UNIVERSITY OF SOUTHAMPTON

Perceptron-Like Large Margin Classifiers

by

Petroula Tsampouka

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

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

June 2007

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

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

Doctor of Philosophy

PERCEPTRON-LIKE LARGE MARGIN CLASSIFIERS

by Petroula Tsampouka

We address the problem of binary linear classification with emphasis on algorithms that lead to separation of the data with large margins. We motivate large margin classification from statistical learning theory and review two broad categories of large margin classifiers, namely Support Vector Machines which operate in a batch setting and Perceptron-like algorithms which operate in an incremental setting and are driven by their mistakes. We subsequently examine in detail the class of Perceptron-like large margin classifiers. The algorithms belonging to this category are further classified on the basis of criteria such as the type of the misclassification condition or the behaviour of the effective learning rate, i.e. the ratio of the learning rate to the length of the weight vector, as a function of the number of mistakes. Moreover, their convergence is examined with a prominent role in such an investigation played by the notion of stepwise convergence which offers the possibility of a rather unified approach. Whenever possible, mistake bounds implying convergence in a finite number of steps are derived and discussed. Two novel families of approximate maximum margin algorithms called CRAMMA and MICRA are introduced and analysed theoretically. In addition, in order to deal with linearly inseparable data a soft margin approach for Perceptron-like large margin classifiers is discussed. Finally, a series of experiments on artificial as well as real-world data employing the newly introduced algorithms are conducted allowing a detailed comparative assessment of their performance with respect to other well-known Perceptron-like large margin classifiers and state-of-the-art Support Vector Machines.

Contents

Acknowledgements

I would like to thank my supervisor Professor John Shawe-Taylor for his advice and support.

Chapter 0

Introduction

0.1 Overview

Machine learning deals with the problem of learning dependencies from data. The machine (algorithm) is provided with a training set consisting of examples drawn independently from the same distribution. We distinguish three main learning problems: the problems of classification (pattern recognition), regression and density estimation. In the classification task the examples are given in the form of instance-label pairs with the labels taking integer values which indicate which out of a certain number of classes the instances belong to. In the regression task each instance is accompanied by a real number related to it either through an unknown deterministic functional dependency or in the most general case through a conditional probability. Finally, in the density estimation task no additional information is provided to the machine other than the instances themselves. In the training process the machine tries to infer a functional relation mapping the input space of instances to the output space which is either the set of discrete values that the labels take (in the classification task) or the set of real numbers (in the regression task). In the density estimation task in which the machine is looking for the underlying distribution governing the data no such mapping exists. Any such function that the machine returns as an output is known as a hypothesis. In the present work we will only be concerned with the binary classification problem in which the labels can take only one out of two values.

The goal of the machine is to construct a hypothesis able to predict the labels of instances which did not participate in the training procedure. If we impose no restriction on the functions which the machine chooses its hypothesis from it might be that most of the instances of an independent set will be wrongly classified by the hypothesis produced even though this hypothesis explains the training data satisfactorily. The above situation can be remedied if the richness of the function class employed by the machine is appropriately restricted. There are various classes of functions which the learning

algorithms draw their hypothesis from with the most prominent ones being the classes of linear functions, polynomials and radial basis functions.

By embedding the data in an appropriate feature space via a nonlinear mapping we can always use a machine employing the class of linear discriminants. The hyperplane generated by the machine which performs the separation of the training examples in the feature space corresponds to a nonlinear curve in the original space. In many learning algorithms the mapping can be efficiently performed by the use of kernels. Therefore, the treatment of classes which contain nonlinear functions can be considered a straightforward extension of the linear case whenever the kernel trick is applicable.

A broad categorisation of the algorithms could be done according to the learning model that they follow. In the first category are those algorithms that follow the online model. According to the online model learning proceeds in trials and the instances become accessible one at a time. More specifically, in each trial the algorithm observes an instance and makes a guess of its binary label. Then, the true label of the instance is revealed and if there is a mismatch between the prediction and the actual label the algorithm incurs a mistake. The algorithm maintains in every trial a hypothesis defining the linear discriminant which is updated with every mistake. A natural extension of online learning is the incremental or sequential setting of learning in which the algorithm cycles repeatedly through the examples until no mistake occurs. The most prominent algorithm in this category is the Perceptron [47]. In the second category belong the algorithms which follow the batch learning model and which have access to all training instances prior to the running of the algorithm. Well-known examples of algorithms in the second category are the Support Vector Machines (SVMs) [9]. The SVMs solve an optimisation problem with a quadratic objective function and linear constraints. SVMs in contrast to Perceptrons minimise the norm of the weight vector defining the solution hyperplane. This is actually equivalent to maximising the minimum distance (margin) of the training instances from that hyperplane.

The algorithms that reach convergence find, in the linearly separable case, a hypothesis consistent with the training data. However, even such solutions may fail to classify correctly unseen (test) data. The ability of a hypothesis to classify correctly test data is known as the ability to generalise. The generalisation ability of a linear classifier is believed to be enhanced as the margin of the training instances from the hyperplane solution becomes larger. This favours considerably algorithms like SVMs which find solutions possessing large margin.

Although the Perceptron algorithm in contrast to SVMs does not insist on finding large margin solutions its simplicity and its online mode of implementation render it very attractive. For this reason a considerable amount of effort has been devoted to developing and analysing online algorithms able to approximate the maximal margin hyperplane

to any accuracy. Such algorithms include the Maximal Margin Perceptron [34], agg-ROMMA [38] and ALMA$_p$ [21]. In order to obtain solutions with margin their misclassification condition (i.e. the condition that decides whether a mistake occurs) becomes stricter and is satisfied not only if the predicted label of the instance is wrong but also in the case of correct classification with a lower than the desirable value of the margin. Our work presented here follows the same line.

0.2 Contributions

Our main contributions are the following:

We developed Perceptron-like algorithms with margin in which the misclassification condition is modified to require a fixed value of the margin. These new algorithms are radically different from the previous approaches which implement a misclassification condition relaxing with time (i.e. with the number of mistakes). With the condition kept fixed two generic classes of algorithms emerged, the one leaving the length of the weight vector determining the hypothesis hyperplane free to grow indefinitely and the other keeping the weight vector normalised to a fixed length. The new algorithms converge to a solution with a fixed value of the margin in a finite number of steps and may be used as modules of more complex algorithmic constructions in order to approximately locate the optimal weight vector. Additionally, we introduced stepwise convergence, the ability of the algorithm to approach the optimum weight vector at each step, and making use of it we developed a unified approach towards proving convergence of Perceptron-like algorithms in a finite number of steps. This research led to [56].

We also introduced the "effective" learning rate, the ratio of the learning rate to the length of the weight vector, and performed a classification of Perceptron-like algorithms into four broad categories according to whether the misclassification condition is fixed or relaxing with time and according to whether the effective learning rate is fixed or decreasing with time. The classification revealed that the Perceptron with margin and ALMA$_2$ belong to the same category whereas the algorithms with fixed margin condition that we described above cover both cases with respect to the behaviour of the effective learning rate. Thus, the existence of algorithms with misclassification condition relaxing with time and constant effective learning rate was left an open issue. Guided by this observation we developed a novel class of algorithms called CRAMMA$^\epsilon$ which belong to this last category and are parametrised by ϵ which determines the power of the number of mistakes in the law according to which the misclassification condition is relaxed. We proved that for a sufficiently small effective learning rate the new class of algorithms converges in a finite number of steps and showed that under some conditions there exists a limit of the parameters involved in which convergence leads to classification with maximum margin. Moreover, in order to deal with linearly inseparable data a soft

margin extension for Perceptron-like large margin classifiers was presented following the approach of [17] and was shown to correspond to a partial optimisation of an appropriate criterion. This research led to [57].

Finally, we constructed MICRA$^{\epsilon,\zeta}$ an approximate maximum margin algorithm in which both the effective learning rate and the misclassification condition are entirely controlled by rules involving powers of the number of mistakes. Since the effective learning rate of MICRA decreases with the number of mistakes there is no condition on its initial value for convergence to occur. CRAMMA may be regarded as a limiting case of MICRA with the parameter ζ controlling the rate of decrease of the effective learning rate set to zero. We provided a theoretical analysis of MICRA and investigated the conditions under which MICRA converges asymptotically to the optimal solution with maximum margin. We also presented a variation of the standard sequential way of cycling through the data which leads to considerable improvement in running times. An extensive comparative experimental investigation revealed that MICRA with $\epsilon \ll 1$ and $\zeta \simeq 1$ is very competitive. This research led to [58].

0.3 Publications

This work has contributed to the following publications:

- Tsampouka, P., Shawe-Taylor, J.: Analysis of generic perceptron-like large margin classifiers. ECML 2005, LNAI **3720** (2005) 750–758, Springer-Verlag
- Tsampouka, P., Shawe-Taylor, J.: Constant rate approximate maximum margin algorithms. ECML 2006, LNAI **4212** (2006) 437–448, Springer-Verlag
- Tsampouka, P., Shawe-Taylor, J.: Approximate maximum margin algorithms with rules controlled by the number of mistakes. In Proceedings of the 24$^{\text{th}}$ International Conference on Machine Learning (2007) 903–910

0.4 Thesis Outline

The present thesis is organised as follows.

Chapter 1 contains an introductory discussion of data representation in the initial instance space and in kernel-induced feature spaces. The properties characterising functions which are kernels are described and examples of such functions are given.

Chapter 2 presents some elements of statistical learning theory in order to elucidate the factors responsible for the generalisation ability of learning machines and provide motivation for the large margin classifiers that will be described in subsequent chapters.

Chapter 3 contains an introduction to Support Vector Machines and related techniques.

Chapter 4 contains a review of some well-known incremental mistake-driven algorithms. The algorithms are presented in an order that depends on their ability to achieve margin. Therefore, we begin with the standard First-Order and the Second-Order Perceptron and subsequently we move to algorithms that succeed in obtaining some margin such as the Perceptron with margin and the relaxation algorithmic procedures. Finally, algorithms such as ALMA$_p$, ROMMA and the Maximal Margin Perceptron that are able to reach the solution with maximum margin are discussed.

In Chapter 5 we present in detail our work on Perceptron-like Large Margin Classifiers. First we attempt a taxonomy of such algorithms. Subsequently, we introduce the notion of stepwise convergence. Then, we proceed to the analysis of the generic Perceptron-like algorithms with the standard margin condition and (a slight modification) of the ALMA$_2$ algorithm followed by the Constant Rate Approximate Maximum Margin Algorithm CRAMMA$^{\epsilon}$, the Mistake-Controlled Rule Algorithm MICRA$^{\epsilon,\zeta}$ and algorithms with fixed margin condition.

Chapter 6 contains a soft margin extension applicable to all Perceptron-like classifiers.

Chapter 7 contains an experimental comparative study involving our algorithms and other well-known large margin classifiers.

Finally, Chapter 8 contains the conclusion of the thesis.

Chapter 1

Kernel-Induced Feature Spaces

1.1 Data Representation

At the core of machine learning theory lies the problem of identifying which category out of many possible ones an object belongs to. To this end a machine (algorithm) is trained using objects from distinct classes in order to learn the properties characterising these classes. After the training phase is completed an unknown object can be assigned to one of the classes on the basis of its properties. The objects used for training are called training points, patterns, instances or observations and will be denoted by $\boldsymbol{x}$. The different classes are distinguished by their label or target value y. In the present thesis we will be concerned only with binary classification, thus restricting y to belong to the set $\{1, -1\}$.

It is important to describe the procedure followed by a learning machine in order to assign a label to an instance $\boldsymbol{x}$ with unknown label. The machine chooses y in such a way that the newly presented point shows some similarity or dissimilarity to the points that already belong to one of the classes. In order to assess the degree of similarity we have to define a similarity measure. A commonly used similarity measure is the inner product which, of course, necessitates a vector representation of the instances in some inner product space. A first stage in the assignment of a label to a newly presented instance involves the computation of the value of a function $f(\boldsymbol{x})$ which maps the n-dimensional input $\boldsymbol{x}$ into the real numbers. The function f usually assumes the linear in $\boldsymbol{x}$ form

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b = \sum_{i=1}^{n} w_i x_i + b \ . \tag{1.1}$$

In the above relationship the n-dimensional vector $\boldsymbol{w}$, called the weight vector, defines the normal to a hyperplane that splits the data space into two halfspaces. The quantity b called bias is related to the distance of the hyperplane from the origin. The test point the label of which is to be determined is then given the label $y = 1$ (i.e. the point belongs

to the positive class) if $f(\boldsymbol{x}) > 0$, otherwise the label $y = -1$ is assigned to it (i.e. the point belongs to the negative class). The parameters $\boldsymbol{w}$ and b defining the hyperplane are determined by the training process.

Let us assume that the weight vector $\boldsymbol{w}$ can be expressed as a linear combination of the l patterns contained in the training set

$$\boldsymbol{w} = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x}_i \ ,$$

a representation which is by no means unique. This expansion in $\boldsymbol{x}_i$'s of the outcome of the training procedure is called the dual representation of the solution. If $\boldsymbol{w}$ is substituted back in (1.1) we obtain

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b \ .$$

We observe that the label

$$y = \text{sign}(f(\boldsymbol{x}))$$

of the test point $\boldsymbol{x}$ is evaluated using only inner products between the test point and the training patterns. This elucidates the role of the inner product as a similarity measure.

1.2 Learning in the Feature Space

Even if the patterns already belong to some dot product space we may choose a nonlinear mapping $\boldsymbol{\phi}$ into a space $\mathcal{H}$ which from now on will be called the feature space. Since the data admit a vector representation the embedding can be expressed as follows

$$\boldsymbol{x} = (x_1, \ldots, x_n) \longrightarrow \boldsymbol{\phi}(\boldsymbol{x}) = (\phi_i(\boldsymbol{x}), \ldots, \phi_N(\boldsymbol{x})) \ .$$

Here n indicates the dimensionality of the original space, usually referred to as the input space, whereas N denotes the dimensionality of the feature space. The components of the vector $\boldsymbol{\phi}(\boldsymbol{x})$ resulting from the mapping into the feature space are called features to be distinguished from the components of the vector $\boldsymbol{x}$ which will be referred to as attributes.

The question then that naturally arises is why should one proceed to a mapping into another space? In the previous section we constructed a hyperplane that classifies the training points into two classes and we used the hyperplane performing the separation to determine the labels of the unseen test points. It may, however, happen that the points are not linearly separable in the input space. In this case a nonlinear tranformation into a feature space where classification is possible with a linear surface can ease the difficulty

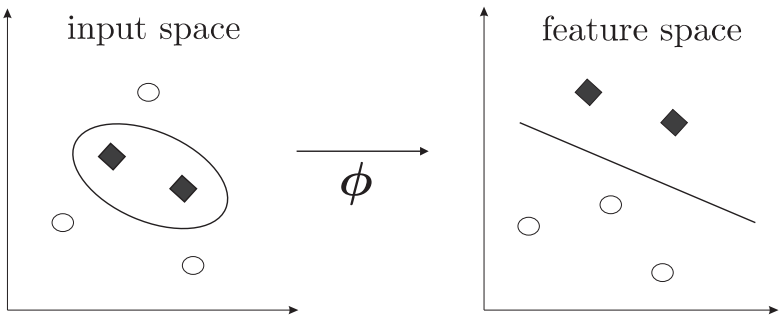

FIGURE 1.1: Linearly inseparable data in the input space can become linearly separable by mapping them via $\boldsymbol{\phi}$ into a higher dimensional feature space. Then a linear decision boundary in the feature space yields a nonlinear curve in the input space.

(Fig. 1.1). There is also a possibility that the original vector representation of $\boldsymbol{x}$ includes many irrelevant attributes. This may disorientate the training procedure from a solution able to successfully predict the unknown labels. Such undesirable situations can be avoided by an embedding of the data into an appropriate feature space which encodes a prior knowledge about the problem and offers a representation with the suitable number of features.

As an example of how to explicitly incorporate our prior knowledge in the feature mapping let us assume that the only relevant information about a given task is contained in monomials of degree 2 formed out of the attributes of $\boldsymbol{x}$. Then, taking for the sake of simplicity the dimensionality of the input space to be equal to 2, we have the following embedding

$$\boldsymbol{x} = (x_1, x_2) \longrightarrow \boldsymbol{\phi}(x_1, x_2) = (x_1^2, x_2^2, x_1 x_2) \ .$$

In the case of an input space of dimensionality equal to n one can construct the feature space of all possible monomials of degree exactly equal to d by an analogous procedure. The dimensionality N of such a feature space is then given by the formula

$$N = \left(\begin{array}{c} n+d-1 \\ d \end{array} \right) = \frac{(n+d-1)!}{d!(n-1)!} \ .$$

It is apparent that a large number of attributes n in combination with a large value of d will eventually lead to an explosion in the number of dimensions of the feature space making the explicit construction of such a feature mapping computationally impractical.

1.3 Implicit Mapping via a Kernel

An inseparable problem can be rendered separable by an appropriate feature mapping. This will enable us to use in that feature space algorithms that may already exist which are able to compute separating hyperplanes. Such algorithms implemented in the feature

space produce functions assigning labels to the unseen patterns which are of the type

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} w_i \cdot \phi_i(\boldsymbol{x}) + b \ . \tag{1.2}$$

Assuming again that the weight vector $\boldsymbol{w}$ admits a dual representation with respect to the transformed training data (1.2) becomes

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}) + b \ . \tag{1.3}$$

The previous relationship involves an explicit non-linear mapping $\boldsymbol{\phi}$ from the space X which the training data and the test point live in into the feature space $\mathcal{H}$. The function $f(\boldsymbol{x})$ is constructed by a linear machine working in the feature space aiming at classifying the training data by means of a hyperplane. The linear surface performing the classification in $\mathcal{H}$ corresponds to a non-linear surface in the input space. Notice that here the inner product of the transformed instances plays the role of the similarity measure. Equation (1.3) suggests that explicit knowledge of the feature mapping would be obsolete if one was able to compute directly the inner product involving the training data and the test point in the feature space. The previous observation that only the knowledge of the inner product suffices if we are interested in constructing appropriate classification functions in the feature space motivates the following definition of the kernel function.

Definition 1.1. A kernel is a function $K : X \text{ x } X \rightarrow \mathbb{R}$, such that for all $x, x' \in X$

$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x}')$$

with $\boldsymbol{\phi}$ a mapping from X to an inner product feature space $\mathcal{H}$.

The kernel function K takes as arguments any two points $\boldsymbol{x}$ and $\boldsymbol{x}'$ and returns a real number measuring the similarity of their images in the feature space. Thus, the kernel may be viewed as a similarity measure "transformed" by the feature mapping. This becomes more obvious if we consider in the place of $\boldsymbol{\phi}$ the identity function which defines a kernel coinciding with the inner product in the input space

$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x} \cdot \boldsymbol{x}' \ .$$

We assumed earlier that the weight vector can be decomposed as a linear combination of the transformed patterns thus giving rise to the dual representation. By taking into account the definition of the kernel (1.3) can be equivalently rewritten as

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b \ . \tag{1.4}$$

The decision rule $f(\boldsymbol{x})$ entails at most l kernel evaluations which are as many as the data contained in the training set. A mere observation of (1.4) reveals that the introduction of the kernel eliminates any complications in the computation of the decision rule $f(\boldsymbol{x})$ stemming from the possibly large dimensionality of the feature space. Therefore, the number of operations involved in the evaluation of the kernel function are not necessarily controlled by the number of features, thereby reducing the computational complexity of the decision rule.

Let us attempt to illustrate how it is possible by constructing directly the kernel to avoid the operations involved in the evaluation of the inner product in the feature space associated with an explicit description of the feature mapping. For our demonstration we choose to evaluate the square of the inner product in the input space and investigate the possibility that it constitutes a kernel. Each vector participating in the inner product is analysed in its components and the square of the product is computed as follows

$$(\boldsymbol{x} \cdot \boldsymbol{x}')^2 = \left(\sum_{i=1}^{n} x_i x_i'\right)^2 = \left(\sum_{i=1}^{n} x_i x_i'\right)\left(\sum_{j=1}^{n} x_j x_j'\right)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j x_i' x_j' = \sum_{(i,j)=(1,1)}^{(n,n)} (x_i x_j)\left(x_i' x_j'\right) \quad .$$

From the preceding analysis we can conclude that the square of the inner product constitutes a kernel which performs a mapping into a feature space consisting of monomials of degree exactly 2

$$\boldsymbol{\phi}(\boldsymbol{x}) = (x_i x_j)_{(i,j)=(1,1)}^{(n,n)} \quad .$$

Notice that the features $x_i x_j$ for $i \neq j$ appear twice leading to a double weight in comparison to the weight of x_i^2. In order to build a space the features of which are the monomials up to degree 2 we have to add a constant parameter c to the inner product before raising it to the second power. The resulting kernel is analysed as follows

$$(\boldsymbol{x} \cdot \boldsymbol{x}' + c)^2 = \sum_{(i,j)=(1,1)}^{(n,n)} (x_i x_j)\left(x_i' x_j'\right) + \sum_{i=1}^{n}\left(\sqrt{2c}x_i\right)\left(\sqrt{2c}x_i'\right) + c^2 \quad .$$

The transformation $\boldsymbol{\phi}$ corresponding to the above kernel comprises as features in addition to monomials of degree 2 monomials of degree 1 and 0 which are weighted according to the parameter c.

1.4 Characterisation of Kernels

The procedure that we followed in the previous section for the construction of polynomial kernels was to first carry out the inner product in the feature space by making use of the

explicit mapping which was known to us and then infer from it how the same result could be derived by employing only quantities associated with the input space. It would be very useful, however, to identify the properties characterising a kernel since this would enable us to directly construct kernels without even knowing the explicit form of the feature mapping. From the definition of the kernel it is easily derived that every kernel is symmetrical in its arguments

$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x}') \cdot \boldsymbol{\phi}(\boldsymbol{x}) = K(\boldsymbol{x}', \boldsymbol{x}) \ .$$

A second property of kernels comes from the Cauchy-Schwarz inequality which yields

$$K^2(\boldsymbol{x}, \boldsymbol{x}') = \left(\boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x}')\right)^2 \leq \|\boldsymbol{\phi}(\boldsymbol{x})\|^2 \left\|\boldsymbol{\phi}(\boldsymbol{x}')\right\|^2 = K(\boldsymbol{x}, \boldsymbol{x}) K(\boldsymbol{x}', \boldsymbol{x}') \ .$$

Nevertheless, none of these properties suffice to ensure that the function under consideration is indeed a kernel. A function is certainly a kernel only if it represents an inner product defined in a feature space to which $\boldsymbol{x} \in X$ is mapped via $\boldsymbol{\phi}$. Hence, we have to identify the conditions that guarantee the existence of such a mapping $\boldsymbol{\phi}$.

Let us consider a finite input space $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_l)$ and suppose $K(\boldsymbol{x}, \boldsymbol{x}')$ is a symmetric function on X. Using this function we can construct a square matrix $\boldsymbol{K}$ the entries (i, j) of which are filled with the values that the function K assumes for every pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$

$$\boldsymbol{K} = \left(K(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)_{(i,j)=1}^{l} \ .$$

Since this matrix is furthermore symmetrical it can be decomposed as $\boldsymbol{K} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$, where $\boldsymbol{\Lambda}$ is a diagonal matrix with elements the eigenvalues λ_i of $\boldsymbol{K}$ and $\boldsymbol{V}$ an orthogonal matrix with entries $\boldsymbol{v}_{ij}$ the columns of which are eigenvectors of $\boldsymbol{K}$. Next we consider the feature mapping $\boldsymbol{\phi}$ which maps $\boldsymbol{x}_i$ to a vector consisting of the ith component of all eigenvectors scaled by the square root of the accordingly indexed eigenvalues

$$\boldsymbol{\phi} : \boldsymbol{x}_i \longrightarrow \boldsymbol{\phi}(\boldsymbol{x}_i) = \left(\sqrt{\lambda_1} v_{i1}, \ldots, \sqrt{\lambda_l} v_{il}\right) \ . \tag{1.5}$$

In order to perform the above mapping we make the assumption that the eigenvalues are non-negative. Looking at (1.5) we realise that with such a mapping we are able to recover any entry k_{ij} of the matrix $\boldsymbol{K}$ if we work out the inner product $\boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j)$

$$\boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}_j) = \sum_{t,r} \delta_{tr} \sqrt{\lambda_t} v_{it} \sqrt{\lambda_r} v_{jr} = \sum_{t,r} v_{it} \Lambda_{tr} v_{jr} = \left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\right)_{ij} = k_{ij} \ .$$

Here δ_{ij} is Kronecker's δ. This implies that $K(\boldsymbol{x}, \boldsymbol{x}')$ is indeed a kernel function corresponding to the feature mapping $\boldsymbol{\phi}$. The constructed feature space with dimensionality at most l is spanned by the l vectors $\boldsymbol{\phi}(\boldsymbol{x}_i), i = 1, \ldots, l$ which resulted from the mapping. Let us assume now that $K(\boldsymbol{x}, \boldsymbol{x}')$ is a kernel function corresponding to a feature mapping $\boldsymbol{\phi}$ defined on X and consider a vector $\boldsymbol{z}$ written as the following linear combination of

$\boldsymbol{\phi}(\boldsymbol{x}_i)$'s

$$\boldsymbol{z} = \sum_{i=1}^{l} v_{is}\boldsymbol{\phi}(\boldsymbol{x}_i) \ .$$

The squared norm of $\boldsymbol{z}$ is given by

$$\|\boldsymbol{z}\|^2 = \left(\sum_i v_{is}\boldsymbol{\phi}(\boldsymbol{x}_i)\right) \cdot \left(\sum_j v_{js}\boldsymbol{\phi}(\boldsymbol{x}_j)\right) = \sum_{i,j} v_{is}v_{js}k_{ij}$$

$$= \left(\mathbf{V}^T\boldsymbol{K}\mathbf{V}\right)_{ss} = \Lambda_{ss} = \lambda_s \ .$$

Since such a squared norm of a vector is a non-negative quantity the eigenvalues of $\boldsymbol{K}$ must be non-negative. The above discussion concerning the eigenvalues of $\boldsymbol{K}$ leads to the following proposition.

Proposition 1.2. *Let X be a finite input space with $K(\boldsymbol{x}, \boldsymbol{x}')$ a symmetrical function with respect to its arguments. Then $K(\boldsymbol{x}, \boldsymbol{x}')$ is a kernel function if and only if the matrix*

$$\boldsymbol{K} = (K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{(i,j)=1}^{l}$$

is positive semi-definite.

In the above discussion that led to Proposition 1.2 we studied the eigenvalue problem of a kernel matrix $\boldsymbol{K}$ constructed by the values that the kernel function assumes on every possible pair drawn from a finite number of points. More specifically, we examined any restrictions that may hold for the eigenvalues λ_s associated with the following eigenvalue problem

$$\boldsymbol{K}\mathbf{v}_s = \lambda_s\mathbf{v}_s \ .$$

An extension of this eigenvalue problem in order to cover the case of an infinite input space X is

$$\int_X K(\boldsymbol{x}, \boldsymbol{x}')\psi(\boldsymbol{x}')d\boldsymbol{x}' = \lambda\psi(\boldsymbol{x}) \ .$$

This is an eigenvalue problem in an infinite dimensional space where the role of the eigenvectors is played by the eigenfunctions ψ. In analogy with Proposition 1.2 the following theorem [42] is concerned with the conditions that K should fulfill in order for it to be a kernel function.

Theorem 1.3. *(Mercer) Let X be a compact subset of $\mathbb{R}^n$. Suppose K is a continuous symmetric function such that the integral operator $T_K : L_2(X) \to L_2(X)$ defined by*

$$(T_K f)(\boldsymbol{x}) = \int_X K(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x}')d\boldsymbol{x}'$$

is positive, meaning that

$$\int_{X\mathrm{x}X} K(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x})f(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}' \geq 0 \tag{1.6}$$

for all $f \in L_2(X)$. Here $L_2(X)$ is the space of measurable functions over X which are square integrable. Let us denote by $\psi_i \in L_2(X)$ the normalised ($\|\psi_i\|_{L_2} = 1$) orthogonal eigenfunctions of T_K associated with the eigenvalues $\lambda_i \geq 0$. Then,

$$\sum_{i=1}^{\infty} \lambda_i < \infty$$

and $K(\boldsymbol{x}, \boldsymbol{x}')$ can be expanded as a uniformly convergent series

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{\infty} \lambda_i \psi_i(\boldsymbol{x}) \psi_i(\boldsymbol{x}') \ . \tag{1.7}$$

Observe that the condition for positivity of the operator T_K can be reduced to the conditions for positive semi-definiteness of a kernel matrix $\boldsymbol{K} = (K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{(i,j)=1}^{l}$. In order to obtain the latter we have to choose in the place of the functions $f(\boldsymbol{x})$ appearing in (1.6) a weighted sum of delta functions at the points $\boldsymbol{x}_i, i = 1, \ldots, l$. The weights of the delta functions will form a vector $\mathbf{v}$ for which

$$\mathbf{v}^T \boldsymbol{K} \mathbf{v} \geq 0$$

holds true. We can assert the converse too, i.e. if the positivity of T_K is violated then we can approximate the integral appearing in (1.6) by a sum over a sufficiently large number of appropriately chosen points which will be negative. This proves that Mercer's theorem provides the general conditions for a function to be characterised as a kernel subsuming the specific case of a finite input space.

Mercer's theorem suggests the feature mapping

$$\boldsymbol{x} = (x_1, \ldots, x_n) \longrightarrow \boldsymbol{\phi}(\boldsymbol{x}) = (\psi_1(\boldsymbol{x}), \ldots, \psi_j(\boldsymbol{x}), \ldots)$$

into the feature space $\mathcal{H}$ which is the Hilbert space $l_2(\boldsymbol{\lambda})$ of all sequences $\boldsymbol{z} = (z_1, z_2, \ldots, z_i, \ldots)$ such that

$$\|\boldsymbol{z}\|^2 = \sum_{i=1}^{\infty} \lambda_i z_i^2 < \infty \ ,$$

where the inner product of sequences $\boldsymbol{x}$ and $\boldsymbol{z}$ is defined by

$$\boldsymbol{x} \cdot \boldsymbol{z} = \sum_{i=1}^{\infty} \lambda_i x_i z_i \ .$$

This is so since the inner product of two feature vectors satisfies

$$\boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x}') = \sum_{i=1}^{\infty} \lambda_i \psi_i(\boldsymbol{x}) \psi_i(\boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}') \ .$$

In order for the kernel to be represented an infinite number of eigenfunctions may be actually required or it may be the case that the infinite series reduces to a finite sum due to the vanishing of λ_i for i sufficiently large. In the latter case involving a space of finite dimensionality $N_{\mathcal{H}}$, where ψ_i's form an orthonormal basis, $K(\boldsymbol{x}, \boldsymbol{x}')$ can be considered as an inner product in $\mathbb{R}^{N_{\mathcal{H}}}$. Since $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x}')$ the components entering the inner product will be determined by the induced mapping $\boldsymbol{\phi}$ on the points $\boldsymbol{x}$

$$\boldsymbol{\phi} : \boldsymbol{x} \longrightarrow (\psi_i(\boldsymbol{x}))_{i=1}^{N_{\mathcal{H}}} \quad .$$

Even when the induced space is of infinite dimensionality we can approximate the kernel function K within some accuracy ϵ if a space of an appropriate dimensionality n is found into which the points are mapped. If the eigenvalues $\lambda_i, i = 1, \ldots, n, \ldots$ are sorted in a non-increasing order and a mapping

$$\boldsymbol{\phi} : \boldsymbol{x} \rightarrow (\psi_i(\boldsymbol{x}), \ldots, \psi_n(\boldsymbol{x}))$$

is performed we can achieve that

$$|K(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x}')| < \epsilon \ .$$

The features ψ_i entering (1.7) have the additional property that they are orthonormal. It seems that the property of orthogonality is inherent in Mercer's theorem and it is related to the eigenfunctions of the integral operator T_K constructed for the specific kernel function. Note that this is not required and we may optionally allow mappings that do not involve orthonormal features. To such a case belongs the kernel which corresponds to all the monomials of second degree. In addition to the flexibility that we have shown as far as the orthogonality is concerned we can also allow for a rescaling of the features

$$\boldsymbol{x} = (x_1, \ldots, x_n) \rightarrow \boldsymbol{\phi}(\boldsymbol{x}) = (b_1\phi_1(\boldsymbol{x}), \ldots, b_j\phi_j(\boldsymbol{x}), \ldots) \ ,$$

where by $b_1, \ldots, b_j, \ldots$ we denote the rescaling factors. We can recover the relation $\boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}')$ if we define an altered inner product

$$\boldsymbol{x} \cdot \boldsymbol{z} = \sum_{j=1}^{\infty} \frac{\lambda_j}{b_i^2} x_j z_j$$

where $\boldsymbol{x}, \boldsymbol{z}$ denote any two points in the feature space. Employing the new definition we obtain

$$\boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x}') = \sum_{j=1}^{\infty} \frac{\lambda_j}{b_j^2} b_j\phi_j(\boldsymbol{x}) b_j\phi_j(\boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}') \ .$$

A dual representation of the solution vector $\boldsymbol{w}$ found by the learning machine leads to a

decision rule $f(\boldsymbol{x})$ written in terms of the values of the kernel function on pairs consisting of the training points $\boldsymbol{x}_i$ and the test point $\boldsymbol{x}$

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i) \cdot \boldsymbol{\phi}(\boldsymbol{x}) + b = \sum_{i=1}^{l} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b \ .$$

If we define a vector $\boldsymbol{\psi}$ living in $\mathcal{H}$ to be a linear combination of $\boldsymbol{\phi}(\boldsymbol{x}_i), i = 1, \ldots, l$

$$\boldsymbol{\psi} = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{\phi}(\boldsymbol{x}_i) \tag{1.8}$$

the decision rule $f(\boldsymbol{x})$ can be equivalently expressed in the primal form

$$f(\boldsymbol{x}) = \sum_{i=1}^{\infty} \lambda_i \psi_i \phi_i(\boldsymbol{x}) + b$$

with (1.8) linking the primal with the dual representation. We should remark here that the immediately preceding expression of $f(\boldsymbol{x})$ which involves explicit mappings of the training and test points in the feature space requires the summation of as many terms as the dimensionality of the feature space as opposed to the dual representation which contains only l terms. In order to choose which of the two representations suits us most we have to weigh the size of the dataset against the dimensionality of the feature space. As we discussed earlier there are situations where the infinite sum in the above equation may be truncated without a significant error.

We conclude this section by pointing out that there is not a unique mapping of the data that leads to a given kernel. Mappings that are associated with different feature spaces even in terms of their dimensionality can result in the same kernel.

1.5 Examples of Kernels

The procedure described in Section 1.3 for constructing kernels associated with embeddings to feature spaces consisting of all the monomials of degree 2 can be extended to kernels representing the inner product of vectors the components of which are monomials of arbitrary degree d. Let us consider the dth order product $(x_{j_1} x_{j_2} \ldots x_{j_d})$ constructed by attributes of the point $\boldsymbol{x}$ in which the indices $j_1, j_2, \ldots, j_d$ run over $1, \ldots, n$, where n is the dimensionality of the original space. Next we form a vector $\boldsymbol{\phi}(\boldsymbol{x})$ the attributes of which are all the dth order products which result after exhausting the combinations on the values of the indices $j_1, \ldots, j_d$. We carry out the inner product $\boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x}')$ which yields

$$\boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x}') = \sum_{j_1} \cdots \sum_{j_d} x_{j_1} x_{j_2} \ldots x_{j_d} x'_{j_1} x'_{j_2} \ldots x'_{j_d}$$

$$= \sum_{j_1} x_{j_1} x'_{j_1} \cdots \sum_{j_d} x_{j_d} x'_{j_d} = \left(\sum_{j} x_j x'_j \right)^d = \left(\boldsymbol{x} \cdot \boldsymbol{x}' \right)^d \ .$$

This means that the inner product of vectors under a mapping which transfers them to a space formed by all the monomials of degree d corresponds to the kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} \cdot \boldsymbol{x}')^d \ .$$

Since the surface that classifies the examples into two categories is a polynomial curve we call this kind of kernels polynomial kernels. For $d = 1$ we obtain a special case of a polynomial kernel which is the linear kernel $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x} \cdot \boldsymbol{x}'$. We have seen before that a space the features of which are monomials up to degree 2 is endowed with the inhomogeneous polynomial kernel $K(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} \cdot \boldsymbol{x}' + c)^2$. Generalising this result to a space consisting of all the monomials up to degree d we end up with the following kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} \cdot \boldsymbol{x}' + c)^d \ .$$

Apart from the polynomial kernels it is worth mentioning the general category of Radial Basis Function (RBF) kernels. Their characteristic is that they are written in the form

$$K(\boldsymbol{x}, \boldsymbol{x}') = g(d(\boldsymbol{x}, \boldsymbol{x}')) \ ,$$

where g denotes a function taking as an argument a metric d on X and mapping it to the space of nonnegative real numbers. The most common metric applied to the points $\boldsymbol{x}$ and $\boldsymbol{x}'$ is the Euclidean distance $d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\| = \sqrt{(\boldsymbol{x} - \boldsymbol{x}') \cdot (\boldsymbol{x} - \boldsymbol{x}')}$. A known kernel falling into this category is the Gaussian kernel [2]

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{\sigma^2} \right) \ ,$$

where σ is a positive parameter. Since the value of the Gaussian kernel depends only on the vectorial difference of 2 points it has the property of being translation invariant implying that $K(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x} + \boldsymbol{x}_0, \boldsymbol{x}' + \boldsymbol{x}_0)$. All the above mentioned kernels share the important property of being invariant under orthogonal transformations. Thus, if a linear mapping of the form $\boldsymbol{x} \rightarrow \boldsymbol{O}\boldsymbol{x}$ takes place, where $\boldsymbol{O}$ is an orthogonal matrix for which $\boldsymbol{O}^{-1} = \boldsymbol{O}^T$ holds, then $K(\boldsymbol{x}, \boldsymbol{x}')= K(\boldsymbol{O}\boldsymbol{x}, \boldsymbol{O}\boldsymbol{x}')$. This property obviously holds for the polynomial kernels since they involve inner products but also for the Gaussian kernel for which the Euclidean distance is employed. Such a matrix $\boldsymbol{O}$ includes the case of a rotation matrix. Consequently, if a rotation of the points is performed and either a polynomial or a Gaussian kernel is employed with the additional assumption that the

learning machine makes use of the training instances exclusively through a kernel we will acquire the same solution as if no rotation was applied to the data. This property renders the results of the learning procedure independent of the coordinate system employed as long as the origin is kept fixed. An extensive treatment of kernels is provided in [48, 54].

Chapter 2

Elements of Statistical Learning Theory

2.1 The General Inference Model

Pattern recognition can be viewed as a problem of extracting knowledge from empirical data. The main approach until the 60's for estimating functions using experimental data was the parametric inference model which is based on the assumption that the unknown functions can be appropriately parametrised. Then the experimental data are used in order to determine the unknown parameters entering the model. Fisher [16] was one of the pioneers in this direction who suggested the maximum likelihood approach as a method of solving problems cast in this form. The original belief that these methods could prove successful in real-world problems originated from the Weierstrass approximation theorem stating that any continuous function defined in a finite interval can be approximated to any level of accuracy by polynomials of an appropriate order. This belief was further reinforced by the fact that whenever an outcome is the result of a large number of interacting independent random factors the distribution underlying their sum can be described satisfactorily by the normal law according to the Central Limit Theorem.

The drawback of the parametric inference model, however, is that its efficiency is strongly dependent on the dimensionality of the space which the data live in. In particular such methods break down in cases involving high-dimensional data spaces. In such spaces the existence of singularities in the function to be approximated is very probable. The existence of only a small number of derivatives for such non-smooth functions demands polynomials of degree increasing with the number of dimensions, if polynomial approximator functions are used, in order to achieve an acceptable level of accuracy. This phenomenon is characterised as "the curse of dimensionality".

In the opposite direction lies the general analysis of statistical inference initiated by Glivenko, Cantelli and Kolmogorov. Glivenko and Cantelli proved that if one uses a very large number of independent observations which follow some distribution then, independently of the actual probability distribution governing the data, one can approximate it to any desired degree of accuracy. Kolmogorov's contribution was the establishment of bounds concerning the rate of convergence to the actual distribution. This theory which makes no assumption about the underlying distribution was called the general inference model as opposed to the particular (parametric) one.

Both approaches have the common goal of finding the function that describes well unseen data by exploiting only a finite data sample. A learning machine that takes as input this sample of data and produces a function able to explain well unseen data is said to have good generalisation ability.

By generalising the Glivenko-Cantelli-Kolmogorov results a theory was developed [60, 61] which links the training procedure of the learning machine involving a finite training sample with its ability to produce rules that work well on examples to be presented to it in the future. According to this theory the only relevant quantity related to the training process is the number of examples that the machine failed to explain based on the rule that has been generated. These wrongly explained examples are characterised as training errors. Central to this theory is the Empirical Risk Minimisation (ERM) principle which associates the small number of training errors with the good generalisation ability of the learning machine.

There are two main issues that this theory should address. The first involves the identification of necessary and sufficient conditions under which the ERM principle is consistent. Consistency of the ERM principle means that the estimated function is able to approach the best possible solution within a given set of functions with an increasing number of observations. The second issue involves quantitative criteria associated with the solution found. For example, we would like to know what the probability of error on unknown examples is when we have already estimated the function that minimises the training errors in a dataset of a given size and how much this probability differs from the one associated with the optimal choice among functions belonging to a given set.

The law of large numbers as established by Glivenko and Cantelli states that the frequency of an event converges to its true probability for a very large number of observations. However, in its original form this law could not assert that for a given set of events the sequence of probabilities of events with the smallest frequency converges to the smallest possible value for this set. This theoretical gap was filled later by the uniform law of large numbers. A prominent role in this theory is played by the Vapnik-Chervonenkis (VC) dimension, which measures the richness of the class of functions implemented by the learning machine. According to this theory consistency in distribution-independent settings exists if the VC dimension is finite. The extension of Kolmogorov's bounds on

the rate of convergence led to bounds which depend on the VC dimension, the number of training errors and the size of the dataset presented to the machine during the training phase.

As the ERM principle suggests one should be interested in estimating a function that makes a small number of errors on the training set because this would automatically imply a small error rate on unseen data (test error). It is expected that if the learning machine employed has at its disposal a large set of candidate functions then it becomes easier to find the function that leads to the smallest possible training error. There exist bounds that constrain the probability of error simultaneously for all the functions in the set and involve quantities like the number of training errors and the VC dimension of the set of functions. A larger VC dimension implies a greater ability of the functions contained in the set to explain the data leading possibly to smaller training errors. The presence of the VC dimension in the bounds can influence the minimum number of errors on unseen data (i.e. the risk suffered). This is an interesting property of the bounds which will be investigated at a later stage of our analysis.

2.2 Learning from Examples

The learning model that will be considered consists of three main parts:

1. The generator of the training instances.
2. The target function (supervisor) operating on the training instances.
3. The learning machine.

The generator is thought of as being a source the statistical properties of which remain invariant during the procedure. Its role is to generate a number l of instances $\boldsymbol{x}$ belonging to an n-dimensional space X which are independently drawn and identically distributed (i.i.d.) according to some unknown distribution. The target function (supervisor), which is unknown to us as well and remains the same as long as the training takes place, receives these instances as input and produces an output y belonging to a space Y which is visible to us. The learning machine tries to estimate the target function employing as the only source of knowledge the instance-target pairs

$$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_l, y_l) \ .$$

Based on these observations the machine attempts to construct a function that predicts well the response of the target function when presented with samples coming from the same distribution. At any stage of the procedure the machine can provide us with an estimation of what the answer of the target function or oracle will be on the next instance.

Nevertheless, one should distinguish the case of constructing a function performing well on the data from the case of ending up with a function lying close to the target function with respect to some metric. Obviously, the latter is a stronger requirement which subsumes the case of imitating the behaviour of the oracle.

2.3 Minimising the Risk Functional

In the previous section we saw that during the training phase the learning machine constructs a function which operates on the data. As a matter of fact the machine chooses the appropriate member from a class of functions $H \subset Y^X$ referred to as the hypothesis class. Y^X denotes the set of all functions which map the input space X onto the output space Y. The selection of the function $f(\boldsymbol{x})$, which is called a hypothesis, can be done according to some predetermined criteria. To make this more formal we define the following functional

$$R = R(f(\boldsymbol{x})) \ ,$$

which depends on the set of admissible functions $f(\boldsymbol{x})$. Among these the function $f_0(\boldsymbol{x})$ has to be found which minimises R which for this reason will be referred to as the risk functional. If we consider that the samples are generated according to a probability distribution $F(\boldsymbol{x}, y)$ then the risk functional can be expressed as the expectation

$$R(f(\boldsymbol{x})) = \int L(y, f(\boldsymbol{x}))dF(\boldsymbol{x}, y) \ .$$

As we have mentioned before the probability distribution is unknown and minimisation can be performed only by employing the empirical data available during the training process. If we call $\mathcal{A}$ the algorithm implemented by the learning machine then for a given sample $\boldsymbol{z} = (\boldsymbol{x}, y)$ and a hypothesis class H we assume that $\mathcal{A}$ produces hypotheses belonging to H, coded formally as $\mathcal{A}(\boldsymbol{z}, H) \in H$.

The function $L(y, f(\boldsymbol{x}))$ appearing in the integral is known as the loss function and is a measure of the discrepancy between the output produced by f and the target. Additionally, we have to impose that the loss function be integrable for any $f(\boldsymbol{x}) \in H$. This loss function can be parametrised by $a \in \Lambda$ allowing us to distinguish the admissible functions belonging to the same hypothesis class. Notice that the set Λ which the parameter a takes values from is connected to the admissible set of functions. In other words we establish a correspondence $a \rightarrow f_a$ between elements a of the set Λ and functions $f_a \in H$. Using $Q(\boldsymbol{z}, a)$ instead of $L(y, f(\boldsymbol{x}))$ the risk functional $R(f_a)$ becomes the function $R(a)$

$$R(a) = \int Q(\boldsymbol{z}, a)dF(\boldsymbol{z}) \ . \tag{2.1}$$

The objective of the algorithm is to infer the specific $a = a_0$ for which $R(a)$ is minimised.

In the literature we come across three basic problems, namely the problem of classification, the problem of regression and finally that of density estimation. In the simplest problem, that of classification, an instance is generated according to $F(\boldsymbol{x})$. The supervisor classifies the new instance to one of k classes according to the conditional probability $F(\omega|\boldsymbol{x})$, where $\omega \in \{1, 2, \ldots, k\}$. For the special case of binary classification the number of classes is fixed to two. In the regression problems there is a functional relationship or more generally a stochastic dependency linking each $\boldsymbol{x}$ to a scalar y which takes values in the range $(-\infty, \infty)$. This dependency is described by the probability $F(y|\boldsymbol{x})$ of y given $\boldsymbol{x}$. In the problem of density estimation we seek to determine the probability density $p(\boldsymbol{x}, a_0)$ among the set of admissible densities $p(\boldsymbol{x}, a), a \in \Lambda$ that corresponds to the unknown distribution $F(\boldsymbol{x})$ which generated the observed instances $\boldsymbol{x}$. The difference from the cases of classification and regression is that the supervisor providing the values of y is missing, so $\boldsymbol{z}$ coincides with $\boldsymbol{x}$.

Various loss functions have been proposed depending on the nature of the problem. For example, for binary classification problems the 0-1 loss has been adopted which is described by

$$L_{0-1}(\omega, \phi(\boldsymbol{x}, a)) = \begin{cases} 0 \text{ if } \omega = \phi(\boldsymbol{x}, a) \\ 1 \text{ if } \omega \neq \phi(\boldsymbol{x}, a) \end{cases} .$$

The argument $\phi(\boldsymbol{x}, a)$ in the loss function denotes the prediction of the machine for a given $\boldsymbol{x}$ on the basis of the function selected from the hypothesis class. The loss function in the binary classification problem is a set of indicator functions that take only two values, either zero or one. For regression the squared loss is commonly used which is given by

$$L_2(y, \phi(\boldsymbol{x}, a)) = (y - \phi(\boldsymbol{x}, a))^2 \ .$$

The loss function in this case does not take values from a finite set but instead can be equal to any nonnegative number. For the density estimation problem the loss function

$$L(\boldsymbol{x}, \phi(x, a)) = -\ln p(\boldsymbol{x}, a)$$

is used which takes values in the range $(-\infty, \infty)$. Such a choice of the loss function is motivated by the fact that the corresponding risk coincides, apart from a constant, with the Kullback-Leibler (KL) distance between the approximate density $p(\boldsymbol{x}, a)$ and $p(\boldsymbol{x}, a_0)$. The minimum value of the risk $R(a_0)$ coincides with the entropy of the distribution associated with $p(\boldsymbol{x}, a_0)$.

As suggested by the ERM principle instead of minimising the risk function one can minimise the empirical risk

$$R_{\text{emp}}(a) = \frac{1}{l} \sum_{i=1}^{l} Q(\boldsymbol{z}_i, a) \ .$$

For the special case of binary classification $Q(\boldsymbol{z}_i, a)$ takes values from the set $\{0, 1\}$. Let us assume that the minimum of the risk is attained at $Q(\boldsymbol{z}, a_0)$ whereas the minimum of the empirical risk at $Q(\boldsymbol{z}, a_l)$. We will consider $Q(\boldsymbol{z}, a_l)$ as an approximation of $Q(\boldsymbol{z}, a_0)$ and try to identify the conditions under which the former converges to the latter.

It is worth pointing out here that since the machine has a restricted set of functions at its disposal the best that we can expect from the algorithm is to estimate a_0 as

$$a_0 = \arg\min_{a \in \Lambda} R(a) \ .$$

Nevertheless, the best estimate lies among the set of all possible functions Y^X mapping the space X onto Y and for that estimate the risk acquires its minimum value $R_{\min}$.

Previously in our discussion we emphasised the importance of the richness of the hypothesis class Λ described by the VC dimension as a decisive factor on which the generalisation ability of the machine depends. The choice of Λ brings forward a dilemma known as the approximation-estimation or bias-variance dilemma. In order for this to become apparent the difference of the risk due to any function corresponding to the parameter $a \in \Lambda$ and the minimum possible value of the risk $R_{\min}$ has to be decomposed as

$$R(a) - R_{\min} = (R(a) - R(a_0)) + (R(a_0) - R_{\min}) \ .$$

The second term on the r.h.s. is characterised as the approximation error. Apparently, as the set of admissible functions is enlarged the feasibility of a function being closer to the best estimate increases. The same does not apply for the first term called the estimation error. In this case a larger hypothesis class means that the risk incurred by any function in the set will be probably further away from the minimum risk incurred by a_0. Therefore, the most successful choice of Λ, that is, the one that contains a minimising the l.h.s. can only result as a trade-off between the approximation and the estimation error.

2.4 Consistency of the Learning Process

The quality of an algorithm $\mathcal{A}$ can be judged by its ability to converge to functions (hypotheses) which lead to risks lying close to the minimum of the risk function attained at $a_0 \in \Lambda$. The algorithm should succeed in that task only by means of the training sample made available to it. The construction of the hypothesis is suggested by the appropriate induction principle which in our case is ERM. Through that principle $\mathcal{A}$ produces a solution hypothesis $a_l \in \Lambda$, depending on the size l of the training set, which leads to $Q(\boldsymbol{z}, a_l)$ from which an expected risk $R(a_l)$ could have been estimated if the distribution generating the sample was known. Thus, the expected risk $R(a_l)$ has to be assessed only through the error incurred on the finite empirical sample after the

completion of the training procedure. There are two fundamental questions that can be raised. What is the relation between the empirical risk and the expected risk for the solution found? Is it possible for the expected risk to approach the smallest feasible value for functions belonging to the set Λ? These matters will be addressed by the study of the consistency of the ERM principle.

Definition 2.1. The principle of empirical risk minimisation is consistent for the set of functions $Q(\boldsymbol{z}, a), a \in \Lambda$ and for the probability distribution function $F(\boldsymbol{z})$ if the following two sequences converge in probability [1] to the same limit

$$R(a_l) \xrightarrow[P]{l\to\infty} \inf_{a\in\Lambda} R(a) \tag{2.2}$$

$$R_{\text{emp}}(a_l) \xrightarrow[P]{l\to\infty} \inf_{a\in\Lambda} R(a) \ . \tag{2.3}$$

The first of the two equations requires that in the limit of an infinite dataset size provided to the machine the sequence of achieved risks on the basis of the functions constructed tends to the minimum one. The second equation requires that in the same limit the sequence of empirical risks estimated on the set of training data given to the machine also tends to the same value.

From the Definition 2.1 it is evident that consistency is a property of the functions that the machine implements and the distribution that generates the data. We would like consistency to be achieved in terms of general criteria characterising the whole class of admissible functions and not specific members of the class. However, even a set of inconsistent functions can be rendered consistent by adding to it a function which minimises the loss $Q(\boldsymbol{z}, a)$. Then, it is easily understood that the minimum of the empirical risk is attained at this function which also coincides with the minimum of the expected risk. In order to tackle such kind of situations we have to exclude functions from the set of admissible ones that lead to trivial satisfaction of the consistency criteria. This can be done by reformulating the definition of the ERM consistency as follows.

Definition 2.2. The ERM principle is strictly (non-trivially) consistent for the set of functions $Q(\boldsymbol{z}, a)$, $a \in \Lambda$ and for the probability distribution function $F(\boldsymbol{z})$ if for any nonempty subset $\Lambda(c), c \in (-\infty, \infty)$ of this set of functions such that

$$\Lambda(c) = \left\{ a : \int Q(\boldsymbol{z}, a) dF(\boldsymbol{z}) \geq c \right\}$$

the minimum of the empirical risks defined over any such subset $\Lambda(c)$ tends to the minimum expected risk for functions of this subset

$$\inf_{a\in\Lambda(c)} R_{\text{emp}}(a) \xrightarrow[P]{l\to\infty} \inf_{a\in\Lambda(c)} R(a) \ .$$

[1] Convergence in probability means that for all $\epsilon > 0$ we have $P\{|R(\alpha_l) - \inf_{\alpha\in\Lambda} R(\alpha)| > \epsilon\} \xrightarrow{l\to\infty} 0$ and $P\{|R_{\text{emp}}(\alpha_l) - \inf_{\alpha\in\Lambda} R(\alpha)| > \epsilon\} \xrightarrow{l\to\infty} 0$, respectively.

If the ERM principle is strictly consistent then (2.2) of Definition 2.1 is automatically fulfilled.

We now give an example in which the first limit in Definition 2.1 holds whereas the second is violated. To illustrate this we consider the set of indicator functions $Q(\boldsymbol{z}, a), a \in \Lambda$. To our class belong only functions Q which take the value zero at a finite number of intervals which as a total have measure ϵ and are equal to one elsewhere. The parameter a distinguishes the functions Q according to the specific intervals at which their value is equal to zero. If a number of examples $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_l$ is supplied to a learning machine working on the basis of ERM principle it will favour the solutions a for which the functions Q become zero at the training points. Consequently, for this class of functions we have

$$R_{\text{emp}}(a_l) = \inf_{a \in \Lambda} \frac{1}{l} \sum_{i=1}^{l} Q(\boldsymbol{z}_i, a) = 0 \ .$$

The expectation of the risk for any function in the set and therefore for the function a_l constructed by the machine is given by

$$R(a) = \int Q(\boldsymbol{z}, a) dF(\boldsymbol{z}) = 1 - \epsilon \ . \tag{2.4}$$

Combining the above two equations we obtain

$$\inf_{a \in \Lambda} R(a) - R_{\text{emp}}(a_l) = 1 - \epsilon \ .$$

Furthermore, due to (2.4) it obviously holds that

$$R(a_l) - \inf_{a \in \Lambda} R(a) = \int Q(\boldsymbol{z}, a_l) dF(\boldsymbol{z}) - \inf_{a \in \Lambda} \int Q(\boldsymbol{z}, a) dF(\boldsymbol{z}) = 0 \ .$$

We conclude that although the expectation of the risk converges to the smallest possible in the set the same does not hold for the empirical risk. Therefore, for that class of functions the ERM principle is not consistent.

The ERM consistency demands some asymptotic criteria that must be met in order for the solution constructed in terms of empirical data to converge to the optimal one. Notice that the optimal solution depends on the real distribution producing the data. We can get some intuition about what is needed to satisfy the criteria by studying another example in which an induction principle somewhat different from the ERM one is used.

