Notes on Support Vector Machines

Fernando Mira da Silva

Fernando.Silva@inesc.pt

Neural Network Group

I N E S C

November 1998

Abstract

This report describes an empirical study of Support Vector Machines (SVM) which aims to review the foundations and properties of SVM based on the graphical analysis of simple binary classification tasks.

SVM theory is briefly reviewed by discussing possible learning rules for linear separable problems and the role of the *classification margin* in generalization. This leads to the formulation of the SVM learning procedure for linear separable problems. It is then shown how it is possible to extend the SVM approach to non-separable problem by relaxing the strict separability constraint of the original formulation.

The application of SVM to nonlinear problems by feature expansion techniques is then briefly reviewd. The effect of the dimension of the feature space on the SVM solution is briefly considered based on a simple bi-dimensional problem. Finally, it is discussed how explicit feature expansion can be avoided by the use of kernels.

A qualitative analysis of the generalization behavior of support vector machines is performed by the analysis of the classification regions found by SVM in simple, synthetic examples. These classification regions are compared with those found by conventional multi-layer and radial basis functions networks.

In the last part of this document, practical problems regarding SVM optimization are briefly discussed and some pointers and references regarding this issue are supplied.

Contents

List of Figures

Chapter 1

Introduction

1.1 Classifiers, learning and generalization

The interest on supervised, automatic learning systems for pattern classification was greatly enhanced by the advances in connectionist and other non-symbolic learning paradigms. In supervised learning, the system is presented with examples of input patterns and the corresponding desired class. During the learning phase, the system is expected to develop a classification model which is able to correctly classify arbitrary input data patterns.

One of the outstanding problems on supervised learning is to identify learning procedures which lead to a system with a good generalization behavior, e. g., which is able to correctly classify previously unseen data patterns. A first and intuitive condition to achieve a good generalization seems to stem from the structure of the training set. In a first approach, one may expect that good generalization depends on how well the training set represents the overall data distribution. However, neither this condition is strictly required, nor constitutes a sufficient condition for optimal generaliztion.

Discussion of generalization and learning often involve the concepts of *computational feasibility* and *computational capacity*. For a system to perform a given classification task, its model structure must be able to implement it. This means that the classification process must be *feasible* by the computational model. This requirement may imply the use of a fairly complex computational model. But, on the other hand, one must consider the *computational capacity* of the model. The computational capacity of a given system may be formally quantified by the Vapnik-Chervonenkis (VC) dimension (Vapnik, 1995). Broadly speaking, the VC dimension of a binary classification system is the maximum number of training samples that can be learned by the system without error for all possible binary labellings of the classification models (Haykin, 1994). The introduction of the VC dimension enables to derive an upper bound for the the misclassification rate in the application domain (*risk*) given the misclassification rate observed on the training set (*empirical risk*). While the theoretical bounds which result from the VC dimension are often considered too conservative (see for instance (Baum and Haussler, 1989; Baum, 1990; Cohn and Tesauro, 1991; Ji and Psaltis, 1992)), the VC dimension remains an important tool for statistical analysis of learning and generalization. A too large VC dimension may lead to an over fitted system, which tends to work as a memory regarding the training samples, but which develops a poor generalization behavior.

In practice, good generalization often requires a trade-off between the computational feasibility of the model and its computational capacity. This is often achieved by including regularization techniques in the learning procedure. Several neural network learning models, as weight sharing (Le Cun et al., 1990) or weight decay (Krogh and Hertz, 1992; Guyon et al., 1992) attempt to reduce the VC dimension without affecting the computational feasibility of the system.

Recently, *support vector machines* (SVM) received a great deal of attention from the machine learning community. Possibly the most attractive feature of SVM stems from the attempt to derive a feasible classifier with a minimum VC-dimension for a given classification task. This report describes an empirical study of Support Vector Machines which aims to review the foundations and properties of SVM through the graphical analysis of simple binary classification tasks.

This report is structured as follows. In chapter 1 the basic principles of linear classification systems are introduced. In chapter 2 it is introduced the optimality principle of support vector machines for linear classification tasks and its application to separable and non-separable problems. In chapter 3 its discussed the extension of SVM to non linear classification problems by feature expansion. In chapter 4 its considered the use of Kernels in non-linear SVMs. In chapter 5 some details regarding implementation of SVM are considered. Finally, in chapter 6, conclusions are presented.

1.2 Linear classification systems

Consider a linearly separable binary classification problem where the training set is defined by P pairs

$$\mathcal{T} = \{(\mathbf{x}_i, d_i); i = 1, \cdots, P\}. \tag{1.1}$$

Each $\mathbf{x}_i \in \mathbf{R}^N$ is an input pattern and $d_i \in \{-1, 1\}$ identifies the target class for input pattern $\mathbf{x}_i$.

By definition, stating that the problem is linearly separable is equivalent to state that there is an hyper-plane in $\mathbf{R}^N$ splitting the input domain such that patterns of the same class lay in the same region of the input space. The solution of the classification problem reduces to find any hyper-plane which has this property. Without further restrictions, there is in general an infinity of

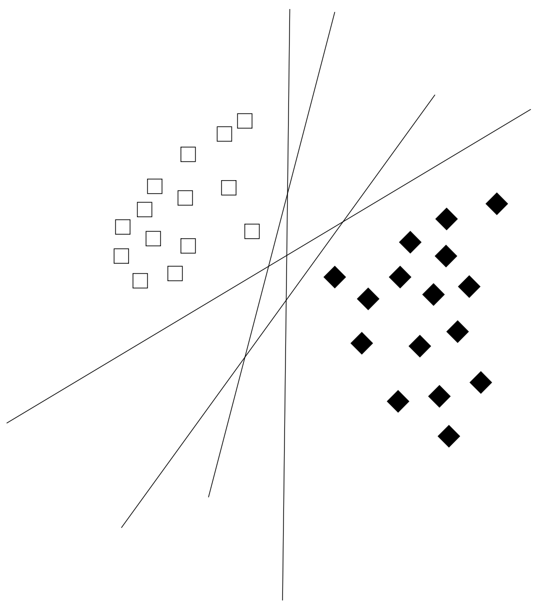

Figure 1.1: Possible solutions for a linearly separable classification problem.

solutions to this problem. Examples of several possible solutions for a simple bi-dimensional case are shown in figure 1.1.

If the domain of the classification problem is the training set $\mathcal{T}$ alone, then any hyper-plane in figure 1.1 provides a correct solution to the classification problem. However, further to the correct classification of all patterns in the training set[1], a good classification system is expected to provide correct classes for previously unseen data patterns. Therefore, to achieve optimal generalization, the classification problem must be restated as follows: among all possible solutions, choose the one which *hopefully* provides the best generalization ability. Clearly, this is an ill-posed problem, since it is impossible to guess the generalization behavior without making strong assumptions on the underlying data distribution.

A linear classification system halves the input space with an hyper-plane $\mathcal{P}$ defined by the equation

$$\mathbf{w}^T\mathbf{x} + b = 0. \tag{1.2}$$

Geometrically, $\mathbf{w}$ is a vector orthogonal to $\mathcal{P}$. The class y_i of pattern $\mathbf{x}_i$ is defined by

$$y_i = \begin{cases} -1 & \text{if} \quad \mathbf{w}^T\mathbf{x} + b < 0 \\ 1 & \text{if} \quad \mathbf{w}^T\mathbf{x} + b \geq 0 \end{cases} . \tag{1.3}$$

[1]In the more general case, the optimal system that minimizes the classifi cation error in the application domain may even misclassify one or more patterns in the training set. However, we assume here that the problem is strictly linearly separable and that there is no noise in the training set samples.

The original perceptron (Rosenblatt, 1958), the adaline (Widrow and Hoff, 1960) or the neuron unit with sigmoidal activation function (Rumelhart et al., 1986) are among the simplest binary classification systems with learning ability. The output of a neuron unit with an activation function $F(.)$ for an input pattern $\mathbf{x}_i$ is given by

$$\begin{aligned} s_i &= F(\sum_{k=1}^{N} w_k x_{i(k)} + b) \\ &= F(\mathbf{w}^T \mathbf{x}_i + b) \end{aligned} \tag{1.4}$$

where $x_{i(k)}$ is the k component of $\mathbf{x}_i$ and $\mathbf{w}$ is a vector of components w_k. The function $F(.)$ is usually a monotonic sigmoidal type function. According to (1.3), the class y_i of input pattern $\mathbf{x}_i$ is obtained from $\mathbf{s}_i$ by

$$y_i = \begin{cases} -1 & \text{if} \quad s_i < t_h \\ 1 & \text{if} \quad s_i \geq t_h \end{cases} . \tag{1.5}$$

where t_h is a pre-specified threshold. In neural network terminology, the parameters w_k are the *weights* and b is the *bias input.*

The learning phase of the sigmoidal unit reduces to find the parameters ($\mathbf{w}$,b) that minimize the total output error on the training set, given by

$$\mathcal{E}(\mathbf{w}, b) = \sum_{i=1}^{P} h(d_i, s_i) \tag{1.6}$$

where $h(d, s)$ is an arbitrary distortion measure. If $\mathcal{E}$ refers to the quadratic error and $F(.)$ is an odd function with values in $]-1, 1[$, then the distortion measure can be simply defined as $h(d, s) = (d - s)^2$. Since $\mathcal{E}(\mathbf{w}, b)$ depend on all training samples, the parameters $(\mathbf{w}_0, b_0)$ found by the minimization process – and hence $\mathcal{P}$ – also depend on all training samples. This is a reasonable result, since $\mathcal{P}$ must correctly classify all training patterns.

However, it is also reasonable to argue that, provided that all training patterns are correctly classified, the *exact* location of $\mathcal{P}$ should only depend on patterns laying near the decision boundary. In fact, these patterns are the most critically relevant for the definition of the exact boundary of the decision region and it is natural to expect that generalization is, above all, dependent on this boundary. Accordingly, one may expect that correctly classified patterns far away from $\mathcal{P}$ should be almost irrelevant regarding the generalization behavior of the classification system.

The hypothesis that the optimal location of $\mathcal{P}$ must only depend on the data patterns near to the decision boundary is a key principle of support vector machines. Under this assumption, the learning process must be able to identify this set of patterns – the support vectors – and it must be able to express the optimal parameters $(\mathbf{w}, b)$ of the classifier only as a function of these samples.

Chapter 2

Optimal linear classifi cation

2.1 The separable case

2.1.1 Formulation

In its simplest form, the learning criteria of support vector machines requires that all patterns within the training set are linearly separable by a single hyper-plane $\mathcal{P}$. This hypothesis is equivalent to state that the parameters $(\mathbf{w}, b)$ of the classifier must meet the set of P constraints

$$(\mathbf{w}^T \mathbf{x}_i + b) d_i \geq 0, \quad i = 1, \cdots, P. \tag{2.1}$$

Let us define the *classification margin* ρ as the minimum distance between $\mathcal{P}$ and the closest pattern to $\mathcal{P}$. This means

$$\rho = \min_i D(\mathbf{x}_i, \mathcal{P}) \tag{2.2}$$

where $D(\mathbf{x}, \mathcal{P})$ is the Euclidean distance from pattern $\mathbf{x}$ to $\mathcal{P}$. This distance is given by

$$D(\mathbf{x}, \mathcal{P}) = \frac{|\mathbf{w}^T \mathbf{x} + b|}{\|\mathbf{w}\|}. \tag{2.3}$$

The classification margin ρ can be seen as a measure of as well the hyper-plane $\mathcal{P}$ performs in the task of separating the two binary classes. In fact, a small value of ρ means that $\mathcal{P}$ is close to one or more samples of one or two classes and, therefore, there is a reasonably high probability that samples not included in the training set may fall in the wrong side of the classification region. On the other hand, if ρ is large, such probability is significantly reduced.

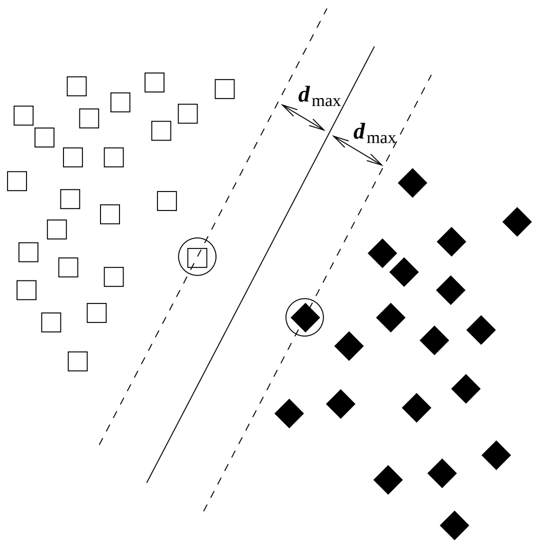

Figure 2.1: Optimal classification hyper-plane. Note that the exact location of the hyper-plane only depends on two support vectors.

The learning criteria of support vector machines reduces to find the set of parameters $(\mathbf{w}, b)$ that maximize the classification margin ρ under the constraints defined by (2.1). The optimal classification hyper-plane according to this criteria for a simple bi-dimensional example is depicted in figure 2.1. Note that, as discussed before, the exact location of $\mathcal{P}$ is only dependent on the *support vectors* closer to the hyper-plane (in this example, two support vectors). One of the relevant features of this classification criteria is that, under certain hypothesis, it can be related with the principle of structural risk minimization (Vapnik, 1992; Guyon et al., 1992), therefore providing theoretical grounds for the justification of a good generalization ability of these systems (Vapnik, 1995; Gunn, 1997; Burges, 1986).

Note that there is one degree of freedom in the definition of $\mathcal{P}$ by the condition $\mathbf{w}^T\mathbf{x} + b = 0$. In fact, all set of parameters $(k\mathbf{w}, kb)$ with $k > 0$ define the same hyper-plane $\mathcal{P}$ and result in the same classification system. Therefore, without loss of generality, it is possible to consider only the subset of parameters $(\mathbf{w}, b)$ that verify the condition

$$\min_i \left\{ \mathbf{w}^T\mathbf{x}_i + b \right\} = 1. \tag{2.4}$$

Given (2.2) and (2.3), this additional constraint is equivalent to state that

$$\|\mathbf{w}\| = \frac{1}{\rho}. \tag{2.5}$$

Furthermore, this condition allow us to rewrite the classification constraint (2.1) as

$$(\mathbf{w}^T\mathbf{x}_i + b)d_i \geq 1, \quad i = 1, \cdots, P. \tag{2.6}$$

Note, once more, that this reduces to sate that once condition (2.1) is satisfied for an arbitrary set of parameters $(\mathbf{w}', b')$, then it is possible to choose a $k > 0$ such that $(\mathbf{w} = k\mathbf{w}', b = kb')$ also verify (2.6).

