bounds

An immediate observation is that aside of trivial cases, one never can *guarantee* such a desired upper bound on the generalization error of a classifier from a given family $\mathcal{F}$ and a given training sample. Indeed, the training sample S is drawn from $\mathcal{D}$ at random, and thus has chances to be "pathological" (e.g., be of large cardinality, but with very close to each other, or even identical to each other, examples; given such a "low informative" sample, one hardly can build upon it a good classifier). To overcome this difficulty, the theoretical error bounds include a *(un)reliability parameter* δ, and a typical result on theoretical error bound looks as follows:

> (*) *Let X and $\mathcal{F}$ satisfy certain assumptions, let S be a training sample of cardinality ℓ, and let $f \in \mathcal{F}$ exhibit a certain behaviour on S. Then, up to probability of "bad sampling" $\leq \delta$, the generalization error of f does not exceed a certain quantity* (the latter quantity may depend on ℓ, $\mathcal{F}$, the quantities characterizing the behaviour of f on S *and* on δ).

Here the words "up to probability of bad sampling $\leq \delta$" mean the following. (*) means implication of the form "*If* f exhibits certain behaviour on S, *then* the generalization error of f does not exceed certain quantity". In general, the validity of this implication can depend on S – the implication can be true for "good" training samples and false for bad ones. Since S is random, we may speak about the "probability mass" of bad training samples (those for which the implication is false). The precise meaning of (*) is that the probability mass of bad training samples is $\leq \delta$.

2.2 Bounds on the generalization errors

2.2.1 Quantifying capacity of $\mathcal{F}$

From the Statistical Learning Theory we are interested to learn about the "capacity" of a prespecified class of potential classifiers. The capacity is measured by the *covering numbers* of the class, which are defined as follows.

Definition 2.1 [cf. [1], Definition 4.8] *Let X be a set, $\mathcal{F}$ be a family of real-valued functions defined on X, let $\gamma > 0$, and let ℓ be a positive integer.*

(i) *Given a sequence M of ℓ points $x_1, ..., x_\ell \in X$, we say that a finite subset $B \subset \mathcal{F}$ is a γ-net for M, if the restriction on M of every function $f \in \mathcal{F}$ can be approximated, within*

accuracy γ, by the restriction onto M of an appropriately chosen function $g \in B$. Thus, $B \subset \mathcal{F}$ is a γ-net for M if and only if

$$\forall (f \in \mathcal{F}) \exists g \in B : |f(x_i) - g(x_i)| < \gamma,\ i = 1, ..., \ell.$$

We define the quantity $\mathcal{M}(\mathcal{F}, M, \gamma)$ as the minimal, over all γ-nets for M, cardinality of the net.

(ii) *We define the ℓ-th covering number of $\mathcal{F}$ as the quantity*

$$\mathcal{N}(\mathcal{F}, \ell, \gamma) = \max_M \{\mathcal{M}(\mathcal{F}, M, \gamma) : M = (x_1, ..., x_\ell) \text{ with } x_i \in X\}$$

For X, $\mathcal{F}$ fixed, the covering numbers form a function of ℓ; this function depends on γ as on a parameter and clearly grows as $\gamma > 0$ decreases.

2.2.2 Quantifying behaviour of a classifier on a sample: margin and margin slacks vector

The behaviour of a potential classifier $f \in \mathcal{F}$ on a given training sample $S = \{(x_1, y_1), ..., (x_\ell, y_\ell)\}$ is quantified by its *margin slack vector* with respect to a given *margin* $\gamma > 0$. Those notions are defined as follows:

Definition 2.2 [cf. [1], Definition 4.20] *Let $f : X \to \mathbf{R}$, let $\gamma > 0$ be a real, and $S = \{(x_1, y_1), ..., (x_\ell, y_\ell)\}$ be training sample with ℓ examples.*

(i) *We say that f has margin γ on S, if*

$$y_i f(x_i) \geq \gamma,\ i = 1, ..., \ell$$

(that is, f separates by 2γ the positive ($y_i = 1$) and the negative ($y_i = -1$) examples from the training sample: $f(x_i) \geq \gamma$ for positive examples, and $f(x_i) \leq -\gamma$ for negative ones.)

(ii) *The margin slack vector ξ of f w.r.t. S, γ is defined as the vector from $\mathbf{R}^\ell$ with the coordinates*

$$\xi_i = \max[0, \gamma - y_i f(x_i)]$$

Note that the margin slack vector ξ of f w.r.t. S, γ is always nonnegative, and it is zero if and only if f has margin γ on S. ξ can be though of as the vector of smallest corrections in the values $f(x_i)$, $i = 1, ..., \ell$ ensuring that the corrected classifier has margin γ on S.

2.2.3 Bounds on generalization error in terms of margin and covering numbers

The first result of the Statistical Learning Theory which will be important for us is as follows:

Theorem 2.1 [[1], Theorem 4.9] *Let $\gamma > 0$, let $\mathcal{F}$ be a set of real-valued functions on X, $\mathcal{D}$ be a probability distribution on $X \times \{-1, 1\}$, let ℓ be a positive integer and $\delta > 0$. Then the following is true, up to $\mathcal{D}$-probability of bad sampling $\leq \delta$:*

if S is an ℓ-element training sample and f has margin γ on S, then

$$\mathrm{err}_{\mathcal{D}}(f) \leq \frac{2}{\ell} \left[\log_2 \left(\mathcal{N}(\mathcal{F}, 2\ell, \gamma/2) \right) + \log_2 \frac{2}{\delta} \right]. \tag{2.2.1}$$

Theorem 2.2.1 gives an upper bound on the generalization error in terms of the margin γ exhibited by a candidate classifier on the training sample; the larger is the margin, the smaller is the bound (recall that covering numbers $\mathcal{N}(\mathcal{F}, 2\ell, \gamma/2)$ can only decrease when γ increases). Thus, *Theorem suggests to quantify the behaviour of a candidate classifier f on training sample S by the margin of f on S and to take, as a classifier yielded by S, the classifier $f \in \mathcal{F}$ which has as large margin on S as possible.*

Bounding the covering numbers

Theorem 2.1 expresses the bounds on the generalization error in terms of classifier's behaviour on training sample (the margin) and the covering numbers of $\mathcal{F}$. To extract from this Theorem the actual error bounds, one needs to evaluate these latter quantities. This can be done in terms of *fat-shattering dimension* $\mathrm{fat}_{\mathcal{F}}(\gamma)$ defined as follows.

Definition 2.3 [cf. [1], Definition 4.12] *Let $\mathcal{F}$ be a family of real-valued functions on X and let $\gamma > 0$.*

(i) *Given a set $M = \{x_1, ..., x_\ell\} \subset X$ of cardinality ℓ, we say that M is γ-shattered by $\mathcal{F}$, if there exist reals $r_1, ..., r_\ell$ such that for every partition of the index set $I = \{1, ..., \ell\}$ into two non-overlapping subsets I_+ and I_- there exist $f \in \mathcal{F}$ such that*

$$f(x_i) \begin{cases} \geq r_i + \gamma, & i \in I_+ \\ < r_i - \gamma, & i \in I_- \end{cases}$$

(ii) *The fat-shattering dimension $\mathrm{fat}_{\mathcal{F}}(\gamma)$ is the largest ℓ such that there exists ℓ-element subset of X which is γ-shattered by $\mathcal{F}$.*

The fact that $\mathcal{F}$ γ-shatters M says that the restriction of $\mathcal{F}$ on M is a "rich enough" family of functions on M; namely, for every partition of M into two non-overlapping parts there exists a function $f \in \mathcal{F}$ such that $f(\cdot)$ is $\geq r(\cdot) + \gamma$ on the first part and $f(\cdot) < r(\cdot) - \gamma$ on the second part; here $r(\cdot)$ is a function associated with M (and thus independent of the partition). The definition of fat-shattering dimension makes sense when $\gamma = 0$, and $\mathrm{fat}_{\mathcal{F}}(0)$ is very similar to the famous *Vapnik-Chervonenkis* (VC) dimension of $\mathcal{F}$ (the latter is defined exactly in the same fashion as $\mathrm{fat}_{\mathcal{F}}(0)$, up to the fact that all r_i in (i) are set to 0).

The relation between the covering numbers and the fat-shattering dimension of $\mathcal{F}$ is given by the following proposition:

Proposition 2.1 [[1], Lemma 4.13] *Let $\mathcal{F}$ be a family of real-valued functions on X taking values in a finite segment $[a, b]$ on the axis. For $0 < \gamma < b - a$, and all integer $\ell \geq d \equiv \mathrm{fat}_{\mathcal{F}}(\gamma)$, one has*

$$\log(\mathcal{N}(\mathcal{F}, \ell, \gamma)) \leq 1 + d \log\left(2e\ell \cdot \frac{b-a}{\gamma}\right) \log\left(4\ell \frac{(b-a)^2}{\gamma^2}\right). \tag{2.2.2}$$

Proposition (2.1) says that as far as the dependence of the bound (2.2.2) on $\mathcal{F}$ is concerned, the bound is "nearly proportional" (that is, proportional up to a *logarithmic* factor) to the fat-shattering dimension $\mathrm{fat}_{\mathcal{F}}(\gamma)$ of $\mathcal{F}$. Thus, computing the upper bound (2.2.2) on the generalization error of a candidate classifier reduces to bounding the fat-shattering dimension $\mathrm{fat}_{\mathcal{F}}(\gamma)$ of $\mathcal{F}$. In the context of SVM, this latter problem should be solved for the simple case when $\mathcal{F}$ is comprised of affine functions, and this is the case we are about to consider.

2.3 Affine classifiers

From now on, we assume that our "universe" X is a bounded subset of real vector space E equipped with a norm $\|\cdot\|$, and that our family $\mathcal{F}$ of potential classifiers is the family $\mathcal{F}_{\rm Aff}(E, \|\cdot\|)$ of *normalized affine functions* on E.

The family $\mathcal{F}_{\rm Aff}(E, \|\cdot\|)$ is defined as follows. Let E^* be the space of continuous linear functionals on E. As usual, we denote the value of a functional $w \in E^*$ at a vector $x \in E$ by $\langle w, x\rangle$, so that $\langle w, x\rangle$ is bilinear in $w \in E^*$ and $x \in E$. Space E^* is equipped with the dual norm

$$\|w\|_* = \sup_{x\in E} \{\langle w, x\rangle : \|x\| \le 1\}.$$

Note that by definition of this norm, we have

$$|\langle w, x\rangle| \le \|w\|_*\|x\| \ \forall w, x; \tag{2.3.1}$$

in fact, for every $x \in E$ there exists $w \in E^*$, $\|w\|_* = 1$, which makes this inequality an equality (Hahn-Banach Theorem).

The family $\mathcal{F}_{\rm Aff}(E, \|\cdot\|)$ is comprised of all affine functions

$$f(x) = \langle w, x\rangle + b$$

with w running through the unit $\|\cdot\|_*$-ball of E^*:

$$\mathcal{F}_{\rm Aff}(E, \|\cdot\|) = \{f(x) = \langle w, x\rangle + b : w \in E^*, \|w\|_* \le 1\}. \tag{2.3.2}$$

Remark 2.1 *Note that in the context of binary classification, the classifiers $f(\cdot)$ and $\lambda f(\cdot)$, $\lambda > 0$, are exactly the same, since what counts, is* $\mathrm{sign}(f(\cdot))$. *It follows that "learning abilities" of the family of <u>all</u> (continuous) affine classifiers are exactly the same as those of classifiers from $\mathcal{F}_{\rm Aff}(E, \|\cdot\|)$. The only purpose of the normalization $\|w\|_* \le 1$ is to simplify the definition of the margin of a classifier on a training sample; without normalization, we were supposed to define the margin as* $\min_i f(x_i)/\|w\|_*$.

2.3.1 Bounding the fat-shattering dimension of the family of normalized affine functions

To the best of our knowledge, the traditional Statistical Learning Theory deals with computing the fat-shattering dimension of the family of normalized affine functions *only in the case when E is a Hilbert space and $\|\cdot\|$ is the inner product norm on E:*

$$\|x\| = \sqrt{(x, x)}.$$

Here $(\cdot, \cdot)$ denotes the inner product on E. In this case, E^* can be canonically identified with E; specifically, every $w \in E$ defines a continuous linear form $(w, \cdot)$ on E, and the resulting mapping of E into E^* is a norm preserving one-to-one correspondence between E and E^*. In the sequel, we extend the results on fat-shattering dimension of $\mathcal{F}_{\rm Aff}$ from the case of Hilbert space E to a wider family of normed spaces, the so called *spaces of type 2*, or, as we prefer to call them, *κ-regular* normed spaces.

Regular normed spaces. Let E be a normed space with a norm $\|\cdot\|$, and let $\pi(x)$ be an equivalent norm on E, that is, $\pi(\cdot)$ is another norm on E satisfying the relation

$$\forall x \in E : \|x\|^2 \leq \pi^2(x) \leq c_\pi \|x\|^2, \tag{2.3.3}$$

with certain constant $c_\pi < \infty$. The notion of a *κ-regular* normed space $(E, \|\cdot\|)$ is defined as follows (see [4]):

Definition 2.4 *Let $(E, \|\cdot\|)$ be a normed space.*

(i) *Let $\pi(\cdot)$ be a norm on E which is equivalent to $\|\cdot\|$, and r be a positive real. We say that $\pi(\cdot)$ is r-smooth, if the function $P(x) = \pi^2(X)$ is continuously Frechet differentiable and satisfies the inequality*

$$\forall (x, z \in E) : P(x+z) \leq P(x) + \langle P'(x), z\rangle + rP(z). \tag{2.3.4}$$

(ii) *Let κ be a positive real. We say that $\|\cdot\|$ is κ-regular, if there exists r and r-smooth norm $\pi(\cdot)$ on E such that $rc_\pi \leq \kappa$, where c_π is the "compatibility constant" of $\pi(\cdot)$ and $\|\cdot\|$, see* (2.3.3).

Here are basic examples of regular normed spaces:

1. *Hilbert space.* The inner product norm $\|\cdot\|_E$ on a Hilbert space E clearly is 1-smooth. Consequently, a Hilbert space is 1-regular (take $\pi(\cdot) \equiv \|\cdot\|_E$).
   Note that Hilbert spaces are "as regular as a normed space could be". Indeed, setting $x = 0$ in (2.3.4), we see that $r \geq 1$; from (2.3.3) follows that $c_\pi \geq 1$ as well, so that nontrivial (that is, of positive dimension) κ-regular spaces with $\kappa < 1$ do not exist.

2. *Spaces $\ell_p(M)$, $2 \leq p < \infty$.* Let M be a set, and let $1 \leq p < \infty$. The space $\ell_p(M)$ is comprised of all real-valued functions $\phi(\cdot)$ on M with countable supports $\{\mu \in M : \phi(\mu) \neq 0\}$ and finite $\|\cdot\|_p$-norms
   $$\|\phi(\cdot)\|_p = \left(\sum_{\mu \in M} |\phi(\mu)|^p\right)^{1/p};$$
   the linear operations on $\ell_p(M)$ are defined in the standard point-wise fashion. The space $\ell_\infty(M)$ is defined similarly, up to the fact that now we define the norm (and require it to be finite) as
   $$\|\phi(\cdot)\|_\infty = \sup_{\mu \in M} |\phi(\mu)|.$$
   It is easily seen (see [4]) that *when $2 \leq p < \infty$, the norm $\|\cdot\|_p$ is r-smooth with $r = p - 1$.* Thus,

   > *The spaces $\ell_p(M)$, $2 \leq p < \infty$, are $(p-1)$-regular.*

   Note that the Hilbert space case is covered by the latter result, where one should set $p = 2$.

3. Similarly to the previous item, *the functional spaces $L_p(\Omega)$ associated with a measure μ on a closed subset $\Omega \subset \mathbf{R}^N$ are $(p-1)$ regular whenever $2 \leq p < \infty$.*

4. *The space $M^{m,n}$ of $m \times n$ matrices $(m, n < \infty)$ equipped with the norm $|A| = \|\sigma(A)\|_p$,* where $\sigma(A)$ is the vector of singular values of A, *is $(2p-1)$-regular, provided $2 \leq p < \infty$.*

The *upper bounds on the* "level of regularity" κ in the above examples deteriorates when $p \to \infty$. This indeed happens in the case of an infinite-dimensional space; however, in the *finite-dimensional* situation κ can be bounded, uniformly in $2 \le p \le \infty$, solely in terms of the dimension, and this bound extremely slowly – logarithmically - deteriorates with the dimension, as can be seen from the following examples:

1. *Space $\ell_p(M)$, $2 \le p \le \infty$, associated with a finite set M, is κ-regular with*

$$\kappa \le O(1)\min\{p-1, \log(\mathrm{Card}(M)+1)\}, \tag{2.3.5}$$

where $O(1)$ is an appropriate absolute constant.
Indeed, we already know that $\ell_p(M)$ is $(p-1)$-regular independently of whether M is or is not finite. Now let M be of finite cardinality N. W.l.o.g. we may assume that $N \ge \exp\{2\}$, since otherwise (2.3.5) is evident. When $2 \le p \le \log N$, (2.3.5) holds true, since the minimum in the right hand side is $p-1$. When $p > \log N \ge 2$, we can use the evident relation

$$2 \le s \le s' \le \infty \Rightarrow \|x\|_s \le \|x\|_{s'} \le N^{\frac{1}{s'}-\frac{1}{s}},\ x \in \mathbf{R}^N$$

to conclude that when choosing $\pi(x) = \|x\|_{\log N}$, (2.3.3) holds true with an absolute constant (namely, $\exp\{2\}$) in the role of c_π. Since the norm $\|\cdot\|_{\log N}$ is, as we already know, $(\log N - 1)$-smooth, we conclude that $\|\cdot\|_p$ is $O(1)\log N$-regular.

2. *Space $M^{m,k}$ of $m \times k$ matrices equipped with the norm $|\cdot|_p$, $2 \le p \le \infty$, is κ-regular with*

$$\kappa = O(1)\min\{p, \log(\min[m,k]+1)\}.$$

An important fact on κ-regular normed spaces is as follows:

Proposition 2.2 [see [4]] *Let E, $\|\cdot\|$ be κ-regular, and let η_i, $i = 1, ..., n$, be independent random vectors taking values in E and such that*

$$\mathbf{E}\{\eta_i\} = 0,\ \mathbf{E}\{\|\eta_i\|^2\} < \infty,\ i = 1, ..., n.$$

Then

$$\mathbf{E}\left\{\|\sum_{i=1}^n \eta_i\|^2\right\} \le \kappa \sum_{i=1}^n \mathbf{E}\{\|\eta_i\|^2\}. \tag{2.3.6}$$

Proof. Let us set $S_i = \sum_{s=1}^i \eta_s$, $i = 1, ..., n$, and let $\pi(\cdot)$ be an r-smooth norm on E satisfying (2.3.3) and such that $rc_\pi \le \kappa$. Setting $P(x) = \pi^2(x)$, we have

$$\begin{aligned}
\mathbf{E}\{P(S_{i+1})\} &\underbrace{\le}_{(a)} \mathbf{E}\left\{P(S_i) + \langle P'(S_i), \eta_{i+1}\rangle + rP(\eta_{i+1})\right\} \\
&\underbrace{=}_{(b)} \mathbf{E}\left\{P(S_i) + rP(\eta_{i+1})\right\} \\
&\underbrace{\le}_{(c)} \mathbf{E}\left\{P(S_i)\right\} + rc_\pi \mathbf{E}\left\{\|\eta_{i+1}\|^2\right\},
\end{aligned}$$

where (a) is given by (2.3.4), (b) comes from the fact that $\mathbf{E}\{\eta_{i+1}\} = 0$ and η_{i+1} is independent of S_i, and (c) comes from the definition of $P(\cdot)$ and from the second inequality in (2.3.3). From the resulting recurrence and the fact that $rc_\pi \le \kappa$ it follows that

$$\mathbf{E}\left\{\pi^2(\sum_{i=1}^n \eta_i)\right\} \le \kappa \sum_{i=1}^n \mathbf{E}\{\|\eta_i\|^2\}.$$

Invoking the left inequality in (2.3.3), formula (2.3.6) follows. ∎

Bounding the fat-shattering dimension. We are now in a position to estimate the fat-shattering dimension of $\mathcal{F}_{\rm Aff}(E, \|\cdot\|)$ w.r.t. $X \subset E$. The result is as follows:

Theorem 2.2 *Let $(E, \|\cdot\|)$ be a κ-regular space, and let X be a subset of a $\|\cdot\|$-ball, centered at the origin, of radius R. Then*

$$\mathrm{fat}_{\mathcal{F}_{\rm Aff}(E,\|\cdot\|)}(\gamma) \le 18\kappa \frac{R^2}{\gamma^2}. \tag{2.3.7}$$

Proof mimics, with some modifications, the proof of Theorem 4.16 in [1].

1^0. Let us set $E_+ = E \times \mathbf{R}^2$, so that a generic element of E^+ is (x, u) with $x \in E$ and $u \in \mathbf{R}^2$. We equip E_+ with the norm

$$\|(x, u)\| = \sqrt{\|x\|^2 + \|u\|_2^2};$$

note that the dual to E_+ space E_+^* clearly can be identified with $E^* \times \mathbf{R}^2$, with

$$\langle (w, v), (x, u) \rangle = \langle w, x \rangle + v^T u$$

and

$$\|(w, v)\|_* = \sqrt{\|w\|_*^2 + \|v\|_2^2}.$$

Moreover, E_+ is κ-regular. Indeed, if $\pi(\cdot)$ is an r-smooth norm on E which is c_π-compatible with the norm $\|\cdot\|$ on E in the sense of (2.3.3) and $rc_\pi \le \kappa$, then the norm

$$\pi_+((x, u)) = \sqrt{\pi^2(x) + \|u\|_2^2}$$

on E_+ clearly is r-smooth and satisfies the relation

$$\|(x, u)\| \le \pi_+((x, u)) \le c_\pi \|(x, u)\| \ \forall (x, u) \in E_+,$$

and therefore E_+ is κ-regular.

2^0. Now let $\gamma > 0$, and let $M = \{x_1, ..., x_\ell\}$ be a subset of X which is γ-shattered by $\mathcal{F} \equiv \mathcal{F}_{\rm Aff}(E, \|\cdot\|)$, and let r_i, $i = 1, ..., \ell$, be the associated reals given by Definition 2.3. We should prove that ℓ does not exceed the right hand side of (2.3.7); there is nothing to prove when $\ell = 1$, thus, we assume that $\ell \ge 2$.

We start with several simple observations.

(a) The property of the pair $M, \{r_1, ..., r_\ell\}$ expressed in Definition 2.3 remains unchanged when we simultaneously shift all r_i's by the same quantity Δ.

Indeed, let $r_i^+ = r_i + \Delta$, and let $I_+ \cup I_-$ be a partition of the index set $I = \{1, ..., \ell\}$; we should verify that there exists an affine function

$$f_+(x) = \langle w, x \rangle + b_+, \ \|w\|_* \le 1$$

satisfying the relations

$$f_+(x_i) \begin{cases} \ge r_i^+ + \gamma, & i \in I_+ \\ < r_i^+ - \gamma, & i \in I_- \end{cases}, \tag{2.3.8}$$

observe that due to the origin of $M, \{r_1, ..., r_\ell\}$, there exists an affine function

$$f(x) = \langle w, x \rangle + b, \ \|w\|_* \le 1,$$

which satisfies relations of the same type as in (2.3.8), but with $r_i^+ = r_i + \Delta$ replaced with r_i. It follows that setting $b_+ = b + \Delta$, we get a function which indeed satisfies (2.3.8). □

In view of (a), we can assume w.l.o.g. that

$$\rho \equiv \max r_i = -\min r_i \tag{2.3.9}$$

(to this end, it suffices to shift r_i's by $\Delta = -\frac{1}{2}\left[\min\limits_i r_i + \max\limits_i r_i\right]$).

(b) We have $\rho + \gamma \le R$.

Indeed, recall that $\ell \ge 2$, so that w.l.o.g. we may assume that $-\rho = r_1 \le r_2 \le ... \le r_\ell = \rho$. Consider a partitioning $I = I_+ \cup I_-$ such that $1 \in I_-$ and $\ell \in I_+$. There should exist a function

$$f(x) = \langle w, x\rangle + b$$

such that $f(x_1) < r_1 - \gamma \equiv -\rho - \gamma$ and $f(x_\ell) \ge r_\ell + \gamma = \rho + \gamma$, whence

$$2(\rho+\gamma) \le f(x_\ell) - f(x_1) = \langle w, x_\ell - x_1\rangle \underbrace{\le}_{(*)} \|w\|_* \|x_\ell - x_1\| \underbrace{\le}_{(**)} 2R,$$

where $(*)$ comes from the definition of the dual norm, and $(**)$ - from the fact that $x_i \in X$ and X is contained in the $\|\cdot\|$-ball of radius R centered at the origin. (b) is proven. □

3^0. Now let us "lift" x_i from E to E_+ by setting

$$x_i^+ = (x_i, (r_i, R)),\ i = 1, ..., \ell,$$

so that

$$\|x_i^+\| = \sqrt{\|x_i\|^2 + r_i^2 + R^2} \le R\sqrt{3}, \tag{2.3.10}$$

(since $|r_i| \le \rho \le R$ by (2.3.9) and (b)).

Let us fix a partition $I = I_+ \cup I_-$. There exists an affine function

$$f(x) = \langle w, x\rangle + b,\ \|w\|_* \le 1,$$

such that

$$\langle w, x_i\rangle + b \begin{cases} \ge r_i + \gamma, & i \in I_+ \\ < r_i - \gamma, & i \in I_- \end{cases} \tag{2.3.11}$$

We claim that the constant term b in this affine function can be modified in such a way that the new function still satisfies (2.3.11), but the updated constant term, let it be called b', satisfies $|b'| \le 2R$. Indeed, when i runs through I, the right hand side quantities in (2.3.11) remain, by (b), in the segment $[-R, R]$, while the quantities $\langle w, x_i\rangle$ remain in the same segment due to $\|w\|_* \le 1$, $\|x_i\| \le R$ (see (2.3.1)). It follows that when "projecting" b onto $[-2R, 2R]$, that is, replacing b with $2R$ in the case of $b > 2R$, replacing b with $-2R$ in the case of $b < -2R$ and keeping b intact in all remaining cases, we preserve validity of (2.3.11). □

Assuming $|b| \le 2R$, let us associate with f the functional

$$w^+ = (w, (-1, b/R)) \in E_+^*.$$

Observe that

$$\|(w, (-1, b/R))\|_* = \sqrt{\|w\|_*^2 + 1 + b^2/R^2} \le \sqrt{6} \tag{2.3.12}$$

(recall that $|b| \leq 2R$ and $\|w\|_* \leq 1$).

Now let us set

$$x^+_{I_+,I_-} = \sum_{i\in I_+} x_i^+ - \sum_{i\in I_-} x_i^+ \in E^+$$

and observe that

$$\begin{array}{rcl} \|w^+\|_*\|x^+_{I_+,I_-}\| & \geq & \langle w^+, x^+_{I_+,I_-}\rangle = \sum\limits_{i\in I_+}[\langle w, x_i\rangle - r_i + b] - \sum\limits_{i\in I_-}[\langle w, x_i\rangle - r_i + b] \\ & \geq & \ell\gamma, \end{array}$$

where the first $\geq$ is given by (2.3.1), and the last one – by (2.3.11). Invoking (2.3.12), we conclude that

$$\|x^+_{I_+,I_-}\| \geq \frac{\ell\gamma}{\sqrt{6}}. \tag{2.3.13}$$

4^0. Relation (2.3.13) says that *for every collection of reals $s_i \in \{-1, 1\}$, one has*

$$\|\sum_{i=1}^{\ell} s_i x_i^+\| \geq \frac{\ell\gamma}{\sqrt{6}};$$

in particular, *if ξ_i are independent of each other random variables taking values ± 1 with probability $1/2$, then*

$$\mathbf{E}\left\{\|\sum_{i=1}^{\ell}\xi_i x_i^+\|^2\right\} \geq \frac{\ell^2\gamma^2}{6}. \tag{2.3.14}$$

On the other hand, E_+, $\|\cdot\|$ is κ-regular, whence, applying Proposition 2.2 to random vectors $\eta_i = \xi_i x_i^+$ and invoking (2.3.10), we get

$$\mathbf{E}\left\{\|\sum_{i=1}^{\ell}\xi_i x_i^+\|^2\right\} \leq \kappa\sum_{i=1}^{\ell}\|x_i^+\|^2 \leq 3R^2\kappa\ell.$$

Combining this relation with (2.3.14), we arrive at (2.3.7). ∎.

Note that the structure of the above proof is identical to the one for the case when E is a Hilbert space (see the proof of Theorem 4.16 in [1]). The only (but crucial) role of the assumption that E is κ-regular is to enable the randomized argument in 4^0, which in the case of inner product norm is given "for free" by specific algebraic properties of the norm.