Once more we are interested in finding out whether the risk function can be optimised (minimised) through the use of empirical data, but we now make the additional assumption that the distribution function is absolutely continuous. In that case the risk

function can be rewritten as follows

$$R(a) = \int Q(\boldsymbol{z}, a) dF(\boldsymbol{z}) = \int Q(\boldsymbol{z}, a) p(\boldsymbol{z}) d\boldsymbol{z} \ ,$$

where $p(\boldsymbol{z})$ is the density function corresponding to the distribution $F(\boldsymbol{z})$. Furthermore, we demand that the loss function be uniformly bounded by a positive constant B ($|Q(\boldsymbol{z}, a)| \leq B$). This is true in the binary classification since $Q(\boldsymbol{z}, a)$ are indicator functions and take the value of either 1 or 0. Let us also assume that the density $p_l(\boldsymbol{z})$ estimated from the data converges in probability to the true density $p(\boldsymbol{z})$ in the L_1 metric, i.e.

$$\int |p_l(\boldsymbol{z}) - p(\boldsymbol{z})| \, d\boldsymbol{z} \overset{l \to \infty}{\underset{P}{\longrightarrow}} 0 \ .$$

We also consider the empirical risk $R^\star_{\text{emp}}$

$$R^\star_{\text{emp}}(a) = \int Q(\boldsymbol{z}, a) p_l(\boldsymbol{z}) d\boldsymbol{z} \tag{2.5}$$

associated with an induction principle different from the usual one. We would like to emphasise that the above modified empirical risk is generally expected to be different from R_{emp} since the empirical density p_l is not necessarily concentrated on the observed data but may assume non-zero values elsewhere. As a result the integral in (2.5) does not always reduce to a finite sum. With these assumptions and the requirement in mind that Q be bounded we will prove that the function $Q(\boldsymbol{z}, a_l)$ which minimises $R^\star_{\text{emp}}(a)$ defined in terms of the estimator p_l converges to the optimal one $Q(\boldsymbol{z}, a_0)$ among all functions minimising $R(a)$. Constructing the supremum over all the functions in the class we obtain

$$\sup_{a \in \Lambda} \left| \int Q(\boldsymbol{z}, a) p(\boldsymbol{z}) d\boldsymbol{z} - \int Q(\boldsymbol{z}, a) p_l(\boldsymbol{z}) d\boldsymbol{z} \right| \leq \sup_{a \in \Lambda} \int |Q(\boldsymbol{z}, a)| \, |p(\boldsymbol{z}) - p_l(\boldsymbol{z})| \, d\boldsymbol{z}$$

$$\leq B \int |p(\boldsymbol{z}) - p_l(\boldsymbol{z})| \, d\boldsymbol{z} \overset{l \to \infty}{\underset{P}{\longrightarrow}} 0 \ . \tag{2.6}$$

Thus, with the expectation taken over the actual distribution the expected risk converges in probability for all functions in the class simultaneously to the risk with the expectation now taken over the empirical probability estimator. From the previous relationship it can be derived that for any ϵ and η there exists a number of examples $l'(\epsilon, \eta)$ such that for any $l > l'(\epsilon, \eta)$ with probability $1 - \eta$

$$\int Q(\boldsymbol{z}, a_l) p(\boldsymbol{z}) d\boldsymbol{z} - \int Q(\boldsymbol{z}, a_l) p_l(\boldsymbol{z}) d\boldsymbol{z} < \epsilon \ , \tag{2.7}$$

$$- \int Q(\boldsymbol{z}, a_0) p(\boldsymbol{z}) d\boldsymbol{z} + \int Q(\boldsymbol{z}, a_0) p_l(\boldsymbol{z}) d\boldsymbol{z} < \epsilon \ . \tag{2.8}$$

hold. Since $Q(\boldsymbol{z}, a_l)$ is the minimiser of $R^{\star}_{\text{emp}}$ the following is true

$$\int Q(\boldsymbol{z}, a_l) p_l(\boldsymbol{z}) d\boldsymbol{z} \leq \int Q(\boldsymbol{z}, a_0) p_l(\boldsymbol{z}) d\boldsymbol{z} \ . \tag{2.9}$$

Combining (2.7), (2.8) and (2.9) we obtain that with probability $1 - \eta$

$$\int Q(\boldsymbol{z}, a_l) dF(\boldsymbol{z}) - \int Q(\boldsymbol{z}, a_0) dF(\boldsymbol{z}) < 2\epsilon \tag{2.10}$$

holds true. Summing up, we can assert that the minimisation of the induction principle (2.5) under the condition of boundedness of $Q(\boldsymbol{z}, a)$ and the convergence of densities guarantees that the risk estimated with $Q(\boldsymbol{z}, a_l)$ converges in probability to the smallest possible in the limit of infinite data sample size. From (2.6) it is apparent that we do not need the empirical density to converge to the true one, but it suffices instead that

$$\sup_{a \in \Lambda} \left| \int Q(\boldsymbol{z}, a) p(\boldsymbol{z}) d\boldsymbol{z} - \int Q(\boldsymbol{z}, a) p_l(\boldsymbol{z}) d\boldsymbol{z} \right| \overset{l \to \infty}{\underset{P}{\longrightarrow}} 0 \ .$$

If we are interested only in the binary classification, $Q(\boldsymbol{z}, a)$ are indicator functions and the integral $\int Q(\boldsymbol{z}, a) p(\boldsymbol{z}) d\boldsymbol{z}$ can be interpreted as the probability $P(A_a)$ of the event $A_a = \{\boldsymbol{z} : Q(\boldsymbol{z}, a) = 1\}$ with respect to the distribution density $p(\boldsymbol{z})$. In an analogous way the integral $\int Q(\boldsymbol{z}, a) p_l(\boldsymbol{z}) d\boldsymbol{z}$ can be interpreted as the probability $Q(A_a)$ of the same event A_a only this time with respect to the empirical density p_l. The following theorem due to Scheffe relates the difference of densities in L_1 metric to the supremum of the difference of the corresponding probabilities:

Theorem 2.3. *Let $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ be densities, let $\mathcal{F}$ be Borel sets of events A and let $P(A) = \int_A p(\boldsymbol{x}) d\boldsymbol{x}$ and $Q(A) = \int_A q(\boldsymbol{x}) d\boldsymbol{x}$ be probabilities of the set $A \in \mathcal{F}$ corresponding to these densities. Then*

$$\sup_{A \in \mathcal{F}} |P(A) - Q(A)| \leq 1/2 \int_{\boldsymbol{x}} |p(\boldsymbol{x}) - q(\boldsymbol{x})| \, d\boldsymbol{x} \ .$$

An immediate consequence of the theorem is that convergence of densities in the L_1 metric leads to convergence over the set of all events of the corresponding probabilities. This implies that convergence of the densities is a stronger condition to impose than convergence of the probabilities, especially if one is interested in the convergence over a subset $F^{\star}$ of the set F of events. Therefore, one should move in the direction of identifying conditions under which this holds true. At this point we should note that apart from asymptotic bounds one can acquire non-asymptotic ones as that of (2.10) on the generalisation of the learning machine depending only on a finite number of observations.

If the empirical risk minimisation principle is adopted instead of the one of (2.5) one can examine by means of the Lebesgue integral in an analogous procedure as earlier whether

the minimal value of the risk function (2.1) can be attained. For this to take place we require that the loss functions be bounded and non-negative and furthermore that the empirical risk estimator $\nu_l(A)$ of an event A converge in probability to the corresponding true probability $P(A)$

$$\sup_{A \in \mathcal{F}^\star} |P(A) - \nu_l(A)| \overset{l \to \infty}{\underset{P}{\longrightarrow}} 0 \ .$$

Notice here that convergence holds for all the events belonging to a subset $\mathcal{F}^\star$ of $\mathcal{F}$. Indeed, it can be proved that minimisation of the empirical risk produces a sequence of values $R(a_l)$ that tend in probability to the minimal value of the risk for increasing numbers of observations.

2.5 Empirical Processes

From what has preceded a connection was revealed between the consistency of the empirical risk minimisation principle and the convergence of the empirical risk estimator of any event in a set to its true probability. We consider the sequence of random variables

$$\xi^l = \sup_{a \in \Lambda} \left| \int Q(\boldsymbol{z}, a) dF(\boldsymbol{z}) - \frac{1}{l} \sum_{i=1}^{l} Q(\boldsymbol{z}_i, a) \right| \ .$$

Here, we make again the hypothesis that a number of examples l were independently generated by the same distribution $F(\boldsymbol{z})$. We call this sequence which depends both on $F(\boldsymbol{z})$ and on the loss function $Q(\boldsymbol{z}, a)$ a two-sided empirical process. As we remarked in the beginning of the section the relation to the consistency of ERM principle motivates us to investigate the conditions under which the empirical process converges in probability to zero, that is

$$P \left\{ \sup_{a \in \Lambda} \left| \int Q(\boldsymbol{z}, a) dF(\boldsymbol{z}) - \frac{1}{l} \sum_{i=1}^{l} Q(\boldsymbol{z}_i, a) \right| > \epsilon \right\} \overset{l \to \infty}{\longrightarrow} 0 \ . \tag{2.11}$$

The above relation describes the uniform convergence of means to their expectations. It is called uniform because it is taken over the set of all admissible functions. One can also consider the one-sided empirical process defined as

$$\xi_+^l = \sup_{a \in \Lambda} \left(\int Q(\boldsymbol{z}, a) dF(\boldsymbol{z}) - \frac{1}{l} \sum_{i=1}^{l} Q(\boldsymbol{z}_i, a) \right)_+ \ ,$$

where

$$(u)_+ = \begin{cases} u \text{ if } u > 0 \\ 0 \ \text{ otherwise} \end{cases} \ .$$

In analogy to the uniform two-sided convergence the uniform one-sided one takes place if

$$P\left\{\sup_{a\in\Lambda}\left(\int Q(\boldsymbol{z},a)dF(\boldsymbol{z})-\frac{1}{l}\sum_{i=1}^{l}Q(\boldsymbol{z}_i,a)\right)>\epsilon\right\}\overset{l\to\infty}{\longrightarrow}0\ . \tag{2.12}$$

If the set of $Q(\boldsymbol{z},a)$, $a\in\Lambda$ is chosen to be the set of indicator functions then the relations (2.11) and (2.12) are interpreted as uniform convergence of frequencies to their probabilities.

According to the law of large numbers if the set of admissible functions contains only one element then the sequence of means converges always to the expectation of the random variable for an increasing number of examples. Specialising this to the binary classification which we are interested in we consider a hypothesis class containing a single indicator function $Q(\boldsymbol{z},a^\star)$ with $a^\star$ denoting here a fixed event. The Bernoulli theorem verifies then that for $l\to\infty$ the frequencies converge to the probability of the event $A_{a^\star}=\{\boldsymbol{z}:Q(\boldsymbol{z},a^\star)>0\}$

$$P\left\{|P\{Q(\boldsymbol{z},a^\star)>0\}-\nu_l\{Q(\boldsymbol{z},a^\star)>0\}|>\epsilon\right\}\overset{l\to\infty}{\longrightarrow}0$$

where $\nu_l\{Q(\boldsymbol{z},a^\star)>0\}$ is used to denote the frequency of the event. In addition to the previous asymptotic bound Chernoff's inequality [12]

$$P\left\{|P\{Q(\boldsymbol{z},a^\star)>0\}-\nu_l\{Q(\boldsymbol{z},a^\star)>0\}|>\epsilon\right\}<2\exp\{-2\epsilon^2 l\}$$

yields bounds on the rate of convergence for a finite number of observations. The law of large numbers can be easily generalised to the case where the hypothesis class consists only of a finite number of functions N. Let $A_k=\{\boldsymbol{z}:Q(\boldsymbol{z},a_k)>0\},\ k=1,2\ldots N$ denote the set of N finite events. The uniform two-sided empirical process for finite events in the Bernoulli scheme is written as

$$\max_{1\le k\le N}|P\{Q(\boldsymbol{z},a_k)>0\}-\nu_l\{Q(\boldsymbol{z},a_k)>0\}|\ .$$

We can place upper bounds on the rate of convergence by application of Chernoff's inequality which gives that

$$\begin{aligned}
&P\left\{\max_{1\le k\le N}|P\{Q(\boldsymbol{z},a_k)>0\}-\nu_l\{Q(\boldsymbol{z},a_k)>0\}|>\epsilon\right\}\\
&\le\sum_{k=1}^{N}P\left\{|P\{Q(\boldsymbol{z},a_k)>0\}-\nu_l\{Q(\boldsymbol{z},a_k)>0\}|>\epsilon\right\}\\
&\le 2N\exp\{-2\epsilon^2 l\}=2\exp\left\{\left(\frac{\ln N}{l}-2\epsilon^2\right)l\right\}\ .
\end{aligned} \tag{2.13}$$

In order to derive convergence from (2.13) for any value of the parameter ϵ

$$\lim_{l\to\infty} \frac{\ln N}{l} \to 0$$

needs to hold. This is true for a finite number of elements N in the hypothesis class.

The difficulties arise when the set $Q(\boldsymbol{z}, a), a \in \Lambda$ includes an infinite number of elements which is the case if a parametrises a functional space. The generalisation of the law of large numbers in the functional space is essentially the uniform convergence of the means to their expectations over the whole set of functions distinguished by the continuous variation of the parameter a. The discussion in the previous section suggested that the condition of uniform convergence is sufficient to guarantee the consistency of the ERM principle. The role of uniform convergence is to enforce the empirical risk to be close to the expected risk uniformly over all the functions in the set for a sufficiently large number of examples.

2.6 The Key Theorem of Learning Theory

In our earlier discussion it turned out that under the assumption of uniform two-sided convergence we can achieve the minimisation of the risk function through the minimisation of the empirical risk. The key theorem [65] introduces the uniform one-sided convergence as not only a sufficient but also as a necessary condition for the strict consistency of ERM.

Theorem 2.4. *Let $Q(\boldsymbol{z}, a), a \in \Lambda$, be a set of functions that satisfy the condition*

$$A \leq \int Q(\boldsymbol{z}, a) dF(\boldsymbol{z}) \leq B \ .$$

Then for the ERM principle to be consistent, it is necessary and sufficient that the empirical risk $R_{\text{emp}}(a)$ converge uniformly to the actual risk $R(a)$ over the set $Q(\boldsymbol{z}, a),\ a \in \Lambda$, in the following sense:

$$\lim_{l\to\infty} P\left\{\sup_{a\in\Lambda}(R(a) - R_{\text{emp}}(a)) > \epsilon\right\} = 0, \ \ \forall\epsilon > 0 \ .$$

We now turn to a discussion linking the one-sided with the two-sided convergence. In an attempt to reveal the relation we decompose the two-sided uniform convergence into the following two relations

$$\lim_{l\to\infty} P\left\{\sup_{a\in\Lambda}(R(a) - R_{\text{emp}}(a)) > \epsilon\right\} = 0 \tag{2.14}$$

or

$$\lim_{l\to\infty} P\left\{\sup_{a\in\Lambda}(R_{\text{emp}}(a) - R(a)) > \epsilon\right\} = 0 \ . \tag{2.15}$$

From this decomposition we recognise the one-sided convergence as the one of the two relations that must be satisfied for the validity of the two-sided convergence. It is possible that the two-sided convergence does not hold because the part (2.15) is violated but part (2.14) is valid leaving thus unaffected the one-sided convergence. Therefore the conclusion is that two-sided convergence is only a sufficient condition for ERM consistency. We can remark here that since we are interested only in the minimisation of the empirical risk the (2.15) part of the two-sided convergence which corresponds to the maximisation of the empirical risk can be violated.

2.7 Entropy and Other Related Concepts

The law of large numbers is valid in the case of a single event and can be readily generalised for a finite number of events. For the uniform law of large numbers to be applicable to functional spaces other tools are needed which will be constructed here. The idea for proving convergence in the case when the hypothesis class H comprises infinitely many events is based on the fact that not all of the events are distinguishable for a given sample $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_l$. Two events are treated as different if at least one example in the sample which belongs to one event does not belong to the other. The number of distinguishable events, which are realised by equal in number effective functions selected from the hypothesis class, depends both on the sample size l and on the form of the functions in H and will be denoted by $N^{\Lambda}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l)$.

In what follows we will be concerned only with indicator functions $Q(\boldsymbol{z}, a)$. Let us take a sequence of vectors $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_l$ produced independently by the same distribution. We construct the following l-dimensional binary vector

$$q(a) = (Q(\boldsymbol{z}_1, a), Q(\boldsymbol{z}_2, a) \ldots Q(\boldsymbol{z}_l, a)), \quad a \in \Lambda$$

the components of which are the values of the loss functions at the points $\boldsymbol{z}_k$, $k = 1, 2, \ldots, l$ in the sequence. For a fixed choice of the parameter $a = a^{\star}$, $q(a^{\star})$ determines one of the vertices of an l-dimensional unit cube. For the number $N^{\Lambda}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l)$ of such vertices it holds that

$$N^{\Lambda}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l) \leq 2^l \ .$$

The quantity 2^l constitutes an upper bound on the number of different labellings that can be given to the sample $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_l$ by means of indicator functions. We define

$$H^{\Lambda}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l) = \ln N^{\Lambda}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l)$$

to be the random entropy of the set of indicator functions $Q(\boldsymbol{z}, a)$ and its expected value with respect to $F(\boldsymbol{z})$

$$H^{\Lambda}(l) = EH^{\Lambda}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l) = \int H^{\Lambda}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l) dF(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l)$$

to be the VC entropy of this set. The condition for ERM consistency turns out to be a condition on the entropy due to the following theorem [60, 61]:

Theorem 2.5. *Under some conditions of measurability on the set of indicator functions* $Q(\boldsymbol{z}, a)$, $a \in \Lambda$, *a necessary and sufficient condition for uniform two-sided convergence is*

$$\frac{H^{\Lambda}(l)}{l} \stackrel{l \to \infty}{\longrightarrow} 0 \ . \tag{2.16}$$

We observe a similarity with the simplest model of finite events discussed in Section 2.5 where the factor $\frac{\ln N}{l}$ is substituted by $\frac{H^{\Lambda}(l)}{l}$ which measures the rate at which the expectation of the diversity of effective functions increases with the number of examples l. In analogy with the simplest model we need the entropy to grow at most sublinearly with l in order to ensure uniform convergence.

We will proceed to the construction of two new concepts besides the VC entropy defined over the data sample and the set of indicator functions $Q(\boldsymbol{z}, a)$. The first of them is the annealed entropy given by

$$H^{\Lambda}_{\text{ann}}(l) = \ln E N^{\Lambda}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l) \ .$$

The second concept is the growth function[2] described by

$$G^{\Lambda}(l) = \ln \sup_{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l} N^{\Lambda}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l) \ .$$

The following inequalities hold between the VC entropy, the annealed entropy and the growth function

$$H^{\Lambda}(l) \leq H^{\Lambda}_{\text{ann}}(l) \leq G^{\Lambda}(l) \ . \tag{2.17}$$

Notice that the annealed entropy is an upper bound on the VC entropy due to the concavity of the logarithmic function.

Equation (2.16) is an asymptotic condition on the VC entropy and does not say anything about the rate of convergence. Two-sided uniform convergence alone cannot guarantee a respectable rate of covergence of the obtained risks $R(a_l)$ to the minimal one $R(a_0)$ which for some function classes can be very slow. We say that the asymptotic rate is fast if for any $l > l_0$ the probability of the obtained risk to be larger than the minimum

[2]Other authors define $\sup_{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l} N^{\Lambda}(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l)$ as the growth function.

one by an amount ϵ decreases exponentially as the number of examples grows

$$P\left\{R(a_l) - \inf_{a\in\Lambda} R(a) > \epsilon\right\} < \exp\{-c\epsilon^2 l\} \ .$$

It is possible to get some information on how fast the risk $R(a_l)$ approaches the minimum one by use of the annealed entropy. Indeed, the condition

$$\lim_{l\to\infty} \frac{H^{\Lambda}_{\text{ann}}(l)}{l} = 0 \tag{2.18}$$

is sufficient for a fast asymptotic rate of convergence.

A third condition on the basis of the growth function constitutes a necessary and sufficient condition for the consistency of ERM for any distribution $F(\boldsymbol{z})$ and furthermore guarantees the fast asymptotic rate. The condition is described by the equation below

$$\lim_{l\to\infty} \frac{G^{\Lambda}(l)}{l} = 0 \ . \tag{2.19}$$

The growth function is the only measure of the representation ability of the functions in a hypothesis class among the three ones discussed above which is independent of the underlying distribution since it does not involve expectations like the VC entropy and the annealed entropy do. The validity of (2.18) on the one hand and the violation of (2.19) on the other implies the existence of distributions $F(\boldsymbol{z})$ for which uniform convergence does not hold.

2.8 Bounds on the Rate of Convergence

In what follows we are going to construct non-asymptotic bounds on the risk [60, 61, 67] achieved by the machine in terms of the concepts introduced in the previous section. The bounds based on the annealed entropy are distribution-dependent but they can also be expressed in terms of the growth function which makes them independent of the distribution. For the set of indicator functions we have:

Theorem 2.6. *The inequality*

$$P\left\{\sup_{a\in\Lambda}\left|\int Q(\boldsymbol{z},a)dF(\boldsymbol{z}) - \frac{1}{l}\sum_{i=1}^{l} Q(\boldsymbol{z}_i,a)\right| > \epsilon\right\} \le 4\exp\left\{\left(\frac{H^{\Lambda}_{\text{ann}}(2l)}{l} - \left(\epsilon - \frac{1}{l}\right)^2\right) l\right\} \tag{2.20}$$

holds true.

Proof Sketch. In order to prove the above inequality we will double the size of our sample of independent and identical observations

$$\boldsymbol{Z}^{2l} = \boldsymbol{z}_1, \ldots, \boldsymbol{z}_l, \boldsymbol{z}_{l+1}, \ldots, \boldsymbol{z}_{2l} \ .$$

For the set of indicator functions the empirical risk represents the frequency of the event $\{\boldsymbol{z} : Q(\boldsymbol{z}, a) > 0\}$. Next we will define the random variables

$$\rho^{\Lambda}(\boldsymbol{Z}^{2l}) = \sup_{a \in \Lambda} \left| \frac{1}{l} \sum_{i=1}^{l} Q(\boldsymbol{z}_i, a) - \frac{1}{l} \sum_{i=l+1}^{2l} Q(\boldsymbol{z}_i, a) \right|$$

and also

$$\pi^{\Lambda}(\boldsymbol{Z}_1) = \sup_{a \in \Lambda} \left| \int Q(\boldsymbol{z}_i, a) dF(\boldsymbol{z}) - \frac{1}{l} \sum_{i=1}^{l} Q(\boldsymbol{z}_i, a) \right| \ ,$$

where $\boldsymbol{Z}_1$ denotes the first half $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_l$ of the sequence. The lemma that follows is a fundamental step towards proving Theorem 2.6.

Lemma 2.7. *The distribution of the random variable $\pi^{\Lambda}(\boldsymbol{Z}_1)$ is connected with the distribution of $\rho^{\Lambda}(\boldsymbol{Z}^{2l})$ through the inequality*

$$P\left\{\pi^{\Lambda}(\boldsymbol{Z}_1) > \epsilon\right\} \leq 2P\left\{\rho^{\Lambda}(\boldsymbol{Z}^{2l}) > \epsilon - \frac{1}{l}\right\} \ . \tag{2.21}$$

By observation of the above inequality we can assert that if one estimates a convergence rate for the r.h.s. of (2.21) that approaches 0 as the sample size l tends to infinity the same holds for the l.h.s. of (2.20). We will omit the proof of the lemma and we will continue by attempting to place an upper bound on the r.h.s. of (2.21). Initially we set $\epsilon_\star = \epsilon - \frac{1}{l}$. The r.h.s. of (2.21) can be written as

$$P\left\{\rho^{\Lambda}(\boldsymbol{Z}^{2l}) > \epsilon_\star\right\} = \int_{\boldsymbol{Z}^{2l}} \theta\left[\rho^{\Lambda}(\boldsymbol{Z}^{2l}) - \epsilon_\star\right] dF(\boldsymbol{Z}^{2l}) \ ,$$

where $\theta(u)$ denotes the step function of u that takes either the value of 1 if $u > 0$ or 0 if $u \leq 0$. In the sequel we consider all the different permutations of the sample $\boldsymbol{Z}^{2l}$ which are $(2l)!$ in total with each one of them denoted as $T_i \boldsymbol{Z}^{2l}, i = 1, \ldots, (2l)!$. Since the event $\theta[\rho - \epsilon_\star]$ in the previous relation depends on $\boldsymbol{Z}^{2l}$ and its measure is estimated on the basis of the joint distribution $F(\boldsymbol{Z}^{2l})$ we can assert that the integral is independent of the different permutations performed on the sample. Therefore we are in a position to write

$$P\left\{\rho^{\Lambda}(\boldsymbol{Z}^{2l}) > \epsilon_\star\right\} = \int_{\boldsymbol{Z}^{2l}} \frac{\sum_{i=1}^{(2l)!} \theta\left[\rho^{\Lambda}(T_i \boldsymbol{Z}^{2l}) - \epsilon_\star\right]}{(2l)!} dF(\boldsymbol{Z}^{2l}) \ . \tag{2.22}$$

Notice also that

$$\theta\left[\rho^{\Lambda}(T_i \boldsymbol{Z}^{2l}) - \epsilon_\star\right] = \theta\left[\sup_{a \in \Lambda} \left| \frac{1}{l} \sum_{i=1}^{l} Q(\boldsymbol{z}_i, a) - \frac{1}{l} \sum_{i=l+1}^{2l} Q(\boldsymbol{z}_i, a) \right| - \epsilon_\star\right]$$

$$= \sup_{a\in\Lambda} \theta\left[\left|\frac{1}{l}\sum_{i=1}^{l} Q(\boldsymbol{z}_i,a) - \frac{1}{l}\sum_{i=l+1}^{2l} Q(\boldsymbol{z}_i,a)\right| - \epsilon_\star\right] = \sup_{a\in\Lambda} \theta\left[\tilde{\rho}^{\Lambda}(T_i\boldsymbol{Z}^{2l},a) - \epsilon_\star\right] \quad . \quad (2.23)$$

Once more we should stress that although a in $Q(\boldsymbol{z},a)$ varies continuously in fact the discriminating ability of the hypothesis class is restricted. The number of a's that lead to different events $\{\boldsymbol{z} : Q(\boldsymbol{z},a) > 0\}$ depends on the type of the member functions and the sample size. Hence, with the assumption that we are dealing with effective functions finite in number we can keep one representative function $a^\star$ from each cluster containing equivalent functions and form a new hypothesis class $\Lambda^\star$. This enables us to discard the continuous variable a and put in its place $a^\star$ which takes values from a finite set of elements. Therefore, we have

$$\sup_{a\in\Lambda} \theta\left[\tilde{\rho}^{\Lambda}(T_i\boldsymbol{Z}^{2l},a) - \epsilon_\star\right] = \sup_{a^\star\in\Lambda^\star} \theta\left[\tilde{\rho}^{\Lambda}(T_i\boldsymbol{Z}^{2l},a^\star) - \epsilon_\star\right] \leq \sum_{a^\star\in\Lambda^\star} \theta\left[\tilde{\rho}^{\Lambda}(T_i\boldsymbol{Z}^{2l},a^\star) - \epsilon_\star\right] \quad . \quad (2.24)$$

By substituting (2.23) back in the integrand of (2.22) we obtain

$$\frac{\sum_{i=1}^{(2l)!} \theta\left[\rho^{\Lambda}(T_i\boldsymbol{Z}^{2l}) - \epsilon_\star\right]}{(2l)!} = \frac{\sum_{i=1}^{(2l)!} \sup_{a\in\Lambda} \theta\left[\tilde{\rho}^{\Lambda}(T_i\boldsymbol{Z}^{2l},a) - \epsilon_\star\right]}{(2l)!}$$

$$\leq \sum_{a^\star\in\Lambda^\star} \frac{\sum_{i=1}^{(2l)!} \theta\left[\tilde{\rho}^{\Lambda}(T_i\boldsymbol{Z}^{2l},a^\star) - \epsilon_\star\right]}{(2l)!} \quad . \quad (2.25)$$

In order to derive the last inequality we made use of (2.24). Each term in the sum over $a^\star$'s on the r.h.s. of (2.25) represents the ratio of the number of different orderings for which

$$\left|\frac{1}{l}\sum_{i=1}^{l} Q(\boldsymbol{z}_i,a^\star) - \frac{1}{l}\sum_{i=l+1}^{2l} Q(\boldsymbol{z}_i,a^\star)\right| > \epsilon_\star$$

is true to the total number of such orderings. Our goal in what follows is to estimate the different ways in which the mistakes are divided between the two empirical risks corresponding to the first l and the last l examples so that the absolute value of their difference is greater than $\epsilon_\star$. Let us assume that m is the total number of points $\boldsymbol{z}_j$ among $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_l,\boldsymbol{z}_{l+1},\ldots,\boldsymbol{z}_{2l}$ for which a mistake occurs with respect to the given function $a^\star$. For these $\boldsymbol{z}_j$'s $Q(\boldsymbol{z}_j,a^\star) = 1$ holds. Only some of the permutations contribute to different results since only the examples in each one of the risks for which $Q(\boldsymbol{z}_j,a^\star) = 1$ play a role and not the different orderings of these examples within each empirical risk. That means that among the $(2l)!$ permutations we have only C_{2l}^{l} leading to different results, where C_{2l}^{l} denotes the combinations of $2l$ elements taken l at a time. Thus, if we choose one representative from each such permutation the sum in the nominator of the r.h.s. of (2.25) can be expressed as the number of representative permutations times $(2l)!/C_{2l}^{l}$. Let us define sets of representative permutations that give rise to exactly k mistakes in the first l examples. The cardinality of each such set for a given k results from the combination of m mistakes taking k at a time and fixing the rest $l-k$ by choosing from

those $2l - m$ in total which are mistake-free. Thus, we obtain

$$\Gamma = \frac{\sum_{i=1}^{(2l)!} \theta\left[\tilde{\rho}^{\Lambda}(T_i \boldsymbol{Z}^{2l}, a^{\star}) - \epsilon_{\star}\right]}{(2l)!} = \sum_{k} \frac{((2l)!/C_{2l}^{l}) C_{m}^{k} C_{2l-m}^{l-k}}{(2l)!} = \sum_{k} \frac{C_{m}^{k} C_{2l-m}^{l-k}}{C_{2l}^{l}}$$

for those k that satisfy

$$\left\{k : \left| \frac{k}{l} - \frac{m-k}{l} \right| > \epsilon_{\star} \right\} \quad .$$

One can show that for Γ the following bound holds

$$\Gamma \leq 2 \exp\left\{-\epsilon_{\star}^{2} l\right\} .$$

Substituting the upper bound on Γ back in (2.25) yields

$$\sum_{a^{\star} \in \Lambda^{\star}} \frac{\sum_{i=1}^{(2l)!} \theta\left[\tilde{\rho}^{\Lambda}(T_i \boldsymbol{Z}^{2l}, a^{\star}) - \epsilon_{\star}\right]}{(2l)!} < 2 \sum_{a^{\star} \in \Lambda^{\star}} \exp\left\{-\epsilon_{\star}^{2} l\right\}$$

$$= 2 N^{\Lambda}(z_1, \ldots, z_{2l}) \exp\left\{-\epsilon_{\star}^{2} l\right\} \quad . \quad (2.26)$$

If we substitute (2.26) in (2.22) we get

$$P\left\{\rho^{\Lambda}(\boldsymbol{Z}^{2l}) > \epsilon_{\star}\right\} < 2 E N^{\Lambda}(z_1, \ldots, z_{2l}) \exp\left\{-\epsilon_{\star}^{2} l\right\} = 2 \exp\left\{\left(\frac{H_{\text{ann}}^{\Lambda}(2l)}{l} - \epsilon_{\star}^{2}\right) l\right\}$$

from which by use of Lemma 2.7 the theorem is proved. □

In order for uniform two-sided convergence to take place for any value of the parameter ϵ, i.e. (2.20) to approach 0 as $l \to \infty$, (2.18) must be satisfied. This implies due to the inequality (2.17) the validity of (2.16) which is a sufficient and necessary condition for the consistency of ERM. Since $H_{\text{ann}}^{\Lambda}(l) \leq G^{\Lambda}(l)$ the previous bound can straightforwardly be rewritten in terms of the growth function

$$P\left\{\sup_{a \in \Lambda} \left| \int Q(\boldsymbol{z}, a) dF(\boldsymbol{z}) - \frac{1}{l} \sum_{i=1}^{l} Q(\boldsymbol{z}_i, a) \right| > \epsilon \right\} \leq 4 \exp\left\{\left(\frac{G^{\Lambda}(2l)}{l} - \left(\epsilon - \frac{1}{l}\right)^{2}\right) l\right\} \quad . \quad (2.27)$$

Notice that (2.19) is a necessary and sufficient condition for distribution-free uniform two-sided convergence. From bounds of the type of (2.27) we can extract information about the generalisation ability of the learning machine. In particular we can deduce

- The risk $R(a_l)$ defined on the basis of the solution a_l found by the learning machine working under the ERM principle in terms of the empirical risk achieved.
- How close $R(a_l)$ is to the minimal possible $\inf_{a \in \Lambda} R(a)$ for the set of functions implemented by the machine.

To do this let us set

$$4\exp\left\{\left(\frac{G^{\Lambda}(2l)}{l}-\left(\epsilon-\frac{1}{l}\right)^{2}\right)l\right\}=\eta\ ,$$

where η is a parameter in the interval $(0,1)$. Solving the above equation with respect to ϵ we obtain

$$\epsilon=\sqrt{\frac{G^{\Lambda}(2l)-\ln\frac{\eta}{4}}{l}}+\frac{1}{l}\ .$$

The bound of (2.27) can be stated equivalently as follows: With probability $1-\eta$ uniformly for all the functions in the set $a\in\Lambda$ the inequality

$$\int Q(\boldsymbol{z},a)dF(\boldsymbol{z})-\frac{1}{l}\sum_{i=1}^{l}Q(\boldsymbol{z}_i,a)\leq\sqrt{\frac{G^{\Lambda}(2l)-\ln\frac{\eta}{4}}{l}}+\frac{1}{l}$$

is true. Since the previous relation is true for all $a\in\Lambda$ it is also true for the value $a=a_l$ minimising the empirical risk after l points have been processed by the machine. Hence, it holds that

$$R(a_l)\leq R_{\text{emp}}(a_l)+\sqrt{\frac{G^{\Lambda}(2l)-\ln\frac{\eta}{4}}{l}}+\frac{1}{l}\ . \tag{2.28}$$

Inequality (2.28) places an upper bound on $R(a_l)$ which approaches the minimum value attained by R_{emp} with an increasing number of points given the consistency of ERM. If a_0 denotes the value that minimises the risk by use of Chernoff's inequality for the single event $A_{a_0}=\{\boldsymbol{z}:Q(\boldsymbol{z},a_0)>0\}$ we get

$$P\left\{\frac{1}{l}\sum_{i=1}^{l}Q(\boldsymbol{z}_i,a_0)-\int Q(\boldsymbol{z},a_0)dF(\boldsymbol{z})>\epsilon\right\}\leq\exp\{-2\epsilon^2 l\}\ .$$

By a procedure analogous to the one applied to (2.27) the above inequality implies that with probability $1-\eta$

$$\int Q(\boldsymbol{z},a_0)dF(\boldsymbol{z})\geq\frac{1}{l}\sum_{i=1}^{l}Q(\boldsymbol{z}_i,a_0)-\sqrt{\frac{-\ln\eta}{2l}} \tag{2.29}$$

is true. Furthermore, we know that since a_l is the minimiser of the empirical risk the following inequality holds

$$\frac{1}{l}\sum_{i=1}^{l}Q(\boldsymbol{z}_i,a_0)-\frac{1}{l}\sum_{i=1}^{l}Q(\boldsymbol{z}_i,a_l)\geq 0\ .$$

Combining the previous relation with (2.28) and (2.29) we obtain that with probability $1-2\eta$

$$R(a_l)-R(a_0)\leq\sqrt{\frac{G^{\Lambda}(2l)-\ln\frac{\eta}{4}}{l}}+\frac{1}{l}+\sqrt{\frac{-\ln\eta}{2l}}\ . \tag{2.30}$$

Again it is obvious that with the assumption of ERM consistency $R(a_l)$ approximates $\inf_a R(a)$ in the limit $l \to \infty$.

2.9 The VC Dimension

Until now we have constructed distribution-free bounds on the rate of convergence of $R(a_l)$ to $\inf_{a\in\Lambda} R_{\text{emp}}(a)$ and of $R(a_l)$ to $\inf_{a\in\Lambda} R(a)$ based on the growth function $G^\Lambda(l)$. However, we cannot estimate the value of the growth function given the dataset size and the admissible functions of the hypothesis class. The following theorem will provide us with an upper bound on the growth function [60, 61] leading to constructive bounds on the rate of convergence.

Theorem 2.8. *The growth function of a set of indicator functions $Q(\boldsymbol{z}, a), a \in \Lambda$ either satisfies the equality*

$$G^\Lambda(l) = l \ln 2$$

or is bounded by the inequality

$$G^\Lambda(l) \begin{cases} = l \ln 2 & \text{if } l \leq h \\ \leq \ln \left(\sum_{i=0}^{h} C_l^i \right) \leq \ln \left(\frac{el}{h} \right)^h = h \left(1 + \ln \frac{l}{h} \right) & \text{if } l > h \end{cases} ,$$

where h is the largest integer for which

$$G^\Lambda(h) = h \ln 2 \ .$$

The theorem says that the growth function can be either linear or at most logarithmic in l as a result of the second branch of $G^\Lambda(l)$. That means that it cannot scale with l slower than linearly but faster than logarithmically. For example $G^\Lambda(l)$ cannot be l^p, with $0 < p < 1$. The quantity h characterises the ability of the functions in the hypothesis class to explain the data and is called the VC dimension of a set of indicator functions [60, 61]. There exists an alternative definition of the VC dimension which is connected to the procedure of estimating it thus leading to constructive bounds.

Definition 2.9. The VC dimension of a set of functions $Q(\boldsymbol{z}, a), a \in \Lambda$, is equal to the largest number h of points $\boldsymbol{z}_1 \ldots, \boldsymbol{z}_l$ that can be separated into two different classes in all the 2^h possible ways using functions from this set. We say then that the VC dimension is the maximum number of points that can be shattered by the set of functions.

One remark that we can add in connection with the above definition is that if for any l there exists a set of l points that can be shattered by the functions in the hypothesis class then the VC dimension is infinite.

For the case where the VC dimension is finite the growth function grows at most logarithmically with the sample size for $l > h$. If we use this upper bound in the place of the growth function we end up with a constructive bound on the rate of convergence

$$P\left\{\sup_{a\in\Lambda}\left|\int Q(\boldsymbol{z},a)dF(\boldsymbol{z}) - \frac{1}{l}\sum_{i=1}^{l} Q(\boldsymbol{z}_i,a)\right| > \epsilon\right\}$$

$$\leq 4\exp\left\{\left(\frac{h(1+\ln(2l/h))}{l} - \left(\epsilon - \frac{1}{l}\right)^2\right)l\right\} \quad . \tag{2.31}$$

It can be easily seen that for a finite VC dimension the growth function increases slower than linearly resulting in $\lim_{l\to\infty} G^{\Lambda}(l)/l \to 0$ which is the condition for distribution-free uniform two-sided convergence. The following theorem [62] states something stronger regarding the role of the VC dimension in the uniform convergence.

Theorem 2.10. *The finiteness of the VC dimension is not only a sufficient but also a necessary condition for uniform convergence of the frequencies of events* $A_a = \{\boldsymbol{z} : Q(\boldsymbol{z},a) = 1\}$ *to their probabilities for any distribution* $F(\boldsymbol{z})$.

Proof. For the validity of our claim it suffices to disprove uniform convergence for a specific distribution. Given that the VC dimension is infinite the equality

$$N^{\Lambda}(\boldsymbol{z}_1,\ldots,\boldsymbol{z}_l) = 2^l$$

holds for some set $\boldsymbol{Z}^l = \boldsymbol{z}_1,\ldots,\boldsymbol{z}_l$. Uniform convergence will fail if for any l and $\epsilon < 1$ there exists a distribution $F(\boldsymbol{z})$ such that

$$\sup_{a\in\Lambda}\left|\int Q(\boldsymbol{z},a)dF(\boldsymbol{z}) - \frac{1}{l}\sum_{i=1}^{l} Q(\boldsymbol{z}_i,a)\right| > 1-\epsilon \quad .$$

is true with probability one. We fix an arbitrary sample $\boldsymbol{Z}^l$ of size l which we expand by a set $\boldsymbol{Z}^\star = \boldsymbol{z}_{l+1},\ldots,\boldsymbol{z}_n$ consisting of $n-l$ points, where n is chosen to be $n > l/\epsilon$. The sample $\boldsymbol{Z}^l \cup \boldsymbol{Z}^\star$ is generated by a uniform distribution concentrated only on the n points, i.e. the probability of any such point is $P(\boldsymbol{z}_i) = 1/n$. Even after expanding the dataset the functions $Q(\boldsymbol{z},a)$ are still able to shatter the new dataset. This enables us to choose out of all the possible dichotomies realised by the functions of the class the one dichotomy which corresponds to $Q(\boldsymbol{z},a^\star)$ taking the value of zero on the points of the subset $\boldsymbol{Z}^l$ and one on the rest of them contained in the subset $\boldsymbol{Z}^\star$. Formally, this implies that

$$\frac{1}{l}\sum_{i=1}^{l} Q(\boldsymbol{z}_i,a^\star) = 0$$

and at the same time that

$$\int Q(\boldsymbol{z},a^\star)dF(\boldsymbol{z}) = \frac{n-l}{n} > 1-\epsilon$$

since $Q(\boldsymbol{z}, a^\star) = 1$ only for $\boldsymbol{z} \in \boldsymbol{Z}^\star$. Hence, with probability one it holds that the supremum over all functions in the set is greater than $1 - \epsilon$ for any l rendering the finiteness of the VC dimension a necessary condition for uniform convergence. □

It is important to point out that one should not confuse the VC dimension with the number of free parameters appearing in a function because this can be proved totally wrong. For example, on the one hand, there can be a class of functions the members of which differ only in one parameter but which, nevertheless, possess an infinite VC dimension. On the other hand one can think of a class of functions which, although described by a high number of free parameters, have a low VC dimension.

A case where the VC dimension can be easily estimated by the number of free parameters involves any hypothesis class that contains indicator functions linear in their parameters a_k

$$Q(\boldsymbol{z}, a) = \theta\left(\sum_{k=1}^{n} a_k \phi_k(\boldsymbol{z})\right), \qquad a_k \in \mathbb{R} \ ,$$

where a is a vector with components $a_k, k = 1, \ldots, n$. The terms $\phi_k(\boldsymbol{z})$ entering the expression of $Q(\boldsymbol{z}, a)$ are linearly independent functions of the sample elements z. The VC dimension of this set of functions equals the number n of free parameters. Application of this case can be considered the class of zero-threshold hyperplanes in the n-dimensional space implemented by a learning machine in the classification task. If we consider hyperplanes possessing some bias the free parameters are increased by one and so is the VC dimension of the set.

For the class of indicator functions non-linear in their parameters the VC dimension can be less than or even exceed the number of parameters. A typical example of the latter case is the following set of indicator functions

$$Q(z, a) = \theta\left(\sin az\right), \qquad z \in (0, 2\pi), \quad a \in (0, \infty)$$

the VC dimension of which is infinite.

2.10 The Structural Risk Minimisation Principle

Let us turn to the study of (2.31) that provides us with the rate of convergence of the empirical risk to the expected one by means of the VC dimension. From this relationship, following the procedure that led to (2.28) from (2.27), one can assert that with probability $1 - \eta$ for all functions parametrised by $a \in \Lambda$

$$R(a) \leq R_{\text{emp}}(a) + \sqrt{\frac{h(1 + \ln(2l/h)) - \ln\frac{\eta}{4}}{l}} + \frac{1}{l} \ . \tag{2.32}$$

The upper bound on the expected risk $R(a)$ is formed by the sum of two contributions, the one coming from $R_{\text{emp}}(a)$ and the other from a term involving the ratio l/h. It is not obvious by mere observation that a minimisation of $R_{\text{emp}}(a)$ as dictated by the ERM principle will lead to the tightest bound on $R(a)$. We can easily verify that if we are dealing with large sample sizes with respect to the VC dimension h the lowest value that the r.h.s. can attain is determined mainly by R_{emp}. Thus, the suggestion of ERM principle that we should try to find the function that classifies the training set with the minimum number of errors seems to be in the right direction since it proves decisive in the determination of the generalisation ability of the machine.

On the other hand if it happens that the ratio l/h is not large then the second summand on the r.h.s. of (2.32), called the confidence interval, plays an important role and the solution that gives the minimum guaranteed risk does not necessarily coincide with the one that comes from the ERM principle. Reducing the VC dimension of the hypothesis class reduces the contribution of the confidence interval but it is reasonable to expect that it increases the training error. Thus, the construction of (2.32) leaves one with the freedom to control the generalisation ability of the learning machine by adjusting two opposing factors namely, the number of training errors on the one hand and the capacity of the function class on the other. Surely, in our attempt for simultaneous minimisation over both terms the bound (2.32) provides us with a quantitative criterion on the basis of which a compromise between the two can be accomplished.

This new criterion called the Structural Risk Minimisation (SRM) principle [62], in contrast to the ERM principle, suggests the minimisation of the bound over both the empirical risk and the confidence interval which is controlled by the capacity of the hypothesis class. Let us define a structure on the set S of functions $Q(\boldsymbol{z}, a), a \in \Lambda$. Consider the nested subset of functions

$$S_1 \subset S_2 \subset \cdots \subset S_k \cdots ,$$

where $S_k = \{Q(\boldsymbol{z}, a) : a \in \Lambda_k\}$. The union of all subsets is denoted by $S^\star = \cup_k S_k$. The subsets are constructed in a way such that the VC dimension of the set S_k of functions is nondecreasing with increasing index k

$$h_1 \leq h_2 \leq \cdots h_k \leq \cdots \quad .$$

We are interested only in classification tasks. This restricts the functions in each element S_k of the structure to the indicator functions $\{Q(\boldsymbol{z}, a_k) \in \{1, 0\}, \;\; a_k \in \Lambda_k\}$. Additionally, for the structure to be admissible we need the VC dimension of each element in the structure to be kept finite and the set $S^\star$ to be everywhere dense in the set S in the L_1 metric. According to the SRM principle given a number of observations one should choose the element from the structure that yields the minimum guaranteed risk.

The policy imposed by this principle to discover the element of the structure with the appropriate capacity that leads to the minimisation of the risk justifies its name.

As in the case of the ERM principle, analogous questions of consistency are raised also in connection with the SRM principle. For example, is it possible for the risk estimated on the basis of the function chosen according to this principle from an element S_k of the structure to converge to the minimum one in S? And if this happens what would be a bound on the rate of convergence?

From (2.30) by setting in the place of the growth function $G^{\Lambda}(2l)$ its upper bound written in terms of the VC dimension assumming $2l > h_k$ and fixing $\eta = 1/l^2$ we obtain that with probability $1 - 2/l^2$

$$R(a_l^k) - R(a_0^k) = \int Q(z, a_l^k) dF(z) - \inf_{a \in \Lambda_k} \int Q(z, a) dF(z)$$

$$\leq \sqrt{\frac{2 \ln l}{2l}} + \sqrt{\frac{h_k \left(\ln \frac{2l}{h_k} + 1 \right) + 2 \ln 2l}{l}} + \frac{1}{l} \ . \qquad (2.33)$$

We recognise in the place of the upper bound the confidence interval. The term $R(a_l^k)$ denotes the risk with respect to a solution found within the functions of the k-th element of the structure whereas $R(a_0^k)$ signifies the minimum risk attainable for functions in the same element. The term on the l.h.s. of the previous relation can be decomposed as

$$R(a_l^k) - R(a_0^k) = R(a_l^k) - R(a_0) + R(a_0) - R(a_0^k) \ .$$

With a little rearrangement this automatically transforms (2.33) into a relation bounding the rate of convergence $V(l) = R(a_l^k) - R(a_0)$

$$V(l) \leq r_k + \sqrt{\frac{2 \ln l}{2l}} + \sqrt{\frac{h_k \left(\ln \frac{2l}{h_k} + 1 \right) + 2 \ln 2l}{l}} + \frac{1}{l} \ , \qquad (2.34)$$

where $r_k = R(a_0^k) - R(a_0)$.

One can show that if one imposes rules for the choice of the appropriate element S_k of the structure that depend on the number l of observations then for l tending to infinity the risk $R(a_l^k)$ approaches the smallest one $R(a_0)$ in the whole structure. Let us denote by $k(l)$ the rule based on the number of observations that discriminates between the subsets of the structure. In terms of the new notation (2.34) becomes

$$V(l) \leq r_{k(l)} + \sqrt{\frac{2 \ln l}{2l}} + \sqrt{\frac{h_{k(l)} \left(\ln \frac{2l}{h_{k(l)}} + 1 \right) + 2 \ln 2l}{l}} + \frac{1}{l} \ . \qquad (2.35)$$

In order for consistency of the SRM to hold we need $V(l)$ to tend to 0 for an increasing number of observations. Indeed, if the element containing the minimiser function a_0

of the expected risk is found $\lim_{l\to\infty} r_{k(l)} = 0$ due to the density of $S^\star$ in S. So for convergence to take place we need additionally the second term to tend to 0 for $l \to \infty$ or equivalently

$$\lim_{l\to\infty} \frac{h_{k(l)} \ln l}{l} = 0,$$

a condition which reminds us of $\lim_{l\to\infty} G^\Lambda(l)/l = 0$ for consistency of the ERM principle to hold. The term $r_{k(l)}$ of (2.35) known as the rate of approximation is related to the deviation of the best approximation in $S_{k(l)}$ from the smallest possible. It is reasonable to expect that as we move to subsets with larger capacity the deviation will become smaller. On the other hand the second term entering (2.35) known as the estimation error measures the deviation of the risk computed on the basis of a function in $S_{k(l)}$ from the smallest possible in $S_{k(l)}$. We anticipate that as $k(l)$ increases the larger this deviation becomes. Therefore, we come to the conclusion that the rate of convergence is governed by two opposing factors.

2.11 The Δ-Margin Hyperplane

The SRM principle motivates us to consider sets of decision rules of increasing complexity measured in terms of the VC dimension which in the case of linear classification depends linearly on the dimensionality of the feature space. If one would like to apply the SRM principle to a classification task one should attempt to construct a structure the elements of which are endowed with a variable VC dimension parametrised appropriately. On the basis of our analysis so far the dimensionality is the only such available parameter. As we discussed in Chapter 1 the use of kernels enables us to employ linear decision rules in feature spaces of dimensionality much larger than the one of the original instance space. However, this high dimensionality renders the SRM principle obsolete because the confidence interval would most probably dominate the r.h.s. of (2.35) instructing us to choose the element of the structure with the smallest cardinality. This is not always in agreement with experimental results indicating that there are cases where the use of kernels is beneficial since it enhances the generalisation ability of the machine. Thus, we are motivated to construct structures the elements of which have a VC dimension depending on an additional parameter. As such a parameter one may use the so-called margin of the decision rule a notion to which we now turn.

The points $\boldsymbol{x}$ belonging to a hyperplane can be described by the equation

$$\boldsymbol{u} \cdot \boldsymbol{x} + b = 0 \ ,$$

where $\boldsymbol{u}$ is the unit vector perpendicular to the hyperplane, whereas b controls the normal distance of the hyperplane from the origin. We call such a hyperplane a Δ-margin separating hyperplane if it assigns a label y to an instance $\boldsymbol{x}$ as follows:

$$y = \begin{cases} 1 & \text{if } \boldsymbol{u} \cdot \boldsymbol{x} + b \geq \Delta \\ 0 & \text{if } |\boldsymbol{u} \cdot \boldsymbol{x} + b| < \Delta \\ \text{-1} & \text{if } \boldsymbol{u} \cdot \boldsymbol{x} + b \leq -\Delta \end{cases} .$$

The classification of an instance is considered correct if the prediction of the label as described above coincides with the true label. As it appears from the previous relationship in order for the instances to be classified correctly it does not suffice to have them on the correct side of the hyperplane but they further need to lie at a distance, called margin, of at least $\Delta > 0$ from it.

In a attempt to investigate the applicability of the SRM principle to structures the elements of which are Δ-margin hyperplanes it remains further to be seen how the VC dimension depends on Δ. Bounds on the VC dimension of Δ-margin separating hyperplanes were obtained by Vapnik [66, 67]. In particular, for Δ-margin hyperplanes possessing no bias the following theorem holds.

Theorem 2.11. *Suppose $X^\star$ is a subset of the input space contained in a sphere of radius R centred at the origin of the feature space. The set of zero-threshold Δ-margin separating hyperplanes defined on $X^\star$ has VC dimension h satisfying*

$$h \leq \min\left(\left[\frac{R^2}{\Delta^2}\right], n\right) .$$

We give a proof along the lines of [5, 14].

Proof. Let us assume that there are r points $\boldsymbol{x}_i$ shattered by the Δ-margin hyperplanes. Then, for all the possible assignments of labels y_i to $\boldsymbol{x}_i$ we have

$$y_i \left(\boldsymbol{u} \cdot \boldsymbol{x}_i\right) \geq \Delta \qquad \forall i \ . \tag{2.36}$$

By summing up (2.36) over i

$$\boldsymbol{u} \cdot \sum_{i=1}^{r} y_i \boldsymbol{x}_i \geq r\Delta$$

and noticing that

$$\boldsymbol{u} \cdot \sum_{i=1}^{r} y_i \boldsymbol{x}_i \leq \left\| \sum_{i=1}^{r} y_i \boldsymbol{x}_i \right\|$$

we get

$$r\Delta \leq \left\| \sum_{i}^{r} y_i \boldsymbol{x}_i \right\| . \tag{2.37}$$

Let us assume that the labels $y_i \in \{-1, +1\}$ accompanying the points are randomly generated and drawn independently from a uniform distribution. The variables to which we give such statistical properties are sometimes known as Rademacher variables. Taking the expectation of $\|\sum_i^r y_i \boldsymbol{x}_i\|^2$ over all the possible labellings of the data and exploiting the linearity of the expectation $\boldsymbol{E}$, we obtain

$$\begin{aligned}\boldsymbol{E}\left\|\sum_{i=1}^{r} y_i \boldsymbol{x}_i\right\|^2 &= \sum_{i=1}^{r} \boldsymbol{E}\left(y_i \boldsymbol{x}_i \cdot \sum_{j=1}^{r} y_j \boldsymbol{x}_j\right) = \sum_{i=1}^{r} \boldsymbol{E}\left(y_i \boldsymbol{x}_i \cdot \left(\left(\sum_{j\neq i}^{r} y_j \boldsymbol{x}_j\right) + y_i \boldsymbol{x}_i\right)\right) \\ &= \sum_{i=1}^{r}\left(\left(\sum_{i\neq j} \boldsymbol{E}(y_i \boldsymbol{x}_i \cdot y_j \boldsymbol{x}_j)\right) + \boldsymbol{E}(y_i \boldsymbol{x}_i \cdot y_i \boldsymbol{x}_i)\right) \\ &= \sum_{i=1}^{r} \boldsymbol{E}\left\|y_i \boldsymbol{x}_i\right\|^2 \quad . \end{aligned} \tag{2.38}$$

The last inequality comes from the fact that the Rademacher variables are independent and have zero mean. Since $\|y_i \boldsymbol{x}_i\| = \|\boldsymbol{x}_i\| \leq R$ (2.38) becomes

$$\boldsymbol{E}\left\|\sum_{i=1}^{r} y_i \boldsymbol{x}_i\right\|^2 \leq rR^2$$

from where we may deduce that for a specific assignment of labels

$$\left\|\sum_{i=1}^{r} y_i \boldsymbol{x}_i\right\|^2 \leq rR^2 \tag{2.39}$$

is true. Combining (2.37) and (2.39) we obtain

$$r\Delta \leq \left\|\sum_{i}^{r} y_i \boldsymbol{x}_i\right\| \leq \sqrt{r}R$$

from where

$$r \leq \frac{R^2}{\Delta^2} \quad .$$

Taking into account that for hyperplanes passing through the origin the maximum number of points shattered does not exceed the dimensionality n of the feature space the bound of Theorem 2.11 follows. □

There exists a similar bound on the VC dimension for Δ-margin hyperplanes possessing bias in which R is the radius of the smallest sphere enclosing the data but not necessarily centred at the origin.

Theorem 2.12. *The set of Δ-margin separating hyperplanes defined on a subset $X^\star$ of the input space has VC dimension h satisfying*

$$h \leq \min\left(\left[\frac{R^2}{\Delta^2}\right], n\right) + 1 \ ,$$

where R is the radius of the smallest sphere centred at some point of the feature space which contains $X^\star$.

It is obvious that this alternative bound on the VC dimension can be significantly tighter than the one appearing in Theorem 2.11. The proof following [67] is different than the one presented earlier and relies mainly on geometrical arguments.

Proof. For r points $\boldsymbol{x}_i$ shattered by the class of linear classifiers with bias let us consider all the possible realisations of two-category classification problems and the corresponding convex hulls formed by the examples belonging to the two classes. There are 2^r possible labellings which will be denoted by $T_1, \ldots, T_{2^r}$. Let $\rho(T_i)$ be the distance between the convex hulls formed if the assignment T_i of the labels is chosen. It is apparent that we can interpret the minimum distance $\min_i \rho(T_i)$ separating the convex hulls constructed according to the various labellings as twice the maximum of the margins that the set of r examples possesses from the decision rules belonging to the class. We demand that

$$\min_i \rho(T_i) > 2\Delta \ . \tag{2.40}$$

The VC dimension h of the class of Δ-margin hyperplanes is the maximum number r of points that can be shattered by the functions contained in that class.