Let us consider again the classification margin. From (2.5) it results that

$$\rho = \frac{1}{\|\mathbf{w}\|}. \tag{2.7}$$

With this condition, the optimization problem can be stated as follows: maximize

$$f'(\mathbf{w}) = \frac{1}{\|\mathbf{w}\|}. \tag{2.8}$$

under constraints

$$(\mathbf{w}^T \mathbf{x}_i + b)d_i - 1 \geq 0, \quad i = 1, \cdots, P \tag{2.9}$$

If there is P patterns on the training set, the optimization will have P constraints. Note that b, while absent from (2.8), is implicitly defined by the constraints (2.9).

In order to avoid the square root associated to $\|\mathbf{w}\|$ and to simplify the overall formulation, instead of the *maximization* of $f'(\mathbf{w})$, it is usually considered the problem of *minimization* of

$$f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.10}$$

where the factor $1/2$ is only included for convenience.

The minimization of $f(\mathbf{w})$ under constraints (2.9) can be achieved by minimizing the Lagrangian. The Lagrangian results from adding to $f(\mathbf{w})$ the left side of constraints (2.9), each one multiplied by a Lagrange multiplier α_i, $i = 1, \cdots, P$. Note that constraints of the form $c_i \geq 0$ are subtracted from the objective function with positive Lagrange multipliers to form the Lagrangian. Therefore, we have

$$\begin{aligned} L &= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{P} \alpha_i \left((\mathbf{w}^T \mathbf{x}_i + b)d_i - 1 \right) \\ &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T \sum_{i=1}^{P} \alpha_i \mathbf{x}_i d_i - b \sum_{i=1}^{P} \alpha_i d_i + \sum_{i=1}^{P} \alpha_i \end{aligned} \tag{2.11}$$

Under this formulation, the optimization problem is reduced to finding the minimization of L with respect to $\mathbf{w}$ and b, subject to

$$\left(\frac{\partial L}{\partial \alpha_i} = 0; \quad \alpha_i \geq 0 \right), \quad i = 1, \cdots, P \tag{2.12}$$

which is equivalent to state that the solution is found at a saddle point of the Lagrangian in respect to the Lagrange multipliers. This is a convex quadratic programming problem, since the objective function is convex and each constraint defines itself a convex set.

While the above optimization problem can be directly solved using any standard optimization package, it requires the explicit computation of $\mathbf{w}$. While this can be considered a quite natural requirement at this stage, avoiding the explicit computation of $\|\mathbf{w}\|$ is a key requirement for the application of kernel function to SVM (this subject will be discussed in further detail in chapter 4). In this problem, the explicit computation of $\mathbf{w}$ can be avoided using the *dual formulation* of Lagrangian theory (Fletcher, 1987). As a side result, this dual formulation will also lead to a simpler optimization problem.

Classical Lagrangian duality enables to replace a given optimization problem by its dual formulation. In this case, the Lagrangian duality states that the minimum of L that is found solving the *primal problem*

$$\min_{\mathbf{w},b} L \left| \left\{ \left(\frac{\partial L}{\partial \alpha_i} = 0; \quad \alpha_i \geq 0 \right), \quad i = 1, \cdots, P \right\} \right. \tag{2.13}$$

occurs at the same point $(\mathbf{w}, b, \alpha_i)$ where it is found the maximum of L that results from solving the *dual problem*

$$\max_{\alpha_i} L \left| \left\{ \frac{\partial L}{\partial \mathbf{w}} = 0; \quad \frac{\partial L}{\partial b} = 0; \quad \alpha_i \geq 0, \quad i = 1, \cdots, P \right\} \right. . \tag{2.14}$$

This particular dual formulation is called the Wolfe dual (Fletcher, 1987)[1].

Solving (2.14) implies to find the points where the derivatives of L with respect to $\mathbf{w}$ and b vanish. Let us consider the derivative with respect to $\mathbf{w}$ first. From (2.10) it results

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}} &= \frac{\partial}{\partial \mathbf{w}} \left[\frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T \sum_{i=1}^{P} \alpha_i \mathbf{x}_i d_i - b \sum_{i=1}^{P} \alpha_i d_i + \sum_{i=1}^{P} \alpha_i \right] \\ &= \mathbf{w} - \sum_{i=1}^{P} \alpha_i \mathbf{x}_i d_i. \end{aligned} \tag{2.15}$$

Hence,

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Leftrightarrow \mathbf{w} = \sum_{i=1}^{P} \alpha_i \mathbf{x}_i d_i. \tag{2.16}$$

Replacing this condition for $\mathbf{w}$ in the Lagrangian yields

$$L = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T \sum_{i=1}^{P} \alpha_i \mathbf{x}_i d_i - b \sum_{i=1}^{P} \alpha_i d_i + \sum_{i=1}^{P} \alpha_i$$

[1] In order to avoid confusion between the primal and dual problems, in optimization literature the Lagrangian is sometimes denoted by L_P or L_D according to the problem being solved. This notation is not used in this document.

$$\begin{aligned}
&= \frac{1}{2}\sum_{k=1}^{P}\alpha_k \mathbf{x}_k^T d_k \sum_{i=1}^{P}\alpha_i \mathbf{x}_i d_i - \sum_{k=1}^{P}\alpha_k \mathbf{x}_k^T d_k \sum_{i=1}^{P}\alpha_i \mathbf{x}_i d_i \\
&\quad -b\sum_{i=1}^{P}\alpha_i d_i + \sum_{i=1}^{P}\alpha_i \\
&= -\frac{1}{2}\sum_{k=1}^{P}\sum_{i=1}^{P}\alpha_k \mathbf{x}_k^T \mathbf{x}_i d_k d_i \alpha_i - b\sum_{i=1}^{P}\alpha_i d_i + \sum_{i=1}^{P}\alpha_i. \qquad (2.17)
\end{aligned}$$

On the other hand, the derivative of L with respect to b is given by

$$\begin{aligned}
\frac{\partial L}{\partial b} &= \frac{\partial}{\partial b}\left[\frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T\sum_{i=1}^{P}\alpha_i \mathbf{x}_i d_i - b\sum_{i=1}^{P}\alpha_i d_i + \sum_{i=1}^{P}\alpha_i\right] \\
&= -\sum_{i=1}^{P}\alpha_i d_i. \qquad (2.18)
\end{aligned}$$

Therefore,

$$\frac{\partial L}{\partial b} = 0 \Leftrightarrow \sum_{i=1}^{P}\alpha_i d_i = 0. \qquad (2.19)$$

Using this condition in (2.17), yields

$$L = -\frac{1}{2}\sum_{k=1}^{P}\sum_{i=1}^{P}\alpha_k \mathbf{x}_k^T \mathbf{x}_i d_k d_i \alpha_i + \sum_{i=1}^{P}\alpha_i. \qquad (2.20)$$

Note that the first term in (2.20) is a quadratic form in the coefficients α_i. Defining the vector

$$\overline{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha]^T \qquad (2.21)$$

and the P-dimensional unity vector $\mathbf{u}$

$$\mathbf{u} = [\mathbf{1}, \mathbf{1}, \mathbf{1}, \cdots, \mathbf{1}]^{\mathbf{T}} \qquad (2.22)$$

(2.20) can be re-written as

$$L = -\frac{1}{2}\overline{\alpha}^T \mathbf{H}\overline{\alpha} + \mathbf{u}^{\mathbf{T}}\overline{\alpha} \qquad (2.23)$$

where $\mathbf{H}$ is a $P \times P$ matrix whose elements h_{ij} are given by

$$h_{ij} = \mathbf{x}_k^T \mathbf{x}_i d_k d_i. \qquad (2.24)$$

The dual formulation enables to restate the optimization problem in the following simplified form: minimize

$$L = -\frac{1}{2}\overline{\alpha}^T \mathbf{H}\overline{\alpha} + \mathbf{u}^{\mathbf{T}}\overline{\alpha} \qquad (2.25)$$

subject to

$$\sum_{i=1}^{P} \alpha_i d_i = 0 \tag{2.26}$$

$$\alpha_i \geq 0, \quad i = 1, \cdots, P \tag{2.27}$$

Note that $\mathbf{w}$ does not appear explicitly in the dual formulation and, moreover, the input patterns only appear in the optimization functional in dot products of the form $\mathbf{x}_k^T \mathbf{x}_i$. As it will be seen in chapter 4, this property is essential for the use of kernels in support vector machines.

2.1.2 Analysis

For the analysis of the solutions found by the optimization procedure is convenient to make use of the Kuhn-Tucker conditions, which are observed at the solution of any convex optimization problem.

Instead of recalling the whole set of Kuhn-Tucker conditions, we will only consider a particular relevant one for the analysis of SVM. This condition states that, for constraints of the form $c_i \geq 0$, one must have $c_i \alpha_i = 0$ at the solution. Note that this is a quite natural condition. In fact, it just means that either $c_i > 0$ at the solution, and in this case the constraint c_i is irrelevant for the specific solution found, and hence $\alpha_i = 0$, or, alternatively, $c_i = 0$ and, in this case, c_i imposes a bound on the solution that implies the intervention of a non null multiplier α_i.

In the SVM case, the Kuhn-Tucker condition means that, at the solution, one must have

$$\alpha_i \left[\left(\mathbf{w}^T \mathbf{x}_i + b \right) d_i - 1 \right] = 0. \tag{2.28}$$

This condition implies that either $\alpha_i = 0$ or, alternatively, that[2]

$$\mathbf{w}^T \mathbf{x}_i + b = d_i \tag{2.29}$$

After the optimization, $\alpha_i = 0$ means that the constraint i can be eliminated from the Lagrangian without affecting the final solution. This means that if $\alpha_i = 0$ than pattern i is irrelevant for the definition of $(\mathbf{w}, b)$. Therefore, it is clear that only those patterns for which $\alpha_i > 0$ contribute for the definition of $\mathcal{P}$. The patterns in the set

$$\mathcal{S} = \{\mathbf{x}_i : (i : \alpha_i > 0)\} \tag{2.30}$$

[2]Recall that since $d_i \in \{-1, 1\}$, $d_i = 1/d_i$

are called the *support vectors* of the support vector machine.

Note that, given (2.29), one have

$$|f(\mathbf{x}_i)| = 1, \quad \forall \mathbf{x}_i \in \mathcal{S} \tag{2.31}$$

and, therefore, all support vectors correspond to the minimum classification margin and lay on the two hyper-planes symmetrical to the decision boundary that define the optimal classification margin.

2.1.3 Classifi er parameters

In the dual formulation, only the Lagrange multipliers α_i of the primal problem are computed. However, the classifier parameters are $\mathbf{w}$ and b. But these can be derived from (2.16), resulting

$$\mathbf{w} = \sum_{i=1}^{P} \alpha_i \mathbf{x}_i d_i. \tag{2.32}$$

On the other hand b can be computed from any support vector applying equation (2.29). In practice, it is usually better to compute b from the average of the results obtained from all support vectors. If N_s is the number of support vectors, and denoting by $\{x_{s(k)}, d_{s(k)}\}$ the k-th support vector and the corresponding class, one may compute b from

$$b = \frac{1}{N_s} \sum_{k=1}^{N_s} \left(d_{s(k)} - \mathbf{w}^T \mathbf{x}_{s(k)} \right) \tag{2.33}$$

Since only the support vectors have $\alpha_i > 0$, the computation of $\mathbf{w}$ in (2.32) can also be performed as a function of these samples only. Therefore, one may rewrite (2.32) as

$$\mathbf{w} = \sum_{k=1}^{N_s} \alpha_{s(k)} d_{s(k)} \mathbf{x}_{s(k)} \tag{2.34}$$

where $\alpha_{s(k)}$ denotes the multiplier for the k-th support vector.

A simple example of a bi-dimensional distribution and the corresponding classification region found by a SVM is depicted in figure 2.2. In this figure each support vector appears enclosed in a circle and the intermediate gray regions correspond to the classification margin where $||\mathbf{w}^T\mathbf{x} - b|| < 1$. Note that only 2 support vectors are used in this case. The classification region found by a 2×1 sigmoidal unit for the same input distribution is depicted in figure 2.3.

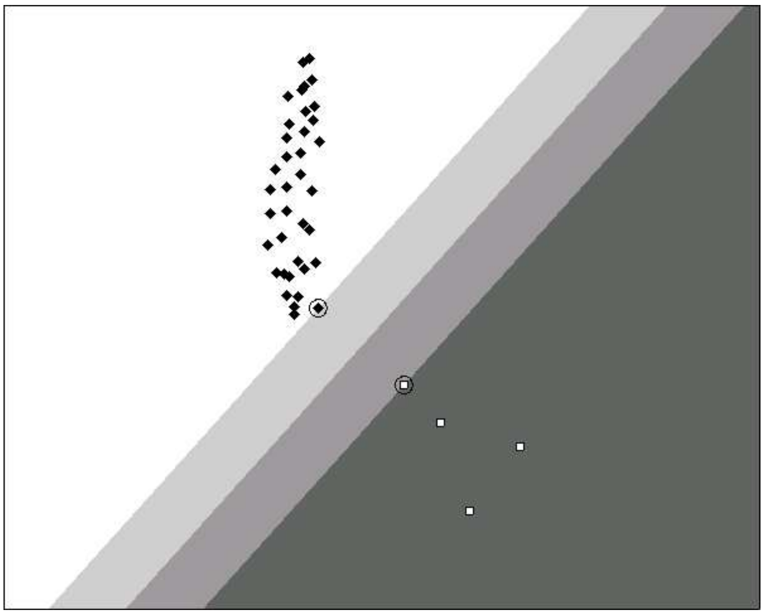

Figure 2.2: Binary classification region found by a SVM in a simple bi-dimensional distribution. $P = 41$ and $N_s = 2$.

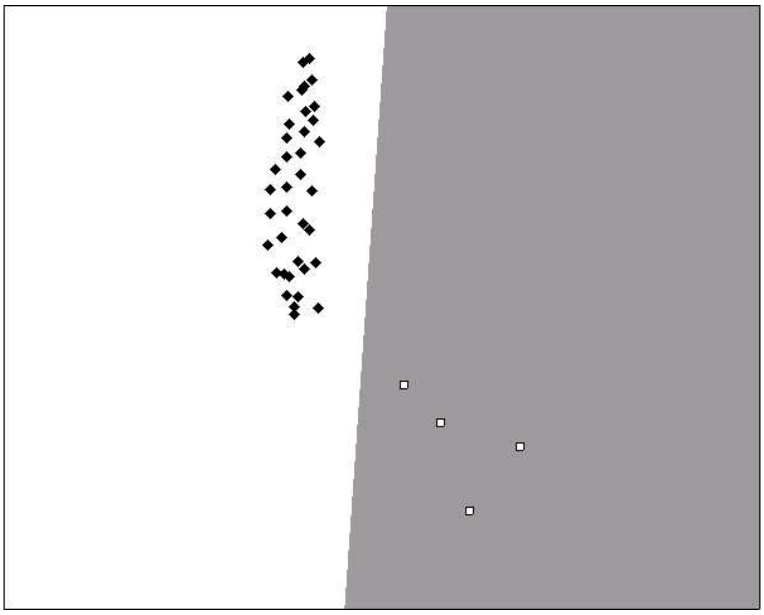

Figure 2.3: Binary classification region found by a simple sigmoidal unit in the same distribution of fig. 2.2

.