2.3.2 Bounding generalization error via margin

When considering the family $\mathcal{F}_{\mathrm{Aff}}(E, \|\cdot\|)$ of affine classifiers in the case when X is inside the centered at the origin $\|\cdot\|$-ball of E, the margin γ of a whatever classifier on a whatever training sample $S = \{(x_1, y_1), ..., (x_\ell, y_\ell)\}$ containing both positive and negative examples can not be larger than R. Indeed, if i, j are such that $y_i = -1$ and $y_j = 1$ and $f(x) = \langle w, x\rangle + b$ is a classifier from the family with margin $\geq \gamma$ on S, then we should have

$$\langle w, x_i\rangle + b \leq -\gamma,\ \langle w, x_j\rangle + b \geq \gamma,$$

whence $2\gamma \leq \langle w, x_j - x_i\rangle$; the latter quantity is $\leq 2R$ due to $\|w\|_* \leq 1$ and $\|x_j - x_i\| \leq 2R$, whence $\gamma \leq R$. It follows that when S is a training sample with both positive and negative

examples and $f(x) = \langle w, x\rangle + b$ is a classifier from $\mathcal{F}_{\rm Aff}(E, \|\cdot\|)$ with margin $\geq \gamma$, then $|b| \leq R$, since with i and j as above we should have

$$b \leq -\gamma - \langle w, x_i\rangle \leq R,\ b \geq \gamma - \langle w, x_j\rangle \geq -R.$$

It follows that *when $X \subset \{x \in E : \|x\| \leq R\}$, we lose nothing when restricting $\mathcal{F}_{\rm Aff}(E, \|\cdot\|)$ to*

$$\mathcal{F} \equiv \mathcal{F}^R_{\rm Aff}(E, \|\cdot\|) = \{f(x) = \langle w, x\rangle + b : \|w\|_* \leq 1, |b| \leq R\}.$$

Note that the classifiers from the latter family, when restricted to X, take their values in the segment $[-2R, 2R]$, so that Proposition 2.1 implies the following upper bound on the covering numbers of the family:

$$\log\mathcal{N}(\mathcal{F}, \ell, \gamma) \leq O(1){\rm fat}_{\mathcal{F}}(\gamma)\log^2\left(\ell\frac{R}{\gamma}\right),\ 0 < \gamma \leq R;$$

from now on, $O(1)$ are appropriate absolute constants. Combining this bound and Theorems 2.1, 2.2, we arrive at the following

Theorem 2.3 [cf. [1], Theorem 4.18] *Let $E, \|\cdot\|$ be a κ-regular normed space, let X be a subset of the centered at the origin $\|\cdot\|$-ball of radius R, and let $\gamma \in (0, R]$. Further, let $\mathcal{D}$ be a probability distribution on $X \times \{-1, 1\}$, ℓ be a positive integer and $\delta \in (0, 1)$. Then the following is true, up to $\mathcal{D}$-probability of bad sampling $\leq \delta$:*

If $S = \{(x_1, y_1), ..., (x_\ell, y_\ell)\}$ is an ℓ-element training sample drawn from $\mathcal{D}$ and an affine function

$$f(x) = \langle w, x\rangle + b$$

with $\|w\|_* \leq 1$ and $|b| \leq R$ has margin γ on S:

$$y_i f(x_i) \geq \gamma\ i = 1, ..., \ell,$$

then the generalization error of the classifier $\mathrm{sign}(f(x))$ *admits upper bound as follows:*

$$\mathrm{err}_{\mathcal{D}}(f) \leq \frac{O(1)}{\ell}\left[\kappa\frac{R^2}{\gamma^2}\log^2\left(\ell\frac{R}{\gamma}\right) + \log\frac{2}{\delta}\right]. \tag{2.3.15}$$

Comments. In the case of Hilbert space E with the inner product norm $\|\cdot\|$, E^* is canonically identified with E; with this identification, $\|\cdot\|_*$ becomes $\|\cdot\|$, and Theorem 2.3 becomes Theorem 4.18 in [1], which is one of the well-known standard results of Machine Learning theory. The power of the latter result in the SVM context comes from the fact that the associated upper bound on the generalization error is "dimension-independent" – what matters, is solely the ratio R/γ of the Hilbert ball containing the set of tentative feature vectors to the geometric margin γ exhibited by the (normalized) affine classifier on the training sample. The dimension-independent nature of the result allows to work with extremely high-dimensional, and even infinite-dimensional, Hilbert feature spaces. Theorem 2.3 extends this option to a much wider family of feature spaces. As an extreme example, consider the case where E is the space $\ell_\infty(M)$ associate with a *finite* set M, that is, E is the space $\mathbf{R}^N$ of the dimension $N = \mathrm{Card}(M)$ equipped with the uniform norm $\|\cdot\|_\infty$. Here E^* can be naturally identified with $E = \mathbf{R}^N$; with this identification, $\langle\xi, x\rangle$ becomes just $\xi^T x$, and $\|\cdot\|_*$ becomes the norm $\|\cdot\|_1$ on $E^* = E = \mathbf{R}^N$. The problem of finding a normalized classifier with the largest possible geometric margin on

the training sample becomes now the popular SVM model with $\|\cdot\|_1$-normalization of tentative classifiers. Theorem 2.3 states that the "generalization abilities" of this model are, up to factor $\log N$, the same as for the standard "Euclidean" model with $\|\cdot\|_2$ in the roles of $\|\cdot\|$ and $\|\cdot\|_*$. Thus, unless the dimension of the feature space becomes astronomically large, $\|\cdot\|_1$-SVMs (that is, SVM models obtained from (1.1.3) by replacing the objective $\|w\|_2^2 \equiv \langle w, w\rangle$ with $\|w\|_1^2$) are nearly as theoretically valid as the standard $\|\cdot\|_2$-SVMs (1.1.3).

> In this respect, it is worthy to mention that in a $\|\cdot\|_1$-SVM we are seeking to separate the sets $S_+ = \{x_i : y_i = 1\}$ and $S_- = \{x_i : y_i = -1\}$ of the positive and the negative training input vectors by a stripe with the largest possible $\|\cdot\|_\infty$-width, while in the $\|\cdot\|_2$-model the goal is to find the separating strip of the largest $\|\cdot\|_2$-width. One could think that since the $\|\cdot\|_\infty$-width of a strip always is $\le$ the $\|\cdot\|_2$-width, the $\|\cdot\|_1$-SVM always is inferior as compared to the $\|\cdot\|_2$-one – the numerical value γ_1 of the geometric margin in the former model always is $\le$ the value γ_2 of the margin in the latter model. This conclusion, however, overlooks the fact that what matters is not the numerical value of the geometric margin itself, but the ratio of this value to the radius R of the smallest ball containing the set X of tentative input vectors. For $\|\cdot\|_1$-SVM, $R = R_1$ should be taken w.r.t. the norm $\|\cdot\|_\infty$, and for $\|\cdot\|_2$-SVM, $R = R_2$ should be taken w.r.t. the norm $\|\cdot\|_2$. It is clear that $R_1 \le R_2$, so that in the ratios γ_1/R_1 and γ_2/R_2 to be compared *both* numerators and denominators are linked by similar inequalities. As a result, we cannot say in advance which one of the ratios is larger (that is, which one of the SVM models is better from the viewpoint of the generalization error); the answer depends on the particular geometry of X, and there are cases when $\|\cdot\|_1$-SVM is much better than the $\|\cdot\|_2$-one. This observation, we believe, makes it clear that our extension of bounds on generalization error from the case of a Hilbert feature space to the case of a regular feature space is more than just an academic exercise.

Similar comments are applicable to Theorem 2.4 below which handles the case of soft margin.

2.3.3 Bounding generalization error via margin and margin slacks

The bounds on generalization error given by Theorem 2.3 do not help much when a "typical" training sample does not admit affine classifiers with reasonable margin (or even with a positive one). The latter situation arises in many applications, and in this case one is interested in bounds on generalization error expressed via margin *and* margin slack vector of an affine classifier. We are about to derive bounds of this type, following the approach presented in [1], Section 4.3, which we extend from the "Hilbert space case" to the one where X is a bounded subset of a κ-regular normed space.

Assumptions

From now on we assume that X is a bounded (not necessary finite) subset of a κ-regular normed space $(E, \|\cdot\|)$, moreover, that we know in advance $R > 0$ such that X is contained in the $\|\cdot\|$-ball, centered at the origin, of radius R. Besides this, we make the following

> Assumption A: *The distribution $\mathcal{D}$ to be learnt is such that if (x, y), (x', y') are two random examples drawn independently from the distribution, then the probability of the event $\{x = x', y \ne y'\}$ is 0.*

Assumption A implies that the probability to draw from $\mathcal{D}$ a "contradictory" training sample $S = \{(x_1, y_1), ..., (x_\ell, y_\ell)\}$ – such that $x_i = x_j$ and $y_i \ne y_j$ for certain pair i, j – equals to 0.

Preliminaries

Extending E to a wider space. Let us fix $p \in [2, \infty]$ (it is the parameter of construction to follow), with the restriction that the choice $p = \infty$ is allowed only in the case when X is finite. Let us set

$$\widehat{E}_p = E \times \ell_p(X),$$

where, as above, $\ell_p(X)$ is the space of all functions $g : X \to \mathbf{R}$ for which the norm

$$\|g\|_p = \begin{cases} \left(\sum\limits_{x \in X} |g(x)|^p \right)^{1/p}, & p < \infty \\ \sup\limits_{x \in X} |g(x)|, & p = \infty \end{cases}$$

is finite (which, in the case of infinite X, implies that the set of points $x \in X$ where $g(x) \neq 0$ is countable). A generic element of E is a pair (x, g) with $x \in E$ and $g(\cdot) \in \ell_p(X)$, and we equip $\widehat{E}_p$ with the norm

$$\|(x, g)\| = \sqrt{\|x\|^2 + \|g\|_p^2} \quad {}^{1)} \tag{2.3.16}$$

It is easily seen that the dual to $\widehat{E}_p$ space is

$$\widehat{E}_p^* = E_* \times \ell_{p_*}(X), \; p_* = \frac{p}{p-1}, \quad {}^{2)}$$

with the value of a functional $(w, h) \in \widehat{E}_p^*$ on a vector $(x, g) \in \widehat{E}_p$ given by

$$\langle (w, h), (x, g) \rangle = \langle w, x \rangle + \sum_{u \in X} h(u) g(u),$$

and the dual norm given by

$$\|(w, h)\|_* = \sqrt{\|w\|_*^2 + \|h\|_{p_*}^2}. \tag{2.3.17}$$

Note that $(E, \|\cdot\|)$ is κ-regular by assumption, while $\ell_p(X)$ is $O(1)\min[p-1, \mathrm{logCard}(X)]$-regular (see the list of examples of regular spaces in Section 2.3.1). In view of these observations, a reasoning completely similar to the one in item 1^0 of the proof of Theorem 2.1 shows that $\widehat{E}_p$ is $\widehat{\kappa}(p)$-regular, where

$$\widehat{\kappa}(p) \leq O(1)\kappa \min[p-1, \mathrm{logCard}(X)]; \tag{2.3.18}$$

in fact, the aforementioned reasoning demonstrates that when E is any one of the regular spaces listed in Section 2.3.1, $\widehat{\kappa}(p)$ admits a better bound:

$$\widehat{\kappa}(p) \leq O(1)\left[\kappa + \min[p-1, \mathrm{logCard}(X)]\right]. \tag{2.3.19}$$

[1)] To avoid messy expressions, we use the same notation $\|\cdot\|$ for norms in E and in $\widehat{E}$; which one of these norms is meant, will be alway clear from the context.

[2)] Recall that X is finite when $p = \infty$.

Embedding X into $\widehat{E}_p$. For $x \in X$, let $\delta_x(\cdot)$ be the function on X given by

$$\delta_x(x') = \begin{cases} 1, & x = x' \\ 0, & x \neq x' \end{cases}$$

Consider the embedding of X into $\widehat{E}_p$ given by

$$x \mapsto \widehat{x} = (x, R\delta_x(\cdot)),$$

and let $\widehat{X} \subset \widehat{E}_p$ be the image of X under this embedding. Note that since X belongs to the centered at the origin $\|\cdot\|$-ball in E of radius R, one has

$$\widehat{x} \in \widehat{X} \Rightarrow \|\widehat{x}\| \leq \widehat{R} \equiv \sqrt{2}R. \tag{2.3.20}$$

Note that an example (x, y) with $x \in X$ induces example $(\widehat{x}, y)$ with $\widehat{x} \in \widehat{X}$, so that the distribution $\mathcal{D}$ to be learnt induces a distribution $\widehat{\mathcal{D}}$ on $\widehat{X} \times \{-1, 1\}$, and a training sample $S = \{(x_1, y_1), ..., (x_\ell, y_\ell)\}$ drawn from $\mathcal{D}$ induces training sample $\widehat{S} = \{(\widehat{x}_1, y_1), ..., (\widehat{x}_\ell, y_\ell)\}$ drawn from $\widehat{\mathcal{D}}$.

Augmenting affine classifiers. Let

$$f(x) = \langle w, x\rangle + b$$

be an affine classifier on E, $S = \{(x_1, y_1), ..., (x_\ell, y_\ell)\}$ be a training sample with $x_i \in X$ which is "non-contradictory" (that is, such that $x_i = x_j$ implies $y_i = y_j$) and $\gamma > 0$ be a margin, and ξ be the slack vector of f w.r.t. S, γ (Definition 2.2), that is,

$$\xi_i = \max[0, \gamma - y_i f(x_i)],\ i = 1, ..., \ell. \tag{2.3.21}$$

We associate with f, S, γ the *augmented classifier* – the affine function $\widehat{f}(\cdot)$ on $\widehat{E}_p$ defined as follows:

• In the case when $x_1, ..., x_\ell$ differ from each other, we set

$$\widehat{f}((x, g)) = f(x) + \langle \sum_{i=1}^{\ell} R^{-1} y_i \xi_i \delta_{x_i}(\cdot), g(\cdot)\rangle \equiv f(x) + \sum_{i=1}^{\ell} R^{-1} y_i \xi_i g(x_i). \tag{2.3.22}$$

• If not all of the vectors $x_1, ..., x_\ell$ are distinct, we partition the index set $I = \{1, ..., \ell\}$ into appropriate number of subsets $I_1, ..., I_{\ell'}$ in such a way that $x_i = x_j$ when i, j belong to the same subset and $x_i \neq x_j$ when i, j belong to different subsets. Note that since S is non-contradictory, we have $\xi_i = \xi_j$ for all i, j belonging to the same subset of the partition; let ξ^s be the common value of ξ_i, $i \in I_s$, x^s be the common value of x_i, $i \in I_s$, and y^s be the common value of y_i, $i \in I_s$. We set

$$\widehat{f}((x, g)) = f(x) + \langle \sum_{s=1}^{\ell'} R^{-1} y^s \xi^s \delta_{x^s}(\cdot), g(\cdot)\rangle \equiv f(x) + \sum_{s=1}^{\ell'} R^{-1} y^s \xi^s g(x^s). \tag{2.3.23}$$

Note that (2.3.22) is a particular case of (2.3.23) corresponding to the situation when $s = \ell$ and all I_s are singletons.

We make the following simple observation:

(!) *Let $\widehat{S} = \{(\widehat{x}_1, y_1), ..., (\widehat{x}_\ell, y_\ell)\}$ be the training sample on $\widehat{X}$ associated with S. Then*

1. *The classifier $\widehat{f}$ has margin $\geq \gamma$ on $\widehat{S}$;*
2. *If $x \in X$ differs from $x_1, ..., x_\ell$, then*
$$\widehat{f}(\widehat{x}) = f(x) \equiv \langle w, x\rangle + b;$$
3. *One has*
$$\widehat{f}((x,g)) = \langle \widehat{w}, (x,g)\rangle + b, \tag{2.3.24}$$
where
$$\widehat{w} = (w, R^{-1}\sum_{s=1}^{\ell'} y^s \xi^s \delta_{x^s}(\cdot)),$$
and therefore
$$\|\widehat{w}\|_* \leq \sqrt{\|w\|_*^2 + R^{-2}\|\xi\|_{p_*}^2}. \tag{2.3.25}$$

Indeed, let $i \leq \ell$ and $s_i \leq \ell'$ be such that $i \in I_{s_i}$. Then

$$\begin{array}{rcl} y_i\widehat{f}(\widehat{x}_i) & = & y_i f(x_i) + y_i\langle R^{-1}\sum\limits_{s=1}^{\ell'} y^s\xi^s\delta_{x^s}(\cdot), R\delta_{x_i}(\cdot)\rangle \\ & = & y_i f(x_i) + y_i\sum\limits_{s=1}^{\ell'} y^{s_i}\xi^{s_i}\delta_{x^s}(x_i) \\ & \underbrace{=}_{(a)} & y_i f(x_i) + y_i y^{s_i}\xi^{s_i} = y_i f(x_i) + \xi_i \underbrace{\geq}_{(b)} \gamma, \end{array}$$

where equality (a) follows from the fact that $\delta_{x^s}(x_i) \neq 0$ if and only if $s = s_i$, in which case $\delta_{x^s}(x_i) = 1$, $y^s = y_i$ and $\xi^s = \xi_i$, while inequality (b) is given by (2.3.21). We have proven item 1 of (!). Items 2 and 3 are evident. □

Bounding generalization error via margin and margin slacks

We now are in a position to prove the following extension of Theorem 2.3:

Theorem 2.4 [cf. [1], Theorems 4.22, 4.24] *Let $(E, \|\cdot\|)$ be a κ-regular normed space, and let X be a subset of the centered at the origin $\|\cdot\|$-ball of radius R. Let us fix $p \in [2, \infty]$ with $p < \infty$ when X is infinite, and real $\gamma \in (0, \sqrt{2}R]$.*

Further, let $\mathcal{D}$ be a probability distribution on $X \times \{-1, 1\}$ satisfying Assumption A, ℓ be a positive integer and $\delta \in (0, 1)$. Then the following is true, up to $\mathcal{D}$-probability of bad sampling $\leq \delta$:

An ℓ-element training sample $S = \{(x_1, y_1), ..., (x_\ell, y_\ell)\}$ drawn from $\mathcal{D}$ is non-contradictory, and if an affine function
$$f(x) = \langle w, x\rangle + b$$
with $|b| \leq \sqrt{2}R$ has slack variable vector ξ w.r.t. γ, S:
$$\xi_i = \max[0, \gamma - y_i f(x_i)],\ i = 1, ..., \ell,$$
and the following norm bound holds true:
$$\sqrt{\|w\|_*^2 + R^{-2}\|\xi\|_{p_*}^2} \leq 1,\ p_* = \frac{p}{p-1}, \tag{2.3.26}$$

then the generalization error of the classifier $\mathrm{sign}(\widehat{f}(\widehat{x}))$*, where* $\widehat{f}(\cdot)$ *is the augmentation of* $f(\cdot)$ *as defined by* S, γ*, satisfies the bound*

$$\mathrm{err}_{\widehat{\mathcal{D}}}(\widehat{f}) \leq \frac{O(1)}{\ell}\left[\widehat{\kappa}(p)\frac{R^2}{\gamma^2}\log^2\left(\ell\frac{R}{\gamma}\right) + \log\frac{2}{\delta}\right]. \tag{2.3.27}$$

Here $\widehat{\kappa}(p)$ *is given by* (2.3.18) *(general case) or by* (2.3.19)*, if* $(E, \|\cdot\|)$ *is any one of the regular spaces listed in Section 2.3.1.*

Proof. The fact that a random sample S drawn form $\mathcal{D}$ is non-contradictory with probability 1 is readily given by Assumption A. The remaining statements follow directly from Theorem 2.3 as applied to $\widehat{E}_p$ and the norm (2.3.16) in the role of $(E, \|\cdot\|)$, $\widehat{X}$ in the role of X, $\widehat{\mathcal{D}}$ in the role of $\mathcal{D}$, $\widehat{S}$ in the role of S, $\widehat{R} = \sqrt{2}R$ in the role of R and $\widehat{f}$ in the role of f; applicability of Theorem 2.3 to the $\widehat{\ }$-data is guaranteed by (2.3.20) and (!). ∎

Note that for S, $\gamma > 0$ given, every affine classifier $f(x) = \langle w, x\rangle + b$, we can always find two positive scale factors λ in such a way that the classifier $\lambda f(x)$ and its margin slack vector taken with respect to S *and* the margin $\bar{\gamma} = \lambda\gamma$ satisfies the norm bound (2.3.26); bound (2.3.27) suggests that the generalization error of the corresponding augmented classifier will be the less the larger is $\bar{\gamma}$[3)]. Thus, *Theorem 2.4 suggests, given a training sample* S*, to impose on a candidate affine classifier* f *and "target margin"* γ *the normalization constraint (2.3.26)* (which is a restriction on f and γ, since ξ depends on both f and γ) *and to seek, under this constraint, for the pair* (f, γ) *with the largest possible* γ*,* that is, to use the classifier yielded by the optimization problem

$$\begin{array}{c}\max\limits_{\gamma,w,b}\left\{\gamma : \|w\|_*^2 + R^{-2}\|\xi(w,b,\gamma)\|_{p_*}^2 \leq 1\right\},\\ \xi_i(w,b,\gamma) = \max[0, \gamma - y_i(\langle w, x_i\rangle + b)]\end{array} \tag{$P_p(S)$}$$

Another recommendation inferred by Theorem 2.4 is in how to choose p. According to bound (2.3.27), the best possible choice of $p \in [2, \infty]$ is $p = 2$. Indeed, this choice makes $\widehat{\kappa}(p)$ as small as possible (in fact, equal to κ) and simultaneously increases the optimal value in the problem $(P_p(S))$, since when p grows from 2 to ∞, $p_* = \frac{p}{p-1}$ decreases from 2 to 1, so that the norm $\|\xi\|_{p_*}$ is nondecreasing in p. It therefore follows that $(P_2(S))$ has a wider feasible set than any one of the problems $(P_p(S))$, $2 \leq p \leq \infty$, and thus has the largest optimal value.

[3)] Strictly speaking, this conclusion is not fully justified at the moment – Theorem 2.4 assumes that the margin is fixed in advance, while we now make it depending on a candidate classifier. This flaw in the reasoning can be eliminated, at the cost of a number of additional technical considerations leading to extra logarithmic factors in the right hand side of (2.3.27). All these nuances are, however, irrelevant, as far as the practical algorithms and applications are concerned.

Chapter 3

The Mirror Prox Algorithm

In this Chapter we outline the basic theory of Proximal Mirror algorithm (MP) proposed in [3], which will be the "working horse" we intend to use in the context of Support Vector Machines. The presentation below follows [3].

3.1 MP: the construction

3.1.1 Saddle Point form of a convex optimization problem

The Mirror Prox algorithm, similar to other advanced first order algorithms for large-scale "well-structured" convex programs [6, 3, 9], is aimed at solving convex programs given in the *saddle point* form

$$\min_{x\in X}\max_{y\in Y} F(x,y), \tag{3.1.1}$$

where

- $X \subset \mathbf{R}^n$, $Y \subset \mathbf{R}^m$ are closed and bounded convex sets,
- $F(x,y) : Z \equiv X \times Y \to \mathbf{R}$ is a smooth (with continuous gradient) function on Z which is convex in $x \in X$ for every $y \in Y$ and is concave in $y \in Y$ for every $x \in X$.

The saddle point problem (3.1.1) gives rise to a pair of convex optimization problems:

- the *primal problem*

$$\min_{x\in X} f(x), \quad f(x) = \max_{y\in Y} F(x,y); \tag{P}$$

- the *dual problem*

$$\max_{y\in Y} g(y), \quad g(y) = \min_{x\in X} F(x,y). \tag{D}$$

The basic facts on these problems are as follows (recall that X, Y are convex compact sets, and F is continuously differentiable convex-concave function on $Z = X \times Y$; to keep the statements to follow true, continuous differentiability of F can be replaced with Lipschitz continuity):

1. The primal and the dual problems are convex, specifically, f is convex on X, and g is concave on Y Lipschitz continuous functions;

2. The optimal values in the problems are equal to each other:

$$\mathrm{Opt}(P) = \mathrm{Opt}(D). \tag{3.1.2}$$

In particular, if $x \in X$ and $y \in Y$, the *duality gap*

$$\mathrm{DualityGap}(x,y) \equiv f(x) - g(y) = [f(x) - \mathrm{Opt}(P)] + \underbrace{[\mathrm{Opt}(P) - \mathrm{Opt}(D)]}_{=0} + [\mathrm{Opt}(D) - g(y)]$$

is always nonnegative and is the sum of residuals, in terms of the objectives, of x and y as approximate solutions to the respective problems.

3. $x_* \in X$ and $y_* \in Y$ are optimal solutions to the respective problems if and only if $(*x_*, y_*)$ is a saddle point of F on $X \times Y$, that is, if and only if

$$\forall (x,y) \in X \times Y : F(x,y_*) \geq F(x_*,y_*) \geq F(x_*,y).$$

Note that in typical applications the problem of interest is just one of the problems (P), (D) (say, the primal one). Nevertheless, it turns out that solving the primal-dual pair (P), (D) of problems simultaneously is more efficient than solving solely the problem of interest (P), since the problems are closely related, and the approximate solutions of one of them carry important information on the other one, e.g., provide bounds on the respective optimal values:

$$\forall (y \in Y) : g(y) \leq \mathrm{Opt}(P); \quad \forall (x \in X) : f(x) \geq \mathrm{Opt}(D)$$

and thus allow to quantify the quality of a candidate solution.

3.1.2 MP: the setup

From now on, speaking about saddle point problem (3.1.1), we set $Z = X \times Y \subset E = \mathbf{R}^{n+m}$. We denote by $\langle z, w \rangle$ the standard inner product $z^T w$ on $\mathbf{R}^{\cdot}$.

Setup for MP as applied to (3.1.1) is given by

• A norm $\|\cdot\|$ on E used to quantify various characteristics of the entities to follow,

• A *distance-generating function* $\omega(z) : Z \to \mathbf{R}$ which should be continuously differentiable on Z and should be *strongly convex* on Z, meaning that

$$\langle \omega'(z') - \omega'(z''), z' - z'' \rangle \geq \kappa \|z' - z''\|^2 \ \forall z', z'' \in Z \tag{3.1.3}$$

for certain $\kappa > 0$ (called the *modulus of strong convexity* of $\omega(\cdot)$ w.r.t. $\|\cdot\|$).

> Note that in spite of the fact that $\omega(\cdot)$ is strongly convex on Z is independent of a particular choice of norm $\|\cdot\|$, the modulus of strong convexity does depend on this choice.

The norm $\|\cdot\|$ and the distance-generating function $\omega(\cdot)$ define several entities which will play important role in the sequel, specifically,

1. *Conjugate norm* $\|\cdot\|_*$ on E given by

$$\|\xi\|_* = \max_z \{\langle \xi, z \rangle : z \in E, \|z\| \leq 1\}; \tag{3.1.4}$$

2. *Local distance* ("prox-term", "Bregman distance") from a variable point $\zeta \in Z$ to a given point $z \in Z$; this distance is

$$\omega_z(\zeta) = \omega(\zeta) - \langle \zeta - z, \omega'(z) \rangle - \omega(z). \tag{3.1.5}$$

Note that this "distance" not necessarily should be a metric on Z (it can be non-symmetric w.r.t. z, ζ and need not satisfy the triangle inequality). The only property of metric inherited by local distance is positivity:

$$\omega_z(\zeta) \geq \frac{\kappa}{2}\|\zeta - z\|^2, \tag{3.1.6}$$

which is an immediate consequence of (3.1.3).

3. *Bregman diameter* $D_{z_0}[Z]$ of Z w.r.t. a point $z_0 \in Z$ and *Bregman diameter* of Z are given by

$$\begin{array}{rcl} D_{z_0}[Z] & = & \max\limits_{z\in Z} \omega_{z_0}(z), \\ D[Z] & = & \max\limits_{\zeta,z\in Z} \omega_z(\zeta) = \max\limits_{z\in Z} D_z[Z]. \end{array} \tag{3.1.7}$$

Note that $D_{z_0}[Z] \leq D[Z] < \infty$ due to compactness of Z and the fact that $\omega(\cdot)$ is continuously differentiable on Z.

4. *Prox-mapping* $P_z(\xi)$ which, for given $z \in Z$, maps a vector $\xi \in E$ onto the point

$$P_z(\xi) = \operatorname*{argmin}_{\zeta\in Z} \left[\langle \xi, \zeta\rangle + \omega_z(\zeta)\right] = \operatorname*{argmin}_{\zeta\in Z} \left[\langle \xi - \omega'(z), \zeta\rangle + \omega(\zeta)\right] \in Z. \tag{3.1.8}$$

Since Z is a compact set, the function $\phi_{\xi,z}(\zeta) = \langle \xi - \omega'(z), \zeta\rangle + \omega(\zeta)$ attains its minimum on Z, and since $\omega(\cdot)$ is strongly convex, the minimizer is unique. It follows that the prox-mapping is well defined. In fact, this mapping is Lipschitz continuous w.r.t. ξ, as stated in the following

Proposition 3.1 *Let $z \in Z$, and let $\xi, \eta \in E$. Then*

$$\|P_z(\xi) - P_z(\eta)\| \leq \frac{1}{\kappa}\|\xi - \eta\|_*, \tag{3.1.9}$$

Proof. Due to the origin of $P_z(\xi)$ and $P_z(\eta)$, we have

$$\begin{array}{ll} (a) & \langle \xi + \omega'(P_z(\xi)) - \omega'(z), P_z(\xi) - w\rangle \leq 0 \, \forall w \in Z, \\ (b) & \langle \eta + \omega'(P_z(\eta)) - \omega'(z), P_z(\eta) - w\rangle \leq 0 \, \forall w \in Z. \end{array}$$

Applying (a) to $w = P_z(\eta)$ and (b) to $P_z(\xi)$ and summing up the results, we get

$$\langle (\xi - \eta) + (\omega'(P_z(\xi)) - \omega'(P_z(\eta)), P_z(\xi) - P_z(\eta)\rangle \leq 0,$$

or, which is the same,

$$\langle \xi - \eta, P_z(\xi) - P_z(\eta)\rangle \leq -\langle \omega'(P_z(\xi)) - \omega'(P_z(\eta)), P_z(\xi) - P_z(\eta)\rangle.$$

By strong convexity of $\omega(\cdot)$, the right hand side in this inequality is $\leq -\kappa\|P_z(\xi) - P_z(\eta)\|^2$, while the left hand side is $\geq -\|\xi - \eta\|_*\|P_z(\xi) - P_z(\eta)\|$, and we arrive at the relation

$$-\|\xi - \eta\|_*\|P_z(\xi) - P_z(\eta)\| \leq -\kappa\|P_z(\xi) - P_z(\eta)\|^2,$$

and (3.1.9) follows. ∎

3.1.3 The conceptual MP algorithm

Given a saddle point problem (3.1.1), we associate with its *cost function* $F : Z \to \mathbf{R}$ the mapping $\Phi : Z \to E$ defined as

$$\Phi(x, y) = (F'_x(x, y), -F'_y(x, y)) \in E, \ z = (x, y) \in Z, \tag{3.1.10}$$

(the so called *monotone mapping* associated with convex-concave function F); note that F is continuously differentiable on Z, so that Φ is well-defined. Moreover, since F has Lipschitz continuous gradient on Z, mapping Φ is Lipschitz continuous, so that

$$\|\Phi(z') - \Phi(z'')\|_* \le L\|z' - z''\| \ \forall z', z'' \in Z. \tag{3.1.11}$$

Where we denote by L the Lipschitz constant of this mapping w.r.t. the pair of norms $\|\cdot\|$, $\|\cdot\|_*$. The "conceptual" algorithm for solving (3.1.1), is as follows:

Conceptual MP algorithm:

Initialization. Choose starting point $z_0 \in Z$ and a sequence of tolerances $\delta_t \ge 0$, $t = 1, 2, ...$

Step t, $t = 1, 2, ...$: Given z_{t-1}, check whether

$$P_{z_{t-1}}(\Phi(z_{t-1})) = z_{t-1}. \tag{3.1.12}$$

If it is the case, claim that z_{t-1} is the solution to the saddle point problem (3.1.1) and terminate. Otherwise choose $\gamma_t > 0$ and a point $w_t \in Z$ such that

$$\begin{array}{l}\langle w_t - P_{z_{t-1}}(\gamma_t\Phi(w_t)), \gamma_t\Phi(w_t)\rangle \\ + \left[\omega(z_{t-1}) + \langle\omega'(z_{t-1}), P_{z_{t-1}}(\gamma_t\Phi(w_t)) - z_{t-1}\rangle - \omega(P_{z_{t-1}}(\gamma_t\Phi(w_t)))\right] \le \delta_t.\end{array} \tag{3.1.13}$$

Set

$$\begin{array}{rcl} z_t & \equiv & (x_t, y_t) = P_{z_{t-1}}(\gamma_t\Phi(w_t)), \\ z^t & \equiv & (x^t, y^t) = \left(\sum\limits_{\tau=1}^{t} \gamma_\tau\right)^{-1} \sum\limits_{\tau=1}^{t} \gamma_\tau w_\tau \end{array}$$

and pass to step $t + 1$.

Note: z^t is the approximate solution built in course of t steps.