As we mentioned previously the VC dimension of indicator functions linear in their parameters equals the number $n + 1$ of such free parameters. Since the classification with non-zero margin is a special case of classification, for linear classifiers with bias the maximum number r of points shattered satisfies

$$r \leq n + 1 \ . \tag{2.41}$$

In the following we will attempt to obtain a tighter bound than the above. For a given number of points confined within a ball of radius R we expect that there exists a special arrangement of them that leads to the maximisation of the margin for all possible dichotomies. Our subsequent argument relies on the almost obvious assumption that the above arrangement of the points is a symmetrical one. Such a symmetrical construction can be realised if we place the r points on an $r-1$ regular simplex the vertices of which lie on the surface of the minimal enclosing sphere of radius R. This is proved formally in [27]. If we come up with an explicit formula giving us the dependence of $\min_i \rho(T_i)$ on r for such an arrangement and solve (2.40) with respect to r we will obtain an upper bound

on the VC dimension h possibly tighter than the one of (2.41). Explicit calculations on a regular simplex give

$$\min_i \rho(T_i) = \begin{cases} \frac{2R}{\sqrt{r-1}} \text{ for } r \text{ even} \\ \\ \frac{2R}{\sqrt{r-1}} \sqrt{\frac{r^2}{r^2-1}} \text{ for } r \text{ odd} \quad . \end{cases} \tag{2.42}$$

For a number of points $r > 10$ the two expressions become almost equal which enables us to use for simplicity only the first of them. Therefore, (2.40) in combination with (2.42) and taking into account the trivial bound (2.41) yields

$$r \leq \min\left(\left[\frac{R^2}{\Delta^2}\right], n\right) + 1 \ .$$

□

By making use of the non-asymptotic bound (2.32) we can assert that

Corollary 2.13. *The probability P_{error} that a test point will not be separated correctly by the Δ-margin hyperplane is bounded from above with probability $1 - \eta$ as follows*

$$P_{\text{error}} \leq \frac{m}{l} + \sqrt{\frac{h\,(1 + \ln 2l) - \ln \frac{\eta}{4}}{l}} + \frac{1}{l} \ . \tag{2.43}$$

Here m is the number of points that were not classified correctly by the Δ-margin hyperplane and h is constrained by Theorem 2.12.

Notice that for the validity of the above statement one must ensure that in the procedure that gave rise to the Δ-margin hyperplane only functions from an appropriately restricted class should be employed. However, such an a priori restriction of the class of functions may be very difficult in practice since the class of Δ-margin hyperplanes depends crucially on the dataset. As a consequence, meaningful choices of the parameter Δ become possible only a posteriori.

2.12 Concluding Remarks

From the discussion in the present chapter it follows that minimisation of the number of errors in the training set although desirable cannot by itself guarantee good generalisation for the solution produced by a learning machine. An important role in the generalisation ability of a learning machine is played by the capacity of the class of functions employed by it which is described in terms of several measures, such as the VC dimension, and controls the confidence interval term. We saw that this capacity can be seriously reduced even for very high dimensional feature spaces if hyperplanes separating the data with large margin are used, thereby leading to a substantial suppression of the

confidence interval term. These considerations give us additional motivation for seeking large margin solution hyperplanes on top of the common belief based on intuition that such hyperplanes might be preferable. In the following chapters we describe two broad categories of algorithms sharing the same objective of obtaining large margin solutions, namely Support Vector Machines and Perceptron-like large margin classifiers.

Chapter 3

Support Vector Machines

3.1 The Motivation behind Support Vector Machines

Corollary 2.13 used in conjunction with Theorem 2.12 suggested that a separation hyperplane possessing some margin if combined with a low training error may result in a better generalisation ability. Moreover, at a second reading Theorem 2.12 reveals that even if the number of dimensions in the space in which the machine works is high its generalisation ability may not be affected if it tries to find solutions characterised by a large margin. This results in a restriction of the capacity of the function class implemented by the machine thus preventing overfitting. We will proceed to the study of learning machines that taking advantage of the findings of statistical learning theory were designed to pursue the optimisation of the aforementioned factors (low training error, large margin) which control the generalisation ability. These learning machines are referred to as Support Vector Machines (SVMs) [9, 66, 67, 14, 48].

SVMs can also be implemented efficiently in feature spaces built by applying nonlinear transformations to the original space in which the points live. The remarkable thing is that the dimensionality of the induced feature spaces does not hinder at all the efficiency of SVMs since the whole algorithmic procedure takes place in dual formulation. Their ability to make exclusive use of kernels enables them to circumvent any implementational burdens and to run even in feature spaces with an enormous number of dimensions.

Another advantage of SVMs coming from the fact that they are formulated in terms of positive semi-definite kernels is that they have a guaranteed convergence to a globally optimal solution. Other pattern recognition algorithms like neural networks [7] in contrast to SVMs come up with solutions which are trapped in local minima.

3.2 Separating Hyperplanes

The class of functions that SVMs employ is the class of hyperplanes defined by

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0, \quad \boldsymbol{w} \in X,\ b \in \mathbf{R}\ , \tag{3.1}$$

which for any test point $\boldsymbol{x} \in X$ induce the corresponding decision function $\hat{y}(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$ taking values from the set $\{1, -1\}$. The solution is uniquely determined if the weight vector $\boldsymbol{w}$ and the bias b are specified by the learning algorithm. In the beginning we will consider those datasets that can be split into two classes by using a hyperplane and will defer the inseparable case for later. We will assume that all the hyperplanes we refer to are able to provide at least a mere separation of the data. Among all the possible hyperplanes we can distinguish the single pair $(\boldsymbol{w}, b)$ that induces a solution with a maximal distance from the nearest points called maximum geometric margin. Usually we omit the qualification geometric and we call it just margin. We can rescale the pair $(\boldsymbol{w}, b)$ by multiplying (3.1) with a constant factor and still describe the same hyperplane. Exploiting this freedom we rescale $(\boldsymbol{w}, b)$ so that the points lying closest to it satisfy the following relationship

$$\min_{i=1,\ldots,l} |\boldsymbol{w} \cdot \boldsymbol{x}_i + b| = 1\ . \tag{3.2}$$

Let us for the moment regard $\boldsymbol{x}_i$ not as some training pattern but rather as an arbitrary vector satisfying (3.2). Then, we can treat (3.2) as an equation describing two separate hyperplanes. The one defined by $\boldsymbol{w} \cdot \boldsymbol{x}_i + b = 1$ lies in the region of the points $\boldsymbol{x}$ characterised as positive ones ($\hat{y}(\boldsymbol{x}) = 1$) and the other defined by $\boldsymbol{w} \cdot \boldsymbol{x}_i + b = -1$ lies in the region of points belonging to the negative class ($\hat{y}(\boldsymbol{x}) = -1$). Each of these two equations defines a hyperplane parallel to the one described by (3.1) at a distance equal to the margin that the training points possess from the solution hyperplane. If the index i is used to indicate those training patterns that are closest to the separating hyperplane then (3.2) can be rewritten as

$$y_i\,(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 1\ . \tag{3.3}$$

A hyperplane for which the pair $(\boldsymbol{w}, b)$ is normalised such that (3.3) holds is said to be written in canonical form. The decision rule $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$ responsible for the assignment of a label to any data point $\boldsymbol{x}$ when multiplied by its label y measures another kind of margin which is called the functional margin. Moreover, the sign of $yf(\boldsymbol{x})$ computed for the example $(\boldsymbol{x}, y)$ signifies correct classification when positive and wrong classification when negative. After bringing the hyperplane in the canonical form we divide (3.3) by $\|\boldsymbol{w}\|$ so that in the place of $\boldsymbol{w}$ its direction $\boldsymbol{u}$ appears. Then the r.h.s.

of the resulting equation represents the geometric margin γ of the dataset

$$\gamma = y_i \left(\boldsymbol{u} \cdot \boldsymbol{x}_i + \frac{b}{\|\boldsymbol{w}\|} \right) = \frac{1}{\|\boldsymbol{w}\|} \ . \tag{3.4}$$

In analogy to the above margin of the dataset which is the minimum distance the positions of the points have from the separating hyperplane (3.1) we can define the margin $\gamma(\boldsymbol{x}, y)$ of any point $(\boldsymbol{x}, y)$ to be equal to the distance of that point from the separating hyperplane. The margin $\gamma(\boldsymbol{x}, y)$ is obtained from the relationship

$$\gamma(\boldsymbol{x}, y) = y \left(\boldsymbol{u} \cdot \boldsymbol{x} + \frac{b}{\|\boldsymbol{w}\|} \right) \ .$$

The positivity of $\gamma(\boldsymbol{x}, y)$ indicates that an instance is correctly classified with respect to the separating hyperplane. The margin of a misclassified point coincides with the negative of its distance from the hyperplane. Furthermore, if the positivity of $\gamma(\boldsymbol{x}, y)$ holds for every training example $(\boldsymbol{x}, y)$ with respect to some hyperplane then the training set is linearly separable. From the relation (3.4) giving the margin for the points closest to the hyperplane it is apparent that the margin is larger if (3.3) is satisfied with lower values of the norm of $\boldsymbol{w}$. Assuming that the functional margin of the closest points to the separating hyperplane is normalised to unity we can look for solutions possessing larger geometric margin by seeking hyperplanes with weight vectors $\boldsymbol{w}$ of lower norm.

In the linearly separable case with margin of at least γ and for the class of γ-margin hyperplanes a worst case bound follows directly from Corollary 2.13 by setting $m = 0$ and substituting the VC dimension h by its upper bound of Theorem 2.12. We also assume that the dimensionality of the space is so high that the term depending on the margin prevails in the determination of the upper bound on h. Thus, with probability $1 - \eta$ the probability of an unseen pattern to give rise to a mistake due to its failure to be classified with margin of at least γ satisfies

$$P_{\text{error}} < \sqrt{\frac{\left(\left[\frac{R^2}{\gamma^2} \right] + 1 \right) (1 + \ln 2l) - \ln \frac{\eta}{4}}{l}} + \frac{1}{l} \ . \tag{3.5}$$

From a mere inspection of (3.5) it is easily understood that we can improve the predictive ability of our training machine if we seek hyperplanes possessing margins near the maximum one. It is worth pointing out that a bound like the one of (3.5) without the square root on the r.h.s. could be derived by assuming from the beginning that there are no errors in the training set instead of setting the number of training errors to zero in (2.43).

Generalisation bounds depending on the margin were also derived in [50]. More specifically the following theorem holds.

Theorem 3.1. *Suppose inputs are drawn independently according to a distribution whose support is contained in a ball in* $\mathbb{R}^n$ *centred at the origin of radius* R*. If we succeed in correctly classifying* l *such inputs by a canonical hyperplane with* $\|\boldsymbol{w}\| = 1/\gamma$ *and with* $|b| \leq R$*, then with confidence* $1 - \eta$ *the generalisation error will be bounded from above by*

$$\epsilon(m, \gamma) = \frac{2}{l}\left(k \log_2\left(\frac{8el}{k}\right)\log_2(32l) + \log_2\frac{8l}{\eta}\right),$$

where $k = [577R^2/\gamma^2]$.

Both (3.5) and Theorem 3.1 are applicable only if we somehow are able to make sure that only functions from the restricted class of γ-margin hyperplanes were considered as acceptable solutions by the machine. This might necessitate that a value of the margin smaller than the one corresponding to the solution found may be employed in the generalisation bounds. In the special case that we know before running that the classifier will find the solution with maximum margin this may be substituted in the above bounds even if the exact value of this margin is not known a priori.

Apart from the theoretical arguments we can rely on our intuition in order to find reasons why the maximum margin is indeed a good property of the solution hyperplanes. Let us train our machine on a set of points and consider a test set which is generated from the training set by adding some noise bounded in norm by the quantity r. This means that the resulting test patterns cannot exceed the boundary surfaces of spheres of radius r centred at the training points. If the margin γ that separates the closest points from the hyperplane is greater than r then all the test points will be classified correctly. A larger margin allows a higher level of noise as this is measured by r without incurring any error and even higher if we are willing to accept a low test error.

3.3 The Optimal Margin Hyperplane

We have expressed earlier the hyperplanes which pass through the points lying closest to the separating hyperplane in the form (3.3). If we make the assumption that we have at least one example in each category in order for the binary classification to have a meaning then we may come up with two such hyperplanes passing through at least two examples belonging to different classes. This implies that a separating hyperplane can be found with these examples having a functional margin with respect to it given by

$$\gamma(\boldsymbol{x}_i, y_i) = y_i\left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right) = 1 \ .$$

The index i denotes here only the examples from both categories which are closest to the hyperplane. Moreover, for all the examples $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, l$ in the dataset we

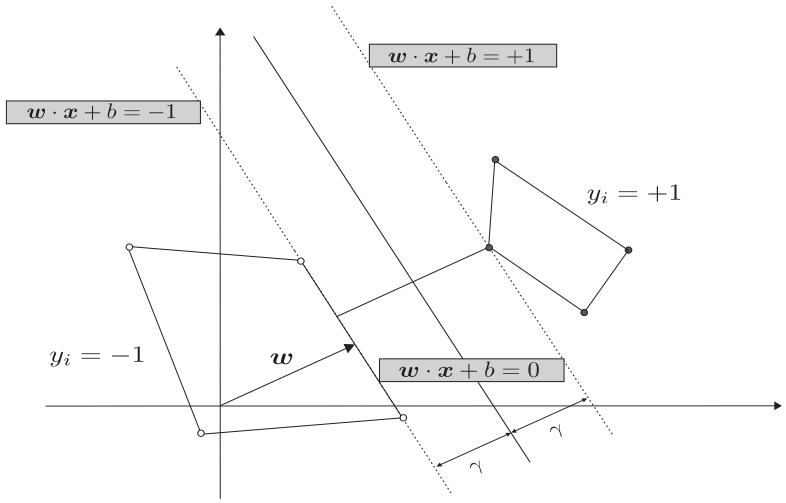

FIGURE 3.1: The optimal hyperplane is the one bisecting the segment that connects the closest points of the convex hulls of the two classes. By requiring the scaling to be such that the point(s) closest to the hyperplane satisfy $|\boldsymbol{w} \cdot \boldsymbol{x}_i + b| = 1$ we obtain a hyperplane in canonical form. The maximum margin γ equals $1/\|\boldsymbol{w}\|$.

can assert that the following inequality holds true

$$y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right) \geq 1 \ .$$

In order to achieve a low generalisation error we have to determine a solution with maximum margin. This is equivalent to seeking a hyperplane with minimum $\|\boldsymbol{w}\|$ which at the same time manages to classify all the points of the training set with functional margin greater or equal to 1 (Fig. 3.1).

The above way of stating the problem of finding separating hyperplanes with maximum margin prompts us to proceed to a formulation in terms of optimisation theory. Specifically, the optimisation problem can be stated formally as follows:

$$\text{minimise}_{\boldsymbol{w},\text{b}} \ \frac{1}{2} \|\boldsymbol{w}\|^2 \ ,$$

$$\text{subject to } \ y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right) \geq 1 \text{ for all } i = 1, \ldots, l \ .$$

We call this formulation the primal optimisation problem. We observe that the objective function $\frac{1}{2} \|\boldsymbol{w}\|^2$ is a strictly convex function of $\boldsymbol{w}$, hence the primal problem admits a unique solution. The qualification "primal" stems from the fact that the original quantities $(\boldsymbol{w}, b)$ defining the solution hyperplane enter the expression as opposed to an equivalent expression which will be described shortly. Because the problem is hard to solve in its primal form we follow a procedure that transforms it into an equivalent form [41]. We begin by writing down the corresponding Lagrangian in a primal form

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\boldsymbol{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1\right] \ , \tag{3.6}$$

where $\alpha_i \geq 0$ are parameters known as the Lagrange multipliers which are in the following considered as the components of a vector $\boldsymbol{a}$. By differentiating $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ with respect to $\boldsymbol{w}$ and b we get

$$\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{l} y_i \alpha_i \boldsymbol{x}_i = \boldsymbol{0} \ , \tag{3.7}$$

$$\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^{l} y_i \alpha_i = 0 \ . \tag{3.8}$$

Solving (3.7) with respect to $\boldsymbol{w}$ and substituting back to (3.6) we obtain the dual form of the Langrangian

$$\begin{aligned} L(\boldsymbol{\alpha}) &= \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j - \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j + \sum_{i=1}^{l} \alpha_i \\ &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \ . \end{aligned} \tag{3.9}$$

Note here that the primal variables $\boldsymbol{w}$ and b have been totally eliminated from the expression of the Lagrangian. Its dual form is written exclusively in terms of the variables $\alpha_i, \ i = 1, \ldots, l$ which for this reason are called dual variables. The new optimisation problem in terms of the dual variables is stated as follows

$$\begin{aligned} &\text{maximise}_{\boldsymbol{\alpha}} \ L(\boldsymbol{\alpha}) \ , \\ &\text{subject to} \ \sum_{i=1}^{l} \alpha_i y_i = 0 \ , \\ &\qquad\qquad \alpha_i \geq 0 \ \ i = 1, \ldots, l \ . \end{aligned}$$

After determining the solution $\boldsymbol{\alpha}^\star$ of the dual problem in terms of the parameters $\alpha_i^\star$ the optimum weight vector $\boldsymbol{w}^\star$ which is the unknown quantity of the primal problem can be derived from (3.7). Solving with respect to $\boldsymbol{w}^\star$ we obtain $\boldsymbol{w}^\star = \sum_{i=1}^{l} \alpha_i^\star y_i \boldsymbol{x}_i$ from which it is clear that the solution weight vector can be represented as a linear combination of the training patterns. Recall that this was a sufficient condition in order for the decision rule to be evaluated by means of a kernel. The other parameter which remains to be specified is the bias $b^\star$ of the hyperplane. The bias will be evaluated using the primal constraints since b does not enter the dual formulation. In this attempt we employ exclusively the active constraints, which are the ones satisfied by the solution as equalities, by solving each one of them with respect to $b^\star$. Since there will be more than one active constraints we perform an average over all the values $b^\star$ found

$$b^\star = \frac{\sum_{i=1}^{s} (y_i - \boldsymbol{w}^\star \cdot \boldsymbol{x}_i)}{s} \ ,$$

where s denotes the number of active constraints.

We will argue in the following that the dual Lagrangian (3.9) is a concave function of $\boldsymbol{\alpha}$. The Hessian matrix formed by taking the second order partial derivatives of $-L(\boldsymbol{\alpha})$ with respect to α_i and α_j consists of the entries $\frac{1}{2}\left(y_i\boldsymbol{x}_i \cdot y_j\boldsymbol{x}_j\right)_{(i,j)=(1,1)}^{(l,l)}$. If the labels accompanying the points are ignored this matrix corresponds to the simplest of the kernels, the linear one, which is positive semi-definite. We can incorporate the labels into the initial mapping transformation $\phi(\boldsymbol{x}_i) = \boldsymbol{x}_i$ and obtain a new mapping $\phi'(\boldsymbol{x}_i) = y_i\boldsymbol{x}_i$ which yields a kernel matrix identical to the Hessian one. Since the Hessian matrix resulting from the Lagragian has entries which can be expressed as inner products of the embedded in some feature space $\mathcal{H}$ training points under the mapping $\phi' : \boldsymbol{x}_i \rightarrow \mathcal{H}$ it is proved to be positive semi-definite. Therefore $L(\boldsymbol{\alpha})$ has a unique maximum attained possibly for more than one different realisations of $\boldsymbol{\alpha}$ since the dual objective is not strictly concave in contrast to the strict convexity of the primal one.

We stressed before that we aspire to solve the primal optimisation problem through the formulation of the dual one which appears easier. It is not obvious, however, that the objective functions corresponding to the primal and dual problems have the same optimal value. In the sequel we will attempt to address this issue and give the conditions under which equality of the objectives may be possible. In general if $(\boldsymbol{w}, b)$ is a pair satisfying the constraints of the primal problem with objective $f(\boldsymbol{w}, b)$ and $\boldsymbol{\alpha}$ is a set of l values satisfying the constraints of the dual problem with objective $L(\boldsymbol{\alpha}) = \inf_{\boldsymbol{w},b} L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ it holds that

$$f(\boldsymbol{w}, b) \geq L(\boldsymbol{\alpha}) \ . \tag{3.10}$$

Under the restrictions imposed on $(\boldsymbol{w}, b)$ and $\boldsymbol{\alpha}$ to lie in the feasibility region of the primal and the dual problem, respectively the value of the dual objective is always bounded from above by the value of the primal. This relationship holding between the two formulations of the optimisation problem is known as the weak duality theorem [41]. This difference that exists between the $f(\boldsymbol{w}, b)$ and $L(\boldsymbol{\alpha})$ measures something that is known as the duality gap. If we come up with values $(\boldsymbol{w}^\star, b^\star)$ and $\boldsymbol{\alpha}^\star$ such that $f(\boldsymbol{w}^\star, b^\star) = L(\boldsymbol{\alpha}^\star)$ then the duality gap will vanish. Nevertheless, we still do not know if there are values of the primal and the dual variables for which equality in the objectives can be achieved. Let us denote by $g_i(\boldsymbol{w}, b) = 1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \leq 0$ the inequality constraints of the primal optimisation problem which in our case stem from the requirement for correct classification of the points by the canonical hyperplane. It is easily proved that since

$$L(\boldsymbol{\alpha}) = \inf_{\boldsymbol{w},b} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) \leq L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = f(\boldsymbol{w}, b) + \sum_{i=1}^{l} \alpha_i g_i(\boldsymbol{w}, b) \leq f(\boldsymbol{w}, b) \tag{3.11}$$

candidate values for the vanishing of the duality gap are surely the ones that minimise the primal Lagrangian since in this case the first inequality in (3.11) becomes an equality.

We will discuss very shortly the additional condition under which the second inequality becomes equality. This can happen for $\boldsymbol{\alpha}^\star$ solving the dual problem since this is the only value for which (3.10) could hold as an equality. By forcing $L(\boldsymbol{\alpha}^\star)$ to attain a maximum we are able to approach and possibly reach the minimum value $f(\boldsymbol{w}^\star, b^\star)$. The next theorem known as the strong duality theorem [41] provides us with the guarantees needed for the dual optimisation problem to have the same objective as the primal. The theorem states that

Theorem 3.2. *Given an optimisation problem with convex domain* $\Omega \subset \mathbf{R}^n$

$$\text{minimise}_{\boldsymbol{w},\text{b}}\ f(\boldsymbol{w}, b)\ ,$$

$$\text{subject to}\ \ g_i(\boldsymbol{w}, b) \leq 0\ \ \text{for all}\ i = 1, \ldots, l\ \ ,$$

where g_i *are affine functions, then the duality gap is zero.*

Apparently these rather mild conditions are satisfied by our primal optimisation problem. Given that the two problems can admit the same objective value for the solutions of the primal and the dual optimisation problem, (3.11) indicates that this can take place only if $\sum_{i=1}^{l} \alpha_i g_i(\boldsymbol{w}, b) = 0$. The Kuhn-Tucker theorem [36] states the conditions for a point $(\boldsymbol{w}^\star, b^\star)$ to be the optimum solution of the primal problem.

Theorem 3.3. *Given an optimisation problem like the one appearing in Theorem 3.2 the sufficient and necessary conditions for a regular point* $(\boldsymbol{w}^\star, b^\star)$ *to be the optimum solution when* $f(\boldsymbol{w}, b)$ *is a convex function with continuous first order partial derivatives and* g_i *affine constraints are the existence of dual variables* $\boldsymbol{\alpha}^\star \geq \mathbf{0}$ *such that* $(\boldsymbol{w}^\star, b^\star)$ *which belongs to the feasibility region of the primal problem* $(g_i(\boldsymbol{w}^\star, b^\star) \leq 0, i = 1 \ldots, l)$ *satisfies*

$$\frac{\partial L(\boldsymbol{w}^\star, b^\star, \boldsymbol{\alpha}^\star)}{\partial \boldsymbol{w}} = \mathbf{0},\ \ \frac{\partial L(\boldsymbol{w}^\star, b^\star, \boldsymbol{\alpha}^\star)}{\partial b} = \mathbf{0}\ \ ,$$
$$\alpha_i^\star g_i(\boldsymbol{w}^\star, b^\star) = 0, i = 1, \ldots, l\ \ .$$

In our case, binary classification with maximum margin, $f(\boldsymbol{w}, b) = \frac{1}{2}\|\boldsymbol{w}\|^2$ satisfies the requirements of convexity with respect to $\boldsymbol{w}$ and continuity of the first order derivatives. The l additional constraints taking the form

$$\alpha_i^\star \left[1 - y_i(\boldsymbol{w}^\star \cdot \boldsymbol{x}_i + b^\star)\right] = 0\ \ i = 1, \ldots, l$$

constitute the Karush-Kuhn-Tucker (KKT) complementarity conditions [29]. At each step of the optimisation procedure the intermediate values of $(\boldsymbol{w}, b)$ before solution is reached and the optimum $(\boldsymbol{w}^\star, b^\star)$ one can be inside the feasible region defined by some of the constraints and on the boundary surfaces of the feasible region regarding some others. In the former case we say that the corresponding constraints are inactive whereas in the latter the constraints are characterised as active ones. The dual variables linked

to constraints being inactive, i.e. holding for points $(\boldsymbol{x}_i, y_i)$ which are not lying on the canonical hyperplanes, assume the value $\alpha_i = 0$. On the contrary, for points which satisfy the relationship $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 1$ corresponding to active constraints the dual variables are allowed by the KKT conditions to satisfy $\alpha_i \geq 0$. As a consequence only these patterns participate eventually in the expansion of the solution weight vector $\boldsymbol{w}^\star$ in terms of the training points

$$\boldsymbol{w}^\star = \sum_{i=1}^{s} \alpha_i^\star y_i \boldsymbol{x}_i$$

where s designates the number of examples for which $\alpha_i^\star > 0$. The larger the value of a parameter $\alpha_i^\star$ is, the higher is the influence of the corresponding pattern on the solution. The examples connected to positive dual variables are called support vectors because the direction and the bias of the hyperplane can be exclusively determined by them. The rest of the examples have no impact on the determination of the hyperplane and if they were identified they could have been discarded at no cost regarding the approximation to the optimal solution.

3.4 Soft Margin Hyperplanes

Until now we made the assumption that the training set is linearly separable and we are seeking separation with maximum margin. By studying (2.43) yielding the probability that an unseen point is wrongly classified by the Δ-margin hyperplane we discover that maximum margin classification is not always the most favourable approach regarding generalisation. In particular we can fix a Δ-margin hyperplane with Δ exceeding the existing margin γ at the expense of a few margin errors if this can reduce the worst case prediction regarding the probability of a margin mistake. In the inseparable case we are left with no choice except considering a zone around a given hyperplane extending at a distance Δ inside the regions which characterise an example either as positive or as negative. The thickness of this zone works as a substitute of the margin which, of course, does not exist and will be called a margin as well. It should be clear from the context when we attribute to the margin its usual meaning and when its relaxed one. The success in the choice of Δ will be assessed after conducting margin queries on test data. The bound of (2.43) can also in this case provide us with some theoretical insights which if combined with the margin errors done on an independent test set can guide us through the procedure of specifying Δ. In order to treat inseparable datasets we have to make some modifications in the formulation of the original optimisation.

Ideally we would like to achieve a solution with a margin of at least Δ but also with the minimum possible number of mistakes. We give the opportunity to the constraints of the primal optimisation problem encountered earlier to hold as equalities by relaxing

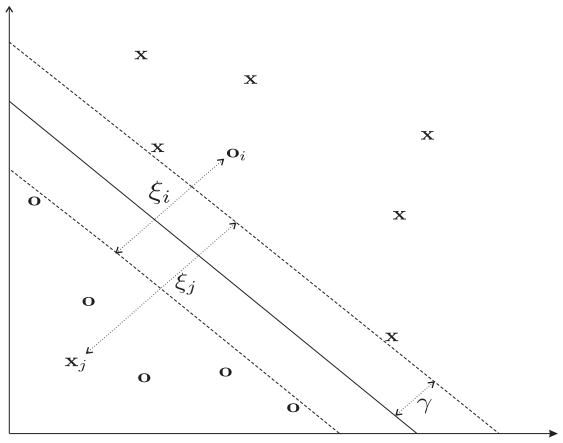

FIGURE 3.2: The slack variables for a classification problem.

them through the introduction of some non-negative quantities $\xi_i, i = 1, \ldots, l$

$$y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b \right) \geq 1 - \xi_i, \quad i = 1, \ldots, l \ . \tag{3.12}$$

The variables $\xi_i, i = 1, \ldots, l$ denoted compactly as $\boldsymbol{\xi}$, are called slack variables [55, 6] (Fig. 3.2). All we have to do now in order to complete the statement of our optimisation problem is to set the objective J to

$$J = \sum_{i=1}^{l} \xi_i^{\sigma} \ ,$$

where σ is a positive parameter tending to zero and impose the additional constraint on the margin written formally as $\|\boldsymbol{w}\|^2 \leq \Delta^{-2}$. The minimisation of the objective corresponds obviously to a minimisation performed on the number of margin mistakes.

Since the above optimisation problem is computationally intractable we allow for a relaxation of it. Specifically, we do not require the construction of a Δ-margin hyperplane but we pursue instead the margin maximisation in a criterion involving a penalty for the margin mistakes as well. Thus, in addition to the usual term $\frac{1}{2} \|\boldsymbol{w}\|^2$ a new term enters this criterion which is proportional to the previously mentioned objective with the parameter σ fixed to strictly positive values. This new concept of optimal hyperplane is called the soft-margin optimal hyperplane [13] and is determined by the minimisation of the following criterion J

$$J = \frac{1}{2} \|\boldsymbol{w}\|^2 + \frac{C}{\sigma} \sum_{i=1}^{l} \xi_i^{\sigma}$$

subject to the constraints (3.12). The free parameter C determines the trade-off between the maximisation of the margin and the minimisation of the sum of ξ_i^{σ}. In the sequel we distinguish two commonly encountered cases according to the value of σ.

First we treat the case where $\sigma = 1$ known as the 1-norm optimisation problem. The problem can be solved by applying techniques analogous to those used for finding the maximum margin in the separable case. In particular we construct the corresponding Lagrangian which after eliminating the primal variables $(\boldsymbol{w}, b, \boldsymbol{\xi})$ assumes a dual form $L(\boldsymbol{\alpha})$ identical with that of (3.9). The problem which we are asked to solve is the maximisation of $L(\boldsymbol{\alpha})$ under the constraints $\sum_{i=1}^{l} \alpha_i y_i = 0$ and

$$0 \leq \alpha_i \leq C \ .$$

Observe that the previous constraint forces the variables α_i to lie between 0 and C and that is why it is called the box constraint [13]. The KKT complementarity conditions for this problem are

$$\alpha_i \left[y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1 + \xi_i \right] = 0, \ \ i = 1, \ldots, l \ , \tag{3.13}$$

$$\xi_i(\alpha_i - C) = 0, \ \ i = 1, \ldots, l \ .$$

The first of them implies that the α_i's are zero for the inactive constraints whereas from the second we conclude that slack variables ξ_i have non-zero values only for $\alpha_i = C$. These α_i's correspond to examples which violate the margin requirement $1/\left\|\boldsymbol{w}\right\|$. The examples for which $0 < \alpha_i < C$ are lying at a distance $1/\left\|\boldsymbol{w}\right\|$ from the separating hyperplane and have zero slacks. Furthermore, they satisfy the constraints (3.12) as equalities enabling us to use them in order to determine the only unknown quantity, namely b given that the slacks vanish.

We turn now to the case where $\sigma = 2$ known as the 2-norm optimisation problem. Following the same procedure as before we obtain the dual Lagrangian $L(\boldsymbol{\alpha})$ free from the primal variables

$$L(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j - \frac{1}{2C} \sum_{i=1}^{l} \alpha_i^2$$

written equivalently as

$$L(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \left(\boldsymbol{x}_i \cdot \boldsymbol{x}_j + \frac{1}{C} \delta_{ij} \right) \ , \tag{3.14}$$

where δ_{ij} is Kronecker's δ. The KKT complementarity conditions assume the same form as in (3.13). From (3.14) we can see that $L(\boldsymbol{\alpha})$ remains essentially the same as the dual Lagrangian occuring in the maximisation of the margin in the separable case except for the term $\frac{1}{C}\delta_{ij}$. The term $(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$ as we have commented before corresponds to the linear kernel and could have been substituted by any kernel. We can incorporate $\frac{1}{C}\delta_{ij}$

into the kernel by weighting its diagonal by the quantity $1/C$ [52, 53, 14]

$$K'(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}') + \frac{1}{C}\delta_{\boldsymbol{x}\boldsymbol{x}'}$$

and solve the maximum margin problem.

Generalisation bounds involving the 1-norm or the 2-norm of the slack vector $\boldsymbol{\xi}$ were derived in [51, 52, 53].

3.5 Implementation Techniques

The maximum margin classification problem stated formally in terms of the optimisation theory leaves us with an objective function to be optimised subject at the same time to a set of constraints. There is a number of techniques that solve the quadratic optimisation problem associated with the formulation of the SVMs. However, implementation difficulties often arise especially if large datasets are involved. This is due to the fact that such techniques demand the kernel matrix to reside permanently in memory leading to a scaling of the memory requirements quadratic in the size of the dataset. To tackle the inefficiency of the standard quadratic programming packages employed for the SVM optimisation task alternative implementations were proposed which were proved in practice to be less time consuming and memory wasteful.

One of the first attempts in this direction was presented in [45]. The idea was to decompose the full optimisation problem into smaller ones and solve each one of them separately. Such techniques became known as decomposition methods. The algorithms following the decomposition approach attempt at each iteration to maximise the dual objective by adjusting only part of the dual variables. This implies that only a subset of the original constraints is taken into account which is called the active or working set while the rest of the variables remain fixed. This strategy is an improvement of an older technique known as chunking [64]. In chunking as the name suggests a chunk of the training set is used and training is performed using an off-the-shelf optimiser. The solution found to this subproblem is then used to test the validity of the constraints imposed on the rest of the dataset. The M points that violate the KKT conditions the most are added to the support vectors that determine the last hypothesis in order to form the new chunk on which the generic SVM will be trained. The parameter M which is fixed by the user determines the growth rate of the chunk. The procedure is repeated until no point is found for which the KKT conditions are violated. It is reasonable to expect that at every iteration new support vectors will be added leading to an increase of the chunk size without of course excluding the possibility that it may decrease as well. Nevertheless, there will be situations where this scenario will be impractical due to memory restrictions since no considerations are taken to keep the size of the subset that is provided to the optimiser within some limits. The problem of uncontrollable growth

of the training subsets leading to an increase in the demand for memory resources can be handled successfully in the decomposition methods since the size of the subset fed to the learning machine is kept fixed throughout the whole procedure. In the sequel we are going to study two widely known algorithms which make use of the decomposition strategy.

We begin with the Sequential Minimal Optimisation (SMO) algorithm devised by Platt [46]. The SMO is an extreme application of the decomposition method where the working set size is fixed to 2. This choice is justified by the fact that an optimisation problem consisting of only two points admits an analytical solution. From the equality constraint $\sum_i^l y_i \alpha_i = 0$ enforced by the presence of a bias term b in the expression of the hypothesis hyperplane it is obvious that if one of the dual variables is modified then another one should also change in order for the equality constraint to hold. Therefore, a working set of size 2 is the minimum active set that can be employed in terms of a decomposition method. We defer for later the description of the criteria according to which two of the Lagrange multipliers, say α_1 and α_2, are selected to form the active set and will turn for the moment to their update. Since the rest of the multipliers are kept fixed the equality constraint yields

$$y_1\alpha_1 + y_2\alpha_2 = y_1\alpha_1^{\text{old}} + y_2\alpha_2^{\text{old}} \ , \tag{3.15}$$

where α_1^{old} and α_2^{old} denote the values of the multipliers under consideration before the update takes place. Let us define the quantity $E_i = \boldsymbol{w} \cdot \boldsymbol{x}_i + b - y_i$. Notice that in the case of a wrong classification of the point $\boldsymbol{x}_i$ by the current hypothesis $-y_i E_i$ coincides with the value of the slack variable ξ_i. Then, we restate the dual objective function in a form consisting of terms depending only on one of the dual variables, say α_2, taking part in the active set and a constant term C' incorporating the contributions of the remaining multipliers which are unchanged during the iteration

$$L(\boldsymbol{\alpha}) = \frac{1}{2}\eta\alpha_2^2 + \left(y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta\alpha_2^{\text{old}}\right)\alpha_2 + C' \ ,$$

where $\eta = 2k_{12} - k_{11} - k_{22} = -(\boldsymbol{x}_1 - \boldsymbol{x}_2)\cdot(\boldsymbol{x}_1 - \boldsymbol{x}_2) = -\left\|\boldsymbol{x}_1 - \boldsymbol{x}_2\right\|^2 \leq 0$. The terms k_{11}, k_{12} and k_{22} denote the corresponding entries of the kernel matrix $\boldsymbol{K}$. Taking the derivative of $L(\boldsymbol{\alpha})$ with respect to α_2 we obtain

$$\frac{\partial L(\boldsymbol{\alpha})}{\partial \alpha_2} = \eta\alpha_2 + \left(y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta\alpha_2^{\text{old}}\right) \ . \tag{3.16}$$

Since $\frac{\partial^2 L(\boldsymbol{\alpha})}{\partial^2 \alpha_2} = \eta \leq 0$ the value of α_2 for which (3.16) vanishes gives an unconstrained maximum of the objective. Solving $\frac{\partial L(\boldsymbol{\alpha})}{\partial \alpha_2} = 0$ with respect to α_2 yields the new value of the multiplier

$$\alpha_2^{\text{new}} = \alpha_2^{\text{old}} + \frac{y_2(E_2^{\text{old}} - E_1^{\text{old}})}{\eta} \ .$$

We treat the general case of imperfect separation by use of the 1-norm penalty on the slack variables. The 2-norm optimisation problem can be handled as a special case of

the 1-norm one by adding a positive constant to the diagonal of the kernel matrix and choosing a parameter C large enough to prevent any of the multipliers from reaching it. In the case of the 1-norm soft margin optimisation α_2^{new} is not free to assume any value but is clipped when it exceeds the maximum allowable value V or when it is less than the minimum feasible one U. If $y_1 \neq y_2$

$$U = \max(0, \alpha_2^{\text{old}} - \alpha_1^{\text{old}}), \quad V = \min(C, C - \alpha_1^{\text{old}} + \alpha_2^{\text{old}}) \ ,$$

whereas if $y_1 = y_2$

$$U = \max(0, \alpha_1^{\text{old}} + \alpha_2^{\text{old}} - C), \quad V = \min(C, \alpha_1^{\text{old}} + \alpha_2^{\text{old}}) \ .$$

After α_2^{new} is clipped α_1^{new} can be obtained by (3.15)

$$\alpha_1^{\text{new}} = \alpha_1^{\text{old}} + y_1 y_2 \left(\alpha_2^{\text{old}} - \alpha_2^{\text{new}} \right) \ .$$

There is a number of heuristics according to which we can pick the two multipliers that provide a substantial contribution to the maximisation of the dual objective. First of all we have to select the first multiplier α_1 to be optimised from those that violate the KKT conditions. The example corresponding to α_1 is chosen preferably amongst the examples $(\boldsymbol{x}_i, y_i)$ which lie on the supporting hyperplanes $(0 < \alpha_i < C)$. If no such example can be found then one sweep through all the examples takes place. The second example is chosen based on the maximisation of the $|E_2 - E_1|$ criterion entering the update of α_2 initially among the non-boundary examples. If the search proves futile in the sense of not delivering a significant progress to the maximisation of the objective every such example is tested ignoring the criterion. If this fails too the search is repeated using the whole training set. The algorithm teminates if values of the multipliers are found which satisfy the KKT conditions to a specified tolerance level.

A special case of the SMO algorithm is the kernel Adatron [18, 19] the origins of which can be traced back in early work [3] on statistical physics of learning. If we set the bias b to zero or to a fixed value we can eliminate the equality constraint from the optimisation problem. This enables us to update only with respect to one example at a time according to the rule

$$\alpha_2^{\text{new}} = \alpha_2^{\text{old}} + \frac{y_2 E_2^{\text{old}}}{\eta} \ ,$$

where $\eta = -k_{22}$. The newly produced value α_2^{new} is subsequently clipped in order to satisfy the box constraints forcing it to lie in the interval $[0, C]$.

Another established algorithm falling into the category of decomposition methods is SVM^{light} [28]. In constrast to SMO where we encounter the smallest possible active set SVM^{light} employs a generalised version of the decomposition strategy involving q free variables which are subject to alterations during each iteration. The working set of size q is selected with criteria that yield a satisfactory progress towards maximisation of the

Lagrangian. To this end the algorithm of Zoutendjik [69] provides the q points which will form the working set for each iteration. A generic quadratic programming method such as the algorithm of Hildreth [26] or the interior point method [59] can play the role of the nucleus which will carry out the optimisation of this reduced working set.

In order to speed up the procedure one could try to guess which are the points of the original dataset which might not end up being support vectors and which are the ones that might end up being misclassified ($\alpha_i = C$) when the optimisation finishes. SVM^{light} employs a set of heuristics in order to identify at early stages of the optimisation procedure such points the multipliers of which are set to the value 0 or C, accordingly. Since the multiplyers α_j corresponding to these points remain fixed during the process we can eliminate them and draw active sets from the rest of the points. Of course, no mechanism can ensure from the beginning that these points will keep playing the same role till the end. For this reason the excluded variables are checked after convergence and if necessary optimisation on the full set is performed. This procedure which attempts to reduce the size of the original problem is known as shrinking. For the algorithm to terminate criteria based on the KKT optimality conditions are used.

Chapter 4

Incremental Algorithms

4.1 Introduction

The algorithms that we consider here are driven by their mistakes. By this we mean that their linear hypothesis is updated each time a training example fails (succeeds) to satisfy a certain classification (misclassification) condition. Mistake-driven algorithms are naturally placed in an online setting in which the machine examines only one example at a time. This distiguishes them from algorithms like SVMs which operate in the so-called batch mode according to which the whole dataset is known to the machine from the beginning. In the online setting the learning proceeds in trials. At each trial the algorithm is presented with an instance $\boldsymbol{x}_i$ and attempts to predict its label. If there is a discrepancy between the predicted and the true label y_i a mistake occurs which forces subsequently the hypothesis to be updated. Such algorithms do not reconsider from scratch their response to future instances on the basis of the examples made available to them until that point but, instead, they incrementally update their internal state. In the online setting there is no distinct training and testing phase as training can go on for ever. The goal is to minimise over a long sequence of trials the discrepancy occuring between the predicted and the true labels as this is measured by an appropriately chosen loss function. There are numerous papers following this scenario for analysing algorithms in this setting. Kivinen and Warmuth [32] present a comprehensive overview of techniques bounding the loss incurred by algorithms belonging both to the Perceptron [47] and the Winnow family [40]. A comparative study involving the Perceptron and the Winnow algorithms is provided in [31]. In the present work, on the contrary, we are interested in an adaptation of the online setting which is usually called the incremental setting. In this scenario the training phase is clearly distinguishable from the testing one. The dataset, finite in size, is cyclically presented to the algorithm in rounds (epochs) until no mistake occurs. The property of the algorithms to come up after a finite number of mistakes with a hypothesis from the concept class which perfectly explains the data signifies their abiltity to converge. The goal of the incremental scenario is to identify

whether an algorithm is able to converge and to place upper bounds on the number of mistakes made before a consistent classifier is constructed. On the basis of the form of their update rule the algorithms are classified into two broad categories, namely the additive and the multiplicative families. In the present thesis we will be exclusively concerned with the additive family of algorithms the most prominent representative of which is the Perceptron algorithm.

In the present chapter we review some well-known algorithms belonging to the additive family starting with algorithms aiming at a mere separation of the data and gradually moving towards algorithms able to provide solutions possessing large margins. No proofs of convergence or derivations of mistake bounds are provided here. Such an analysis of Perceptron-like large margin classifiers is the subject of the next chapter.

4.2 The Augmented Space

As discussed many times so far the parameters that control the choice of the hypothesis in the linear case are the weight vector $\boldsymbol{w}$ that determines the direction of the hyperplane and the bias b. Moreover, the distance of the hyperplane from the origin is $|b|/\|\boldsymbol{w}\|$. The linear function that performs the binary classification of the training instances $\boldsymbol{x}_k$ has the following form

$$f(\boldsymbol{x}_k) = \boldsymbol{w} \cdot \boldsymbol{x}_k + b \ .$$

In the case that $f(\boldsymbol{x}_k) > 0$, $\boldsymbol{x}_k$ belongs to the first class and is characterised as a positive example whereas if the inverse inequality holds $\boldsymbol{x}_k$ belongs to the second class and is characterised as a negative one. To each example a label y_k is associated taking values from the set $\{-1, 1\}$ which indicates the category to which it belongs. Notice that the function $f(\boldsymbol{x}_k)$ can be elegantly written as the inner product of the augmented weight vector $\boldsymbol{a} = [\boldsymbol{w}\ w_0]$ (where $w_0 = b/\rho_0$) with the augmented training instance $\boldsymbol{y}_k = [\boldsymbol{x}_k\ \rho_0]$ both belonging to a space augmented by one additional dimension [15] in which all instances have the additional coordinate ρ_0. If a reflection in the augmented space with respect to the origin of the negatively labelled $\boldsymbol{y}_k$'s is assumed we have $y_k f(\boldsymbol{x}_k) = \boldsymbol{a} \cdot \boldsymbol{y}_k > 0$ for all the training patterns. Thus, there is no need to discriminate between positive and negative examples and the use of the labels becomes obsolete.

Let us consider a hyperplane characterised by an augmented weight vector $\boldsymbol{a}$. The geometric margin of the dataset in the original space with respect to this hyperplane is $\min_k \{\boldsymbol{a} \cdot \boldsymbol{y}_k\}/\|\boldsymbol{w}\|$. The corresponding margin in the augmented space with respect to hyperplanes passing through the origin is $\min_k \{\boldsymbol{a} \cdot \boldsymbol{y}_k\}/\|\boldsymbol{a}\|$. Obviously, the above margin in the augmented space is smaller than the geometric margin in the original space since $\|\boldsymbol{a}\| \geq \|\boldsymbol{w}\|$.

4.3 The Perceptron Algorithm

The most well-known primal space iterative algorithm which is used for binary classification of linearly separable data is the Perceptron algorithm [47]. The Perceptron uses an additive rule to update the previous hypothesis each time a "mistake" occurs. At each trial a new instance is received by the algorithm which afterwards proceeds to a guess concerning its label based on the current hypothesis. Each time the classification (misclassification) condition $\boldsymbol{a}_t \cdot \boldsymbol{y}_k > 0$ ($\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq 0$) is violated (satisfied) the algorithm fails to make a successful guess. Such a mistake causes $\boldsymbol{a}$ to be updated in the following way

$$[\boldsymbol{w}_{t+1} \; w_{0t+1}] = [\boldsymbol{w}_t \; w_{0t}] + \eta y_k \, [\boldsymbol{x}_k \; \rho_0] \quad ,$$

where η denotes the learning rate and determines the degree to which the old solution is affected by the update. In the case of the Perceptron algorithm the learning rate plays no role if the weight vector is initialised to zero. This can be easily justified by the fact that the different values of the learning rate lead only to a rescaling of the hypothesis weight vector. This is true bearing in mind that the algorithm considers the training points as incorrectly classified based only on the direction of the weight vector. So for reasons of simplicity η might as well be set equal to 1. Here the subscript t is meant to indicate that the function f is t-dependent through the time dependence of the weight vector. Apparently, the values of $f_t(\boldsymbol{y}_k)$ that are considered correct are only the positive ones since, due to the reflection, $\boldsymbol{y}_k$ incorporates the information carried by the label. If the notation of the augmented space is adopted the update rule can be written compactly as

$$\boldsymbol{a}_{t+1} = \boldsymbol{a}_t + \boldsymbol{y}_k \quad . \tag{4.1}$$

Notice that if the original space is considered without any embedding of the data in the augmented space the Perceptron algorithm constructs hyperplanes that pass through the origin. In other words it is a zero-threshold algorithm which chooses solutions that possess zero bias. The trick of moving the data in an augmented space leads again to hyperplanes with zero bias in this space which nevertheless possess some bias in the original space. Observe that the hypotheses constructed by the Perceptron algorithm are linear combinations of the training examples with positive coefficients.

The update rule of the Perceptron algorithm can be considered as a result of a gradient descent procedure. The function on which the gradient search is based is called the Perceptron criterion function and in an online setting is described by

$$J_p(\boldsymbol{a}_t) = (-\boldsymbol{a}_t \cdot \boldsymbol{y}_k)_+ \quad , \tag{4.2}$$

where $(\alpha)_+ = \max(\alpha, 0)$. We can easily see that J_p is positive only for those $\boldsymbol{y}_k$ that are misclassified by $\boldsymbol{a}_t$ and measures their functional margin with respect to the current

hyperplane. The new value of $\boldsymbol{a}_{t+1}$ is calculated then by

$$\boldsymbol{a}_{t+1} = \boldsymbol{a}_t - \nabla_{\boldsymbol{a}_t} J_p(\boldsymbol{a}_t)$$

which coincides with (4.1).

An advantage of the Perceptron and related algorithms is the existence of upper bounds [44] on the number of updates which guarantee convergence in a finite number of steps. In the Perceptron case the bound on the number of mistakes is given by

$$t \leq \left(\frac{R}{\gamma_{\mathrm{d}}}\right)^2 \quad .$$

It depends solely on the maximum zero-threshold margin γ_{d} and the radius R of the minimum sphere centred at the origin that contains the training examples.

4.4 The Second-Order Perceptron Algorithm

Apart from the standard Perceptron algorithm there also exist other algorithms competing with it in the task of separating the data into two classes the bounds of which exhibit a more attractive behaviour depending on the case. As such a prominent example we will briefly review and comment on the properties of the Second-order Perceptron algorithm [10] which can be viewed as an incremental variant of the Whitened Perceptron [10]. The algorithm might be considered as an adaptation to online binary classification of the ridge regression and its analysis is inspired by an instance of Vovk's general aggregating algorithm [68] which is called the Forward algorithm in [4].

The Second-order Perceptron extends the well-known Perceptron algorithm by taking into consideration second order data information such as the correlation matrix of the training patterns. This algorithm exploits the inherent geometrical properties of the data. As the authors of [10] argue this algorithm has better mistake bounds in some cases and exhibits an improved generalisation ability compared to that of the classical Perceptron algorithm in the experiments conducted. The cases which are shown to be advantageous for the Second-order Perceptron include settings in which the data are not scattered evenly in the volume of a hypersphere centred at the origin but mostly reside along certain directions. These directions are linked to the dominant eigenvectors of the dataset correlation matrix. The bounds and the performance of the standard Perceptron algorithm are ruled by the quantity $\left(\frac{R}{\gamma_{\mathrm{d}}}\right)^2$. If X is a matrix the columns of which are the patterns contained in the sequence $\mathcal{S} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_K\}$, then the correlation matrix M is given by $M = XX^T$. We have changed notations from vector dot product to matrix notation. The superscript T designates the transpose of a vector or a matrix. The margin is defined by $\gamma_{\mathrm{d}} = \min_k \boldsymbol{u}^T \boldsymbol{y}_k$, where $\boldsymbol{u}$ is the unit norm weight

vector normal to the hyperplane which separates the data with margin γ_{d}. Since

$$K{\gamma_{\mathrm{d}}}^2 \leq \sum_{k=1}^{K} \left(\boldsymbol{u}^T \boldsymbol{y}_k\right)^2 \leq \sum_{k=1}^{K} \|\boldsymbol{y}_k\|^2 = Tr(M)$$

the worst case estimate of the time needed for convergence of the Perceptron algorithm is bounded from below by a quantity involving the trace of M. Moreover, points lying near the separating hyperplane and close to the surface of the minimum enclosing sphere influence the behaviour of the algorithm most. Consider the case in which an example lies near the optimum separating hyperplane and its length is close to R. Then, if this point is misclassified by the current hypothesis weight vector it is possible that the resulting new hypothesis moves further from the optimum direction $\boldsymbol{u}$ instead of approaching it. This happens because, although the Perceptron update rule forces the quantity $\boldsymbol{a}_t \cdot \boldsymbol{u}$ to increase at every step t, it does not guarantee the same for $\boldsymbol{u}_t \cdot \boldsymbol{u}$ which can in fact decrease. The implication of this is that the current hypothesis weight vector overshoots the feasible solution region, a situation in which the algorithm might be misled causing a slowing down of the convergence procedure.

As an example of how an appropriate transformation of the data could improve the time bound of an algorithm over the one of the standard Perceptron we shall mention the Whitened Perceptron algorithm. This variant differs from the standard one in that it needs the whole sequence of examples in advance in order to proceed to the following mapping of the data

$$\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_K\} \rightarrow \left\{M^{-1/2}\boldsymbol{y}_1, M^{-1/2}\boldsymbol{y}_2, \ldots, M^{-1/2}\boldsymbol{y}_K\right\} \ ,$$

where $M^{-1/2}$ is called the whitening transform. The existence of $M^{-1/2}$ is guaranteed due to the positive definiteness of M. This mapping results in a correlation matrix which is the identity matrix I_n with n being the dimensionality of the space of the patterns. The new instances even after the transformation remain linearly separable. The transformed data can be separated by a hyperplane with normal vector $\boldsymbol{z} = M^{1/2}\boldsymbol{u}$. This can be easily verified by constructing the product $\boldsymbol{z}^T M^{-1/2}\boldsymbol{y}_k = \boldsymbol{u}^T\boldsymbol{y}_k$ which is definitely positive since the optimal direction $\boldsymbol{u}$ classifies all $\boldsymbol{y}_k$s correctly with margin at least γ_{d}. Then, the margin that the transformed data possess with respect to $\boldsymbol{z}$ is at least $\gamma'_{\mathrm{d}} = \gamma_{\mathrm{d}} / \|\boldsymbol{z}\| = \gamma_{\mathrm{d}} / \left\|M^{1/2}\boldsymbol{u}\right\|$. Therefore an upper bound on the number of mistakes t for the Whitened Perceptron is given in analogy to the standard Perceptron by

$$t \leq \frac{1}{{\gamma_{\mathrm{d}}}^2} \max_k \left(\boldsymbol{y}_k^T M^{-1} \boldsymbol{y}_k\right) \left(\boldsymbol{u}^T M \boldsymbol{u}\right) \ .$$

Observing the bound we can say that this can be significantly smaller than $\left(\frac{R}{\gamma_{\mathrm{d}}}\right)^2$ if either the patterns are very correlated since $\boldsymbol{y}_k^T M^{-1}\boldsymbol{y}_k$ becomes then small or $\boldsymbol{u}$ is strongly correlated with a nondominant eigenvector of M.

The Second-order Perceptron algorithm can be considered as an incremental variant of the Whitened one. The algorithm maintains a n-row matrix X_{t-1} at time step $t-1$ which initially is empty. Each time t a new instance is received by the algorithm an augmented matrix $S_t = [X_{t-1}\ \boldsymbol{y}_k]$ is built. Since the negatively labelled points $\boldsymbol{y}_k$ are reflected in the augmented space with respect to the origin correct classification of $\boldsymbol{y}_k$ is designated by $\left(\boldsymbol{a}_t^T \boldsymbol{y}_k\right) > 0$, where $\boldsymbol{a}_t = \left(\alpha I_n + S_t S_t^T\right)^{-1} \boldsymbol{v}_{t-1}$. An α multiple of the identity matrix I_n is added to the correlation matrix $S_t S_t^T$ in order to bypass the impossibility to invert $S_t S_t^T$ in the case that it is singular. If there is a mismatch between the predicted and the true label of the pattern then an update of $\boldsymbol{v}_{t-1}$ similar in form to that of the standard Perceptron algorithm takes place $\boldsymbol{v}_t = \boldsymbol{v}_{t-1} + \boldsymbol{y}_k$ with $\boldsymbol{v}_0 = \boldsymbol{0}$ and X_t becomes $X_t = S_t$. The parameter α which cannot be deduced ahead of running affects considerably the performance of the algorithm. In a way it captures the information about the existence of specific directions along which the data are scattered and how well aligned is the solution vector to the normal to these directions. Notice that S_t contains only those patterns which are associated with mistaken past trials. This is also the main difference with the Forward algorithm in which S_t keeps track of all patterns seen so far by the algorithm.