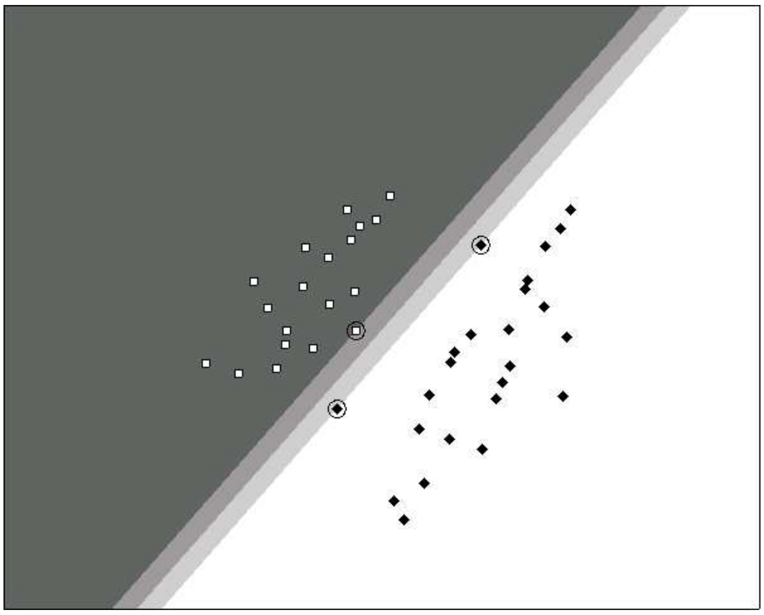

Figure 2.4: .
Binary classifi cation region found by a SVM. $P = 43$ and $N_s = 3$.

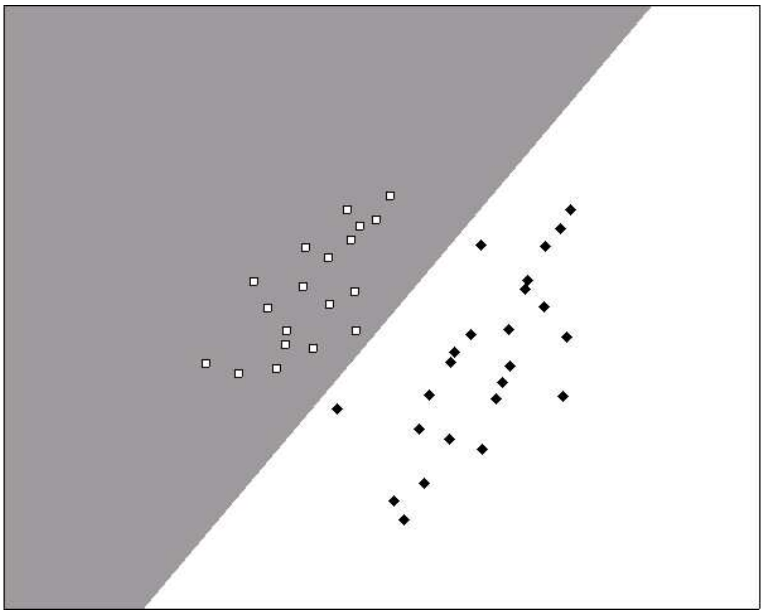

Figure 2.5: Binary classification region found by a sigmoidal unit in the same distribution of fig. 2.4
.

A second example where the structure of the distribution leads to three support vectors is depicted in figures 2.4. Note that in the linear separable case, if there is some degree of randomness in the input distribution the SVM is almost always defined by two or three support vectors. The solution found by a sigmoidal unit for the same input distribution is depicted and 2.5.

2.2 The non separable case

2.2.1 Misclassification margin

The above analysis assumes that all training samples can be correctly classified by a single decision hyper-plane. However, this is seldom the case. Even assuming that the structure and nature of the problem is purely linear, it is necessary to take into account that training samples often result from observations of (noisy) real world data, yielding training data often populated by misclassified, ambiguous or outlier samples. In these cases, the constrained optimization problem outlined before does not converge, since it is not possible to satisfy the classification constraints (equation (2.9)). In such cases, the optimization process will result in an increasing and unbounded sequence of Lagrange multipliers for those samples that can not be correctly classified.

A simple solution to this problem was proposed in (Cortes and Vapnik, 1995). The idea is to soft the constraints defined by (2.9) such that some of patterns are allowed to lay inside the classification margin ($|(f(\mathbf{x})| < 1$) or even in the wrong side of the classification boundary. This is done by including in the classification constraints *slack variables* which create room for a *misclassification margin*. More specifically, for each pattern i in the training set is introduced a slack variable ξ_i and the constraints (2.9) are re-written as

$$(\mathbf{w}^T\mathbf{x}_i + b)d_i \geq 1 - \xi_i, \quad i = 1, \cdots, P \tag{2.35}$$

$$\xi_i \geq 0, \quad i = 1, \cdots, P. \tag{2.36}$$

Of course, the optimization problem must now provide some way of computing also the new parameters ξ_i. Desirably, one would like that all ξ_i vanish or, if this is not possible, that their values get as small as possible. One way to accomplish this result is to include in the objective function a penalty term to weight and minimize the contribution of the ξ_i parameters. In these conditions, the objective function defined by (2.10) is rewritten as

$$f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{P}\xi_i \tag{2.37}$$

where C is a positive constant that weights the penalty term. Now the optimization problem must be re-stated as follows: minimize (2.37) subject to constraints (2.35) and (2.36).

2.2.2 Optimization procedure

As before, the constrained minimization of (2.37) can be performed by the minimization of the Lagrangian

$$\begin{aligned} L &= \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{P}\xi_i - \sum_{i=1}^{P}\alpha_i\left((\mathbf{w}^T\mathbf{x}_i + b)d_i - 1 + \xi_i\right) - \sum_{i=1}^{P}\mu_i\xi_i \\ &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T\sum_{i=1}^{P}\alpha_i\mathbf{x}_i d_i - b\sum_{i=1}^{P}\alpha_i d_i + \sum_{i=1}^{P}\alpha_i \\ &\quad + \sum_{i=1}^{P}\xi_i\left(C - \alpha_i - \mu_i\right) \end{aligned} \tag{2.38}$$

where the additional Lagrange multipliers μ_i were introduced in order to support the additional constraints on the slack variables ($\xi_i \geq 0$).

At this point, we may restate the formulation of the primal and dual problems. The primal problem corresponds to the solution of

$$\min_{\mathbf{w},b,\xi_i} L \;\Bigg|\; \left\{\left(\frac{\partial L}{\partial \alpha_i} = 0;\;\; \frac{\partial L}{\partial \mu_i} = 0;\;\; \alpha_i \geq 0;\;\; \mu_i \geq 0\right),\;\; i = 1, \cdots, P\right\} \tag{2.39}$$

while the Wolf dual is now

$$\max_{\alpha_i,\mu_i} L \;\Bigg|\; \left\{\frac{\partial L}{\partial \mathbf{w}} = 0;\;\; \frac{\partial L}{\partial b} = 0;\;\; \left(\frac{\partial L}{\partial \xi_i} = 0;\;\; \alpha_i \geq 0;\;\; \mu_i \geq 0\right),\;\; i = 1, \cdots, P\right\}. \tag{2.40}$$

The partial derivatives of L in respect to $\mathbf{w}$, b and ξ_i can be easily computed. From (2.38), it becomes clear that the partial derivatives in respect to $\mathbf{w}$ and b are the same as in the separable case, therefore also providing conditions (2.16) and (2.19). It remains the derivative of L with respect to ξ_i. This derivative can be computed as

$$\begin{aligned} \frac{\partial L}{\partial \xi_k} &= \frac{\partial}{\partial \xi_k}\left[\frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T\sum_{i=1}^{P}\alpha_i\mathbf{x}_i d_i - b\sum_{i=1}^{P}\alpha_i d_i + \sum_{i=1}^{P}\alpha_i \right. \\ &\qquad \left. + \sum_{i=1}^{P}\xi_i\left(C - \alpha_i - \mu_i\right)\right] \\ &= C - \alpha_k - \mu_k \end{aligned} \tag{2.41}$$

and, therefore,

$$\frac{\partial L}{\partial \xi_i} = 0 \Leftrightarrow \mu_i = C - \alpha_i \tag{2.42}$$

Replacement of this result, (2.16) and (2.19) in (2.38) makes all terms in ξ_i to vanish and results again in exactly the same objective function defined by (2.25). But one additional constraint must be included in the optimization procedure. In fact, the Lagrangian multipliers μ_i do not appear in the simplified objective function since they are implicitly defined by (2.42). However, since one must have $\mu_i \geq 0$, this implies that

$$\alpha_i \leq C, \;\; i = 1, \cdots, P \tag{2.43}$$

and therefore this additional constrain must be included in optimization process.

In synthesis, the optimization problem in the non separable case can now be restated as follows: minimize

$$L = -\frac{1}{2}\overline{\alpha}^T \mathbf{H} \overline{\alpha} + \mathbf{u^T}\overline{\alpha} \tag{2.44}$$

subject to

$$\sum_{i=1}^{P} \alpha_i d_i = 0 \tag{2.45}$$

$$0 \leq \alpha_i \leq C, \quad i = 1, \cdots, P \tag{2.46}$$

2.2.3 Analysis

The analysis of the Kuhn-Tucker conditions in the non-linear case provides some further insight concerning the role and effect of the C parameters in the optimization procedure. Further to the conditions for the α_i multipliers, there are now the conditions that relate the constraints $\xi_i \geq 0$ and the multipliers μ_i. At the solution, one must have $\xi_i \mu_i = 0$. This means that one of two case may occur:

- If the slack variables ξ_i vanish, the corresponding samples are correctly classified, being located at the border or outside the classification margin. For these samples, one have

  $$\mu_i = C - \alpha_i. \tag{2.47}$$

  If, furthermore, these samples are far from the border, one will have $\alpha_i = 0$ and, therefore, one will have $\mu_i = C$.

- If $\mu_i = 0$, than $\xi_i > 0$. This means that the optimization process needed to make use of the slack margin ξ_i. In this case, the sample $\mathbf{x}_i$ is either outside the classification margin (if $0 < \xi \leq 1$) or misclassified (if $\xi > 1$). In all these cases, the Kuhn-Tucker condition imposes that $\mu_i = 0$ and, therefore, that $\alpha_i = C$. This means that a sample for which α_i reaches the upper bound C on the optimization process corresponds to a misclassified sample or, at least, an *out of margin* sample (in the sense that $||\mathbf{w}^T\mathbf{x} + b|| < 1$). Note that all samples in these conditions will correspond to support vectors of the classifier.

2.2.4 Classifier parameters

Let us consider again the problem of computing the classifier parameters $\mathbf{w}$ and b.

As before, $\mathbf{w}$ can be computed from the support vectors by (2.34). However, for the computation of b the Kuhn-Tucker conditions must be again revisited. Note that with the introduction of the slack variables, the condition (2.28) at the solution is now expressed as

$$\alpha_i \left[\left(\mathbf{w}^T \mathbf{x}_i + b \right) d_i - 1 + \xi_i \right] = 0. \tag{2.48}$$

Since the slack variables ξ_i are not explicitly computed, b must now be computed from $\mathbf{w}$ and any support vector $\underline{\text{such that } \xi_i = 0}$. As before, it is usually safer to use the average that results from all support vectors that verify such conditions. Therefore, in the non separable case, the average defined by (2.33) must exclude, all support vectors for which $\alpha_i = C$.

It remains the problem of choosing a good value of C for a given classification problem. Unhappily, the optimal value of C is strongly dependent on the problem under analysis and, specifically, on the noise, ambiguities or / and outliers that occur in the training set. In practice, C is usually fixed by a trial and error procedure.

2.2.5 Experimental results

Figures 2.6 and 2.7 contribute to a better understanding of the effect of different values of C in a given optimization problem. Both figures refer to the same non-linearly separable classification problem. In the captions, N_e denotes the number of misclassified samples and N_{om} the number of *out of margin* samples. As it can be seen, a lager value of C contributes to reduce the number of samples for which $\xi_i > 0$. This suggests that, in practice, one would like to have C as large as possible, in order to try to reduce the effect of outliers and misclassified samples. However, $C = \infty$ corresponds to do not impose any upper bound on α_i and, as it was already discussed, such solution only exists if the problem is linearly separable. Therefore, in practice, it is necessary to find a trade-off between a large C and a numerically feasible solution of the optimization problem.

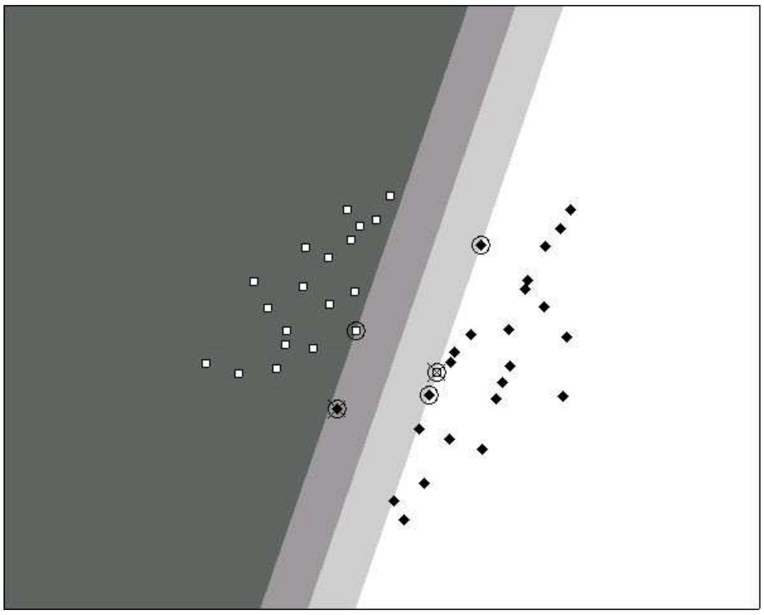

Figure 2.6: Non-linearly separable problem. Classification regions found by a SVM with $C = 1000$ ($P = 44$, $N_s = 5$, $N_e = 2$, $N_{om} = 1$).