Convergence properties of the Conceptual MP (CMP) algorithm are summarized in the following proposition:

Proposition 3.2 *If CMP as applied to convex-concave saddle point problem* (3.1.1) *terminates at certain step t according to* (3.1.12)*, then z_{t-1} is a saddle point of F on $Z = X \times Y$, so that x_{t-1} is optimal for (P), and y_{t-1} is optimal for (D). Further, for every $t \ge 1$ such that CPM does not terminate at or before step t, the following efficiency estimate holds true:*

$$\mathrm{DualityGap}(x^t, y^t) \le \frac{\sum\limits_{\tau=1}^{t} \delta_\tau + D_{z_0}[Z]}{\sum\limits_{\tau=1}^{t} \gamma_\tau}. \tag{3.1.14}$$

Proof. Assume that CPM terminates at step t according to (3.1.12). Then the convex function $\phi(z) = \langle \Phi(z_{t-1}) - \omega'(z_{t-1}), z - z_{t-1}\rangle + \omega(z)$ attains its minimum over $z \in Z$ at $z = z_{t-1}$, whence

$$\langle \Phi(z_{t-1}), z - z_{t-1}\rangle \geq 0 \ \forall z \in Z,$$

or, setting $z_{t-1} = (x_{t-1}, y_{t-1})$ and recalling the definition of Φ,

$$\langle F'_x(x_{t-1}, y_{t-1}), x - x_{t-1}\rangle + \langle -F'_y(x_{t-1}, y_{t-1}), y - y_{t-1}\rangle \geq 0 \, \forall z = (x, y) \in X \times Y.$$

Recalling also that F is convex in x and concave in y (the left hand side of this inequality is $\leq [F(x, y_{t-1}) - F(x_{t-1}, y_{t-1})] + [F(x_{t-1}, y_{t-1}) - F(x_{t-1}, y)]$) we arrive at the following relation

$$F(x, y_{t-1}) - F(x_{t-1}, y) \geq 0 \ \forall (x, y) \in X \times Y.$$

Minimizing the left hand side in $x \in X$, $y \in Y$, we get

$$g(y_{t-1}) - f(x_{t-1}) \geq 0,$$

whence

$$\mathrm{DualityGap}(x_{t-1}, y_{t-1}) = f(x_{t-1}) - g(y_{t-1}) \leq 0.$$

Since the duality gap is always nonnegative and is zero iff the corresponding x, y are optimal solutions to the respective problems (P), (D), we conclude that x_{t-1} is an optimal solution to (P), y_{t-1} is an optimal solution to $()D)$, and thus (x_{t-1}, y_{t-1}) is a saddle point of F, as claimed.

Now assume that CMP did not terminate at or before step t. For $z, u \in Z$ let us set

$$H_z(u) = \langle \omega'(z), z - u\rangle - \omega(z). \tag{3.1.15}$$

Let $u \in Z$. For $0 \leq \tau \leq t$ we have

$$z_\tau = P_{z_{\tau-1}}(\gamma_\tau \Phi(w_\tau)) = \operatorname*{argmin}_{z \in Z} \left[\langle \gamma_\tau \Phi(w_\tau) - \omega'(z_{\tau-1}), z\rangle + \omega(z)\right],$$

which gives us the first inequality in the following chain:

$$\begin{array}{rcl}
0 & \geq & \langle \gamma_\tau \Phi(w_\tau) + \omega'(z_\tau) - \omega'(z_{\tau-1}), z_\tau - u\rangle \\
& = & \langle \gamma_\tau \Phi(w_\tau), w_\tau - u\rangle + \langle \gamma_\tau \Phi(w_\tau), z_\tau - w_\tau\rangle \\
& & + \underbrace{[\langle \omega'(z_\tau), z_\tau - u\rangle - \omega(z_\tau)]}_{H_{z_\tau}(u)} + \omega(z_\tau) \\
& & - \underbrace{[\langle \omega'(z_{\tau-1}), z_{\tau-1} - u\rangle - \omega(z_{\tau-1})]}_{H_{z_{\tau-1}}(u)} - \omega(z_{\tau-1}) - \langle \omega'(z_{\tau-1}), z_\tau - z_{\tau-1}\rangle \\
& = & H_{z_\tau}(u) - H_{z_{\tau-1}}(u) + \langle \gamma_\tau \Phi(w_\tau), w_\tau - u\rangle \\
& & + \underbrace{[\langle \gamma_\tau \Phi(w_\tau), z_\tau - w_\tau\rangle + [\omega(z_\tau) - \omega(z_{\tau-1}) - \langle \omega'(z_{\tau-1}), z_\tau - z_{\tau-1}\rangle]]}_{\geq -\delta_\tau \text{ by (3.1.13)}}
\end{array}$$

It follows that

$$1 \leq \tau \leq t \Rightarrow \langle \gamma_\tau \Phi(w_\tau), w_\tau - u\rangle \leq H_{z_{\tau-1}}(u) - H_{z_\tau}(u) + \delta_\tau;$$

summing up these inequalities over $\tau = 1, ..., t$, we get

$$\begin{aligned}
\sum_{\tau=1}^{t} \gamma_\tau \langle \Phi(w_\tau), w_\tau - u \rangle &\leq H_{z_0}(u) - H_{z_t}(u) + \sum_{\tau=1}^{t} \delta_\tau \\
&= \sum_{\tau=1}^{t} \delta_\tau + [\langle \omega'(z_0), z_0 - u \rangle - \omega(z_0) - \langle \omega'(z_t), z_t - u \rangle + \omega(z_t)] \\
&= \sum_{\tau=1}^{t} \delta_\tau + \Big[\langle \omega'(z_0), z_0 - u \rangle - \omega(z_0) + \underbrace{[\omega(z_t) + \langle \omega'(z_t), u - z_t \rangle]}_{\leq \omega(u)} \Big] \\
&\leq \sum_{\tau=1}^{t} \delta_\tau + \Big[\omega(u) - \langle \omega'(z_0), u - z_0 \rangle - \omega(z_0) \Big] \\
&\leq \sum_{\tau=1}^{t} \delta_\tau + D_{z_0}[Z].
\end{aligned}$$

We have arrived at the inequality

$$\sum_{\tau=1}^{t} \gamma_\tau \langle \Phi(w_\tau), w_\tau - u \rangle \leq \sum_{\tau=1}^{t} \delta_\tau + D_{z_0}[Z]; \tag{3.1.16}$$

setting $w_t = (\xi_t, \eta_t)$, $u = (x, y)$, the left hand side in this inequality is

$$\begin{aligned}
&\sum_{\tau=1}^{t} \gamma_\tau \Big[\langle F'_x(\xi_\tau, \eta_\tau), \xi_\tau - x \rangle + \langle F'_y(\xi_\tau, \eta_\tau), y - \eta_\tau \rangle \Big] \\
\geq\ &\sum_{\tau=1}^{t} \gamma_\tau \left[[F(\xi_\tau, \eta_\tau) - F(x, \eta_\tau)] + [F(\xi_\tau, y) - F(\xi_\tau, \eta_\tau)] \right] = \sum_{\tau=1}^{t} \gamma_\tau \left[F(\xi_\tau, y) - F(x, \eta_\tau) \right] \\
&\quad \left[\begin{array}{l} \text{since } \langle F'_x(\xi_\tau, \eta_\tau), \xi_\tau - x \rangle \geq F(\xi_\tau, \eta_\tau) - F(x, \eta_\tau) \text{ by convexity of } F \text{ in } x \text{ and} \\ \langle F'_y(\xi_\tau, \eta_\tau), y - \eta_\tau \rangle \geq F(\xi_\tau, y) - F(\xi_\tau, \eta_\tau) \text{ by concavity of } F \text{ in } y \end{array} \right] \\
\geq\ &\left(\sum_{\tau=1}^{t} \gamma_\tau \right) [F(x^t, y) - F(x, y^t)] \\
&\quad \text{[due to } z^t = (x^t, y^t) = \left(\sum_{\tau=1}^{t} \gamma_\tau \right)^{-1} \sum_{\tau=1}^{t} \gamma_\tau (\xi_\tau, \eta_\tau) \text{ and convexity-concavity of } F]
\end{aligned}$$

Thus, (3.1.16) implies that

$$F(x^t, y) - F(x, y^t) \leq \frac{\sum_{\tau=1}^{t} \delta_\tau + D_{z_0}[Z]}{\sum_{\tau=1}^{t} \gamma_\tau}.$$

The resulting inequality is valid for all $u = (x, y) \in X \times Y$, and the right hand side is independent of u. The maximum of the left hand side in $u = (x, y) \in Z$ is $f(x^t) - g(y^t) = \text{DualityGap}(x^t, y^t)$, and (3.1.14) follows. ∎

3.1.4 From conceptual to implementable algorithm

The error bound (3.1.14) suggests that what we would be interested in is to ensure that the "residuals" δ_τ are as small as possible (ideally, $\delta_\tau = 0$) and stepsizes γ_τ are as large as possible. The question, of course, is how to ensure the crucial condition (3.1.13), that is,

$$\begin{aligned}
&\langle w_t - P_{z_{t-1}}(\gamma_t \Phi(w_t)), \gamma_t \Phi(w_t) \rangle \\
&+ \left[\omega(z_{t-1}) + \langle \omega'(z_{t-1}), P_{z_{t-1}}(\gamma_t \Phi(w_t)) - z_{t-1} \rangle - \omega(P_{z_{t-1}}(\gamma_t \Phi(w_t))) \right] \leq \delta_t.
\end{aligned}$$

with "small" δ_t and "large" γ_t. Note that the quantity in the parentheses in the left hand side of our target inequality is ≤ 0 due to the convexity of $\omega(\cdot)$ and in fact is even $\le -\frac{\kappa}{2}\|z_{t-1} - P_{z_{t-1}}(\gamma_t\Phi(w_t)\|^2$ due to strong convexity of $\omega(\cdot)$, so that all we need is to make small enough the first term $\langle w_t - P_{z_{t-1}}(\gamma_t\Phi(w_t)), \gamma_t\Phi(w_t)\rangle$ in the left hand side of the inequality. A conceptually simplest way to do this would be to ensure that

$$P_{z_{t-1}}(\gamma_t\Phi(w_t)) = w_t,$$

that is, w_t should be a fixed point of the mapping

$$w \mapsto \Psi_{z_{t-1}}(w) \equiv P_{z_{t-1}}(\gamma_t\Phi(w)) : Z \to Z.$$

The outlined approach (corresponding to the so called "idealized" proximal point algorithm) seems (and in general is) impractical – to find a fixed point of a nontrivial mapping is not easier than to solve the problem of interest. The crucial observation made in [3] is that *finding the desired fixed point is relatively easy, provided that γ_t is not too large*. Specifically, the mapping $w \mapsto \Psi_{z_{t-1}}(w)$ is the superposition of two mappings:

• the mapping $w \mapsto \gamma_t\Phi(w) : Z \to E$; by (3.1.11), this mapping is Lipschitz continuous with constant $\gamma_t L$, from the metric on Z given by $\|\cdot\|$ to the metric on E given by $\|\cdot\|_*$;

• the mapping $\xi \mapsto P_{z_{t-1}}(\xi) : E \to Z$ which is Lipschitz continuous, with constant κ^{-1}, from the metric on E given by $\|\cdot\|_*$ to the metric on Z given by $\|\cdot\|$ (Proposition 3.1).

It follows that *the mapping $w \mapsto \Psi_{z_{t-1}}(w) : Z \to Z$ is Lipschitz continuous, with constant $\widehat{\kappa} = \kappa^{-1}\gamma_t L$ w.r.t. the metric on Z given by $\|\cdot\|$*. In particular, *when $\widehat{\kappa} \le 1/\sqrt{2}$* (that is, $\gamma_t \le \frac{\kappa}{\sqrt{2}L}$), *the mapping is a contraction, with coefficient $\le 1/\sqrt{2}$*, so that its fixed point w_* (which does exist – the mapping is a contraction!) can be rapidly approximated by iterating the mapping itself:

$$w^0 \in Z,\ w^s = \Psi_{z_{t-1}}(w^{s-1}) \equiv P_{z_{t-1}}(\gamma_t\Phi(w^{s-1})) \Rightarrow \|w^s - w_*\| \le \widehat{\kappa}^s\|w^0 - w_*\| \le 2^{-s/2}\|w^0 - w_*\|. \tag{3.1.17}$$

Thus, we arrive at the following *basic implementation* of step t in the Conceptual MP algorithm:

Algorithm 3.1 [Mirror-Prox Algorithm, Basic Implementation]
Step t, $t = 1, 2, ...$: Given $z_{t-1} \in Z$, set $\gamma_t = \frac{\kappa}{\sqrt{2}L}$, check whether $z_{t-1} = P_{z_{t-1}}(\gamma_t\Phi(z_{t-1}))$. If it is the case, terminate and claim that z_{t-1} is a saddle point of F on $X \times Y$, otherwise iterate, starting with

$$w^0 = z_{t-1}$$

using the updating

$$w^{s-1} \mapsto w^s = P_{z_{t-1}}(\gamma_t\Phi(w^{s-1})) \tag{3.1.18}$$

until the termination condition

$$\begin{array}{l}\langle w^{s-1} - P_{z_{t-1}}(\gamma_t\Phi(w^{s-1})), \gamma_t\Phi(w^{s-1})\rangle \\ + [\omega(z_{t-1}) + \langle\omega'(z_{t-1}), w^s - z_{t-1}\rangle - \omega(w^s)] \le 0\end{array} \tag{3.1.19}$$

is met. When it happens for the first time, the value of s being s_t, set

$$w_t = w^{s_t-1},\ z_t = w^{s_t},$$

thus ensuring (3.1.13) *with $\delta_t = 0$, and pass to step $t+1$.*

The approximate solution generated during t steps is the point

$$z^t = \left(\sum_{\tau=1}^t \gamma_\tau\right)^{-1} \sum_{\tau=1}^t \gamma_\tau w_\tau. \tag{3.1.20}$$

Note that when w_* differs from z_{t-1} (which is the case exactly when $z_{t-1} \neq P_{z_{t-1}}(\gamma_t \Phi(z_{t-1}))$), the iterates w^s converge to w_*, so that the quantity

$$[\omega(w^s) + \langle \omega'(z_{t-1}), w^s - z_{t-1}\rangle - \omega(w^s)]$$

has a negative limit, while the quantity

$$\langle \underbrace{w^{s-1} - P_{z_{t-1}}(\gamma_t \Phi(w^{s-1}))}_{w^{s-1}-w^s}, \gamma_t \Phi(w^{s-1})\rangle$$

converges to 0 as a geometric progression with the ratio $2^{-1/2}$ (since $\|w^{s-1}-w^s\| \leq \|w^{s-1}-w_*\| + \|w^s - w_*\| \leq 2^{1-(s-1)/2}\|w^0 - w_*\|$ by (3.1.17), while $\|\gamma_t \Phi(w^{s-1})\|_*$ remains bounded by (3.1.11). It follows that the termination condition (3.1.19) eventually will be met, and the associated number s_t of the "inner steps" is, for all practical purposes, a moderate constant. In fact, the situation is even better: the analysis carried out in [3] demonstrates that *the termination condition is met in at most 2 inner steps*, that is, $s_t \leq 2$. We have arrived at the following result:

Theorem 3.1 [3]. *Let the convex-concave saddle point problem* (3.1.1) *satisfying* (3.1.11) *be solved by Basic MP algorithm (that is, Conceptual MP algorithm with Basic implementation of the steps). Then the approximate solution $z^t = (x^t, y^t)$ obtained during $t = 1, 2, ...$ steps is either an exact solution to the saddle point problem, or an approximate solution with duality gap which can be bounded as*

$$\mathrm{DualityGap}(x^t, y^t) \leq \frac{\sqrt{2} L D_{z_0}[Z]}{\kappa t}, \tag{3.1.21}$$

and every step requires at most 2 computations of $\Phi(\cdot)$ and at most 2 computations of the values of the prox-mapping $P_z(\xi)$.

Proof. The error bound (3.1.21) is readily given by (3.1.14) (we are in the situation of $\delta_\tau \equiv 0$ and $\gamma_\tau \equiv \frac{\kappa}{\sqrt{2}L}$). The complexity of a step is readily given by the preceding remarks. ∎

Remark 3.1 *From* (3.1.21) *it follows that the influence of the choice of the distance-generating function $\omega(\cdot)$ and the starting point on the theoretical performance of MP can be "summarized" by the complexity parameter*

$$\Omega = \frac{D_{z_0}[Z]}{\kappa}.$$

3.2 Optimizing the setup

3.2.1 Building blocks

Error bound (3.1.21) explains, in particular, how the performance of the MP algorithm depends on the choice of the "parameters" $\|\cdot\|$, $\omega(\cdot)$ underlying the construction and allows to optimize, to some extent, the algorithm w.r.t. these parameters. We present here the corresponding recommendations (cf. [3]), restricting ourselves with the situations which are relevant to our intended SVM applications. Specifically, assume that

1. Both X and Y in (3.1.1) are closed convex subsets of *standard sets* $\widehat{X}$, $\widehat{Y}$, where a *standard set* $\widehat{W}$ is a set representable as

   $$\widehat{W} = V_1 \times ... \times V_k,$$

   with the direct factors ("standard blocks") $V_\ell \in \mathbf{R}^{n_\ell}$, $\ell = 1, ..., k$ being of the following types:

(a) *p-ball* $B_p(R) = \{v \in \mathbf{R}^d : \|v\|_p \le R\}$, or the nonnegative part $B_p^+(R) = \{v \in B_p(R) : v \ge 0\}$, of such a ball, where $1 \le p \le \infty$.
We shall refer to $B_2(R)$ as to an *Euclidean ball*, to $B_1(R)$ as to a *hyperoctahedron*, to $B_1^+(R)$ as to a *full-dimensional simplex*, and to $B_\infty(R)$ as to a *box*;

(b) *Standard simplex* $\{v \in \mathbf{R}^d : v \ge 0, \sum_\ell v_\ell = R\}$;

(c) *Extended simplex* $\{v = (v^1, ..., v^L) : v^s \in \mathbf{R}^{n_s}, \sum_{s=1}^{L} \|v^s\|_2 \le R\}$
(this block was *not* considered in [3]; it is relevant to SVM problems with "adjustable" kernels, see Section 4.1.3).

We denote by X_i, $i = 1, ..., k_X$, the standard blocks corresponding to X, so that

$$X \subset \underbrace{X_1}_{\subset \mathbf{R}^{n_1}} \times ... \times \underbrace{X_{k_X}}_{\subset \mathbf{R}^{n_{k_X}}} \subset \mathbf{R}^n = \mathbf{R}^{n_1+...+n_{k_X}},$$

and by Y_j, $j = 1, ..., k_Y$, the standard blocks corresponding to Y, so that

$$Y \subset \underbrace{Y_1}_{\subset \mathbf{R}^{m_1}} \times ... \times \underbrace{Y_{k_Y}}_{\subset \mathbf{R}^{m_{k_Y}}} \subset \mathbf{R}^m = \mathbf{R}^{m_1+...+m_{k_Y}}.$$

Consequently,

$$\begin{array}{c} Z \equiv X \times Y \subset Z_1 \times ... \times Z_k \in E, \\ E = \mathbf{R}^{n_1} \times ... \times \mathbf{R}^{n_{k_X}} \times \mathbf{R}^{m_1} \times ... \times \mathbf{R}^{m_{k_Y}} \equiv \mathbf{R}^{d_1} \times ... \times \mathbf{R}^{d_k}, \\ k = k_X + k_Y,\ Z_p = \left\{ \begin{array}{ll} X_p, & p \le k_X \\ Y_{p-k_X}, & k_X < p \le k \end{array} \right. ,\ d_p = \left\{ \begin{array}{ll} n_p, & p \le k_X \\ m_{p-k_X}, & k_x < p \le k \end{array} \right. \end{array} \tag{3.2.1}$$

Thus, $z = (x, y) \in E$ is representable as $z = (z[1], ..., z[k])$, where $z[p] \in \mathbf{R}^{d_p}$.

2. The cost function $F(x, y)$ is bilinear:

$$F(x, y) = a^T x + b^T y + \sum_{i=1}^{k_X} \sum_{j=1}^{k_Y} z^T[k_X + j] A^{ij} z[i],\ z = (x, y) \in Z, \tag{3.2.2}$$

where A^{ij} are $m_i \times n_j$ matrices. We now define $d_p \times d_q$ matrices B^{pq}, $p, q = 1, ..., k$, as follows:

$$B^{pq} = \left\{ \begin{array}{ll} 0_{d_p \times d_q}, & p, q \le k_x \text{ or } p, q > k_x \\ [A^{p,q-k_x}]^T, & p \le k_x, q > k_x \\ -A^{q,p-k_x}, & p > k_x, q \le k_x \end{array} \right. \tag{3.2.3}$$

and define $(d_1 + ... + d_k) \times (d_1 + ... + d_k)$-matrix B as a block matrix with blocks B^{pq}, $1 \le p, q \le k$; note that this matrix is skew-symmetric: $B^{pq} = -[B^{qp}]^T$. With this notation, the mapping $\Phi(z)$, $z = (x, y)$, associated, via (3.1.10), with the bilinear cost function (3.2.2) becomes the affine mapping

$$\Phi(z) = \phi + Bz,\ \phi = \left[\begin{array}{c} a \\ -b \end{array} \right]. \tag{3.2.4}$$

We associate with every one of the blocks Z_p, $1 \le p \le k$ (see (3.2.1)) "local data" – respective norm $\|\cdot\|$, distance-generating function $\omega(\cdot)$, starting point and parameters κ, D – according to the following rules. Let V be the block in question. Then

1. When $V = B_p(R)$ is a p-ball in $\mathbf{R}^d$, $1 \le p \le 2$, or $V = B_p^+(R)$ is the nonnegative part of such a ball, we set

$$\begin{gathered}\widehat{p} = \max\left[p, 1 + \tfrac{1}{2\log(d)}\right],\\ \|v\| = \|v\|_p,\ \omega(v) = \tfrac{1}{\widehat{p}}\sum_{i=1}^d |v_i|^{\widehat{p}},\ v^{(0)} = 0,\ \kappa = (\widehat{p}-1)d^{\frac{2}{\widehat{p}}-\frac{2}{p}}R^{\widehat{p}-2},\ D = \tfrac{1}{\widehat{p}}R^{\widehat{p}},\\ \Omega = O(1)\tfrac{R^2 d^{\frac{2}{\widehat{p}}-\frac{2}{p}}}{\widehat{p}-1} \le O(1)R^2\log(d);\end{gathered} \tag{3.2.5}$$

   Here and in what follows $v^{(0)}$ is the starting point associated with the block, κ is the modulus of strong convexity of $\omega(\cdot)$ w.r.t. $\|\cdot\|$ on V, D is an upper bound on the quantity $D_{v^{(0)}}[V]$ and Ω is an upper bound on the complexity parameter $\frac{D}{\kappa}$;

2. When $V = B_p(R)$ is a p-ball in $\mathbf{R}^d$, $2 \le p \le \infty$, or $V = B_p^+(R)$ is the nonnegative part of such a ball, we set

$$\|v\| = \sqrt{v^T v},\ \omega(v) = \frac{1}{2}v^T v,\ v^{(0)} = 0,\ \kappa = 1,\ D = \frac{1}{2}d^{1-\frac{2}{p}}R^2,\ \Omega = \frac{R^2 d^{1-\frac{2}{p}}}{2}; \tag{3.2.6}$$

3. When $V = \{v \in \mathbf{R}^d : v \ge 0, \sum\limits_\ell v_\ell \le R\}$ is a full-dimensional simplex or $V = \{v \in \mathbf{R}^d : v \ge 0, \sum\limits_\ell v_\ell = R\}$ is a standard simplex, we set

$$\begin{gathered}\|v\| = \|v\|_1,\ \omega(v) = \sum_\ell (R^{-1}v_\ell + d^{-1}\delta)\log(R^{-1}v_\ell + d^{-1}\delta),\\ v^{(0)} = d^{-1}(R, ..., R)^T, \kappa = R^{-2}(1+\delta),\ D = (1+\delta)\log(d),\ \Omega = O(1)R^2\log(d),\end{gathered} \tag{3.2.7}$$

   where $\delta \in (0,1)$ is a once for ever fixed "regularization parameter" which we set to $1.e$-3;

4. When $V = \{v \in \mathbf{R}^d : \|v\|_1 \le R\}$ is a hyperoctahedron, we set

$$\begin{array}{rcl}\|v\| = \|v\|_1,\ v^{(0)} = 0, \kappa = R^{-2}(1+\delta),\ D = (1+(2d)^{-1}\delta)\log(2d),\ \Omega = O(1)R^2\log(2d)\\ \omega(v) & = & \min\limits_{u,w}\left\{\sum\limits_i [u_i\log(u_i) + w_i\log(w_i)] : \begin{array}{l} u, w \ge 0\\ u - w = R^{-1}v\\ \sum\limits_i [u_i + w_i] = 1 + \delta\end{array}\right\}\\ & = & \sum\limits_i \frac{\sqrt{R^{-2}v_i^2 + \theta(v)} - R^{-1}v_i}{2}\log\left(\frac{\sqrt{R^{-2}v_i^2 + \theta(v)} - R^{-1}v_i}{2}\right)\\ & & + \sum\limits_i \frac{\sqrt{R^{-2}v_i^2 + \theta(v)} + R^{-1}v_i}{2}\log\left(\frac{\sqrt{R^{-2}v_i^2 + \theta(v)} + R^{-1}v_i}{2}\right),\\ \theta(v) & : & \sum\limits_i \sqrt{R^{-2}v_i^2 + \theta(v)} = 1 + (2d)^{-1}\delta\end{array} \tag{3.2.8}$$

   Here, as in the previous item, $\delta = 1.$e-3 is a regularization parameter.

Remark 3.2 *Note that hyperoctahedron is equipped with two "local data" – those given by* (3.2.5) *in the case of* $p = 1$ *and those given by* (3.2.8)*. While both sets of data share the*

common norm and the same complexity parameter, they differ in the choice of the distance generating function. In the sequel, working with hyperoctahedrons, we will use the latter data (which seem to result in the slightly better practical performance of MP) rather than the former ones.

5. Finally, when $V = \{v = (v^1, ..., v^L) : v^s \in \mathbf{R}^{n_s}, \sum_{s=1}^{L} \|v^s\|_2 \le R\}$ is an extended simplex, we set

$$\begin{array}{rclcrcl} \theta & = & \left\{ \begin{array}{ll} 2, & L \le 2 \\ 1 + \frac{1}{\log(L)}, & L \ge 3 \end{array} \right. & & \|v\| & = & \sum_{s=1}^{L} \|v^s\|_2 \\ v^{(0)} & = & 0 & & \kappa & = & (\theta - 1) L^{1-\theta} R^{-2} \ge \frac{R^{-2}\exp\{-1\}}{\max[1, \log(L)]} \\ D & = & \theta^{-1} & & \Omega & = & O(1) R^2 \max[1, \log(L)] \\ \omega(v) & = & \theta^{-1} R^{-\theta} \sum_{s=1}^{L} \|v^s\|_2^\theta & & & & \end{array} \tag{3.2.9}$$

We have the following

Proposition 3.3 *In all aforementioned cases the function $\omega(\cdot)$ is strongly convex, with the indicated modulus κ w.r.t. the indicated norm $\|\cdot\|$, on V, $D \ge D_{v^{(0)}}[V]$ and $\Omega \ge \frac{D}{\kappa}$.*

Proof. 1^0. We start with the following simple fact:

Lemma 3.1 *Let $\|\cdot\|$ be a norm on $\mathbf{R}^d$, let $V \subset \mathbf{R}^D$ be a closed convex set with a nonempty interior, and let $\omega(\cdot)$ be a continuously differentiable function on V which is twice continuously differentiable everywhere on V outside the union U of finitely many proper linear subspaces of $\mathbf{R}^d$. Then the necessary and sufficient condition for $\omega(\cdot)$ to be strongly convex on V, with modulus $\kappa > 0$ w.r.t. $\|\cdot\|$, is that*

$$h^T \omega''(v) h \ge \kappa \|h\|^2 \ \ \forall (v \in (\mathrm{int} V) \backslash U, h \in \mathbf{R}^d). \tag{3.2.10}$$

Proof. *Necessity:* Assume that $\omega(\cdot)$ is strongly convex on V, with modulus κ w.r.t. $\|\cdot\|$, and let us prove that (3.2.10) takes place. Let $v \in V' \equiv V \backslash U$ and $h \in E$. The set V' is open, so that for all small enough positive t we have $[v, v + th] \in V'$. We have $\langle \omega'(v + th) - \omega'(v), th \rangle \ge \kappa \|th\|^2$, whence

$$\langle \frac{\omega'(v + th) - \omega'(v)}{t}, h \rangle \ge \kappa \|h\|^2.$$

When $t \to +0$, the left hand side in this relation converges to $h^T \omega''(v) h$, and (3.2.10) follows.