We shall now turn to some theoretical properties of the Second-order Perceptron algorithm. It is proved that the number of mistakes made on a finite sequence of examples is bounded from above by

$$t \leq \inf_{\gamma>0} \min_{\|\boldsymbol{u}\|=1} \left(\frac{D_\gamma(\boldsymbol{u};\mathcal{S})}{\gamma} + \frac{1}{\gamma} \sqrt{\left(\alpha + \boldsymbol{u}^T X_t X_t^T \boldsymbol{u}\right) \sum_{i=1}^{n} \ln(1+\lambda_i/\alpha)} \right) \quad , \tag{4.3}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of $X_m X_m^T$ which consists only of the m points that were wrongly classified during the trials. The infimum with respect to the free parameter γ is taken in order to make the bound tighter. The quantity $D_\gamma(\boldsymbol{u};\mathcal{S}) = \sum_k D_\gamma(\boldsymbol{u};\boldsymbol{y}_k)$ is the sum of hinge losses $D_\gamma(\boldsymbol{u};\boldsymbol{y}_k) = \max\{0, \gamma - \boldsymbol{u}^T \boldsymbol{y}_k\}$ for every pattern in the sequence. If we consider the linearly separable case this bound, which can embody a repeated recycling through the patterns until convergence of the algorithm, reduces to

$$t \leq \left(\frac{1}{\gamma_{\mathrm{d}}} \sqrt{\left(\alpha + \boldsymbol{u}^T X_t X_t^T \boldsymbol{u}\right) \sum_{i=1}^{n} \ln(1+\lambda_i/\alpha)} \right) \quad . \tag{4.4}$$

The term $\lambda_u = \boldsymbol{u}^T X_t X_t^T \boldsymbol{u}$ can be small as in the case of the Whitened Perceptron if $\boldsymbol{u}$ is aligned with the eigenvector associated with $\min_i \lambda_i$. In the limit $\alpha \to \infty$ the bounds (4.3) and (4.4) give the known mistake bounds characterising the behaviour of the Perceptron in the inseparable and the separable case [22], respectively bearing in mind that $\sum_i \lambda_i \leq tR^2$.

Next we proceed to an investigation of the bounds (4.3) and (4.4) in order to find

the values of α which make the Second-order Perceptron more advantageous than the standard one. By bringing the aforementioned bounds in a form that can be directly compared with the ones holding for the Perceptron we may conclude that if $\lambda_u < \frac{r}{2}$ with $r = \frac{R^2 t}{n}$ then the choice $\alpha = r\lambda_u/(r - 2\lambda_u)$ favours the Second-order Perceptron. On the other hand if $\lambda_u \geq \frac{r}{2}$ there is no finite value of α for which the bound of the Second-order Perceptron becomes better and its performance approaches that of the standard Perceptron only for $\alpha \to \infty$.

In practice it is impossible to know λ_u since we ignore the optimal direction $\boldsymbol{u}$, so we can proceed alternatively in two ways. The first one is to let α increase with every mistaken trial t following the rule $\alpha_t = cR^2 t$ with $c > 0$ adjusted empirically. The linear dependence of α_t on t is justified from the following observation. The value of λ_u appearing in (4.3) and (4.4) is bounded from above and below (for the linearly separable case) by terms linear in the number of mistakes. It can be shown that the minimal speed of growth of the bounds can be ensured with α growing linearly with t. The second scenario eliminates α and enforces the need to introduce in the place of $\left(\alpha I_n + S_t S_t^T\right)^{-1}$ the pseudoinverse $\left(S_t S_t^T\right)^+$ which exists in all cases.

4.5 The Perceptron Algorithm with Margin

It is generally believed that the larger the margin of the dataset the greater is the generalisation of the learning machine [66, 50]. However, all the incremental algorithms that were discussed until now produce hyperplanes that ensure the mere separation of the data if the set is linearly separable. None of the algorithms with a misclassification condition of the form $\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq 0$ described so far were able to guarantee a fraction of the maximum margin that the dataset possesses. The most obvious and immediate way to enforce the generation of solutions that possess even a slight proportion of the margin is to change the condition.

The Perceptron algorithm in its original form cannot guarantee a minimum margin, a drawback which can be eliminated by a slight modification of its misclassification condition [15] to

$$\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq b,$$

where b is a positive constant (not to be confused with the bias which in this formulation is incorporated in the augmented vectors). In this case the condition not only demands correct classification of the points but also ensures that this is done with some positive margin. This allows the standard Perceptron algorithm to achieve at least a known fraction of the maximum margin [35] if certain requirements are fulfilled which is, however, achieved at a larger computational cost. These requirements depend on the parameters of the algorithm which are the threshold b of the misclassification condition and the

learning rate η of the update rule which in this case cannot be set to an arbitrary value with no effect on the solution produced.

4.6 Relaxation Procedures

Besides $J_p(\boldsymbol{a}_t)$ other criterion functions can be constructed, the minimisation of which can be achieved by the choice of a solution vector $\boldsymbol{a}_t$. A typical example is

$$J_q = (\boldsymbol{a}_t \cdot \boldsymbol{y}_k)^2 \ ,$$

where again $\boldsymbol{y}_k$ denotes the example that was misclassified by the zero threshold condition. The main advantage of this function over (4.2) is that its gradient is differentiable allowing $\boldsymbol{a}_t$ to approach more smoothly a solution vector. However, there is also the danger that $\boldsymbol{a}_t$ be trapped in the region near the hyperplane that just separates the data. Thus, in spite of the time spent the algorithm may converge to a solution possessing zero margin. We can avoid this awkward situation by modifying the criterion function to

$$J_r = \frac{1}{2}\frac{(\boldsymbol{a}_t \cdot \boldsymbol{y}_k - b)^2}{\|\boldsymbol{y}_k\|^2} \ .$$

In this case $\boldsymbol{y}_k$ corresponds to a training pattern that satisfies a misclassification condition $\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq b$ analogous to that of the Perceptron with margin. This modification forces the algorithm to keep running until a solution that possesses some margin is found. The division with the norm of the training pattern attenuates the effect that long examples lying near the separating hyperplane have on J_r. The gradient descent algorithm that results from this criterion function is known as the relaxation rule [1, 15] and its update is described by

$$\boldsymbol{a}_{t+1} = \boldsymbol{a}_t + \eta\frac{b - \boldsymbol{a}_t \cdot \boldsymbol{y}_k}{\|\boldsymbol{y}_k\|^2}\boldsymbol{y}_k$$

for those $\boldsymbol{y}_k$ that fail to exceed the predetermined margin b. This correction rule is justified since it has also a simple geometrical interpretation demonstrated in Fig 4.1. Let us examine only the quantity $r_k = (b - \boldsymbol{a}_t \cdot \boldsymbol{y}_k) / \|\boldsymbol{y}_k\|$. The term $\boldsymbol{a}_t \cdot \boldsymbol{y}_k$ appearing in r_k describes the locus of (the endpoints of) all the weight vectors that define hyperplanes which separate the pattern $\boldsymbol{y}_k$ with margin less than b. This locus is a hyperplane normal to $\boldsymbol{y}_k$ at a distance $(\boldsymbol{a}_t \cdot \boldsymbol{y}_k) / \|\boldsymbol{y}_k\|$ from the origin. The quantity r_k can be viewed as the distance that separates the weight vectors of the locus from the hyperplane which is the locus $\{\boldsymbol{a}_t : \boldsymbol{a}_t \cdot \boldsymbol{y}_k = b\}$ of the weight vectors that just satisfy the misclassification condition. Notice that this hyperplane is also normal to the misclassified point and parallel to the locus with margin less than b. Its distance from the origin is equal to $b/\|\boldsymbol{y}_k\|$. It is pretty obvious that if η is set to unity then with a single update the point can be corrected, thus "relaxing" the tension created by $\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq b$. If $\eta < 1$ more than one updates are needed to correct the point. The value of η controls the number of

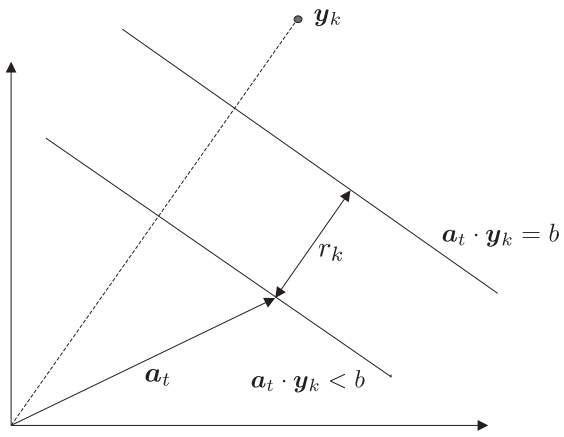

FIGURE 4.1: The weight vector $\boldsymbol{a}_t$ needs to cover the distance r_k in order to just violate the constraint $\boldsymbol{a}_t \cdot \boldsymbol{y}_k > b$. If $\eta = 1$ $\boldsymbol{a}_t$ moves exactly to the hyperplane $\boldsymbol{a}_t \cdot \boldsymbol{y}_k = b$.

the updates the algorithm should spend if it insisted repeatedly on classifying correctly that single pattern. The conditions $\eta < 1$ and $\eta > 1$ are known as underrelaxation and overrelaxation, respectively.

4.7 The Approximate Large Margin Algorithm

Although the Perceptron algorithm with margin has the property of achieving some margin which was missing from the original one, its apparent drawback is that it cannot theoretically guarantee convergence to solutions possessing the maximum margin. This shortcoming is not present in other algorithms which also build upon the idea of an analogous modified condition.

The Approximate Large Margin Algorithm (ALMA$_p$) [21], which belongs to the category of the p-norm Perceptrons [25], is accompanied with theoretical guarantees of achieving solutions of near maximum margin with respect to the p-norm. The subscript p which takes values in the interval [1,∞) is associated with the p-norm $\|\boldsymbol{x}_k\|_p$ of the training patterns. If the p-norm is used for $\boldsymbol{x}_k$ then its dual q is used for the norm for the weight vectors that the algorithm produces. The norms p and q are dual if they are related as follows

$$\frac{1}{p} + \frac{1}{q} = 1 \ . \tag{4.5}$$

For example, if the 1-norm is used for the weight vectors then its dual ∞-norm should be employed for $\|\boldsymbol{x}_k\|_p$ ($\|\boldsymbol{x}\|_\infty = \lim_{p\to\infty}(\sum_{i=1}^{n} |\boldsymbol{x}_i|^p)^{1/p} = \max_i |\boldsymbol{x}_i|$). The 2-norm as (4.5) indicates is self-dual. Notice that ALMA$_p$ as constructed by Gentile works only for data that lie on the surface of a unit sphere. However, the normalisation of the data to unit length certainly alters the original topology of the data and the value of the margin. ALMA$_p$ can approximate the maximum margin hypothesis within any desired level of accuracy by using parameters appropriately tuned. ALMA$_p$ is based on the model of

online learning with a series of mistaken trials determining the final weight vector. The initial value of $\boldsymbol{a}_t$ is set to zero. The misclassification condition is modified as in the case of the Perceptron algorithm with margin to accommodate a positive threshold and assumes the form

$$\boldsymbol{a}_t \cdot \boldsymbol{y}'_k \leq (1-\alpha)\gamma_t \ .$$

The misclassification condition involves the normalised instances which are denoted by $\boldsymbol{y}'_k = \boldsymbol{y}_k / \|\boldsymbol{y}_k\|_p$. The α parameter appearing in the condition is associated with the fraction of the margin attained by the algorithm while γ_t decreases with the number of mistakes made following the rule

$$\gamma_t = B\sqrt{p-1}\frac{1}{\sqrt{t}} \ .$$

B is a positive quantity fixed ahead of running by the user. The ALMA$_p$ algorithm employs an intermediate update rule each time a mistake occurs which is given by

$$\boldsymbol{a}'_t = g^{-1}(g(\boldsymbol{a}_t) + \eta_t \boldsymbol{y}'_k) \ .$$

The function g maps vectors with a given q-norm to vectors with the same p-norm. The mapping $g = (g_1, g_2, \ldots, g_n) : \mathbf{R}^n \to \mathbf{R}^n$, with n the dimensionality of the instance space, is given by

$$g_i(\boldsymbol{a}) = \frac{\mathrm{sign}(a_i)|a_i|^{q-1}}{\|\boldsymbol{a}\|_q^{q-2}} \ . \tag{4.6}$$

From (4.6) it is apparent that the function g_i modifies appropriately each one of the components a_i of the weight vector $\boldsymbol{a}$ in such a way that the resulting weight vector $g(\boldsymbol{a})$ has a p-norm equal to $\|\boldsymbol{a}\|_q$, where p and q are a pair of dual norms. The inverse transform $g^{-1}(\boldsymbol{a})$ is performed in an analogous manner and is obtained by substituting in (4.6) p in the place of q. We should mention that for $p = q = 2$ both $g(\boldsymbol{a})$ and $g^{-1}(\boldsymbol{a})$ yield the identity function. In such a case the intermediate update rule of ALMA$_2$ reduces to the one of the ordinary Perceptron. However, the learning rate η_t in contrast to that of the Perceptron is reduced explicitly with time according to

$$\eta_t = \frac{C}{\sqrt{p-1}}\frac{1}{\sqrt{t}} \ ,$$

where C is a positive parameter. The weight vector $\boldsymbol{a}'_t$ given in the intermediate update rule should be normalised to unit length whenever its norm exceeds unity. Thus, the final value $\boldsymbol{a}_{t+1}$ of the weight vector at the end of the trial is computed by

$$\boldsymbol{a}_{t+1} = \frac{\boldsymbol{a}'_t}{\max\{1, \|\boldsymbol{a}'_t\|_q\}} \ .$$

This normalisation has as a consequence that $\|\boldsymbol{a}_t\|_q$ never exceeds the boundaries of the unit ball. The norm of the weight vector in the Perceptron algorithm increases "on

the average" with time whereas the learning rate remains fixed to a constant value. As the result of the above, the patterns found misclassified later in the sequence have less influence on the construction of the hypothesis hyperplane than the misclassified examples that occurred earlier. The impact that the growth of the weight vector has on the significance of later updates in the determination of the solution is imitated in ALMA$_p$ by the explicit reduction of the learning rate while preventing the norm of the weight vector to exceed unity. By choosing the parameter values $B = \sqrt{8}/\alpha$ and $C = \sqrt{2}$ it can be proved that the parameter α determines the guaranteed fraction of the maximum directional margin achieved by ALMA$_p$, that is for any α a proportion $1 - \alpha$ of the maximum margin is secured. Specifically, if α is set to 1 it corresponds to an algorithm that does not require any margin at all while as α tends to zero the whole margin is recovered.

Experiments showed that ALMA$_2$ performs well in problem settings where the instance space is sparse and a dense target vector is involved. However, when the situation is reversed meaning that the target hyperplanes have only a few relevant components the ability of ALMA$_2$ to learn quickly degrades as the number of relevant attributes decreases. We can evade this deterioration in the performance if $p > 2$ is used. Gentile and Littlestone in [20] observed that by setting $p = 2 \ln n$ ALMA$_p$ is made similar to multiplicative algorithms such as the Winnow [40] and the Weighted Majority [37] algorithms. The use of this norm allows ALMA$_p$ to share the same benefits as the aforementioned algorithms with respect to the rate of convergence when pursuing sparse target vectors.

4.8 The Relaxed Online Maximum Margin Algorithm

Another very well-known algorithm addressing the problem of finding the maximum margin hyperplane is the aggressive variant of the Relaxed Online Maximum Margin Algorithm (ROMMA) [38] which is implemented in an incremental setting. Before we proceed to its presentation it would be useful to discuss what would be the ideal online maximum margin algorithm. Assuming linear separability of the dataset, the ideal online maximum margin algorithm considers a modified version of the objective function appearing in the SVM formulation. Specifically, instead of minimising the norm of the weight vector $\|\boldsymbol{a}_t\|$ subject to the constraints $(\boldsymbol{a}_t \cdot \boldsymbol{y}_k) \geq 1$ for all k in the sequence of examples, it would choose to minimise $\|\boldsymbol{a}_t\|$ subject to the constraints $(\boldsymbol{a}_t \cdot \boldsymbol{y}_k) \geq 1$ for all the patterns $\boldsymbol{y}_k$ that have previously been seen by the algorithm. The ROMMA algorithm in order to employ a simple update rule relaxes the above constraints in a fashion that attempts to preserve the online mode of the ideal maximum margin algorithm. Each time a new training example is presented to the algorithm the condition $\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq 0$ for incorrect classification of $\boldsymbol{y}_k$ is checked. In the case that a prediction

mistake occurs the algorithm proceeds to an update of the current hypothesis $\boldsymbol{a}_t$. Originally $\boldsymbol{a}_1 = \mathbf{0}$ and after the first trial which is certainly successful the algorithm sets $\boldsymbol{a}$ to be the shortest weight vector $\boldsymbol{a}_2$ that satisfies $\{\boldsymbol{a} : \boldsymbol{a} \cdot \boldsymbol{y}_{k_1} \geq 1\}$. Here $\boldsymbol{y}_{k_1}$ is the first misclassified example which coincides with the first example in the sequence. When a second prediction mistake occurs in connection with the second misclassified example, say $\boldsymbol{y}_{k_2}$, the new hypothesis follows again the ideal online maximum margin paradigm and $\boldsymbol{a}_3$ is determined as the shortest $\boldsymbol{a}$ which fulfills the combined constraint $\{\boldsymbol{a} : \boldsymbol{a} \cdot \boldsymbol{y}_{k_1} \geq 1\} \cap \{\boldsymbol{a} : \boldsymbol{a} \cdot \boldsymbol{y}_{k_2} \geq 1\}$. In order for the algorithm to keep at most two constraints at every step it proceeds differently from that point on. After the next prediction mistake $\boldsymbol{a}_4$ is determined to be the shortest $\boldsymbol{a}$ ensuring that

$$\{\boldsymbol{a} : \boldsymbol{a}_3 \cdot \boldsymbol{a} \geq \|\boldsymbol{a}_3\|\} \cap \{\boldsymbol{a} : \boldsymbol{a} \cdot \boldsymbol{y}_{k_3} \geq 1\}$$

holds. Generalising the procedure for any subsequent step t we call the constraint $H_t = \{\boldsymbol{a}_{t+1} : \boldsymbol{a}_t \cdot \boldsymbol{a}_{t+1} \geq \|\boldsymbol{a}_t\|\}$ the old constraint whereas $\{\boldsymbol{a}_{t+1} : \boldsymbol{a}_{t+1} \cdot \boldsymbol{y}_k \geq 1\}$ is referred to as the new constraint. Both of them must be satisfied together with the requirement for the shortest $\boldsymbol{a}$ possible at the t-th update involving the pattern $\boldsymbol{y}_k$ which has caused the prediction mistake. The old constraint gives a kind of inertia regarding changes in the solution since the feasible weight vectors are preferably chosen from solutions in the vicinity of the old one in order to keep their norm small. So the old constraint represents the tendency for conservativeness and determines the extent to which the old solution contributes to the new weight vector.

From the discussion above it is obvious that the algorithm needs only to solve a quadratic programming problem with two constraints. We will complement the description of the algorithm with an investigation of how an appropriate solution can be found satisfying the above constraints without resorting to quadratic optimisation. In fact, this will provide us with an efficient way of implementing the algorithm with the mere use of a simple update rule. It can be proved that both the new and the old constraint, with the latter holding after the first mistaken trial, are binding constraints. This means that they hold as equalities $\{\boldsymbol{a}_{t+1} : \boldsymbol{a}_{t+1} \cdot \boldsymbol{y}_k = 1\}$ and $\{\boldsymbol{a}_{t+1} : \boldsymbol{a}_t \cdot \boldsymbol{a}_{t+1} = \|\boldsymbol{a}_t\|\}$. Each of these constraints describes a hyperplane which is the locus of (the endpoints of) all the weight vectors that satisfy each one of the abovementioned equalities. Only the weight vector that ends at the intersection of both hyperplanes ensures the simultaneous satisfaction of both constraints. The update rule is given by the solution of the system consisting of the two constraints written compactly as

$$A\boldsymbol{a}_{t+1} = \boldsymbol{b} \ ,$$

where $A = \begin{pmatrix} \boldsymbol{a}_t^T \\ \boldsymbol{y}_k^T \end{pmatrix}$ and $\boldsymbol{b} = \begin{pmatrix} \|\boldsymbol{a}_t\| \\ 1 \end{pmatrix}$. It is presumed in this notation that the vector $\boldsymbol{a}_{t+1}$ multiplies separately each entry of the column vector A. Notice that in the general case the system is underdetermined since it has only two equations and n unknowns

with n being the dimensionality of the instance space. In this occasion the solution that minimises the squared error is also the one with the smallest norm and is given by

$$\begin{aligned}\boldsymbol{a}_{t+1} &= A^T \left(AA^T\right)^{-1} \boldsymbol{b} \\ &= \left(\frac{\|\boldsymbol{y}_k\|^2 \|\boldsymbol{a}_t\|^2 - (\boldsymbol{a}_t \cdot \boldsymbol{y}_k)}{\|\boldsymbol{y}_k\|^2 \|\boldsymbol{a}_t\|^2 - (\boldsymbol{a}_t \cdot \boldsymbol{y}_k)^2}\right) \boldsymbol{a}_t + \left(\frac{\|\boldsymbol{a}_t\|^2 (1 - (\boldsymbol{a}_t \cdot \boldsymbol{y}_k))}{\|\boldsymbol{y}_k\|^2 \|\boldsymbol{a}_t\|^2 - (\boldsymbol{a}_t \cdot \boldsymbol{y}_k)^2}\right) \boldsymbol{y}_k \ .\end{aligned}$$

Apart from the ROMMA algorithm that we just briefly analysed there exists a variant of it which claims to achieve a predefined δ approximation of the maximum margin ($0 < \delta \leq 1$). This variant is called aggressive ROMMA. Its name is justified by the fact that an update takes place not only after a prediction mistake but also after any trial in which $\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq 1 - \delta$. In this case in contrast to the simple ROMMA the old constraint may not be active. This means that there exist trials in which only $\boldsymbol{a}_{t+1} \cdot \boldsymbol{y}_k = 1$ has to be satisfied by the $\boldsymbol{a}_{t+1}$ with the shortest length and this is ensured if

$$\boldsymbol{a}_{t+1} = \frac{\boldsymbol{y}_k}{\|\boldsymbol{y}_k\|^2} \ . \tag{4.7}$$

The old constraint is not binding provided the inequalities

$$1 - \delta \geq \boldsymbol{a}_t \cdot \boldsymbol{y}_k \geq \|\boldsymbol{y}_k\|^2 \|\boldsymbol{a}_t\|^2 \tag{4.8}$$

are satisfied. This condition comes from the substitution of (4.7) in the old constraint which if we want it to be automatically satisfied (4.8) should hold. Otherwise, we apply the same update as in ROMMA.

4.9 The Maximal Margin Perceptron

The incremental algorithms that we discussed until now follow the approach of updating the weight vector that determines the separating hyperplane each time one of the training patterns satisfies the misclassification condition. The update performed on the weight vector changes its direction in a way that tends to leave the misclassified pattern on the correct side of the hyperplane with a margin at least as large as the one required by the condition. The condition is lowered as the algorithm progresses, usually due to an increasing norm of the weight vector, and this allows the algorithm to eventually converge to a solution possessing a positive margin. A different approach to the large margin classification problem would be to look for the segment corresponding to the minimum distance between the convex hulls of the positively and negatively labelled patterns. By approximately locating the points of the two convex hulls that are closest to each other and taking the line segment that connects them leads in a straightforward manner to the determination of the margin and to the construction of the solution

hyperplane by just performing an orthogonal bisection of this line segment. This is the underlying idea behind the Maximal Margin Perceptron (MMP) [34] which builds upon previous work in control theory [23, 43]. A similar direction is also adopted in [30] which, nevertheless, considers learning only in a batch mode and involves more than one patterns in the determination of the update rule.

The MMP algorithm receives, as usual, a training sample consisting of l examples $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)$ with a subset X^+ of them being positively labelled and a subset X^- being negatively labelled. After the training phase is over the algorithm outputs the parameters $(\boldsymbol{w}, b)$ which define the solution hyperplane. At every step t of the training procedure two weight vectors, namely $\boldsymbol{w}_t^+$ and $\boldsymbol{w}_t^-$, are kept which are the position vectors of two points belonging to the convex hulls $\mathrm{co}X^+$ and $\mathrm{co}X^-$ of the positive and negative examples, respectively. As such, these points are determined by convex combinations of the training patterns

$$\boldsymbol{w}_t^+ = \sum_{i \in I^+} \alpha_{t,i} \boldsymbol{x}_i \qquad \boldsymbol{w}_t^- = \sum_{i \in I^-} \alpha_{t,i} \boldsymbol{x}_i \ ,$$

where I^+ and I^- denote the set of indices of the positive and negative examples, respectively. For the variables $0 \leq \alpha_{t,i} \leq 1, i = 1, \ldots, l$ it additionally holds that $\sum_{i \in I^+} \alpha_{t,i} = \sum_{i \in I^-} \alpha_{t,i} = 1$. We denote by $\boldsymbol{\alpha}_t$ the whole sequence $\alpha_{t,i}, i = 1, \ldots, l$ for a fixed step t of the algorithm. Variables, instead, without the subscript t refer to their values after the algorithm has converged. The vectors $\boldsymbol{w}^+$ and $\boldsymbol{w}^-$ are called support centers of the hyperplane since the solution hyperplane is perpendicular to the weight

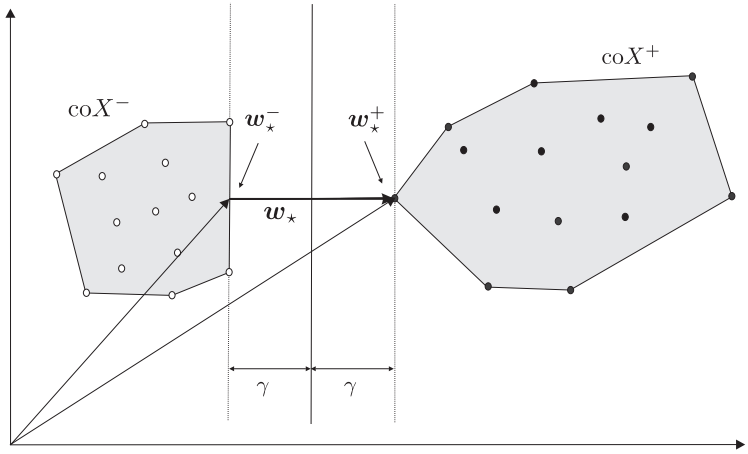

FIGURE 4.2: The support centers $\boldsymbol{w}_\star^- \in \mathrm{co}X^-$, $\boldsymbol{w}_\star^+ \in \mathrm{co}X^+$ determining the optimal hyperplane are the positions of the points of the convex hulls lying closest to each other. The optimal hyperplane is perpendicular to $\boldsymbol{w}_\star$ and bisects the line segment $[\boldsymbol{w}_\star^-, \boldsymbol{w}_\star^+]$.

vector $\boldsymbol{w}$

$$\boldsymbol{w} = \boldsymbol{w}^+ - \boldsymbol{w}^- = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x}_i \ .$$

The bias is also determined from the support centers through the relation

$$b = -\frac{\|\boldsymbol{w}^+\|^2 - \|\boldsymbol{w}^-\|^2}{2} \ .$$

As shown in Fig. 4.2 the optimal solution $\boldsymbol{w}_\star$ is given by the vector that connects the positions $\boldsymbol{w}_\star^- \in \mathrm{co}X^-$, $\boldsymbol{w}_\star^+ \in \mathrm{co}X^+$ of the points of the convex hulls lying closest to each other

$$\|\boldsymbol{w}_\star\| = \left\|\boldsymbol{w}_\star^+ - \boldsymbol{w}_\star^-\right\| = \min_{(\boldsymbol{w}^+,\boldsymbol{w}^-)\in(\mathrm{co}X^+\times\mathrm{co}X^-)} \left\|\boldsymbol{w}^+ - \boldsymbol{w}^-\right\| \ .$$

The margin achieved by a hyperplane $(\boldsymbol{w}, b)$ defined by the pair of support centers $(\boldsymbol{w}^+, \boldsymbol{w}^-)$ is given by $\gamma(\boldsymbol{w}^+, \boldsymbol{w}^-) \equiv \min_i y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b)/\|\boldsymbol{w}\|$. Obviously, $\gamma(\boldsymbol{w}^+, \boldsymbol{w}^-)$ is less than the margin $\gamma = \frac{\|\boldsymbol{w}_\star\|}{2} \leq \frac{\|\boldsymbol{w}\|}{2}$ corresponding to the optimal solution $\boldsymbol{w}_\star$. Combining all the above we have

$$\gamma(\boldsymbol{w}^+, \boldsymbol{w}^-) \leq \gamma \leq \frac{\|\boldsymbol{w}\|}{2} \ . \tag{4.9}$$

From (4.9) by taking into account the definition of $\gamma(\boldsymbol{w}^+, \boldsymbol{w}^-)$ we can easily deduce that for every suboptimum $\boldsymbol{w}_t$ it holds that

$$\max_i y_i \left(\boldsymbol{w}_t^{(y_i)} - \boldsymbol{x}_i\right) \cdot \boldsymbol{w}_t > 0 \ .$$

Notice that $\boldsymbol{w}_t^{(y_i)}$ refers to $\boldsymbol{w}_t^+$ ($\boldsymbol{w}_t^-$) if $y_i = +1$ ($y_i = -1$). Let us define the new quantities $G(\boldsymbol{\alpha}_t; i)$ and $H(\boldsymbol{\alpha}_t; i)$ which have to be reevaluated at every step

$$G(\boldsymbol{\alpha}_t; i) \equiv y_i \left(\boldsymbol{w}_t^{(y_i)} - \boldsymbol{x}_i\right) \cdot \boldsymbol{w}_t \ ,$$

$$H(\boldsymbol{\alpha}_t; i) \equiv \left\|\boldsymbol{w}_t^{(y_i)} - \boldsymbol{x}_i\right\| \ .$$

The margin achieved at a given step t is determined by

$$\gamma(\boldsymbol{w}_t^+, \boldsymbol{w}_t^-) = \frac{\|\boldsymbol{w}_t\|}{2} - \frac{\max_i G(\boldsymbol{\alpha}; i)}{\|\boldsymbol{w}_t\|} \ . \tag{4.10}$$

As the optimum margin is approached the second summand of (4.10) tends to zero.

The goal of the MMP algorithm is to find the points belonging to the convex hulls of the positive and negative examples that give rise to the minimum distance between the hulls. Since the distance at every step of the algorithm is described by $\|\boldsymbol{w}_t\|$ we need update rules that will ensure that

$$\|\boldsymbol{w}_{t+1}\|^2 \leq \|\boldsymbol{w}_t\|^2 - \theta^2 \ . \tag{4.11}$$

The term $-\theta^2$ on the r.h.s. of (4.11) was added in order to guarantee that at every step of the algorithm $\|\boldsymbol{w}_t\|$ will decrease. Inequality (4.11) holds at every step if we apply the following two mutually exclusive updating schemes to the sequence $\boldsymbol{\alpha}_t$. We consider MMP in an incremental setting which requires examining only one example $\boldsymbol{x}_i$ at a time. If $G(\boldsymbol{\alpha}_t; i) \geq 0$ and $\boldsymbol{x}_i \neq \boldsymbol{w}_t^{(y_i)}$ the *IncreaseStep* scheme is applied which involves setting

$$\alpha_{t+1,j} = \begin{cases} \tau_0 \delta_{ij} + (1-\tau_0)\alpha_{t,j} & \text{if } j \in I^{(y_i)} \\ \alpha_{t,j} & \text{otherwise} \end{cases} ,$$

where δ_{ij} is Kronecker's delta and $\tau_0 = \min\left(1, \frac{G(\boldsymbol{\alpha}_t;i)}{H^2(\boldsymbol{\alpha}_t;i)}\right) > 0$. If $G(\boldsymbol{\alpha}_t; i) < 0$ and $\boldsymbol{x}_i \neq \boldsymbol{w}_t^{(y_i)}$ then the *DecreaseStep* scheme is applied which involves setting

$$\alpha_{t+1,j} = \begin{cases} -\tau_0 \beta_{t,i} \delta_{ij} + (1+\tau_0\beta_{t,i})\alpha_{t,j} & \text{if } j \in I^{(y_i)} \\ \alpha_{t,j} & \text{otherwise} \end{cases} ,$$

where $\beta_{t,i} = \frac{\alpha_{t,i}}{1-\alpha_{t,i}}$ and $\tau_0 = \min\left(1, \frac{-G(\boldsymbol{\alpha}_t;i)}{\beta_{t,i}H^2(\boldsymbol{\alpha}_t;i)}\right) > 0$. For the *IncreaseStep* update the term θ^2 entering (4.11) is determined by

$$\theta^2 = \theta_{\text{incr}}(\boldsymbol{\alpha}_t; i) = \min\left(\frac{G^2(\boldsymbol{\alpha}_t; i)}{H^2(\boldsymbol{\alpha}_t; i)}, G(\boldsymbol{\alpha}_t; i)\right) > 0 ,$$

whereas in the case of a *DecreaseStep* it is given by

$$\theta^2 = \theta_{\text{decr}}(\boldsymbol{\alpha}_t; i) = \chi(\alpha_{t,i}) \min\left(\frac{G^2(\boldsymbol{\alpha}_t; i)}{H^2(\boldsymbol{\alpha}_t; i)}, -\beta_{t,i} G(\boldsymbol{\alpha}_t; i)\right) > 0$$

with $\chi(\alpha_{t,i}) = 1$ if $0 < \alpha_{i,t} < 1$ and $\chi(\alpha_{t,i}) = 0$ otherwise.

For the previous updates geometric justifications exist which we will try to briefly illuminate. We stressed earlier that at every step only one example $\boldsymbol{x}_i$ is examined. With respect to $\boldsymbol{x}_i$ one virtual example $\tilde{\boldsymbol{x}}_i$ can be defined by $\tilde{\boldsymbol{x}}_i \equiv \sum_{i \neq j \in I^{(y_i)}} \alpha_j \boldsymbol{x}_j / (1-\alpha_i)$ which will prove useful in the following. From its definition it is obvious that $\tilde{\boldsymbol{x}}_i$ is equivalent to the support center $\boldsymbol{w}_t^{(y_i)}$ with the contribution of $\boldsymbol{x}_i$ being removed and the rest of the multipliers being rescaled in order to ensure that their sum equals 1. In each updating scheme we consider the line segment $[\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i]$ connecting $\boldsymbol{x}_i$ with $\tilde{\boldsymbol{x}}_i$. Whenever an update takes place the new support center $\boldsymbol{w}_{t+1}^{(y_i)}$ is shifted towards either $\boldsymbol{x}_i$ or $\tilde{\boldsymbol{x}}_i$ depending on which of the two points the support center $\boldsymbol{w}_t^{(-y_i)}$ is closest to. More specifically, the new support center $\boldsymbol{w}_{t+1}^{(y_i)}$ is always the position of the trace of the normal projection of $\boldsymbol{w}_t^{(-y_i)}$ onto the line segment $[\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i]$ whenever this trace lies within the segment. Otherwise, the new support center is set to the endpoint of the segment lying closest to this trace. If the condition is fulfilled for *IncreaseStep* to take place then $\boldsymbol{w}_{t+1}^{(y_i)}$ approaches $\boldsymbol{x}_i$ and this causes the corresponding multiplier in the dual representation of $\boldsymbol{w}_t^{(y_i)}$ to increase up to 1, otherwise $\boldsymbol{w}_{t+1}^{(y_i)}$ moves away from $\boldsymbol{x}_i$ and this causes the corresponding multiplier to decrease and even vanish. It is apparent that when the

DecreaseStep updating scheme is active unlearning occurs for any example possessing margin greater than $\frac{\|\boldsymbol{w}_t\|}{2}$, a quality that is indicated by the validity of $G(\boldsymbol{\alpha}_t; i) < 0$. By unlearning we mean the weakening of the contribution of the specific example in the dual expansion of the corresponding support center. So, it is possible that the final expression of the support center may not depend on such examples. The above discussion is summarised in Fig. 4.3.

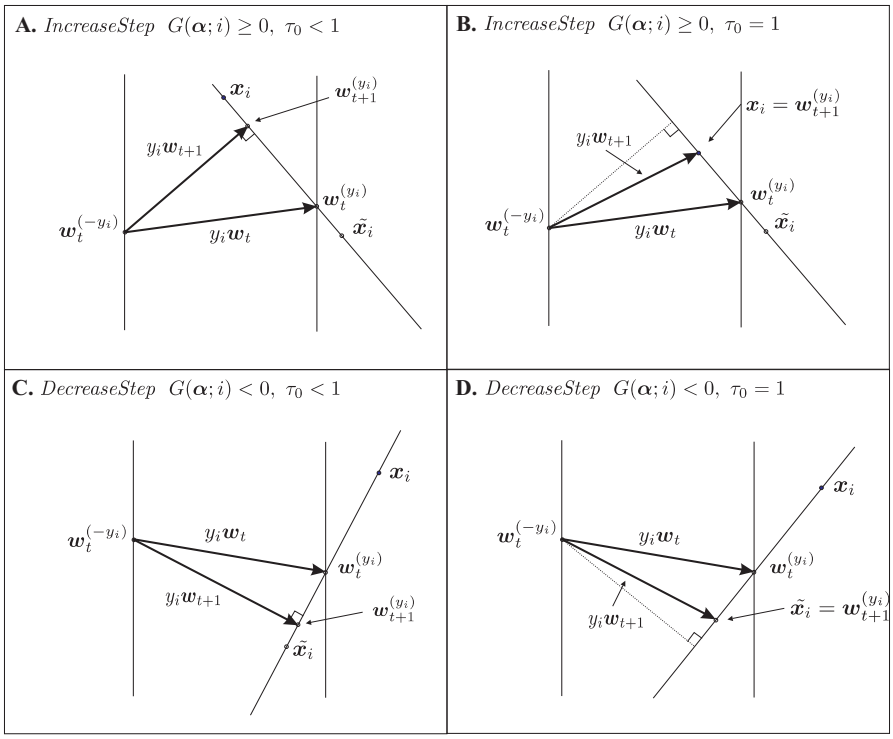

FIGURE 4.3: Geometric illustration of the basic updates. If the projection falls within the segment $[\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i]$ either the *IncreaseStep* (Figure **A.**) or the *DecreaseStep* (Figure **C.**) takes place with $\tau_0 < 1$. Otherwise $\boldsymbol{w}_{t+1}^{(y_i)}$ is shifted either to $\boldsymbol{x}_i$ (Figure **B.**) or $\tilde{\boldsymbol{x}}_i$ (Figure **D.**).

Now that we have defined the update rules we are in a position to state the basic MMP algorithm which can be applied in an incremental setting. The basic MMP algorithm examines every example in turn in order to confirm whether

$$\theta_{\text{incr}}(\boldsymbol{\alpha}_t; i) \geq \|\boldsymbol{w}_t\|^2 \theta_0^2 \tag{4.12}$$

holds true in which case an *IncreaseStep* update will follow or whether

$$\theta_{\text{decr}}(\boldsymbol{\alpha}_t; i) \geq \|\boldsymbol{w}_t\|^2 \theta_0^2 \tag{4.13}$$

is valid which will trigger a *DecreaseStep* update. In both cases $0 < \theta_0 < 1$ is a free parameter. The conditions (4.12) and (4.13) impose a minimum progress determined by $\theta_t^2 = \|\boldsymbol{w}_t\|^2 \theta_0^2$ towards the minimisation of the norm of the weight vector $\|\boldsymbol{w}_t\|$. The examples are recycled repeatedly until no example is found which satisfies either one of

the above conditions forcing subsequently the algorithm to exit. There is also a bound asserting that the algorithm will make no more updates than

$$t_{\max} \leq \frac{2}{\theta_0^2} \ln \frac{D}{2\gamma} \ ,$$

where $D = \max_{i,j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ is the diameter of the dataset. Upon exit of the algorithm the margin γ' attained satisfies

$$\gamma' \geq \gamma - D\theta_0 \ . \tag{4.14}$$

By adjusting the parameter θ_0 we can approximate the maximum margin to any desired level of accuracy and in the limit that $\theta_0 \to 0$ the whole margin is obtained. In order to have a fraction $1 - \epsilon$ of the maximum margin guaranteed ahead of running we need to set $\theta_0 = \frac{\gamma\epsilon}{D}$ as can be seen by substituting the above θ_0 back in (4.14) which yields

$$\gamma' \geq \gamma(1 - \epsilon) \ .$$

However, this is impossible due to the lack of knowledge regarding the value of the maximum margin γ ahead of running.

In an attempt to circumvent this drawback a new stopping criterion is introduced which allows the algorithm to exit after it has secured a given fraction of the maximum margin. From (4.10) and the fact that $\|\boldsymbol{w}_t\| \geq \|\boldsymbol{w}_\star\|$ we get

$$\frac{\gamma(\boldsymbol{w}^+, \boldsymbol{w}^-)}{\gamma} \geq \frac{2\gamma(\boldsymbol{w}^+, \boldsymbol{w}^-)}{\|\boldsymbol{w}_t\|} = 1 - \max_i \frac{2G(\boldsymbol{\alpha}_t; i)}{\|\boldsymbol{w}_t\|^2} \ .$$

From the previous inequality it is apparent that the quantity $\max_i \frac{2G(\boldsymbol{\alpha}_t;i)}{\|\boldsymbol{w}_t\|^2}$ plays the role of the approximation accuracy ϵ. This motivates the construction of an algorithm in which the examples are recycled as long as

$$\max_i \frac{2G(\boldsymbol{\alpha}_t; i)}{\|\boldsymbol{w}_t\|^2} > \epsilon \ .$$

We can choose to perform the updates in an online manner by upgrading the support centers for all points in turn irrespective of the validity of (4.12) and (4.13). Another possibility that might lead to a speed up of the convergence procedure is to perform updates in a batch manner to which we now turn. This version of the basic algorithm is known as the greedy MMP. In the greedy MMP, instead of naively processing each example presented to the algorithm sequentially, we can choose to implement any one of the three following scenarios: perform update 1) on the point that leads to the maximum decrease $d\boldsymbol{w}^2 \equiv \|\boldsymbol{w}_t\|^2 - \|\boldsymbol{w}_{t+1}\|^2$, 2) on the example with index $i_{t+1} = \arg\max_j(\theta_{\text{incr}}(\boldsymbol{\alpha}_t; j), \theta_{\text{decr}}(\boldsymbol{\alpha}_t; j))$ or 3) on the example that maximises $G(\boldsymbol{\alpha}_t; i)$. The main difference from the basic MMP algorithm is that we may require a specific fraction of the maximum margin ahead of running.

Chapter 5

Analysis of Perceptron-Like Large Margin Classifiers

5.1 Preliminaries

In what follows we make the assumption that we are given a training dataset which, even if initially not linearly separable can, by an appropriate feature mapping into a space of a higher dimension [2, 9, 66, 48, 54] be classified into two categories by a linear classifier. This higher dimensional space in which the patterns are linearly separable will be the considered space. By adding one additional dimension and placing all patterns in the same position at a distance ρ in that dimension we construct an embedding of our data into the augmented space. The advantage of this embedding is that the linear hypothesis in the augmented space becomes homogeneous. Thus, only hyperplanes passing through the origin in the augmented space need to be considered even for tasks requiring bias. Throughout our discussion a reflection with respect to the origin in the augmented space of the negatively labelled patterns is assumed in order to allow for a uniform treatment of both categories of patterns. Also, we use the notation $R = \max_k \|\boldsymbol{y}_k\|$ and $r = \min_k \|\boldsymbol{y}_k\|$, where $\boldsymbol{y}_k$ is the k^{th} augmented pattern. Obviously, $R \geq r \geq \rho$.

The relation characterising optimally correct classification of the training patterns $\boldsymbol{y}_k$ by a weight vector $\boldsymbol{u}$ of unit norm in the augmented space is

$$\boldsymbol{u} \cdot \boldsymbol{y}_k \geq \gamma_{\text{d}} \quad \forall k \ . \tag{5.1}$$

The quantity γ_{d}, which we call the maximum directional margin [56], is defined as

$$\gamma_{\text{d}} = \max_{\boldsymbol{u}':\|\boldsymbol{u}'\|=1} \min_k \{\boldsymbol{u}' \cdot \boldsymbol{y}_k\}$$

and is obviously bounded from above by r. The maximum directional margin determines the maximum distance from the origin in the augmented space of the hyperplane normal to $\boldsymbol{u}$ placing all training patterns on the positive side and coincides with the maximum margin in the augmented space with respect to hyperplanes passing through the origin if no reflection is assumed. In the determination of this hyperplane only the direction of $\boldsymbol{u}$ is exploited with no reference to its projection onto the original space. As a consequence the maximum directional margin is not necessarily realised with the same weight vector that gives rise to the maximum geometric margin γ in the original space. Notice, however, that the existence of a directional margin means that there exists a geometric margin at least as large as the directional one.

5.2 Relating the Directional to the Geometric Margin

It is possible to place an upper bound on the optimal geometric margin of a training set in terms of the optimal directional one, thereby leading to an estimate of the optimal geometric margin. If we denote by $\boldsymbol{a} = [\boldsymbol{w}\ w_0]$ a weight vector in the augmented space that classifies the patterns correctly and $\boldsymbol{y}_k = [\boldsymbol{x}_k\ \rho_0]$, the geometric margin $\gamma(\boldsymbol{a})$ of the set is

$$\gamma(\boldsymbol{a}) = \frac{1}{\|\boldsymbol{w}\|}\min_k \{\boldsymbol{w}\cdot\boldsymbol{x}_k + w_0\rho_0\} = \frac{1}{\|\boldsymbol{w}\|}\min_k \{\boldsymbol{a}\cdot\boldsymbol{y}_k\} = \frac{\|\boldsymbol{a}\|}{\|\boldsymbol{w}\|}\gamma_{\rm d}(\boldsymbol{a})\ , \tag{5.2}$$

where $\gamma_{\rm d}(\boldsymbol{a})$ is the corresponding directional margin. Notice that $|w_0|\rho/\|\boldsymbol{w}\|$ (with $\rho = |\rho_0|$) is the distance from the origin of the hyperplane normal to $\boldsymbol{w}$ which cannot exceed $R_x = \max_k \|\boldsymbol{x}_k\|$. Hence, $|w_0|/\|\boldsymbol{w}\| \leq R_x/\rho$. We now observe that

$$\|\boldsymbol{w}\| \leq \sqrt{\|\boldsymbol{w}\|^2 + w_0^2} = \|\boldsymbol{a}\|$$

but also

$$\|\boldsymbol{a}\| = \sqrt{\|\boldsymbol{w}\|^2 + w_0^2} \leq \|\boldsymbol{w}\|\sqrt{1 + \frac{R_x^2}{\rho^2}} = \|\boldsymbol{w}\|\,\frac{R}{\rho}$$

given that $R^2 = \rho^2 + R_x^2$. Then, (5.2) leads to

$$\gamma_{\rm d}(\boldsymbol{a}) \leq \gamma(\boldsymbol{a})$$

but also to

$$\gamma(\boldsymbol{a}) \leq \frac{R}{\rho}\gamma_{\rm d}(\boldsymbol{a})\ .$$

In the case that the weight vector $\boldsymbol{a}$ is the optimal one $\boldsymbol{a}_{\rm opt}$ maximising the geometric margin taking into account that $\gamma_{\rm d} = \max_{\boldsymbol{a}} \gamma_{\rm d}(\boldsymbol{a}) \geq \gamma_{\rm d}(\boldsymbol{a}_{\rm opt})$ and setting $\gamma \equiv \gamma(\boldsymbol{a}_{\rm opt})$ we obtain

$$\gamma \leq \frac{R}{\rho}\gamma_{\rm d}\ . \tag{5.3}$$

Moreover, we have that $\gamma = \gamma(\boldsymbol{a}_{\text{opt}}) = \max_{\boldsymbol{a}} \gamma(\boldsymbol{a}) \geq \max_{\boldsymbol{a}} \gamma_{\text{d}}(\boldsymbol{a}) = \gamma_{\text{d}}$, i.e.

$$\gamma_{\text{d}} \leq \gamma \ . \tag{5.4}$$

Combining (5.3) and (5.4) we finally get

$$1 \leq \frac{\gamma}{\gamma_{\text{d}}} \leq \frac{R}{\rho} \ . \tag{5.5}$$

In the limit $\rho \to \infty$, $R/\rho \to 1$ and from (5.5) $\gamma_{\text{d}} \to \gamma$. Thus, with ρ increasing the optimal directional margin γ_{d} approaches the optimal geometric one γ [56].

5.3 Taxonomy of Perceptron-Like Large Margin Classifiers

We concentrate on algorithms that update the augmented weight vector $\boldsymbol{a}_t$ by adding a suitable positive amount in the direction of the misclassified (according to an appropriate condition) training pattern $\boldsymbol{y}_k$. The general form of such an update rule is

$$\boldsymbol{a}_{t+1} = \frac{\boldsymbol{a}_t + \eta_t f_t \boldsymbol{y}_k}{N_{t+1}} \ , \tag{5.6}$$

where η_t is the learning rate which could depend (usually explicitly) on the number t of updates that have taken place so far and f_t an implicit function of the current step (update) t, possibly involving the current weight vector $\boldsymbol{a}_t$ and/or the current misclassified pattern $\boldsymbol{y}_k$, which we require to be positive and bounded, i.e.

$$0 < f_{\min} \leq f_t \leq f_{\max} \ . \tag{5.7}$$

We also allow for the possibility of normalising the newly produced weight vector $\boldsymbol{a}_{t+1}$ to a desirable length through a factor N_{t+1}. For the Perceptron algorithm η_t is constant, $f_t = 1$ and $N_{t+1} = 1$. Each time the predefined misclassification condition is satisfied by a training pattern the algorithm proceeds to the update of the weight vector. Thus, t (also called "time") keeps track of the number of updates which coincides with the number of mistakes (satisfactions of the misclassification condition). In the present section we adopt the convention of initialising t from 1.

A sufficiently general form of the misclassification condition is

$$\boldsymbol{u}_t \cdot \boldsymbol{y}_k \leq C(t) \ , \tag{5.8}$$

where $\boldsymbol{u}_t$ is the weight vector $\boldsymbol{a}_t$ normalised to unit length and $C(t) > 0$ if we require that the algorithm achieves a positive margin. If $\boldsymbol{a}_1 = \boldsymbol{0}$ we treat the first pattern in the sequence as misclassified. We distinguish two cases depending on whether $C(t)$ is

bounded from above by a strictly decreasing function of t which tends to zero or remains bounded from above and below by positive constants.

In the first case the minimum directional margin required by such a condition becomes lower than any fixed value provided t is large enough. Algorithms with such a condition have the advantage of achieving some fraction of the unknown existing margin provided they converge. Examples of such algorithms are the well-known standard Perceptron algorithm with margin [15, 35, 39, 56], ALMA$_2$ [21], CRAMMA [57] and MICRA [58]. In the standard Perceptron algorithm with margin the misclassification condition takes the form

$$\boldsymbol{u}_t \cdot \boldsymbol{y}_k \leq \frac{b}{\|\boldsymbol{a}_t\|} \ , \tag{5.9}$$

where $c_1(t-1) \leq \|\boldsymbol{a}_t\| \leq c_2\sqrt{t-1}$ with b, c_1, c_2 positive constants (see Section 5.5). In the ALMA$_2$ algorithm the misclassification condition is

$$\boldsymbol{u}_t \cdot \boldsymbol{y}_k \leq \frac{b}{\|\boldsymbol{a}_t\|\sqrt{t}} \ , \tag{5.10}$$

in which $c_3\sqrt{t-1} \leq \|\boldsymbol{a}_t\| \leq R$ with b, c_3 positive constants (see Section 5.6). Notice the striking similarity characterising the behaviour of $C(t)$ in the Perceptron and ALMA$_2$ algorithms.

In the second case the condition amounts to requiring a directional margin, assumed to exist, which is not lowered arbitrarily with t. In particular, if $C(t)$ is equal to a constant β [56] (5.8) becomes

$$\boldsymbol{u}_t \cdot \boldsymbol{y}_k \leq \beta \tag{5.11}$$

and successful termination of the algorithm leads to a solution with margin larger than β. Obviously, convergence is not possible unless $\beta < \gamma_{\mathrm{d}}$. In this case an organised search through the range of possible β values is necessary.

An alternative classification of the algorithms with the perceptron-like update rule (5.6) is according to the dependence on t of the "effective" learning rate [57]

$$\eta_{\mathrm{eff}\,t} \equiv \frac{\eta_t R}{\|\boldsymbol{a}_t\|} \tag{5.12}$$

which controls the impact that an update has on the current weight vector. More specifically, $\eta_{\mathrm{eff}\,t}$ determines the update of the direction $\boldsymbol{u}_t$

$$\boldsymbol{u}_{t+1} = \frac{\boldsymbol{u}_t + \eta_{\mathrm{eff}\,t} f_t \boldsymbol{y}_k / R}{\|\boldsymbol{u}_t + \eta_{\mathrm{eff}\,t} f_t \boldsymbol{y}_k / R\|} \ . \tag{5.13}$$

Again we distinguish two categories depending on whether $\eta_{\mathrm{eff}\,t}$ is bounded from above by a strictly decreasing function of t which tends to zero or remains bounded from above and below by positive constants. We do not consider the case that $\eta_{\mathrm{eff}\,t}$ increases

indefinitely with t since, as we will argue in the next section, we do not expect such algorithms to converge always in a finite number of steps.

In the first category belong the Perceptron algorithm with both the standard misclassification condition (5.9) and the fixed directional margin one of (5.11) [56] in which η_t remains constant and $\|\boldsymbol{a}_t\|$ is bounded from below by a positive linear function of t. Also to the same category belongs the ALMA$_2$ algorithm in which η_t decreases as $1/\sqrt{t}$ and MICRA. The similarity of the standard Perceptron with margin and ALMA$_2$ algorithms with respect to the behaviour of $\eta_{\mathrm{eff}\,t}$ is apparent if we consider the bounds obeyed by $\|\boldsymbol{a}_t\|$ in these two cases. Moreover, in both algorithms $\eta_{\mathrm{eff}\,t}$ is proportional to $C(t)$.

As an example of algorithms belonging to the second category we mention algorithms with the fixed directional margin condition of (5.11), $\|\boldsymbol{a}_t\|$ normalised to the target margin value β and fixed learning rate [56]. To the same category also belongs the CRAMMA algorithm.

In summary, the misclassification condition of a perceptron-like algorithm could, roughly speaking, either be "relaxed" with the number of updates (i.e. with t) or remain practically constant. Similarly, the effective learning rate could either be reduced with t or remain practically constant. Thus, we are led to four broad categories of algorithms. In subsequent sections we shall present an analysis of the algorithms mentioned above which are representative but sufficiently general cases belonging to all of these categories.

5.4 Stepwise Convergence

A very desirable property of an algorithm is certainly progressive convergence at each step meaning that at each update $\boldsymbol{u}_t$ moves closer to the optimal direction $\boldsymbol{u}$. This, of course, does not imply convergence to $\boldsymbol{u}$ even in an infinite number of steps. Let us assume that

$$\boldsymbol{u}_t \cdot \boldsymbol{u} > 0 \ . \tag{5.14}$$

Because of (5.14) the criterion for stepwise angle convergence [56], namely

$$\Delta \equiv \boldsymbol{u}_{t+1} \cdot \boldsymbol{u} - \boldsymbol{u}_t \cdot \boldsymbol{u} > 0 \ ,$$

can be equivalently expressed as a demand for positivity of $\mathcal{D}$

$$\mathcal{D} \equiv (\boldsymbol{u}_{t+1} \cdot \boldsymbol{u})^2 - (\boldsymbol{u}_t \cdot \boldsymbol{u})^2 = 2\eta_{\mathrm{eff}\,t} f_t (\boldsymbol{u}_t \cdot \boldsymbol{u}) \left\| \boldsymbol{u}_t + \eta_{\mathrm{eff}\,t} f_t \frac{\boldsymbol{y_k}}{R} \right\|^{-2} \frac{\mathcal{A}}{R} \ , \tag{5.15}$$

where use has been made of (5.13) and $\mathcal{A}$ is defined by

$$\mathcal{A} \equiv \boldsymbol{y}_k \cdot \boldsymbol{u} - (\boldsymbol{u}_t \cdot \boldsymbol{u})(\boldsymbol{y}_k \cdot \boldsymbol{u}_t) - \frac{\eta_{\mathrm{eff}\,t} f_t}{2R} \left(\|\boldsymbol{y}_k\|^2 (\boldsymbol{u}_t \cdot \boldsymbol{u}) - \frac{(\boldsymbol{y}_k \cdot \boldsymbol{u})^2}{(\boldsymbol{u}_t \cdot \boldsymbol{u})} \right) \ . \tag{5.16}$$

Positivity of $\mathcal{A}$ leads to positivity of $\mathcal{D}$ on account of (5.7) and (5.14) and consequently to stepwise convergence. Actually, as we shall prove shortly, convergence occurs in a finite number of steps provided that after some time $\mathcal{A}$ becomes bounded from below by a positive constant and $\eta_{\mathrm{eff}\,t}$ remains bounded from above and below by positive constants or decreases indefinitely but not faster than $1/t$. Following this rather unified approach one can examine whether an algorithm enters sooner or later the stage of stepwise convergence and terminates successfully in a finite number of steps [56].