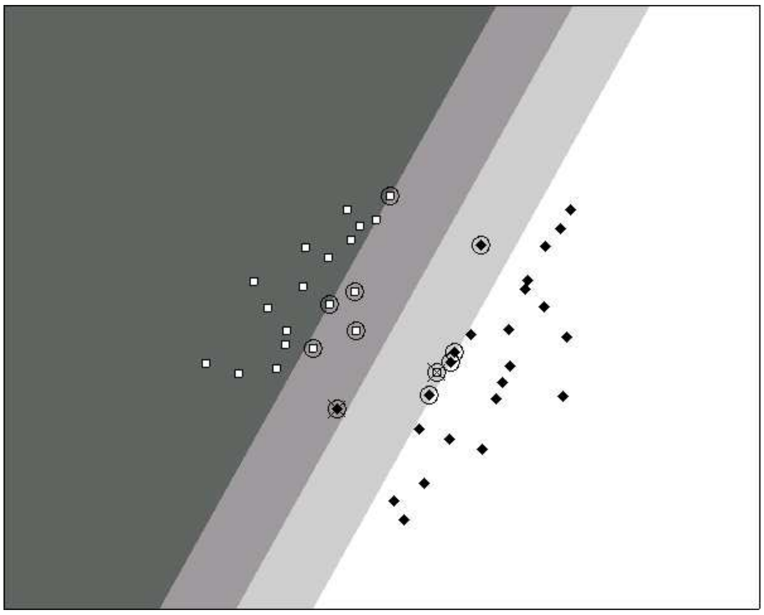

Figure 2.7: Same distribution of figure 2.6. Classification regions found by a SVM with $C = 1$ ($P = 44$, $N_s = 5$, $N_e = 2$, $N_{om} = 8$).

Chapter 3

Non linear classifi cation

3.1 Feature expansion

As described before, support vector machines can only be applied to linear classification problems. However, the vast majority of real world classification problems are in fact non-linear by nature. Hopefully, the SVM framework was elegantly extended to non-linear problems.

One conventional solution to solve non-linear classification problems with linear classifiers is to generate intermediate features from the input data such that the input patterns become linearly separable in the feature space. In most cases, this result can be achieved performing simple but adequate non-linear transformations of the input data. This is the basic strategy adopted, for example, by multilayer neural networks or multilayer perceptron (MLP). In any MLP, the output layer acts as a linear classifier. However, the pre-processing of the input by one or more non-linear hidden layers may perform a transformation of the input data such that the output layer work on a linearly separable feature space. This basic idea behind MLP networks can be traced back to the work of Minsky and Papert (Minsky and Papert, 1969).

In multilayer networks, the transformation performed by the front-end layers is is usually optimized by an error minimization procedure (usually backpropagation), which eases the task of finding an intermediate transformation of the input data such that the output layer may perform on a linearly separable feature space. An equivalent procedure to find an optimized transformation of the input data for SVM was not found yet. However, for some classes of problems, one may expect that a (fixed) non-linear projection of the input space in an higher dimension space $\mathcal{H}$ may generate a linearly separable feature space.

More formally, given a non linear separable training set $\mathcal{T}$, we assume that it is possible to

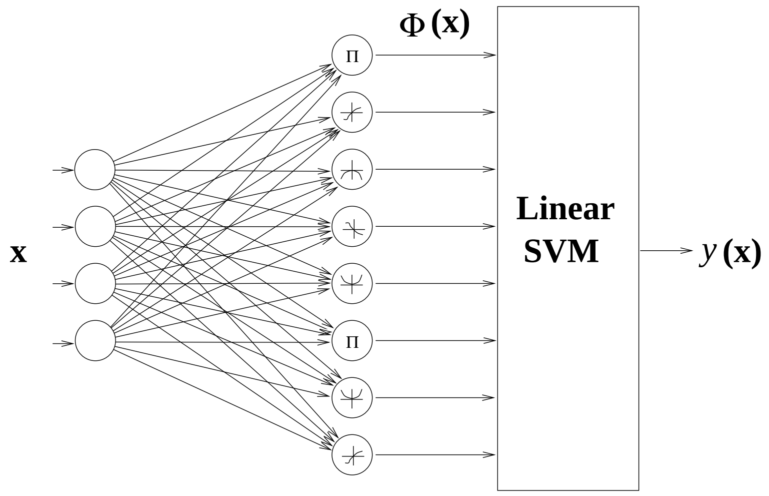

Figure 3.1: Extension of SVM to non linear problems through feature expansion.

find a non linear transformation $\Phi(\mathbf{x})$ of $\mathbf{R}^N$ in $\mathbf{R}^M$ (possibly with $M > N$) such that the set

$$\mathcal{T}' = \{(\Phi(\mathbf{x}_i), d_i)\,,\ \ i = 1, \cdots, P\}. \tag{3.1}$$

is linearly separable. It is clear that if such transformation is found, all the SVM theory may be applied to the feature space $\mathcal{H}$ where $\Phi(\mathbf{x})$ lives, therefore providing a basis for an optimal non-linear classification. This method is graphically depicted in figure 3.1.

3.2 Orthonormalized polynomial feature expansion

In order to test this feature expansion approach on a non-linear classification problem we considered the bi-dimensional classification problem represented in figure 3.2. The classes layout correspond to a *chess board* pattern, thus resulting in a reasonably hard classification problem.

In order to solve this problem with a SVM, several classification tests based on feature spaces of different dimensions were performed. In this group of tests, only polynomial transformations of the input data were performed. More specifically, for each polynomial order p, the feature space $\mathcal{H}$ was build from all features resulting from products of $\mathbf{x}$ components of total order lower or equal to p. For example, denoting by $\Phi^p(\mathbf{x})$ the expansion of order p and by x_i the i-th component of $\mathbf{x}$, the following transformations were used for of $p = 1, \cdots, 3$:

- $p = 1$.

$$\Phi^1(\mathbf{x}) = [x_1, x_2]^T. \tag{3.2}$$

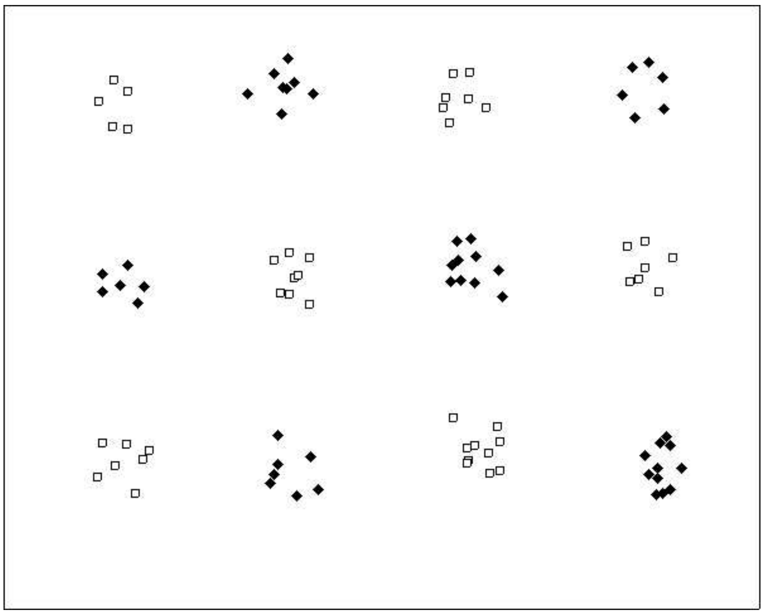

Figure 3.2: Non linear *chess board* type classification problem. $P = 92$.

- $p = 2$.

$$\Phi^2(\mathbf{x}) = [x_1, x_2, x_1^2, x_1x_2, x_2^2]^T. \tag{3.3}$$

- $p = 3$.

$$\Phi^3(\mathbf{x}) = [x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3]^T. \tag{3.4}$$

Note that this feature expansion yields, for order p, an output space $\mathcal{H}$ of dimension

$$M = \mathrm{Dim}(\mathcal{H}) = \sum_{k=1}^{p} (k+1) \tag{3.5}$$

and, therefore, the dimension of $\mathcal{H}$ increases considerably with p. On the other extreme, if $p = 1$ the SVM reduces to the linear classifier considered before.

Note that the components of $\Phi(\mathbf{x})$ result from different functions of only two components (x_1 and x_2) and, therefore, they are usually highly correlated. Moreover, the individual components of $\Phi(\mathbf{x})$ have quite different variances depending on the order of each output feature. Due to these facts, it is usually difficult to perform a successful optimization procedure directly on the feature space $\Phi(\mathbf{x})$. In order to avoid these problems, it was performed an equalization of the feature space by making

$$\Phi_o(\mathbf{x}) = \mathbf{T}(\Phi(\mathbf{x}) - \Phi_m) \tag{3.6}$$

where $\Phi_m = E[\Phi(\mathbf{x})]$ denotes the ensemble mean of $\Phi(\mathbf{x})$. $\mathbf{T}$ is a $M \times M$ transformation matrix defined as

$$\mathbf{T} = \mathbf{R}_{\Phi}^{-\frac{1}{2}} \tag{3.7}$$

where $\mathbf{R}_{\Phi}$ is the covariance matrix of $\Phi(\mathbf{x})$:

$$\mathbf{R}_{\Phi_m} = E\left[\left(\Phi(\mathbf{x}) - \Phi_m\right)\left(\Phi(\mathbf{x}) - \Phi_m\right)^T\right]. \tag{3.8}$$

Note that with this transformation, the covariance matrix of the final features $\Phi_o(\mathbf{x})$ is

$$\mathbf{R}_{\Phi_o} = \mathbf{I} \tag{3.9}$$

and, therefore, the feature space becomes orthonormalized. It was observed that this equalization of the feature space yielded a much easier SVM optimization procedure.

3.3 Test results

In order to assess the properties of polynomial features of different orders, a first set of classification tests was performed with $p = 1, \cdots, 5$. All tests were performed with $C = 100$. The results of this first set of tests is depicted on figures 3.3 to 3.7. As it would be expected, as p (and hence Dim($\mathcal{H}$)) increases, a more elaborate boundary line is build by the SVM operating on $\mathcal{H}$. As a result, the number of misclassified samples decreases with the increase of p and a correct classification of all training patterns is reached for $p = 5$.

Since a perfect classification is found for $p = 5$, one may question what happens for higher values of p. In order to answer this question, we repeated the classification tests with increasing values of p. Figures 3.8 and 3.9 show the result of these tests for $p = 11$ ($\mathrm{Dim}(\mathcal{H}) = 77$) and $p = 13$ ($\mathrm{Dim}(\mathcal{H}) = 104$). As it can be observed, increasing p beyond $p = 5$ yields an increasing complex boundary line. In fact, a too high dimension of $\mathcal{H}$ only contributes for the uniformization of the distances of each feature vector to $\mathcal{P}$ (note that this is an expected consequence of the topological properties of high dimensional spaces).

For $p = 13$ and above, the SVM optimization procedure converges for solutions in which all samples lie in the boundary of the classification margin ($|f(\mathbf{x}_i)| = 1, \forall i$) and, therefore, all data patterns become support vectors. These solutions correspond to increasingly fragmented classification regions and the regularization effect of SVM is, in these cases, significantly lower. Note that for $p \geq 13$ the dimensionality of the feature space becomes higher than the number of training samples ($\mathrm{Dim}(\mathcal{H}) > \mathcal{P}$). This can be a a possible justification for the type of solution found by the optimization procedure.

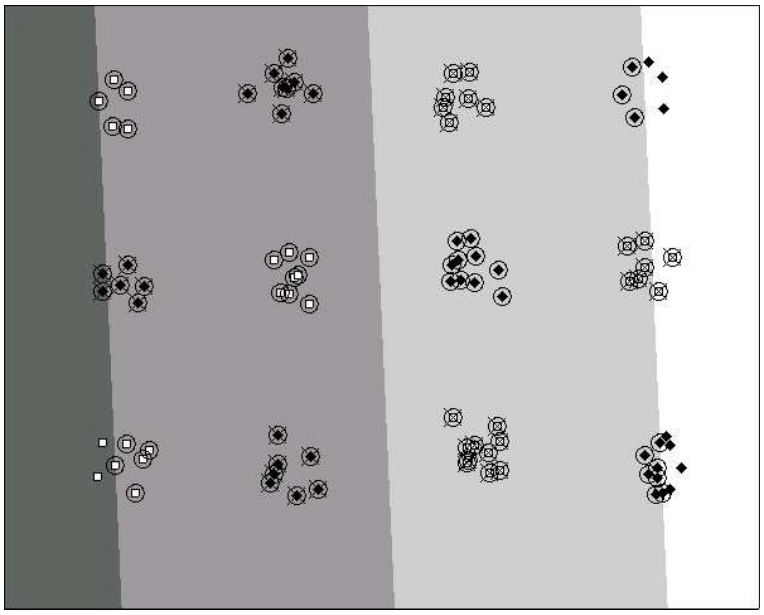

Figure 3.3: SVM classification of the *chess board* distribution of figure 3.2. Linear case ($p = 1$). $\mathrm{Dim}(\mathcal{H}) = 2$. $N_s = 83$. $N_e = 45$. $N_{om} = 75$.

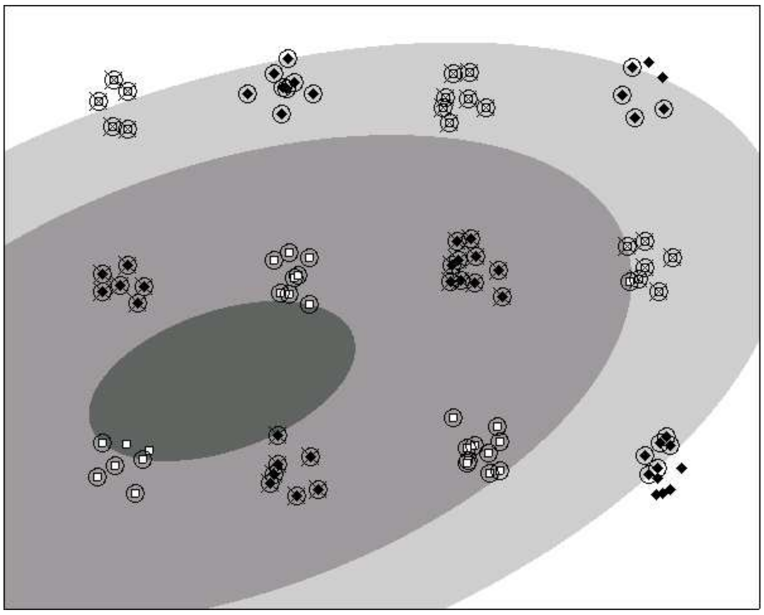

Figure 3.4: SVM classification of the *chess board* distribution of figure 3.2. Polynomial feature expansion with $p = 2$. $\mathrm{Dim}(\mathcal{H}) = 2$. $N_s = 83$. $N_e = 45$. $N_{om} = 75$.