Sufficiency: Assume that (3.2.10) takes place, and let us verify that $\omega(\cdot)$ is strongly convex, with modulus κ w.r.t. $\|\cdot\|$, on V, that is,

$$\langle \omega'(v'') - \omega'(v'), v'' - v' \rangle \ge \kappa \|v' - v''\|^2 \ge 0$$

whenever $v', v'' \in V$. Since $\omega'(\cdot)$ is continuous on V, the left hand side in the target inequality is continuous in $v', v'' \in V$; therefore it suffices to verify this inequality for v', v'' belonging to a dense in V subset, e.g., for $v', v'' \in V'$. Assuming $v', v'' \in V'$, and taking into account the structure of V', all but finitely many points of the segment $[v', v'']$ belong to V'. In other words, setting $h = v'' - v'$ and $v_t = v' + th$, $0 \le t \le 1$, there exist finitely many values $[0 <] t_1 < t_2 < ... < t_k [< 1]$ of the parameter t such that the point v_t belongs to V' whenever

$t \in [0,1]$ is distinct from $t_1, ..., t_k$. The points $t_1, ..., t_k$ split the segment $[0,1]$ into $k+1$ segments $\Delta_0, ..., \Delta_k$. Let $\tau' < \tau''$ be two points of one of these segments, and let us verify that

$$\langle \omega'(v_{\tau''}) - \omega'(v_{\tau'}), h \rangle \geq \kappa(\tau'' - \tau')\|h\|^2. \tag{3.2.11}$$

Indeed, the function $\phi(\tau) = \omega(v_\tau)$ is twice continuously differentiable on $[\tau', \tau'']$, and

$$\phi''(\tau) = h^T \omega''(v_\tau) h \geq \kappa \|h\|^2$$

by (3.2.10) due to $v_t \in B'$, whence

$$\langle \omega'(v_{\tau''}) - \omega'(v_{\tau'}), h \rangle \equiv \phi'(\tau'') - \phi'(\tau') \geq (\tau'' - \tau')\kappa\|h\|^2,$$

as claimed. Now, since ω' is continuous on V, relation (3.2.11) implies, by continuity in τ', τ'', that

$$\langle \omega'(v_{t_{\ell+1}}) - \omega'(v_{t_\ell}), h \rangle \geq \kappa(t_{\ell+1} - t_\ell)\|h\|^2,$$

where t_ℓ, $t_{\ell+1}$ are the left and the right endpoints of Δ_ℓ (so that $t_0 = 0$ and $t_{k+1} = 1$). Summing up the resulting inequalities over $\ell = 0, 1, ..., k$, we get

$$\langle \omega'(v'') - \omega'(v'), v'' - v' \rangle \equiv \langle \omega'(v'') - \omega'(v'), h \rangle \geq \kappa(t_{k+1} - t_0)\|h\|^2 \equiv \kappa\|v'' - v'\|^2,$$

as claimed. □

2^0. Now let us prove the results stated in Proposition for the case when V is the p-ball $B_p(R)$ in $\mathbf{R}^d$ or a nonnegative part of such a ball and $1 \leq p \leq 2$. Let us verify first that $\omega(v) = \frac{1}{\widehat{p}} \sum_{i=1}^d |v_i|^{\widehat{p}}$ is κ-strongly convex w.r.t. $\|\cdot\|_p$ on V. The function is clearly continuously differentiable everywhere on V and is twice continuously differentiable on the set $V' = \{v \in \mathrm{int}V : v_i \neq 0,\ i = 1, ..., d\}$. Now let $v \in V'$ and $h \in \mathbf{R}^d$. We have

$$h^T \omega'(v) = \sum_{i=1}^d |v_i|^{\widehat{p}-1} \mathrm{sign}(v_i) h_i \ \Rightarrow\ h^T \omega''(v) h = (\widehat{p} - 1) \sum_{i=1}^d |v_i|^{\widehat{p}-2} h_i^2, \tag{3.2.12}$$

whence,

$$\widehat{p} = 2 \Rightarrow h^T \omega''(v) h = \|h\|_2^2 \geq d^{\frac{2}{\widehat{p}} - \frac{2}{p}} \|h\|_p^2, \tag{3.2.13}$$

where the concluding inequality is readily given by the standard inequality

$$u \in \mathbf{R}^d, 1 \leq s \leq r \leq \infty \Rightarrow \|u\|_r \leq \|u\|_s \leq d^{\frac{1}{s} - \frac{1}{r}} \|u\|_r. \tag{3.2.14}$$

Now let $\widehat{p} < 2$. Then

$$\begin{aligned}
\|h\|_{\widehat{p}}^2 &= \left(\sum_i |h_i|^{\widehat{p}} \right)^{\frac{2}{\widehat{p}}} = \left(\sum_i \left[|h|_i^2 |v_i|^{\widehat{p}-2} \right]^{\frac{\widehat{p}}{2}} \left[|v_i|^{\widehat{p}} \right]^{\frac{2-\widehat{p}}{2}} \right)^{\frac{2}{\widehat{p}}} \\
&\leq \left(\left(\sum_i |h_i|^2 |v_i|^{\widehat{p}-2} \right)^{\frac{\widehat{p}}{2}} \left(\sum_i |v_i|^{\widehat{p}} \right)^{\frac{2-\widehat{p}}{2}} \right)^{\frac{2}{\widehat{p}}} \\
&\qquad \text{[we have used Hölder Inequality]} \cdot \\
&= \left(\sum_i |h_i|^2 |v_i|^{\widehat{p}-2} \right) \left(\sum_i |v_i|^{\widehat{p}} \right)^{\frac{2-\widehat{p}}{\widehat{p}}} \\
&\leq \frac{\|v\|_{\widehat{p}}^{2-\widehat{p}}}{\widehat{p}-1} h^T \omega''(v) h \\
&\qquad \text{[we have used (3.2.12)]}
\end{aligned}$$

We have arrived at the relation

$$\widehat{p} < 2 \;\Rightarrow\; h^T\omega''(v)h \geq \frac{\widehat{p}-1}{\|v\|_{\widehat{p}}^{2-\widehat{p}}}\|h\|_{\widehat{p}}^2 \geq \frac{(\widehat{p}-1)d^{\frac{2}{\widehat{p}}-\frac{2}{p}}}{\|v\|_p^{2-\widehat{p}}}\|h\|_p^2, \tag{3.2.15}$$

where the concluding inequality is given by (3.2.14) combined with $p \leq \widehat{p}$. Observing that

$$\frac{2}{\widehat{p}} - \frac{2}{p} = 2\frac{p-\widehat{p}}{p\widehat{p}} \begin{cases} = 0, & \widehat{p} = p \\ \geq -\frac{1}{\log(d)}, & \widehat{p} > p \end{cases},$$

we obtain from (3.2.13), (3.2.15), by taking into account that $\|v\|_p \leq R$ for $v \in V$ and the definition of κ in (3.2.5), that

$$h^T\omega''(v)h \geq \kappa\|h\|_p^2$$

for all $v \in V'$ and all $h \in \mathbf{R}^d$; invoking Lemma 3.1, we see that $\omega(\cdot)$ is κ-strongly convex w.r.t. $\|\cdot\|$ on V. Further, we have

$$D_{v^{(0)}}[V] = \max_{v\in V}\left[\frac{1}{\widehat{p}}\sum_i |v_i|^{\widehat{p}} - 0 - \langle 0, v\rangle\right] = \frac{1}{\widehat{p}}\max_{v\in V}\|v\|_{\widehat{p}}^{\widehat{p}} \leq R^{\widehat{p}} = D,$$

and finally $\Omega = \frac{D_{v^{(0)}}[V]}{\kappa} = \frac{R^2 d^{\frac{2}{p}-\frac{2}{\widehat{p}}}}{\widehat{p}-1} \leq O(1)R^2\log(d)$, as claimed.

3^0. Now consider the case when V is p-ball $B_p(R)$ in $\mathbf{R}^d$ or the nonnegative part of such a ball and $2 \leq p \leq \infty$. The fact that the function $\omega(v) = \frac{1}{2}v^Tv$ is 1-strongly convex w.r.t. the norm $\|v\| \equiv \|v\|_2 = \sqrt{v^Tv}$ is evident, so that all we need is to show that $D_0[V]$ is bounded from above by the quantity D indicated in (3.2.6). This is immediate:

$$\begin{array}{rcl} D_0[V] & = & \max\limits_{v\in V}\left[\frac{1}{2}v^Tv - \frac{1}{2}0^T0 - v^T\cdot 0\right] = \max\limits_{v\in V}\frac{1}{2}v^Tv \\ & \leq & \max\limits_{v:\|v\|_p\leq R}\frac{1}{2}\|v\|_2^2 \\ & \leq & \frac{R^2d^{1-\frac{2}{p}}}{2}, \end{array}$$

where the concluding $\leq$ (in fact, it is equality) is given by (3.2.14).

4^0. For simplexes (both full-dimensional and standard) the results stated in Proposition 3.3 are proved in [3] (where also the case of p-ball with $p = 2$ is considered). For hyperoctahedron, the required results can be easily derived from those related to the standard simplex; we skip the derivation. The only case which remains to be considered is the one of extended simplex, and this is the case we shall investigate now.

In view of Lemma 3.1, in order to prove that $\omega(\cdot)$ is strongly convex on V, with modulus κ w.r.t. the norm $\|\cdot\|$, we should verify that

$$h^T\omega''(v)h \geq \kappa\|h\|^2 \;\forall(v \in V', h \in E = \mathbf{R}^{n_1} \times ... \times \mathbf{R}^{n_L}), \tag{3.2.16}$$

where $V' = \{v \in \mathrm{int}V : v^s \neq 0,\ s = 1, ..., L\}$. Let us demonstrate that this indeed is the case. Let $v = (v^1, ..., v^L) \in V'$ and $h = (h^1, ..., h^L) \in E$, and let us set

$$\begin{array}{rcl} d\nu^s & = & \frac{d}{dt}\Big|_{t=0}\|v^s + th^s\|_2 = r_s^Th^s,\ r_s = \|v^s\|_2^{-1}v^s; \\ d^2\nu^s & = & \frac{d^2}{dt^2}\Big|_{t=0}\|v^s + th^s\|_2 = \frac{[h^s]^T(I - r_sr_s^T)h^s}{\|v^s\|_2}. \end{array} \tag{3.2.17}$$

We now have

$$\begin{aligned}
R^\theta h^T\omega'(v+th) &= \tfrac{d}{dt}\theta^{-1}\sum_{s=1}^L\|v^s+th^s\|_2^\theta=\sum_{s=1}^L\|v^s+th^s\|_2^{\theta-1}\tfrac{d}{dt}\|v^s+th^s\|_2\\
\Rightarrow R^\theta h^T\omega''(v)h &= \tfrac{d}{dt}\Big|_{t=0}R^\theta h^T\omega'(v+th)=\tfrac{d}{dt}\Big|_{t=0}\left[\sum_{s=1}^L\|v^s+th^s\|_2^{\theta-1}\tfrac{d}{dt}\|v^s+th^s\|_2\right]\\
&= (\theta-1)\sum_{s=1}^L\|v^s\|_2^{\theta-2}\left(\tfrac{d}{dt}\Big|_{t=0}\|v^s+th^s\|_2\right)^2\\
&\quad+\sum_{s=1}^L\|v^s\|_2^{\theta-1}\tfrac{d^2}{dt^2}\Big|_{t=0}\|v^s+th^s\|_2\\
&= \sum_{s=1}^L\left[(\theta-1)\|v^s\|_2^{\theta-2}(d\nu^s)^2+\|v^s\|_2^{\theta-1}d^2\nu^s\right]\\
&= \sum_{s=1}^L\left[(\theta-1)\|v^s\|_2^{\theta-2}(r_s^Th_s)^2+\|v^s\|_2^{\theta-1}\|v^s\|_2^{-1}([h^s]^T(I-r_sr_s^T)h^s)\right]\\
&\geq (\theta-1)\sum_{s=1}^L\|v^s\|_2^{\theta-2}\|h^s\|_2^2.
\end{aligned}$$

We, therefore, have arrived at the inequality

$$h^T\omega''(v)h\geq R^{-\theta}(\theta-1)\sum_{s=1}^L\|v^s\|_2^{\theta-2}\|h^s\|_2^2. \tag{3.2.18}$$

We now have

$$\begin{aligned}
\|h\|^2 &= \left(\sum_{s=1}^L\|h^s\|_2\right)^2=\left(\sum_{s=1}^L[\|h^s\|_2\|v^s\|_2^{\frac{\theta-2}{2}}]\|v^s\|_2^{\frac{2-\theta}{2}}\right)^2\\
&\leq \left(\sum_{s=1}^L\|h^s\|_2^2\|v^s\|_2^{\theta-2}\right)\left(\sum_{s=1}^L\|v^s\|_2^{2-\theta}\right)\\
&\qquad\qquad\text{[Cauchy Inequality]}\\
&\leq \left[h^T\omega''(v)h\right]^{\frac{R^\theta\sum_{s=1}^L\|v^s\|_2^{2-\theta}}{\theta-1}}\\
&\qquad\qquad\text{[see (3.2.18)]}
\end{aligned}$$

whence

$$h^T\omega''(x)h\geq\frac{(\theta-1)R^{-\theta}}{\sum_{s=1}^L\|v^s\|_2^{2-\theta}}\|h\|^2. \tag{3.2.19}$$

When $L\leq 2$, we have $\theta=2$, and (3.2.19) implies that

$$h^T\omega''(v)h\geq R^{-2}\|h\|^2. \tag{3.2.20}$$

Now let $L\geq 3$, so that $\theta=1+\delta$, $\delta=\frac{1}{\log(L)}\in(0,1)$. Then

$$\begin{aligned}
\sum_{s=1}^L\|v_s\|_2^{2-\theta} &= \sum_{s=1}^L\|v_s\|_2^{1-\delta}\leq\max_{t_s\geq0,\sum_{s=1}^L t_s\leq R}\sum_{s=1}^L t_s^{1-\delta}\\
&= L(R/L)^{1-\delta}=R^{1-\delta}L^\delta=R^{1-\delta}\exp\{1\},
\end{aligned}$$

so that (3.2.19) results in the relation

$$h^T\omega''(v)h\geq(\theta-1)\exp\{-1\}R^{-2}\|h\|^2, \tag{3.2.21}$$

which combines with (3.2.20) to imply (3.2.16) and thus, the fact that $\omega(\cdot)$ is κ-strongly convex w.r.t. $\|\cdot\|$ on V. The relation

$$v \in V \Rightarrow \omega(v) - \omega(0) - v^T\omega'(0) \equiv \omega(v) = \theta^{-1}R^{-\theta}\sum_{s=1}^{L}\|v^s\|_2^\theta \le D \equiv \theta^{-1}$$

is evident (note that $\theta > 1$ and $\sum\limits_s \|v^s\|_2 \le R$ for $v \in V$). The fact that Ω given by (3.2.9) is an upper bound on D/κ is equally evident. ∎

3.2.2 The setup for MP

The outlined rules equip every block Z_p with a certain norm $\|\cdot\|_{(p)}$ on $\mathbf{R}^{d_p}$, distance-generating function $\omega_p(\cdot)$ on Z_p which is κ_p-strongly convex on Z_p w.r.t. $\|\cdot\|_{(p)}$, starting point $z^{(p)} \in Z_p$, an upper bound D_p on the quantity $D_{z^{(p)}}[Z_p]$, and an upper bound Ω_p on the complexity parameter d_p/κ_p. The setup for MP as applied to problem (3.1.1) with the bilinear cost function (3.2.2) is "assembled" from the outlined data, specifically, as follows.

• Let L_{pq} be the norm of the linear mapping

$$v \mapsto B^{pq}v : \mathbf{R}^{d_q} \to \mathbf{R}^{d_p}$$

induced by the norm $\|\cdot\|_{(q)}$ on the argument space and the norm $\|\cdot\|_{(p)}^*$ (the norm conjugate to $\|\cdot\|_{(p)}$) on the image space:

$$L_{pq} = \max_{u\in\mathbf{R}^{d_q}}\left\{\|B^{pq}u\|_{(p)}^* : \|u\|_{(q)} \le 1\right\},$$

or, recalling the definition of a conjugate norm,

$$L_{pq} = \max_{u\in\mathbf{R}^{d_q}, v\in\mathbf{R}^{d_p}}\left\{v^TB^{pq}u : \|v\|_{(p)} \le 1, \|u\|_{(q)} \le 1\right\} = L_{qp}, \tag{3.2.22}$$

where the concluding equality is readily given by the fact that $B^{qp} = -[B^{pq}]^T$.

• Let us look at "assemblings" of the form

$$\begin{array}{c} z_0 = (z^{(1)}, ..., z^{(k)}),\ \omega(z) = \sum\limits_{p=1}^{k}\gamma_p\omega_p(z[p]), \\ \|z\| = \sqrt{\sum\limits_{p=1}^{k}\mu_p^2\|z[p]\|_{(p)}^2} \quad \left[\Leftrightarrow \|z\|_* = \sqrt{\sum\limits_{p=1}^{k}\mu_p^{-2}[\|z[p]\|_{(p)}^*]^2}\right] \end{array} \tag{3.2.23}$$

where $\mu_p > 0$, $\gamma_p > 0$ are parameters of the construction. We can easily express the bounds on the associated quantities $D_{z_0}[Z]$, κ, and L in terms of μ, γ. A straightforward computation, based on Proposition 3.3, implies that

$$D_{z_0}[\omega] \le \widetilde{D} = \sum_p \gamma_p D_p,\ \kappa \ge \widetilde{\kappa} = \min_p \frac{\gamma_p\kappa_p}{\mu_p^2},\ L \le \widetilde{L} = \lambda_{\max}\left(\left[\mu_p^{-1}\mu_q^{-1}L_{pq}\right]_{p,q}\right), \tag{3.2.24}$$

where $\lambda_{\max}(A)$ is the maximal eigenvalue of a symmetric matrix A. Now, what matters for the error bound of MP, the setup being given by $\omega(\cdot)$, $\|\cdot\|$, z_0, is the quantity $\kappa^{-1}D_{z_0}[Z]L$ – the less it is, the better. It is natural to look for the assembling which results in the smallest possible

upper bound $\widetilde{\kappa}^{-1}\widetilde{D}\widetilde{L}$ on this quantity. This problem can be easily solved; the optimal solution is given by (3.2.23) with the parameters γ_p, μ_p defined as

$$\gamma_p = \frac{\sigma_p}{D_p},\ \mu_p = \sqrt{\gamma_p\kappa_p},\ \text{where } \sigma_p = \frac{\sum\limits_q M_{pq}}{\sum\limits_{p,q} M_{pq}} \text{ and } M_{pq} = L_{pq}\sqrt{\frac{D_pD_q}{\kappa_p\kappa_q}}. \tag{3.2.25}$$

For the resulting assembling, one has

$$\widetilde{\kappa} = \widetilde{D} = 1, \quad \widetilde{L} = \sum_{p,q} L_{pq}\sqrt{\frac{D_pD_q}{\kappa_p\kappa_q}}. \tag{3.2.26}$$

The efficiency estimate for the resulting MP algorithm as applied to (3.1.1), (3.2.2) is

$$\mathrm{DualityGap}(x^t, y^t) \le \frac{\sqrt{2}\sum\limits_{p,q} L_{pq}\sqrt{\frac{D_pD_q}{\kappa_p\kappa_q}}}{t} \le \frac{\sqrt{2}\sum\limits_{p,q} L_{pq}\sqrt{\Omega_p\Omega_q}}{t}. \tag{3.2.27}$$

Chapter 4

Saddle Point Reformulation of SVM Models

4.1 Kernel-generated SVM models

As we have seen in Chapter 2, Statistical Learning Theory in its SVM-oriented form suggests to associate with a given training sample $S = \{(x_i, y_i)\}_{i=1}^{\ell}$, an optimization problem of the form

$$\begin{array}{c} \max\limits_{\gamma,w,b} \left\{\gamma : \|w\|_*^2 + R^{-2}\|\xi(w,b,\gamma)\|_{p_*}^2 \le 1\right\}, \\ \xi_i(w,b,\gamma) = \max[0, \gamma - y_i(\langle w, x_i\rangle + b)] \end{array} \qquad (P_p(S))$$

and to use the augmented classifier (2.3.23) given by an optimal solution to this problem. In the above optimization problem,

- $x_i \in X \subset E$ are the feature vectors from the sample, y_i are their labels, and X is a subset in a given normed space $(E, \|\cdot\|)$ known to support the marginal distribution of the feature vectors;
- $R < \infty$ is (a given upper bound on) the radius of $\|\cdot\|$-ball, centered at the origin, which contains X;
- The design variable w varies in the space E^* dual to E, and $\|\cdot\|_*$ is the norm on E^* dual to the norm $\|\cdot\|$ on E;
- $p \in [2, \infty]$ is a given parameter (which should be finite in the case when X is infinite), and $p_* = \frac{p}{p-1}$.

4.1.1 The kernel-generated case

In the majority of SVM models, E is a specific inner product space defined as follows:

1. We start with
   - The *attribute space* – a given abstract set $\widetilde{X}$ which, in order to avoid unessential technical difficulties, we assume to be finite. The points of $\widetilde{X}$ are called *attribute vectors* (usually they indeed are vectors, since $\widetilde{X}$ typically is given as a subset in certain $\mathbf{R}^d$).

- A *kernel* – a real-valued function $K(u,v) : \widetilde{X} \times \widetilde{X} \to \mathbf{R}$, which possesses the following properties:
  (a) $K(u,v) = K(v,u)$ for all $u, v \in \widetilde{X}$;
  (b) Whenever m is a positive integer and $u_1, ..., u_m$ are m distinct points from $\widetilde{X}$, the matrix $[K(u_i, u_j)]_{1 \le i,j \le m}$ is positive definite;
  (c) $K(u,u) \le R^2$ for all u and certain $R \in (0, \infty)$.

2. The kernel K defines an Euclidean space E, namely, as follows: the elements of E are real-valued functions $g[\cdot] : \widetilde{X} \to \mathbf{R}$, and the inner product on E is given by

$$\langle g[\cdot], h[\cdot] \rangle_K = \sum_{u,v \in \widetilde{X}} K(u,v) g[u] h[v] \tag{4.1.1}$$

   (due to the properties of K, this indeed is an inner product). The corresponding norm is

$$\|g\|_K = \sqrt{\sum_{u \in \widetilde{X}} K(u,u) g^2(u)} \tag{4.1.2}$$

3. Note that $\widetilde{X}$ can be naturally embedded into E; specifically, we can associate with a point $u \in \widetilde{X}$ the element $\phi(u) \in E$ given by

$$\phi(u)[v] = \begin{cases} 1, & v = u \\ 0, & v \ne u \end{cases},$$

   so that

$$\langle \phi(u), \phi(v) \rangle_K = K(u,v), \ \ \|\phi(u)\|_K = \sqrt{K(u,u)}. \tag{4.1.3}$$

4. Finally, the actual distribution $\widetilde{\mathcal{D}}$ to be learnt is a distribution on $\widetilde{X} \times \{-1, 1\}$, so that the actual training sample $\widetilde{S}$ is the sequence of examples $(\widetilde{x}_i, y_i)$ with $\widetilde{x}_i \in \widetilde{X}$ and $y_i \in \{-1, 1\}$. Identifying feature vectors $\widetilde{x}_i$ with their images $x_i = \phi(\widetilde{x}_i)$, we identify the "attribute space" examples $(\widetilde{x}_i, y_i)$ with their "feature space images" (x_i, y_i). With this identification of examples, $\widetilde{D}$ becomes identified with a distribution $\mathcal{D}$ on $X \times \{-1, 1\}$, where X is the image of $\widetilde{X}$ under the embedding $u \mapsto \phi(u) : \widetilde{X} \to E$, samples $\widetilde{S}$ drawn from $\widetilde{D}$ become identified with samples S drawn from $\mathcal{D}$, etc. Note that by (4.1.3) X is contained in the centered at the origin $\|\cdot\|_K$-ball of radius $R = \max\limits_{u \in \widetilde{X}} \sqrt{K(u,u)}$.

In the just outlined framework, E is an Euclidean space, so that E can be naturally identified with its dual; with this identification, the linear functional associated with $w \in E$ is just $w(x) = \langle w, x \rangle_K$, $x \in E$. Needless to say that with this identification the norm dual to $\|\cdot\|_K$ is this norm itself, so that problem $(P_p(S))$ reads

$$\begin{array}{c} \max\limits_{\gamma,w,b} \left\{ \gamma : \langle w, w \rangle_K + R^{-2} \|\xi(w,b,\gamma)\|_{p_*}^2 \le 1 \right\}, \\ \xi_i(w,b,\gamma) = \max[0, \gamma - y_i(\langle w, x_i \rangle_K + b)] \end{array} \tag{4.1.4}$$

where $x_i = \phi(\widetilde{x}_i) \in E$ are given points from X ("feature vectors of the examples") and y_i are their labels.

4.1.2 Reformulating problem (4.1.4)

Let γ, w, b be a feasible solution to (4.1.4). When replacing w with its orthogonal projection w' onto the linear span of x_i, $i = 1, ..., \ell$, we preserve the validity of the constraints and can only reduce the value of the objective. It follows that we lose nothing when adding to the constraints of (4.1.4) an additional requirement $w \in \mathrm{Lin}(x_1, ..., x_\ell)$, or, equivalently, the requirement that $w = \sum_{j=1}^{\ell} \alpha_j y_j \widetilde{x}_j$. Treating α_j and γ, b as our new variables, problem (4.1.4) becomes

$$\gamma_{\mathrm{opt}} = \max_{\alpha\in\mathbf{R}^\ell,\gamma,b,\xi} \left\{ \gamma : \begin{array}{c} \sum_{i,j=1}^{\ell} K(\widetilde{x}_i, \widetilde{x}_j) y_i y_j \alpha_i \alpha_j + R^{-2}\|\xi\|_{p_*}^2 \le 1 \\ \xi \ge 0 \\ \xi_i \ge \gamma - y_i \left(\sum_j K(\widetilde{x}_i, \widetilde{x}_j) y_j \alpha_j + b \right), \\ i = 1, ..., \ell \end{array} \right\} \tag{4.1.5}$$

Our first goal is to rewrite this problem in an equivalent, better suited for our further goals, form and, second, to add some extra flexibility to the resulting problem.

Equivalent reformulation of (4.1.5). Note that the optimal value in (4.1.5) clearly is positive, and the problem can be rewritten in the form of

$$\mathrm{Opt} = \min_{\alpha\in\mathbf{R}^\ell,b,\xi} \left\{ \sqrt{\sum_{i,j=1}^{\ell} K(\widetilde{x}_i, \widetilde{x}_j) y_i y_j \alpha_i \alpha_j + R^{-2}\|\xi\|_{p_*}^2} : \begin{array}{c} \xi \ge 0 \\ \xi_i \ge 1 - y_i \sum_j K(\widetilde{x}_i, \widetilde{x}_j) y_j \alpha_j + b, \\ i = 1, ..., \ell \end{array} \right\}. \tag{4.1.6}$$

Indeed, if $(\alpha, \gamma > 0, b, \xi)$ is a feasible solution to (4.1.5), then the collection $(\gamma^{-1}\alpha, \gamma^{-1}b, \gamma^{-1}\xi)$ is a feasible solution to (4.1.6) with the value of the objective $\le \gamma^{-1}$; vice versa, if (α, b, ξ) is a feasible solution to (4.1.6) with certain value of the objective a, then the collection $(a^{-1}\alpha, a^{-1}, a^{-1}b, a^{-1}\xi)$ is a feasible solution to (4.1.5) with the value of the objective a^{-1}. It follows that (4.1.5) has positive optimal value the optimal values in (4.1.5) and (4.1.6) are linked by the relation

$$\mathrm{Opt} = \frac{1}{\gamma_{\mathrm{opt}}}, \tag{4.1.7}$$

and whenever (α, b, ξ) is an optimal solution to (4.1.6), the collection $\mathrm{Opt}^{-1}(\alpha, 1, b, \xi)$ is an optimal solution to (4.1.5).

From an optimization program to a classifier. Note that the classifier yielded by an optimal solution (α^*, b^*, ξ^*) to (4.1.6) at every attribute vector $\widetilde{x}$ distinct from the attribute vectors $\widetilde{x}_i$, $i = 1, ..., \ell$ from the training sample equals to $f_*(\widetilde{x}) = \langle \sum_j \alpha_j^* y_j x_j, x \rangle_K + b^*$ (see Statement (!) in Section 2.3.3). Recalling the definition of $\langle \cdot, \cdot \rangle_K$, we see that

$$\widetilde{x} \notin \{\widetilde{x}_i\}_{i=1}^{\ell} \Rightarrow f_*(\widetilde{x}) = \sum_{j=1}^{\ell} K(\widetilde{x}, \widetilde{x}_j) y_j \alpha_j^* + b^*, \tag{4.1.8}$$

so that not only the problem yielding the classifier, but the classifier itself can be expressed solely in terms of the kernel K.

Adding flexibility. We have seen in Chapter 2 that the "soft margin" SVM approach results in optimization problem $(P_p(S))$ which, in the kernel-generated case, can be posed as (4.1.6); in fact, theory suggests a specific choice of p, namely, $p = 2$. Note, however, that the Statistical Learning Theory underlying this formulation is, in some sense, too "rough" to provide detailed SVM-oriented optimization models. For this reason, in the SVM practice, one adds some flexibility to (4.1.6), which allows to "adjust", to some extent, the SVM optimization program to a specific classification problem in order to get better generalization error, or to get a sparse (with a relatively small number of nonzero α_i's) separator, etc.

The major "adjustable components" in (4.1.6) are as follows.

1. *Adding flexibility to the objective,* specifically, by replacing the objective

$$\sqrt{\sum_{i,j=1}^{\ell} K(\widetilde{x}_i, \widetilde{x}_j) y_i y_j \alpha_i \alpha_j + R^{-2}\|\xi\|_{p_*}^2}$$

with one of the form

$$\sqrt{\sum_{i,j=1}^{\ell} K(\widetilde{x}_i, \widetilde{x}_j) y_i y_j \alpha_i \alpha_j + \Gamma\|\xi\|_{p_*}^2}, \tag{4.1.9}$$

where $\Gamma > 0$ is a control parameter. When building a classifier, one solves problem (4.1.6) with the objective modified according to (4.1.9) for a series of values of Γ and chooses "seemingly the best" of the corresponding classifiers by applying additional tests, e.g., by measuring the quality of a classifier on a *validation sample*, see Section 5.2.2.
Observe that adjusting the parameter $\Gamma > 0$ in the resulting parametric optimization problem

$$\min_{\alpha\in\mathbf{R}^\ell, b, \xi} \left\{ \sqrt{\sum_{i,j=1}^{\ell} K(\widetilde{x}_i, \widetilde{x}_j) y_i y_j \alpha_i \alpha_j + \Gamma\|\xi\|_{p_*}^2} : \begin{array}{c} \xi \geq 0 \\ \xi_i \geq 1 - y_i \left(\sum_j K(\widetilde{x}_i, \widetilde{x}_j) y_j \alpha_j + b \right), \\ i = 1, ..., \ell \end{array} \right\}, \tag{4.1.10}$$

is equivalent to adjusting parameter $\rho > 0$ in the parametric problem

$$\min_{\alpha\in\mathbf{R}^\ell, b, \xi} \left\{ \|\xi\|_{p_*} : \begin{array}{c} \|\alpha\|_{2,y,K} \equiv \sqrt{\sum_{i,j} K(\widetilde{x}_i, \widetilde{x}_j) y_i y_j \alpha_i \alpha_j} \leq \rho \\ \xi \geq 0 \\ \xi_i \geq 1 - y_i \left(\sum_j K(\widetilde{x}_i, \widetilde{x}_j) y_j \alpha_j + b \right), \\ i = 1, ..., \ell \end{array} \right\}. \tag{4.1.11}$$

Indeed, let α^Γ, b^Γ be the (α, b)-component of an optimal solution to (4.1.10). By evident reasons, with $\rho = \|\alpha^\Gamma\|_{2,y,K}$, every optimal solution to (4.1.11) is optimal for (4.1.10). It would be easy to show that under mild regularity assumptions the opposite is also true: the optimal solution of (4.1.11) corresponding to a given value $\rho > 0$ of the parameter, is an optimal solution to (4.1.10) for a properly chosen value $\Gamma(\rho) > 0$ of Γ, but we do not need such a result: from what we have already seen it is clear that the parametric problem (4.1.11) is *at least* as flexible as (4.1.10), so that we lose nothing when focusing on the former problem rather than on the latter one.

2. *Replacing $\|\alpha\|_{2,y,K}$ with another norm $\|\alpha\|$ on $\mathbf{R}^\ell$* adds even more flexibility to (4.1.11). The resulting parametric family of optimization problems is

$$\min_{\alpha,b,\xi}\left\{\|\xi\|_{p_*}:\ \begin{array}{c}\|\alpha\|\le\rho\\ \xi\ge 0\\ \xi_i\ge 1-((Q\alpha)_i+by_i)\,,\\ i=1,...,\ell\end{array}\right\}, \tag{4.1.12}$$

where

$$Q=[Q_{ij}=y_iK(\widetilde{x}_i,\widetilde{x}_j)y_j]_{1\le i,j\le\ell}\in\mathbf{R}^{\ell\times\ell}. \tag{4.1.13}$$

Aside of the choice $\|\alpha\|=\|\alpha\|_{2,y,K}$, a notable candidate to the role of the norm $\|\cdot\|$ is the ℓ_1-norm

$$\|\alpha\|_1=\sum_{i=1}^{\ell}|\alpha_i|;$$

in many cases, this choice results in "sparse classifier" – with much smaller number of nonzero coefficients α_i^* in (4.1.8) than the one yielded by the choice $\|\cdot\|=\|\cdot\|_{2,y,K}$.