We now prove that under the conditions just stated stepwise convergence leads to convergence in a finite number of steps. From our assumptions it follows that there is a t_0 such that for $t \geq t_0$, $\mathcal{A} \geq \mathcal{A}_1$ and $\eta_1/t \leq \eta_{\mathrm{eff}\,t} \leq \eta_2$, where $\mathcal{A}_1$, η_1, η_2 are positive constants. As a consequence, taking into account (5.7) and using the inequality $\|\boldsymbol{x}+\boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ we have

$$\|\boldsymbol{u}_t + \eta_{\mathrm{eff}\,t} f_t \boldsymbol{y}_k / R\| \leq 1 + \eta_2 f_{\max} \ .$$

Moreover, $(\boldsymbol{u}_t \cdot \boldsymbol{u})$ is an increasing function of t for $t \geq t_0$ due to the positivity of $\mathcal{A}$. All the above considerations lead to

$$\mathcal{D} \geq 2\frac{\eta_1}{t} f_{\min} (\boldsymbol{u}_{t_0} \cdot \boldsymbol{u}) (1 + \eta_2 f_{\max})^{-2} \frac{\mathcal{A}_1}{R} \equiv \frac{\mathcal{D}_1}{t} \ .$$

Thus,

$$(\boldsymbol{u}_{t+1} \cdot \boldsymbol{u})^2 - (\boldsymbol{u}_t \cdot \boldsymbol{u})^2 \geq \frac{\mathcal{D}_1}{t} \ .$$

A repeated application of the above inequality $(t+1-t_0)$ times yields

$$(\boldsymbol{u}_{t+1} \cdot \boldsymbol{u})^2 - (\boldsymbol{u}_{t_0} \cdot \boldsymbol{u})^2 \geq \mathcal{D}_1 \sum_{m=t_0}^{t} m^{-1} \geq \mathcal{D}_1 \int_{t_0}^{t} m^{-1} dm = \mathcal{D}_1 \ln \frac{t}{t_0}$$

from where the bound

$$t \leq t_0 \exp\{\mathcal{D}_1^{-1}\}$$

on the number of steps t until convergence is obtained.

Finally, we are going to discuss the behaviour of the algorithms with an effective learning rate growing indefinitely with t. We make the assumption that the algorithm converges after entering a stage of stepwise convergence in a finite number of steps. Given that the above assumption holds for t larger than a critical value t_{c}, $\boldsymbol{u}_t \cdot \boldsymbol{u}$ will increase sufficiently such that $\left(\|\boldsymbol{y}_k\|^2 (\boldsymbol{u}_t \cdot \boldsymbol{u}) - (\boldsymbol{y}_k \cdot \boldsymbol{u})^2 / (\boldsymbol{u}_t \cdot \boldsymbol{u}) \right)$ becomes positive. Multiplication of such a positive term with a sufficiently large $\eta_{\mathrm{eff}\,t}$ will then make $\mathcal{A}$ negative contradicting our assumption. In any other case the algorithm having an effective learning rate growing with t should converge "accidentally" in a finite number of steps without entering the stage of stepwise convergence.

5.5 Generic Perceptron-Like Algorithms with the Standard Margin Condition

We first consider an algorithm with the general update rule (5.6), constant learning rate $\eta_t = \eta$ and $N_{t+1} = 1$. The misclassification condition is assumed to have the standard form

$$\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq b \tag{5.17}$$

equivalent to (5.9) with the convention that t is initiallised from zero. We also assume that $\boldsymbol{a}_t$ is initially set to zero, i.e. $\boldsymbol{a}_0 = \boldsymbol{0}$. A pseudocode describing its implementation is given in Fig. 5.1.

As we shall see shortly (see (5.19) below) the length of the weight vector $\boldsymbol{a}_t$ is bounded from below by a positive linear function of t and consequently both the function $C(t) = b/\left\|\boldsymbol{a}_t\right\|$ of (5.8) (see (5.9)) and the effective learning rate $\eta_{\mathrm{eff}\,t} = \eta R/\left\|\boldsymbol{a}_t\right\|$ are bounded from above by a positive function linear in $1/t$ which vanishes in the limit $t \to \infty$.

In order to analyse such algorithms we calculate an upper bound on the number of updates until a solution is found, thereby extending Novikoff's theorem [44, 39]. From the difference of the inner products of $\boldsymbol{u}$ with the weight vector $\boldsymbol{a}_t$ at successive time steps we have

$$\boldsymbol{a}_{t+1} \cdot \boldsymbol{u} - \boldsymbol{a}_t \cdot \boldsymbol{u} = \eta f_t \boldsymbol{y}_k \cdot \boldsymbol{u} \geq \eta f_{\min} \gamma_{\mathrm{d}} \ . \tag{5.18}$$

A repeated application of (5.18) t times, taking into account that $\boldsymbol{a}_0 = \boldsymbol{0}$, implies

$$\left\|\boldsymbol{a}_t\right\| \geq \boldsymbol{a}_t \cdot \boldsymbol{u} \geq \eta f_{\min} \gamma_{\mathrm{d}} t \ , \tag{5.19}$$

which gives us a lower bound on $\left\|\boldsymbol{a}_t\right\|$. By calculating the difference of the squared norms of the weight vectors in consecutive steps we obtain

$$\left\|\boldsymbol{a}_{t+1}\right\|^2 - \left\|\boldsymbol{a}_t\right\|^2 = \eta^2 f_t^2 \left\|\boldsymbol{y}_k\right\|^2 + 2\eta f_t \boldsymbol{y}_k \cdot \boldsymbol{a}_t \leq \eta^2 f_{\max}^2 R^2 + 2\eta f_{\max} b \ . \tag{5.20}$$

Require: A linearly separable augmented training set with reflection assumed $S = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l)$
Input: η, b
Initialisation:
$t = 0$, $\boldsymbol{a}_0 = \boldsymbol{0}$

repeat until no update made within the **for** loop
  for $k = 1$ to l **do**
    if $\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq b$ **then**
      $\boldsymbol{a}_{t+1} = \boldsymbol{a}_t + \eta f_t \boldsymbol{y}_k$
      $t \leftarrow t + 1$

FIGURE 5.1: Generic Perceptron-like algorithm with the standard margin condition.

A repeated application of (5.20) t times, taking into account that $\boldsymbol{a}_0 = \mathbf{0}$, leads to the following upper bound on $\|\boldsymbol{a}_t\|$

$$\|\boldsymbol{a}_t\| \leq \sqrt{(\eta^2 f_{\max}^2 R^2 + 2\eta f_{\max} b)t} \ . \tag{5.21}$$

Combining (5.19) and (5.21) we get a squeezing relationship

$$\eta f_{\min}\gamma_d t \leq \boldsymbol{a}_t \cdot \boldsymbol{u} \leq \|\boldsymbol{a}_t\| \leq \sqrt{(\eta^2 f_{\max}^2 R^2 + 2\eta f_{\max} b)t}$$

from which the following bound on the number of updates for convergence is derived

$$t \leq t_{\mathrm{N}} \equiv \frac{f_{\max}^2}{f_{\min}^2}\frac{R^2}{\gamma_{\mathrm{d}}^2}\left(1 + \frac{2}{\eta f_{\max}}\frac{b}{R^2}\right) \ . \tag{5.22}$$

It would be interesting to estimate the margin that the algorithm is able to achieve. By substituting Novikoff's time t_{N} into (5.21) we obtain a time-independent upper bound on $\|\boldsymbol{a}_t\|$, namely $\|\boldsymbol{a}_{t_{\mathrm{N}}}\| = \eta f_{\min}\gamma_{\mathrm{d}} t_{\mathrm{N}}$, which, in turn, provides a lower bound $\beta_{\min} = b/\|\boldsymbol{a}_{t_{\mathrm{N}}}\|$ on the directional margin $\beta = b/\|\boldsymbol{a}_t\|$ appearing in (5.9). Thus, the guaranteed fraction (i.e. the lower bound on the fraction) of the maximum directional margin γ_{d} that the algorithm is able to achieve is

$$f_{\mathrm{b}} = \frac{\beta_{\min}}{\gamma_{\mathrm{d}}} = \frac{f_{\min}}{f_{\max}}\left(2 + \eta f_{\max}\frac{R^2}{b}\right)^{-1} = \frac{1}{2}\frac{f_{\min}}{f_{\max}}\left(1 - \frac{f_{\max}^2}{f_{\min}^2}\frac{R^2}{\gamma_{\mathrm{d}}^2}t_{\mathrm{N}}^{-1}\right) \ . \tag{5.23}$$

The above guaranteed fraction of γ_{d} acquires a maximum of $\frac{1}{2}\frac{f_{\min}}{f_{\max}} \leq \frac{1}{2}$ for $b \gg \eta R^2$ [35] or $t_{\mathrm{N}} \gg \frac{R^2}{\gamma_{\mathrm{d}}^2}$.

We next turn to a discussion of stepwise convergence. From (5.19) it is clear that for $t > 0$ (5.14) holds. Also, $\boldsymbol{y}_k \cdot \boldsymbol{u}$ appearing in $\mathcal{A}$ of (5.16) is definitely positive due to (5.1) whereas $\|\boldsymbol{a}_t\|$ becomes larger with time because of (5.19), thereby making eventually the term of $\mathcal{A}$ linear in $\eta_{\mathrm{eff}\,t}$ negligible. Moreover, (5.9) shows that the term $(\boldsymbol{u}_t \cdot \boldsymbol{u})(\boldsymbol{y}_k \cdot \boldsymbol{u}_t)$ is suppressed with time. Thus, for time t larger than a critical time t_{c} positivity of $\mathcal{A}$ and consequently of $\mathcal{D}$ (of (5.15)) is accomplished. By placing bounds on the terms in $\mathcal{A}$ using (5.1), (5.17) and (5.19) we obtain

$$\mathcal{A} \geq \gamma_{\mathrm{d}} - \frac{1}{2\eta f_{\min}\gamma_{\mathrm{d}} t}\left(2b + \eta f_{\max}(R^2 - \gamma_{\mathrm{d}}^2)\right) \ . \tag{5.24}$$

From the above inequality the time sufficient for stepwise convergence to begin is

$$t_{\mathrm{c}} \equiv \frac{1}{2}\frac{f_{\max}}{f_{\min}}\frac{R^2}{\gamma_{\mathrm{d}}^2}\left(1 + \frac{2}{\eta f_{\max}}\frac{b}{R^2} - \frac{\gamma_{\mathrm{d}}^2}{R^2}\right) < \frac{1}{2}\frac{f_{\min}}{f_{\max}}t_{\mathrm{N}} \ .$$

Therefore, unless the algorithm terminates much before Novikoff's time bound is exhausted it will definitely enter the phase of stepwise convergence. Given that, on account of (5.19) and (5.21), $\eta_{\mathrm{eff}\,t} = \eta R/\|\boldsymbol{a}_t\|$ for $t > 0$ is bounded from above and does

not decrease with t faster than $1/t$ stepwise convergence leads to an alternative proof of convergence in a finite number of steps.

In our analysis so far we required that the function f_t appearing in the update rule of (5.6) be bounded as in (5.7) in order for the algorithm to converge. However, although a positive and bounded f_t is a sufficient condition for convergence it is by no means a necessary one. To illustrate the above statement we consider the function

$$f_t = \frac{b_{\mathrm{u}} - \boldsymbol{a}_t \cdot \boldsymbol{y}_k}{\|\boldsymbol{y}_k\|^2}$$

with b_{u} even slightly larger than the parameter b of the misclassification condition of (5.17). This update is a minor modification of the well-known relaxation algorithm with margin [15] in which $b_{\mathrm{u}} = b$ so that f_t is allowed to vanish. We observe that

$$f_{\min} = \frac{b_{\mathrm{u}} - b}{R^2} > 0$$

leading to a lower bound on $\|\boldsymbol{a}_t\|$ as in (5.19). In contrast, no upper bound $f_{\max}$ exists since f_t can increase indefinitely if $\boldsymbol{a}_t \cdot \boldsymbol{y}_k$ is negative and large in absolute value. Nevertheless, we can obtain an upper bound on $\|\boldsymbol{a}_t\|$ as we shall see shortly. To this end we calculate the difference of the squared norms of the weight vectors in consecutive steps

$$\|\boldsymbol{a}_{t+1}\|^2 - \|\boldsymbol{a}_t\|^2 = 2\eta(2-\eta)\frac{b_{\mathrm{u}} - \boldsymbol{a}_t \cdot \boldsymbol{y}_k}{\|\boldsymbol{y}_k\|^2}\left\{\frac{b_{\mathrm{u}}}{2-\eta} - \frac{1}{2}(b_{\mathrm{u}} - \boldsymbol{a}_t \cdot \boldsymbol{y}_k)\right\} \tag{5.25}$$

and we notice that the r.h.s. of the above equation has a maximum with respect to the quantity $(b_{\mathrm{u}} - \boldsymbol{a}_t \cdot \boldsymbol{y}_k)$ for

$$(b_{\mathrm{u}} - \boldsymbol{a}_t \cdot \boldsymbol{y}_k)_{\max} = \frac{b_{\mathrm{u}}}{2-\eta} \ ,$$

provided $0 < \eta < 2$. Substituting this value in (5.25) we obtain

$$\|\boldsymbol{a}_{t+1}\|^2 - \|\boldsymbol{a}_t\|^2 \leq \frac{\eta}{(2-\eta)}\frac{b_{\mathrm{u}}^2}{r^2}$$

where $r = \min_k \|\boldsymbol{y}_k\|$. Then a repeated application of the above inequality t times, taking into account that $\boldsymbol{a}_0 = \boldsymbol{0}$, leads to the upper bound

$$\|\boldsymbol{a}_t\|^2 \leq \frac{\eta}{(2-\eta)}\frac{b_{\mathrm{u}}^2}{r^2}t \ . \tag{5.26}$$

Combining (5.19) and (5.26) we get the squeezing relationship

$$\eta f_{\min}\gamma_{\mathrm{d}} t \leq \boldsymbol{a}_t \cdot \boldsymbol{u} \leq \|\boldsymbol{a}_t\| \leq \frac{b_{\mathrm{u}}}{r}\sqrt{\frac{\eta}{2-\eta}t}$$

from which the following time bound for convergence is derived

$$t \leq \frac{1}{\eta(2-\eta)} \left(\frac{b_{\mathrm{u}}}{b_{\mathrm{u}} - b} \right)^2 \frac{R^4}{r^2 {\gamma_{\mathrm{d}}}^2} \ .$$

5.6 The ALMA$_2$ Algorithm

Here we briefly review the ALMA$_2$ algorithm [21] slightly modified in order to accommodate patterns which are not normalised to unit length. The update rule is the one of (5.6) with $f_t = 1$ and $\eta_t = \eta/\sqrt{t}$ assuming that the starting value of t is 1. The length of the newly produced weight vector $\boldsymbol{a}_{t+1}$ is subsequently normalised to R through the factor N_{t+1} only if it exceeds that value. The misclassification condition is given by

$$\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq \frac{b}{\sqrt{t}}$$

(which is equivalent to (5.10)) and the initial value of the weight vector is set to $\boldsymbol{a}_1 = \boldsymbol{0}$. A description of the algorithm in pseudocode appears in Fig. 5.2.

As we shall see shortly (see (5.30) below) for $t > 1$ the length of the weight vector $\boldsymbol{a}_t$ is bounded from below by a positive increasing function of t and consequently both the function $C(t) = b/(\|\boldsymbol{a}_t\| \sqrt{t})$ of (5.8) (see (5.10)) and the effective learning rate $\eta_{\mathrm{eff}\,t} = \eta R/(\|\boldsymbol{a}_t\| \sqrt{t})$ are bounded from above by a positive decreasing function of t which vanishes in the limit $t \to \infty$.

Next we proceed to a proof of convergence of the ALMA$_2$ algorithm along the lines of [21]. Taking the inner product of (5.6) with the optimal direction $\boldsymbol{u}$, employing (5.1)

Require: A linearly separable augmented training set with reflection assumed $S = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l)$
Define: $R = \max_k \|\boldsymbol{y}_k\|$
Input: α, η
Fix: $b_1 = \frac{1-\alpha}{\alpha} \left(\frac{1}{\eta} + \frac{3\eta}{2} \right) R^2$
Initialisation:
$t = 1$, $\boldsymbol{a}_1 = \boldsymbol{0}$

repeat until no update made within the **for** loop
  for $k = 1$ to l **do**
    if $\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq b_t$ **then**
      $\boldsymbol{a}'_t = \boldsymbol{a}_t + \frac{\eta}{\sqrt{t}} \boldsymbol{y}_k$
      $\boldsymbol{a}_{t+1} = \boldsymbol{a}'_t R / \max(R, \|\boldsymbol{a}'_t\|)$
      $t \leftarrow t + 1$
      $b_t = b_1/\sqrt{t}$

FIGURE 5.2: The Approximate Large Margin Algorithm ALMA$_2$.

and repeatedly applying the resulting inequality we have

$$R \geq \|\boldsymbol{a}_{t+1}\| \geq \boldsymbol{a}_{t+1} \cdot \boldsymbol{u} = \frac{\boldsymbol{a}_t \cdot \boldsymbol{u} + \eta_t \boldsymbol{y}_k \cdot \boldsymbol{u}}{N_{t+1}} \geq \frac{\boldsymbol{a}_t \cdot \boldsymbol{u}}{N_{t+1}} + \frac{\eta_t \gamma_{\mathrm{d}}}{N_{t+1}}$$
$$\geq \frac{\boldsymbol{a}_1 \cdot \boldsymbol{u}}{N_{t+1} N_t \cdots N_2} + \gamma_{\mathrm{d}} \left(\frac{\eta_t}{N_{t+1}} + \frac{\eta_{t-1}}{N_{t+1} N_t} + \cdots + \frac{\eta_1}{N_{t+1} N_t \cdots N_2} \right) \ . \tag{5.27}$$

For the normalisation factor N_{m+1} we can derive the inequality

$$N_{m+1}^{-1} \geq (1 + 2A/m)^{-\frac{1}{2}} \equiv r_m \ ,$$

where

$$A = \eta \left(\eta/2 + b/R^2 \right) \ ,$$

which if substituted in (5.27) and given that $\boldsymbol{a}_1 \cdot \boldsymbol{u} = 0$ leads to

$$\frac{R}{\gamma_{\mathrm{d}}} \geq \frac{\|\boldsymbol{a}_{t+1}\|}{\gamma_{\mathrm{d}}} \geq \sum_{m=1}^{t} \eta_m \prod_{j=m}^{t} r_j = \sum_{m=1}^{t} \eta_m r_m \prod_{j=m+1}^{t} r_j \geq \frac{\eta}{\sqrt{2A+t}} \sum_{m=1}^{t} \prod_{j=m+1}^{t} r_j$$
$$\geq \frac{\eta}{\sqrt{2A+t}} \sum_{m=1}^{t} \left(\frac{m}{t} \right)^A \geq \frac{\eta}{\sqrt{2A+t}} \frac{1}{t^A} \int_0^t m^A dm = \frac{\eta}{\sqrt{2A+t}} \frac{t}{A+1} \ . \tag{5.28}$$

In (5.28) we made use of

$$\eta_m r_m = \frac{\eta}{\sqrt{2A+m}} \geq \frac{\eta}{\sqrt{2A+t}}$$

and

$$-\ln \prod_{j=m+1}^{t} r_j = \frac{1}{2} \sum_{j=m+1}^{t} \ln \left(1 + \frac{2A}{j} \right) \leq \sum_{j=m+1}^{t} \frac{A}{j} \leq A \int_m^t \frac{dj}{j} = \ln \left(\frac{t}{m} \right)^A \ .$$

Thus, from (5.28) we obtain the relation

$$R \geq \|\boldsymbol{a}_{t+1}\| \geq \boldsymbol{a}_{t+1} \cdot \boldsymbol{u} \geq \frac{\eta \gamma_{\mathrm{d}}}{A+1} \frac{t}{\sqrt{2A+t}} \ . \tag{5.29}$$

From (5.29) one can easily show that $\|\boldsymbol{a}_t\|$ satisfies the inequalities

$$c_3 \sqrt{t-1} \leq \|\boldsymbol{a}_t\| \leq R \ , \tag{5.30}$$

where $c_3 = \eta \gamma_{\mathrm{d}} \left((A+1)\sqrt{2A+1} \right)^{-1}$, which we referred to in Section 5.3. From (5.29) one also gets the bound

$$t \leq t_{\mathrm{b}} \equiv \left(\frac{A+1}{\eta} \right)^2 \left(\frac{R}{\gamma_{\mathrm{d}}} \right)^2 + 2A \ . \tag{5.31}$$

Noticing that

$$\left(\frac{\gamma_{\rm d}R}{b}\right)^2(t_{\rm b}+1) \le \frac{R^4}{b^2}\left(\left(\frac{A+1}{\eta}\right)^2+2A+1\right) \le \frac{R^4}{b^2}\left(\frac{A+1}{\eta}+\eta\right)^2 \ ,$$

using (5.10) and given that $\|\boldsymbol{a}_t\| \le R$ one can show that the fraction of the directional margin achieved satisfies

$$f \ge \frac{1}{\gamma_{\rm d}}\frac{b/R}{\sqrt{t_{\rm b}+1}} \ge 1-\alpha \ ,$$

where $\alpha \in (0,1]$ is related to the parameters η and b as follows

$$\frac{b}{R^2} = \frac{1-\alpha}{\alpha}\left(\frac{1}{\eta}+\frac{3}{2}\eta\right) \ . \tag{5.32}$$

Thus, we have proved the following theorem.

Theorem 5.1. *The* ALMA_2 *algorithm of Fig. 5.2 converges after at most*

$$\frac{1}{4\alpha^2}\left(\frac{2}{\eta}+(3-2\alpha)\eta\right)^2\left(\frac{R}{\gamma_{\rm d}}\right)^2+\frac{1}{\alpha}\left(2+(3-2\alpha)\eta^2\right)-2$$

mistakes to a zero-threshold hyperplane with directional margin of at least $(1-\alpha)\gamma_{\rm d}$.

We can partially optimise the value of η by minimising the dominant term proportional to $(R/\gamma_{\rm d})^2$ on the r.h.s. of (5.31) keeping fixed either b or α. In the former case we obtain the value $\eta = \sqrt{2}$ (also employed in [21]) whereas in the latter we obtain the value

$$\eta = \sqrt{\frac{2}{3-2\alpha}} \ .$$

Once η is fixed b is determined from (5.32).

We next turn to a discussion of stepwise convergence. From (5.29) it is clear that for $t > 1$ (5.14) holds. Also, $\boldsymbol{y}_k \cdot \boldsymbol{u}$ appearing in $\mathcal{A}$ of (5.16) is definitely positive due to (5.1) whereas the term of $\mathcal{A}$ linear in $\eta_{{\rm eff}\,t}$ becomes eventually negligible due to the supression of the effective learning rate with time. Moreover, (5.10) shows that the term $(\boldsymbol{u}_t \cdot \boldsymbol{u})(\boldsymbol{y}_k \cdot \boldsymbol{u}_t)$ is suppressed with time. Thus, for time t larger than a critical time $t_{\rm c}$ positivity of $\mathcal{A}$ and consequently of $\mathcal{D}$ (of (5.15)) is accomplished. By placing bounds on the terms in $\mathcal{A}$ using (5.1), (5.10), (5.29) and the inequality $\sqrt{1+x} \le 1+x/2$ we obtain

$$\mathcal{A} \ge \gamma_{\rm d} - \frac{(A+1)\sqrt{2A+t-1}}{\eta\gamma_{\rm d}(t-1)\sqrt{t}}\left(b+\frac{1}{2}\eta R^2\right) \ge \gamma_{\rm d} - \frac{R^2(A+1)\sqrt{2A+t-1}}{\eta\gamma_{\rm d}(t-1)\sqrt{t-1}}\frac{A}{\eta}$$

$$= \gamma_{\rm d} - \frac{R^2(A+1)}{\eta^2\gamma_{\rm d}}\frac{A}{(t-1)}\sqrt{2\frac{A}{t-1}+1} \ge \gamma_{\rm d} - \frac{R^2(A+1)}{\eta^2\gamma_{\rm d}}\frac{A}{(t-1)}\left(\frac{A}{t-1}+1\right) \ .$$

The critical time $t_{\rm c}$ is determined by setting the r.h.s. of the last inequality equal to zero

$$\frac{A}{(t_{\rm c}-1)}\left(\frac{A}{t_{\rm c}-1}+1\right)=\frac{\eta^2}{A+1}\left(\frac{\gamma_{\rm d}}{R}\right)^2\equiv\omega\ . \tag{5.33}$$

Then, using the inequality $\sqrt{1+x}\geq 1+x/2-x^2/8$ for $x\geq 0$, we have

$$\frac{A}{(t_{\rm c}-1)}=\frac{1}{2}\left(-1+\sqrt{1+4\omega}\right)\geq\omega(1-\omega)\ .$$

One can easily see that ω cannot exceed 2. Let us first assume that $\omega\leq 1/2$. Then, using the inequality $(1-x)^{-1}\leq 1+2x$ for $0\leq x\leq 1/2$, we obtain

$$t_{\rm c}\leq\frac{(A+1)A}{\eta^2}\left(\frac{R}{\gamma_{\rm d}}\right)^2+2A+1=t_{\rm b}+1-\frac{1}{\omega}<t_{\rm b}\ .$$

In the case that $1/2\leq\omega\leq 1$

$$\frac{A}{(t_{\rm c}-1)}\geq\frac{1}{2}\left(-1+\sqrt{3}\right)\geq\frac{1}{3}$$

or

$$t_{\rm c}\leq 3A+1=2A+(A+1)\leq t_{\rm b}$$

since $A+1\leq\left(\frac{A+1}{\eta}\right)^2\left(\frac{R}{\gamma_{\rm d}}\right)^2$ given that $1/\omega\geq 1$. Finally, in the case that $1\leq\omega\leq 2$

$$\frac{A}{(t_{\rm c}-1)}\geq\frac{1}{2}\left(-1+\sqrt{5}\right)\geq\frac{1}{2}$$

from where

$$t_{\rm c}\leq 2A+1<t_{\rm b}$$

since $\frac{A+1}{\eta}\geq\left(\frac{\eta}{2}+\frac{1}{\eta}\right)\geq\sqrt{2}$. We conclude that unless the algorithm terminates much before the time bound $t_{\rm b}$ is exhausted, it will definitely enter the phase of stepwise convergence. Given that $\eta_{{\rm eff}\,t}=\eta R/(\|\boldsymbol{a}_t\|\sqrt{t})$ for $t>1$, on account of (5.29), is bounded from above by a positive constant and does not decrease with t faster than $1/t$, stepwise convergence leads to convergence in a finite number of steps.

5.7 The Constant Rate Approximate Maximum Margin Algorithm CRAMMA$^{\epsilon}$

We consider algorithms with the general update rule (5.6) and constant effective learning rate $\eta_{{\rm eff}\,t}=\eta_{\rm eff}$ in which the misclassification condition takes the form of (5.8) with

$$C(t)=\frac{\beta}{t^{\epsilon}}\ , \tag{5.34}$$

where β and ϵ are positive constants. We additionally make the choice $f_t = 1$. Finally, we assume that $\boldsymbol{u}_1$, the initial value of $\boldsymbol{u}_t$, is chosen to be the unit vector in the direction of the first training pattern in order for (5.14) to hold. This is true since according to the update (5.13) $\boldsymbol{u}_t$ is a linear combination with positive coefficients of the training patterns $\boldsymbol{y}_k$ all of which have positive inner products with the optimal direction $\boldsymbol{u}$ because of our assumption of the existence of a positive directional margin. Since the above $C(t)$ does not depend on $\|\boldsymbol{a}_t\|$ and given that the update (5.13) of $\boldsymbol{u}_t$ depends on $\|\boldsymbol{a}_t\|$ only through η_{eff} the algorithm does not depend separately on η_t and $\|\boldsymbol{a}_t\|$ but only on their ratio i.e. on η_{eff}.

Let us begin our analysis with a short discussion of stepwise convergence which will indicate the necessity of imposing constraints on η_{eff}. It is not difficult to see that (5.1), (5.8) and (5.34) lead to a lower bound on the relevant quantity $\mathcal{A}$ of (5.16)

$$\mathcal{A} \geq \gamma_{\text{d}} - \frac{\beta}{t^{\epsilon}} - \frac{1}{2}\eta_{\text{eff}} R \ . \tag{5.35}$$

By requiring that the r.h.s. of (5.35) be positive we derive a sufficient condition for the onset of stepwise convergence

$$\eta_{\text{eff}} < 2\frac{\gamma_{\text{d}} - \beta/t^{\epsilon}}{R} \ . \tag{5.36}$$

Taking the limit $t \to \infty$ on the r.h.s. of (5.36) we obtain the constraint

$$\eta_{\text{eff}} < 2\frac{\gamma_{\text{d}}}{R}$$

on the constant effective learning rate η_{eff} in order for the algorithm to eventually enter the stage of stepwise convergence. By regarding (5.36) as a constraint on t we obtain the critical time t_c

$$t_{\text{c}} \equiv \left(\frac{2\frac{\gamma_{\text{d}}}{R} - \eta_{\text{eff}}}{\beta} \right)^{-\frac{1}{\epsilon}}$$

for the onset of stepwise convergence. Obviously, the further η_{eff} is from $2\frac{\gamma_{\text{d}}}{R}$ the earlier the stepwise convergence will begin.

Although only η_{eff} plays a role we still prefer to think of it as arising from a weight vector normalised to the constant value β

$$\|\boldsymbol{a}_t\| = \beta$$

and a learning rate having a fixed value as well

$$\eta_t = \eta \ .$$

This is equivalent to normalising the weight vector to the variable margin value $C(t)$ that the algorithm is after, assuming at the same time a variable learning rate $\eta_t = \eta/t^{\epsilon}$. Having in mind the meaning of the directional margin in the augmented space the

Require: A linearly separable augmented training set with reflection assumed $S = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l)$
Define:
For $k = 1, \ldots, l$
$R = \max_k \|\boldsymbol{y}_k\|$, $\bar{\boldsymbol{y}}_k = \boldsymbol{y}_k / R$
Input: η_{eff}, $\beta_1 \, (= \beta / R)$
Initialisation:
$t = 1$, $\boldsymbol{u}_1 = \bar{\boldsymbol{y}}_1 / \|\bar{\boldsymbol{y}}_1\|$

repeat until no update made within the **for** loop
  for $k = 1$ to l **do**
    if $\boldsymbol{u}_t \cdot \bar{\boldsymbol{y}}_k \leq \beta_t$ **then**
      $\boldsymbol{u}_{t+1} = \frac{\boldsymbol{u}_t + \eta_{\text{eff}} \bar{\boldsymbol{y}}_k}{\|\boldsymbol{u}_t + \eta_{\text{eff}} \bar{\boldsymbol{y}}_k\|}$
      $t \leftarrow t + 1$
      $\beta_t = \beta_1 / t^{\epsilon}$

FIGURE 5.3: The Constant Rate Approximate Maximum Margin Algorithm CRAMMA$^{\epsilon}$.

geometric interpretation of such a choice becomes clear: The algorithm is looking for the hyperplane tangent to a hypersphere centred at the origin of the augmented space of radius $\|\boldsymbol{a}_t\|$ equal to the target margin value $C(t)$ which leaves all the augmented (with a reflection assumed) patterns on the positive side. The t-independent value of the learning rate η might also be considered as dependent on (a power of) β, i.e.

$$\eta = \eta_0 \left(\frac{\beta}{R} \right)^{1-\delta} \quad ,$$

where η_0, δ are positive constants. Thus, we are led to an effective learning rate which scales with $\frac{\beta}{R}$ like

$$\eta_{\text{eff}} = \eta_0 \left(\frac{\beta}{R} \right)^{-\delta} \quad . \tag{5.37}$$

The above algorithm with constant effective learning rate η_{eff} and misclassification condition described by (5.8) and (5.34) involving the power ϵ of t will be called the Constant Rate Approximate Maximum Margin Algorithm CRAMMA$^{\epsilon}$ [57] and is presented in Fig. 5.3. A justification of the qualification of the algorithm as an "Approximate Maximum Margin" one stems from the following theorem.

Theorem 5.2. *The* CRAMMA$^{\epsilon}$ *algorithm of Fig. 5.3 converges in a finite number of steps provided* $\eta_{\text{eff}} < \frac{1}{2} \left(\sqrt{1 + 8 \frac{\gamma_{\text{d}}}{R}} - 1 \right)$. *Moreover, if* η_{eff} *is given a dependence on* β *through the relation* $\eta_{\text{eff}} = \eta_0 \left(\frac{\beta}{R} \right)^{-\delta}$ *the directional margin* γ'_{d} *achieved by the algorithm tends in the limit* $\frac{\beta}{R} \to \infty$ *to the maximum one* γ_{d} *provided* $0 < \epsilon\delta < 1$.

Proof. Taking the inner product of (5.13) with the optimal direction $\boldsymbol{u}$ we have

$$\boldsymbol{u}_{t+1} \cdot \boldsymbol{u} = \left(\boldsymbol{u}_t \cdot \boldsymbol{u} + \eta_{\text{eff}} \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}}{R} \right) \left\| \boldsymbol{u}_t + \eta_{\text{eff}} \frac{\boldsymbol{y}_k}{R} \right\|^{-1} \quad . \tag{5.38}$$

Here

$$\left\| \boldsymbol{u}_t + \eta_{\text{eff}} \frac{\boldsymbol{y}_k}{R} \right\|^{-1} = \left(1 + 2\eta_{\text{eff}} \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}_t}{R} + \eta_{\text{eff}}^2 \frac{\|\boldsymbol{y}_k\|^2}{R^2} \right)^{-\frac{1}{2}}$$

from where, by using the inequality $(1+x)^{-\frac{1}{2}} \geq 1 - x/2$, we get

$$\left\| \boldsymbol{u}_t + \eta_{\text{eff}} \frac{\boldsymbol{y}_k}{R} \right\|^{-1} \geq 1 - \eta_{\text{eff}} \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}_t}{R} - \eta_{\text{eff}}^2 \frac{\|\boldsymbol{y}_k\|^2}{2R^2} \ .$$

Then, (5.38) becomes

$$\boldsymbol{u}_{t+1} \cdot \boldsymbol{u} \geq \left(\boldsymbol{u}_t \cdot \boldsymbol{u} + \eta_{\text{eff}} \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}}{R} \right) \left(1 - \eta_{\text{eff}} \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}_t}{R} - \eta_{\text{eff}}^2 \frac{\|\boldsymbol{y}_k\|^2}{2R^2} \right) \ .$$

Thus, we obtain for $\Delta \equiv \boldsymbol{u}_{t+1} \cdot \boldsymbol{u} - \boldsymbol{u}_t \cdot \boldsymbol{u}$

$$\frac{R}{\eta_{\text{eff}}} \Delta \geq \boldsymbol{y}_k \cdot \boldsymbol{u} - (\boldsymbol{u}_t \cdot \boldsymbol{u})(\boldsymbol{y}_k \cdot \boldsymbol{u}_t) - \frac{\eta_{\text{eff}}}{2R} \left(\|\boldsymbol{y}_k\|^2 \boldsymbol{u}_t \cdot \boldsymbol{u} + 2(\boldsymbol{y}_k \cdot \boldsymbol{u})(\boldsymbol{y}_k \cdot \boldsymbol{u}_t) \right) - \frac{\eta_{\text{eff}}^2}{2R^2} \|\boldsymbol{y}_k\|^2 \boldsymbol{y}_k \cdot \boldsymbol{u} \ .$$

By employing (5.1), (5.14) and (5.34) we get a lower bound on Δ

$$\Delta \geq \eta_{\text{eff}} \left(\frac{\gamma_{\text{d}}}{R} - \frac{\eta_{\text{eff}}}{2} - \frac{\eta_{\text{eff}}^2}{2} \right) - \eta_{\text{eff}} (1 + \eta_{\text{eff}}) \frac{\beta}{R} t^{-\epsilon} \ . \tag{5.39}$$

From the misclassification condition it becomes obvious that convergence of the algorithm is impossible as long as $\beta / t^{\epsilon} > \gamma_{\text{d}}$. Therefore we may assume that

$$t > t_0 \equiv \left(\frac{\beta}{\gamma_{\text{d}}} \right)^{\frac{1}{\epsilon}} \ . \tag{5.40}$$

A repeated application of (5.39) $(t - [t_0])$ times yields

$$\boldsymbol{u}_{t+1} \cdot \boldsymbol{u} - \boldsymbol{u}_{[t_0]+1} \cdot \boldsymbol{u} \geq \eta_{\text{eff}} \left(\frac{\gamma_{\text{d}}}{R} - \frac{\eta_{\text{eff}}}{2} - \frac{\eta_{\text{eff}}^2}{2} \right) (t - [t_0]) - \eta_{\text{eff}} (1 + \eta_{\text{eff}}) \frac{\beta}{R} \sum_{m=[t_0]+1}^{t} m^{-\epsilon}$$

with $[t_0]$ denoting the integer part of t_0. By employing the inequality

$$\sum_{m=[t_0]+1}^{t} m^{-\epsilon} \leq \int_{t_0}^{t} m^{-\epsilon} dm + t_0^{-\epsilon} = \frac{t^{1-\epsilon} - t_0^{1-\epsilon}}{1-\epsilon} + t_0^{-\epsilon}$$

and taking into account (5.14) we finally obtain

$$1 \geq \eta_{\text{eff}} \left(\frac{\gamma_{\text{d}}}{R} \right) \chi (t - t_0) - \eta_{\text{eff}} (1 + \eta_{\text{eff}}) \frac{\beta}{R} \frac{\left(t^{1-\epsilon} - t_0^{1-\epsilon} \right)}{1-\epsilon} - \omega \ . \tag{5.41}$$

Here

$$\chi \equiv \left(1 - \frac{\eta_{\text{eff}}}{2} (1 + \eta_{\text{eff}}) \frac{R}{\gamma_{\text{d}}} \right) \quad \text{and} \quad \omega \equiv \eta_{\text{eff}} (1 + \eta_{\text{eff}}) \frac{\gamma_{\text{d}}}{R} \ .$$

Let us define the new variable $\tau \geq 0$ through the relation

$$t = t_0 (1 + \tau) = \left(\frac{\beta}{\gamma_{\text{d}}} \right)^{\frac{1}{\epsilon}} (1 + \tau) \ . \tag{5.42}$$

In terms of τ (5.41) becomes

$$\frac{1}{\eta_{\text{eff}}}\left(\frac{\beta}{R}\right)^{-\frac{1}{\epsilon}}\left(\frac{\gamma_{\text{d}}}{R}\right)^{\left(\frac{1}{\epsilon}-1\right)}(1+\omega)\geq\chi\tau-(1+\eta_{\text{eff}})\,\frac{(1+\tau)^{1-\epsilon}-1}{1-\epsilon}\ . \tag{5.43}$$

Let $g(\tau)$ be the r.h.s. of the above inequality. Since $\chi>0$, given that η_{eff} obeys the constraint $\eta_{\text{eff}}<\frac{1}{2}\left(\sqrt{1+8\frac{\gamma_{\text{d}}}{R}}-1\right)$, it is not difficult to verify that $g(\tau)$ (with $\tau\geq 0$) is unbounded from above and has a single extremum, actually a minimum, at $\tau_{\min}=(1+\eta_{\text{eff}})^{\frac{1}{\epsilon}}\chi^{-\frac{1}{\epsilon}}-1>0$ with $g(\tau_{\min})<0$. Moreover, the l.h.s. of (5.43) is positive. Therefore, given that $g(0)=0$ there is a single value τ_{b} of τ where (5.43) holds as an equality which provides an upper bound on τ

$$\tau\leq\tau_{\text{b}} \tag{5.44}$$

satisfying $\tau_{\text{b}}>\tau_{\min}>0$. Combining (5.42) and (5.44) we obtain the bound on the number of updates

$$t\leq t_{\text{b}}\equiv\left(\frac{\beta}{\gamma_{\text{d}}}\right)^{\frac{1}{\epsilon}}(1+\tau_{\text{b}}) \tag{5.45}$$

proving that the algorithm converges in a finite number of steps. From (5.45) and taking into account the misclassification condition we obtain a lower bound $\beta/(t_{\text{b}}+1)^{\epsilon}$ on the margin γ'_{d} achieved. Thus, the fraction f of γ_{d} that the algorithm achieves satisfies

$$1\geq f\equiv\frac{\gamma'_{\text{d}}}{\gamma_{\text{d}}}\geq f_{\text{b}}\equiv\frac{\beta/\gamma_{\text{d}}}{(t_{\text{b}}+1)^{\epsilon}}=\left(1+\tau_{\text{b}}+t_0^{-1}\right)^{-\epsilon}\ . \tag{5.46}$$

Let us assume that $\frac{\beta}{R}\to\infty$ in which case from $\eta_{\text{eff}}=\eta_0\left(\frac{\beta}{R}\right)^{-\delta}$ we have that $\eta_{\text{eff}}\to 0$. Consequently $\chi\to 1$, $\omega\to 0$ and (5.43) becomes

$$\frac{1}{\eta_0}\left(\frac{\beta}{R}\right)^{-\left(\frac{1}{\epsilon}-\delta\right)}\left(\frac{\gamma_{\text{d}}}{R}\right)^{\left(\frac{1}{\epsilon}-1\right)}\geq\tau-\frac{(1+\tau)^{1-\epsilon}-1}{1-\epsilon}\ . \tag{5.47}$$

Provided $\epsilon\delta<1$ the l.h.s. of the above inequality vanishes in the limit $\frac{\beta}{R}\to\infty$. Then, since $\tau_{\min}$ vanishes as well, the r.h.s. of the inequality becomes a strictly increasing function of τ and (5.47) obviously holds as an equality only for $\tau=0$. Therefore,

$$\tau_{\text{b}}\to\tau_{\min}\to 0\quad\text{ as }\quad\frac{\beta}{\text{R}}\to\infty\ . \tag{5.48}$$

Combining (5.46) with (5.48) and noticing that $t_0^{-1}\to 0$ as $\frac{\beta}{R}\to\infty$ we conclude that $f\to 1$ or

$$\gamma'_{\text{d}}\to\gamma_{\text{d}}\quad\text{ as }\quad\frac{\beta}{\text{R}}\to\infty\ .$$

□

Remark 5.3. In the case $\epsilon=\frac{1}{2}$ by solving the quadratic equation derived from (5.43) we obtain explicitly an upper bound t_{b} on the number of updates and a lower bound f_{b} on

the fraction f of the margin that the algorithm achieves. They are the ones of (5.45) and (5.46), respectively with

$$\tau_{\rm b} = \left\{ \frac{1+\eta_{\rm eff}}{\chi} + \sqrt{\left(\frac{1+\eta_{\rm eff}}{\chi} - 1\right)^2 + \eta_{\rm eff}^{-1} \left(\frac{\beta}{R}\right)^{-2} \frac{\gamma_{\rm d}\,(1+\omega)}{\chi R}} \right\}^2 - 1 \ . \tag{5.49}$$

As $\frac{\beta}{R} \to \infty$, $\eta_{\rm eff} = \eta_0 \left(\frac{\beta}{R}\right)^{-\delta} \to 0$, $\chi \to 1$ and $\omega \to 0$. Then, $\tau_{\rm b} \to 0$ given that $\eta_{\rm eff}^{-1} \left(\frac{\beta}{R}\right)^{-2} = \eta_0^{-1} \left(\frac{\beta}{R}\right)^{\delta-2} \to 0$ if $0 < \delta < 2$. This demonstrates explicitly the statement of Theorem 5.2. Explicit bounds $t_{\rm b}$ and $f_{\rm b}$ are also obtainable for $\epsilon = 2$.

We now turn to special cases which are not covered by Theorem 5.2.

$\epsilon\delta = 1$:

If $\epsilon\delta = 1$ the l.h.s. of (5.47) becomes $\frac{1}{\eta_0} \left(\frac{\gamma_{\rm d}}{R}\right)^{\left(\frac{1}{\epsilon}-1\right)}$ which does not vanish in the limit $\frac{\beta}{R} \to \infty$. Therefore, $\tau_{\rm b}$ tends to a non-zero value depending on η_0. If, however, $\eta_0 \gg \left(\frac{\gamma_{\rm d}}{R}\right)^{\left(\frac{1}{\epsilon}-1\right)}$ the bound $\tau_{\rm b}$ can become very small leading to a guaranteed fraction of the margin achieved very close to 1.

$\epsilon = \delta = 1$: In this case as $\epsilon \to 1$ (5.47) becomes

$$\frac{1}{\eta_0} \geq \tau - \ln(1+\tau) \ .$$

For $\eta_0 = 1$ we obtain the bound $\tau_{\rm b} \simeq 2.15$ leading to a fraction of the maximum margin $f \geq (1+\tau_{\rm b})^{-1} \simeq 0.32$. By choosing larger values of η_0 we can make the value of the guaranteed fraction approach unity. In this particular case, however, it is possible to obtain better bounds on the number of updates leading to larger estimates for the guaranteed fraction of the margin by different proof techniques. Following the technique of [21], provided the inequalities $\eta_0 \left(1 + \eta_0^2 R^2/\beta^2\right)^{-1} \leq 1$ and $\eta_0 < \beta\gamma_{\rm d}/\sqrt{6}R^2$ are satisfied, we can obtain for $\epsilon = 1$ the upper bound

$$t \leq \frac{2}{\eta_0}\frac{\beta}{\gamma_{\rm d}} \left(1 + \frac{\eta_0\gamma_{\rm d}}{\beta}\right) \left(1 + \frac{\eta_0^2 R^2}{\beta^2}\right) + \frac{8}{3} \left(\frac{R}{\gamma_{\rm d}}\right)^2 \left(1 + \frac{\eta_0\gamma_{\rm d}}{\beta}\right)^2 \left(1 + \frac{\eta_0^2 R^2}{\beta^2}\right)^2 + 1 \tag{5.50}$$

on t and the lower bound

$$f \geq \frac{\eta_0}{2} \left\{ \left(1 + \frac{\eta_0 R}{\beta}\right) \left(1 + \frac{\eta_0^2 R^2}{\beta^2}\right) + \frac{4}{3}\frac{\eta_0 R^2}{\beta\gamma_{\rm d}} \left(1 + \frac{\eta_0 R}{\beta}\right)^2 \left(1 + \frac{\eta_0^2 R^2}{\beta^2}\right)^2 + \frac{\eta_0 R}{\beta} \right\}^{-1} \tag{5.51}$$

on the fraction f of the margin achieved. In the limit $\frac{\beta}{R} \to \infty$ we see that $f \geq \frac{\eta_0}{2}$ which saturating the constraint on η_0 could become $f \geq \frac{1}{2}$. By imposing the more relaxed

constraint $\eta_0 \left(1 + \eta_0^2 R^2/\beta^2\right)^{-1} \leq 2$ we can show that in the limit $\frac{\beta}{R} \to \infty$

$$f \geq \frac{2\eta_0}{3} \left(1 + \sqrt{1 + \frac{8}{3}\frac{\gamma_{\rm d}^2}{R^2}}\right)^{-1} \quad . \tag{5.52}$$

In this limit the constraint on η_0 allows η_0 values as large as 2. This fact combined with the observation that the ratio $\gamma_{\rm d}/R$ can be made very small by placing the patterns at a larger distance from the origin in the augmented space leads to a guaranteed fraction $\frac{2}{3}$ of the margin for the largest allowed value of η_0. Thus, our earlier conclusion that for $\epsilon = \delta = 1$ the guaranteed fraction of the margin achieved as $\frac{\beta}{R} \to \infty$ increases with η_0 is confirmed by this alternative technique. A proof of the above statements is provided in Appendix A.

5.8 The Mistake-Controlled Rule Algorithm MICRA$^{\epsilon,\zeta}$

From our discussion in Section 5.3 it becomes obvious that a Perceptron-like algorithm with the additive update (5.6) is uniquely determined by the functions $C(t)$, $\eta_{{\rm eff}\,t}$ and f_t. In particular, it does not depend on $\|\boldsymbol{a}_t\|$ as long as the above functions are $\|\boldsymbol{a}_t\|$-independent. If this is the case the update (5.6) of $\boldsymbol{a}_t$ can be replaced by the update (5.13) of $\boldsymbol{u}_t$. Our purpose here is to examine the sufficiently large subclass of such algorithms with $f_t = 1$ and $C(t)$, $\eta_{{\rm eff}\,t}$ inversely proportional to powers of t which counts the number of mistakes and determine sufficient conditions under which algorithms in the above subclass converge asymptotically to the optimal solution hyperplane. Such an investigation can be regarded as a generalisation of our earlier analysis of CRAMMA.

We consider algorithms having an update rule given by (5.13) with $f_t = 1$, an effective learning rate

$$\eta_{{\rm eff}\,t} = \frac{\eta}{t^\zeta} \tag{5.53}$$

and a misclassification condition

$$\boldsymbol{u}_t \cdot \boldsymbol{y}_k \leq \frac{\beta}{t^\epsilon} \quad . \tag{5.54}$$

Here η, ζ, β and ϵ are positive constants. The case $\zeta = 0$, corresponding to a constant effective learning rate, was treated in Section 5.7. We assume that the initial value $\boldsymbol{u}_1$ of $\boldsymbol{u}_t$ is the unit vector in the direction of the first training pattern. Then, (5.14) holds since on account of (5.13) $\boldsymbol{u}_t$ is a linear combination with positive coefficients of the training patterns $\boldsymbol{y}_k$ all of which have positive inner products with the optimal direction $\boldsymbol{u}$ because of (5.1). Obviously, the algorithm is $\|\boldsymbol{a}_t\|$-independent. The above (family of) algorithm(s) parametrised in terms of the exponents ϵ and ζ will be called the Mistake-Controlled Rule Algorithm MICRA$^{\epsilon,\zeta}$ [58] and is summarised in Fig. 5.4.

Require: A linearly separable augmented training set with reflection assumed $S = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l)$
Define:
For $k = 1, \ldots, l$
$R = \max_k \|\boldsymbol{y}_k\|\,, \;\; \bar{\boldsymbol{y}}_k = \boldsymbol{y}_k / R$
Fix: $\eta, \;\; \beta_1 \, (= \beta / R)$
Initialisation:
$t = 1, \;\; \boldsymbol{u}_1 = \bar{\boldsymbol{y}}_1 / \|\bar{\boldsymbol{y}}_1\|, \;\; \eta_{\mathrm{eff}\,1} = \eta$

repeat until no update made within the **for** loop
  for $k = 1$ to l **do**
    if $\boldsymbol{u}_t \cdot \bar{\boldsymbol{y}}_k \leq \beta_t$ **then**
      $\boldsymbol{u}_{t+1} = \frac{\boldsymbol{u}_t + \eta_{\mathrm{eff}\,t} \bar{\boldsymbol{y}}_k}{\|\boldsymbol{u}_t + \eta_{\mathrm{eff}\,t} \bar{\boldsymbol{y}}_k\|}$
      $t \leftarrow t + 1$
      $\beta_t = \beta_1 / t^{\epsilon}, \;\; \eta_{\mathrm{eff}\,t} = \eta / t^{\zeta}$

FIGURE 5.4: The Mistake-Controlled Rule Algorithm MICRA$^{\epsilon,\zeta}$.

We begin again our analysis with a short discussion of stepwise convergence. It is not difficult to see that (5.1), (5.53) and (5.54) lead to a lower bound on $\mathcal{A}$ of (5.16)

$$\mathcal{A} \geq \gamma_{\mathrm{d}} - \frac{\beta}{t^{\epsilon}} - \frac{\eta}{t^{\zeta}} \frac{R}{2} \;\; . \tag{5.55}$$

By requiring that the r.h.s. of (5.55) be positive one determines the critical time t_{c} for the onset of stepwise convergence. In addition convergence occurs always in a finite number of steps only if

$$0 < \zeta \leq 1$$

since in this case $\eta_{\mathrm{eff}\,t} = \eta / t^{\zeta}$ does not fall with t faster that $1/t$.

Theorem 5.4. *The* MICRA$^{\epsilon,\zeta}$ *algorithm of Fig. 5.4 converges in a finite number of steps provided* $\zeta \leq 1$. *Moreover, if* η *is given a dependence on* β *through the relation* $\eta = \eta_0 \left(\frac{\beta}{R}\right)^{-\delta}$ *the directional margin* γ'_{d} *that the algorithm achieves tends in the limit* $\frac{\beta}{R} \to \infty$ *to the maximum directional margin* γ_{d} *provided* $0 < \epsilon\delta + \zeta < 1$.