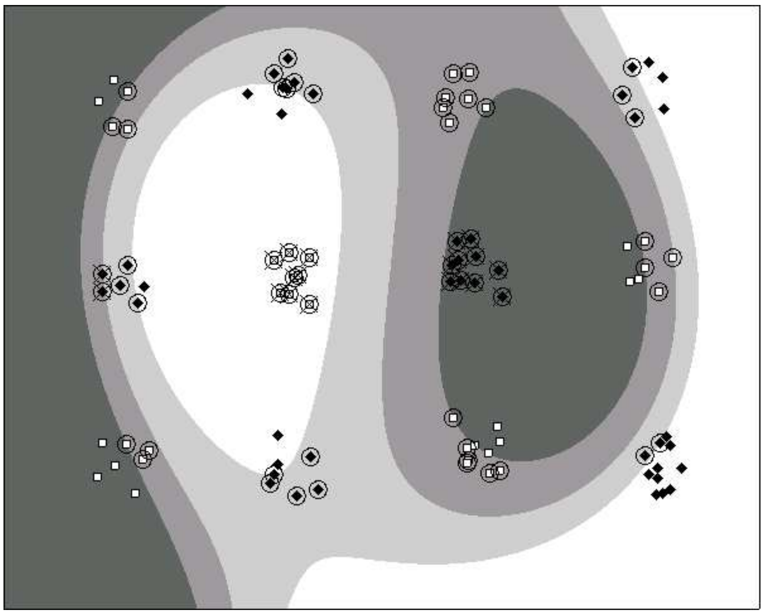

Figure 3.5: SVM classification of the *chess board* distribution of figure 3.2. Polynomial feature expansion with $p = 3$. $\mathrm{Dim}(\mathcal{H}) = 9$. $N_s = 62$. $N_e = 20$. $N_{om} = 33$.

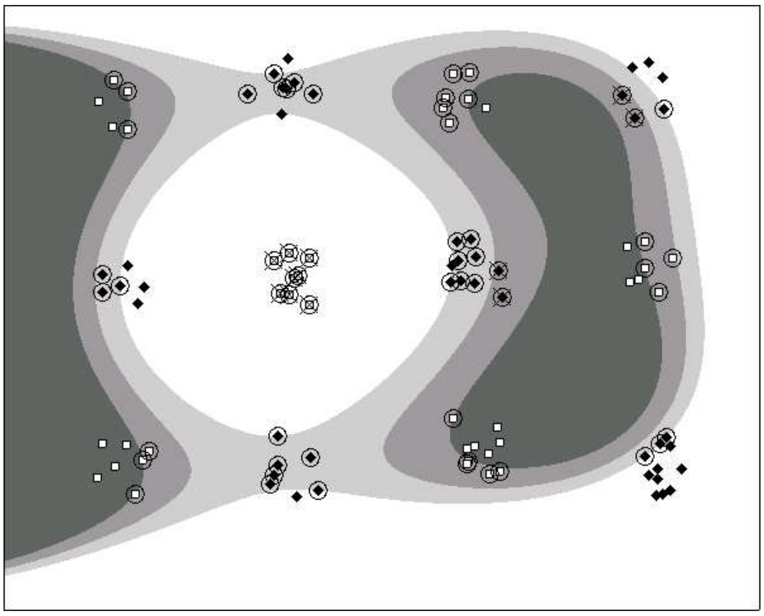

Figure 3.6: SVM classification of the *chess board* distribution of figure 3.2. Polynomial feature expansion with $p = 4$. $\mathrm{Dim}(\mathcal{H}) = 14$. $N_s = 59$. $N_e = 12. N_{om} = 35$.

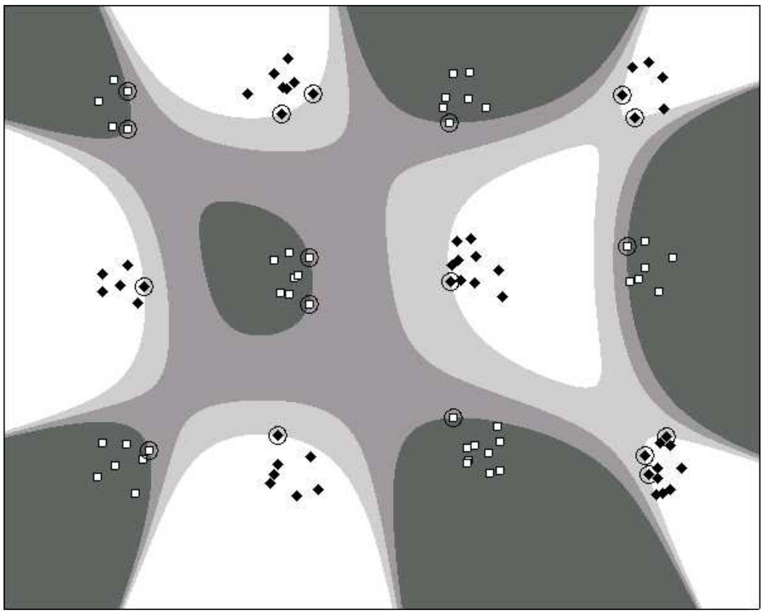

Figure 3.7: SVM classification of the *chess board* distribution of figure 3.2. Polynomial feature expansion with $p = 5$. $\mathrm{Dim}(\mathcal{H}) = 20$. $N_s = 18$. $N_e = 0$. $N_{om} = 0$.

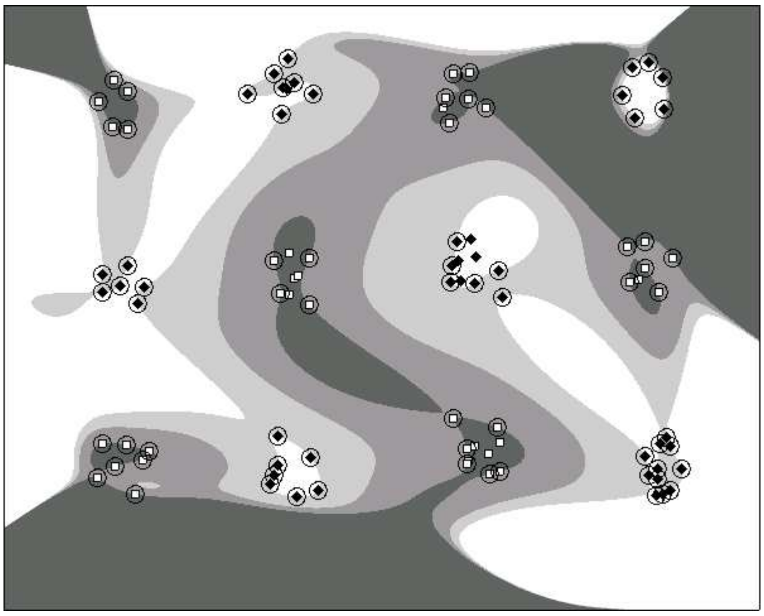

Figure 3.8: SVM classification of the *chess board* distribution of figure 3.2. Polynomial feature expansion with $p = 11$. $\mathrm{Dim}(\mathcal{H}) = 77$. $N_s = 77$. $N_e = 0$. $N_{om} = 0$.

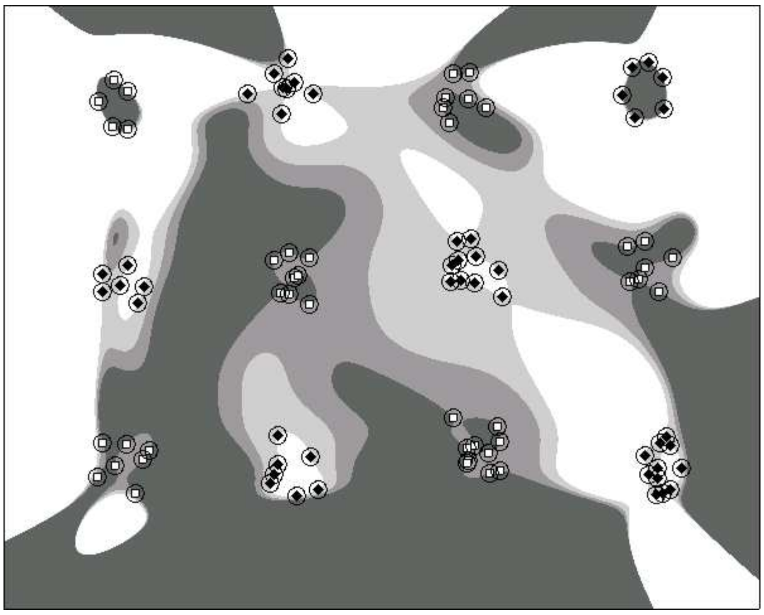

Figure 3.9: SVM classification of the *chess board* distribution of figure 3.2. Polynomial feature expansion with $p = 13$. $\mathrm{Dim}(\mathcal{H}) = 104$. $N_s = 92$. $N_e = 0$. $N_{om} = 0$.

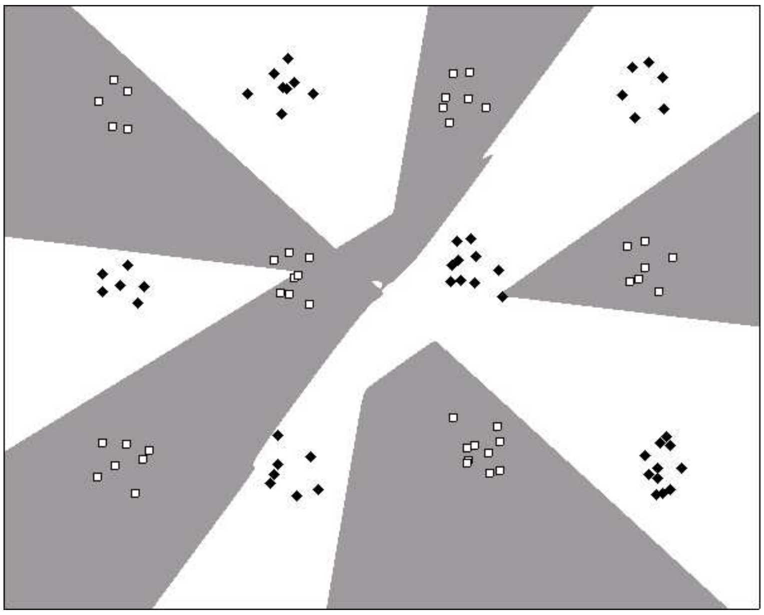

Figure 3.10: MLP classification of the *chess board* distribution of figure 3.2. $2 \times 8 \times 1$ network. $N_e = 0$.

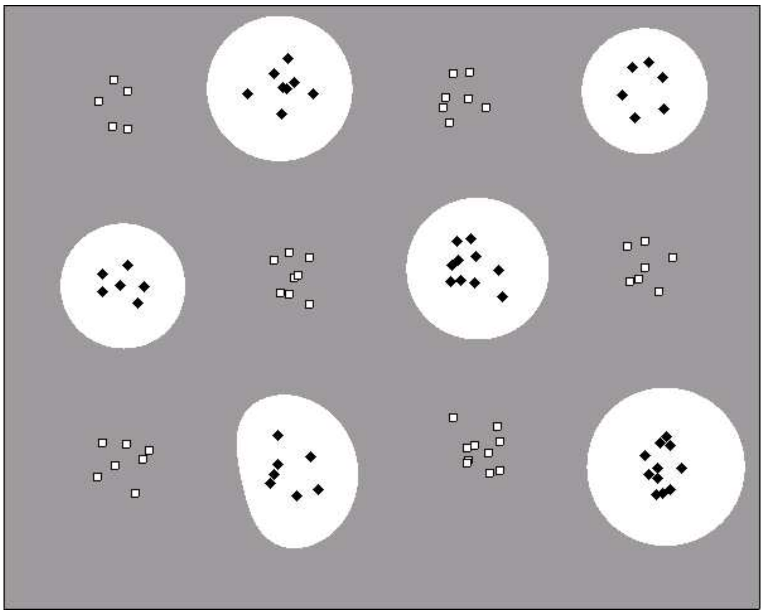

Figure 3.11: RBF classification of the *chess board* distribution of figure 3.2. network with 8 input gaussians. $N_e = 0$.

In order to compare these results with those found by conventional classifiers, the same classification problem was solved using a $2 \times 8 \times 1$ MLP and a radial basis function (RBF) network with 8 input gaussians. The classification regions found in each case are depicted in figures 3.10 and 3.11. Note that both networks share the same number of free parameters ($W = 33$), but the particular choice of 8 hidden units was rather arbitrary at this stage. As it would be expected, the solution found by the MLP network is based on a segmentation of the input space with hyper-planes such that all training samples are correctly classified. Apparently, the particular segmentation found by the MLP is quite arbitrary, and the solution depicted on figure 3.10 will hardly provide a reasonable generalization behavior (but it must be emphasized that input distributions of this type, punctuated by local data clusters, are among the most difficult problems for MLP classifiers). On the other hand, it becomes clear that the RBF network provides here a quite simple and natural result, based on centering one (or more) gaussians in each input cluster of one of the two classes [1]. Apparently, this result will provide a better generalization behavior than the one obtained by the MLP network. But, in any case, the decision boundary build by the SVM with $p = 5$ seems a much better match of the classification regions that one would heuristically expect to be associated with the structure of this classification problem.

At this stage, one remark must be done regarding the tests performed through all this docu-

[1]Note that the desired value of the white square patterns was set to 0 for the RBF networks, while the classifi cation threshold was set to 0.5. This strategy simplifi es considerably the RBF task. On the other hand, it must be considered that if the desired values of the target classes are interchanged, the solution found is based on gaussians centered on the complementary class, therefore yielding a rather different decision boundary.

ment. Since our aim is to perform a qualitative comparison of typical solutions found by different classifiers under similar conditions, it was not used any *test set* to provide a criteria for early stopping of the learning procedure in RBF and MLP networks. It is well known that early stopping may remarkably improve the generalization ability of MLP networks, namely when the number of free parameters is close to the number of training patterns (as it happens in the majority of the tests described herein). This means that the solutions found by MLP networks presented in this document are certainly punctuated by over-learning. But it must be considered that our goal is also to assess the regularization effect of the learning criteria itself. In this sense, all tests were performed under similar and reasonably fair conditions.

Chapter 4

Non linear SVM and kernels

4.1 Definition

As it was discussed in chapter 3, the feature expansion approach requires the explicit translation of the training patterns from a (usually) low dimensional space $\mathcal{L}$ to an (usually) much higher dimensional space $\mathcal{H}$ were the problem is linearly separable. In this section, we discuss how the theory of Kernels may avoid the explicit computation of the patterns in the high dimensional space $\mathcal{H}$.

In chapter 2 it was shown how SVM classification can be reduced to a constrained optimization on the Lagrangian multipliers α_i described by equations (2.44), (2.45) and (2.46). When this problem is carried out on the $\mathcal{H}$ space, it suffices to replace the definition (2.24) of the elements of the $\mathbf{H}$ matrix by

$$h_{ij} = \Phi^T\left(\mathbf{x}_i\right)\Phi\left(\mathbf{x}_k\right)d_i d_k. \tag{4.1}$$

Therefore, the optimization problem itself only requires the computation of the dot products $\Phi^T\left(\mathbf{x}_k\right)\Phi\left(\mathbf{x}_i\right)$. Now, assume that we define this dot product as direct function of $\mathbf{x}_i$ and $\mathbf{x}_k$. Or, by other words, that we define a function $K(\mathbf{x}_i, \mathbf{x}_k)$ (called kernel function) such that

$$K(\mathbf{x}_i, \mathbf{x}_k) = \Phi^T\left(\mathbf{x}_i\right)\Phi\left(\mathbf{x}_k\right). \tag{4.2}$$

If $K(\mathbf{x}_i, \mathbf{x}_k)$ can be directly computed from $\mathbf{x}_i$ and $\mathbf{x}_k$, we avoid the explicit computation of the features $\Phi(\mathbf{x}_i)$. In this case, the elements of the matrix $\mathbf{H}$ can be simply defined as

$$h_{ij} = K(\mathbf{x}_i, \mathbf{x}_k)d_i d_k. \tag{4.3}$$

In the more general case, one may even define directly the kernel $K(\mathbf{x}_i, \mathbf{x}_k)$ without the explicit definition of $\Phi(\mathbf{x})$. Of course, some care must be taken when choosing kernel functions. A valid kernel requires that some transformation of $\mathcal{L}$ into $\mathcal{H}$ exists such that the kernel represents in fact a dot product in $\mathcal{H}$, even if such mapping does not need to be explicitly defined. Note that, for some kernel functions, $\mathcal{H}$ may even live on an infinite dimensional Hilbert space.