Remark 4.1 *When $\|\cdot\|=\|\cdot\|_{2,y,K}$, from optimality conditions as applied to* (4.1.12) *it follows that at the optimum, all α-variables are nonnegative. Thus, in the case of $\|\cdot\|=\|\cdot\|_{2,y,K}$ we lose nothing when adding to the constraints of* (4.1.12) *the constraint $\alpha\ge 0$.*

4.1.3 Adjusting the kernel

In the outlined presentation, the kernel K was treated as a given in advance entity. In SVM practice, the choice of the kernel, which is *the* crucial component of the approach, is based mainly on utilizing a specific structure of a particular classification problem at hand. However, this structure usually suggests to use a kernel from certain parametric family, e.g., from the family of Gaussian kernels

$$K(\widetilde{x},\widetilde{y})=\exp\{\|\widetilde{x}-\widetilde{y}\|_2^2/(2\sigma^2)\}\qquad[\widetilde{x},\widetilde{y}\in\widetilde{X}\subset\mathbf{R}^d]$$

while the question of how to choose kernel's parameter(s) (e.g., σ for a Gaussian kernel) remains open. It would be attractive to make these parameters part of decision variables in problem (4.1.12), but such an attempt usually results in a *nonconvex* and thus difficult ro solve optimization model. Instead, practitioners usually adjust the kernel by solving (4.1.12) for a number of different values of the kernel parameters and then use "seemingly the best" of the resulting classifiers (cf. the situation with tuning the parameter ρ). To the best of our knowledge, there exists only one particular case where adjusting the kernel can be made a part of the optimization process. This case is as follows. Assume that we are given L candidate kernels $K_s(u,v):\widetilde{X}\times\widetilde{X}\to\mathbf{R}$, $s=1,...,L$. Further, let $\lambda_s>0$ be "scale parameters" for these kernels. According to the outlined scheme, every one of the kernels $\lambda_sK_s(\cdot,\cdot)$ defines a Euclidean "feature space" $(E_s,\langle\cdot,\cdot\rangle_s)$; the elements of this space are real-valued functions $\phi[u]$ on $\widetilde{X}$, and the inner product is

$$\langle\phi[\cdot],\psi[\cdot]\rangle_s=\lambda_s\sum_{u,v\in\widetilde{X}}K_s(u,v)\phi[u]\psi[v].$$

We can now consider the direct product

$$E=E_1\times...\times E_L$$

of the resulting Euclidean spaces as our new feature space; the elements of this space are ordered collections $\overrightarrow{\phi} = (\phi_1, ..., \phi_L)$ of real-valued functions on $\overrightarrow{X}$, and the inner product is

$$\langle \overrightarrow{\phi}, \overrightarrow{\psi} \rangle_E = \sum_{s=1}^{L} \langle \phi_s, \psi_s \rangle_s.$$

We now can embed the set $\widetilde{X}$ into E via the mapping

$$\widetilde{x} \mapsto \overrightarrow{x} \equiv \overrightarrow{\phi}(\widetilde{x}) = (\phi_1(\widetilde{x}), ..., \phi_L(\widetilde{x})) \in E,$$

where, as above,

$$\phi(\widetilde{x})[u] = \begin{cases} 1, & u = \widetilde{x} \\ 0, & u \neq \widetilde{x} \end{cases}.$$

Note that when $\widetilde{x}', \widetilde{x}'' \in \widetilde{X}$, we have

$$\langle \overrightarrow{x'}, \overrightarrow{x''} \rangle_E = \sum_{s=1}^{L} \lambda_s K_s(\widetilde{x}', \widetilde{x}''). \tag{4.1.14}$$

With this embedding in the role of our previous embedding $\widetilde{x} \mapsto x = \phi(\widetilde{x})$, the analogy of problem (4.1.4) is

$$\max_{\gamma, \{w^s\}_{s=1}^L, b, \xi} \left\{ \gamma : \begin{array}{l} \sum\limits_{s=1}^{L} \lambda_s \langle w^s, w^s \rangle_{K_s} + \Gamma \|\xi\|_{p_*}^2 \leq 1 \\ \xi \geq 0 \\ \xi_i \geq \gamma - y_i \left(\sum\limits_{s=1}^{L} \lambda_s \langle w^s, x_i \rangle_{K_s} + b \right), \qquad i = 1, ..., \ell \end{array} \right\}. \tag{4.1.15}$$

Same as in the single-kernel case, the latter problem is equivalent to the problem (cf. (4.1.6))

$$\min_{\substack{\{\alpha^s \in \mathbf{R}^\ell\}_{s=1}^L, \\ b, \xi}} \left\{ \sqrt{\sum_{i,j=1}^{\ell} \sum_{s=1}^{L} \lambda_s K_s(\widetilde{x}_i, \widetilde{x}_j) y_i y_j \alpha_i^s \alpha_j^s + \Gamma \|\xi\|_{p_*}^2} : \begin{array}{l} \xi \geq 0 \\ \xi_i \geq 1 - y_i \left(\sum\limits_{j=1}^{\ell} \sum\limits_{s=1}^{L} \lambda_s K_s(\widetilde{x}_i, \widetilde{x}_j) y_j \alpha_j^s + b \right) \end{array} \right\}, \tag{4.1.16}$$

and the resulting problem can be viewed as a member of the parametric family (cf. (4.1.11))

$$\min_{\{\alpha^s \in \mathbf{R}^\ell\}_{s=1}^L, b, \xi} \left\{ \|\xi\|_{p_*} : \begin{array}{l} \sqrt{\sum\limits_{i,j=1}^{\ell} \sum\limits_{s=1}^{L} \lambda_s K_s(\widetilde{x}_i, \widetilde{x}_j) y_i y_j \alpha_i^s \alpha_j^s} \leq \rho \\ \xi \geq 0 \\ \xi_i \geq 1 - y_i \left(\sum\limits_{j=1}^{\ell} \sum\limits_{s=1}^{L} \lambda_s K_s(\widetilde{x}_i, \widetilde{x}_j) y_j \alpha_j^s + b \right) \end{array} \right\} \tag{4.1.17}$$

with $\rho > 0$ being the parameter. The classifier yielded by a candidate solution to this problem, evaluated at a point $\widetilde{x}$ distinct from all attribute vectors $\widetilde{x}_i$, $i = 1, ..., \ell$, from the training sample, is (cf. (4.1.8))

$$f(\widetilde{x}) = \sum_{j=1}^{\ell} \sum_{s=1}^{L} \lambda_s K_s(\widetilde{x}, \widetilde{x}_j) y_j \alpha_j^s + b. \tag{4.1.18}$$

For the time being, we have tacitly considered the weights λ_s as parameters of the construction; starting from this point, let us treat them as decision variables of (4.1.17). When multiplying

the weights λ_s *and* ρ by a common positive factor, we clearly do not change the optimization problem; therefore we lose nothing when normalizing λ's as

$$\sum_s \lambda_s = 1.$$

We, therefore, have arrived at the parametric optimization problem

$$\min_{\{\alpha^s\in\mathbf{R}^\ell\}_{s=1}^L,b,\xi,\lambda}\left\{\|\xi\|_{p_*}:\begin{array}{l}\sqrt{\sum\limits_{i,j=1}^{\ell}\sum\limits_{s=1}^{L}\lambda_s K_s(\widetilde{x}_i,\widetilde{x}_j)y_iy_j\alpha_i^s\alpha_j^s}\le\rho\\ \xi\ge 0\\ \xi_i\ge 1-y_i\left(\sum\limits_{j=1}^{\ell}\sum\limits_{s=1}^{L}\lambda_s K_s(\widetilde{x}_i,\widetilde{x}_j)y_j\alpha_j^s+b\right)\\ \lambda>0,\sum\limits_s\lambda_s=1\end{array}\right\}. \tag{4.1.19}$$

As it is written, this problem is *not* convex, due to the presence of products of λ's and α's. We are about to demonstrate that this difficulty can be overcome. Indeed, let us pass in (4.1.19) from variables α_j^s to $\beta_j^s=\lambda_s\alpha_j^s$. The resulting problem is

$$\min_{\{\beta^s\in\mathbf{R}^\ell\}_{s=1}^L,b,\xi,\lambda}\left\{\|\xi\|_{p_*}:\begin{array}{l}\sqrt{\sum\limits_{s=1}^{L}\lambda_s^{-1}\left[\sum\limits_{i,j=1}^{\ell}K_s(\widetilde{x}_i,\widetilde{x}_j)y_iy_j\beta_i^s\beta_j^s\right]}\le\rho\\ \xi\ge 0\\ \xi_i\ge 1-y_i\left(\sum\limits_{j=1}^{\ell}\sum\limits_{s=1}^{L}K_s(\widetilde{x}_i,\widetilde{x}_j)y_j\beta_j^s+b\right)\\ \lambda>0,\sum\limits_s\lambda_s=1\end{array}\right\}. \tag{4.1.20}$$

Now, for reals a_s, $s=1,...,L$, the quantity

$$\inf_\lambda\left\{\sum_{s=1}^L\lambda_s^{-1}a_s^2:\lambda>0\sum_{s=1}^L\lambda_s=1\right\}$$

clearly equals to

$$(\sum_s|a_s|)^2.$$

This observation as applied with $a_s=\sqrt{\sum\limits_{i,j}K_s(\widetilde{x}_i,\widetilde{x}_j)y_iy_j\beta_i^s\beta_j^s}$ allows to carry out partial optimization in λ in (4.1.20); the resulting problem is

$$\min_{\{\beta^s\in\mathbf{R}^\ell\}_{s=1}^L,b,\xi}\left\{\|\xi\|_{p_*}:\begin{array}{l}\sum\limits_{s=1}^{L}\|\beta^s\|_{2,y,K_s}\le\rho\\ \xi\ge 0\\ \xi_i\ge 1-y_i\left(\sum\limits_{j=1}^{\ell}\sum\limits_{s=1}^{L}K_s(\widetilde{x}_i,\widetilde{x}_j)y_j\beta_j^s+b\right)\end{array}\right\}. \tag{4.1.21}$$

This parametric problem is convex; moreover, this problem is of the form (4.1.12), with the vector $(\beta^1,...,\beta^L)$ in the role of α, the norm

$$\|(\beta^1,...,\beta^L)\|=\sum_{s=1}^L\|\beta^s\|_{2,y,K_s}$$

in the role of $\|\cdot\|$ and the $\ell \times L\ell$ matrix

$$[Q_1, ..., Q_L], \ Q_s = [y_i K_s(\widetilde{x}_i, \widetilde{x}_j) y_j]_{1 \le i,j \le \ell} \tag{4.1.22}$$

in the role of Q. Note that the feasible domain in the variables β as given by the norm constraint $\|\beta\| \le \rho$ of the resulting optimization problem

$$\min_{\beta=(\beta^1,...,\beta^L)\in\mathbf{R}^{L\ell}, b, \xi} \left\{ \|\xi\|_{p_*} : \begin{array}{l} \|\beta\| \equiv \sum\limits_{s=1}^{L} \|\beta^s\|_{2,y,K_s} \le \rho \\ \xi \ge 0 \\ \xi_i \ge 1 - ((Q\beta)_i + b y_i) \end{array} \right\} \tag{4.1.23}$$

is an extended simplex. Note also that the classifier (4.1.18) is readily given by the β-variables:

$$f(\widetilde{x}) = \sum_{j=1}^{\ell} \sum_{s=1}^{L} K_s(\widetilde{x}, \widetilde{x}_j) y_j \beta_j^s + b. \tag{4.1.24}$$

4.2 Saddle point reformulation of SVM models

Our current goal is to reformulate the SVM optimization problem (4.1.12) (which includes, as particular cases, all other SVM problems considered so far) in a saddle point form, as required by the Mirror Prox Algorithm. We start with the following simple observation:

Lemma 4.1 *Let $u \in \mathbf{R}^d$, and let ξ be the vector with coordinates*

$$\xi_i = \max[0, u_i].$$

Then, for every $p \in [1, \infty]$, one has

$$\|\xi\|_{p_*} = \max_{\lambda \in B_p^+(1)} \lambda^T u \equiv \max_{\lambda} \left\{ \lambda^T u : \lambda \ge 0, \|\lambda\|_p \le 1 \right\}; \tag{4.2.1}$$

here, as always, $p_* = \frac{p}{p-1}$.

Proof. By Hölder Inequality, whenever $\lambda \in B_p^+(1)$, we have

$$\|\xi\|_{p_*} \ge \|\lambda\|_p \|\xi\|_{p_*} \ge \lambda^T \xi \ge \lambda^T u,$$

where the concluding inequality follows from the fact that $\lambda \ge 0$ and $\xi \ge u$. Thus, the right hand side in (4.2.1) is $\le$ the left hand side one. To prove the opposite inequality, it suffices to consider the case when $\xi \ne 0$ (otherwise the desired inequality is evident). Assuming $1 < p < \infty$, setting

$$\lambda_i = \frac{\xi_i^{p_*-1}}{\left(\sum\limits_j \xi_j^{p_*} \right)^{\frac{1}{p}}},$$

and taking into account that $p(p_* - 1) = p_*$, we get $\lambda \ge 0$,

$$\|\lambda\|_p = \frac{\sum\limits_i \xi_i^{p_*}}{\left(\sum\limits_i \xi_i^{p_*} \right)} = 1$$

and

$$\sum_i \lambda_i u_i = \frac{\sum\limits_{i:\xi_i>0} \xi_i^{p_*-1} u_i}{\left(\sum\limits_j \xi_j^{p_*}\right)^{\frac{1}{p}}} = \frac{\sum\limits_{i:\xi_i>0} \xi_i^{p_*-1}\xi_i}{\left(\sum\limits_j \xi_j^{p_*}\right)^{\frac{1}{p}}} = \|\xi\|_{p_*}^{p_*-\frac{p_*}{p}} = \|\xi\|_{p_*}.$$

Thus, the right hand side in (4.2.1) is $\geq$ the left hand side one. ∎

Now, for α, b fixed, the optimal, in terms of the objective, choice of ξ clearly is the vector with the coordinates

$$\xi_i = \max\left[0, 1 - ((Q\alpha)_i + by_i)\right],$$

and the corresponding value of the objective is the p_*-norm of this vector, that is, by Lemma 4.1, the quantity

$$F(\alpha, b) \equiv \max_{\lambda \in B_p^+(1)} \left[\sum_i \lambda_i - \lambda^T Q\alpha - b\sum_i \lambda_i y_i\right].$$

Thus, *problem* (4.1.12) *is nothing but the saddle point problem*

$$\min_{\alpha:\|\alpha\|\leq\rho, b\in\mathbf{R}} \max_{\lambda\in B_p^+(1)} \left[\sum_i \lambda_i - \lambda^T Q\alpha - b\sum_i \lambda_i y_i\right] \tag{4.2.2}$$

with bilinear cost function.

In fact, we can eliminate the variable b, thus arriving at a problem which is better suited for solving via MP. To this end, note that for α given, minimization of $F(\alpha, b)$ in $b \in R$ is just an explicit univariate convex program and as such can be easily solved by Bisection. Thus, problem (4.1.12), which is in fact,

$$\min_{\alpha:\|\alpha\|\leq\rho, b\in\mathbf{R}} F(\alpha, b)$$

can be reduced to the problem

$$\min_{\alpha:\|\alpha\|\leq\rho} \widetilde{F}(\alpha),\ \widetilde{F}(\alpha) = \inf_{b\in R} F(\alpha, b). \tag{4.2.3}$$

We now have

$$\begin{aligned}
\widetilde{F}(\alpha) &= \inf_{b\in R} F(\alpha, b) = \inf_{b\in R} \max_{\lambda\in B_p^+(1)} \left[\sum_i \lambda_i - \lambda^T Q\alpha - b\sum_i \lambda_i y_i\right] \\
&= \max_{\lambda\in B_p^+(1)} \inf_{b\in R} \left[\sum_i \lambda_i - \lambda^T Q\alpha - b\sum_i \lambda_i y_i\right] \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{[by von Neumann Lemma]} \\
&= \max_{\lambda\in B_p^+(1)} \left[\sum_i \lambda_i - \lambda^T Q\alpha + \begin{cases} 0, & \sum\limits_i \lambda_i y_i = 0 \\ -\infty, & \sum\limits_i \lambda_i y_i \neq 0 \end{cases}\right] \\
&= \max_{\lambda\in B_p^+(1):\sum\limits_i y_i\lambda_i=0} \left[\sum_i \lambda_i - \lambda^T Q\alpha\right].
\end{aligned}$$

Consequently, problem (4.2.3) reduces to the saddle point problem

$$\min_{\alpha:\|\alpha\|\leq\rho} \max_{\lambda\in B_p^+(1), \sum\limits_i y_i\lambda_i=0} \left[\sum_i \lambda_i - \lambda^T Q\alpha\right] \tag{4.2.4}$$

We have therefore arrived at the following

Theorem 4.1 *The SVM problem* (4.1.12) *can be reduced to the saddle point problem* (4.2.4) *with bilinear cost function.*

4.3 "Plain" SVM models

In the previous Section, we have been focusing on the case where the feature space E is generated by a kernel, and we have been seeking for an affine classifier in the form of a linear combination of feature vectors from the training sample. An alternative is to represent E as the standard coordinate space $\mathbf{R}^N$ with the inner product $\langle x, y\rangle = x^T y$ and represent tentative affine classifiers in the form $f(x) = w^T x + b$, without taking care of representing w as a linear combination of sample feature vectors and even of whether such a representation is possible. We may think of this alternative as of the case where

- the set $\widetilde{X}$ of possible values of attribute vectors is a subset of $\mathbf{R}^N$, the feature vectors are identical with the attribute ones, and
- the kernel is "linear":

$$K(\widetilde{x}_i, \widetilde{x}_j) = \widetilde{x}_i^T \widetilde{x}_j. \tag{4.3.1}$$

In this case the Euclidean space E as defined in Section 4.1.1 can be naturally identified with the linear span of $\widetilde{X}$. Indeed, we can associate with a function $g[\cdot] : \widetilde{X} \to \mathbf{R}$ the vector $x_g = \sum_{\widetilde{x}\in\widetilde{X}} g[\widetilde{x}]\widetilde{x} \in \mathbf{R}^N$, thus identifying elements of E with vectors from $\mathbf{R}^N$. It is immediately seen that with this identification, the kernel-generated inner product in E becomes the standard inner product $x^T y$, the mapping $\widetilde{X} \mapsto X$ becomes the identity, and E itself becomes the linear span of $\widetilde{X}$. If the latter space is less than the entire $\mathbf{R}^N$, we lose nothing by extending E to $\mathbf{R}^N$.

The situation in question is a particular case of the general kernel-generated situation considered in Section 4.1.1. In this situation, problem (4.1.4), which was the starting point of all our developments in Section 4.1.1, reads

$$\begin{array}{c}\max\limits_{\gamma, w\in\mathbf{R}^N, b} \left\{\gamma : w^T w + R^{-2}\|\xi(w,b,\gamma)\|_{p_*}^2 \le 1\right\}, \\ \xi_i(w,b,\gamma) = \max[0, \gamma - y_i(w^T x_i + b)] \\ .\end{array} \tag{4.3.2}$$

Our first step in Section 4.1.1 was to observe that at optimality, w is a linear combination of x_i, so that we can pass from the decision vector w to the vector α of coefficients in the representation $w = \sum_i \alpha_i y_i x_i$. In our current situation, we can skip this step and work with decision vector w "as it is". Depending on the structure of the data, this modification may have two important advantages. First, it may happen that the dimension N of our attribute≡feature space $\mathbf{R}^N$ is much less than the cardinality ℓ of the training sample; whenever this is the case, problem (4.3.2) is of much smaller dimension than the associated "α-problem" (4.1.5). Second, in many cases, especially in large-scale ones, the feature vectors are sparse, and as we shall see below, when working with w-variables "as they are", we have better possibilities to utilize this sparsity than when working with α-variables.

We now can process problem (4.3.2) in exactly the same fashion as in Sections 4.1, 4.2,

arriving first at the optimization problem

$$\min_{w\in\mathbf{R}^N,b,\xi}\left\{\|\xi\|_{p_*}:\begin{array}{c}\|w\|\le\rho\\ \xi\ge 0\\ \xi_i\ge 1-y_i\left(w^Tx_i+b\right),\\ i=1,...,\ell\end{array}\right\} \tag{4.3.3}$$

(this is the "w-analogy" of problem (4.1.12)) and then converting the latter problem to the saddle point form

$$\min_{w:\|w\|\le\rho}\max_{\lambda\in B_p^+(1),\sum_i y_i\lambda_i=0}\left[\sum_i\lambda_i[1-y_iw^Tx_i]\right] \tag{4.3.4}$$

(this is the analogy of (4.2.4)). In order to distinguish the current SVM models from those derived in Sections 4.1, 4.2, we shall refer to the former models as to *plain* SVM models, and to the latter ones as to *kernel* SVM models. Note that in fact both types of models are associated with kernels (the plain ones – with the linear kernel (4.3.1)); the actual difference is in how we represent a classifier, either as a combination of the linear forms associated with training feature vectors (kernel models) or just as a linear form on $E=\mathbf{R}^N$ (plain models).

We are now in a position to explain what are the computational advantages, of plain models as compared to kernel ones in the case of sparse data. Indeed, let us compare the saddle point problem (4.3.4) with its kernel version, assuming, for the sake of definiteness, that we use in both cases the 2-norm of classifiers:

- *Plain model:*

$$\min_{w:w^Tw\le\rho^2}\max_{\lambda\in B_p^+(1),\sum_i y_i\lambda_i=0}\left[\sum_i\lambda_i-w^TP\lambda\right],\quad P=[y_1x_1,...,y_\ell x_\ell]$$

- *Kernel model $w\leftarrow P\alpha$:*

$$\min_{\alpha:\alpha^TQ\alpha\le\rho^2}\max_{\lambda\in B_p^+(1),\sum_i y_i\lambda_i=0}\left[\sum_i\lambda_i-\alpha^TQ\lambda\right],\quad Q=P^TP.$$

When solving the outlined saddle point problems by Mirror-Prox Algorithm (or any other first-order iterative method), the computational effort per step is dominated by the necessity to multiply given vectors by P and P^T in the case of plain model and multiply by Q, in the case of the kernel model. When P is sparse, Q is typically much less so, and the required multiplications in the case of the plain model are much cheaper computationally than in the case of the kernel one, provided that we work with Q "as a whole" and do not keep in mind the multiplicative representation $Q=P^TP$ (the latter is *not* directly suggested by the kernel model).

Chapter 5

Processing SVM Models via MP

In this chapter, we explain how to process the SVM models in their saddle point forms derived in Chapter 4 via the mirror Prox Algorithm, discuss the corresponding implementation issues and present some numerical results.

5.1 The strategy

5.1.1 The target

Observe that the saddle point reformulations (4.2.4), (4.3.4) of the kernel, and the corresponding plain SVM models are of the same generic form

$$\min_{\zeta:\|\zeta\|\le\rho} \underbrace{\max_{\lambda\in B_p^+(1),\sum_i y_i\lambda_i=0} \left[\mathbf{1}^T\lambda - \lambda^T Q\zeta\right]}_{F(\zeta)} \qquad \left[\mathbf{1}=(1,...,1)^T\in\mathbf{R}^\ell,\ B_p^+(1)=\left\{\xi\in\mathbf{R}^\ell : \xi\ge 0, \|\xi\|_p\le 1\right\}\right], \tag{5.1.1}$$

where

• *In the case of a kernel SVM model* from Section 4.1.1

$$Q=[y_iK(\widetilde{x}_i,\widetilde{x}_j)y_j]_{1\le i,j\le\ell}\in\mathbf{R}^{\ell\times\ell}, \tag{5.1.2}$$

$\zeta\in\mathbf{R}^\ell$ is what was called α in Section 4.1.2, the classifier yielded by a candidate solution (ζ,λ) to (5.1.1) is

$$f(\tilde{x})=\sum_{j=1}^{\ell} K(\widetilde{x},\widetilde{x}_j)y_j\zeta_j + b(\zeta), \tag{5.1.3}$$

where

$$b(\zeta)\in \operatorname*{Argmin}_b \|\xi[\zeta]\|_{p_*},\ (\xi[\zeta])_i=\max[0,1-(Q\zeta)_i-by_i] \tag{5.1.4}$$

and the norm $\|\zeta\|$ is either the kernel norm

$$\|\zeta\|_{2,y,K}=\sqrt{\zeta^TQ\zeta},$$

or it is another norm on $\mathbf{R}^\ell$ (from now on, we restrict this norm to be a standard $\|\cdot\|_r$-norm with $1\le r\le 2$), cf. (4.1.12), (4.1.13), (4.2.4);

• *In the case of an SVM model with adjustable kernel* from Section 4.1.3,

$$Q = \Big[\underbrace{[y_i K_1(\widetilde{x}_i, \widetilde{x}_j) y_j]_{i,j}}_{Q_1}, \underbrace{[y_i K_2(\widetilde{x}_i, \widetilde{x}_j) y_j]_{i,j}}_{Q_2}, ..., \underbrace{[y_i K_L(\widetilde{x}_i, \widetilde{x}_j) y_j]_{i,j}}_{Q_L}\Big] \in \mathbf{R}^{\ell \times L\ell}, \tag{5.1.5}$$

$\zeta \in \mathbf{R}^{L\ell}$ is what was called $\beta = (\beta^1, ..., \beta^L)$ in Section 4.1.3, the classifier yielded by a candidate solution (ζ, λ) to (5.1.1) is

$$f(\widetilde{x}) = \sum_{s=1}^{L} \sum_{j=1}^{\ell} \lambda_s K_s(\widetilde{x}, \widetilde{x}_j) y_j \zeta_j^s + b(\zeta), \tag{5.1.6}$$

where $\zeta^1, ..., \zeta^L$ are consecutive blocks of size ℓ in ζ and $b(\zeta)$ is given by (5.1.4), and $\|\zeta\|$ is the norm

$$\|\zeta\| = \sum_{s=1}^{L} \|\zeta^s\|_{2,y,K_s} \equiv \sum_{s=1}^{L} \sqrt{[\zeta^s]^T Q_s \zeta^s}, \tag{5.1.7}$$

cf. (4.1.23), (4.1.24), (4.2.4);
• *In the case of plain SVM models* from Section 4.3,

$$Q = \begin{bmatrix} y_1 x_1^T \\ y_2 x_2^T \\ \cdots \\ y_\ell x_\ell^T \end{bmatrix} \in \mathbf{R}^{\ell \times N}, \tag{5.1.8}$$

N being the dimension of the attribute≡feature space, $\zeta \in \mathbf{R}^N$ is what was called w in Section 4.3, the classifier yielded by a candidate solution (ζ, λ) to (5.1.1) is

$$f(\widetilde{x}) = \zeta^T \widetilde{x} + b(\zeta), \tag{5.1.9}$$

where $b(\zeta)$ is given by (5.1.4), and $\|\cdot\|$ is a norm on $\mathbf{R}^N$ (which from now on we restrict to be a standard $\|\cdot\|_r$-norm with $1 \le r \le 2$), cf. (4.3.3), (4.3.4).

5.1.2 The method

The setup

Our strategy is to solve problem (5.1.1) by the Mirror Prox Algorithm from Chapter 3 with the setup presented in Section 3.2. Specifically, our setup is as follows.

ζ-component. The domain X of ζ in (5.1.1) is the $\|\cdot\|$-ball of radius ρ in certain $\mathbf{R}^M$, and the norm for this component (we use in the MP setup) is exactly the norm $\|\cdot\|$. The distance-generating function for this component is chosen as follows:
• $\|\cdot\|$ *is the kernel-generated norm* $\|\cdot\|_{2,y,K}$ (kernel-generated SVM model with the kernel norm of $\alpha \equiv \zeta$):

$$\begin{array}{c} \omega_X(\zeta) = \frac{1}{2}\|\zeta\|_{2,y,K}^2 = \frac{1}{2}\sum\limits_{i,j} y_i y_j K(\widetilde{x}_i, \widetilde{x}_j)\alpha_i \alpha_j \\ \Big[\kappa = 1,\ D = \frac{1}{2}\rho^2\Big] \end{array}; \tag{5.1.10}$$

here and in what follows, κ, D are characteristic parameters of the pair "norm, distance-generating function" introduces in Chapter 3. Note that (5.1.10) is given by (3.2.5) with $p = 2$; indeed, we are in the situation where X is a ball of radius ρ in Euclidean space.

• $\|\zeta\|$ *is the norm* $\sum_{s=1}^{L}\|\zeta^s\|_{2,y,K_s}$ (SVM model with adjustable kernel, see (5.1.5) – (5.1.7)):

$$\begin{array}{c}\omega_X(\zeta)=\theta^{-1}\rho^{-\theta}\sum\limits_{s=1}^{L}\|\zeta^s\|_{2,y,K_s}^{\theta},\quad \theta=\begin{cases}2, & L\le 2\\ 1+\frac{1}{\log(L)}, & L\ge 3\end{cases}\\ \left[\kappa=(\theta-1)L^{1-\theta}\rho^{-2},\ D=\tfrac{1}{2}\rho^2\right]\end{array} \tag{5.1.11}$$

this relation is given by (3.2.9); indeed, we are in the case when X is an extended simplex.

• $\|\cdot\|$ *is the standard* $\|\cdot\|_r$*-norm,* $1\le r\le 2$ (kernel-generated SVM model with non-kernel norm of $\zeta\equiv\alpha$ and plain SVM model):

$$\begin{array}{c}\omega_X(\zeta)=\frac{1}{\widehat{r}}\sum\limits_{i=1}^{M}|\zeta_i|^{\widehat{r}},\quad \widehat{r}=\max\left[r,1+\frac{1}{2\log(M)}\right],\\ \left[(\widehat{r}-1)M^{\frac{2}{\widehat{r}}-\frac{2}{r}}\rho^{\widehat{r}-2},\ D=\tfrac{1}{\widehat{r}}\rho^{\widehat{r}}\right]\end{array}, \tag{5.1.12}$$

where M is the dimension of ζ (see (3.2.5) and note that what was called p in the latter formula is now denoted by r).