Proof. Taking the inner product of (5.13) with the optimal direction $\boldsymbol{u}$, expanding $\|\boldsymbol{u}_t + \eta_{\mathrm{eff}\,t} \boldsymbol{y}_k / R\|^{-1}$ and using the inequality $(1+x)^{-\frac{1}{2}} \geq 1 - x/2$ we have

$$\begin{aligned} \boldsymbol{u}_{t+1} \cdot \boldsymbol{u} &= \left(\boldsymbol{u}_t \cdot \boldsymbol{u} + \eta_{\mathrm{eff}\,t} \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}}{R}\right) \left(1 + 2\eta_{\mathrm{eff}\,t} \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}_t}{R} + \eta_{\mathrm{eff}\,t}^2 \frac{\|\boldsymbol{y}_k\|^2}{R^2}\right)^{-\frac{1}{2}} \\ &\geq \left(\boldsymbol{u}_t \cdot \boldsymbol{u} + \eta_{\mathrm{eff}\,t} \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}}{R}\right) \left(1 - \eta_{\mathrm{eff}\,t} \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}_t}{R} - \eta_{\mathrm{eff}\,t}^2 \frac{\|\boldsymbol{y}_k\|^2}{2R^2}\right) \;\; . \end{aligned}$$

Thus, we obtain for $\Delta \equiv \boldsymbol{u}_{t+1} \cdot \boldsymbol{u} - \boldsymbol{u}_t \cdot \boldsymbol{u}$

$$\begin{aligned} \frac{R}{\eta_{\mathrm{eff}\,t}} \Delta \geq \boldsymbol{y}_k \cdot \boldsymbol{u} - (\boldsymbol{u}_t \cdot \boldsymbol{u})(\boldsymbol{y}_k \cdot \boldsymbol{u}_t) - \frac{\eta_{\mathrm{eff}\,t}}{2R} \Big(\|\boldsymbol{y}_k\|^2 \, \boldsymbol{u}_t \cdot \boldsymbol{u} + 2(\boldsymbol{y}_k \cdot \boldsymbol{u})(\boldsymbol{y}_k \cdot \boldsymbol{u}_t) \Big) \\ - \frac{\eta_{\mathrm{eff}\,t}^2}{2R^2} \|\boldsymbol{y}_k\|^2 \, \boldsymbol{y}_k \cdot \boldsymbol{u} \;\; . \end{aligned}$$

By employing (5.1), (5.54) and (5.14) we get a lower bound on Δ

$$\Delta \geq \eta_{\mathrm{eff}\,t}\left(\frac{\gamma_{\mathrm{d}}}{R}-\frac{\eta_{\mathrm{eff}\,t}}{2}-\frac{\eta_{\mathrm{eff}\,t}^2}{2}\right)-\eta_{\mathrm{eff}\,t}\left(1+\eta_{\mathrm{eff}\,t}\right)\frac{\beta}{R}t^{-\epsilon}\ . \tag{5.56}$$

From the misclassification condition it becomes obvious that convergence of the algorithm is impossible as long as $\beta/t^{\epsilon}>\gamma_{\mathrm{d}}$. Therefore we may assume that

$$t>t_0\equiv\left(\frac{\beta}{\gamma_{\mathrm{d}}}\right)^{\frac{1}{\epsilon}}\ .$$

A repeated application of (5.56) $(t-[t_0])$ times, where $[t_0]$ denotes the integer part of t_0, yields

$$\boldsymbol{u}_{t+1}\cdot\boldsymbol{u}-\boldsymbol{u}_{[t_0]+1}\cdot\boldsymbol{u}\geq\eta\frac{\gamma_{\mathrm{d}}}{R}\sum_{m=[t_0]+1}^{t}m^{-\zeta}-\frac{\eta^2}{2}\sum_{m=[t_0]+1}^{t}m^{-2\zeta}-\frac{\eta^3}{2}\sum_{m=[t_0]+1}^{t}m^{-3\zeta}$$

$$-\eta\frac{\beta}{R}\sum_{m=[t_0]+1}^{t}m^{-(\zeta+\epsilon)}-\eta^2\frac{\beta}{R}\sum_{m=[t_0]+1}^{t}m^{-(2\zeta+\epsilon)}\ .$$

By employing the inequalities

$$\sum_{m=[t_0]+1}^{t}m^{-\theta}\geq\int_{t_0+1}^{t}m^{-\theta}dm\geq\frac{t^{1-\theta}-t_0^{1-\theta}}{1-\theta}-t_0^{-\theta}$$

and

$$\sum_{m=[t_0]+1}^{t}m^{-\theta}\leq\int_{t_0}^{t}m^{-\theta}dm+t_0^{-\theta}=\frac{t^{1-\theta}-t_0^{1-\theta}}{1-\theta}+t_0^{-\theta}$$

for $\theta>0$ and taking into account (5.14) we finally obtain

$$1\geq\eta\frac{\gamma_{\mathrm{d}}}{R}\left(\frac{t^{1-\zeta}-t_0^{1-\zeta}}{1-\zeta}\right)-\frac{\eta^2}{2}\left(\frac{t^{1-2\zeta}-t_0^{1-2\zeta}}{1-2\zeta}\right)-\frac{\eta^3}{2}\left(\frac{t^{1-3\zeta}-t_0^{1-3\zeta}}{1-3\zeta}\right)$$

$$-\eta\frac{\beta}{R}\left(\frac{t^{1-(\zeta+\epsilon)}-t_0^{1-(\zeta+\epsilon)}}{1-(\zeta+\epsilon)}\right)-\eta^2\frac{\beta}{R}\left(\frac{t^{1-(2\zeta+\epsilon)}-t_0^{1-(2\zeta+\epsilon)}}{1-(2\zeta+\epsilon)}\right)-\omega\ . \tag{5.57}$$

Here

$$\omega\equiv\frac{\gamma_{\mathrm{d}}}{R}\eta t_0^{-\zeta}\left(2+\eta t_0^{-\zeta}\right)+\frac{1}{2}\eta^2t_0^{-2\zeta}\left(1+\eta t_0^{-\zeta}\right)>0\ .$$

Let us define the new variable $\tau\geq0$ through the relation

$$t=t_0\left(1+\tau\right)=\left(\frac{\beta}{\gamma_{\mathrm{d}}}\right)^{\frac{1}{\epsilon}}\left(1+\tau\right)\ . \tag{5.58}$$

In terms of τ (5.57) becomes

$$\left(\eta t_0^{1-\zeta}\right)^{-1}\left(\frac{\gamma_{\rm d}}{R}\right)^{-1}(1+\omega)\geq g(\tau)\equiv\frac{(1+\tau)^{1-\zeta}-1}{1-\zeta}-\frac{(1+\tau)^{1-(\zeta+\epsilon)}-1}{1-(\zeta+\epsilon)}$$

$$-\frac{R}{2\gamma_{\rm d}}\eta t_0^{-\zeta}\frac{(1+\tau)^{1-2\zeta}-1}{1-2\zeta}-\frac{R}{2\gamma_{\rm d}}\eta^2 t_0^{-2\zeta}\frac{(1+\tau)^{1-3\zeta}-1}{1-3\zeta}-\eta t_0^{-\zeta}\frac{(1+\tau)^{1-(2\zeta+\epsilon)}-1}{1-(2\zeta+\epsilon)}\ . \tag{5.59}$$

Notice that for $\zeta=1$ the first term of $g(\tau)$ becomes $\ln(1+\tau)$. Since $0<\zeta\leq 1$, $g(\tau)$ (with $\tau\geq 0$) is unbounded from above. Moreover, its derivative $g'(\tau)$ satisfies the relation

$$(1+\tau)^{\zeta}g'(\tau)=1-(1+\tau)^{-\epsilon}-\frac{R}{2\gamma_{\rm d}}\eta t_0^{-\zeta}(1+\tau)^{-\zeta}-\frac{R}{2\gamma_{\rm d}}\eta^2 t_0^{-2\zeta}(1+\tau)^{-2\zeta}$$
$$-\eta t_0^{-\zeta}(1+\tau)^{-(\zeta+\epsilon)}\ .$$

The r.h.s. of the above equation is a monotonically increasing function of τ which is negative at $\tau=0$ and tends to 1 as $\tau\to\infty$. Therefore $g'(\tau)$ has a single root at $\tau=\tau_{\rm min}$ which corresponds to a minimum of $g(\tau)$ with $g(\tau_{\rm min})<0$. Moreover, the l.h.s. of (5.59) is positive. Thus, given that $g(0)=0$, there is a single value $\tau_{\rm b}$ of τ where (5.59) holds as an equality which provides an upper bound on τ

$$\tau\leq\tau_{\rm b} \tag{5.60}$$

satisfying $\tau_{\rm b}>\tau_{\rm min}>0$. Combining (5.58) and (5.60) we obtain the bound on the number of updates

$$t\leq t_{\rm b}\equiv\left(\frac{\beta}{\gamma_{\rm d}}\right)^{\frac{1}{\epsilon}}(1+\tau_{\rm b}) \tag{5.61}$$

proving that the algorithm converges in a finite number of steps. From (5.61) and taking into account the misclassification condition (5.54) we obtain a lower bound $\beta/(t_{\rm b}+1)^{\epsilon}$ on the margin $\gamma'_{\rm d}$ achieved. Thus, the fraction f of $\gamma_{\rm d}$ that the algorithm achieves satisfies

$$1\geq f\equiv\frac{\gamma'_{\rm d}}{\gamma_{\rm d}}\geq f_{\rm b}\equiv\frac{\beta/\gamma_{\rm d}}{(t_{\rm b}+1)^{\epsilon}}=\left(1+\tau_{\rm b}+t_0^{-1}\right)^{-\epsilon}\ . \tag{5.62}$$

Let us assume that $\frac{\beta}{R}\to\infty$ in which case from $\eta=\eta_0\left(\frac{\beta}{R}\right)^{-\delta}$ and given that $0<\epsilon\delta+\zeta<1$ we have $\eta t_0^{1-\zeta}\sim\left(\frac{\beta}{R}\right)^{\frac{1-\zeta-\epsilon\delta}{\epsilon}}\to\infty$ whereas $\eta t_0^{-\zeta}\sim\left(\frac{\beta}{R}\right)^{-\frac{\zeta+\epsilon\delta}{\epsilon}}\to 0$. Consequently the l.h.s. of (5.59) vanishes in the limit $\frac{\beta}{R}\to\infty$ whereas its r.h.s. (i.e. $g(\tau)$) becomes a strictly increasing function for $\tau>0$ (i.e. $\tau_{\rm min}\to 0$) since $(1+\tau)^{\zeta}g'(\tau)=1-(1+\tau)^{-\epsilon}>0$. Obviously, (5.59) holds as an equality only for $\tau=0$. Therefore,

$$\tau_{\rm b}\to\tau_{\rm min}\to 0\quad\text{ as }\quad\frac{\beta}{\rm R}\to\infty\ . \tag{5.63}$$

Combining (5.62) with (5.63) and noticing that $t_0^{-1} \to 0$ as $\frac{\beta}{R} \to \infty$ we conclude that $f \to 1$ or

$$\gamma_{\rm d}' \to \gamma_{\rm d} \quad \text{as} \quad \frac{\beta}{\rm R} \to \infty \ .$$

□

Remark 5.5. In the case that $\zeta + 2\epsilon = 1$ with $\frac{1}{2} < \zeta < 1$ it is possible to obtain explicitly an upper bound $t_{\rm b}$ on the number of updates and a lower bound $f_{\rm b}$ on the fraction f of the margin that the algorithm achieves. First we observe that since $1 - 2\zeta$, $1 - 3\zeta$ and $1 - (2\zeta + \epsilon)$ are negative it is allowed to set the terms $(1+\tau)^{1-2\zeta}$, $(1+\tau)^{1-3\zeta}$ and $(1+\tau)^{1-(2\zeta+\epsilon)}$ to zero in the r.h.s. of (5.59). Then, the resulting less restrictive inequality with ζ expressed in terms of ϵ becomes

$$A^2 \geq \left((1+\tau)^{\epsilon} - 1\right)^2 \ , \tag{5.64}$$

where

$$A^2 = \frac{2\epsilon}{\eta}\left(\frac{\beta}{R}\right)^{-2}\frac{\gamma_{\rm d}}{R}(1+\omega) + \frac{\epsilon\eta}{1-4\epsilon}\left(\frac{\beta}{R}\right)^{2-\frac{1}{\epsilon}}\left(\frac{\gamma_{\rm d}}{R}\right)^{\frac{1}{\epsilon}-3}$$
$$+\frac{\epsilon\eta^2}{2-6\epsilon}\left(\frac{\beta}{R}\right)^{4-\frac{2}{\epsilon}}\left(\frac{\gamma_{\rm d}}{R}\right)^{\frac{2}{\epsilon}-5} + \frac{2\epsilon\eta}{1-3\epsilon}\left(\frac{\beta}{R}\right)^{2-\frac{1}{\epsilon}}\left(\frac{\gamma_{\rm d}}{R}\right)^{\frac{1}{\epsilon}-2} \ .$$

Notice that $\epsilon < \frac{1}{4}$ if $\frac{1}{2} < \zeta < 1$. By solving the equation derived from (5.64) we obtain explicitly the bounds $t_{\rm b}$ and $f_{\rm b}$. They are the ones of (5.61) and (5.62), respectively with

$$\tau_{\rm b} = (1 + |A|)^{\frac{1}{\epsilon}} - 1 \ .$$

In the present case $0 < \epsilon\delta + \zeta < 1$ is equivalent to $2 - \frac{1}{\epsilon} < \delta < 2$. Then, with $\eta = \eta_0\left(\frac{\beta}{R}\right)^{-\delta}$ as $\frac{\beta}{R} \to \infty$ we get $|A| \to 0$ leading to $\tau_{\rm b} \to 0$. This demonstrates explicitly the statement of Theorem 5.4. It is worth emphasising, however, that $|A|$ may be small even if $\frac{\beta}{R}$ is not large if $\frac{\gamma_{\rm d}}{R}$ and ϵ are sufficiently small.

Example: If $\epsilon = \zeta = \frac{1}{2}$ and moreover $\delta = 0$, i.e. η does not depend on β, $\epsilon\delta + \zeta = \frac{1}{2}$ and the condition of Theorem 5.4 is satisfied. Therefore such an algorithm attains asymptotically as $\frac{\beta}{R} \to \infty$ the maximum directional margin. The above algorithm is a version of ALMA_2 in which the weight vector instead of being confined within a ball centred at the origin is normalised to a constant length which remains fixed during the asymptotic procedure. Thus, ALMA_2 can be thought of as belonging to the MICRA family. Then, the analysis of [21] confirms our conclusion regarding asymptotic convergence to the optimal solution hyperplane in this special case. In the case, instead, that $\epsilon = \zeta = \frac{1}{2}$ but $\delta = 1$, i.e. $\eta = \eta_0\left(\frac{\beta}{R}\right)^{-1}$, $\epsilon\delta + \zeta = 1$ and the condition of Theorem 5.4 is violated. This case would correspond to a version of ALMA_2 with the parameter b entering the misclassification condition set to $b = \beta^2$ and the weight vector normalised to the constant length β which, however, does not remain fixed during the asymptotic procedure $\frac{\beta}{R} \to \infty$. Since the condition of Theorem 5.4 is violated we are unable to

prove asymptotic convergence of such an algorithm to the maximal margin solution. The same conclusion is reached if the proof technique of [21] is employed which gives a lower bound

$$f_{\mathrm{b}} = \left(1 + \frac{1}{\eta_0} + \frac{3}{2}\eta_0 \frac{R^2}{\beta^2}\right)^{-1}$$

on the fraction of the maximum directional margin achieved by the algorithm. As $\frac{\beta}{R} \to \infty$ we get $f_{\mathrm{b}} \to \frac{\eta_0}{(1+\eta_0)} < 1$. We see that a "slight" modification of the asymptotic procedure is able to affect the ability of a Perceptron-like algorithm to attain the solution with maximum margin. We believe that the inability in some cases of the Perceptron algorithm with margin, in contrast to ALMA$_2$, to approach the maximal margin solution is due to such "slight" differences between the two algorithms regarding the asymptotic procedure.

Efficient Implementation: A completely equivalent formulation of MICRA is obtained if the update rule (5.6) with $f_t = N_{t+1} = 1$ and $\eta_t = \|\boldsymbol{a}_t\| \, \eta_{\mathrm{eff}\,t}/R$ is employed and the misclassification condition (5.54) is reexpressed as $\boldsymbol{a}_t \cdot \boldsymbol{y}_k \leq \|\boldsymbol{a}_t\| \, \beta/t^{\epsilon}$. Such a formulation apart from bearing a close resemblance to the Perceptron algorithm has the additional advantage of being computationally more efficient. A pseudocode implementing this formulation is given in Fig. 5.5.

Require: A linearly separable augmented training set with reflection assumed $S = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k, \ldots, \boldsymbol{y}_l)$

Fix: η, β

Define: $R = \max_k \|\boldsymbol{y}_k\|$, $q_k = \|\boldsymbol{y}_k\|^2$, $\bar{\eta} = \eta/R$

Initialisation: $t = 1$, $\boldsymbol{a}_1 = \boldsymbol{y}_1$, $\|\boldsymbol{a}_1\| = \|\boldsymbol{y}_1\|$, $\eta_1 = \|\boldsymbol{a}_1\| \, \bar{\eta}$, $\beta_1 = \|\boldsymbol{a}_1\| \, \beta$

repeat until no update made within the **for** loop
  for $k = 1$ to l **do**
    $p_{tk} = \boldsymbol{a}_t \cdot \boldsymbol{y}_k$
    if $p_{tk} \leq \beta_t$ **then**
      $\boldsymbol{a}_{t+1} = \boldsymbol{a}_t + \eta_t \boldsymbol{y}_k$
      $\|\boldsymbol{a}_{t+1}\| = \sqrt{\|\boldsymbol{a}_t\|^2 + \eta_t \left(2p_{tk} + \eta_t q_k\right)}$
      $t \leftarrow t + 1$
      $\eta_t = \|\boldsymbol{a}_t\| \, \bar{\eta}/t^{\zeta}$, $\beta_t = \|\boldsymbol{a}_t\| \, \beta/t^{\epsilon}$

FIGURE 5.5: An efficient implementation of MICRA$^{\epsilon,\zeta}$.

5.9 Algorithms with Fixed Directional Margin Condition

In this section we examine algorithms in which the misclassification condition assumes the form of (5.11) which amounts to requiring a minimum directional margin that is not lowered with the number of updates [56]. The condition (5.11) involving only the direction of the weight vector motivates new positive and bounded functions f_t like the functions $f_t = 1 - \beta \frac{\boldsymbol{u}_t \cdot \boldsymbol{y}_k}{\|\boldsymbol{y}_k\|^2}$ and $f_t = \frac{\beta_u - \boldsymbol{u}_t \cdot \boldsymbol{y}_k}{\|\boldsymbol{y}_k\|}$ with $\beta_u > \beta$ in addition to the

commonly used $f_t = 1$. Convergence of such algorithms requires that $\beta < \gamma_{\rm d}$. Since $\gamma_{\rm d}$ is not known these algorithms will be useful only as components of a more complex algorithmic implementation exploring efficiently the range of allowed values of β.

5.9.1 Generic Perceptron-Like Algorithms with Fixed Margin Condition

Here we consider algorithms with the general update rule (5.6), constant learning rate $\eta_t = \eta$ and $N_{t+1} = 1$. A pseudocode describing their implementation is given in Fig. 5.6. We begin with a discussion of stepwise convergence. A repeated application of (5.18) assuming that $\boldsymbol{a}_t$ is initially set to zero leads again to (5.19). As a consequence for $t > 0$ (5.14) is once more recovered. Therefore, positivity of $\mathcal{D}$ of (5.15) is equivalent to stepwise convergence. Placing a lower bound on the $\eta_{{\rm eff}\,t}$-independent part of $\mathcal{A}$ of (5.16), using (5.1) and (5.11), we obtain

$$\boldsymbol{y}_k \cdot \boldsymbol{u} - (\boldsymbol{u}_t \cdot \boldsymbol{u})(\boldsymbol{y}_k \cdot \boldsymbol{u}_t) \geq \gamma_{\rm d} - \beta \ , \tag{5.65}$$

which is definitely positive. Furthermore, because of (5.19) the terms of $\mathcal{A}$ linear in $\eta_{{\rm eff}\,t}$, which are not necessarily positive, become less important with time leading to positivity of $\mathcal{A}$ and consequently of $\mathcal{D}$ for t larger than a critical time $t_{\rm c}$. Thus, we can place a lower bound on $\mathcal{A}$

$$\mathcal{A} \geq \gamma_{\rm d} - \beta - \frac{1}{2}\frac{f_{\max}}{f_{\min}}\frac{1}{\gamma_{\rm d} t}(R^2 - \gamma_{\rm d}^2) \ . \tag{5.66}$$

From (5.66) the estimated time sufficient for the onset of stepwise convergence is

$$t_{\rm c} \equiv \frac{1}{2}\frac{f_{\max}}{f_{\min}}\frac{R^2}{\gamma_{\rm d}^2}\frac{\left(1 - \frac{\gamma_{\rm d}^2}{R^2}\right)}{\left(1 - \frac{\beta}{\gamma_{\rm d}}\right)} \ . \tag{5.67}$$

It is obvious that for $t > t_{\rm c}$ $\mathcal{A}$ is bounded from below by a positive constant. Moreover, since we initially set the weight vector to zero, $\boldsymbol{a}_t$ is entirely generated by the first t updates and its norm satisfies the obvious bound

$$\|\boldsymbol{a}_t\| \leq \eta f_{\max} R t \ . \tag{5.68}$$

Require: A linearly separable augmented training set with reflection assumed $S = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l)$
Input: β
Initialisation:
$t = 0$, $\boldsymbol{a}_0 = \boldsymbol{0}$

repeat until no update made within the **for** loop
  for $k = 1$ to l **do**
    if $\boldsymbol{u}_t \cdot \boldsymbol{y}_k \leq \beta$ **then**
      $\boldsymbol{a}_{t+1} = \boldsymbol{a}_t + \eta f_t \boldsymbol{y}_k$
      $t \leftarrow t + 1$

FIGURE 5.6: Generic Perceptron-like algorithm with fixed margin condition.

Thus, $\eta_{\mathrm{eff}\,t}$ does not fall with t faster than $1/t$. Moreover, for $t > 1$ $\eta_{\mathrm{eff}\,t}$ is bounded from above because of (5.19). Therefore all the conditions are satisfied in order for stepwise convergence to lead to convergence in a finite number of steps.

We now proceed to a derivation of a time bound. Our procedure will be to provide a tighter upper bound on $\|\boldsymbol{a}_t\|$ than the one of (5.68) which together with the lower one of (5.19) will finally be combined in a Novikoff-like squeezing relationship. For the derivation of an upper bound we first use (5.6) to obtain

$$\|\boldsymbol{a}_{t+1}\|^2 = \|\boldsymbol{a}_t\|^2 \left(1 + 2\frac{\eta f_t}{\|\boldsymbol{a}_t\|}\boldsymbol{y}_k \cdot \boldsymbol{u}_t + \left(\frac{\eta f_t}{\|\boldsymbol{a}_t\|}\right)^2 \|\boldsymbol{y}_k\|^2\right) \quad .$$

Taking the square root and using the inequality $\sqrt{1+x} \leq 1 + x/2$ we have

$$\|\boldsymbol{a}_{t+1}\| \leq \|\boldsymbol{a}_t\| \left(1 + \frac{\eta f_t}{\|\boldsymbol{a}_t\|}\boldsymbol{y}_k \cdot \boldsymbol{u}_t + \frac{1}{2}\left(\frac{\eta f_t}{\|\boldsymbol{a}_t\|}\right)^2 \|\boldsymbol{y}_k\|^2\right) \quad .$$

We now observe that the difference of $\|\boldsymbol{a}_t\|$ at successive time instants satisfies the inequality

$$\|\boldsymbol{a}_{t+1}\| - \|\boldsymbol{a}_t\| \leq \eta f_{\max}\beta + \frac{\eta}{2}\frac{f_{\max}^2}{f_{\min}}\frac{R^2}{\gamma_{\mathrm{d}}}\frac{1}{t} \quad .$$

Here we have made use of (5.11) and (5.19). A repeated application of the above inequality $(t - N)$ times gives

$$\|\boldsymbol{a}_t\| - \|\boldsymbol{a}_N\| \leq \eta f_{\max}\beta(t-N) + \frac{\eta}{2}\frac{f_{\max}^2}{f_{\min}}\frac{R^2}{\gamma_{\mathrm{d}}}\left(\frac{1}{N} + \frac{1}{N+1} + \cdots + \frac{1}{t-1}\right) \quad . \tag{5.69}$$

Replacing $\|\boldsymbol{a}_N\|$ by its obvious upper bound from (5.68) and employing the inequality

$$\sum_{k=n_1}^{n_2} \frac{1}{k} \leq \int_{n_1}^{n_2} \frac{dk}{k} + \frac{1}{n_1} = \ln\frac{n_2}{n_1} + \frac{1}{n_1}$$

we get the upper bound

$$\|\boldsymbol{a}_t\| \leq \eta f_{\max}\left\{RN + \beta(t-N) + \frac{1}{2}\frac{f_{\max}}{f_{\min}}\frac{R^2}{\gamma_{\mathrm{d}}}\left(\ln\frac{t-1}{N} + \frac{1}{N}\right)\right\} \tag{5.70}$$

on $\|\boldsymbol{a}_t\|$. Squeezing $\|\boldsymbol{a}_t\|$ between its lower bound of (5.19) and its upper bound of (5.70) we obtain a relation

$$\frac{t-N}{C_N + \ln\sqrt{t-1}} \leq \left(\frac{f_{\max}}{f_{\min}}\frac{R}{\gamma_{\mathrm{d}}}\right)^2 \left(1 - \frac{f_{\max}}{f_{\min}}\frac{\beta}{\gamma_{\mathrm{d}}}\right)^{-1} \tag{5.71}$$

constraining the growth of t. Here

$$C_N = N\frac{f_{\min}}{f_{\max}}\frac{\gamma_{\mathrm{d}}}{R}\left(1 - \frac{f_{\min}}{f_{\max}}\frac{\gamma_{\mathrm{d}}}{R}\right) - \frac{1}{2}\left(\ln N - \frac{1}{N}\right) \quad .$$

Minimising the upper bound of (5.70) with respect to N we obtain the optimal value

$$N_{\rm opt} = \left[\frac{1}{2} \frac{f_{\max}}{f_{\min}} \frac{R}{\gamma_{\rm d}} \left(1 - \frac{\beta}{R} \right)^{-1} \right] + 1 \ ,$$

where $[x]$ denotes the integer part of x. We would like to point out that unless $f_{\min}\gamma_{\rm d} - f_{\max}\beta$ is positive (5.71) does not lead to an upper bound on t. However, this failure of obtaining an upper bound on the number of steps does not reflect lack of convergence which has already been proved independently. Of course, for the perceptron-like algorithm of this type where $f_t = 1$ we have an upper bound for all β less than $\gamma_{\rm d}$ which interestingly enough has a dependence on the difference $\gamma_{\rm d} - \beta$. The same difference appears in the expression for the critical time $t_{\rm c}$ of (5.67) irrespective of the function f_t employed. Another interesting property of all algorithms of this class, provided $\boldsymbol{a}_0 = 0$ and f_t depends on $\boldsymbol{a}_t$ only through $\boldsymbol{u}_t$, is their independence from the learning rate η, a property shared by the perceptron algorithm with zero margin. This can be justified by the fact that a rescaling of η results in a rescaling of $\boldsymbol{a}_t$ by the same factor which does not affect either the hyperplane normal to $\boldsymbol{a}_t$ or the classification condition. This independence from η is apparent in both (5.67) and (5.71).

5.9.2 Algorithms with Constant Effective Learning Rate and Fixed Margin Condition

Here we examine algorithms with the fixed directional margin condition in which the effective learning rate $\eta_{{\rm eff}\,t}$ remains constant. One possible realisation of such algorithms is obtained if we keep the length of the weight vector fixed assuming a constant learning rate but in the following we will not make such an assumption. Thus, in order to avoid using the length of the weight vector we will only employ (5.13) as an update rule assuming that f_t does not depend on $\|\boldsymbol{a}_t\|$. We demand that $\boldsymbol{u}_t \cdot \boldsymbol{u} > 0$ for all t which requires an appropriate choice of the initial condition. We choose the initial unit vector $\boldsymbol{u}_0$ in the direction of one (or a linear combination with positive coefficients) of the $\boldsymbol{y}_k$'s. For definiteness we choose the direction of the first training pattern. Then, due to the

Require: A linearly separable augmented training set with reflection assumed $S = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l)$

Define:

For $k = 1, \ldots, l$

$R = \max_k \|\boldsymbol{y}_k\|\,, \quad \bar{\boldsymbol{y}}_k = \boldsymbol{y}_k / R$

Input: $\bar{\beta}\,(= \beta/R)\,, \quad \eta_{\rm eff}$

Initialisation: $t = 0,\ \boldsymbol{u}_0 = \bar{\boldsymbol{y}}_1 / \|\bar{\boldsymbol{y}}_1\|$

repeat until no update made within the **for** loop

  for $k = 1$ to l **do**

    if $\boldsymbol{u}_t \cdot \bar{\boldsymbol{y}}_k \leq \bar{\beta}$ **then**

      $\boldsymbol{u}_{t+1} = \dfrac{\boldsymbol{u}_t + \eta_{\rm eff} f_t \bar{\boldsymbol{y}}_k}{\|\boldsymbol{u}_t + \eta_{\rm eff} f_t \bar{\boldsymbol{y}}_k\|}$

      $t \leftarrow t + 1$

FIGURE 5.7: Constant effective learning rate algorithm with fixed margin condition.

form of the update rule and the positivity of f_t the weight vector is a linear combination with positive coefficients of the training patterns. Therefore, since according to (5.1) $\boldsymbol{y}_k$ satisfies $\boldsymbol{y}_k \cdot \boldsymbol{u} > 0$ the same is true for $\boldsymbol{u}_t$. A pseudocode description of the algorithms under consideration appears in Fig. 5.7.

Positivity of $\boldsymbol{u}_t \cdot \boldsymbol{u}$ allows us to use positivity of $\mathcal{D}$ of (5.15) as a criterion for stepwise convergence. Taking a closer look at $\mathcal{A}$ of (5.16) reveals that the η_{eff}-independent term remains positive throughout the algorithm. For the term linear in η_{eff} which has no definite sign we conclude that an appropriate choice of η_{eff} can render it smaller than the η_{eff}-independent one, thereby leading to stepwise convergence from the first step of the algorithm. More specifically, by placing lower bounds using (5.1) and (5.11) we have for $\mathcal{A}$

$$\mathcal{A} \geq \gamma_{\text{d}} - \beta - \frac{\eta_{\text{eff}} f_{\max}}{2R}(R^2 - \gamma_{\text{d}}^2) \ . \tag{5.72}$$

Positivity of $\mathcal{A}$ and $\mathcal{D}$ is achieved for values of η_{eff} smaller than the critical value $\eta_{\text{eff c}}$

$$\eta_{\text{eff c}} \equiv \frac{2}{f_{\max}} \frac{(\gamma_{\text{d}} - \beta)R}{\left(R^2 - \gamma_{\text{d}}^2\right)} \ . \tag{5.73}$$

Taking into account (5.14) and (5.72) and given that η_{eff} is constant stepwise convergence from the first step implies convergence in a finite number of steps.

After having shown that the algorithm converges step by step our next move will be to place an upper bound on the number of updates.

Taking the inner product of (5.13) with the optimal direction $\boldsymbol{u}$, expanding $\left\| \boldsymbol{u}_t + \eta_{\text{eff}} f_t \frac{\boldsymbol{y}_k}{R} \right\|^{-1}$ and using the inequality $(1+x)^{-\frac{1}{2}} \geq 1 - x/2$ we have

$$\begin{aligned} \boldsymbol{u}_{t+1} \cdot \boldsymbol{u} &= \left(\boldsymbol{u}_t \cdot \boldsymbol{u} + \eta_{\text{eff}} f_t \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}}{R} \right) \left(1 + 2\eta_{\text{eff}} f_t \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}_t}{R} + \eta_{\text{eff}}^2 f_t^2 \frac{\|\boldsymbol{y}_k\|^2}{R^2} \right)^{-\frac{1}{2}} \\ &\geq \left(\boldsymbol{u}_t \cdot \boldsymbol{u} + \eta_{\text{eff}} f_t \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}}{R} \right) \left(1 - \eta_{\text{eff}} f_t \frac{\boldsymbol{y}_k \cdot \boldsymbol{u}_t}{R} - \eta_{\text{eff}}^2 f_t^2 \frac{\|\boldsymbol{y}_k\|^2}{2R^2} \right) \ . \end{aligned}$$

Thus, we obtain for $\Delta = \boldsymbol{u}_{t+1} \cdot \boldsymbol{u} - \boldsymbol{u}_t \cdot \boldsymbol{u}$

$$\begin{aligned} \frac{R}{\eta_{\text{eff}} f_t} \Delta \geq \boldsymbol{y}_k \cdot \boldsymbol{u} - (\boldsymbol{u}_t \cdot \boldsymbol{u})(\boldsymbol{y}_k \cdot \boldsymbol{u}_t) - \frac{\eta_{\text{eff}} f_t}{2R} \Big(\|\boldsymbol{y}_k\|^2 \boldsymbol{u}_t \cdot \boldsymbol{u} + 2(\boldsymbol{y}_k \cdot \boldsymbol{u})(\boldsymbol{y}_k \cdot \boldsymbol{u}_t) \Big) \\ - \frac{\eta_{\text{eff}}^2 f_t^2}{2R^2} \|\boldsymbol{y}_k\|^2 \boldsymbol{y}_k \cdot \boldsymbol{u} \ . \end{aligned}$$

We now observe that Δ can be bounded from below by a constant

$$\Delta \geq \eta_{\text{eff}} f_{\min} \left\{ \frac{\gamma_d - \beta}{R} - \frac{1}{2} \eta_{\text{eff}} f_{\max} \left(1 + \frac{2\beta}{R} \right) - \frac{1}{2} \eta_{\text{eff}}^2 f_{\max}^2 \right\} . \tag{5.74}$$

Here we made use of (5.1) and (5.11). A repeated application of (5.74) gives

$$\boldsymbol{u}_t \cdot \boldsymbol{u} - \boldsymbol{u}_0 \cdot \boldsymbol{u} \geq \frac{f_{\min}}{f_{\max}} \left\{ \frac{\gamma_{\mathrm{d}} - \beta}{R} \left(\eta_{\mathrm{eff}} f_{\max}\right) - \frac{1}{2} \left(1 + \frac{2\beta}{R}\right) \left(\eta_{\mathrm{eff}} f_{\max}\right)^2 - \frac{1}{2} \left(\eta_{\mathrm{eff}} f_{\max}\right)^3 \right\} t.$$

By observing that $\boldsymbol{u}_t \cdot \boldsymbol{u} - \boldsymbol{u}_0 \cdot \boldsymbol{u} < 1$ since $\boldsymbol{u}_0 \cdot \boldsymbol{u} > 0$ we obtain the time bound

$$t < \frac{f_{\max}}{f_{\min}} \left\{ \frac{\gamma_{\mathrm{d}} - \beta}{R} \left(\eta_{\mathrm{eff}} f_{\max}\right) - \frac{1}{2} \left(1 + \frac{2\beta}{R}\right) \left(\eta_{\mathrm{eff}} f_{\max}\right)^2 - \frac{1}{2} \left(\eta_{\mathrm{eff}} f_{\max}\right)^3 \right\}^{-1}. \tag{5.75}$$

The above time bound can be optimised with respect to the parameter η_{eff}. The resulting optimal value of η_{eff} is approximately given by

$$\eta_{\mathrm{eff\,opt}} = \frac{1}{f_{\max}} \frac{(\gamma_{\mathrm{d}} - \beta)}{R} \left(1 + \frac{2\beta}{R}\right)^{-1}.$$

Substituting the optimal value of η_{eff} into (5.75) we obtain the optimised time bound

$$t < t_{\mathrm{b_1}} \equiv 2 \frac{f_{\max}}{f_{\min}} \frac{R^2}{(\gamma_{\mathrm{d}} - \beta)^2} \left(1 + \frac{2\beta}{R}\right) \left(1 - \frac{\gamma_{\mathrm{d}} - \beta}{R} \left(1 + \frac{2\beta}{R}\right)^{-2}\right)^{-1}. \tag{5.76}$$

From the above expression we observe that our time bound $t_{\mathrm{b_1}}$ is analogous to the one of the Perceptron without margin with the main differences being a factor of 2 and the replacement of γ_{d}^2 by $(\gamma_{\mathrm{d}} - \beta)^2$.

It is possible to proceed to a derivation of an upper bound on t following the Novikoff-like technique of [21]. Taking the inner product of (5.13) (with the denominator of its r.h.s. being denoted N_{t+1}) with the optimal direction $\boldsymbol{u}$, employing (5.1), (5.7) and repeatedly applying the resulting inequality we have

$$\begin{aligned} \boldsymbol{u}_t \cdot \boldsymbol{u} &= \frac{\boldsymbol{u}_{t-1} \cdot \boldsymbol{u} + \eta_{\mathrm{eff}} f_{t-1} \boldsymbol{y}_k \cdot \boldsymbol{u}/R}{N_t} \geq \frac{\boldsymbol{u}_{t-1} \cdot \boldsymbol{u}}{N_t} + \frac{\eta_{\mathrm{eff}} f_{\min} \gamma_{\mathrm{d}}/R}{N_t} \\ &\geq \frac{\boldsymbol{u}_0 \cdot \boldsymbol{u}}{N_t N_{t-1} \cdots N_1} + \frac{\eta_{\mathrm{eff}} f_{\min} \gamma_{\mathrm{d}}}{R} \left(\frac{1}{N_t} + \frac{1}{N_t N_{t-1}} + \cdots + \frac{1}{N_t N_{t-1} \cdots N_1} \right). \end{aligned} \tag{5.77}$$

For the normalisation factor N_m we can derive the inequality

$$N_m^{-1} \geq \left(1 + \eta_{\mathrm{eff}}^2 f_{\max}^2 + 2\eta_{\mathrm{eff}} f_{\max} \frac{\beta}{R}\right)^{-\frac{1}{2}} \equiv r$$

which combined with (5.77) yields

$$1 > \frac{\eta_{\mathrm{eff}} f_{\min} \gamma_{\mathrm{d}}}{R} \sum_{m=1}^{t} r^m = \frac{\eta_{\mathrm{eff}} f_{\min} \gamma_{\mathrm{d}}}{R} r \frac{1 - r^t}{1 - r} \tag{5.78}$$

given that $\boldsymbol{u}_t \cdot \boldsymbol{u} \leq 1$ and $\boldsymbol{u}_0 \cdot \boldsymbol{u} > 0$. Equation (5.78) can be easily rewritten as

$$r^t > 1 - \frac{R}{\eta_{\text{eff}} f_{\min} \gamma_{\text{d}}} \left(\frac{1}{r} - 1 \right) \quad . \tag{5.79}$$

Since the l.h.s. of (5.79) is a monotonically decreasing function of t tending to 0 as $t \to \infty$ an upper bound on t is obtainable only if the r.h.s. of (5.79) is positive or equivalently if η_{eff} satisfies the inequality

$$\eta_{\text{eff}} < 2 \frac{(f_{\min} \gamma_{\text{d}} - f_{\max} \beta) R}{f_{\max}^2 R^2 - f_{\min}^2 \gamma_{\text{d}}^2} \quad . \tag{5.80}$$

Provided (5.80) is satisfied (5.79) gives the upper bound

$$t < t_{\text{b}_2} \equiv - \frac{\ln \left(1 - (\eta_{\text{eff}} f_{\min} \gamma_{\text{d}})^{-1} R \left(r^{-1} - 1 \right) \right)}{\ln r^{-1}} \tag{5.81}$$

on the number t of updates. From (5.80) it is apparent that this Novikoff-like proof technique does not lead to a proof of convergence in a finite number of steps unless $f_{\min} \gamma_{\text{d}} > f_{\max} \beta$. An analogous phenomenon was observed in Section 5.9.1 when again a Novikoff-like technique was employed. In contrast, proof techniques relying on stepwise convergence do not give rise to such restrictions. In the case $f_t = 1$ the bound (5.80) on η_{eff} coincides with the critical value of (5.73). If we assume $f_t = 1$ we see that as $\beta \to \gamma_{\text{d}}$ the bound of (5.80) tends to 0 linearly with $\gamma_{\text{d}} - \beta$. Taking the limit $\eta_{\text{eff}} \to 0$ in (5.81) we obtain $t_{\text{b}_2} \to -(\eta_{\text{eff}} \beta / R)^{-1} \ln((\gamma_{\text{d}} - \beta)/\gamma_{\text{d}})$ implying that in the limit $\beta \to \gamma_{\text{d}}$, assuming $\eta_{\text{eff}} \sim (\gamma_{\text{d}} - \beta)/R$, the bound t_{b_2} behaves like $t_{\text{b}_2} \sim (\gamma_{\text{d}} - \beta)^{-1} \gamma_{\text{d}}^{-1} R^2 \ln((\gamma_{\text{d}} - \beta)^{-1} \gamma_{\text{d}})$. An analogous behaviour is exhibited by the bound (5.71) in Section 5.9.1. The bound t_{b_1} of (5.76), instead, goes to infinity like $(\gamma_{\text{d}} - \beta)^{-2} R^2$.

5.9.3 Mistake-Controlled Rule Algorithms with Fixed Margin Condition

Require: A linearly separable augmented training set with reflection assumed $S = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_l)$

Define:
For $k = 1, \ldots, l$
$R = \max_k \|\boldsymbol{y}_k\|$, $\bar{\boldsymbol{y}}_k = \boldsymbol{y}_k / R$

Input: $\bar{\beta} \, (= \beta / R)$, η

Initialisation:
$t = 1$, $\boldsymbol{u}_1 = \bar{\boldsymbol{y}}_1 / \|\bar{\boldsymbol{y}}_1\|$, $\eta_{\text{eff}\,1} = \eta$

repeat until no update made within the **for** loop
  for $k = 1$ to l **do**
    if $\boldsymbol{u}_t \cdot \bar{\boldsymbol{y}}_k \leq \bar{\beta}$ **then**
      $\boldsymbol{u}_{t+1} = \frac{\boldsymbol{u}_t + \eta_{\text{eff}\,t} f_t \bar{\boldsymbol{y}}_k}{\|\boldsymbol{u}_t + \eta_{\text{eff}\,t} f_t \bar{\boldsymbol{y}}_k\|}$
      $t \leftarrow t + 1$
      $\eta_{\text{eff}\,t} = \eta / t^{\zeta}$

FIGURE 5.8: Mistake-controlled rule algorithm with fixed margin condition.

For the sake of completeness we consider in the present section algorithms with the fixed margin misclassification condition of (5.11) and an effective learning rate given explicitly as the inverse of a power of the number of mistakes, i.e.

$$\eta_{\mathrm{eff}\,t} = \frac{\eta}{t^{\zeta}} \ . \tag{5.82}$$

Here η, ζ are positive constants and moreover

$$\zeta \leq 1 \ . \tag{5.83}$$

Since both (5.11) and (5.82) do not involve the length of the weight vector and provided f_t is also $\|\boldsymbol{a}_t\|$-independent we may use (5.13) as an update rule. Additionally, we assume that the initial unit length weight vector $\boldsymbol{u}_1$ is chosen in the direction of one of the $\boldsymbol{y}_k$'s such that $\boldsymbol{u}_t \cdot \boldsymbol{u} > 0$ for all t. A pseudocode description of the algorithms under consideration appears in Fig. 5.8.

For a proof of convergence of the present class of algorithms we rely on arguments based on the notion of stepwise convergence. Positivity of $\boldsymbol{u}_t \cdot \boldsymbol{u}$ allows us to use positivity of $\mathcal{D}$ of (5.15) as a criterion for stepwise convergence. Taking a closer look at $\mathcal{A}$ of (5.16) reveals that the $\eta_{\mathrm{eff}\,t}$-independent term remains positive throughout the algorithm. Furthermore, because of (5.82) the terms of $\mathcal{A}$ linear in $\eta_{\mathrm{eff}\,t}$, which are not necessarily positive, become less important with time leading to positivity of $\mathcal{A}$ and consequently of $\mathcal{D}$ for t larger than a critical time t_{c}. More specifically, using (5.1), (5.11) and (5.82) we can place a lower bound on $\mathcal{A}$

$$\mathcal{A} \geq \gamma_{\mathrm{d}} - \beta - \frac{\eta}{t^{\zeta}} \frac{f_{\max}}{2R} (R^2 - \gamma_{\mathrm{d}}^2) \ . \tag{5.84}$$

Requiring positivity of the r.h.s. of (5.84) the estimated time sufficient for the onset of stepwise convergence is

$$t_{\mathrm{c}} \equiv \left(\frac{\eta f_{\max}}{2} \frac{(R^2 - \gamma_{\mathrm{d}}^2)}{(\gamma_{\mathrm{d}} - \beta) R} \right)^{\frac{1}{\zeta}} \ . \tag{5.85}$$

Taking into account (5.14) and (5.84) and given that $\eta_{\mathrm{eff}\,t}$ does not decrease with t faster than $1/t$ because of (5.83) stepwise convergence implies convergence in a finite number of steps.

5.9.4 Algorithmic Implementations

In this section we briefly present algorithmic implementations which exploit the algorithms with fixed directional margin condition in order to find solution hyperplanes with almost optimal directional margin.

Our first implementation makes repeated use of the fixed directional margin algorithms only. In each round of its application the algorithm looks for a fixed unrelaxed directional

margin β according to the classification condition $\boldsymbol{u}_t \cdot \boldsymbol{y}_k > \beta$. Each round lasts until the condition is satisfied by all the training patterns or until an upper bound on the number of checks over the training set is reached. The range of values that β can take and therefore the interval that the algorithm should search extends from 0 to $r = \min_k \|\boldsymbol{y}_k\|$. The search can be performed efficiently by using a procedure similar to the Bolzano-bisection method. Initially a margin $\beta = \frac{r}{2}$ is asked for with a step parameter being set to $\frac{r}{2}$. If the algorithm comes up with a solution vector $\boldsymbol{a}$ satisfying the imposed margin constraint without exhausting the upper number of checks the round is considered successful. The weight vector $\boldsymbol{a}$ is stored as the best solution found so far and is exploited as the initial value $\boldsymbol{a}_0$ of the next trial. This way the procedure of finding a better solution in a subsequent round is speeded up substantially since such an $\boldsymbol{a}$ probably lies closer to a weight vector which gives rise to a larger margin than the weight vector $\boldsymbol{a}_0 = \boldsymbol{0}$ (or $\boldsymbol{a}_0 = \boldsymbol{y}_1 / \|\boldsymbol{y}_1\|$) and thus constitutes a better guess as an initial condition. One could also envisage using the final weight vector of an unsuccessful previous round as the initial weight vector of a subsequent one until the first successful trial is reached. At the end of each trial the step is divided by 2. In the case that a trial ends successfully the target value of the margin β in the next round is calculated by adding to the previous one the present step otherwise β is reduced by the same amount. Therefore, on the condition that the upper number of checks is set to a sufficiently large value, the procedure guarantees that the deviation of the margin β from the maximum one is reduced by a factor of 2 in each round. The algorithm is terminated when the step reaches a certain predefined desirable level, thereby determining dynamically the number of rounds.

A second possibility is to first use the standard Perceptron algorithm with margin in order to obtain a solution with a guaranteed fraction of the existing directional margin given by (5.23) and then attempt to incrementally boost [1] the margin found this way by repeatedly employing the fixed directional margin condition algorithms. The initial condition of each round of boosting will be the final weight vector of the previous round and the step by which the target margin increases will be determined as a fraction of the margin found in the first stage. The algorithm ends with the first unsuccessful trial. An analogous boosting procedure could follow a first stage of successful employment of the Bolzano-bisection method.

Finally, a third possibility is to use the algorithms of Section 5.9.1 in an incremental way. Such an implementation assumes the existence of a certain minimum value of the margin from which the algorithm starts searching incrementally. An estimate of an upper bound of the margin, which will be useful in determining the step of the incremental search, can be obtained by running a standard Perceptron algorithm and employing Novikoff's time bound (5.22). Thus, even if the Perceptron algorithm does not converge after M mistakes the directional margin γ_{d} is bounded from above by $\sqrt{(R^2 + 2b/\eta)/M}$.

[1] Boosting in this context should not be confused with the one of [49].

Chapter 6

Linearly Inseparable Feature Spaces

6.1 Introduction

Until now we made the assumption that the training patterns are linearly separable with margin γ_{d} in the augmented feature space with respect to hyperplanes passing through the origin. This enabled us to use the Perceptron-like algorithms of Chapter 5 to achieve separation of the data with some positive margin. Separability in the feature space may be achieved through the introduction of kernels [2, 9, 66, 48, 54] which map the data into a higher dimensional space. If the number of dimensions of that space is large enough the capacity of the hypothesis functions viewed in the original space is sufficiently high in order for the data to be separated. However, this procedure has the danger of overfitting the data leading to poor generalisation. In the case that the patterns are not linearly separable the algorithms of Chapter 5 are not able to converge to a solution but oscillate continually in their effort to correct instantaneously a misclassified pattern.

An old technique that bypasses this difficulty and is applicable to both linearly separable and linearly inseparable data is the one of the minimum squared error. This is done by seeking the hyperplane that separates the patterns with a fixed functional margin $\boldsymbol{a} \cdot \boldsymbol{y}_k = 1, \forall k$. This amounts to solving the system of linear equalities that are induced by the linear constraints imposed by each one of the patterns. In its general form this system does not have a solution in the ordinary sense since it can be overdetermined but it may still accept a solution that minimises the squared error $\sum_k (\boldsymbol{a} \cdot \boldsymbol{y}_k - 1)^2$. However, this procedure cannot guarantee a vanishing training error in the separable case.

Another way of dealing with situations where the data are not linearly separable in the feature space is to employ techniques which instead of insisting on finding the maximal

margin hyperplane following the so-called hard margin policy they adopt a more tolerant point of view represented by the so-called soft margin. The hard margin is determined only by those training patterns lying closest to the separating hyperplane. In the soft margin approach, instead, all the patterns which fail to satisfy a target margin value with respect to a given hyperplane play a role. Thus, optimality could lead to a few points failing to meet the margin requirement or even being misclassified as long as the distance of the majority from the hyperplane exceeds the predefined margin value.

The notion of soft margin first appeared in the context of linear programming [6] in order to deal with inseparable data but became very popular with the advent of SVMs. As we already discussed in Chapter 3 the σ-norm soft margin problem in the SVM formulation is stated as the optimisation task

$$\text{minimise}_{\boldsymbol{w},b,\xi_k} \quad \|\boldsymbol{w}\|^2 + C\sum_{k=1}^{l}\xi_k^\sigma$$

$$\text{subject to} \quad y_k(\boldsymbol{w}\cdot\boldsymbol{x}_k + b) \geq 1 - \xi_k \quad \forall k,$$

$$\xi_k \geq 0 \quad \forall k.$$

Here $\boldsymbol{w}$ is the weight vector and the quantity C appearing in the objective function is a positive constant. The quantities ξ_k, as many as the training patterns, are called slack variables [55] and are introduced in order to allow for violations of the margin condition by some training patterns. Notice that in the above optimisation the patterns as well as the weight vector are not augmented. For this reason there is a bias term b and the labels y_k accompany explicitly the patterns.

Freund and Shapire [17] have shown how a function of the margin distribution different from the minimum margin one can be used to bound the number of mistakes of an online Perceptron algorithm, thereby providing one possible extension of Novikoff's theorem for the inseparable case. Their proof technique, very similar to that of [33], extends the instance space by as many dimensions as the number of patterns placing each pattern at a distance $|\Delta|$ from the origin in the corresponding dimension. As a result the training set in the extended space becomes linearly separable. This technique was also used in order to derive generalisation error bounds involving the new margin distribution [52, 53]. An interesting result in this connection is the observation (see Section 3.4) that the hard margin optimisation task in the extended space is equivalent to the soft margin optimisation in the original instance space if the 2-norm of the slack variables ($\sigma = 2$ in the above discussion) is employed [53].

In the sequel, following the approach of [17], we show how one moves in the direction of minimising an objective function $\mathcal{J}$ involving the new margin distribution by making use of Perceptron-like algorithms which, however, are seeking a hard margin in the extended

space [57]. This may not be surprising in the light of the result just mentioned regarding the equivalence between the hard margin optimisation in the extended space and the soft margin one in the original space. Nevertheless, we hope that our analysis, which does not rely on convex optimisation theory, will contribute to a better understanding of what an algorithm running in the extended space actually achieves with respect to the original space. We also provide some results which may be regarded as generalisations of Novikoff's theorem for the Perceptron algorithm with margin to the inseparable case. The theorem of Freund and Shapire that we mentioned earlier belongs to this category of results. Although our instance space prior to its extension is the augmented one in the present chapter the instances $\boldsymbol{y}_k$ are explicitly accompanied by their labels y_k since we found it convenient not to assume a reflection with respect to the origin.

6.2 A Soft Margin Approach for Perceptron-Like Large Margin Classifiers

Theorem 6.1. *Let $((\boldsymbol{y}_1, y_1), \ldots, (\boldsymbol{y}_l, y_l))$ be a sequence of l labelled instances, $\boldsymbol{u}$ a unit vector and $\gamma > 0$. Define $d_i = \max\{0, \gamma - y_i \boldsymbol{u} \cdot \boldsymbol{y}_i\}$ and set $D = \sqrt{\sum_i d_i^2}$. In addition define an extended instance space $\boldsymbol{y}_i^{\text{ext}} = (\boldsymbol{y}_i, \Delta\delta_{1i}, \ldots, \Delta\delta_{li})$ parametrised by Δ, where δ_{ij} is Kronecker's δ.*

1. *Let $\Gamma_{\Delta\text{opt}}$ be the maximum margin in the extended space with respect to hyperplanes passing through the origin. Then, for any $\boldsymbol{u}$ and γ,*

$$\frac{1}{\Gamma_{\Delta\text{opt}}^2} \leq \mathcal{J}(\boldsymbol{u}, \gamma, \Delta) \equiv \frac{1}{\gamma^2} + \frac{1}{\Delta^2}\left(\frac{D}{\gamma}\right)^2 \quad . \tag{6.1}$$

2. *Assume that a zero-threshold algorithm converges in the extended space to a solution vector $\boldsymbol{a}^{\text{ext}}$ which describes a hyperplane passing through the origin with margin Γ_Δ. Let $\boldsymbol{u} = \boldsymbol{a}/\|\boldsymbol{a}\|$ and $\gamma = \Gamma_\Delta \left\|\boldsymbol{a}^{\text{ext}}\right\| / \|\boldsymbol{a}\|$, where $\boldsymbol{a}$ is the projection of $\boldsymbol{a}^{\text{ext}}$ onto the original instance space. Then, employing such a $\boldsymbol{u}$ and γ provided by the algorithm, we have*

$$\frac{1}{\Gamma_{\Delta\text{opt}}^2} \leq \mathcal{J}(\boldsymbol{u}, \gamma, \Delta) \leq \frac{1}{\Gamma_\Delta^2} \quad . \tag{6.2}$$

Proof. 1. Notice that

$$\mathcal{J}(\boldsymbol{u}, \gamma, \Delta) = \frac{Z^2}{\gamma^2}$$

with

$$Z = \sqrt{1 + \frac{D^2}{\Delta^2}} \quad .$$

Then, (6.1) is equivalent to $\gamma/Z \leq \Gamma_{\Delta\text{opt}}$ which is proved in [17]. For the sake of completeness we repeat the proof here. For an arbitrary unit vector $\boldsymbol{u}$ and any $\gamma > 0$ let us consider the extended unit-length prediction vector

$$\boldsymbol{u}_{\text{d}} = \frac{1}{Z}\left(\boldsymbol{u}, y_1\frac{d_1}{\Delta}, \ldots, y_i\frac{d_i}{\Delta}, \ldots, y_l\frac{d_l}{\Delta}\right) \ . \tag{6.3}$$

We have

$$y_i\boldsymbol{u}_{\text{d}} \cdot \boldsymbol{y}_i^{\text{ext}} = \frac{1}{Z}\left(y_i\boldsymbol{u}\cdot\boldsymbol{y}_i + d_i\right) \geq \frac{1}{Z}\left(y_i\boldsymbol{u}\cdot\boldsymbol{y}_i + (\gamma - y_i\boldsymbol{u}\cdot\boldsymbol{y}_i)\right) = \frac{\gamma}{Z} \tag{6.4}$$

which demonstrates that the extended prediction vector $\boldsymbol{u}_{\text{d}}$ achieves a margin of at least γ/Z. Obviously, if $\Gamma_{\Delta\text{opt}}$ is the maximum margin in the extended space with respect to hyperplanes passing through the origin

$$\frac{\gamma}{Z} \leq \Gamma_{\Delta\text{opt}} \ . \tag{6.5}$$

2. Let us assume that a zero-threshold algorithm converges in the extended space to a weight vector $\boldsymbol{a}^{\text{ext}}$

$$\boldsymbol{a}^{\text{ext}} = \|\boldsymbol{a}\|\left(\boldsymbol{u}, y_1\frac{d_1'}{\Delta}, \ldots, y_i\frac{d_i'}{\Delta}, \ldots, y_l\frac{d_l'}{\Delta}\right) \ .$$

Here $\boldsymbol{a}$ is the projection of $\boldsymbol{a}^{\text{ext}}$ onto the original instance space and $\boldsymbol{u}$ is the unit vector in the direction of $\boldsymbol{a}$. Let Γ_Δ be the margin achieved by

$$\boldsymbol{u}^{\text{ext}} = \frac{\boldsymbol{a}^{\text{ext}}}{\|\boldsymbol{a}^{\text{ext}}\|} = \frac{1}{Z'}\left(\boldsymbol{u}, y_1\frac{d_1'}{\Delta}, \ldots, y_i\frac{d_i'}{\Delta}, \ldots, y_l\frac{d_l'}{\Delta}\right) \ ,$$

where

$$Z' = \frac{\|\boldsymbol{a}^{\text{ext}}\|}{\|\boldsymbol{a}\|} = \sqrt{1 + \frac{D'^2}{\Delta^2}}$$

with $D' = \sqrt{\sum_i d_i'^2}$ and let us define

$$\gamma = \Gamma_\Delta Z' \ .$$

We have

$$y_i\boldsymbol{u}^{\text{ext}} \cdot \boldsymbol{y}_i^{\text{ext}} = \frac{1}{Z'}\left(y_i\boldsymbol{u}\cdot\boldsymbol{y}_i + d_i'\right) \geq \Gamma_\Delta = \frac{\gamma}{Z'} \tag{6.6}$$

from where

$$d_i' \geq \gamma - y_i\boldsymbol{u}\cdot\boldsymbol{y}_i \ . \tag{6.7}$$

The above inequality, taking into account the definition of d_i, leads to

$$|d_i'| \geq d_i \geq 0 \tag{6.8}$$

and consequently to

$$Z' \geq Z \ . \tag{6.9}$$

Therefore, taking into consideration the definition of γ we obtain

$$\frac{\gamma}{Z} \geq \frac{\gamma}{Z'} = \Gamma_\Delta \ . \tag{6.10}$$

We see that the extended prediction vector $\boldsymbol{u}_\mathrm{d}$ of (6.3) constructed by making use of $\boldsymbol{u}$ and γ which are obtained from the solution vector $\boldsymbol{u}^\mathrm{ext}$ achieves a margin at least as large as the one achieved by $\boldsymbol{u}^\mathrm{ext}$. The last inequality leads to

$$\mathcal{J}(\boldsymbol{u}, \gamma, \Delta) \leq \Gamma_\Delta^{-2} \tag{6.11}$$

given that $Z^2/\gamma^2 = \mathcal{J}(\boldsymbol{u}, \gamma, \Delta)$. The proof is completed by combining (6.1) and (6.11).

□

Remark 6.2. Let us assume that the zero-threshold algorithm is a Perceptron-like algorithm with update rule

$$\boldsymbol{a}_{t+1}^\mathrm{ext} = \boldsymbol{a}_t^\mathrm{ext} + \eta f_t y_k \boldsymbol{y}_k^\mathrm{ext} \ ,$$

where $\eta f_t > 0$, and initial condition $\boldsymbol{a}_0^\mathrm{ext} = \sum_k \alpha_k y_k \boldsymbol{y}_k^\mathrm{ext}$ with $\alpha_k \geq 0$. From the above initialisation, the update rule and the definition of the extended space we have that $d_i' \geq 0$.

Remark 6.3. If the algorithm converges to the maximal margin hyperplane passing through the origin in the extended space then $\Gamma_\Delta = \Gamma_{\Delta\mathrm{opt}}$. Moreover, (6.1) is equivalent to $\gamma/Z \leq \Gamma_{\Delta\mathrm{opt}}$ which combined with (6.10) and given that $\Gamma_\Delta = \Gamma_{\Delta\mathrm{opt}}$ gives $Z' = Z$ or $D' = D$ from where $|d_i'| = d_i$ follows taking into account (6.8). In addition, $d_i' \geq 0$. Indeed, if $d_i' < 0$ then $d_i = 0$ because of (6.7) and the definition of d_i. But in this case $d_i' = d_i = 0$ contradicting our assumption that $d_i' < 0$. Thus, for the optimal extended space solution $d_i' = d_i$.

Remark 6.4. Setting $\boldsymbol{w} = \boldsymbol{u}/\gamma$, $\xi_i = |d_i'|/\gamma \geq d_i/\gamma = \max\{0, 1 - y_i \boldsymbol{w} \cdot \boldsymbol{y}_i\}$ and $C = \Delta^{-2}$ yields

$$\frac{1}{\gamma^2} + \frac{1}{\Delta^2}\left(\frac{D'}{\gamma}\right)^2 = \|\boldsymbol{w}\|^2 + C\sum_i \xi_i^2 \ .$$

We recognise the objective function of the primal form of the 2-norm soft margin optimisation problem in which the role of the constraints is played by (6.8) but the bias term is missing since it is, at least partially, incorporated in the augmented weight vector $\boldsymbol{w}$. If the optimal solution is found $d_i' = d_i$ and the "slack" variables ξ_i become $\xi_i = \max\{0, 1 - y_i \boldsymbol{w} \cdot \boldsymbol{y}_i\}$.