The validity of a given kernel function may be asserted by the Mercer's condition (Vapnik, 1995). For a given kernel function $K(\mathbf{x}, \mathbf{z})$, there exists a valid expansion

$$K(\mathbf{x}, \mathbf{z}) = \sum_k \Phi_k(\mathbf{x})\Phi_k(\mathbf{z}) \tag{4.4}$$

($\Phi_k(\mathbf{x})$ denoting the k-th component of $\Phi(\mathbf{x})$) if for any function $g(\mathbf{x})$ in $\mathbf{R}^N \to \mathbf{R}$ such that

$$\int_\infty (g(\mathbf{x}))^2 \, d\mathbf{x} < \infty \tag{4.5}$$

then

$$\int_\infty \int_\infty K(\mathbf{x}, \mathbf{z}) g(\mathbf{x}) g(\mathbf{z}) d\mathbf{x} d\mathbf{z} \geq 0. \tag{4.6}$$

Note that the validity of a kernel function requires that (4.6) is satisfied for any finite energy function $g(\mathbf{x})$ (finite L_2 norm). This may not be easy to prove, even for quite simple kernel types. On the other hand, it must be emphasized that some kernels often used do not satisfy Mercer's condition but, nevertheless, they are able to provide reasonable solutions to the optimization problem.

4.2 Kernel functions

Different types of SVM classifiers may be derived for a same problem by using different kernel functions. Some of the most used kernel functions include:

- Polynomial:

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z})^d, \;\; d = 1, 2, \cdots. \tag{4.7}$$

- The previous kernel function may result on an Hessian with 0 determinant. In order to avoid this problem, the following alternative polynomial kernel is often used:

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z} + 1)^d, \;\; d = 1, 2, \cdots. \tag{4.8}$$

- Radial basis function:

$$K(\mathbf{x},\mathbf{z}) = e^{\left(\frac{\|\mathbf{x}-\mathbf{z}\|^2}{\sigma^2}\right)}. \tag{4.9}$$

- Sigmoidal type (*two layers neural network kernel*):

$$K(\mathbf{x},\mathbf{z}) = \tanh\left(a\mathbf{x}^T\mathbf{z} - c\right). \tag{4.10}$$

Note that polynomial kernels always correspond to finite dimension feature spaces. Consider for example a polynomial kernel of order $d = 2$ defined on a bi-dimensional space. We have

$$\begin{aligned} K(\mathbf{x},\mathbf{z}) &= (\mathbf{x}^T\mathbf{z}+1)^2 \\ &= (x_1z_1 + x_2z_2 + 1)^2 \\ &= 1 + 2x_1z_1 + x_1^2z_1^2 + 2x_2z_2 \qquad (4.11) \\ &\quad +2x_1x_2z_1z_2 + x_2^2z_2^2 \qquad (4.12) \end{aligned}$$

and therefore the implicit mapping performed is given by

$$\Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ x_1^2 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}. \tag{4.13}$$

The last two kernel (radial and sigmoidal) correspond to an infinite dimension $\mathcal{H}$ spaces. It must be emphasized that the proper choice of the kernel function is one of the least clarified issues on the use of SVM.

4.3 Function and parameters evaluation

It was previously shown how the kernel function can avoid the explicit computation of the features $\Phi(\mathbf{x})$ in the optimization process. However, it remains to discuss how the SVM can be used for classification. In fact, the separating hyper-plane $\mathcal{P}$ lives in $\mathcal{H}$, and the evaluation of the output class for an arbitrary input pattern $\mathbf{x}$ requires the computation of

$$s = \mathbf{w}^T\Phi(\mathbf{x}) + b \tag{4.14}$$

But $\mathbf{w}$ lives in $\mathcal{H}$, which suggests that explicit operations in $\mathcal{H}$ must be performed.

In fact, this can be avoided. From (2.34), one have

$$\mathbf{w} = \sum_{k=1}^{N_s} \alpha_{s(k)} d_{s(k)} \Phi\left(\mathbf{x}_{s(k)}\right). \tag{4.15}$$

Replacement of (4.15) on (4.14), yields

$$s = \left(\sum_{k=1}^{N_s} \alpha_{s(k)} d_{s(k)} \Phi^T\left(\mathbf{x}_{s(k)}\right) \Phi\left(\mathbf{x}\right)\right) + b \tag{4.16}$$

and therefore

$$s = \left(\sum_{k=1}^{N_s} \alpha_{s(k)} d_{s(k)} K\left(\mathbf{x}_{s(k)}, \mathbf{x}\right)\right) + b. \tag{4.17}$$

Equation (4.17) provides a simple and effective way of computing the output s avoiding the explicit knowledge of $\Phi(\mathbf{x})$.

The computation of the b parameter was not addressed yet, but it can be performed by a similar approach. From (4.17), it is clear that choosing any support vector j such that $0 < \alpha_j < C$, one may evaluate

$$b = d_j - \sum_{k=1}^{N_s} \alpha_{s(k)} d_{s(k)} K\left(\mathbf{x}_{s(k)}, \mathbf{x}_j\right). \tag{4.18}$$

Once again, it is usually numerically safer to compute b from the average of all support vectors for which $0 < \alpha_j < C$.

For a radial basis function kernel, (4.17) assumes the form

$$s = \left(\sum_{k=1}^{N_s} \alpha_{s(k)} d_{s(k)} e^{\left(\frac{\|\mathbf{x}_{s(k)} - \mathbf{x}\|^2}{\sigma^2}\right)}\right) + b \tag{4.19}$$

and therefore the final classifier is equivalent to an RBF network with N_s Gaussian of identical variance, each one centered in a given support vector and with one output weight given by of $\alpha_{s(k)} d_{s(k)}$.

Much in the same way, for a sigmoidal kernel one have

$$s = \left(\sum_{k=1}^{N_s} \alpha_{s(k)} d_{s(k)} \tanh\left(a\mathbf{x}_{s(k)}^T \mathbf{x} - c\right)\right) + b \tag{4.20}$$

and, in this case, the classification performed by the SVM is equivalent to the output of a $N \times N_s \times 1$ MLP network, with a linear output unit with bias b. In this equivalent network, the N_s hidden units

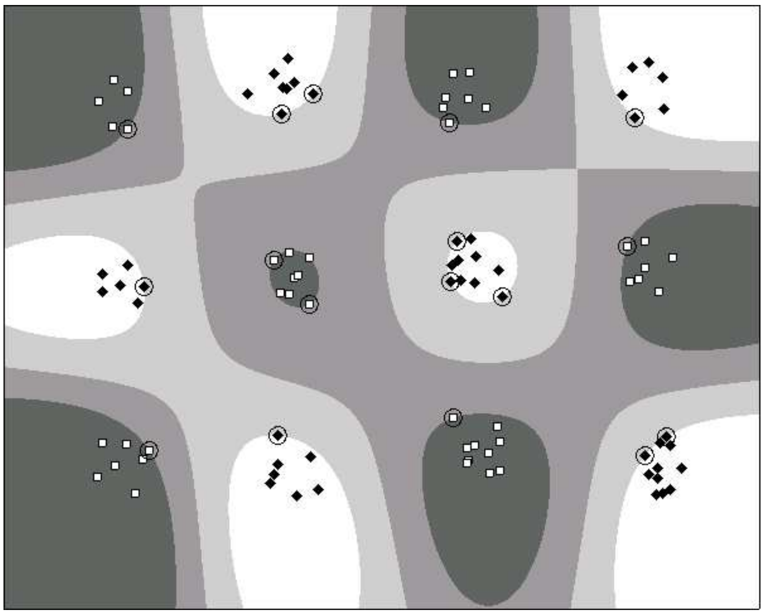

Figure 4.1: SVM classification of the *chess board* distribution of figure 3.2. RBF kernel with $\sigma^2 = 1$. $C = \infty$. $N_s = 17$. $N_e = 0$. $N_{om} = 0$.

have an hyperbolic tangent activation function, with an weight from input i to hidden unit k given by $x_{s(k)(i)}$ and an weight from hidden unit k to the output of of $\alpha_{s(k)} d_{s(k)}$.

In both cases, it is however remarkable that the SVM learning criteria avoids the explicit specification of the number of Gaussian or hidden units, which results spontaneously from the optimization process.

4.4 Experimental results

4.4.1 The *chess board* problem revisited

At this point it is interesting to consider again the *chess board* problem and compare the classification region obtained by explicit feature expansion methods and the one resulting from a SVM with a kernel. The result is depicted in figure 4.1 for an RBF kernel with $\sigma^2 = 1$.

Note that this result is similar to the best result obtained with polynomial feature expansion ($p = 5$). However, in this case, a quite good solution was found without the need to specify the dimension of $\mathcal{H}$ or any learning parameter[1]. This is certainly one of the most interesting features of the use of kernels in SVM.

[1]The weight coeffi cient C is the only training parameter which must be set in SVM optimization procedure. However, in this particular problem, it was not required an upper bound on the multipliers.

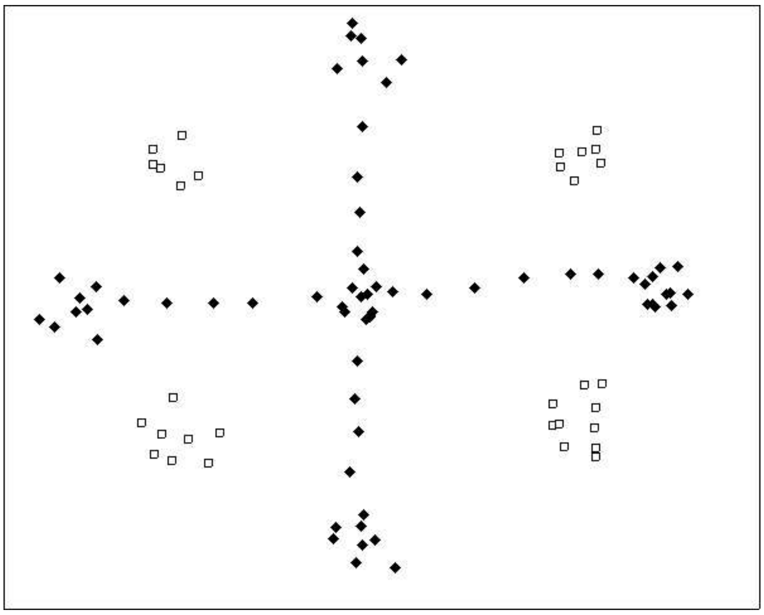
Figure 4.2: Data distribution in the cross problem (P=96).

.

4.4.2 The cross problem

Separable case

The second problem problem addressed with SVM and a RBF kernel function was the quite simple distribution depicted in figure 4.2. This distribution has 31 and 65 patterns in each one of the two classes, in a total of 96 patterns.

The results of this problem using an SVM classifier with RBF kernel and a $2 \times 8 \times 1$ MLP network are depicted in figures 4.3 and 4.4. Both classifiers were able to correctly classify all samples of the training set but, once again, the MLP solution is far less reasonable than the one found by SVMs.

The some classification problem was also solved with two RBF networks. The first one, with 8 gaussians (figure 4.5), has the same number of parameters as the MLP network, while the second, with 14 gaussians (figure 4.6), enables a solution with the same number of gaussians than the one found by the SVM. It is clear that both networks find much more reasonable solutions than the one found by the MLP network, and probably both networks would generalize well. However, the solution found by the SVM seems again much more balanced and plausible than those found by both RBF networks, which are apparently much more marked by the circular shape of the gaussian.

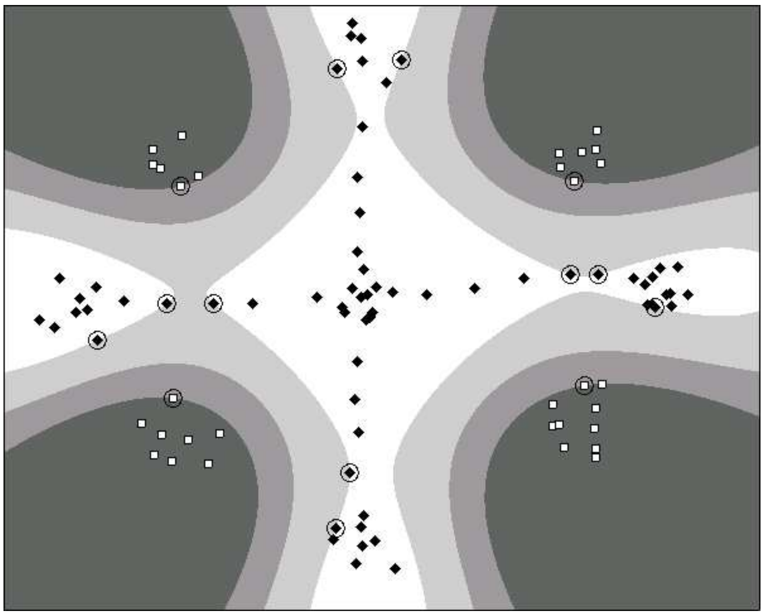

Figure 4.3: Classification region for the cross problem with a SVM with a RBF kernel. $C = \infty$. $N_s = 14$. $N_e = 0$. $N_{om} = 0$.

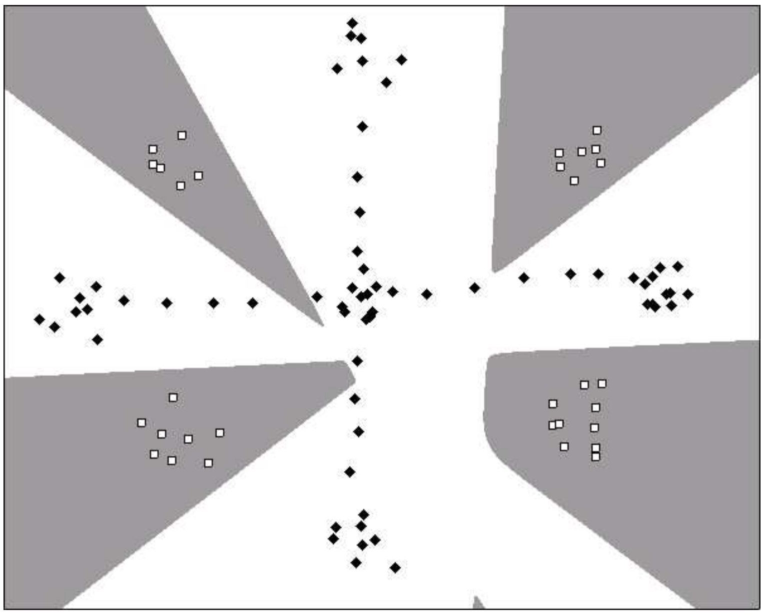

Figure 4.4: Classification region for the cross problem with a $2 \times 8 \times 1$ MLP. $N_e = 0$.