For the case of $r=1$, we have an alternative distance-generating function (see (3.2.8))

$$\begin{array}{rcl}\omega_X(\zeta) & = & \min\limits_{u,w}\left\{\sum\limits_i[u_i\log(u_i)+w_i\log(w_i)]:\ \begin{array}{l}u,w\ge 0\\ u-w=\rho^{-1}\zeta\\ \sum\limits_i[u_i+w_i]=1+\delta\end{array}\right\}\\ & = & \sum\limits_i\frac{\sqrt{\rho^{-2}\zeta_i^2+\theta(\zeta)}-\rho^{-1}\zeta_i}{2}\log\left(\frac{\sqrt{\rho^{-2}\zeta_i^2+\theta(\zeta)}-\rho^{-1}\zeta_i}{2}\right)\\ & & +\sum\limits_i\frac{\sqrt{\rho^{-2}\zeta_i^2+\theta(\zeta)}+\rho^{-1}\zeta_i}{2}\log\left(\frac{\sqrt{\rho^{-2}\zeta_i^2+\theta(\zeta)}+\rho^{-1}\zeta_i}{2}\right),\\ \theta(\zeta) & : & \sum\limits_i\sqrt{\rho^{-2}\zeta_i^2+\theta(\zeta)}=1+(2d)^{-1}\delta\\ & & \left[\kappa=\rho^{-2}(1+\delta),\ D=(1+\delta)\log(2M)\right]\end{array} \tag{5.1.13}$$

where M is the dimension of ζ and $\delta=$ 1.e-3.

λ-component. The domain Y of λ in (5.1.1) is the intersection of the nonnegative part of the unit p-ball in $\mathbf{R}^\ell$ and the hyperplane $\{\lambda:\sum_i y_i\lambda_i=0\}$ passing through the origin. For computational reasons, we restrict ourselves with the cases of $p=2$ and $p=\infty$ only[1)] and equip the space of λ's with the norm and the distance-generating function as given by (3.2.6), that is,

$$\|\lambda\|=\|\lambda_2\|\equiv\sqrt{\lambda^T\lambda}$$

and

$$\begin{array}{c}\omega_Y(\lambda)=\frac{1}{2}\lambda^T\lambda\\ \left[\kappa=1,\ D=\tfrac{1}{2}\ell^{1-\frac{2}{p}}\right].\end{array} \tag{5.1.14}$$

The starting point for MP is the origin in the (ζ,λ)-space.

After X, Y are equipped with the respective norms and distance-generating functions, the norm and the distance-generating function for the domain $Z=X\times Y$ of the saddle point problem (5.1.1) is built as explained in Section 3.2.2.

[1)] "Theoretically valid" range of p is $2\le p\le\infty$; replacing this set with its endpoints hardly makes any practical difference.

Computing the prox-mapping

Due to simplicity of the domain Z of the saddle point problem (5.1.1), computing the prox-mapping is a computationally easy task which requires just Bisection. Indeed, due to the direct product structure of Z and the additive structure of $\omega(\cdot)$, the computation reduces to solving a pair of optimization problems

$$\min_{\zeta:\|\zeta\|\le\rho}[\omega_X(\zeta)+\langle\xi,\zeta\rangle] \tag{5.1.15}$$

and

$$\min_{\lambda\ge0,\|\lambda\|_p\le1,\sum\limits_i y_i\lambda_i=0}\left[\omega_Y(\lambda)+\xi^T\lambda\right], \tag{5.1.16}$$

where $\langle\xi,\zeta\rangle$ is either the kernel-generated inner product (kernel-type SVM models with kernel-generated norm of ζ), or the standard inner product $\xi^T\zeta$ (kernel-type SVM models with r-norm of ζ and plain SVM models).
We explain next how to solve these problems in two particular cases, skipping explanation for the remaining (equally easy) situations.

Solving (5.1.15) in the case of (5.1.2). In view of (5.1.10), (5.1.15) reduces to

$$\min_{\zeta:\zeta^TQ\zeta\le\rho^2}\left\{\frac{1}{2}\zeta^TQ\zeta+\xi^TQ\zeta\right\},$$

where Q is the positive definite matrix given by (5.1.2). The solution ζ_* is immediate: we compute $\gamma=\sqrt{\xi^TQ\xi}$ and set

$$\zeta_*=\begin{cases}-\xi, & \gamma\le\rho\\ -\frac{\rho}{\gamma}\xi, & \gamma>\rho\end{cases}$$

Solving (5.1.16). Recalling that $B_p^+(1)=\{\lambda\in\mathbf{R}^\ell:\lambda\ge0,\|\lambda\|_p\le1\}$ and applying Lagrange duality, we observe that the optimal solution to (5.1.16) is

$$\lambda_*=\operatorname*{argmin}_{\lambda\in B_p^+(1)}\left\{\frac{1}{2}\lambda^T\lambda+(\xi+\theta_*y)^T\lambda\right\},$$

where

$$\theta_*=\operatorname*{argmax}_{\lambda\in B_p^+(1)}\phi(\theta),\ \phi(\theta)=\min_{\lambda\in B_p^+(1)}\left\{\frac{1}{2}\lambda^T\lambda+(\xi+\theta y)^T\lambda\right\}. \tag{5.1.17}$$

Since the only cases we are interested in are those of $p=2$ and $p=\infty$, the value and a supergradient of the (clearly concave) function $\phi(\cdot)$ are easily computable. For example, in the case of $p=\infty$, the minimizer $\lambda(\theta)$ of the function $\left\{\frac{1}{2}\lambda^T\lambda+(\xi+\theta y)^T\lambda\right\}$ over $\lambda\in B_p^+(1)$ is readily given by the relations

$$(\lambda(\theta))_i=\begin{cases}0, & \xi_i+\theta y_i\ge0\\ -(\xi_i+\theta y_i), & -1\le\xi_i+\theta y_i\le0\\ 1, & \xi_i+\theta y_i\le-1\end{cases};$$

with $\lambda(\theta)$ being computed, we immediately get $\phi(\theta)$ and

$$\phi'(\theta)=y^T\lambda(\theta).$$

We can find now the solution θ_* to the univariate concave optimization program (5.1.17) by Bisection, and thus get the solution $\lambda(\theta_*)$ to the problem of interest (5.1.16).

The algorithm

The proposed version of the MP algorithm follows the description of the Basic MP algorithm (Algorithm 3.1) up to two points: *incorporating "aggressive" stepsize policy* and *mechanism for building optimality gap.*

Aggressive stepsize policy. Recall that the only requirement on the stepsize $\gamma_t > 0$ and the point $w_t = w^{s_t-1}$ in the Mirror Prox algorithm is to satisfy the target inequality (3.1.19). Theorem 3.1 states that with the constant stepsize $\gamma_t = \gamma_{\rm safe} \equiv \frac{\kappa}{\sqrt{2}L}$, this inequality is satisfied after at most two inner iterations (3.1.18). At the same time, by Proposition 3.2 the accuracy t-th approximate solution $z^t = (\zeta^t, \lambda^t)$ measured in terms of our objective

$$\min_{\|\zeta\|\le\rho} F(\zeta), \quad F(\zeta) = \max_{\lambda\in B_p^+(1), \sum_i y_i\lambda_i=0} \left[\mathbf{1}^T\lambda - \lambda^T Q\zeta\right] \tag{5.1.18}$$

associated with the saddle point problem (5.1.1) is given by

$$F(\zeta^t) - \min_{\|\zeta\|\le\rho} F(\zeta) \le \frac{A}{\sum_{\tau=1}^t \gamma_t}, \tag{5.1.19}$$

where A is defined solely by the data and the setup and completely independent of the stepsize policy. From the latter bound, it follows that the (upper bound on the) accuracy after t steps is governed by the quantity $\Gamma_t = \sum_{\tau=1}^T \gamma_t$, while the "computational cost" of this accuracy is the total number I_t of inner iterations in course of t steps of the algorithm. Thus, the quantity $\Theta_t = \Gamma_t/I_t$ can be though of as the "performance to cost" ratio for our algorithm – the larger is Θ_t, the better. With Basic implementation, this ratio is, independently of t, the quantity $\Theta^{\rm safe} = \gamma_{\rm safe}/2$ (indeed, with this implementation, $\Gamma_t = t\gamma_{\rm safe}$, while I_t is (at most) $2t$. Note, however, that the performance-to-cost ratio $\Theta^{\rm safe}$ comes from the worst-case-oriented theoretical considerations and as such could perhaps be improved in practical computations. The aggressive stepsize policy, we use in our implementation of the MP algorithm, tries to improve the performance-to-cost ratio in the simplest possible fashion, specifically, as follows. We fix some "threshold" number k of inner iterations per step of the algorithm (in our implementation, this threshold is 3). When running inner iterations at step t, we start with certain (recursively defined, see below) initial value $\gamma_t^1 \ge \gamma_{\rm safe}$ of the stepsize and run the inner iterations

$$w^s = P_{z_{t-1}}(\gamma_t^s \Phi(w^{s-1})), s = 0, 1, ... \tag{5.1.20}$$

(cf. (3.1.18)) with $\gamma_t^1 = \gamma_t^2 = ... = \gamma_t^k$. If step t is terminated according to the rule

$$\begin{array}{l}\langle w^{s-1} - P_{z_{t-1}}(\gamma_t^s\Phi(w^{s-1})), \gamma_t^s\Phi(w^{s-1})\rangle \\ + [\omega(z_{t-1}) + \langle\omega'(z_{t-1}), w^s - z_{t-1}\rangle - \omega(w^s)] \le 0\end{array} \tag{5.1.21}$$

(cf. (3.1.19)) in course of the first k inner iterations, we set $\gamma_t = \gamma_t^s$ and start the step $t+1$ with increased value of the stepsize: $\gamma_{t+1}^1 = \gamma_+\gamma_t$, ($\gamma_+ > 1$ is a fixed factor). In the opposite case (step t does not terminate in course of the first k inner iterations), we start to reduce the current stepsizes by another fixed factor $\gamma_- < 1$ until the value $\gamma_{\rm safe}$ is reached, that is, we continue inner iterations (5.1.20) with

$$\gamma_t^{s+1} = \max[\gamma_-\gamma_t^s, \gamma_{\rm safe}]$$

until the termination condition (5.1.21) is met. We then set $\gamma_t = \gamma_t^s$ and start the step $t+1$ with $\gamma_{t+1}^1 = \gamma_t$.

In our implementation, we use $\gamma_+ = 1.2$ and $\gamma_- = 0.5$. Numerous experiments demonstrate that although the outlined stepsize policy usually requires, at average, more than 2 inner iterations per step of the algorithm, this increase in the computational price of a step is more than compensated by progress in the resulting average stepsize γ_t, so that the "performance-to-cost" ratios Θ_t usually are by order of magnitudes larger than the "benchmark" ratio $\Theta^{\rm safe}$.

Generating approximate solutions and optimality gaps. Observe that if $(\bar\zeta, \bar\lambda)$ is a feasible solution to (5.1.1) *and we have already computed the vectors* $Q\bar\zeta$ *and* $Q^T\bar\lambda$, then we can easily compute the quantities

$$\begin{aligned} F(\bar\zeta) &= \max_{\lambda\in B_p^+(1), \sum_i y_i\lambda_i = 0} \left[\mathbf{1}^T\lambda - \lambda^T Q\bar\zeta\right] \\ \underline{F}(\bar\lambda) &= \min_{\zeta:\|\zeta\|\le\rho} \left[\mathbf{1}^T\bar\lambda - \bar\lambda^T Q\zeta\right]. \end{aligned}$$

Note that $F(\bar\zeta)$ is the value at $\bar\zeta$ of the actual objective we are interested to minimize over $\zeta \in X = \{\zeta : \|\zeta\| \le \rho\}$; consequently, the approximate solution of the problem $\min_X F(\zeta)$ (we can build at certain moment) is the best (with the smallest value of F) of the ζ-components of all feasible pairs (ζ, λ) for which we have computed so far the vectors $Q\zeta$ and $Q^T\lambda$. We denote by F^t the value of F at this best found so far solution. Now, for $\bar\lambda \in Y = \{\lambda \in B_p^+(1), \sum_i y_i\lambda_i = 0\}$, the quantities $\underline{F}(\bar\lambda)$ are *lower* bounds on the optimal value F_* in the problem of interest:

$$F_* \equiv \min_{\zeta\in X}\max_{\lambda\in Y}\left[\mathbf{1}^T\lambda - \lambda^T Q\zeta\right] \ge \min_{\zeta\in X}\left[\mathbf{1}^T\bar\lambda - \bar\lambda^T Q\zeta\right] = \underline{F}(\bar\lambda).$$

It follows that we can build lower bounds on the optimal value in the problem as the largest of the quantities $\underline{F}(\lambda)$, $\lambda \in Y$, computed so far. Denoting by F_t the lower bound built after t steps, the t-th *optimality gap* is the quantity $\Delta_t = F^t - F_t$; the gap bounds from above the non-optimality, in terms of the objective F, of the best approximate solution to the problem $\min_X F$ built in course of steps $1, ..., t$.

The bottom line is as follows: In the Algorithm 3.1, an approximate solution ζ^t to the problem built after t steps is the ζ-component of the weighted average (3.1.20) of the search points $w_\tau = (\zeta_\tau, \lambda_\tau)$, $\tau \le t$; without additional computations, we even do not know what $F(\zeta^t)$ is. In actual computations it seems to be more reasonable to choose as ζ^t the best (with the smallest value of F) of the points $\zeta \in X$, where we have computed so far the values $F(\zeta)$ of the objective. The MP algorithm, as applied to (5.1.1), computes at every inner iteration the vector $\Phi(\zeta, \lambda)$ at a current search point (ζ, λ) of the saddle point problem, that is, computes $Q\zeta$ and $Q^T\lambda$, and we can use "for free" these matrix-vector products in order to build approximate solutions to the problem and the lower bounds on its optimal value. In contrast to this, in the Basic MP method we do *not* compute $\Phi(\cdot)$ at approximate saddle points (z^t, λ^t) given by (3.1.20), and thus do not compute the associated values of the objective F and of the lower bound $\underline{F}$. In our implementation we add these computations in order to include the approximate saddle points (ζ^t, λ^t) in the outlined scheme for building approximate solutions and optimality gaps. However, to reduce the total computational effort, we compute $\Phi(\zeta^t, \lambda^t)$ (and thus $F(\zeta^t)$ and $\underline{F}(\lambda^t)$) for a sub-sequence of values of t only (in our implementation - once per every 25 steps of the method).

##	Data	Attributes	Density	Training sample	Validation sample
1	arcene60_40	10000	0.5	$60 = 29 + 31$	$40 = 15 + 25$
2	dexter200_100	20000	0.005	$200 = 100 + 100$	$100 = 50 + 50$
3	dorothea600_200	100000	0.009	$600 = 58 + 542$	$200 = 20 + 180$
4	gisette4000_2000	5000	0.1	$4000 = 2038 + 1962$	$2000 = 962 + 1038$
5	madelon1500_500	500	1.0	$1500 = 769 + 731$	$500 = 231 + 269$
6	g5000x15000x05_2000	5000	0.04	$15000 = 7462 + 7538$	$2000 = 1039 + 961$
7	internet_ads2500_780	1558	0.01	$2500 = 364 + 2136$	$780 = 95 + 685$
8	lymphoma60_36	4026	1.00	$60 = 40 + 20$	$36 = 22 + 14$
9	colon40_22	2000	1.00	40=12+28	22=10+12
10	bupa245_100	6	1.0	$245 = 105 + 140$	$100 = 40 + 60$
11	cleveland_heart200_97	13	0.75	$200 = 93 + 107$	$97 = 44 + 53$
12	ionosphere251_100	34	0.9	$251 = 161 + 90$	$100 = 64 + 36$
13	mushroom6000_2124	22	0.2	$6000 = 3120 + 2880$	$2124 = 1088 + 1036$
14	musk1300_176	166	1.0	$300 = 141 + 159$	$176 = 66 + 110$
15	pima600_168	8	0.9	$600 = 214 + 386$	$168 = 54 + 114$
16	sonar108_100	60	1.0	$108 = 64 + 44$	$100 = 57 + 43$
17	wdbc400_169	30	1.0	$400 = 152 + 248$	$169 = 60 + 109$

Table 5.1: 17 data sets used in numerical experiments.

5.2 Experiments: data and methodology

The outlined version of the Mirror Prox algorithm was implemented in MATLAB and tested on a number of SVM data. In this section, we describe the data and our experimentation methodology.

5.2.1 The data

Basically all data used in our experiments were found in Internet; the only exception is the data set g5000x15000x05_2000 we have generated. We have no detailed knowledge of the origin of most of the data (which in any case is irrelevant in our primarily computational context). All we know is that the first five data sets were used in the competition in feature selection techniques which preceded the NIPS Workshop on Feature Extraction (Whistler British Colombia, Canada, 2003). The remaining data seemingly are real-life ones.

Information on the 17 data sets we have used is presented in Table 5.1. This information includes the dimension of the attribute vectors $\widetilde{x}_i$, *data density* – average fraction of nonzero entries in these vectors, cardinality ℓ of the training sample and the numbers ℓ_+, ℓ_- of positive (with $y_i = 1$) and negative ($y_i = -1$) examples in the sample (in Table 5.1, these numbers are presented as $\ell = \ell_+ + \ell_-$) and similar numbers for the *validation sample*. The latter is not used in the SVM model itself; its only purpose is to test the quality of the classifier yielded by this model. Note that in all data sets available for us, only training samples are present; we have split each of these samples in two parts to be used as the training and the validation samples in our experiments. The examples from the original sample were split into the two parts at

random, with chosen in advance cardinalities of the parts; this partitioning was never revised.

By reasons which will be explained in a moment, it is natural to divide the data in Table 5.1 into "large-scale data" 1 – 9 (the dimension of the attribute vectors ranging from 500 to 100,000) and "low-dimensional data" 10 – 17 (the attribute dimension ranging from 6 to 166).

5.2.2 Methodology of experimentation

The experimentation methodology we have used stems from the fact that *the goal of our research is to investigate the potential of advanced large-scale convex optimization techniques*, specifically, the Mirror Prox algorithms, *in the context of SVM models rather than to solve actual binary classification problems.* This goal motivates a number of important decisions we have made as far as experimentation is concerned, specifically

1. In our experimentation, *we focus solely on plain SVM models.*

   The reason is that for our purposes already results obtained on plain SVM models provide sufficient, although, perhaps, not 100% complete, understanding of the potential of the MP algorithm in the SVM applications; at the same time, working with plain models, we have no need in choosing an appropriate kernel, which by itself is a highly time-consuming and heavily data-dependent task; since we do not know the origin of the majority of our data sets, we are not in a position to carry this task out. It should be added that at the most large-scale data sets (## 1 – 6 in Table 5.1), to the best of our understanding, do not require kernels to be processed reasonably well; the same is true for a significant part of the remaining data sets.

   Another argument in favour of plain models in our context can be best of all explained on an example. Consider, e.g., the data `mushroom6000_2124` (# 13 in Table 5.1). All we need to process the corresponding plain SVM model is to store in RAM a 6000×22 data matrix Q with the attribute vector being the rows; the computational cost of an inner iteration in MP is, essentially, the cost of computing $Q\zeta$ and $Q^T\lambda$ for given ζ, λ. Thus, the required RAM is about 1 megabytes (Mb), assuming double precision (that is, 8 bytes per real), and the cost of an inner iteration is just about 500,000 floating point operations. Now assume that we intend to process the same data set via a kernel-type SVM model with, say, Gaussian kernel. The corresponding matrix Q given by (5.1.2) is a symmetric 6000×6000 matrix which is fully dense. It takes as much as 144 Mb of RAM to store the (lower triangular part of the) matrix, and as much as 72,000,000 of floating point operations to multiply this matrix by a vector – we get over than 100-fold increase in computational resources per inner iteration of MP. This dramatic growth in computational effort when passing from the plain to the kernel-generated SVM could perhaps be justified if our goal were to build as good classifier for `mushroom6000_2124` as possible, but it hardly could be justified with our actual goal.

2. We do not pay too much attention to tuning the parameters of the SVM models, for example, the parameter ρ in (5.1.1).

   If our only goal were to get good classifiers for the data sets, our primary effort would be on the "optimal" choice of ρ and two other model parameters r and p (these parameters are responsible for the choice of the norm in the constraint $\|\zeta\| \leq \rho$ and the norm in which the vector of slacks is measured). For example, we tune the model parameters by processing a series of problems (5.1.1) coming from the same data set and differing from

each other in model parameters only, to test the resulting classifiers on the validation sample and to apply a kind of rough optimization over model parameters aimed at specifying the parameters resulting in the smallest possible classification error at the validation sample. With our actual goal, that is, evaluating the potential of MP in the SVM context, such a "fine tuning" would be more or less waste of time: if MP performs well for a "representative", although small, set of different model parameters, there are all reasons to expect that it will perform equally well for other values of the parameters in their "represented" range.

3. When evaluating the performance of MP, we keep in mind the ultimate purpose of the computational process – building classifier instead of focusing on optimization aspects only.

   Methodologically, this point is very important when "computationally cheap" techniques for large-scale convex optimization are concerned. As it was explained in Introduction, these techniques are unable to guarantee high-accuracy solutions with reasonable iteration count; at the present level of our knowledge, the latter is possible only with polynomial time optimization techniques with prohibitively expensive, in the large-scale case, iterations. Thus, the only point in solving large-scale convex problems by computationally cheap methods stems from the desire "to buy reduced accuracy at reduced (and thus affordable) price", in hope that the "reduced accuracy" already is sufficient for our applications. Whether this hope is or is not justified, and thus whether it makes or does not make sense to use "cheap" methods, depends on the method *and* on the application. It follows that in our context, at least one of the major criteria for evaluating MP is *how rapidly the quality of the resulting classifiers stabilizes with the iteration count*. Note that this criterion has little in common, both with the quality of the classifier produced by the SVM model and with the rate of convergence of the MP, as applied to this model. Indeed, imagine that in the first few iterations MP produces a classifier with the classification error, as evaluated at the validation sample, as large as 15%, while a high-accuracy solution of the same SVM problem (obtained in much larger time by an interior point method or with many iterations of MP) produces a classifier with error 14%. By itself, classification error of 14% or 15% is "meaningful", nevertheless, in terms of the particular application, the method could be qualified as pretty good – just in few (cheap!) iterations, it produces a solution, which *in terms of the particular application we are interested in*, is almost as good as a much more computationally expensive high-accuracy solution.

5.2.3 Organization of experiments

Our implementation of the guidelines outlined in the previous Section is as follows:

1. With every one of the 17 data sets presented in Table 5.1, we associate several saddle point problems (5.1.1) corresponding to with $\|\zeta\| \equiv \|\zeta\|_r$. All these problems represent *plain* SVM models and differ from each other only in the values of the model parameters ρ, r, p. Specifically, we allow for the parameters to take, independently of each other, the values as follows:

   $$\rho: \ 0.1, 1.0, 10.0; \quad r: 1, 2; \quad p: 1, \infty,$$

   which gives us $12 = 3 \times 2 \times 2$ problems per data set.

Restricting the values of r and p with the endpoints of the respective theoretically allowed segments $1 \le r \le 2$ and $2 \le p \le \infty$ seems quite natural. Our preliminary experiments (which we do not report here) demonstrate that the chosen values 0.1, 1.0, 10.0 of ρ are "representative" for the majority of our data sets.

2. Recall that in the case when the domain X of ζ in (5.1.1) is $\|\cdot\|_1$-ball (that is, the model parameter r is 1), we have two alternative choices of the corresponding distance-generating function, namely, the function (5.1.12) with $r = 1$ and the function (5.1.13); in the sequel, we refer to these distance-generating functions as to the *power-like* and the *entropy-like*, respectively. Thus, 6 of the above 12 saddle point problems – those with $r = 1$ – can be processed by two different versions of MP each. As a result, every data set from Table 5.1 gives rise to $18 = 12 + 6$ numerical experiments, which amounts to total of $17 \cdot 18 = 306$ numerical experiments. In fact the number of numerical experiments was slightly larger – 316, since some of the data were processed with additional values of ρ.

3. A particular numerical experiment was run until the first occurrence of any one of the following two events:

   - Arriving at 1% "semi-relative accuracy": $\mathrm{Accur}(t) \le 0.01$, where
   $$\mathrm{Accur}(t) = \frac{\Delta_t}{\max[1, F(\zeta^t)]} \tag{5.2.1}$$
   Recall that ζ^t is the approximate solution built in course of t steps, $F(\zeta) = \max\limits_{\lambda \in B_p^+(1), \sum_i y_i \lambda_i = 0} \left[\mathbf{1}^T\lambda - \lambda^T Q \zeta\right]$ is the objective of interest, and Δ_t is the optimality gap built in course of t steps, which is an upper bound on the quantity $F(\zeta^t) - \min\limits_{\|\zeta\| \le \rho} F(\zeta)$
   - Executing 1000 steps of the MP algorithm.

4. Once per every 25 iterations and upon termination, the computational process was interrupted in order to test the classification errors of the "current classifier" (the one yielded by the best found so far solution to the problem of interest $\min\limits_{\|\zeta\| \le \rho} F(\zeta)$).

5. In course of every experiment, we have recorded the following quantities describing the computational process:

   (a) Progress in optimality gap
   $$\mathrm{PrG}(t) = \frac{\Delta_t}{\Delta_1} = \frac{F^t - F_t}{F^1 - F_1} \tag{5.2.2}$$
   (recall that $F^t = F(\zeta^t)$ and F_t are the best found in t steps upper, respectively, lower bound on the optimal value in the problem of interest $\min\limits_{\|\zeta\| \le \rho} F(\zeta)$);

   (b) Relative accuracy
   $$\mathrm{RelAccur}(t) = \frac{F^t - \mathrm{LwB}}{F^t + \text{1.e-12}}, \tag{5.2.3}$$
   where LwB is the best known lower bound on the optimal value in the problem of interest (in most of the cases, this is the lower bound F_T built by MP upon termination.

(c) Classification errors $\mathrm{ErrTr}(t)$, $\mathrm{ErrVl}(t)$ of the classifier built at iteration t on the training and the validation samples, respectively; these quantities are in the sequel referred to as *training* and *validation* errors, respectively.

(d) The total number $\mathrm{InnerItr}(t)$ of inner iterations in course of t steps; this is the natural "intrinsic" computational cost of t steps, not directly affected by the size of the data and computer's performance.

(e) CPU time $\mathrm{CPU}(t)$ required to run t steps, with time used to test the classifiers excluded; this is the actual (affected by the size of the data and computer's performance) cost of t steps of the algorithm.

6. Finally, SVM models with $r \in \{1, 2\}$ and $p \in \{1, \infty\}$ in their optimization form

$$\min_{\zeta:\|\zeta\|_r \le \rho, b\in\mathbf{R}, \xi} \{\|\xi\|_{p_*} : \xi_i \ge \max[0, 1 - ((Q\zeta)_i + y_i b)],\ i = 1, ..., \ell\}$$

clearly are just linear programs ($r = 1, p = \infty$), or linearly constrained convex quadratic programs ($r = 1, p = 2$), or convex quadratically constrained quadratic programs ($r = 2$); all these problems can be solved by Interior Point polynomial-time methods. In order to get a kind of benchmark for evaluating the MP algorithm, part of our experiments were accompanied by solving the problems by the state-of-the-art commercial IP solver `mosekopt`, which is known as nearly the best commercial solver for linear and linearly constrained convex quadratic programs, and the only commercial solver capable to solve quadratically constrained programs.

5.3 Experiments: numerical results

It would be unreasonable to present detailed data on every one of 316 numerical experiments we have carried out; even detailed data on 18 experiments related to a *single* data set would be too much. What we are about to do, is to present a number of "integral slices" of the all accumulated information and then detailed information on selected experiments.

5.3.1 Statistics of results

Overall performance

Data on the overall performance of the MP algorithm on our data are given in Table 5.2 and on Figures 5.1 – 5.2. In Table 5.2, we display the *averages* of various performance characteristics of the MP algorithm, while the histograms on Figures 5.1 – 5.2 give an impression on the distributions of the performance characteristics, which are most important in our context, that is,

• relative accuracy, which seems to be the major characteristic of the solution process considered in purely optimization perspective, and

• the resulting validation error (i.e., classification error on the validation sample) which is the major entity of interest in the SVM context.

A detailed explanation of what is presented in Table 5.2 is given in the footnotes accompanying the table; explanation of what is displayed at the figures is in their captions. There are, however, two specific points related to our tables and figures:

1. When displaying statistics of "time-dependent" performance characteristics (like progress in optimality gap, or current relative accuracy, or classification error of current classifier – all these quantities depend on the iteration number), one should take into account that strictly speaking, these entities exist in an experiment-dependent "time horizon"; indeed, what is relative accuracy at step 200, if the solution process has reached the required relative accuracy 0.01 and thus was terminated (see item 3 in Section 5.2.3) as early as at step 100? How should the results of such an experiment contribute to the distribution of relative accuracy at step 200? Discarding these results from the corresponding statistics would be misleading – it could give an impression that the distribution of accuracy reached by the method "spoils" as the number of steps grows, which definitely is not the case. Our remedy here is to think of all experiments as lasting for all our allowed 1000 steps of MP and to extend the performance characteristics, like progress in optimality gap, beyond their actual "life span" by the values they have reached at the end of this span.

   Another difficulty of the same flavour stems from the fact that the error, as evaluated on the validation sample (same as on the training one), of the classifier built in course of t steps not necessarily decreases as t grows; in many experiments, this error first decreases and then grows or oscillates (this phenomenon could be thought of as "over-tuning"). Needless to say, in actual applications the classifier to be forwarded to the end-user should be the best, as evaluated on the validation sample, of the classifiers built in course of the solution process, rather than the classifier associated with the best solution to the optimization model we are processing. To capture this point in our statistics, we define the classification error obtained during t steps as the error of the *best* classifier we have built and tested so far (which is not the same as the error of the *last* classifier we have tested so far).

2. Another specific remark has to do with Figure 5.1 (and all subsequent figures representing relative accuracy). In the SVM models it is possible to reach the *exact* optimal value in a finite number of steps. The latter is what should be expected in the case when an "ideal" affine classification is possible (that is, a classification resulting in the zero slack vector). In these cases, ideal classification is given by a "massive" (with a nonempty interior) set of affine classifiers, and it may happen (and indeed it happened in our experiments) that our iterative process reaches this optimal set in finite number of steps. As a result, in part of our experiments the optimality gap reaches zero value, which makes it impossible to display the relative accuracy on a histogram with logarithmic scale along the accuracy axis. In order to avoid this difficulty, when drawing histograms of the progress in optimality gap, we replaced all values of RelAccur, which are $<$ 1.e-10 with 1.e-10, which results in "high bins" at the very left of the corresponding histograms; in fact, these bins represent the percentage of problems solved to optimality.

Comments on the performance characteristics of MP presented in Table 5.2 and on Figures 5.1 – 5.2 are as follows:

1. *Performance of MP as an optimization method* seems to be quite reasonable:

   - At average, the relative accuracy achieved by the method is about 0.09 (see Tab. 5.2) vs. our target accuracy 0.01, which is not that attractive, the target accuracy was achieved in over 80% of the experiments (Fig. 5.1.A.3), with the average number of

iterations about 341[2)] (Tab. 5.2), nearly 3 times less than our allowed maximum of 1000 iterations, and average CPU time as small as 73.5 sec[3)]. Moreover, in about 22% of the experiments the optimization problems were solved to optimality (Fig. 5.1.A.3).