Theorem 6.1 shows that minimisation of the objective function $\mathcal{J}$ is equivalent to finding the maximum margin in the extended space. The $\boldsymbol{u}$ and γ for which the minimum

$\mathcal{J}_{\min}$ is attained determine uniquely both the maximum margin $\Gamma_{\Delta\text{opt}} = \mathcal{J}_{\min}^{-\frac{1}{2}}$ and the direction $\boldsymbol{u}_{\text{opt}}^{\text{ext}}$ of the optimal weight vector in the extended space which is given by (6.3). Moreover, (6.2) provides an estimate of the deviation of the value of $\mathcal{J}$ achieved as a result of an incomplete optimisation from $\mathcal{J}_{\min}$ if we have an estimate of the difference between Γ_Δ and $\Gamma_{\Delta\text{opt}}$.

Theorem 6.5. *Let $((\boldsymbol{y}_1, y_1), \ldots, (\boldsymbol{y}_l, y_l))$ be a sequence of l labelled instances out of which the last $(l-n)$ are separable by a zero-threshold hyperplane. Also let $\boldsymbol{u}$ be a unit vector and $\gamma > 0$. Define $d_i = \max\{0, \gamma - y_i \boldsymbol{u} \cdot \boldsymbol{y}_i\}$ and set $D = \sqrt{\sum_i d_i^2}$. In addition define an extended instance space parametrised by Δ in which the i-th instance is $\boldsymbol{y}_i^{\text{ext}} = (\boldsymbol{y}_i, \Delta\delta_{1i}, \ldots, \Delta\delta_{ni})$.*

1. *Let $\Gamma_{\Delta\text{opt}}$ be the maximum margin in the extended space with respect to hyperplanes passing through the origin. Then for any $\boldsymbol{u}$ and γ satisfying $d_i = 0$ for $i > n$*

$$\frac{1}{\Gamma_{\Delta\text{opt}}^2} \leq \mathcal{J}(\boldsymbol{u}, \gamma, \Delta) \equiv \frac{1}{\gamma^2} + \frac{1}{\Delta^2}\left(\frac{D}{\gamma}\right)^2 \quad .$$

2. *Assume that a zero-threshold algorithm converges in the extended space to a solution vector $\boldsymbol{a}^{\text{ext}}$ which describes a hyperplane passing through the origin with margin Γ_Δ. Let $\boldsymbol{u} = \boldsymbol{a}/\|\boldsymbol{a}\|$ and $\gamma = \Gamma_\Delta \left\|\boldsymbol{a}^{\text{ext}}\right\| / \|\boldsymbol{a}\|$, where $\boldsymbol{a}$ is the projection of $\boldsymbol{a}^{\text{ext}}$ onto the original instance space. Then, employing such a $\boldsymbol{u}$ and γ provided by the algorithm, we have $d_i = 0$ for $i > n$ and*

$$\frac{1}{\Gamma_{\Delta\text{opt}}^2} \leq \mathcal{J}(\boldsymbol{u}, \gamma, \Delta) \leq \frac{1}{\Gamma_\Delta^2} \quad .$$

Proof. 1. Let us equivalently define the extended instance space such that the extended i-th instance becomes

$$\boldsymbol{y}_i^{\text{ext}} = (\boldsymbol{y}_i, \Delta\delta_{1i}, \ldots, \Delta\delta_{ni}, 0\delta_{(n+1)i}, \ldots .0\delta_{li}) \quad . \tag{6.12}$$

Then, the argument is quite analogous to the one that led to the corresponding statement in Theorem 6.1. The only difference is that (6.4), although for $i \leq n$ holds as it is, for $i > n$ has to be slightly reexpressed as follows

$$y_i \boldsymbol{u}_{\text{d}} \cdot \boldsymbol{y}_i^{\text{ext}} = \frac{1}{Z}(y_i \boldsymbol{u} \cdot \boldsymbol{y}_i) = \frac{1}{Z}(y_i \boldsymbol{u} \cdot \boldsymbol{y}_i + d_i) \geq \frac{1}{Z}(y_i \boldsymbol{u} \cdot \boldsymbol{y}_i + (\gamma - y_i \boldsymbol{u} \cdot \boldsymbol{y}_i)) = \frac{\gamma}{Z} \quad .$$

Here use has been made of the fact that $d_i = 0$ for $i > n$.

2. Let us enlarge the extended instance space as in (6.12) and trivially embed the solution vector found by the algorithm into this enlarged space. It immediately follows that $d_i' = 0$ for $i > n$. We may then repeat the arguments that led to the corresponding statement in Theorem 6.1. The only difference is that (6.6),

although for $i \leq n$ holds as it is, for $i > n$ has to be slightly rewritten as

$$y_i \boldsymbol{u}^{\text{ext}} \cdot \boldsymbol{y}_i^{\text{ext}} = \frac{1}{Z'} (y_i \boldsymbol{u} \cdot \boldsymbol{y}_i) = \frac{1}{Z'} \left(y_i \boldsymbol{u} \cdot \boldsymbol{y}_i + d_i' \right) \geq \Gamma_\Delta = \frac{\gamma}{Z'} \ .$$

Here we made use of the fact that $d_i' = 0$ for $i > n$. From (6.8) follows that $d_i = 0$ for $i > n$ given that $d_i' = 0$ for $i > n$.

□

Theorem 6.5 shows that in the case that it is known that a subset of the dataset is linearly separable it is possible by defining an appropriate minimally extended instance space to obtain minimisation of the objective function $\mathcal{J}$ subject to the constraints that the d_i's corresponding to the instances which belong to the separable subset vanish.

We conclude this section with a well-known lower bound on the margin $\Gamma_{\Delta\text{opt}}$ of the extended space which, in contrast to the bound (6.5), has the advantage of depending only on Δ and the number l of instances. Let $\boldsymbol{u}^{\text{ext}} = \text{sign}(\Delta) l^{-\frac{1}{2}} (\boldsymbol{0}, y_1, y_2, \ldots, y_l)$ be an extended unit vector with vanishing projection onto the original instance space. It is straightforward to see that

$$y_i \boldsymbol{u}^{\text{ext}} \cdot \boldsymbol{y}_i^{\text{ext}} = \frac{|\Delta|}{\sqrt{l}}$$

meaning that $\boldsymbol{u}^{\text{ext}}$ achieves a margin of $|\Delta|/\sqrt{l}$. Thus,

$$\Gamma_{\Delta\text{opt}} \geq \frac{|\Delta|}{\sqrt{l}} \ . \tag{6.13}$$

6.3 Generalising Novikoff's Theorem to the Inseparable Case

The following theorem generalises the theorem of Freund and Shapire [17] to the case of the Perceptron algorithm with margin.

Theorem 6.6. *Let $((\boldsymbol{y}_1, y_1), \ldots, (\boldsymbol{y}_l, y_l))$ be a sequence of labelled instances with $\|\boldsymbol{y}_i\| \leq R$. Also let $\boldsymbol{u}$ be a unit vector and $\gamma > 0$. Define $d_i = \max\{0, \gamma - y_i \boldsymbol{u} \cdot \boldsymbol{y}_i\}$ and set $D = \sqrt{\sum_i d_i^2}$. Then the number of mistakes of the online Perceptron algorithm with learning rate η and margin parameter b on this sequence is bounded by*

$$\left(\frac{\sqrt{R^2 + 2b/\eta} + D}{\gamma} \right)^2 \ .$$

Proof. The extended instances satisfy $\left\| \boldsymbol{y}_i^{\text{ext}} \right\| \leq R_{\max}$ with $R_{\max} = \sqrt{R^2 + \Delta^2}$. Moreover, according to Theorem 6.1 for the maximum margin $\gamma_{\max}$ in the extended space

with respect to zero-threshold hyperplanes we have

$$\frac{1}{\gamma_{\max}^2} = \frac{1}{\Gamma_{\Delta\text{opt}}^2} \leq \frac{1}{\gamma^2} + \frac{1}{\Delta^2}\left(\frac{D}{\gamma}\right)^2 \quad .$$

Then the upper bound on the number of steps t until convergence of the Perceptron algorithm in the extended space follows immediately by combining the above inequality with Novikoff's bound (5.22) (with $f_{\max} = f_{\min} = 1$)

$$t \leq \frac{1}{\gamma_{\max}^2}\left(R_{\max}^2 + \frac{2b}{\eta}\right) \leq \frac{1}{\gamma^2}\left(1 + \frac{D^2}{\Delta^2}\right)\left(R^2 + \Delta^2 + \frac{2b}{\eta}\right) \quad . \tag{6.14}$$

The r.h.s. of (6.14) is optimised for $\Delta^2 = D\sqrt{R^2 + 2b/\eta}$ leading to the bound stated in Theorem 6.6. The proof is completed by observing that the Perceptron makes exactly the same mistakes in the original and in the extended space during the first epoch. □

Remark 2.4. Setting $b = 0$ in Theorem 6.6 we obtain the theorem of Freund and Shapire [17].

Theorem 6.7. *Let $((\boldsymbol{y}_1, y_1), \ldots, (\boldsymbol{y}_l, y_l))$ be a sequence of labelled instances with $\|\boldsymbol{y}_i\| \leq R$. Also let $\boldsymbol{u}$ be a unit vector and $\gamma > 0$. Define $d_i = \max\{0, \gamma - y_i \boldsymbol{u} \cdot \boldsymbol{y}_i\}$ and set $D = \sqrt{\sum_i d_i^2}$. For each value of the parameter Δ let*

$$\mathcal{J}_{\min}(\Delta) \equiv \min_{\boldsymbol{u},\gamma}\left\{\frac{1}{\gamma^2} + \frac{1}{\Delta^2}\left(\frac{D}{\gamma}\right)^2\right\} \quad .$$

The Perceptron algorithm with margin parameter b and learning rate η converges in an extended space in which the i-th instance is $\boldsymbol{y}_i^{\text{ext}} = (\boldsymbol{y}_i, \Delta\delta_{1i}, \ldots, \Delta\delta_{li})$ in at most

$$\mathcal{J}_{\min}(\Delta)\left(R^2 + \Delta^2 + \frac{2b}{\eta}\right)$$

steps (updates) to a solution vector $\boldsymbol{a}^{\text{ext}}$ describing a zero-threshold hyperplane with margin Γ_Δ. Let $\boldsymbol{u} = \boldsymbol{a}/\|\boldsymbol{a}\|$ and $\gamma = \Gamma_\Delta \left\|\boldsymbol{a}^{\text{ext}}\right\| / \|\boldsymbol{a}\|$, where $\boldsymbol{a}$ is the projection of $\boldsymbol{a}^{\text{ext}}$ onto the original instance space. Then, employing such a $\boldsymbol{u}$ and γ provided by the Perceptron algorithm, we have

$$\mathcal{J}_{\min}(\Delta) \leq \frac{1}{\gamma^2} + \frac{1}{\Delta^2}\left(\frac{D}{\gamma}\right)^2 \leq \left(2 + \frac{\eta}{b}(R^2 + \Delta^2)\right)^2 \mathcal{J}_{\min}(\Delta) \quad .$$

Proof. The extended instances satisfy $\left\|\boldsymbol{y}_i^{\text{ext}}\right\| \leq R_{\max}$ with $R_{\max} = \sqrt{R^2 + \Delta^2}$. Moreover, according to Theorem 6.1, $1/\sqrt{\mathcal{J}_{\min}(\Delta)}$ is the maximum margin $\gamma_{\max}$ in the extended space with respect to zero-threshold hyperplanes. Then the upper bound on the number of steps t until convergence of the Perceptron algorithm in the extended space

follows immediately from Novikoff's bound (5.22) (with $f_{\max} = f_{\min} = 1$)

$$t \leq \frac{1}{\gamma_{\max}^2} \left(R_{\max}^2 + \frac{2b}{\eta} \right) \ .$$

Additionally, notice that

$$\Gamma_\Delta \geq f_{\mathrm{b}} \gamma_{\max} = \frac{f_{\mathrm{b}}}{\sqrt{\mathcal{J}_{\min}(\Delta)}}$$

or equivalently

$$\frac{1}{\Gamma_\Delta^2} \leq f_{\mathrm{b}}^{-2} \mathcal{J}_{\min}(\Delta) \ . \tag{6.15}$$

Here

$$f_{\mathrm{b}} = \left(2 + \frac{\eta}{b} R_{\max}^2 \right)^{-1} = \left(2 + \frac{\eta}{b} (R^2 + \Delta^2) \right)^{-1}$$

is the guaranteed fraction of the maximum margin achieved by the Perceptron in the extended space (see (5.23) with $f_{\max} = f_{\min} = 1$). Substituting (6.15) in (6.2) completes the proof. □

Theorem 6.6 gives an upper bound on the number of mistakes that the Perceptron algorithm makes when running on the original linearly inseparable instance space during the first epoch only. Theorem 6.7, instead, provides an upper bound on the number of mistakes that the Perceptron algorithm makes when running until convergence on the linearly separable extended instance space. It is important to realise, however, that in this last case the bound involves the optimal value $\mathcal{J}_{\min}(\Delta) = \min_{\boldsymbol{u},\gamma} \mathcal{J}(\boldsymbol{u}, \gamma, \Delta)$ of the quantity $\mathcal{J}(\boldsymbol{u}, \gamma, \Delta) = \gamma^{-2} + \Delta^{-2} \left(D/\gamma \right)^2$ which refers to the original linearly inseparable instance space. Moreover, convergence of the Perceptron algorithm in the extended space achieves an incomplete optimisation of the quantity $\mathcal{J}(\boldsymbol{u}, \gamma, \Delta)$ in the original instance space.

Chapter 7

Implementation and Experiments

In the present chapter we provide experimental results aiming at verifying our analysis and assessing the ability of various algorithms described in previous chapters to achieve fast convergence to a certain approximation of the optimal hyperplane in the feature space where the patterns are linearly separable. First we perform a comparative study involving only Perceptron-like algorithms (PLAs). Subsequently, we describe a variation of the standard incremental scenario for such algorithms which enables us to reduce the computational cost. Finally, we attempt a comparison of PLAs with SVMs in which PLAs are represented by MICRA and SVMs by algorithms based on decomposition methods. We conclude the chapter with a brief evaluation of our experimental results.

7.1 Comparative Study of PLAs

The algorithms that will be involved in our comparative study of PLAs are the standard Perceptron with margin, ALMA$_2$, aggressive ROMMA, CRAMMA$^{\epsilon}$, MICRA$^{\epsilon,\zeta}$ and some algorithmic implementations, discussed in Section 5.9.4, which involve the fixed directional margin condition algorithms. For MICRA we use a β-independent η ($\delta = 0$) and ϵ, ζ values for which, in most cases, the analysis of Remark 5.5 applies. Our goal in this comparison involving only PLAs will be to obtain a given value of the margin in as few updates as possible.

Before we proceed we will try to be more specific about the algorithmic implementations based on the algorithms of Section 5.9. One such algorithmic implementation that will be considered uses a standard Perceptron algorithm with margin at a first stage in order to obtain an estimate of the margin that the dataset possesses. This is followed at a second stage by the constant effective learning rate algorithm of Section 5.9.2 (with $f_t = 1$) aiming at boosting the margin found by the Perceptron algorithm. The step in the boosting stage is set as a certain fraction of the margin β_{p} found by the Perceptron.

Also, the effective learning rate in the boosting stage is chosen in the form $\eta_{\text{eff}} = \lambda\beta/R$ with the coefficient λ parametrising our ignorance about the relevant quantity $\gamma_{\text{d}} - \beta$. Actually we employ a double boosting stage. When during the first such stage a certain predefined maximum number of epochs is exceeded unsuccessfully the second stage with an effective learning rate involving a smaller parameter λ takes place only once (i.e. only for the value of the margin condition for which the first stage of boosting was unsuccessful). The maximum number of epochs in the second stage of boosting is equal to the one in the first. The algorithm terminates even if the second stage is successful. This implementation will be denoted as "Perceptron+boosting". Another implementation uses a Bolzano bisection procedure in the first stage instead of the standard Perceptron with margin employing again the algorithms of Section 5.9.2 (with $f_t = 1$) both in the Bolzano and the boosting stage with η_{eff} parametrised as above. In all the experiments we set the minimum distance between the target margins in consecutive trials of the Bolzano procedure to 0.0001. Also in the Bolzano stage we use the final weight vector of an unsuccessful previous round as the initial weight vector of a subsequent round until the first successful trial is reached. In this case we choose a single boosting stage. In addition, the step of the boosting stage is fixed as a certain fraction of the largest value β_{b} of the margin condition for which the Bolzano bisection procedure is successful (which is very close to the margin found by the Bolzano stage). This implementation will be denoted as "Bolzano+boosting". We also employ the algorithm with fixed margin condition of Section 5.9.1 and an update rule involving the function $f_t = \frac{\beta_u - \boldsymbol{u}_t \cdot \boldsymbol{y}_k}{\|\boldsymbol{y}_k\|}$ with $\beta_u = 1.001\beta$ in an incremental way for a predefined number of steps with each step involving a predefined maximum number of epochs. As we noted in Section 5.9.4 such an implementation assumes a certain minimum value of the margin from which the algorithm starts searching incrementally and the knowledge of a reasonably good upper bound on the margin in order to determine the step of the incremental search. This algorithmic implementation will be denoted in the sequel as "Incremental". The experimental results that will be reported for the above algorithmic implementations do not correspond to separate runnings of the algorithm but are obtained as intermediate values during a single running on each dataset.

7.1.1 Separable Data

We first consider the case of linearly separable datasets. In such a case the feature space in our experiments will be the initial instance space.

7.1.1.1 The Sonar Dataset

The dataset of the sonar classification problem of [24] consists of 208 instances each with 60 attributes obtainable from the UCI repository [8]. The dataset represents the sonar signals bouncing off metal and rock cylinders. In all our experiments with this set the

TABLE 7.1: Experimental results for the reduced sonar dataset. The directional margin γ'_{d}, the number of epochs (eps) and updates (upds) are given for ALMA_2.

ALMA_2 with $\eta = \sqrt{2}$			
α	$10^3\gamma'_{\mathrm{d}}$	eps	upds
0.8	5.018	26,669	290,523
0.7	6.058	46,481	591,460
0.6	6.785	79,924	1,140,016
0.5	7.246	142,813	2,217,010
0.4	7.613	269,936	4,517,010
0.3	7.907	571,544	10,170,589
0.2	8.109	1,518,543	28,339,339
0.1	8.272	7,074,770	137,693,241
0.05	8.341	30,399,057	603,233,250

data are embedded in the augmented space at a distance $\rho = 1$ from the origin in the additional dimension.

TABLE 7.2: Experimental results for the reduced sonar dataset. The directional margin γ'_{d}, the number of epochs (eps) and updates (upds) are given for the Perceptron and $\mathrm{CRAMMA}^{0.5}$ with $\eta_{\mathrm{eff}} = 0.001(\frac{\beta}{R})^{-1}$.

Perceptron				$\mathrm{CRAMMA}^{0.5}$			
$\frac{b}{\eta R^2}$	$10^3\gamma'_{\mathrm{d}}$	eps	upds	$\frac{\beta}{R}$	$10^3\gamma'_{\mathrm{d}}$	eps	upds
0.7	5.164	15,651	189,313	0.58	5.110	17,170	190,542
1.02	5.846	19,366	251,534	0.78	5.840	23,806	271,193
1.795	6.600	27,793	402,849	1.13	6.594	34,409	432,445
3.9	7.266	53,388	820,261	1.685	7.291	60,569	785,941
5.4	7.453	71,912	1,117,124	2	7.461	79,658	1,047,756
20	7.802	252,938	3,977,612	3	7.816	153,699	2,143,988
30	7.845	376,879	5,930,214	3.1	7.847	161,573	2,273,854
90	7.906	1,119,031	17,647,271	3.92	7.991	240,255	3,502,155
200	7.923	2,479,699	39,131,402	6.2	8.183	535,618	8,350,654
1000	7.934	12,372,127	195,358,932	30	8.367	10,011,400	186,826,387

First we analyse the training dataset of the sonar classification problem, consisting of 104 instances, as selected for the aspect-angle dependent experiment in [24]. This subset of the full sonar dataset will be called here the reduced sonar dataset. If the choice $\rho = 1$ is made $R \simeq 3.8121$ and $\gamma_{\mathrm{d}} \simeq 0.00841$. Our experimental results for ALMA_2, the Perceptron, $\mathrm{CRAMMA}^{0.5}$, agg- ROMMA and $\mathrm{MICRA}^{0.05,0.9}$ are presented in Tables 7.1, 7.2 and 7.3. We see that ALMA_2 is the slowest by far in every respect. Also, $\mathrm{MICRA}^{0.05,0.9}$ is certainly the fastest as far as the number of updates is concerned with agg-ROMMA needing fewer epochs in the vicinity of the maximum margin γ_{d}. Moreover, the data suggest that the Perceptron is not able to obtain margins arbitrarily close to the maximum one. $\mathrm{CRAMMA}^{0.5}$, instead, with an effective learning rate scaling with β according to Theorem 5.2 (with $\delta = 1$) shows no difficulty in approaching γ_{d}. The same holds for $\mathrm{MICRA}^{0.05,0.9}$ since the condition of Theorem 5.4 is satisfied.

TABLE 7.3: Experimental results for the reduced sonar dataset. The directional margin γ'_{d}, the number of epochs (eps) and updates (upds) are given for the algorithms agg-ROMMA and MICRA$^{0.05,0.9}$.

agg-ROMMA				MICRA$^{0.05,0.9}$ with $\eta = 50$			
δ	$10^3\gamma'_{\mathrm{d}}$	eps	upds	$10^3\frac{\beta}{R}$	$10^3\gamma'_{\mathrm{d}}$	eps	upds
0.5	5.055	15,336	210,228	2.4	5.242	11,422	104,925
0.4	5.839	19,661	307,344	2.8	5.902	15,215	140,633
0.3	6.558	25,937	466,874	3.2	6.629	20,854	200,516
0.2	7.278	37,648	778,412	3.6	7.303	33,956	331,057
0.1	7.851	63,503	1,546,595	4.04	7.864	74,309	706,274
0.08	7.992	75,049	1,865,629	4.165	8.004	98,110	939,695
0.05	8.187	108,123	2,716,711	4.43	8.192	199,378	1,932,165
0.01	8.367	700,361	14,079,715	4.95	8.367	1,153,031	11,610,899

TABLE 7.4: Experimental results for the reduced sonar dataset. The directional margin γ'_{d}, the number of epochs (eps) and updates per epoch (upds/ep) are given for the CRAMMA^1 algorithm with the parameter η_0 taking the values $\eta_0 = 1, 2, 5$ and $\delta = 1$.

$\frac{\beta}{R}$	$\eta_0 = 1$			$\eta_0 = 2$			$\eta_0 = 5$		
	$10^3\gamma'_{\mathrm{d}}$	eps	$\frac{\text{upds}}{\text{ep}}$	$10^3\gamma'_{\mathrm{d}}$	eps	$\frac{\text{upds}}{\text{ep}}$	$10^3\gamma'_{\mathrm{d}}$	eps	$\frac{\text{upds}}{\text{ep}}$
1000	6.70	36,815	15.6						
2000	7.25	65,415	16.1						
3000	7.47	94,909	16.2						
5000	7.61	154,057	16.3	7.68	137,493	18.1			
7000	7.67	212,595	16.4	7.81	187,984	18.2			
10000	7.72	299,476	16.5	7.91	264,053	18.3			
20000	7.78	593,527	16.5	8.03	517,399	18.4	8.08	465,477	20.3
40000	7.82	1,179,629	16.5	8.08	1,024,704	18.4	8.22	905,121	20.5
60000	7.83	1,766,427	16.5	8.10	1,531,253	18.4	8.27	1,346,366	20.6

TABLE 7.5: Experimental results for the reduced sonar dataset. The directional margin γ'_{d}, the number of epochs (eps) and updates per epoch (upds/ep) are given for CRAMMA^1 with $\eta_{\mathrm{eff}} = 0.06(\frac{\beta}{R})^{-0.6}$.

$\frac{\beta}{R}$	1000	3000	5000	10000	20000	40000	60000	100000
$10^3\gamma'_{\mathrm{d}}$	6.69	7.45	7.67	7.93	8.10	8.22	8.26	8.31
eps	36,539	89,456	138,787	258,166	486,452	925,161	1,350,509	2,177,247
upds/ep	15.6	17.2	17.9	18.6	19.4	20.1	20.5	21.1

For values of $\epsilon \geq 1$ in CRAMMA η_{eff} can no longer scale with β like $\left(\frac{\beta}{R}\right)^{-1}$ if the algorithm is to approach γ_{d} arbitrarily close. This is illustrated in Tables 7.4, 7.5 and 7.6. From Table 7.4 we see that if $\epsilon = \delta = 1$ the directional margin achieved by CRAMMA approaches as β/R grows an upper bound which, however, becomes larger as η_0 becomes larger. This is in agreement with the analysis of the special cases following Theorem 5.2. Also, from Tables 7.5 and 7.6 we see that CRAMMA with $\epsilon = 1$ and $\epsilon = 2$ is able to approach γ_{d} arbitrarily close if δ takes the sufficiently small values $\delta = 0.6$ and $\delta = 0.3$, respectively (satisfying $0 < \epsilon\delta = 0.6 < 1$).

TABLE 7.6: Experimental results for the reduced sonar dataset. The directional margin $\gamma'_{\rm d}$, the number of epochs (eps) and updates per epoch (upds/ep) are given for the CRAMMA^2 algorithm with $\eta_{\rm eff} = 0.4(\frac{\beta}{R})^{-0.3}$.

$\frac{\beta}{R}$	10^6	10^7	10^8	10^9	10^{10}	10^{11}	10^{12}	10^{13}
$10^3\gamma'_{\rm d}$	1.03	3.66	5.52	6.69	7.37	7.80	8.10	8.27
eps	8,534	7,243	14,264	35,849	98,873	281,397	821,499	2,443,708
upds/ep	8.3	14.7	18.5	21.1	23.0	24.9	26.4	27.8

TABLE 7.7: The number of updates (upds) required to achieve $\gamma'_{\rm d} \simeq 0.00819$ in the reduced sonar dataset with MICRA and ALMA$_2$. For MICRA various ϵ, ζ values are considered and the η values employed are given.

ϵ, ζ	0.005, 0.99	0.05, 0.9	0.1, 0.8	0.15, 0.7	0.2, 0.6	0.2, 0.5	0.5, 0.5	ALMA$_2$
η	90	60	17	4.4	1.2	0.28	0.35	
upds/10^6	1.53	1.86	2.32	2.89	3.57	3.74	7.54	53.4

We also present in Table 7.7 the number of updates required to achieve a margin $\gamma'_{\rm d} \simeq 0.00819$ using MICRA with several ϵ, ζ values and ALMA$_2$. For ALMA$_2$ the accuracy parameter α was set to $\alpha = 0.1527$ ($\eta = \sqrt{2}$). From Table 7.7 it becomes clear that small ϵ's combined with relatively large ζ's lead to faster convergence. This is also consistent with our earlier observation that MICRA$^{0.05,0.9}$ is faster than CRAMMA$^{0.5}$.

TABLE 7.8: Experimental results for the reduced sonar dataset. The directional margin $\gamma'_{\rm d}$, the number of epochs (eps) and updates (upds) are given for the Incremental, the Perceptron+boosting and the Bolzano+boosting algorithms.

Incremental			Perceptron+boosting			Bolzano+boosting		
$10^3\gamma'_{\rm d}$	eps	upds	$10^3\gamma'_{\rm d}$	eps	upds	$10^3\gamma'_{\rm d}$	eps	upds
1.022	2,264	48,041	3.298	9,046	72,812	5.978	36,589	573,185
2.009	3,007	60,211	4.006	9,335	73,781	6.201	42,996	619,457
3.005	4,174	79,858	5.087	10,223	78,238	6.866	73,478	848,233
4.011	5,541	105,006	5.826	11,867	88,409	7.087	87,580	959,582
5.006	7,788	157,705	6.560	17,841	143,310	7.309	104,515	1,085,093
6.005	11,545	256,814	7.272	35,361	301,791	7.529	131,117	1,279,072
7.001	18,961	482,340	7.631	62,856	536,125	7.756	176,242	1,628,747
8.000	42,069	1,271,432	7.999	110,609	942,704	7.974	246,622	2,156,854

In Table 7.8 we present the experimental results on the reduced sonar dataset for the Incremental, the Perceptron+boosting and the Bolzano+boosting algorithms. In implementing the Incremental scenario we first attempted successfully to find a solution possessing a margin value of 0.0001 and subsequently starting from a value of 0.001 we proceeded in steps of 0.001. In the Perceptron+boosting scenario the relevant for the Perceptron stage parameter $b/(\eta R^2)$ was set to the value 0.1. Also, the step of the boosting scenario was chosen as $0.2\beta_{\rm p}$. The parameters λ controlling the effective learning rates of the two boosting stages were set to the values 0.1 and 0.03, respectively, whereas the maximum number of epochs in each step during the boosting stages

was set to 30,000. In the Bolzano stage of the Bolzano+boosting scenario we set the maximum number of epochs in each trial to 2,000 and the parameter λ controlling η_{eff} to the value 0.5. The step of the boosting scenario was set to $0.05\beta_{\text{b}}$, λ was given the value 0.1 and the maximum number of epochs in each boosting step was set to 50,000. Comparing the results of Table 7.8 with the ones of Tables 7.1, 7.2 and 7.3 we observe that the algorithmic implementations with fixed margin condition perform impressively well for values of the margin up to 90-95% of the maximum. More specifically, the Incremental algorithm is faster than agg-ROMMA with respect to the number of epochs whereas the Perceptron+boosting algorithm is as fast as MICRA with respect to the number of updates. Of course, the approximate algorithmic implementations involving the algorithms with fixed margin condition are not expected to be able to approach the maximum margin solution with infinite accuracy.

TABLE 7.9: Experimental results for the full sonar dataset. The directional margin γ'_{d}, the number of epochs (eps) and updates (upds) are given for ALMA_2.

ALMA_2 with $\eta = \sqrt{2}$			
α	$10^4\gamma'_{\text{d}}$	eps	upds
0.9	4.827	590,503	6,983,057
0.8	6.948	1,369,844	17,109,556
0.7	8.117	2,661,057	36,675,901
0.6	8.921	4,754,748	73,573,111
0.5	9.514	8,741,022	146,643,468
0.4	9.888	17,352,052	302,755,125
0.3	10.189	38,633,099	689,442,423
0.2	10.435	105,851,542	1,932,030,238

We also analyse the full sonar dataset (208 instances, 60 attributes). If the choice $\rho = 1$ is made $R \simeq 4.05347$ and $\gamma_{\text{d}} \simeq 0.00108$. Our experimental results for ALMA_2, the Perceptron, $\text{CRAMMA}^{0.5}$, agg- ROMMA and $\text{MICRA}^{0.05,0.9}$ are presented in Tables 7.9, 7.10 and 7.11. We see that ALMA_2 is again the slowest by far. Also, $\text{MICRA}^{0.05,0.9}$ is now the fastest in every respect. It is also surprising that in the full sonar dataset, unlike the case of the reduced one, the Perceptron performs better than CRAMMA for larger values of the margin and seems now able to obtain margins arbitrarily close to the maximum one. Nevertheless, $\text{CRAMMA}^{0.5}$ encounters no difficulty in approaching γ_{d} since the effective learning rate scales with β according to Theorem 5.2 (with $\delta = 1$). It is worth noticing the relatively poor performance of agg-ROMMA which becomes faster than the Perceptron and CRAMMA only in the vicinity of the maximum margin γ_{d}.

In Table 7.12 we present the experimental results on the full sonar dataset for the Incremental, the Perceptron+boosting and the Bolzano+boosting algorithms. In implementing the incremental scenario we first attempted successfully to find a solution possessing a margin value of 0.00001 and subsequently starting from a value of 0.0001245 we proceeded in steps of 0.0001. The choice 0.0001245 for the starting value was made on purpose in order to obtain results facilitating comparison with the results obtained

TABLE 7.10: Experimental results for the full sonar dataset. The directional margin $\gamma'_{\rm d}$, the number of epochs (eps) and updates (upds) are given for the Perceptron and CRAMMA$^{0.5}$ with $\eta_{\rm eff} = 0.0009(\frac{\beta}{R})^{-1}$.

Perceptron				CRAMMA$^{0.5}$			
$\frac{b}{\eta R^2}$	$10^4\gamma'_{\rm d}$	eps	upds	$\frac{\beta}{R}$	$10^4\gamma'_{\rm d}$	eps	upds
0.1	3.013	317,236	3,768,696	0.15	3.566	252,880	2,913,907
0.21	4.782	373,765	4,959,623	0.212	4.952	231,481	3,299,900
0.25	5.054	403,845	5,427,076	0.25	5.271	252,482	3,733,395
0.35	6.014	447,728	6,523,043	0.33	6.111	312,643	4,803,955
0.565	7.185	586,319	8,979,027	0.46	7.208	425,100	6,830,466
0.856	8.096	762,101	12,424,800	0.64	8.100	624,641	10,270,636
1.33	8.841	1,098,515	18,192,933	0.85	8.858	941,508	15,303,304
1.6	9.084	1,272,066	21,453,825	1	9.143	1,203,891	19,669,949
2.45	9.573	1,796,255	31,840,232	1.3	9.607	1,829,201	30,251,396
6.5	10.251	4,304,174	79,697,293	2.4	10.250	5,431,100	90,383,282
16	10.526	10,241,501	191,253,084	3.8	10.502	12,735,868	215,412,511
100	10.692	62,640,551	1,173,868,363	10	10.687	81,646,924	1,438,684,876

TABLE 7.11: Experimental results for the full sonar dataset. The directional margin $\gamma'_{\rm d}$, the number of epochs (eps) and updates (upds) are given for the agg-ROMMA and MICRA$^{0.05,0.9}$ algorithms.

agg-ROMMA				MICRA$^{0.05,0.9}$ with $\eta = 150$			
δ	$10^4\gamma'_{\rm d}$	eps	upds	$10^4\frac{\beta}{R}$	$10^4\gamma'_{\rm d}$	eps	upds
0.9	1.811	180,303	2,417,577	1	2.106	112,702	1,331,940
0.8	3.414	228,621	3,286,905	1.8	3.573	142,422	1,662,302
0.7	4.886	290,748	4,522,890	2.6	5.134	181,877	2,144,917
0.6	6.095	372,599	6,409,592	3.2	6.212	227,749	2,719,002
0.5	7.177	487,850	9,264,967	3.8	7.242	292,363	3,593,417
0.4	8.053	732,700	13,651,702	4.3	8.123	371,230	4,675,575
0.3	8.840	1,132,067	21,875,812	4.76	8.870	485,894	6,227,397
0.2	9.558	1,827,925	38,971,344	5.3	9.610	796,517	9,939,642
0.1	10.224	3,514,909	92,847,921	5.92	10.253	1,848,445	24,475,813
0.05	10.527	6,572,401	193,211,437	6.31	10.530	3,805,949	51,288,513
0.01	10.745	37,759,162	969,757,899	7.1	10.748	26,133,573	360,112,068

using the other algorithms. In the Perceptron+boosting scenario the relevant for the Perceptron stage parameter $b/(\eta R^2)$ was set to the value 0.1. Also, the step of the boosting scenario was chosen as $0.2\beta_{\rm p}$. The parameters λ controlling the effective learning rates of the two boosting stages were set again to the values 0.1 and 0.03, respectively, whereas the maximum number of epochs in each step during the boosting stages was set to 300,000. In the Bolzano stage of the Bolzano+boosting scenario we set the maximum number of epochs in each trial to 30,000 and the parameter λ controlling $\eta_{\rm eff}$ to the value 1. The step of the boosting scenario was set to $0.005\beta_{\rm b}$, λ was given the value 0.25 and the maximum number of epochs in each boosting step was set to 10,000.

Comparing the results of Table 7.12 with the ones of Tables 7.9, 7.10 and 7.11 we observe once more that the algorithmic implementations with fixed margin condition, with the exception of the Bolzano+boosting scenario, perform impressively well for values of the margin up to 90-95% of the maximum. More specifically, the Perceptron+boosting scenario is approximately 2 times faster than MICRA for values of the margin close to 95% of the maximum. It is worth pointing out that the boosting stage of the Perceptron+boosting algorithm is extremely efficient requiring only a relatively low number of epochs or updates in order to upgrade the solutions with low margin values provided by the Perceptron stage.

TABLE 7.12: Experimental results for the full sonar dataset. The directional margin γ'_{d}, the number of epochs (eps) and updates (upds) are given for the Incremental, the Perceptron+boosting and the Bolzano+boosting algorithms.

Incremental			Perceptron+boosting			Bolzano+boosting		
$10^4\gamma'_{\mathrm{d}}$	eps	upds	$10^4\gamma'_{\mathrm{d}}$	eps	upds	$10^4\gamma'_{\mathrm{d}}$	eps	upds
0.114	42,253	1,685,953	4.222	317,402	3,769,039	3.596	408,312	8,894,285
1.248	44,496	1,743,753	4.852	317,825	3,770,643	3.700	408,348	8,894,364
2.246	50,450	1,920,776	5.461	318,264	3,772,358	3.905	408,474	8,894,648
3.250	62,365	2,298,515	6.041	318,675	3,774,000	4.021	408,609	8,895,017
4.245	85,500	3,116,392	6.636	319,163	3,776,124	4.204	410,252	8,900,020
5.248	116,793	4,266,063	7.236	320,227	3,781,114	4.401	424,310	8,940,481
6.245	156,377	5,808,864	7.842	322,895	3,797,249	4.502	439,443	8,976,933
7.246	205,259	7,781,493	8.445	328,812	3,842,018	4.615	461,978	9,035,328
8.246	269,331	10,544,159	9.049	357,818	4,071,167	4.714	483,663	9,089,719
9.247	402,128	16,613,580	9.657	499,316	5,388,230	4.805	509,931	9,153,537
10.245	1,029,414	47,619,950	10.246	1,053,977	12,081,623	4.858	524,286	9,189,881

7.1.1.2 The Artificial Dataset LS-10

TABLE 7.13: Experimental results for the artificial dataset LS-10. The directional margin γ'_{d}, the number of epochs (eps) and updates (upds) are given for ALMA$_2$.

ALMA$_2$ with $\eta = \sqrt{2}$			
α	$10^3\gamma'_{\mathrm{d}}$	eps	upds
0.9	1.560	89,633	316,096
0.8	2.168	227,394	839,473
0.7	2.393	509,266	1,946,876
0.6	2.553	1,030,497	4,180,679
0.5	2.721	1,775,427	8,356,361
0.4	2.761	3,582,989	17,853,849
0.3	2.816	7,954,304	41,564,517
0.2	2.855	22,124,883	118,767,267
0.1	2.885	108,209,065	588,576,481

The binary artificial dataset known as LS-10 has instances with 10 attributes the values of which are produced according to a uniform distribution in the interval [0,1]. The

attributes x_i of the instances belonging to the first class satisfy the inequality $x_1 + \cdots + x_5 < x_6 + \cdots + x_{10}$ with the attributes of the instances of the other satisfying the inverse inequality. To perform our experiments we produced a LS-10 dataset with 1000 instances equally divided into two classes. In all our experiments with this set the data are embedded in the augmented space at a distance $\rho = 1$ from the origin in the additional dimension. For the specific dataset produced according to the above procedure the choice $\rho = 1$ leads to $R \simeq 2.749$ and $\gamma_{\rm d} \simeq 0.00291$.

TABLE 7.14: Experimental results for the artificial dataset LS-10. The directional margin $\gamma'_{\rm d}$, the number of epochs (eps) and updates (upds) are given for the Perceptron and CRAMMA[1] with $\eta_{\rm eff} = 0.015(\frac{\beta}{R})^{-0.95}$.

Perceptron				CRAMMA[1]			
$\frac{b}{\eta R^2}$	$10^3\gamma'_{\rm d}$	eps	upds	$\frac{\beta}{R}$	$10^3\gamma'_{\rm d}$	eps	upds
0.15	1.356	53,375	182,079	6	1.408	2,312	12,092
0.3	1.787	70,914	266,191	8	1.831	1,862	12,381
0.43	2.052	92,782	351,609	10	2.124	1,807	13,683
0.8	2.335	148,389	583,895	16	2.370	1,836	18,776
1.25	2.530	199,433	841,741	24	2.547	2,298	26,484
1.62	2.648	216,945	1,025,056	39.5	2.661	3,129	41,397
1.89	2.698	249,217	1,206,358	42	2.698	3,202	43,633
2.5	2.754	308,453	1,572,668	75	2.789	4,740	74,211
5	2.813	539,763	3,000,567	130	2.817	8,419	127,015
11	2.866	1,126,815	6,463,168	270	2.869	17,862	259,291
30	2.892	3,000,437	17,495,732	700	2.892	50,141	665,720
55	2.901	5,457,872	32,001,008	1600	2.901	124,393	1,516,142

TABLE 7.15: Experimental results for the artificial dataset LS-10. The directional margin $\gamma'_{\rm d}$, the number of epochs (eps) and updates (upds) are given for the algorithms agg-ROMMA and MICRA$^{0.05,0.9}$.

agg-ROMMA				MICRA$^{0.05,0.9}$ with $\eta = 1.1$			
δ	$10^3\gamma'_{\rm d}$	eps	upds	$10^3\frac{\beta}{R}$	$10^3\gamma'_{\rm d}$	eps	upds
0.8	1.402	24,653	92,923	0.72	1.554	119	2,783
0.7	1.808	33,467	145,912	0.95	1.877	122	2,835
0.6	2.102	44,397	218,501	1.16	2.142	249	3,307
0.5	2.360	62,347	330,751	1.3	2.374	519	4,127
0.4	2.532	103,951	543,610	1.43	2.573	1,089	5,928
0.3	2.632	188,071	1,001,102	1.52	2.663	1,747	8,303
0.2	2.691	372,954	2,273,537	1.55	2.700	2,176	9,591
0.1	2.804	792,503	5,230,325	1.68	2.828	4,891	19,077
0.07	2.838	1,144,998	7,618,917	1.719	2.852	6,204	24,387
0.05	2.866	1,610,371	10,658,391	1.77	2.874	10,988	40,238
0.02	2.892	4,057,295	26,332,267	1.867	2.892	27,260	97,522
0.01	2.901	8,169,278	52,805,473	1.949	2.901	60,240	214,054

Our experimental results for ALMA_2, the Perceptron, CRAMMA^1, agg- ROMMA and $\text{MICRA}^{0.05,0.9}$ are presented in Tables 7.13, 7.14 and 7.15. We see that ALMA_2 is again the slowest by far. $\text{MICRA}^{0.05,0.9}$ is again the fastest by far in every respect followed by CRAMMA^1. The Perceptron and agg-ROMMA are much slower than MICRA and CRAMMA with the Perceptron being faster than agg-ROMMA for larger margins and able to approach the maximum margin $\gamma_{\text{d}} \simeq 0.00291$. Since we chose the value $\epsilon = 1$ for CRAMMA a value $\delta = 0.95 < 1$ had to be chosen according to Theorem 5.2.

In Table 7.16 we present the experimental results on the LS-10 dataset for the Incremental, the Perceptron+boosting and the Bolzano+boosting algorithms. In implementing the incremental scenario we first attempted successfully to find a solution possessing a margin value of 0.0002 and subsequently we proceeded in steps of 0.0003. In the Perceptron+boosting scenario the relevant for the Perceptron stage parameter $b/(\eta R^2)$ was set to the value 0.01. Also, the step of the boosting scenario was chosen as $0.1\beta_{\text{p}}$. The parameters λ controlling the effective learning rates of the two boosting stages were set again to the values 0.1 and 0.03, respectively, whereas the maximum number of epochs in each step during the boosting stages was set to 3,000. In the Bolzano stage of the Bolzano+boosting scenario we set the maximum number of epochs in each trial to only 50 and the parameter λ controlling η_{eff} to the value 0.5. The step of the boosting scenario was set to $0.005\beta_{\text{b}}$, λ was given the value 0.025 and the maximum number of epochs in each boosting step was set to 5,000. From Table 7.16 we see that the fastest algorithm in all respects is the Incremental followed by the Bolzano+boosting and the Perceptron+boosting. If we take into account the results of Tables 7.13, 7.14 and 7.15 we may say that $\text{MICRA}^{0.05,0.9}$ is more or less comparable to the last two algorithms in the above classification.

TABLE 7.16: Experimental results for the artificial dataset LS-10. The directional margin γ'_{d}, the number of epochs (eps) and updates (upds) are given for the Incremental, the Perceptron+boosting and the Bolzano+boosting algorithms.

Incremental			Perceptron+boosting			Bolzano+boosting		
$10^3\gamma'_{\text{d}}$	eps	upds	$10^3\gamma'_{\text{d}}$	eps	upds	$10^3\gamma'_{\text{d}}$	eps	upds
1.114	115	874	2.404	4,450	14,327	2.693	1,270	11,218
1.411	152	1,066	2.478	4,458	14,351	2.762	1,869	12,659
1.703	195	1,316	2.556	4,470	14,391	2.846	4,592	19,081
2.009	247	1,615	2.629	4,480	14,421	2.856	8,467	30,473
2.308	308	1,963	2.714	4,494	14,459	2.871	12,177	41,782
2.600	473	3,269	2.780	4,544	14,582	2.880	16,582	54,970
2.901	4,297	32,273	2.850	7,564	25,263	2.891	20,337	66,377

7.1.1.3 The Dataset WBC_{-11}

The linearly separable dataset WBC_{-11} consists of 672 instances each with 9 attributes. We constructed it from the Wisconsin Breast Cancer (WBC) dataset obtainable from

TABLE 7.17: Experimental results for the WBC_{-11} dataset. The directional margin γ'_{d}, the number of epochs (eps) and updates (upds) are given for ALMA_2.

ALMA_2 with $\eta = \sqrt{2}$			
α	$10^2\gamma'_{\text{d}}$	eps	upds
0.9	1.324	259,019	973,702
0.8	1.783	648,397	2,704,552
0.7	2.008	1,446,449	6,254,522
0.6	2.141	3,003,292	13,320,424
0.5	2.228	6,126,330	27,666,245
0.4	2.290	12,869,852	58,927,860
0.3	2.336	29,574,927	136,925,778
0.2	2.373	83,529,043	390,186,724
0.1	2.402	409,746,613	1,928,029,375

TABLE 7.18: Experimental results for the WBC_{-11} dataset. The directional margin γ'_{d}, the number of epochs (eps) and updates (upds) are given for the Perceptron and $\text{CRAMMA}^{0.5}$ with $\eta_{\text{eff}} = 0.00006(\frac{\beta}{R})^{-1}$.

Perceptron				$\text{CRAMMA}^{0.5}$			
$\frac{b}{\eta R^2}$	$10^2\gamma'_{\text{d}}$	eps	upds	$\frac{\beta}{R}$	$10^2\gamma'_{\text{d}}$	eps	upds
0.1	0.861	151,723	501,541	0.021	0.899	1,998	10,174
0.17	1.162	188,455	686,703	0.038	1.213	3,566	18,066
0.26	1.413	238,097	946,966	0.056	1.430	5,314	26,836
0.4	1.650	330,060	1,401,984	0.195	1.659	49,750	239,778
0.65	1.880	464,581	2,058,524	0.329	1.885	106,224	524,046
0.97	2.033	623,833	2,894,811	0.455	2.033	172,418	859,626
1.8	2.197	1,059,020	4,980,423	0.64	2.201	292,732	1,458,022
4.1	2.321	2,241,988	10,761,773	0.81	2.322	418,691	2,088,673
8.5	2.374	4,499,804	21,798,933	1.05	2.374	667,983	3,356,490
45	2.415	23,240,723	113,406,210	4.7	2.415	14,220,901	64,987,024

the UCI repository by first omitting the 16 instances with missing attributes and subsequently removing from the dataset containing the remaining 683 instances the 11 instances having the positions 2, 4, 191, 217, 227, 245, 252, 286, 307, 420 and 475. In our experiments we chose the value $\rho = 30$ for the parameter ρ of the augmented space which led to $R = \sqrt{1716}$ and $\gamma_{\text{d}} \simeq 0.0243$.

Our experimental results for ALMA_2, the Perceptron, $\text{CRAMMA}^{0.5}$, agg- ROMMA and $\text{MICRA}^{0.1,0.8}$ are presented in Tables 7.17, 7.18 and 7.19. We see that ALMA_2 is again the slowest by far. The superiority of the performance of $\text{MICRA}^{0.1,0.8}$ on this dataset is remarkable. $\text{CRAMMA}^{0.5}$ performs also very well taking easily the second position among the above mentioned algorithms. The Perceptron and agg-ROMMA are much slower than MICRA and CRAMMA with the Perceptron being faster than agg-ROMMA for all values of the margin and able to approach $\gamma_{\text{d}} \simeq 0.0243$.

TABLE 7.19: Experimental results for the WBC_{-11} dataset. The directional margin γ'_{d}, the number of epochs (eps) and updates (upds) are given for the algorithms agg-ROMMA and $\text{MICRA}^{0.1,0.8}$.

agg-ROMMA				$\text{MICRA}^{0.1,0.8}$ with $\eta = 2.3$			
δ	$10^2\gamma'_{\text{d}}$	eps	upds	$10^3\frac{\beta}{R}$	$10^2\gamma'_{\text{d}}$	eps	upds
0.8	0.839	154,575	552,707	0.5	0.937	872	4,087
0.7	1.122	186,442	827,561	0.7	1.348	896	4,373
0.6	1.403	223,103	1,153,779	0.82	1.477	978	4,949
0.5	1.642	280,170	1,630,857	0.95	1.682	1,042	5,335
0.4	1.847	374,253	2,382,480	1.45	1.897	24,945	106,884
0.3	2.030	547,813	3,541,471	1.64	2.047	36,277	161,918
0.2	2.195	868,845	5,784,868	1.86	2.203	59,002	276,094
0.1	2.318	2,071,529	13,931,792	2.07	2.324	96,899	467,369
0.05	2.373	4,590,007	31,156,487	2.22	2.376	150,039	755,815
0.03	2.394	8,036,424	54,749,006	2.39	2.400	303,195	1,429,303
0.01	2.415	25,361,230	174,388,827	2.7	2.415	1,037,611	4,533,155

TABLE 7.20: Experimental results for the WBC_{-11} dataset. The directional margin γ'_{d}, the number of epochs (eps) and updates (upds) are given for the Incremental, the Perceptron+boosting and the Bolzano+boosting algorithms.

Incremental			Perceptron+boosting			Bolzano+boosting		
$10^2\gamma'_{\text{d}}$	eps	upds	$10^2\gamma'_{\text{d}}$	eps	upds	$10^2\gamma'_{\text{d}}$	eps	upds
0.626	206	1,205	1.497	264,516	1,076,473	1.509	9,042	637,047
0.908	238	1,328	1.647	264,951	1,077,845	1.643	9,428	637,880
1.203	437	2,075	1.797	266,206	1,081,684	1.790	10,303	639,854
1.502	7,816	57,388	1.947	343,809	1,425,412	1.972	11,490	642,942
1.800	18,888	148,344	2.096	438,366	1,852,761	2.082	12,548	646,294
2.101	40,091	332,117	2.246	533,354	2,286,890	2.228	96,599	972,145
2.400	233,306	2,200,969	2.396	681,987	2,950,450	2.301	139,606	1,146,128

In Table 7.20 we present the experimental results on the WBC_{-11} dataset for the Incremental, the Perceptron+boosting and the Bolzano+boosting algorithms. In implementing the incremental scenario we first attempted successfully to find a solution possessing a margin value of 0.001 and subsequently starting from the value 0.003 we proceeded in steps of 0.003. In the Perceptron+boosting scenario the relevant for the Perceptron stage parameter $b/(\eta R^2)$ was set to the value 0.3. Also, the step of the boosting scenario was chosen as $0.1\beta_{\text{p}}$. The parameters λ controlling the effective learning rates of the two boosting stages were set again to the values 0.1 and 0.03, respectively, whereas the maximum number of epochs in each step during the boosting stages was set to 100,000. In the Bolzano stage of the Bolzano+boosting scenario we set the maximum number of epochs in each trial to only 700 and the parameter λ controlling η_{eff} to the value 0.5. The step of the boosting scenario was set to $0.025\beta_{\text{b}}$, λ was given the value 0.1 and the maximum number of epochs in each boosting step was set to 30,000. From Table 7.20 we see that the fastest scenario involving the fixed margin condition algorithms is the Incremental followed by the Bolzano+boosting. If we take into account the results of

Tables 7.17, 7.18 and 7.19 we may say that only MICRA$^{0.05,0.9}$ is faster than the fixed margin condition scenarios.

7.1.2 Inseparable Data

For linearly inseparable datasets we consider an instance space extended by as many dimensions as the instances where each instance is placed at a distance $|\Delta|$ from the origin in the corresponding dimension. Then, relying on the analysis of Section 6.2, we seek separation in this extended space with large margin.

An important property of the extended space is the existence of a lower bound on the maximum directional margin $\Gamma_{\Delta\text{opt}}$ in that space given by (6.13) which depends only on the parameter Δ of the extended space and the number l of patterns. The existence of such a lower bound is very helpful because it allows us to determine acceptable values of η_{eff} for CRAMMA$^{\epsilon}$. Indeed, if we set

$$\eta_{\text{eff}} = \eta_0 \left(\frac{\beta}{R}\right)^{-\delta} \quad \text{with} \quad \eta_0 < \frac{1}{2}\left(\sqrt{1+8\frac{|\Delta|}{R\sqrt{l}}} - 1\right)$$

and $0 < \epsilon\delta < 1$ the conditions of Theorem 5.2 are automatically satisfied for $\beta > R$.

We also take advantage of another property of the extended space in order to attempt an assessment of the relative deviation of the margin Γ_Δ found from the (unknown) maximum $\Gamma_{\Delta\text{opt}}$: the quantities D and D' defined in Section 6.2 for which $D' \geq D$ holds become equal, according to Remark 6.3, if the optimal extended solution vector is found. Thus, we may take the relative deviation

$$\frac{\delta D}{D} \equiv \frac{D' - D}{D}$$

as a measure of the departure from optimality. In order to test this idea we will compare in our experiments $\delta D/D$ with the relative deviation

$$\frac{\delta\Gamma}{\Gamma} \equiv \frac{\Gamma_{\Delta\text{opt}} - \Gamma_\Delta}{\Gamma_{\Delta\text{opt}}}$$

of Γ_Δ from $\Gamma_{\Delta\text{opt}}$.

7.1.2.1 The WBC Dataset

The linearly inseparable Wisconsin Breast Cancer (WBC) dataset, obtainable from the UCI repository, comprises 683 instances each with 9 attributes after ignoring the 16 instances with missing attributes. We embed the data in the augmented space at a distance $\rho = 10$ from the origin in the additional dimension and we construct the extended

instance space with a parameter $\Delta = 1$. This leads to $R = \sqrt{917}$ and to a maximum margin $\Gamma_{\Delta\text{opt}} \simeq 0.13033$ with respect to zero-threshold hyperplanes in the extended (and augmented) space.

TABLE 7.21: Experimental results for the WBC dataset (extended with $\Delta = 1$). The relative deviations $\frac{\delta D}{D}$ and $\frac{\delta \Gamma}{\Gamma}$, the margin Γ_Δ and the number of epochs (eps) and updates (upds) are given for ALMA_2.

ALMA_2 with $\eta = \sqrt{2}$					
α	$10\frac{\delta D}{D}$	$10\frac{\delta \Gamma}{\Gamma}$	$10^3\Gamma_\Delta$	eps	upds
0.9	3.86	4.05	77.52	1,520	16,533
0.8	2.72	2.58	96.73	4,039	50,778
0.7	1.82	1.77	107.23	7,230	118,230
0.6	1.13	1.16	115.24	11,666	248,460
0.5	0.81	0.81	119.76	20,147	512,248
0.4	0.53	0.53	123.36	36,614	1,084,841
0.3	0.33	0.33	126.01	77,085	2,518,152
0.2	0.19	0.19	127.84	203,016	7,184,571
0.1	0.08	0.08	129.33	945,965	35,542,411

TABLE 7.22: Experimental results for the WBC dataset (extended with $\Delta = 1$). The relative deviations $\frac{\delta D}{D}$ and $\frac{\delta \Gamma}{\Gamma}$, the margin Γ_Δ and the number of epochs (eps) and updates (upds) are given for the Perceptron and $\text{CRAMMA}^{0.5}$ with $\eta_{\text{eff}} = \frac{1.7}{R\sqrt{683}}\left(\frac{\beta}{R}\right)^{-1}$.