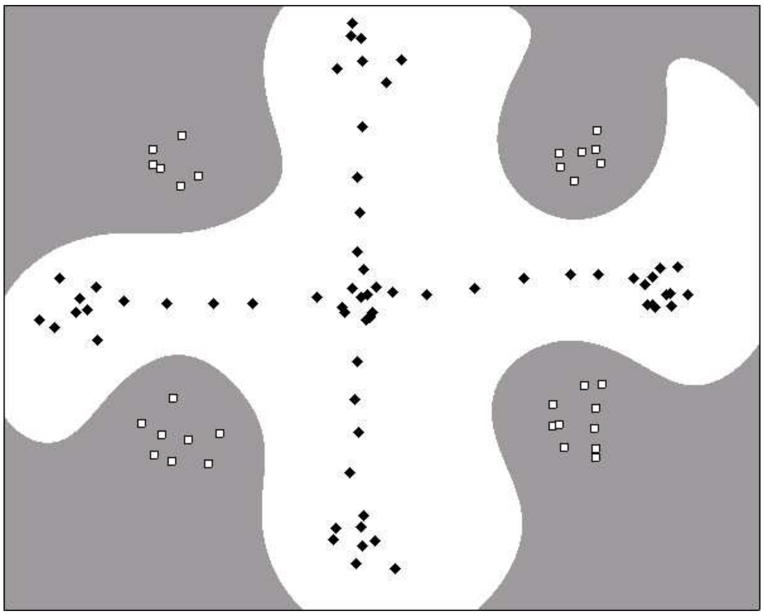

Figure 4.5: Classification region for the cross problem with an eight gaussian RBF network. $N_e = 0$.

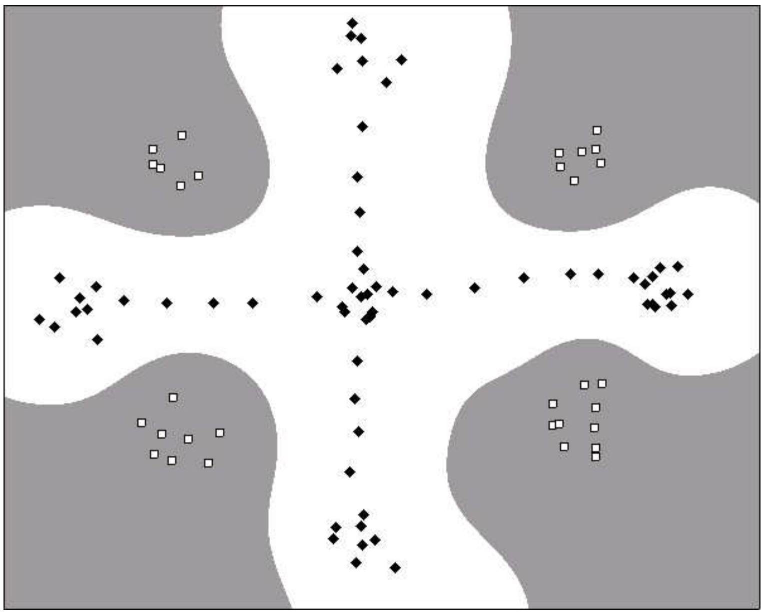

Figure 4.6: Classification region for the cross problem with a 14 gaussian RBF network. $N_e = 0$.

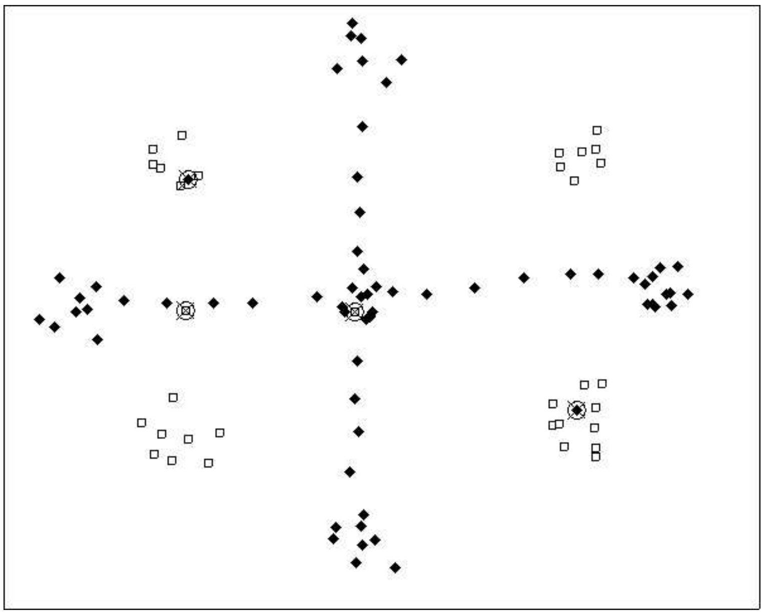

Figure 4.7: Data distribution in the cross problem with 4 outliers (P=100).

.

Noisy case

In order to assess the performance of the several classifiers in the presence of noise or outliers, the cross problem was again tested but, this time, four outliers were added to the data distribution (see figure 4.7). The results of this problem using a SVM classifier with RBF the kernel and a $2 \times 8 \times 1$ MLP network are depicted in figures 4.8 and 4.9.

In these, neither of these two classifiers was able to correctly classify all training samples. But, again, it must be emphasized how the MLP quickly reacted trying to correctly classify all outliers samplers and, therefore, yielding a quite unplausible solution with only one misclassified sample. On the other hand, the SVM reacted in a much more moderate way, basically making use of the slack variables to build a a classification boundary much closer to the one derived in the absence of outliers, and leaving the four outliers misclassified.

The some classification problem was again repeated with pure RBF networks. This time, the first test was performed again with eight gaussians (figure 4.10) and a second one with 23 gaussians (figure 4.11) in order to offer the same degrees of freedom of the solution found by SVM. In both cases the RBF performed reasonably well, being remarkable that in the first case the classifier basically ignored the outliers and built a solution also quite similar to the one found with the clean data distribution. However, with 23 gaussians, the solution found by the RBF network, with only one misclassified sample, is already clearly subject to over-learning, yielding a less reasonable decision boundary

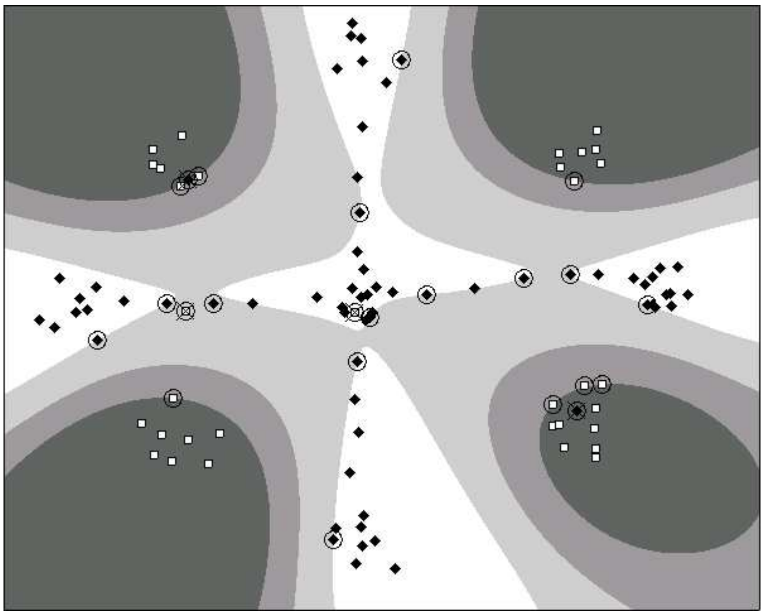

Figure 4.8: Classification region for the cross problem with 4 outliers and a SVM with a RBF kernel. $C = 100$. $N_s = 23$. $N_e = 4$. $N_{om} = 8$.

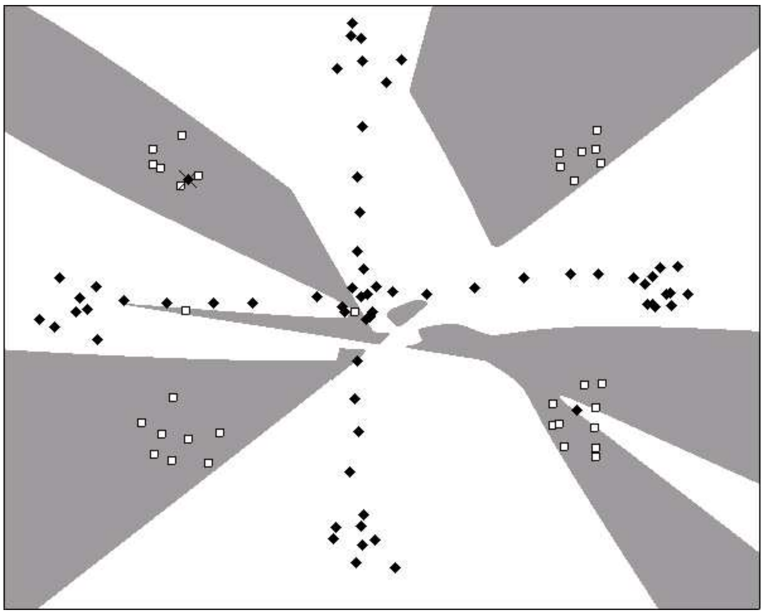

Figure 4.9: Classification region for the cross problem with 4 outliers and a $2 \times 8 \times 1$ MLP. $N_e = 1$.

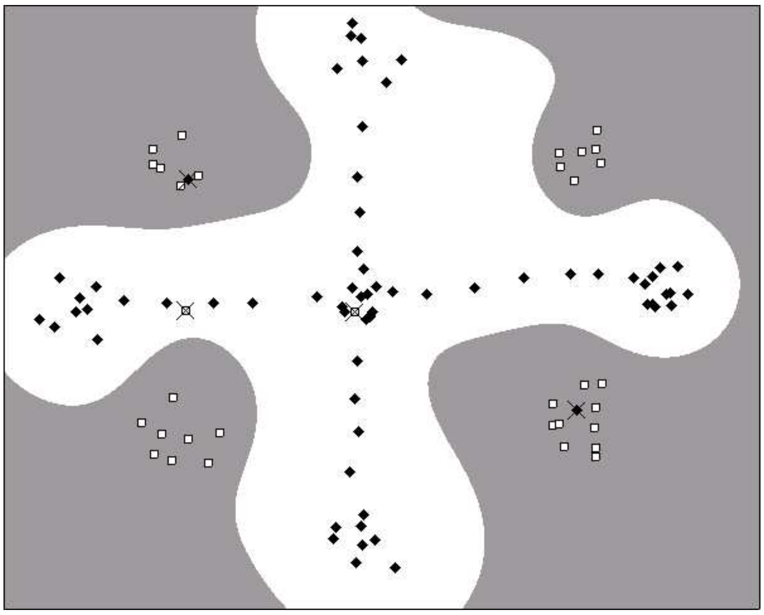

Figure 4.10: Classification region for the cross problem with four outliers and an eight gaussian RBF network. $N_e = 4$.

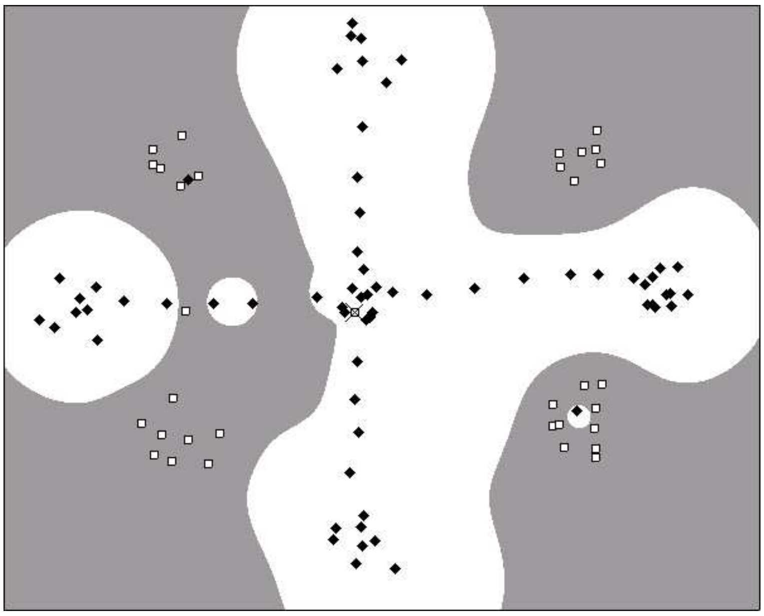

Figure 4.11: Classification region for the cross problem with four outliers and a 23 gaussian RBF network. $N_e = 1$.

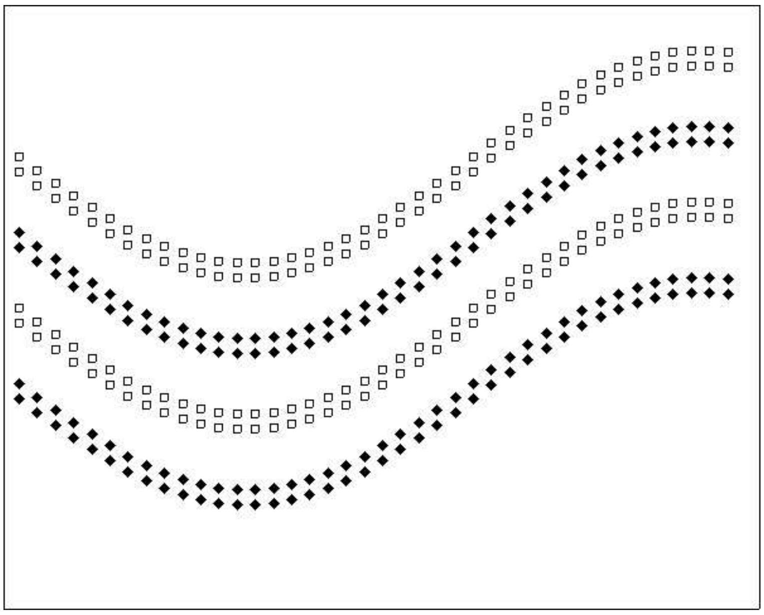

Figure 4.12: Source data distribution for the sinusoidal problem. P=320

4.4.3 The sinusoidal problem

Separable case

The sinusoidal problem represents one case where the data distribution is certainly far away from those usually found in real world problems. But any classifier must build a quite complex boundary line to correctly classify this problem and, therefore, it was considered an interesting case study.