- Good news are that the performance characteristics of MP on large-scale data ## 1 – 9 are significantly better than on all the data: while the percentage of experiments where the target accuracy was achieved is about the same 80%, now the optimality was reached in as much as 40% of the experiments. Moreover, in 22% of the experiments, optimality was reached in less than 50 iterations, and in 38% of them – in less than 200 iterations, see Fig. 5.1.B. The average iteration count on large-scale data was 264 (cf. the average 341 over all data), with average CPU time of 130 sec.

2. As far as *the quality of classifiers* yielded by the MP algorithm, the situation seems to be really nice:

   - At average, the classification error of the resulting classifiers, as evaluated on the validation samples, is about 19% (all data) and about 18% (large-scale data), see Table 5.2. In over 35% of experiments with all data and in over 47% of experiments with large-scale data, the errors of the resulting classifiers were less than 10%, see Fig. 5.2. By itself, these results primarily reflect the properties of the data sets and do not say much about the performance of the MP algorithm in our application. What *does* say a lot of this performance – and what are really good news about it – is that the best classification error achieved in course of an experiment is, typically, achieved pretty fast. Indeed, data in Table 5.2 indicate that
     - at average, both for all data and for the large-scale data, the classification error achieved in just 50 steps of an experiment, is pretty close to the error achieved during the entire experiment;
     - the average number of steps resulting in the best classifier is as small as 131.5 for all data, and 105.2 for large-scale data, the corresponding average CPU times being just 21.1, respectively, 35.6 sec;
     - the average number of steps before a "nearly the best" classifier is built (that is, with the error within the factor 1.1 of the best error achieved in the experiment) is just 100.4 for all data and 87.0 for large-scale data.

     This phenomenon – "what can be achieved, can be achieved fast" – is also clearly seen on Fig. 5.2 – the percentage of experiments where a "good" (less than 10%) classification error is achieved in just 50 iterations is pretty close to the percentage of experiments where this error is achieved during the entire run.

3. A nice feature of the classifiers produced by the MP algorithm is their *sparsity* – the number of features (in our plain SVM models, these are exactly the same as attributes) used in a classifier is, at average over all experiments and over experiments with large-scale data ##

[2)] in the sequel, in order to save space, we do not present the data on the total number of inner iterations required by experiments, only the data on the number of steps (in the sequel called "iterations"). The reason is that with our aggressive stepsize policy, both the quantities – the total number of inner iterations and the total number of steps – are proportional to each other with nearly independent of an experiment coefficient about 4.5.

[3)] All computations were carried out on IBM ThinkPad PC with Intel Pentium 1.86 GHz processor and 1 Gb of RAM.

1 – 9, just about 11%, respectively, 8%, of the total number of attributes. It follows that the MP-generated classifiers, at least with our data sets, allow for nice *feature selection* – selecting relatively small number of features/attributes responsible for classification. This selection, important by its own right, could be used to reduce computational expenses in computing classifiers: after a classifier which uses a relatively small number of features and possesses a reasonable classification error is built (which typically happens essentially earlier than the computational process terminates), we could discard all attributes/features not used by the classifier and to resolve the problem on the smaller, in terms of the number of attributes, and thus easier to process computationally, training sample.

"Common wisdom" says that sparse classifiers usually are yielded by SVM models with $\|\cdot\|_1$-norm in the role of $\|\cdot\|$ in (5.1.1). At least with our data sets, this is not the case: it is seen from Tables 5.3 – 5.4 that $\|\cdot\|_2$ in the role of $\|\cdot\|$ leads to equally sparse classifiers.

When solving an optimization problem by an iterative method, approximate solutions usually are not sparse, even when the optimal solution is so; typically, entries in approximate solutions corresponding to zero entries in the optimal one are pretty small, but not exactly zero. To account for this phenomenon, in our experiments we subject the classifiers produced by the algorithm to a kind of "purification". Specifically, at step t, in order to build the classifier associated with the best found so far solution ζ^t, we replace with zeros all "negligible" entries ζ_i^t in ζ^t – those satisfying the relation $|\zeta_i^t| \leq 10^{-4} \max_j |\zeta_j^t|$, while keeping the remaining entries intact. The data on sparsity and classification errors we present here relate to the "purified" classifiers.

Performance on various SVM models

Recall that in our experiments we deal with four types of plain SVM models differing from each other in the norms we use to quantify the magnitudes of a linear form on the attribute≡feature space and of a vector of slack variables. Specifically, we work with $\|\cdot\|_r$-norm in the role of $\|\cdot\|$ in (5.1.1) and the norm $\|\cdot\|_{\frac{p}{p-1}}$ to measure the magnitude of a vector of slacks, allowing for the following four combinations of the values of r, p:

- $r = 1$, $p = \infty$ (this corresponds to the LP SVMs);
- $r = 1$, $p = 2$;
- $r = 2$, $p = \infty$;
- $r = 2$, $p = 2$.

The performance characteristics of MP on every one of the four SVM models in question is presented in Tables 5.3 – 5.4 and on Figures 5.3 – 5.4.

Comments to the data in Tables 5.3 – 5.4 and on Figures 5.3 – 5.4 are as follows.

1. As far as "purely optimization" performance characteristics of the MP algorithm are concerned, *at average* all of them are more or less the same for all four SVM models, and it is difficult to select the "clearly winning" model. Say, the SVM models with $r = 1$ seem to result in better relative accuracy and in worse progress in optimality gap than models with $r = 2$, see Tables 5.3 – 5.4; note that from purely optimization viewpoint, both characteristics seem to be equally important... Note, however, that the LP SVM models ($r = 1$, $p = \infty$) require more time-consuming processing. For example, for these models, average, over all experiments, iteration count and CPU time are 382 iterations and 122

Subset of data		$t=25$	$t=50$	$t=100$	$t=200$	$t=400$	$t=800$	$t=1000$
Data ## 1 – 17[i)] (316 exper.)	Itr_fin[a)]	Progress in optimality gap PrG(t)[b)]						
	340.8	0.377	0.270	0.193	0.146	0.108	0.077	0.071
	CPU[c)]	Relative accuracy RelAccur(t)[d)]						
	73.50	0.332	0.270	0.186	0.145	0.128	0.094	0.092
	Itr_bst[e)]	Classification error, training sample						
	131.5	0.150	0.127	0.117	0.112	0.108	0.106	0.106
	CPU_bst[f)]	Classification error, validation sample						
	21.1	0.228	0.211	0.199	0.194	0.192	0.191	0.191
	Itr_N_bst[g)]	Density of classifier[h)]						
	100.4	0.153	0.130	0.120	0.115	0.111	0.109	0.109
Data ## 1 – 9[i)] (164 exper.)	Itr_fin[a)]	Progress in optimality gap PrG(t)[b)]						
	264.2	0.353	0.222	0.150	0.121	0.099	0.074	0.068
	CPU[c)]	Relative accuracy RelAccur(t)[d)]						
	129.35	0.417	0.326	0.186	0.128	0.113	0.098	0.096
	Itr_bst[e)]	Classification error, training sample						
	105.2	0.112	0.083	0.073	0.070	0.067	0.065	0.065
	CPU_bst[f)]	Classification error, validation sample						
	35.6	0.203	0.188	0.180	0.180	0.179	0.178	0.178
	Itr_N_bst[g)]	Density of classifier[h)]						
	87.0	0.117	0.089	0.079	0.076	0.073	0.071	0.070

The data in the table are *averages*, across all experiments, of the performance characteristics as follows:

[a)]: number of steps of MP before termination, see item 3 in Section 5.2.3

[b)]: see (5.2.2)

[c)]: elapsed CPU time (sec), see item 5e in Section 5.2.3

[d)]: see (5.2.2)

[e)]: step of MP resulting in the best found in the experiment classification error *on the validation sample*

[f)]: CPU time (sec) required by the first Itr_bst[e)] steps

[g)]: the first step of MP where the classification error on the validation sample is at most 1.1 times larger than the best registered in the experiment

[h)]: the ratio of nonzero entries in the classifier to the dimension of the attribute vectors

[i)]: see Table 5.1

Table 5.2: Average performance characteristics of Mirror Prox algorithm

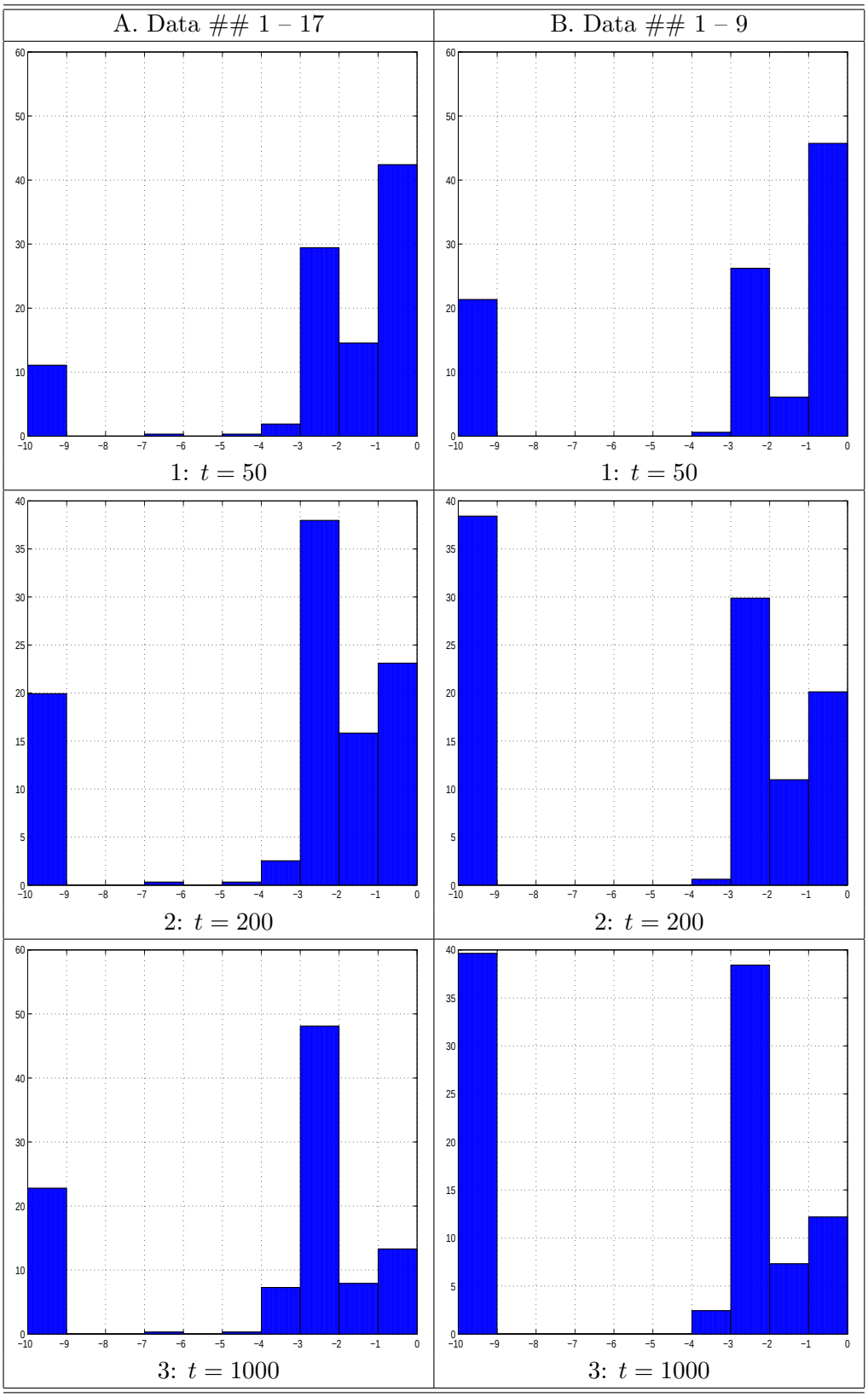

Figure 5.1: Distribution of relative accuracy RelAccur(t), see (5.2.3). Along the X-axis: decimal log of RelAccur(t); along the Y-axis: percentage of experiments in the bin.

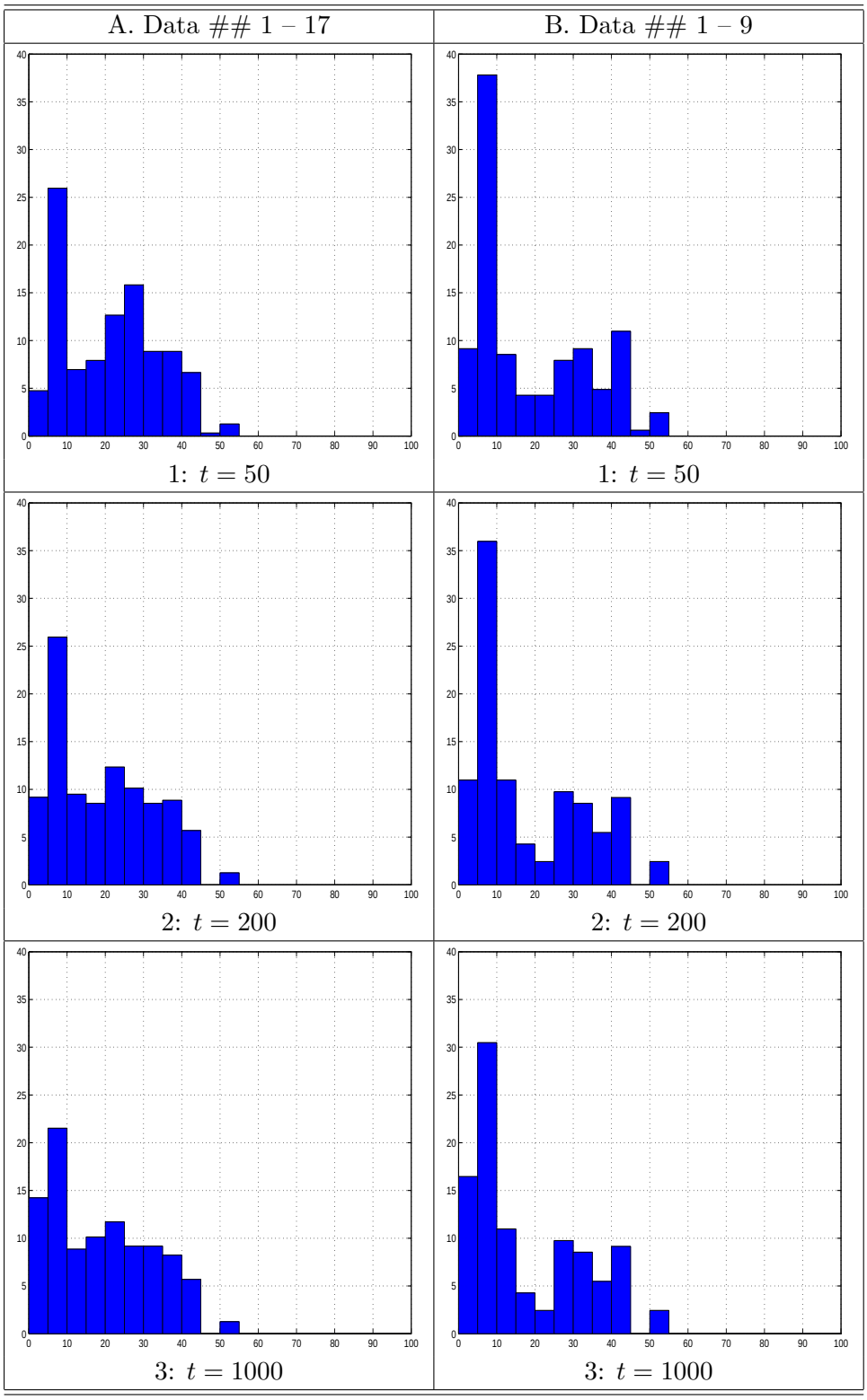

Figure 5.2: Distribution of validation errors ErrVl(t) (in %%), see item 5c in Section 5.2.3. Along the X-axis: values of error in percent. Along the Y-axis: percentage of experiments in the bin.

sec vs. 321 iterations and 31 sec for models with $r = p = 2$ (Table 5.3); the same averages for large-scale data are 336 iterations and 219 sec for LP models vs. 208 iterations and 50 sec for models with $r = p = 2$ (Table 5.4).

When passing from average performance characteristics to their distributions, the "winning" SVM model seem to be those with $r = 2$. For example, for both the models $r = 2, p = \infty$ and $r = p = 2$, the percentage of experiments where the problems were solved to optimality is over 30% for all experiments and over 50% (!) for experiments with large-scale data; the corresponding figures for SVM models with $r = 1$ are about 20% and 35%, respectively.

> Note that when passing from $p = 2$ to $p = \infty$, the theoretical efficiency bound of MP "spoils" by factor $\sqrt{\ell}$, ℓ being the cardinality of the training sample, see Chapter 3, and this phenomenon is independent of the value of r. While the data in Tables 5.3 – 5.4 demonstrate that passing from $r = 1$, $p = \infty$ to $r = 1$, $p = 2$ indeed slows down the optimization process (although by factor much less than predicted by theory), there is seemingly no slowing down when passing from $r = 2, p = \infty$ to $r = 2, p = 2$.

The bottom line is: from four SVM models in question, those better suited to processing via the MP algorithm are models with the Euclidean norm $\|\cdot\|_2$ of ζ.

2. The conclusion we have just made is in full accordance with the data on classification errors, as evaluated on validation samples, of the classifiers yielded by the four SVM models in question. While the average, over all data, classification errors associated with the four models in question are nearly the same (Table 5.3), and the average, over large-scale data, classification errors coming from models with $r = 2$ are only marginally better than those coming from models with $r = 1$, the data on distributions of these errors definitely are in favour of the former models. Indeed, from Fig. 5.4 we conclude that the models with $r = 1$ result in "good" (less than 10%) validation errors in 35% of all experiments and 45% of experiments with large-scale data; for models with $r = 2$, these figures improve to over 40% and over 50%.

All in all, the data in Tables 5.3 – 5.4 and on Figures 5.3 – 5.4 witness in favour of SVM models with $r = 2$, and among these models – slightly in favour of those with $p = 2$.

Performance with various setups

Recall that in SVM models with $\|\cdot\|_1$-norm in the role of $\|\cdot\|$ in (5.1.1), we have two alternative choices of the distance-generating function for the domain $\{\zeta : \|\zeta\|_1 \leq \rho\}$ of the ζ-component: the entropy-like function (5.1.13) and the power-like function (5.1.12). A natural research question is which one of these alternatives is more attractive. The corresponding data are given in Tables 5.5 – 5.6 and on Fig. 5.6. These data suggest that there are no essential difference in performance of the two versions of MP in question; the power-like distance-generating function yields a slightly more time-consuming and slightly more accurate, in the optimization perspective, algorithm. At the same time, both versions of MP are the same as far as the quality of the resulting classifiers is concerned (the latter could be predicted in advance – in both cases, the classifiers come from approximate solutions to the same saddle point problem).

SVM model		$t = 25$	$t = 50$	$t = 100$	$t = 200$	$t = 400$	$t = 800$	$t = 1000$
	Itr_fin[a)]	Progress in optimality gap PrG(t)[b)]						
$r = 1$, $p = \infty$	382.7	0.394	0.261	0.185	0.142	0.110	0.081	0.076
$r = 1$, $p = 2$	311.0	0.447	0.323	0.217	0.159	0.123	0.086	0.080
$r = 2$, $p = \infty$	332.3	0.246	0.196	0.149	0.112	0.077	0.053	0.045
$r = 2$, $p = 2$	320.7	0.329	0.256	0.207	0.158	0.107	0.072	0.067
	CPU[c)]	Relative accuracy RelAccur(t)[d)]						
$r = 1$, $p = \infty$	121.68	0.354	0.298	0.188	0.129	0.111	0.069	0.064
$r = 1$, $p = 2$	62.05	0.314	0.263	0.170	0.132	0.115	0.103	0.102
$r = 2$, $p = \infty$	37.51	0.345	0.262	0.196	0.179	0.163	0.112	0.110
$r = 2$, $p = 2$	30.58	0.309	0.231	0.201	0.171	0.158	0.112	0.112
	Itr_bst[e)]	Classification error, training sample						
$r = 1$, $p = \infty$	159.4	0.167	0.137	0.126	0.121	0.117	0.114	0.114
$r = 1$, $p = 2$	111.6	0.163	0.134	0.121	0.116	0.112	0.111	0.111
$r = 2$, $p = \infty$	122.4	0.119	0.110	0.103	0.099	0.095	0.093	0.092
$r = 2$, $p = 2$	121.9	0.120	0.110	0.104	0.098	0.094	0.093	0.093
	CPU_bst[f)]	Classification error, validation sample						
$r = 1$, $p = \infty$	28.4	0.231	0.210	0.198	0.193	0.191	0.190	0.190
$r = 1$, $p = 2$	21.0	0.239	0.221	0.205	0.200	0.199	0.198	0.198
$r = 2$, $p = \infty$	14.1	0.216	0.207	0.199	0.193	0.191	0.189	0.189
$r = 2$, $p = 2$	12.9	0.210	0.200	0.190	0.186	0.183	0.183	0.183
	Itr_N_bst[g)]	Density of classifier[h)]						
$r = 1$, $p = \infty$	114.9	0.166	0.136	0.125	0.120	0.116	0.114	0.114
$r = 1$, $p = 2$	89.9	0.162	0.134	0.120	0.116	0.112	0.111	0.111
$r = 2$, $p = \infty$	104.2	0.135	0.126	0.120	0.116	0.112	0.110	0.109
$r = 2$, $p = 2$	87.4	0.137	0.127	0.121	0.115	0.111	0.111	0.110

For footnotes, see Table 5.2.

Table 5.3: Average performance of MP for various SVM models, all data.

SVM model		$t = 25$	$t = 50$	$t = 100$	$t = 200$	$t = 400$	$t = 800$	$t = 1000$
	Itr_fin[a)]	Progress in optimality gap PrG(t)[b)]						
$r = 1$, $p = \infty$	335.5	0.411	0.238	0.166	0.136	0.116	0.086	0.080
$r = 1$, $p = 2$	246.0	0.442	0.281	0.163	0.126	0.109	0.082	0.075
$r = 2$, $p = \infty$	204.4	0.170	0.125	0.105	0.086	0.063	0.045	0.040
$r = 2$, $p = 2$	207.8	0.231	0.163	0.137	0.113	0.077	0.062	0.058
	CPU[c)]	Relative accuracy RelAccur(t)[d)]						
$r = 1$, $p = \infty$	219.36	0.468	0.402	0.218	0.135	0.122	0.088	0.084
$r = 1$, $p = 2$	109.66	0.413	0.337	0.173	0.110	0.085	0.082	0.080
$r = 2$, $p = \infty$	55.17	0.380	0.242	0.150	0.142	0.133	0.124	0.122
$r = 2$, $p = 2$	49.97	0.353	0.223	0.180	0.136	0.130	0.126	0.125
	Itr_bst[e)]	Classification error, training sample						
$r = 1$, $p = \infty$	130.0	0.139	0.098	0.086	0.084	0.080	0.077	0.077
$r = 1$, $p = 2$	92.1	0.129	0.092	0.077	0.074	0.071	0.070	0.070
$r = 2$, $p = \infty$	84.4	0.064	0.059	0.055	0.053	0.049	0.048	0.047
$r = 2$, $p = 2$	99.7	0.064	0.058	0.055	0.052	0.048	0.047	0.046
	CPU_bst[f)]	Classification error, validation sample						
$r = 1$, $p = \infty$	48.1	0.209	0.184	0.175	0.174	0.173	0.172	0.172
$r = 1$, $p = 2$	36.4	0.219	0.200	0.188	0.188	0.187	0.186	0.186
$r = 2$, $p = \infty$	21.3	0.176	0.173	0.172	0.171	0.170	0.170	0.169
$r = 2$, $p = 2$	21.3	0.176	0.173	0.172	0.171	0.170	0.170	0.169
	Itr_N_bst[g)]	Density of classifier[h)]						
$r = 1$, $p = \infty$	103.0	0.138	0.097	0.086	0.083	0.080	0.077	0.077
$r = 1$, $p = 2$	81.5	0.128	0.092	0.077	0.074	0.071	0.070	0.070
$r = 2$, $p = \infty$	73.5	0.098	0.093	0.089	0.087	0.083	0.082	0.081
$r = 2$, $p = 2$	77.3	0.098	0.093	0.090	0.087	0.083	0.082	0.081

For footnotes, see Table 5.2.

Table 5.4: Average performance of MP on various SVM models, large-scale data ## 1 – 9.

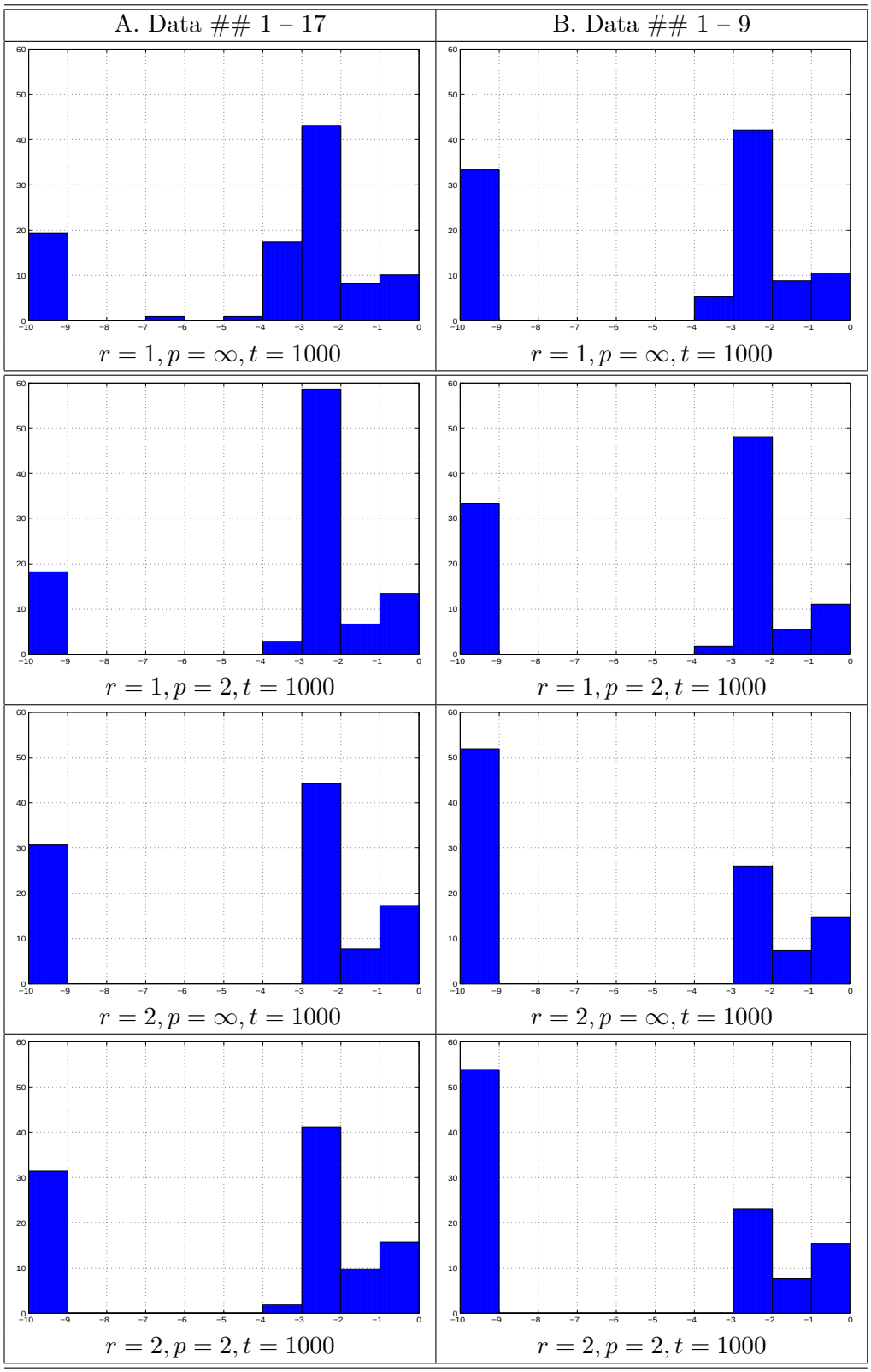

Figure 5.3: Distribution of relative accuracy RelAccur(t), see (5.2.3), for various SVM models. Along the X-axis: decimal log of RelAccur(t); along the Y-axis: percentage of experiments in the bin.

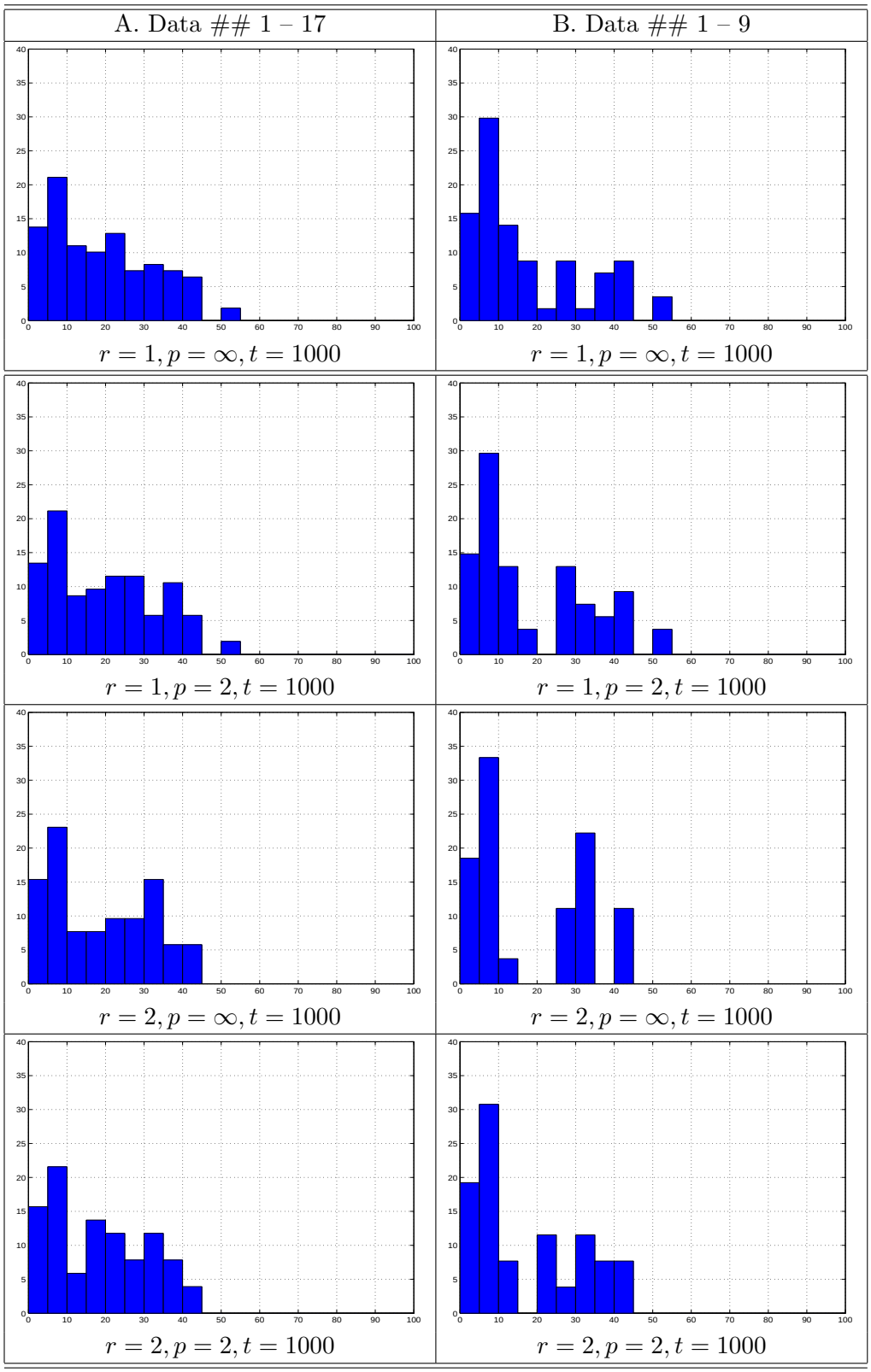

Figure 5.4: Distribution of validation errors ErrVl(t) (in %%), see item 5c in Section 5.2.3, for various SVM models. Along the X-axis: values of error in percent. Along the Y-axis: percentage of experiments in the bin.