Perceptron						$\text{CRAMMA}^{0.5}$					
$\frac{b}{\eta R^2}$	$10\frac{\delta D}{D}$	$10\frac{\delta \Gamma}{\Gamma}$	$10^3\Gamma_\Delta$	eps	upds	$\frac{\beta}{R}$	$10\frac{\delta D}{D}$	$10\frac{\delta \Gamma}{\Gamma}$	$10^3\Gamma_\Delta$	eps	upds
0.22	4.20	4.49	71.82	1,553	20,358	0.313	5.46	4.47	72.10	1,001	17,958
0.34	3.59	3.60	83.38	1,926	27,705	0.46	4.24	3.62	83.09	1,637	28,786
0.64	2.70	2.73	94.74	3,049	46,592	0.682	3.01	2.75	94.47	2,871	49,358
1.48	1.76	1.70	108.15	4,824	95,244	1.19	1.77	1.71	108.01	5,717	111,833
3.5	0.90	0.87	119.05	8,300	206,468	1.98	0.85	0.87	118.93	11,114	254,331
4.95	0.64	0.63	122.15	11,563	285,567	2.3	0.62	0.65	121.91	13,993	328,997
8.1	0.45	0.44	124.62	18,730	457,333	2.8	0.42	0.44	124.58	18,648	464,755
70	0.18	0.18	128.04	160,306	3,843,624	5.18	0.17	0.17	128.07	55,548	1,502,210
700	0.15	0.15	128.37	1,599,408	38,336,600	11.11	0.08	0.08	129.27	208,338	6,773,596

Our experimental results for ALMA_2, the Perceptron, $\text{CRAMMA}^{0.5}$, agg- ROMMA and $\text{MICRA}^{0.05,0.9}$ are presented in Tables 7.21, 7.22 and 7.23. We see that ALMA_2 is the slowest for large values of the margin. $\text{MICRA}^{0.05,0.9}$ is again the fastest by far with respect to the number of updates with agg-ROMMA needing fewer epochs in order to converge to large margin hyperplanes. Moreover, the Perceptron is apparently unable to approach the maximum margin arbitrarily close. $\text{CRAMMA}^{0.5}$, instead, is able to approach $\Gamma_{\Delta\text{opt}} \simeq 0.13033$ as close as one wishes since the choice $\eta_{\text{eff}} = \frac{1.7}{R\sqrt{683}}\left(\frac{\beta}{R}\right)^{-1}$ satisfies the conditions of Theorem 5.2. It is worth noticing that for all algorithms the quantity $\delta D/D$ proves a surprisingly accurate measure of the relative deviation $\delta\Gamma/\Gamma$ of Γ_Δ from $\Gamma_{\Delta\text{opt}}$, especially if the margins attained are relatively large.

TABLE 7.23: Experimental results for the WBC dataset (extended with $\Delta = 1$). The relative deviations $\frac{\delta D}{D}$ and $\frac{\delta \Gamma}{\Gamma}$, the margin Γ_Δ and the number of epochs (eps) and updates (upds) are given for the agg-ROMMA and MICRA$^{0.05,0.9}$.

agg-ROMMA						MICRA$^{0.05,0.9}$ with $\eta = 20$					
δ	$10\frac{\delta D}{D}$	$10\frac{\delta \Gamma}{\Gamma}$	$10^3\Gamma_\Delta$	eps	upds	$10^3\frac{\beta}{R}$	$10\frac{\delta D}{D}$	$10\frac{\delta \Gamma}{\Gamma}$	$10^3\Gamma_\Delta$	eps	upds
0.5	4.65	4.43	72.59	1,734	13,983	3.8	3.79	4.34	73.77	1,278	7,810
0.4	3.59	3.53	84.36	2,439	20,822	4.5	3.28	3.45	85.40	1,878	11,583
0.3	2.48	2.64	95.91	3,138	33,728	5.2	2.19	2.60	96.48	2,586	18,792
0.2	1.55	1.69	108.36	4,964	62,751	6.1	1.39	1.66	108.69	4,245	40,824
0.1	0.76	0.86	119.16	9,238	169,588	7.02	0.68	0.83	119.57	6,632	105,964
0.07	0.54	0.60	122.48	12,181	273,864	7.35	0.52	0.59	122.59	9,298	151,695
0.05	0.40	0.43	124.68	14,860	409,956	7.54	0.37	0.43	124.70	12,104	183,643
0.02	0.17	0.17	128.08	21,777	976,028	7.99	0.14	0.17	128.09	23,946	334,565
0.01	0.08	0.08	129.28	28,205	1,554,492	8.4	0.05	0.06	129.49	48,106	734,629

In Table 7.24 we present the experimental results on the WBC dataset for the Incremental, the Perceptron+boosting and the Bolzano+boosting algorithms. In implementing the incremental scenario we first attempted successfully to find a solution possessing a margin value of 0.038 and we subsequently proceeded in steps of 0.03. In the Perceptron+boosting scenario the relevant for the Perceptron stage parameter $b/(\eta R^2)$ was set to 1. Also, the step of the boosting scenario was chosen as $0.05\beta_{\rm p}$. The parameters λ controlling the effective learning rates of the two boosting stages were set to the values 0.09 and 0.03, respectively, whereas the maximum number of epochs in each step during the boosting stages was set to 5,000. In the Bolzano stage of the Bolzano+boosting scenario we set the maximum number of epochs in each trial to only 200 and the parameter λ controlling $\eta_{\rm eff}$ to the value 0.5. The step of the boosting scenario was set to $0.1\beta_{\rm b}$, λ was given the value 0.1 and the maximum number of epochs in each boosting step was set to 10,000. From Table 7.24 we see that the fastest scenario involving the fixed margin condition algorithms is the Perceptron+boosting followed by the Incremental. Their performance seems remarkable even if we take into account the results of Tables 7.21, 7.22 and 7.23. Actually, the Perceptron+boosting scenario competes closely with MICRA$^{0.05,0.9}$ for margins up to 98% of the maximum.

TABLE 7.24: Experimental results for the WBC dataset (extended with $\Delta = 1$). The margin Γ_Δ, the number of epochs (eps) and updates (upds) are given for the Incremental, the Perceptron+boosting and the Bolzano+boosting algorithms.

Incremental			Perceptron+boosting			Bolzano+boosting		
$10^3\Gamma_\Delta$	eps	upds	$10^3\Gamma_\Delta$	eps	upds	$10^3\Gamma_\Delta$	eps	upds
38.02	1,150	17,946	107.72	4,474	69,929	98.08	11,367	191,687
68.05	3,201	66,323	117.83	7,349	92,461	110.41	15,949	238,592
98.01	6,325	164,701	122.94	11,162	135,456	116.47	19,379	274,348
128.00	21,557	1,040,212	128.07	21,060	331,914	122.62	24,475	337,940

7.1.2.2 The Votes Dataset

TABLE 7.25: Experimental results for the votes dataset (extended with $\Delta = 1$). The relative deviations $\frac{\delta D}{D}$ and $\frac{\delta \Gamma}{\Gamma}$, the margin Γ_Δ and the number of epochs (eps) and updates (upds) are given for ALMA_2.

ALMA_2 with $\eta = \sqrt{2}$					
α	$10\frac{\delta D}{D}$	$10\frac{\delta \Gamma}{\Gamma}$	$10^3\Gamma_\Delta$	eps	upds
0.9	4.57	4.53	91.85	49	508
0.8	4.37	4.37	94.53	70	1,039
0.7	2.90	2.82	120.68	115	1,975
0.6	1.94	2.00	134.47	175	3,655
0.5	1.61	1.52	142.52	331	7,239
0.4	1.12	1.06	150.22	588	14,446
0.3	0.74	0.68	156.54	1,196	32,101
0.2	0.40	0.38	161.69	3,113	88,595
0.1	0.16	0.15	165.39	14,487	428,568

TABLE 7.26: Experimental results for the votes dataset (extended with $\Delta = 1$). The relative deviations $\frac{\delta D}{D}$ and $\frac{\delta \Gamma}{\Gamma}$, the margin Γ_Δ and the number of epochs (eps) and updates (upds) are given for the Perceptron and $\text{CRAMMA}^{0.5}$ with $\eta_{\text{eff}} = \frac{1.7}{R\sqrt{435}} \left(\frac{\beta}{R}\right)^{-1.3}$.

Perceptron						$\text{CRAMMA}^{0.5}$					
$\frac{b}{\eta R^2}$	$10\frac{\delta D}{D}$	$10\frac{\delta \Gamma}{\Gamma}$	$10^3\Gamma_\Delta$	eps	upds	$\frac{\beta}{R}$	$10\frac{\delta D}{D}$	$10\frac{\delta \Gamma}{\Gamma}$	$10^3\Gamma_\Delta$	eps	upds
0.73	3.59	3.60	107.53	45	735	0.622	2.57	3.59	107.61	64	767
0.85	3.06	3.32	112.15	47	808	0.661	2.60	3.34	111.91	51	772
0.95	2.24	2.62	123.98	49	869	1.1	2.15	2.70	123.15	95	1,458
2.2	1.52	1.63	140.54	82	1,644	1.8	1.18	1.66	140.14	174	2,972
6.65	0.74	0.74	155.53	214	4,502	3	0.54	0.77	154.99	365	6,828
15.9	0.35	0.36	161.99	491	10,350	4.9	0.27	0.36	161.89	803	16,510
50	0.17	0.18	164.89	1,521	31,937	6.95	0.15	0.18	164.94	1,473	32,102
300	0.10	0.11	166.12	9,068	190,142	10	0.10	0.10	166.24	2,811	65,160
10^5	0.09	0.10	166.29	3,019,034	63,282,759	15.9	0.05	0.05	167.19	6,558	162,803

The votes dataset, obtainable from the UCI repository, comprises 435 instances each with 16 attributes. This dataset consists of the votes of each of the U.S. Congressmen on 16 key issues. We attribute to a "yes" vote the value +1, to a "no" vote the value -1 and to an unspecified vote the value 0. We embed the data in the augmented space at a distance $\rho = 1$ from the origin in the additional dimension and we construct the extended instance space with a parameter $\Delta = 1$. This leads to $R = \sqrt{18}$ and a maximum margin $\Gamma_{\Delta\text{opt}} \simeq 0.16799$ with respect to zero-threshold hyperplanes in the extended (and augmented) space.

Our experimental results for ALMA_2, the Perceptron, $\text{CRAMMA}^{0.5}$, agg- ROMMA and $\text{MICRA}^{0.05,0.9}$ are presented in Tables 7.25, 7.26 and 7.27. We see that ALMA_2 is the slowest. $\text{MICRA}^{0.05,0.9}$ is again the fastest with respect to the number of updates and agg-ROMMA needs fewer epochs in order to converge to hyperplanes with large margin.

TABLE 7.27: Experimental results for the votes dataset (extended with $\Delta = 1$). The relative deviations $\frac{\delta D}{D}$ and $\frac{\delta\Gamma}{\Gamma}$, the margin Γ_Δ and the number of epochs (eps) and updates (upds) are given for the agg-ROMMA and MICRA$^{0.05,0.9}$.

agg-ROMMA						MICRA$^{0.05,0.9}$ with $\eta = 5$					
δ	$10\frac{\delta D}{D}$	$10\frac{\delta\Gamma}{\Gamma}$	$10^3\Gamma_\Delta$	eps	upds	$10^4\frac{\beta}{R}$	$10\frac{\delta D}{D}$	$10\frac{\delta\Gamma}{\Gamma}$	$10^3\Gamma_\Delta$	eps	upds
0.5	2.45	3.54	108.52	39	443	327	2.49	3.50	109.19	37	460
0.4	2.83	3.44	110.19	46	586	340	2.27	3.22	113.91	43	489
0.3	2.07	2.49	126.08	55	820	411	1.66	2.27	129.79	51	678
0.2	1.41	1.55	141.92	88	1,421	460	1.21	1.44	143.84	71	932
0.1	0.65	0.68	156.53	156	3,282	542	0.56	0.66	156.97	162	2,323
0.05	0.33	0.34	162.27	213	6,253	580	0.28	0.32	162.62	255	4,209
0.03	0.16	0.17	165.12	301	9,885	609	0.16	0.17	165.12	446	7,891
0.02	0.10	0.10	166.29	504	15,055	629	0.09	0.10	166.37	721	13,119
0.01	0.05	0.05	167.21	1,187	37,444	658.7	0.04	0.04	167.24	1,498	29,075

Moreover, the Perceptron, although faster than CRAMMA$^{0.5}$ away from the maximum margin, is apparently again unable to approach $\Gamma_{\Delta\text{opt}}$ arbitrarily close. CRAMMA$^{0.5}$, instead, is able to approach $\Gamma_{\Delta\text{opt}} \simeq 0.16799$ as close as one wishes since the choice $\eta_{\text{eff}} = \frac{1.7}{R\sqrt{435}}\left(\frac{\beta}{R}\right)^{-1.3}$ satisfies the conditions of Theorem 5.2. Once more the quantity $\delta D/D$ proves for all algorithms an accurate measure of the relative deviation $\delta\Gamma/\Gamma$ of Γ_Δ from $\Gamma_{\Delta\text{opt}}$.

TABLE 7.28: Experimental results for the votes dataset (extended with $\Delta = 1$). The margin Γ_Δ, the number of epochs (eps) and updates (upds) are given for the Incremental, the Perceptron+boosting and the Bolzano+boosting algorithms.

Incremental			Perceptron+boosting			Bolzano+boosting		
$10^3\Gamma_\Delta$	eps	upds	$10^3\Gamma_\Delta$	eps	upds	$10^3\Gamma_\Delta$	eps	upds
45.06	26	386	130.49	56	937	134.23	178	5,594
85.03	51	769	141.44	72	1,045	141.16	326	7,260
125.02	102	1,673	153.05	143	1,784	157.95	387	7,508
165.00	1,648	70,699	164.82	391	6,275	165.06	636	11,496

In Table 7.28 we present the experimental results on the votes dataset for the Incremental, the Perceptron+boosting and the Bolzano+boosting algorithms. In implementing the incremental scenario we first attempted successfully to find a solution possessing a margin value of 0.045 and we subsequently proceeded in steps of 0.04. In the Perceptron+boosting scenario the relevant for the Perceptron stage parameter $b/(\eta R^2)$ was set to 1. Also, the step of the boosting scenario was chosen as $0.1\beta_{\text{p}}$. The parameters λ controlling the effective learning rates of the two boosting stages were set again to the values 0.09 and 0.03, respectively, whereas the maximum number of epochs in each step during the boosting stages was set to 150. In the Bolzano stage of the Bolzano+boosting scenario we set the maximum number of epochs in each trial to only 50 and the parameter λ controlling η_{eff} to the value 0.5. The step of the boosting scenario was set to $0.05\beta_{\text{b}}$, λ was given the value 0.05 and the maximum number of epochs in each boosting step

was set to 250. From Table 7.28 we see that the fastest scenario involving the fixed margin condition algorithms is the Perceptron+boosting followed by the Bolzano+boosting. Their performance seems again remarkable even when the results of Tables 7.25, 7.26 and 7.27 are taken into account. Once more the Perceptron+boosting scenario competes closely with MICRA$^{0.05,0.9}$ for margins up to 98% of the maximum.

7.2 A "Reduction" Procedure for PLAs

A large proportion of the computer time required for the convergence of a PLA is devoted to checking the validity of the misclassification condition as the training patterns are presented sequentially to the algorithm in rounds (epochs). If one had a good way of guessing which patterns are more likely to be misclassified one could present them more often to the algorithm, thereby increasing the proportion of computer time devoted to updating the hypothesis. This motivates us to attempt to reduce the computational cost by forming a reduced "active set" of patterns consisting of the ones found misclassified during each epoch which are then cyclically presented to the algorithm for N mini-epochs unless no update occurs during a mini-epoch. Subsequently, a new full epoch involving all the patterns takes place giving rise to a new active set. The algorithm terminates only if no mistake occurs during a full epoch. This procedure clearly amounts to a different way of sequentially presenting the patterns to the algorithm and should not affect the applicability of convergence theorems. An algorithm incorporating the above procedure will be referred to as "reduced".

We now apply this "reduction" procedure to MICRA and attempt to assess its benefits. We compared experimentally MICRA and "reduced" MICRA (red-MICRA) on the reduced sonar, the WBC$_{-11}$ and WBC datasets with the results presented in Tables 7.29, 7.30 and 7.31. We observe that the number of full epochs is much lower for red-MICRA

TABLE 7.29: Experimental results for the reduced sonar dataset. The directional margin $\gamma'_{\rm d}$, the number of epochs (eps) and updates (upds) are given for MICRA$^{0.05,0.9}$ and the number of full epochs (f-eps), mini epochs (m-eps) and updates (upds) for red $-$ MICRA$^{0.05,0.9}$.

MICRA$^{0.05,0.9}$, $\eta = 50$				red $-$ MICRA$^{0.05,0.9}$, $\eta = 45$, $N = 80$				
$10^4 \frac{\beta}{R}$	$10^3\gamma'_{\rm d}$	eps	upds	$10^4 \frac{\beta}{R}$	$10^3\gamma'_{\rm d}$	f-eps	m-eps	upds
24	5.242	11,422	104,925	22.8	5.260	3,939	18,554	45,436
28	5.902	15,215	140,633	27	5.947	4,607	24,259	60,709
32	6.629	20,854	200,516	30.6	6.664	6,589	34,849	88,347
36	7.303	33,956	331,057	34.6	7.312	9,902	57,260	148,010
40.4	7.864	74,309	706,274	38.8	7.876	16,863	114,199	312,645
41.7	8.006	99,851	950,918	40.3	8.011	22,062	158,925	454,118
44.3	8.192	199,378	1,932,165	42.6	8.195	32,415	276,936	875,417
49.5	8.367	1,153,031	11,610,899	47.8	8.368	80,621	1,171,391	5,749,228

TABLE 7.30: Experimental results for the WBC_{-11} dataset. The directional margin γ'_{d}, the number of epochs (eps) and updates (upds) are given for $\mathrm{MICRA}^{0.1,0.8}$ and the number of full epochs (f-eps), mini epochs (m-eps) and updates (upds) for $\mathrm{red-MICRA}^{0.1,0.8}$.

$\mathrm{MICRA}^{0.1,0.8}$, $\eta = 2.3$				$\mathrm{red-MICRA}^{0.1,0.8}$, $\eta = 2.1$, $N = 80$				
$10^5 \frac{\beta}{R}$	$10^2\gamma'_{\mathrm{d}}$	eps	upds	$10^5 \frac{\beta}{R}$	$10^2\gamma'_{\mathrm{d}}$	f-eps	m-eps	upds
145	1.897	24,945	106,884	142	1.901	943	30,122	83,734
164	2.047	36,277	161,918	171	2.070	1,628	70,216	221,760
186	2.203	59,002	276,094	201	2.207	2,408	138,214	585,967
207	2.324	96,899	467,369	222	2.327	3,313	209,862	932,693
222	2.376	150,039	755,815	233	2.376	4,298	270,979	1,224,162
239	2.400	303,195	1,429,303	238	2.405	5,545	301,791	1,343,680
270	2.415	1,037,611	4,533,155	251	2.415	8,162	477,356	2,187,647

TABLE 7.31: Experimental results for the WBC dataset (extended with $\Delta = 1$). The margin Γ_Δ, the number of epochs (eps) and updates (upds) are given for $\mathrm{MICRA}^{0.05,0.9}$ and the number of full epochs (f-eps), mini epochs (m-eps) and updates (upds) for $\mathrm{red-MICRA}^{0.05,0.9}$.

$\mathrm{MICRA}^{0.05,0.9}$, $\eta = 20$				$\mathrm{red-MICRA}^{0.05,0.9}$, $\eta = 30$, $N = 20$				
$10^4 \frac{\beta}{R}$	$10^3\Gamma_\Delta$	eps	upds	$10^4 \frac{\beta}{R}$	$10^3\Gamma_\Delta$	f-eps	m-eps	upds
52	96.48	2,586	18,792	51	97.44	242	2,416	14,621
61	108.69	4,245	40,824	60	109.04	571	4,684	29,486
70.2	119.57	6,632	105,964	69	119.85	646	7,814	67,463
75.6	124.77	12,401	187,185	74.1	124.99	781	11,474	127,895
79.9	128.09	23,946	334,565	79.4	128.13	1,233	18,758	296,149
84	129.49	48,106	734,629	83.9	129.50	2,322	39,452	713,999

without a simultaneous blow up in the number of mini-epochs. This should certainly result in a serious reduction of the computational cost, as expected. What comes maybe as a surprise is that in many cases we have a serious reduction in the number of updates whereas less often we observe an increase. Overall, however, it seems that the reduction procedure is indeed computationally beneficial.

7.3 Comparison of MICRA with SVMs

A comparison of MICRA with SVMs, unlike PLAs, could only involve the CPU-time required to achieve a certain approximation of the hyperplane giving rise to the maximum geometric margin γ in the feature space where the patterns are linearly separable. PLAs like MICRA become extremely slow in the vicinity of the maximum directional margin γ_{d} which is attainable only asymptotically. Moreover, γ_{d} approaches γ only in the limit where the augmented space parameter $\rho \to \infty$. As a consequence, MICRA could converge faster than SVMs only to a solution hyperplane with geometric margin γ' slightly lower than γ. We choose to compare red-MICRA with SVMs at a margin value

larger than 99% of γ. Although it is straightforward to formulate MICRA (or red-MICRA) in dual space we will treat it here, unless otherwise specified, as a primal space algorithm. For linearly separable datasets our feature space will be the initial instance space whereas for linearly inseparable ones, unless otherwise specified, an instance space extended by as many dimensions as the instances will be considered where each instance is placed at a distance $|\Delta|$ from the origin in the corresponding dimension. This amounts to employing linear kernels and for inseparable data a soft margin approach involving the 2-norm of the slacks.

In our experiments SVMs are represented by algorithms based on decomposition methods which are many orders of magnitude faster than standard SVMs. More specifically, red-MICRA is compared with LIBSVM [11], an improved version of SMO, and SVM^{light}. For both algorithms we choose $m = 400$MB for the memory parameter and $C = 10^5$ (approximating $C = \infty$) for the 1-norm soft margin parameter since we are dealing with a hard margin problem in the appropriate feature space. Also, the working set size parameter q of SVM^{light} is fixed to the default value $q = 10$. For each dataset we obtain values of the geometric margin γ' corresponding to two different values of the accuracy parameter ϵ both for LIBSVM and SVM^{light}. The larger value of the margin obtained by these algorithms corresponds to $\epsilon = 0.001$ and is regarded as a good approximation to the maximum geometric margin γ. We require that the margin γ' achieved by red-MICRA be larger than 99% of the larger margins (corresponding to $\epsilon = 0.001$) and larger than the lower margins (corresponding to $\epsilon > 0.001$) obtained by both LIBSVM and SVM^{light}. We take advantage of the sparsity in the attributes of the initial space only if these attributes are binary. We also take into account the enormous sparsity present in the attributes associated with the additional dimensions of the extended instance space. The experiments were conducted on a 1.8 GHz Intel Pentium M processor with 504 MB RAM running Windows XP. The codes written in C++ were run using Microsoft's Visual C++ 5.0 compiler.

Table 7.32 contains the results of our comparative study of LIBSVM, SVM^{light} and red-MICRA on several UCI datasets with I/O excluded from the CPU-times reported. The value of the accuracy parameter ϵ corresponding to the lower value of the margin is set to $\epsilon = 0.03$ for LIBSVM and $\epsilon = 0.015$ for SVM^{light}. The sonar (meaning here the reduced sonar), the WBC and the votes datasets are described already. The ionosphere (iono) dataset consists of 351 instances each with 34 attributes. The tic-tac-toe (ttt) dataset consists of 958 instances each with 9 attributes taking values from the set $\{\mathrm{x}, \mathrm{o}, \mathrm{b}\}$ represented as $\{1, -1, 0\}$. The german (germ) dataset consists of 1000 instances each with 24 attributes. Finally, the linearly separable mushroom (mush) dataset consists of 5644 instances after removing the ones with missing attributes. Each instance has 22 categorical attributes replaced here by 125 binary ones out of which exactly 22 are true. We believe that from Table 7.32 it is fair to conclude that, roughly speaking, red-MICRA is of speed comparable to that of decomposition SVMs.

TABLE 7.32: Results of a comparative study of LIBSVM, SVM^{light} and red-MICRA on several UCI datasets.

data set	Δ	LIBSVM				SVM^{light}				red − MICRA$^{0.05,0.9}$					
		$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	ρ	η	N	$10^5\frac{\beta}{R}$	$10^2\gamma'$	Secs
sonar	0	0.8451	0.17	0.8405	0.10	0.8460	6.85	0.8388	4.84	1	45	80	462.2	0.8406	3.60 [1]
iono	1	10.554	0.06	10.389	0.05	10.551	0.30	10.448	0.19	$\frac{3}{2}$	10	10	2929	10.449	0.07
votes	1	16.846	0.02	16.708	0.02	16.841	0.18	16.690	0.11	1	5	20	6385	16.718	0.02
WBC	1	13.034	0.12	12.848	0.09	13.033	0.81	12.929	0.45	2	25	20	837.6	12.932	0.35
ttt	1	10.300	0.47	10.183	0.27	10.295	3.35	10.185	1.35	$\frac{1}{2}$	8	20	5334	10.203	0.05
germ	25	95.361	0.62	94.055	0.45	95.332	2.96	94.217	1.82	8	30	50	908.9	94.415	0.36
mush	0	36.551	0.58	35.988	0.33	36.538	0.17	36.103	0.11	0	4.5	50	12535	36.212	0.10

TABLE 7.33: Results of a comparative study of LIBSVM, SVM^{light} and red-MICRA on several UCI datasets with non-linear kernels.

data set	LIBSVM				SVM^{light}				red − MICRA$^{0.05,0.9}$					
	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	ρ	η	N	$10^5\frac{\beta}{R}$	$10^2\gamma'$	Secs
iono	96.581	0.03	94.795	0.02	96.569	0.48	94.573	0.33	50	350	400	146	95.758	0.12
votes	191.79	0.03	188.81	0.02	191.71	0.07	188.70	0.05	10	70	300	1034	190.00	0.04
WBC	19242	0.31	19013	0.20	19259	7.64	19138	5.64	4000	600	300	56.7	19141	1.31
ttt	415.35	0.42	409.65	0.16	415.19	1.10	408.60	0.33	14	6	30	6626	411.28	0.28
germ	3.4487	1.01	3.3763	0.92	3.4480	0.91	3.3857	0.77	0.1	5	30	5570	3.4159	0.88

For the linearly inseparable datasets considered above (iono, votes, WBC, ttt, germ) we repeated the comparative study using sufficiently powerful non-linear kernels able to allow linear separation in the appropriate feature space. Thus, in this case we no longer need to resort to soft margin approaches. For all datasets we used non-homogeneous polynomial kernels of degree $d = 4$ with parameter $c = 1$ except for the german dataset for which we used a Gaussian kernel with parameter $\sigma = 1$. In this study we had, of course, to employ the dual space formulation of red-MICRA. Our results are summarised in Table 7.33 where the value of the accuracy parameter ϵ corresponding to the lower value of the margin is set to $\epsilon = 0.03$ for LIBSVM and $\epsilon = 0.02$ for SVM^{light}. We see again that the speed of red-MICRA is of the same order of magnitude as the one of decomposition SVMs. Nevertheless, the results indicate that the primal space formulation is more advantageous for red-MICRA.

We also analysed several subsets of the Adult (32561 instances, 123 binary attributes) and of the Web (49749 instances, 300 binary attributes) datasets in the version of [46] with results presented in Table 7.34 and Table 7.35, respectively. Here $\Delta = 1$. Also, in both tables the lower value of the margin for LIBSVM corresponds to $\epsilon = 0.03$. For the Adult dataset no augmentation is required ($\rho = 0$) and the lower value of the margin for SVM^{light} corresponds to $\epsilon = 0.025$. For the Web dataset, instead, we do perform an augmentation for red-MICRA with parameter $\rho = 0.25$. Also, the lower value of

[1] Value obtained using the dual space formulation.

the margin for SVM^{light} in Table 7.35 is obtained with $\epsilon = 0.02$. We observe that the CPU-time required for red-MICRA to converge is shorter and exhibits a better scaling behaviour with the size of the dataset. Moreover, the shortage of memory as the dataset size grows apparently slows down LIBSVM. In contrast, SVM^{light} and red-MICRA are not affected.

TABLE 7.34: Results of a comparative study of LIBSVM, SVM^{light} and red-MICRA on several subsets of the Adult dataset.

subset size	LIBSVM				SVM^{light}				red − MICRA$^{0.05,0.9}$				
	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	η	N	$10^2\frac{\beta}{R}$	$10^2\gamma'$	Secs
1605	3.9383	1.41	3.9022	1.07	3.9375	3.02	3.8877	1.58	20	100	1.918	3.9038	0.63
3185	2.7437	5.55	2.7187	4.29	2.7434	11.3	2.7093	6.23	25	100	1.400	2.7187	1.73
6414	1.9292	22.5	1.9094	17.6	1.9290	71.3	1.9097	37.7	45	300	1.025	1.9111	5.83
11220	1.4499	73.2	1.4348	58.6	1.4497	283.4	1.4342	141.7	65	300	0.798	1.4356	14.7
16100	1.2069	389.7	1.1927	312.3	1.2068	638.2	1.1923	318.6	80	500	0.673	1.1950	28.7
22696	1.0154	1511.8	1.0030	1040.2	1.0154	1291.1	1.0042	683.5	95	500	0.580	1.0062	45.9
32561	0.8526	3902.3	0.8424	2484.5	0.8525	2733.8	0.8432	1439.4	105	600	0.492	0.8441	75.0

TABLE 7.35: Results of a comparative study of LIBSVM, SVM^{light} and red-MICRA on several subsets of the Web dataset.

subset size	LIBSVM				SVM^{light}				red − MICRA$^{0.05,0.9}$				
	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	η	N	$10^2\frac{\beta}{R}$	$10^2\gamma'$	Secs
2477	10.448	0.57	10.292	0.51	10.445	0.30	10.312	0.18	25	10	1.681	10.344	0.07
4912	7.0079	2.07	6.8967	1.83	7.0067	1.10	6.8909	0.61	25	10	1.212	6.9393	0.20
9888	4.8784	8.95	4.7970	7.82	4.8772	5.45	4.8072	3.22	30	10	0.868	4.8316	0.86
24692	2.9555	115.5	2.9066	90.2	2.9549	66.9	2.9111	32.1	50	10	0.535	2.9265	4.82
49749	2.1094	725.0	2.0723	635.8	2.1089	360.2	2.0771	176.4	70	10	0.405	2.0894	18.3

We repeated the comparative analysis on the subsets of the Adult dataset without exploiting the sparsity in the attributes of the initial instance space (only 14 at most out of the 123 binary attributes are true) in order to examine the extent to which the algorithms under investigation are able to take advantage of this sparsity. The parameters used are the same as in our previous analysis except for the memory parameter of LIBSVM and SVM^{light} which now had to be reduced from $m = 400$MB to $m = 360$MB in order to improve the performance of LIBSVM. Our results are presented in Table 7.36. By comparing Table 7.34 with Table 7.36 we see that although all three algorithms are able to take advantage of the sparsity red-MICRA seems to benefit the most.

Finally, we conducted an experiment with the very large multiclass Covertype dataset (581012 instances, 54 attributes) obtainable from the UCI repository, and studied the classification problem of the first class versus all the others treating again the whole dataset as a training set. Due to the memory difficulties encountered by LIBSVM we compared red-MICRA only with SVM^{light} for which we obtained only one margin value

TABLE 7.36: Results of a comparative study of LIBSVM, SVM^{light} and red-MICRA on several subsets of the Adult dataset without exploiting the sparsity of the input space.

subset	LIBSVM				SVM^{light}				red − MICRA$^{0.05,0.9}$				
size	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	$10^2\gamma'$	Secs	η	N	$10^2\frac{\beta}{R}$	$10^2\gamma'$	Secs
1605	3.9383	3.2	3.9022	2.9	3.9375	9.2	3.8877	4.7	20	100	1.918	3.9038	2.3
3185	2.7437	13.2	2.7187	11.9	2.7434	45.6	2.7093	25.1	25	100	1.400	2.7187	7.0
6414	1.9292	53.3	1.9094	48.2	1.9290	208.7	1.9097	110.7	45	300	1.025	1.9111	26.3
11220	1.4499	165.4	1.4348	150.3	1.4497	669.6	1.4342	336.5	65	300	0.798	1.4356	65.5
16100	1.2069	1363.0	1.1927	1105.3	1.2068	1456.7	1.1923	719.3	80	500	0.673	1.1950	118.5
22696	1.0154	5130.7	1.0030	3558.1	1.0154	2888.9	1.0042	1518.5	95	500	0.580	1.0062	170.7
32561	0.8526	12271.9	0.8424	8138.9	0.8525	6007.6	0.8432	3147.4	105	600	0.492	0.8441	268.4

corresponding to an accuracy parameter $\epsilon = 0.01$. Such a value of ϵ is sufficiently small to guarantee a margin γ' larger than 0.99γ. In the experiment we rescaled the dataset by multiplying all the attributes with 0.001 and their sparsity was fully exploited. From the results desplayed in Table 7.37 red-MICRA appears approximately 10 times faster than SVM^{light}.

TABLE 7.37: Results of a comparative study of SVM^{light} and red-MICRA on the Covertype dataset.

data	Δ	SVM^{light}			red − MICRA$^{0.05,0.9}$					
size		ϵ	$10^3\gamma'$	Secs	ρ	η	N	$10^5\frac{\beta}{R}$	$10^3\gamma'$	Secs
581012	10	0.01	15.774	47987.7	2	70	400	336	15.789	4728.0

Before concluding our comparative study of red-MICRA and SVMs we would like to point out that if we do not insist on margin values very close to the maximum margin the advantage of red-MICRA becomes more apparent. For an illustration we repeated the experiment on the Covertype dataset setting the accuracy parameter ϵ of SVM^{light} to the value $\epsilon = 0.1$. From the results desplayed in Table 7.38 we observe that for margin values close to 90% of the maximum margin γ red-MICRA appears approximately 50 times faster than SVM^{light}.

TABLE 7.38: Results of a comparative study of SVM^{light} and red-MICRA on the Covertype dataset for margin values lower than 99% of γ.

data	Δ	SVM^{light}			red − MICRA$^{0.05,0.9}$					
size		ϵ	$10^3\gamma'$	Secs	ρ	η	N	$10^5\frac{\beta}{R}$	$10^3\gamma'$	Secs
581012	10	0.1	14.517	28021.1	2	70	400	250	14.541	561.1

7.4 An evaluation

Our first comparative experimental study involved only PLAs. From the results reported one may conclude with very high confidence that ALMA, in spite of its theoretical merits, is by far the slowest among the large margin algorithms considered. The standard Perceptron with margin showed consistently a very good behaviour for values of the margin not very close to the maximum one. Also, CRAMMA with the appropriate choice of the effective learning rate competes with the Perceptron away from the maximum margin but becomes certainly faster in the vicinity of the optimal solution since it is provably able to approach such a solution arbitrarily close. Finally, agg-ROMMA and MICRA proved in all experiments to be the fastest among the large margin PLAs with agg-ROMMA having the tendency to require in some cases fewer epochs and MICRA always fewer updates in order to converge. On the other hand, the algorithmic implementations employing the fixed margin condition algorithms, although unable to approach the maximum margin solution extremely close, exhibited remarkable performance for margins of the order of $90 - 95\%$ of the maximum. Even more surprisingly, for such values of the margin they proved in some cases to be as fast as or even faster than MICRA.

In the comparative experimental study between PLAs and SVMs we have chosen MICRA as the best candidate to represent PLAs on the basis of its performance in the first comparative study. From our results we believe that it is fair to draw the conclusion that red-MICRA, incorporating a reduction technique in order to improve its speed, is able to compete with decomposition SVMs and proves much faster for large datasets and linear kernels.

Chapter 8

Conclusion

Large margin classification is desirable because it is intuitively appealing but most importantly because it is well-motivated from statistical learning theory. The subject of the present thesis was the development, theoretical analysis and experimental evaluation of Perceptron-like algorithms able to classify data with large margins from the separating hyperplane in an attempt to provide viable alternatives to the popular Support Vector Machines.

Our theoretical analysis based on the notion of stepwise convergence revealed that a Perceptron-like algorithm with an initial weight vector in the span of the data converges in a finite number of steps if the classification condition requires margin values which become, sooner or later, smaller than the maximum margin and the effective learning rate becomes eventually small enough without decreasing faster than linearly with the number of mistakes. However, the above conditions, which are easily satisfied by many algorithms including the fixed margin condition algorithms presented here, are by no means adequate to guarantee convergence to the optimal solution hyperplane. It turns out that convergence to the solution with maximum margin is obtained only asymptotically and provided the parameters of the algorithm scale appropriately with the relevant parameter controlling the asymptotic procedure. Assuming that the classification condition and the effective learning rate are governed by simple power-law rules involving the number of mistakes we obtained sufficient conditions for asymptotic convergence to the maximum margin solution. These conditions are not very constraining, thereby demonstrating that convergence to the optimal solution is not a property of some very special algorithmic constructions but rather characterises larger groups of algorithms. Thus, our analysis led to a significant enlargement of the class of Perceptron-like large margin classifiers. The same analysis, however, showed clearly that even a slight modification of such an algorithm may result in its failure to achieve the maximum margin if the conditions characterising its behaviour during the asymptotic procedure are violated. Such a sensitivity in the conditions governing the behaviour of the algorithms during the asymptotic process may be the reason for the inability of the standard Perceptron

with margin, unlike ALMA, to converge to the maximum margin solution in spite of the striking similarities of these two algorithms. The most general conditions, however, ensuring that a Perceptron-like algorithm is able to achieve the maximum margin are still to be discovered.

Our experimental analysis, on the other hand, revealed that the Perceptron-like large margin classifiers developed here converge in most cases faster than other similar algorithms. More importantly, the red-MICRA algorithm with slow relaxation of the misclassification condition and relatively fast decrease of the effective learning rate with the number of mistakes proved comparable in speed or faster than decomposition method SVMs, especially for large datasets. We should stress, however, that the performance of most of our algorithms depends on the choice of the (initial) value of the effective learning rate. Moreover, there is no reliable way of choosing in advance the value of the parameter that determines how close the achieved margin is to the maximum one in spite of the fact that there are lower bounds on the fraction of the maximum margin that the algorithm achieves. Thus, the usefulness of such algorithms depends on the experience of the user. Nevertheless, we find it remarkable that simple extensions of the old Perceptron algorithm are so competitive.

Appendix A

In the present Appendix we proceed to a sketch of the derivation of (5.50), (5.51) and (5.52) following the technique of [21].

Taking the inner product of (5.13) (with the denominator of its r.h.s. being denoted N_{t+1}) with the optimal direction $\boldsymbol{u}$, employing (5.1) and repeatedly applying the resulting inequality we have

$$1 \geq \boldsymbol{u}_{t+1} \cdot \boldsymbol{u} = \frac{\boldsymbol{u}_t \cdot \boldsymbol{u} + \eta_{\text{eff}} \boldsymbol{y}_k \cdot \boldsymbol{u}/R}{N_{t+1}} \geq \frac{\boldsymbol{u}_t \cdot \boldsymbol{u}}{N_{t+1}} + \frac{\eta_{\text{eff}} \gamma_{\text{d}}/R}{N_{t+1}}$$
$$\geq \frac{\boldsymbol{u}_1 \cdot \boldsymbol{u}}{N_{t+1} N_t \cdots N_2} + \frac{\eta_{\text{eff}} \gamma_{\text{d}}}{R} \left(\frac{1}{N_{t+1}} + \frac{1}{N_{t+1} N_t} + \cdots + \frac{1}{N_{t+1} N_t \cdots N_2} \right) \ . \tag{A.1}$$

For the normalisation factor N_{m+1} we can derive the inequality

$$N_{m+1}^{-1} \geq \alpha^{-1} \left(1 + 2A/m\right)^{-\frac{1}{2}} \equiv r_m \ ,$$

where

$$\alpha = \left(1 + \eta_{\text{eff}}^2\right)^{\frac{1}{2}} = \left(1 + \eta_0^2 R^2/\beta^2\right)^{\frac{1}{2}}$$

and

$$A = \eta_{\text{eff}} \beta \alpha^{-2}/R = \eta_0 \alpha^{-2} \ ,$$

which if substituted in (A.1) leads to

$$\frac{1}{\eta_0} \frac{\beta}{\gamma_{\text{d}}} \geq \sum_{m=1}^{t} \prod_{j=m}^{t} r_j \geq \sum_{m=2}^{t} \prod_{j=m}^{t} r_j = r_t \sum_{m=2}^{t} \prod_{j=m}^{t-1} r_j \geq r_t \sum_{m=2}^{t} \alpha^{m-t} \left(\frac{m-1}{t-1} \right)^A \tag{A.2}$$

given that $\boldsymbol{a}_1 \cdot \boldsymbol{u} > 0$ and $\eta_{\text{eff}} = \eta_0 R/\beta$. At the last stage of the previous inequality we made use of

$$-\ln \prod_{j=m}^{t-1} r_j = \sum_{j=m}^{t-1} \ln r_j^{-1} \leq (t-m) \ln \alpha + \sum_{j=m}^{t-1} \frac{A}{j} \leq \ln \alpha^{t-m} + A \int_{m-1}^{t-1} \frac{dj}{j} \ .$$

Taking into account (5.40) and the fact that $A \leq \eta_0$ we have that $(1 + 2A/t)^{\frac{1}{2}} \leq (1 + 2\eta_0 \gamma_{\text{d}}/\beta)^{\frac{1}{2}} \leq 1 + \eta_0 \gamma_{\text{d}}/\beta$. Using the latter inequality and setting $l = m - 1$ (A.2)

gives

$$1+\frac{1}{\eta_0}\frac{\beta}{\gamma_\mathrm{d}} \geq \alpha^{-t}(t-1)^{-A}\sum_{l=1}^{t-1} l^A\alpha^l \ . \tag{A.3}$$

Let us first assume that $A \leq 1$. Then, since $l/(t-1) \leq 1$, we can set $A = 1$ in (A.3) and using

$$\sum_{l=1}^{n} l\alpha^l = \alpha\frac{d}{d\alpha}\sum_{l=1}^{n}\alpha^l = \alpha\frac{d}{d\alpha}\left(\alpha\frac{\alpha^n-1}{\alpha-1}\right) = \frac{n\alpha^{n+1}}{(\alpha-1)^2}\left\{(\alpha-1)-\frac{1-\alpha^{-n}}{n}\right\}$$

obtain

$$\frac{1-\alpha^{-(t-1)}}{t-1} \geq (\alpha-1)\left\{1-(\alpha-1)\left(1+\frac{1}{\eta_0}\frac{\beta}{\gamma_\mathrm{d}}\right)\right\} \ . \tag{A.4}$$

The r.h.s. of (A.4) is certainly positive if $\eta_\mathrm{eff} < \gamma_\mathrm{d}/R$ or $\eta_0 < \beta\gamma_\mathrm{d}/R^2$. Since the l.h.s. is a monotonically decreasing function of t vanishing in the limit $t \to \infty$, (A.4) gives rise to an upper bound on t. To obtain an approximation of this upper bound (i.e. obtain a less restrictive upper bound) we employ the relation $\alpha^{-(t-1)} = e^{-(t-1)\ln\alpha}$ and the inequalities $(1-e^{-x})/x < 1-x/2+x^2/6$ for $x > 0$, $(x-1)-(x-1)^2/2 < \ln x < (x-1)$ for $x > 1$ and $1/\ln x < x/(x-1)$ for $1 < x \leq 2$. Then, from (A.4) we have

$$1-(\alpha-1)\left(1+\frac{1}{\eta_0}\frac{\beta}{\gamma_\mathrm{d}}\right) \leq \frac{1-\alpha^{-(t-1)}}{(\alpha-1)(t-1)} = \frac{1-e^{-\ln\alpha(t-1)}}{(\alpha-1)(t-1)}$$

$$\leq \frac{1-e^{-\ln\alpha(t-1)}}{\ln\alpha(t-1)} \leq 1-\frac{1}{2}\ln\alpha(t-1)+\frac{1}{6}\ln^2\alpha(t-1)^2$$

or

$$(t-1)^2-3\frac{t-1}{\ln\alpha}+6\frac{\alpha-1}{\ln^2\alpha}\left(1+\frac{1}{\eta_0}\frac{\beta}{\gamma_\mathrm{d}}\right) \geq 0 \ .$$

The last relation, making use of the inequalities mentioned above, yields

$$(t-1)^2-\frac{3}{\alpha-1}(t-1)+6\frac{\alpha^2}{\alpha-1}\left(1+\frac{1}{\eta_0}\frac{\beta}{\gamma_\mathrm{d}}\right) \geq 0$$

which gives the expected upper bound on $(t-1)$, namely the smallest positive root of the corresponding quadratic equation

$$t-1 \leq \frac{3}{2}\frac{1}{\alpha-1}\left(1-\sqrt{1-\frac{8}{3}\alpha^2(\alpha-1)\left(1+\frac{1}{\eta_0}\frac{\beta}{\gamma_\mathrm{d}}\right)}\right)$$

$$\leq \frac{3}{2}\frac{1}{\alpha-1}\left\{1-\left(1-\frac{4}{3}\alpha^2(\alpha-1)\left(1+\frac{1}{\eta_0}\frac{\beta}{\gamma_\mathrm{d}}\right)-\frac{32}{9}\alpha^4(\alpha-1)^2\left(1+\frac{1}{\eta_0}\frac{\beta}{\gamma_\mathrm{d}}\right)^2\right)\right\}$$

$$= 2\alpha^2\left(1+\frac{1}{\eta_0}\frac{\beta}{\gamma_\mathrm{d}}\right)+\frac{16}{3}\alpha^4(\alpha-1)\left(1+\frac{1}{\eta_0}\frac{\beta}{\gamma_\mathrm{d}}\right)^2 \ .$$

Real roots exist if $\eta_0 < \beta\gamma_\mathrm{d}/\sqrt{6}R^2$ and the smallest one was approximated by employing

the inequality $\sqrt{1-x} \geq 1 - x/2 - x^2/2$. Taking into account that $(\alpha - 1) \leq \frac{\eta_0^2 R^2}{2\beta^2}$ we arrive at the bound of (5.50) from which (5.51) is readily derivable.

If $A \leq 2$, again because $l/(t-1) \leq 1$, we can set $A = 2$ in (A.3). Then, using

$$\sum_{l=1}^{n} l^2 \alpha^l = \alpha \frac{d}{d\alpha} \sum_{l=1}^{n} l\alpha^l = \frac{n\alpha^{n+1}}{(\alpha-1)^3} \left\{ n(\alpha-1)^2 - 2(\alpha-1) + (\alpha+1)\frac{1-\alpha^{-n}}{n} \right\} \ ,$$

we get

$$\frac{\alpha-1}{t-1} - \frac{1-\alpha^{-(t-1)}}{(t-1)^2} \geq \frac{1}{2}(\alpha-1)^2 \left\{ 1 - (\alpha-1)\left(1 + \frac{1}{\eta_0}\frac{\beta}{\gamma_\mathrm{d}}\right) \right\} \ . \tag{A.5}$$

The l.h.s. of (A.5) can be shown to be a strictly decreasing function of t vanishing as $t \to \infty$ whereas its r.h.s. is positive if $\eta_0 < \beta\gamma_\mathrm{d}/R^2$. Thus, (A.5) leads to an upper bound on t. To find an approximation of such a bound we employ again the relation $\alpha^{-(t-1)} = e^{-(t-1)\ln\alpha}$ and the additional inequality $(1 - e^{-x})/x > 1 - x/2 + x^2/6 - x^3/24$ for $x > 0$ in (A.5) to obtain the less restrictive relation

$$(t-1)^3 - \frac{4}{\alpha-1}(t-1)^2 + 12\frac{\alpha^2}{\alpha-1}\left(1 + \frac{1}{\eta_0}\frac{\beta}{\gamma_\mathrm{d}}\right)(t-1) + 12\frac{\alpha^4}{(\alpha-1)^2} \geq 0 \ .$$

In the limit $\frac{\beta}{R} \to \infty$ the above inequality is satisfied if t is bounded from above by the smallest positive root of the corresponding cubic equation. This leads to (5.52).

Bibliography

[1] Agmon, S.: The relaxation method for linear inequalities. Canadian J. Math. **6** (1954) 382–392

[2] Aizerman, M. A., Braverman, E.M., Rozonoer L.I.: Theoretical foundations of the potential function method in pattern recognition learning. Automation and remote control **25** (1964) 821–837

[3] Anlauf, J. K., Biehl M.: The adatron: an adaptive perceptron algorithm. Europhysics Letters **10** (1989) 687–692

[4] Azoury, K., Warmuth, M. K.: Relative loss bounds for on-line density estimation with the exponential family of distributions. Machine Learning **43** (2001) 211–246

[5] Bartlett, P., Shawe-Taylor, J.: Generalization performance of support vector machines and other pattern classifiers. In Advances in Kernel Methods-Support Vector Learning (1999) MIT Press

[6] Bennett, K. P., Mangasarian, O. L.: Robust linear programming discrimination of two linearly inseparable sets. Optimisation Methods and Software **1** (1992) 23–34

[7] Bishop, C.: Neural networks for pattern recognition (1995) Oxford University Press

[8] Blake, C. L., Merz, C. J.: UCI repository of machine learning databases. (1998) http://www.ics.uci.edu/∼ mlearn/MLRepository.html

[9] Boser, B. E., Guyon, I. M., Vapnik, V. N.: A training algorithm for optimal margin classifiers. Proceedings of the 5^{th} Annual ACM Workshop on Computational Learning Theory (1992) 144–152, ACM Press

[10] Cesa-Bianchi, N., Conconi, A., Gentile, C.: A second-order perceptron algorithm. Siam Journal of Computing **34**(3) (2005) 640–668

[11] Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. (2001) Software available at http://www.csie.ntu.edu.tw/∼ cjlin/libsvm.

[12] Chernoff, H.: A measure of asymptotic efficiency of test of a hypothesis based on the sum of observations. Ann. Math. Stat. **23** (1952) 493–507

[13] Cortes, C., Vapnik, V.: Support vector networks. Machine Learning **20** (1995) 273–297

[14] Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines (2000) Cambridge, UK: Cambridge University Press

[15] Duda, R.O., Hart, P.E.: Pattern Classsification and Scene Analysis (1973) Wiley-Interscience

[16] Fisher, R. A.: Contributions to Mathematical Statistics (1952) J. Wiley, New York

[17] Freund, Y., Shapire, R. E.: Large margin classification using the perceptron algorithm. Machine Learning **37**(3) (1999) 277–296

[18] Friess, T., Harrison, R.: Support vector neural networks: the kernel adatron with bias and soft margin. University of Sheffield, Department of ACSE, Technical Report (1998) ACSE-TR-752

[19] Friess, T., Cristianini, N., Campbell, C.: The kernel-Adatron: a fast and simple training procedure for support vector machines. In Proceedings of the 15^{th} International Conference on Machine Learning (1998) 188–196

[20] Gentile, C., Littlestone, N.: The robustness of the p-norm algorithms. In Proceedings of the 12th Conference on Computational Learning Theory (1999) 1–11

[21] Gentile, C.: A new approximate maximal margin classification algorithm. Journal of Machine Learning Research **2** (2001) 213–242

[22] Gentile, C.: The robustness of the p-norm algorithms. Machine Learning **53** (2003) 265–299

[23] Gilbert, G.: Minimising the quadratic form on a convex set. SIAM J. Contr. **4** (1966) 61–79

[24] Gorman, R. P., Sejnowski, T. J.: Analysis of hidden units in a layered network trained to classify sonar targets. Neural Networks **1** (1988) 75–89

[25] Grove, A., Littlestone, N., Schuurmans, D.: General convergence results for linear discriminant updates. In Proceedings of the 10th Conference on Computational Learning Theory (1997) 171–183

[26] Hildreth, C.: A quadratic programming procedure. Naval Research Logistics Quarterly **4** (1957) 79–85

[27] Hush, D., Scovel, C.: On the VC dimension of bounded margin classifiers. Machine Learning **45** (2001) 33–44

[28] Joachims, T.: Making large-scale svm learning practical. In Advances in Kernel Methods-Support Vector Learning (1999) MIT Press

[29] Karush, W.: Minima of Functions of Several Variables with Inequalities as Side Constraints. Department of Mathematics, University of Chicago, Msc Thesis (1939)

[30] Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: A fast iterative nearest point algorithm for support vector machine classifier design. IEEE Transactions on Neural Networks **11**(1) (1999) 124–136

[31] Kivinen, J., Warmuth, M. K.: The perceptron algorithm vs. winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. Proceedings of the 8^{th} Annual Conference on Computational Learning Theory (1995) ACM Press

[32] Kivinen, J., Warmuth, M. K.: Additive versus exponentiated gradient updates for linear prediction. Information and Computation **132**(1) (1997) 1–64

[33] Klasner, N., Simon, H. U.: From noise-free to noise-tolerant and from on-line to batch learning. In Proceedings of the 8th Conference on Computational Learning Theory (1995) 250–257

[34] Kowalczyk, A.: Maximal margin perceptron. In Advances in Large Margin Classifiers (1999) MIT Press

[35] Krauth, W., Mézard, M.: Learning algorithms with optimal stability in neural networks. Journal of Physics **A 20** (1987) L745–L752

[36] Kuhn, H., Tucker, A.: Nonlinear programming. In Proceedings of 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics (1951) 481–492, University of California Press

[37] Littlestone, N., Warmuth, M. K.: The weighted majority algorithm. Information and Computation **108**(2) (1994) 212–261

[38] Li, Y., Long, P.: The relaxed online maximum margin algorithm. Journal of Machine Learning **46** (2002) 361-387

[39] Li, Y., Zaragoza, H., Herbrich, R., Shawe-Taylor, J., Kandola, J.: The perceptron algorithm with uneven margins. In Proceedings of the 19^{th} International Conference on Machine Learning (2002) 379–386

[40] Littlestone, N.: Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. Machine Learning **2** (1988) 285–318

[41] Luenberger, D.: Linear and Nonlinear Programming (1984) Addison-Wesley

[42] Mercer, T.: Functions of positive and negative type and their connection with the theory of integral equations. Trans. Lond. Philos. Soc. A, **209** (1909) 415–446

[43] Mitchell, B. F., Dem'yanov, V. F., Malozenov, V. N.: Finding the point of a polyhedron closest to the origin. SIAM J. Contr. **12** (1974) 19–26

[44] Novikoff, A. B. J.: On convergence proofs on perceptrons. In Proceedings of the Symposium on the Mathematical Theory of Automata Volume 12 (1962) 615–622, Polytechnic Institute of Brooklyn

[45] Osuna, E., Freund, R., Girosi, F.: An improved training algorithm for support vector machines. Neural Networks for Signal Processing VII-Proc. IEEE NNSP 97 (1997) 276–285

[46] Platt, J. C.: Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research (1998)

[47] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review **65**(6) (1958) 386–408

[48] Schölkopf, B., Smola, A.J.: Learning with Kernels (2002) MIT Press

[49] Shapire, R.: The strength of weak learnability. Machine Learning **5**(2) (1990) 197–227

[50] Shawe-Taylor, J., Bartlett, P. L., Williamson, R. C., Anthony, M.: Structural risk minimization over data-dependent hierarchies. IEEE Transactions on Information Theory **44**(5) (1998) 1926–1940

[51] Shawe-Taylor, J., Cristianini, N.: Margin distribution and soft margin. In Advances in Large Margin Classifiers (1999) MIT Press

[52] Shawe-Taylor, J., Cristianini, N.: Margin distribution bounds on generalisation. In Proceedings of the European Conference on Computational Learning Theory, Euro-COLT'99, (1999)

[53] Shawe-Taylor, J., Cristianini, N.: Further results on the margin distribution. In Proceedings of the 12th Annual Conference on Computational Learning Theory, COLT'99, (1999)

[54] Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis (2004) Cambridge, UK: Cambridge University Press

[55] Smith, F. W.: Pattern classifier design by linear programming. IEEE Transactions on Computers, **C-17** (1968) 367–372

[56] Tsampouka, P., Shawe-Taylor, J.: Analysis of generic perceptron-like large margin classifiers. ECML 2005, LNAI **3720** (2005) 750–758, Springer-Verlag

[57] Tsampouka, P., Shawe-Taylor, J.: Constant rate approximate maximum margin algorithms. ECML 2006, LNAI **4212** (2006) 437–448, Springer-Verlag

[58] Tsampouka, P., Shawe-Taylor, J.: Approximate maximum margin algorithms with rules controlled by the number of mistakes. In Proceedings of the 24$^{\text{th}}$ International Conference on Machine Learning (2007) 903–910

[59] Vanderbei, R.: Loqo: An interior point code for quadratic programming. Technical Report MSR-TR-98-14, Microsoft Research (1998)

[60] Vapnik, V. N., Chervonenkis, A.Ja.: On the uniform convergence of relative frequencies of events to their probabilities. Sov. Math. Dokl. **181**(4) (1968)

[61] Vapnik, V. N., Chervonenkis, A.Ja.: On the uniform convergence of relative frequencies of events to their probabilities. Theory Probab. Apl. **16** (1971) 264–280

[62] Vapnik, V. N., Chervonenkis, A.Ja.: Theory of Pattern Recognition (1974) Nauka, Moscow

[63] Vapnik, V. N., Chervonenkis, A.Ja.: Necessary and sufficient conditions for the uniform convergence of the means to their expectations. Theory Probab. Appl. **26** (1981) 532–553

[64] Vapnik, V. N.: Estimation of Dependencies Based on Empirical Data (1982) Springer-Verlag

[65] Vapnik, V. N., Chervonenkis, A.Ja.: The necessary and sufficient conditions for consistency of the method of empirical risk minimisation. Pattern Recogn. and Image Analysis **1**(3) (1989) 284–305

[66] Vapnik, V. N.: The Nature of Statistical Learning Theory (1995) Springer Verlag

[67] Vapnik, V. N.: Statistical Learning Theory (1998) Wiley

[68] Vovk, V.: Aggregating strategies. In Proceedings of the 3rd Annual Workshop on Computational Learning Theory (1990) 372–383

[69] Zoutendijk, G.: Methods of Feasible Directions: a Study in Linear and Non-linear Programming (1970) Elsevier