SetUp		$t = 25$	$t = 50$	$t = 100$	$t = 200$	$t = 400$	$t = 800$	$t = 1000$
All data								
	Itr_fin[a)]	Progress in optimality gap PrG(t)[b)]						
entropy-like	335.4	0.366	0.262	0.196	0.154	0.121	0.089	0.084
power-like	359.6	0.473	0.319	0.206	0.148	0.112	0.078	0.072
	CPU[c)]	Relative accuracy RelAccur(t)[d)]						
entropy-like	73.47	0.322	0.249	0.166	0.131	0.120	0.104	0.100
power-like	111.13	0.348	0.311	0.193	0.129	0.106	0.068	0.066
	Itr_bst[e)]	Classification error, training sample						
entropy-like	132.9	0.140	0.127	0.120	0.115	0.113	0.111	0.111
power-like	139.1	0.189	0.144	0.126	0.122	0.117	0.114	0.114
	CPU_bst[f)]	Classification error, validation sample						
entropy-like	21.9	0.222	0.210	0.200	0.195	0.194	0.193	0.193
power-like	27.5	0.248	0.220	0.202	0.198	0.196	0.195	0.194
	Itr_N_bst[g)]	Density of classifier[h)]						
entropy-like	88.8	0.148	0.135	0.129	0.124	0.121	0.120	0.119
power-like	116.1	0.188	0.143	0.125	0.121	0.116	0.114	0.114
Large-scale data ## 1 – 9								
	Itr_fin[a)]	Progress in optimality gap PrG(t)[b)]						
entropy-like	253.1	0.344	0.203	0.143	0.122	0.103	0.076	0.072
power-like	328.8	0.505	0.312	0.185	0.140	0.121	0.091	0.083
	CPU[c)]	Relative accuracy RelAccur(t)[d)]						
entropy-like	130.77	0.423	0.306	0.162	0.118	0.109	0.102	0.099
power-like	199.36	0.458	0.431	0.228	0.127	0.099	0.069	0.066
	Itr_bst[e)]	Classification error, training sample						
entropy-like	95.4	0.095	0.084	0.078	0.076	0.073	0.071	0.071
power-like	126.8	0.171	0.106	0.086	0.082	0.079	0.076	0.076
	CPU_bst[f)]	Classification error, validation sample						
entropy-like	36.9	0.191	0.184	0.181	0.180	0.180	0.179	0.179
power-like	47.6	0.235	0.200	0.182	0.181	0.180	0.179	0.179
	Itr_N_bst[g)]	Density of classifier[h)]						
entropy-like	79.1	0.112	0.100	0.094	0.092	0.089	0.088	0.088
power-like	105.3	0.169	0.105	0.086	0.082	0.078	0.076	0.075

For footnotes, see Table 5.2.

Table 5.5: Average performance characteristics of MP with different setups on SVM models with $r = 1$.

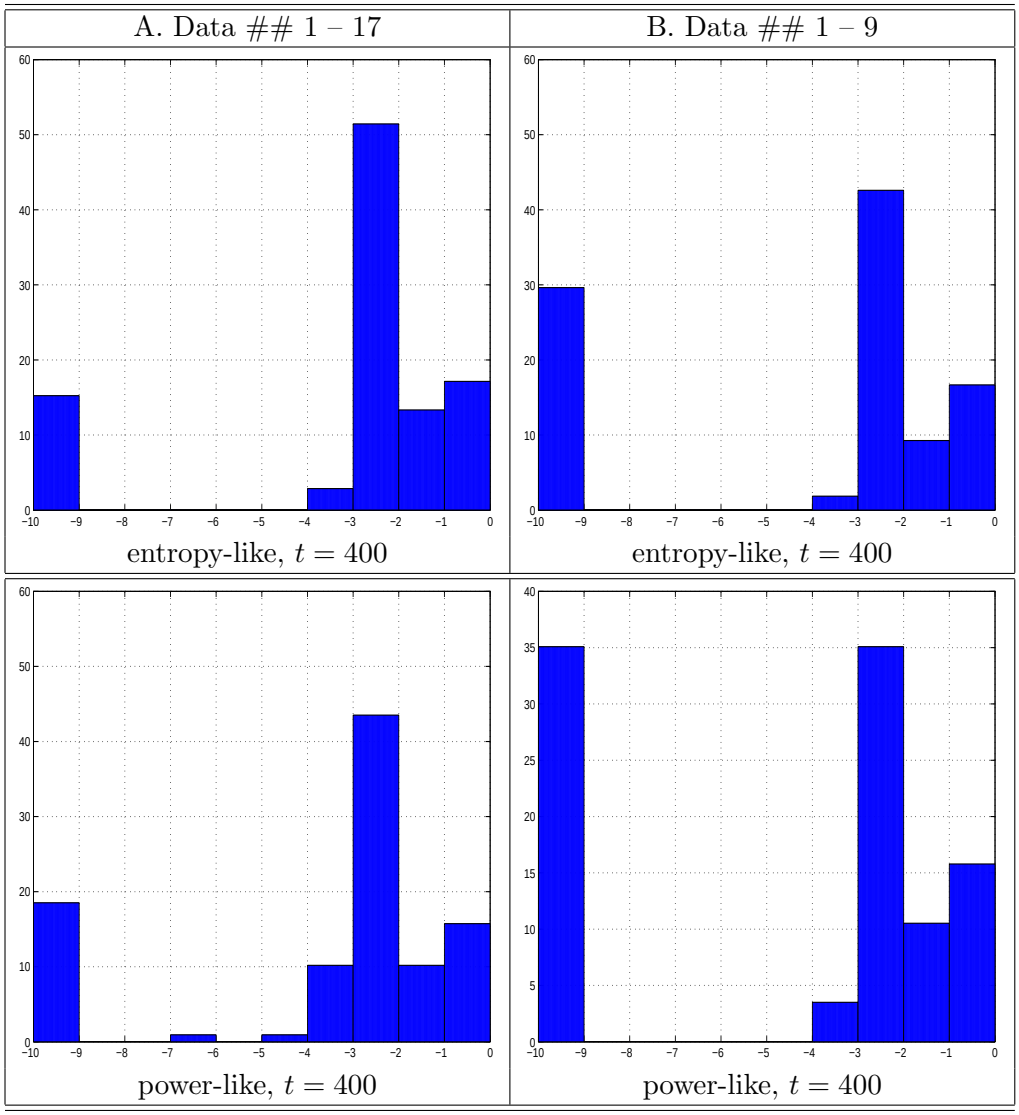

Figure 5.5: Distribution of relative accuracy RelAccur(t), see (5.2.3), for various MP setups on SVM models with $r = 1$. Along the X-axis: decimal log of RelAccur(t); along the Y-axis: percentage of experiments in the bin.

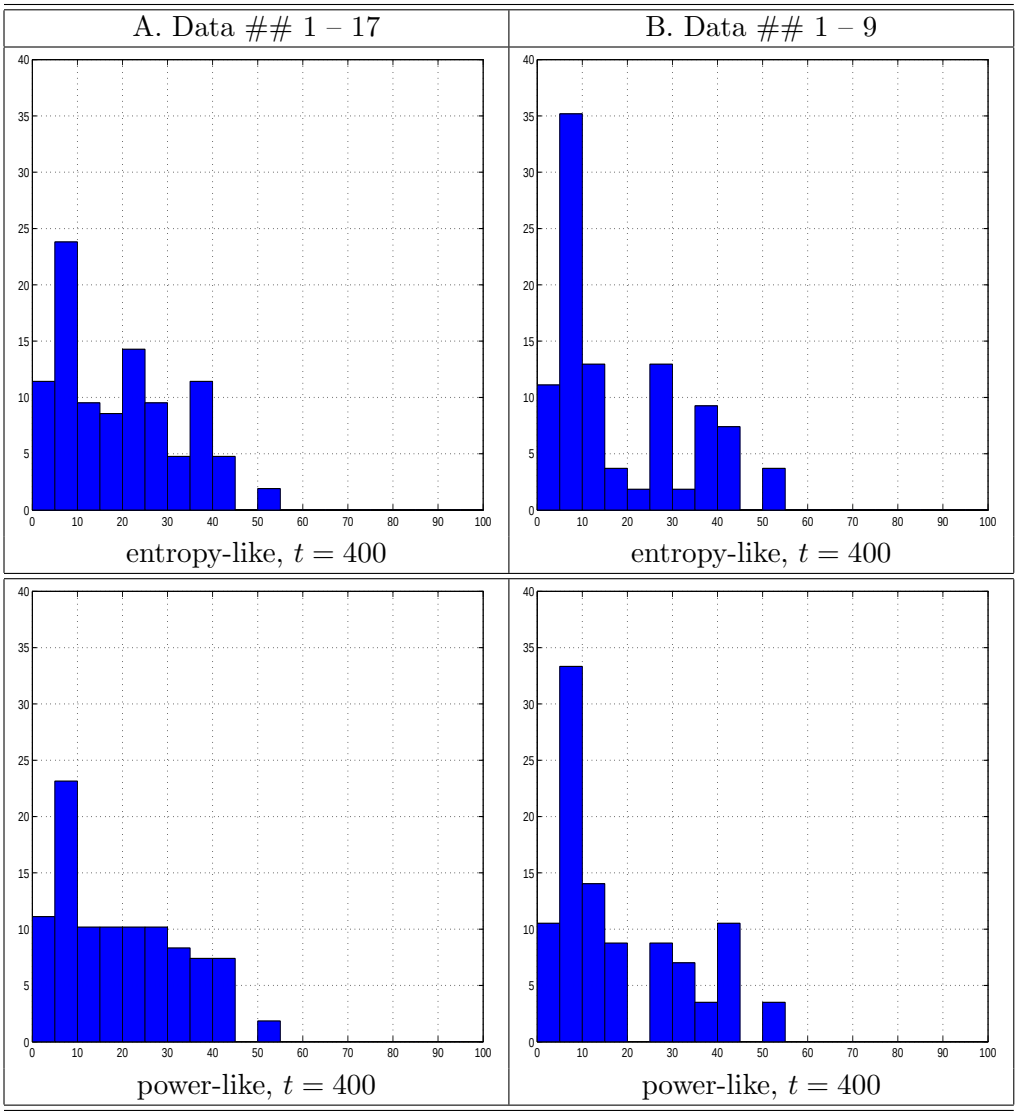

Figure 5.6: Distribution of validation errors ErrVl(t) (in %%), see item 5c in Section 5.2.3, for various MP setups on SVM models with $r = 1$. Along the X-axis: values of error in percent. Along the Y-axis: percentage of experiments in the bin.

5.3.2 Results for selected experiments

Tables 5.6 – 5.7 represent the best, in terms of the resulting validation error, results obtained on various data sets. Specifically, from all experiments with a particular data set from Table 5.1, we select the one which results in the best validation error, and present detailed information on this experiment. For description of the quantities displayed in the Tables, see footnotes to Table 5.6.

The data in Tables 5.6 – 5.7 are in full accordance with the observations we have made so far (the best classifier built in a computational process is built well before the termination of the process, the SVM model with $r = p = 2$ appears to outperform other models, etc.). An important new information presented in Tables 5.6 – 5.7 is the one on comparison of the MP algorithm with the state-of-the-art Interior Point commercial solver `mosekopt` (data in brackets [...] in Tables 5.6 – 5.7). The results of this comparison can be summarized as follows.

1. As it could be expected, on successful termination, `mosekopt` produces much more accurate solutions to the optimization problems in question than the MP algorithm (the `mosekopt` optimal values, listed in the lower parts of "LwB_fin$^{o)}$"-rows in Tables 5.6 – 5.7, are accurate within all digits presented in the tables). However:

   - More accurate IP solutions typically result in *worse* classifiers than less accurate MP solutions. Indeed, there is *a single* data set (data # 16) where the IP classifier is better than the MP one, and in this case the progress in quality is just marginal (24% validation error for the IP classifier vs. 25% error for the MP one). In contrast, *there are many cases where the MP classifier is much better than the IP one* (validation errors 20.0% vs. 27.5% for data #1, 7.5% vs. 17.5% for data # 3, 38.4% vs. 47.4% for data # 5, 9.1% vs. 18.1% for data # 9, 7.2% vs. 14.1% for data # 11...).

2. As it could be expected, on low- and medium-scale data ## 10 – 17 running times of the IP solver are negligible and significantly less than those of the MP algorithm; this fact, however, is of minor importance, since *all* the times in question are pretty small (the *maximal* running time of the MP algorithm here is just about $62'$). In contrast to this, *on large-scale data, IP is much more time-consuming than MP*. Indeed,

   - The *maximal* time of building the best MP-classifier on the data ## 1 – 9 is $238.5'$; the corresponding quantity for IP solver is as large as $2963'$, which is *5.3 times the total CPU time spent by MP to build the best classifiers on all our 9 large-scale data sets* (!).
   - In large-scale experiments MP was essentially more reliable than IP; the latter just failed to process data # 6 (MATLAB "Out of Memory" abnormal termination) and has severe difficulties with data ## 5, 9 ("Unknown" status of solution as reported by `mosekopt`[4)]).
   - Last, but not least, note that straightforward comparison of CPU times is by itself biased in favour of the IP solver. Indeed, the latter is an optimized `.mex`-executable, while MP is implemented via plain MATLAB scripts. Although the dominating, in

[4)] Such an abnormal termination means that `mosekopt` is unable to meet its built-in criteria for reaching optimality. As a result, the solution found by the solver may be pretty far from optimality, as is the case with data # 9: here the optimal value as reported by the IP solver is 11.666, while the true optimal value, successfully found by MP, is 0.000

the large-scale case, computations in MP (that is, the matrix-vector multiplications) use highly optimized built-in MATLAB functions, about 30% of computations (e.g., computing the prox-mappings) could be highly accelerated with a C-implementation.

Early termination. We just have made two claims:

I. *The classifiers built by the MP algorithm are essentially better than the classifiers yielded by the Interior Point algorithm;*

II. *The MP-based classifiers are significantly cheaper computationally than the IP-based ones.*

While the first of these claims is fully supported by the data in Tables 5.6 – 5.7, the second claim, strictly speaking, does *not* follow directly from these data. Indeed, recall that in course of running the MP, once per every 25 iterations and upon termination we build the classifier associated with the best found so far solution to the optimization problem and compute the validation error of this classifier. Note that from the purely optimization perspective, the quality of the best found so far solution can only improve with time; in contrast to this, the associated validation error may oscillate with time, so that a "late" classifier can be worse than an "early" one. Clearly, a current classifier can be easily compared, in terms of its quality, with already built ones, but not with the classifiers to be built in the future. It follows that the best classifier, independently of how early it is actually built, can be identified only in retrospective, upon termination of the solution process. For example, Table 5.6 says that on data # 3, the best classifier was built after 50 iterations, which took just 109.3 sec; however, this fact can be established only in retrospective, after the entire computation, which took as many as 1000 iterations (total CPU time 2767 sec). The question is, whether we can somehow reduce the "wasted" running time – the one which does not improve classifier's quality. The simplest policy here could be to terminate the computations at the first step (among those where the classifiers are built and tested) where the current classifier turns out to be worse in terms of quality than its predecessor. Table 5.8 illustrates the performance of this policy. In this table,

• column A1 contains the best validation errors achieved with MP in the experiments listed in Tables 5.6 – 5.7; columns B1 and C1 display the corresponding iteration count and CPU time, respectively;

• column A2 presents the validation error obtained with the just outlined rule for early termination, column B2 – the iteration count corresponding to the classifier underlying this error, and column B3 – the number of iterations until "early termination". Columns C2 and C3 display the CPU times corresponding to B2 and B3;

• column A3 displays the validation error of the IP-classifier, and column C4 – the running time of the IP algorithm `modsekopt`.

We see that among the large-scale data sets there is a single one (# 7) where our simple policy for early termination of MP results in a "meaningful" loss in the classification error (6.9% vs. 3.1%). At the same time, this policy reduces the total CPU time of MP on the large-scale data ## 1 – 9 from 3515 sec to 455 sec (by factor 7.8) and makes the *total* running time of MP on the large-scale data just 15% of the running time of `mosekopt` on a *single* large-scale data set # 4.

Performance of early termination on medium- and small-scale data sets is equally attractive, but this is of no interest: the running times of the original version of MP on these data sets are pretty small, and there are no reasons exchange quality for speed.

	Data ##								
	1	2	3	4	5	6	7	8	9
#Attrib[u)]	10000	20000	100000	5000	500	5000	1558	4026	2000
#Train_ex[v)]	60	200	600	4000	1500	15000	2500	60	40
#Valid_ex[w)]	40	100	200	2000	500	2000	780	36	22
Err_Vl_bst[a)]	0.200	0.080	0.075	0.023	0.384	0.079	0.031	0.056	0.091
	[0.275]	[0.090]	[0.175]	[0.026]	[0.474]	[**Failure**]	[0.031]	[0.083]	[0.181]
Nnz_Vl_bst[b)]	0.987	0.304	0.001	0.160	1.000	0.085	0.990	1.000	0.011
	[0.988]	[0.305]	[0.001]	[0.980]	[1.000]	[**Failure**]	[0.633]	[1.000]	[0.758]
Itr_Vl_bst[c)]	41	11	50	200	100	28	900	10	25
ItrIn_Vl_bst[d)]	192	44	236	962	456	116	4672	42	120
CPU_Vl_bst[e)]	5.000	0.590	109.310	238.460	28.270	48.890	121.200	0.730	2.140
	[4.800]	[1.000]	[26.200]	[**2963.000**]	[311.300]	[**Failure**]	[6.000]	[1.900]	[2.900]
Err_Tr_bst[f)]	0.000	0.000	0.010	0.000	0.252	0.074	0.024	0.000	0.075
	[0.000]	[0.000]	[0.010]	[0.000]	[0.219]	[**Failure**]	[0.006]	[0.000]	[0.075]
Nnz_Tr_bst[g)]	0.987	0.304	0.001	0.150	0.998	0.085	0.990	1.000	0.008
Itr_tr_bst[h)]	25	11	200	150	975	28	900	10	75
ItrIn_Tr_bst[i)]	122	44	944	718	4450	116	4672	42	374
CPU_Tr_bst[j)]	3.060	0.590	558.890	174.720	276.060	48.890	121.200	0.730	6.520
	[4.800]	[1.000]	[26.200]	[**2963.000**]	[311.300]	[**Failure**]	[6.000]	[1.900]	[2.900]
Itr_tot[k)]	41	11	1000	233	1000	28	1000	10	124
ItrIn_tot[l)]	192	44	4720	1104	4554	116	5180	42	374
CPU_tot[m)]	5.00	0.59	2767.48	276.20	276.06	48.89	131.28	0.73	8.61
	[4.800]	[1.000]	[26.200]	[**2963.000**]	[311.300]	[**Failure**]	[6.000]	[1.900]	[2.900]
Obj_fin[n)]	0.000	0.000	28.359	0.000	31.997	113.511	15.849	0.000	11.666
LwB_fin[o)]	0.000	0.000	27.739	0.000	0.000	112.728	0.000	0.000	11.539
	[0.000]	[0.000]	[28.207]	[0.000]	[?**30.226**?]	[**Failure**]	[7.237]	[0.000]	[?**11.666**?]
RelAccur_fin[p)]	0.000	0.000	0.022	0.000	1.000	0.007	1.000	0.000	0.011
PrgG_fin[q)]	0.000	0.000	0.005	0.000	0.826	0.006	0.480	0.000	0.006
ρ[r)]	10.0	10.0	10.0	0.1	10.0	10.0	10.0	10.0	1.0
r[s)]	2.0 (p)	2.0 (p)	1.0 (p)	1.0 (p)	2.0 (p)	1.0 (e)	2.0 (p)	2.0 (p)	1.0 (p)
p[t)]	2.0	2.0	∞	2.0	2.0	2.0	2.0	2.0	∞

[a)]: best validation error achieved
[b)]: sparsity of classifier in [a)]
[c)]: iteration resulting in [a)]
[d)]: # of inner iterations resulting in [a)]
[e)]: CPU time (sec) resulting in [a)]
[f)]: best training error achieved
[g)]: sparsity of classifier in [f)]
[h)]: iteration resulting in [f)]
[i)]: # of inner iterations resulting in [f)]
[j)]: CPU time (sec) resulting in [f)]
[k)]: total # of iterations
[l)]: total # of inner iterations
[m)]: total CPU time (sec)
[n)]: best value of $F(\cdot)$ achieved, see (5.1.1)
[o)]: best lower bound on $\min_{\|\zeta\| \le \rho} F(\zeta)$
[p)]: final relative accuracy, see (5.2.3)
[q)]: final progress in optimality gap, see (5.2.2)
[r)]: ρ, see (5.1.1)
[s)]: $\|\zeta\| \equiv \|\zeta\|_r$ in (5.1.1); e/p stands for entropy/power-type distance-generating function
[t)]: p in (5.1.1)
[u)]: # of attributes
[v)]: cardinality of training sample
[w)]: cardinality of validation sample

Numbers in brackets [...] – results for Interior Point method `mosekopt`.

Table 5.6: Best experiments, data ## 1 – 9.

	Data ##							
	10	11	12	13	14	15	16	17
#Attrib[u)]	6	13	34	22	166	8	60	30
#Train_ex[v)]	245	200	251	6000	300	600	108	400
#Valid_ex[w)]	100	97	100	2124	176	168	100	169
Err_Vl_bst[a)]	0.310	0.072	0.100	0.165	0.188	0.220	0.250	0.024
	[0.340]	[0.144]	[0.100]	[0.169]	[0.290]	[0.238]	[0.240]	[0.030]
Nnz_Vl_bst[b)]	1.000	1.000	0.971	1.000	1.000	1.000	1.000	1.000
	[1.00]	[1.000]	[0.971]	[0.273]	[1.000]	[1.000]	[1.000]	[0.300]
Itr_Vl_bst[c)]	50	200	100	125	100	950	25	200
ItrIn_Vl_bst[d)]	232	938	484	596	504	4568	112	934
CPU_Vl_bst[e)]	0.830	3.160	1.560	46.310	3.110	37.460	0.220	6.260
	[0.100]	[0.100]	[0.300]	[0.200]	[0.500]	[0.100]	[0.100]	[0.100]
Err_Tr_bst[f)]	0.237	0.140	0.028	0.166	0.000	0.210	0.028	0.040
	[0.249]	[0.155]	[0.032]	[0.167]	[0.000]	[0.205]	[0.037]	[0.038]
Nnz_Tr_bst[g)]	1.000	1.000	0.971	1.000	1.000	1.000	1.000	1.000
Itr_tr_bst[h)]	125	300	100	125	275	200	50	600
ItrIn_Tr_bst[i)]	596	1410	484	596	1362	948	230	2794
CPU_Tr_bst[j)]	2.120	4.830	1.560	46.310	8.700	7.530	0.580	18.790
	[0.100]	[0.100]	[0.300]	[0.200]	[0.500]	[0.100]	[0.100]	[0.100]
Itr_tot[k)]	253	1000	127	170	550	1000	70	1000
ItrIn_tot[l)]	1192	4714	610	802	2698	4798	324	4638
CPU_tot[m)]	4.25	15.64	1.97	62.19	17.31	38.46	0.86	30.57
	[0.100]	[0.100]	[0.300]	[0.200]	[0.500]	[0.100]	[0.100]	[0.100]
Obj_fin[n)]	165.070	9.702	5.771	2721.534	0.000	308.801	4.441	46.556
LwB_fin[o)]	163.435	7.566	5.714	2700.887	0.000	299.939	4.397	37.679
	[164.961]	[9.691]	[5.758]	[2706.000]	[0.000]	[307.552]	[4.421]	[41.4089]
RelAccur_fin[p)]	0.010	0.220	0.010	0.008	0.000	0.029	0.010	0.191
PrgG_fin[q)]	0.008	0.151	0.004	0.004	0.000	0.021	0.004	0.029
ρ[r)]	1.0	10.0	10.0	10.0	10.0	1.0	10.0	2.0
r[s)]	1.0 (p)	1.0 (e)	2.0 (p)	1.0 (e)	2.0 (p)	1.0 (e)	2.0 (p)	1.0 (e)
p[t)]	∞	2.0	2.0	∞	∞	∞	2.0	∞

For footnotes, see Table 5.6.

Numbers in brackets [...] – results for Interior Point method `mosekopt`.

Table 5.7: Best experiments, data ## 10 – 17.

Data #	A1	A2	A3	B1	B2	B3	C1	C2	C3	C4
1	0.200	0.200	0.275	41	41	41	5.00	5.00	5.00	4.800
2	0.080	0.080	0.090	11	11	11	0.59	0.59	0.59	1.000
3	0.075	0.075	0.175	50	50	75	109.31	109.31	197.02	26.200
4	0.023	0.026	0.026	200	125	150	238.46	140.49	174.72	**2963.000**
5	0.384	0.396	0.474	100	25	50	28.27	6.78	13.67	311.300
6	0.079	0.079	**Failure**	28	28	28	48.89	48.89	48.89	**Failure**
7	0.031	0.069	0.031	900	50	75	121.20	6.60	9.99	6.000
8	0.056	0.056	0.083	10	10	10	0.73	0.73	0.73	1.900
9	0.091	0.091	0.181	25	25	50	2.14	2.14	4.31	2.900
10	0.310	0.310	0.340	50	50	75	0.83	0.83	1.27	0.100
11	0.072	0.134	0.144	200	75	100	3.16	1.08	1.49	0.100
12	0.100	0.120	0.100	100	25	50	1.56	0.38	0.75	0.300
13	0.165	0.165	0.169	125	150	170	46.31	55.37	62.19	0.500
14	0.188	0.188	0.290	100	100	125	3.11	3.11	3.88	0.200
15	0.220	0.238	0.238	950	200	225	37.46	7.53	8.48	0.100
16	0.250	0.250	0.240	25	25	50	0.22	0.22	0.58	0.100
17	0.024	0.053	0.030	200	125	150	6.26	3.81	4.67	0.100

A1: best validation error achieved with MP
A2: best validation error achieved with early termination of MP
A3: best validation error achieved with IP method
B1: iteration count for A1
B2: iteration count for A2
B3: iteration count before early termination
C1: CPU time for B1, sec
C2: CPU time for B2, sec
C3: CPU time for B3, sec
C4: CPU time for `mosekopt`, sec

Table 5.8: Performance of Mirror-Prox algorithm with early termination.

5.4 Experiments: conclusions

The outlined experimental data suggest the following qualitative conclusion:

The Mirror Prox algorithm seems to be a highly attractive computational tool for processing large-scale SVM models; it allows to get reasonably good classifiers at an essentially lower computational cost than the Interior Point polynomial time methods. As an additional bonus, *the quality of MP-originating classifiers in our experiments is never worse, and in many cases – essentially better than the quality of classifiers associated with the precise Interior Point solutions to the corresponding optimization models.*

> The first of these two conclusions is in full accordance with the theoretical considerations which led us to the idea to use "computationally cheap" first-order methods to process large-scale SVM models. The second, somehow less expected, conclusion reflects the phenomenon which is observed in other applications with noisy data (e.g., reconstruction of medical images from noisy data). In these applications, solving the optimization problem is a tool rather than a goal, and high accuracy in optimization terms does not automatically mean high quality in a particular application, and more often than not, it happens that as the optimization process goes on, this quality first improves and then starts to deteriorate. What happens can be informally explained as follows. At the initial phase of optimization problem, improving the objective means adjusting the solution to "essential structure" of the data, and this is exactly what we want. However, at certain "turning" point in time, due to the noisy nature of the data, further progress in the objective is achieved mainly by adjusting the solution to the noise component of the data, which is exactly *opposite* to what we want. In many cases it happens that small improvements in the objective after the turning point is passed require significant (and in fact counter-productive) changes in the solution. In situations like this, high accuracy is not only expensive – it is counter-productive; and there are good chances to believe that this is what happens with large-scale SVM models.

Finally, it should be stressed that the encouraging conclusions we have outlined, while fully supported by our experiments, definitely should not be considered as final – we have worked with a particular family of SVM data, restricted ourselves with plain SVM models, etc. We, however, believe that our results justify further research on the potential of advanced techniques for large-scale convex optimization in the SVM context.

Bibliography

[1] N. Cristiani, J. Shawe-Taylor, *An Intorduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, 2000.

[2] Vapnik, V., *Statistical Learning Theory*, Wiley, 1998.

[3] Nemirovski, A., "Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems" – *SIAM Journal on Optimization* **v.** 15 (2004), 229-251.

[4] Nemirovski, A., "Regular Banach spaces and large deviations of random sums", http://iew3.technion.ac.il/Home/Users/SECOND.php?Nemirovski+Arkadi+Nemirovski+1+4

[5] Nemirovski, A.S., Yudin, D.B., *Problem complexity and Method Efficiency in Optimizatoon*, J. Wiley & Sons, 1983.

[6] Nesterov, Yu. (2003), "Smooth minimization of nonsmooth functions" – CORE Discussion Paper 2003/12, February 2003. http://www.core.ucl.ac.be/services/COREdp03.html

[7] Nemirovskii, A., and Nesterov, Yu., "Optimal methods for smooth convex optimization" (in Russian) – *Jurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, v. 25 (1985) No. 3 (the journal is translated into English as *USSR J. Comput. Math. & Math. Phys.*)

[8] Nesterov, Yu., and Nemirovskii, A. *Interior point polynomial methods in Convex Programming. - SIAM Series in Applied Mathematics*, SIAM: Philadelphia, 1994

[9] Nesterov, Yu. (2003), "Dual extrapolation and its applications for solving variational inequalities and related problems" – CORE Discussion Paper 2003/68, September 2003. http://www.core.ucl.ac.be/services/COREdp03.html