The source data distribution was build in a rather deterministic way and it is represented in figure 4.12.

The solution found in this classification problem by a SVM with an RBF kernel and a $2 \times 8 \times 1$ MLP network are depicted in figures 4.13 and 4.14. Again, both classifiers were able to correctly classify all data samples. However, while the SVM built an elegant and balanced boundary line between both classes, the MLP solution was again marked by the straight lines which mark the sigmoidal units decision hyper-planes. While this solution built a reasonable decision boundary, it is clear that, for some samples, the classification margin is rather small.

Once again, we tried two RBF networks in this problem. The first one, based on the fixed reference of 8 gaussians, is depicted in figure 4.15. Its is clear that, in this particular case, the number of gaussians in the RBF network was far away from the minimum required to achieve a correct classification of all training patterns, therefore providing an odd – and useless – solution. On the second test, the RBF network was tested with the same number of gaussians (35) used by

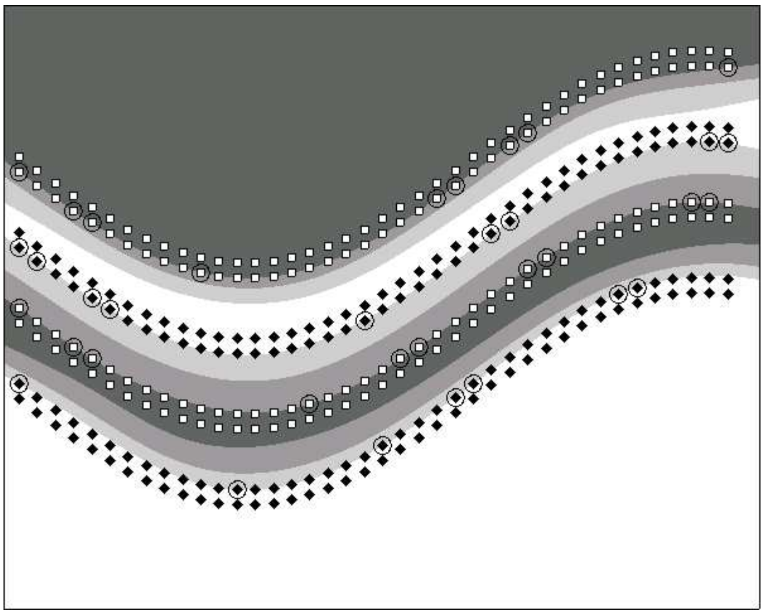

Figure 4.13: Sinusoidal problem. Solution found by a SVM network with a RBF kernel. $C = \infty$. $N_s = 35$. $N_e = 0$. $N_{om} = 0$

the SVM machine. In this case, the RBF network was able to find a reasonable solution that correctly classifies all data patterns. However, once more, the effect of the circular shaped gaussians is still noticeable on the boundary line.

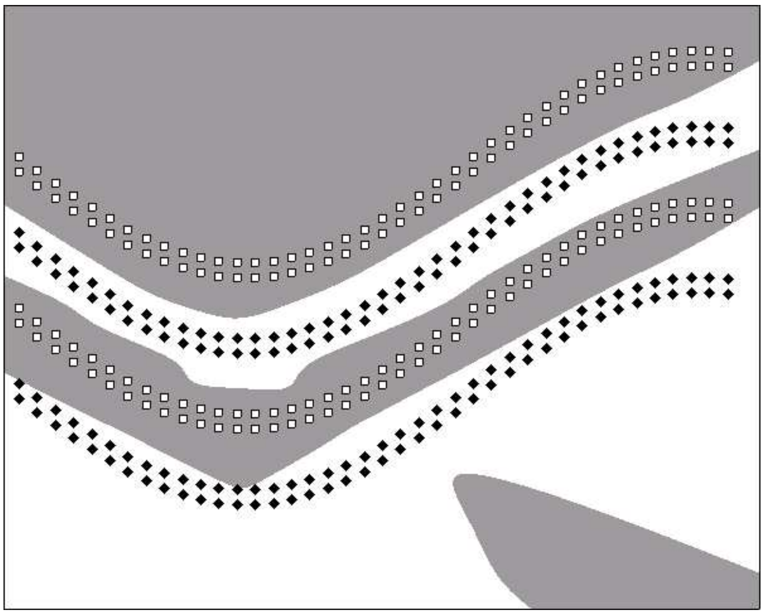

Figure 4.14: .
Sinusoidal problem. Solution found by a $2 \times 8 \times 1$ MLP network. $N_e = 0$.

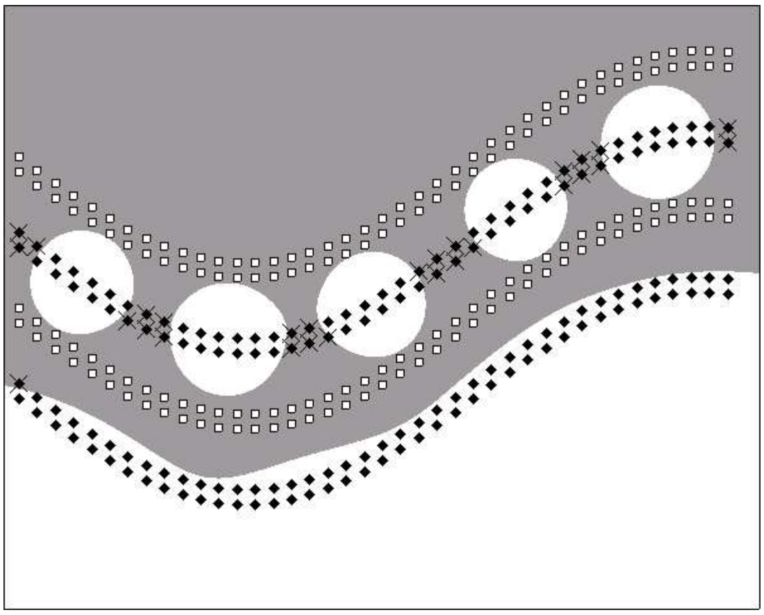

Figure 4.15: Sinusoidal problem. Solution found by a RBF network (eight gaussians). $N_e = 28$.

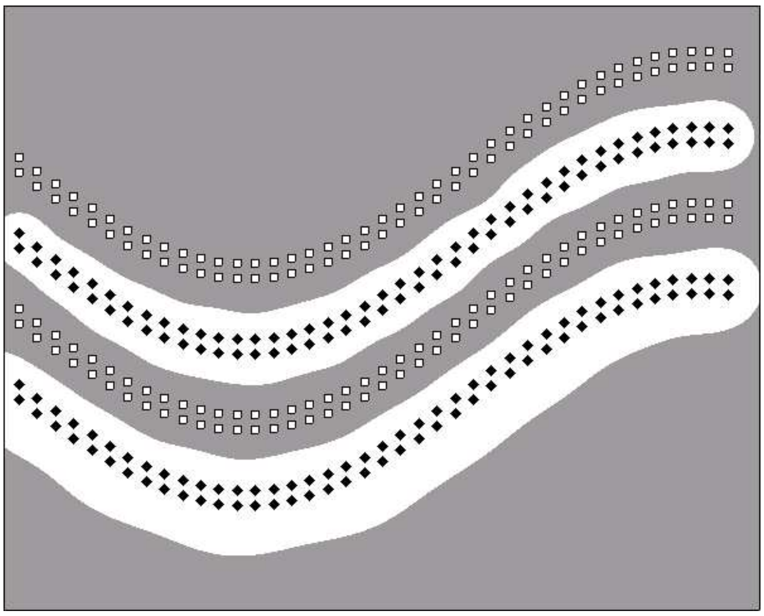

Figure 4.16: Sinusoidal problem. Solution found by a RBF network (35 gaussians). $N_e = 0$.

Chapter 5

Implementation issues

As discussed in chapter 2, training a support vector machine is equivalent to the minimization of the Lagrangian

$$L = -\frac{1}{2}\overline{\alpha}^T \mathbf{H}\overline{\alpha} + \mathbf{u^T}\overline{\alpha}, \tag{5.1}$$

subject to fairly simple linear constraints. This problem can be easily solved using standard optimization tools.

When the size of the training set is small, the *Matlab*$^{©}$ standard quadratic programming function **qp** can be directly used to build the SVM. The **qp** *Matlab* function has the syntax

$$\mathbf{x} = \mathrm{QP}(\mathbf{H}, \mathbf{f}, \mathbf{A}, \mathbf{b}, \mathbf{x}^l, \mathbf{x}^h) \tag{5.2}$$

and solves the generic quadratic problem

$$\min_{bfx} \frac{1}{2}\mathbf{x}^T \mathbf{H}\mathbf{x} + \mathbf{f}^T\mathbf{x} \tag{5.3}$$

subject to

$$\begin{aligned} \mathbf{A}\mathbf{x} &\leq \mathbf{b} && (5.4)\\ x_k^l &\leq x_k \leq x_k^h. && (5.5) \end{aligned}$$

The **qp** Matlab function is supplied in source code, therefore providing a good starting point for analysis and implementation of a simple quadratic programming package. It main drawback remains the need to fully build the **H** matrix before the optimization procedure. Therefore, this approach requires a memory amount in the order of $O(P^2)$ (P being the training set size).

Another approach to the implementation of SVM is to write dedicated code based on any standard non-linear programming package. The **DONLP2** package, available from the Netlib repository, is a powerful nonlinear programming package developed by Prof. Dr. Peter Spellucci (e-mail *spellucci@mathematik.th-darmstadt.de*). It uses an SQP algorithm and dense-matrix linear algebra. Source for DONLP2 is available by ftp from netlib ftp servers, e.g.,

ftp://netlib.bell-labs.com/netlib/opt/donlp2.tar.

DONLP2 was originally written in Fortran, but it can be translated to C by the widely available **f2c** translator. The main **DONLP2** module is an optimization routine which is parametrized via global variables included in standard Fortran *common* blocks. All the examples included in this document were computed by a dedicated C program which makes use of the DONLP2 package via a simplified set of C interface routines.

As stated before, the SVM optimization procedure implies to solve a fairly large quadratic programming problem. DONLP2, for instance, requires an amount of memory larger than $24P^2$ bytes, which easily grows to more than 2GBytes of memory with less than 10000 training samples. On the other hand, each iteration of the optimization procedure is a complex process which may require a large amount of time for large values of P.

One simple way of reducing the size of the optimization procedure is the elimination, in each iteration, of all patterns for which the multiplier α_i vanishes. This procedure was used in some of the tests reported in this document. Since it enables the progressive reduction of the effective dimension of the training set, it generates a much faster optimization after the first few iterations. Note, however, that since the monotonic evolution of the α_i is not guaranteed, the final solution found by this procedure may slightly differ from the one obtained keeping the initial training set size fixed.

Another procedure for building SVMs from large databases was proposed in (Osuna et al., 1997). The proposed method splits the optimization procedure in several smaller subtasks. Within each subtask, a given number of parameters is kept fixed and the optimization is performed over the remaining multipliers. This procedure is then iteratively solved for each parameter subset, until a stop criteria is met. The authors report that this procedure enabled to build a SVM with about 100,000 support vectors from a data base with 110,000 samples.

A full implementation of SVM is the SVM^{light} package, developed by Thorsten Joachims. Further information can be found at

http://www-ai.cs.uni-dortmund.de/FORSCHUNG/VERFAHREN/SVM_LIGHT/svm_light.eng.html.

Chapter 6

Conclusions

Support Vector Machines rely on the concept of optimal margin classification. While this is a reasonably understood concept in linear classification problems, its scope and conceptual basis is less trivial in nonlinear applications.

SVM address nonlinear problems by projecting the input space in a high dimensional feature space where the problem is assumed to be linearly separable. However, application of the optimal margin concept to the feature space requires some further discussion. In fact, the classification boundary that is derived by the SVM theory in the input data space is strongly dependent on the transformation that is performed from the input space to the feature space. This is affected by the choice of the feature expansion or the kernel function. But since different feature expansions lead to different solutions, the classifier found by the SVM theory for nonlinear problems is not unique and does not complies to a universal optimality criteria.

It was also discussed how SVM classifiers depends on the upper bound defined for the Lagrange multipliers. While it is clear that the proper choice of this parameter is dependent on the amount of *noise* on the data set, there is not yet an optimized procedure to set this parameter.

Nevertheless, the examples shown in this short report show that support vector machines exhibit a remarkable regularization behavior, often providing better classification boundaries than those found by other standard classifiers, namely MLP and RBF networks. When used with RBF kernels(Girosi, 1997), SVM provide an interesting solution to the choice of the centers of Radial Basis Functions, playing an important role in the choice of solutions which are sparse and, at the same time, can be related to the principle of Structural Risk Minimization. This point provides a strong motivation for the analysis of SVM as classifiers with near optimal generalization properties.

References

Baum, E. (1990). When are k-nearest neighbour and back propagation accurate for feasible sized sets of examples? In Almeida, L. B. and Wellekens, C. J., editors, *Neural Networks*, Lecture Notes in Computer Science, pages 69–80. EURASIP Workshop on Neural Networks, Sesimbra, Portugal, Springer-Verlag.

Baum, E. and Haussler, D. (1989). What size net gives valid generalization? In Touretzky, D., editor, *Advances in Neural Information Processing Systems 1*, pages 81–90, San Mateo, CA. Morgan Kaufmann.

Burges, C. J. C. (1986). A tutorial on support vector machines for pattern recognition. In *Adavnces in Knowledge Discovery and Data Mining@, the Microstructure of Cognition*. Kluwer Academic Press, Cambridge, MA.

Cohn, D. and Tesauro, G. (1991). Can neural networks do better than the Vapnik-Chervonenkis bounds? In Lippman, R., Moody, J., and Touretzky, D., editors, *Advances in Neural Information Processing Systems 3*, pages 911–917, San Mateo, CA. Morgan Kaufmann.

Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20:273–297.

Fletcher, R. (1987). *Practical Methods of Optimization*. John Wiley and Sons, Inc.

Girosi, F. (1997). An equivalence between sparse approximation and support vector machines. Technical Report A.I. Memo No. 1606, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

Gunn, S. (1997). Support vector machines for classification and regression. Technical Report http://www.isis.ecs.soton.ac.uk/research/svm/svm.pdf, University of Southampton.

Guyon, I., Vapnik, V., Bottou, L., and Solla, S. A. (1992). Structural risk minimization for character recognition. In Hanson, S., Cowan, J., and Giles, C., editors, *Advances in Neural Information Processing Systems 4*, pages 471–479, San Mateo, CA. Morgan Kaufman.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. Macmillan.

Ji, C. and Psaltis, D. (1992). The VC-dimension versus the statistical capacity of multilayer networks. In Hanson, S., Cowan, J., and Giles, C., editors, *Advances in Neural Information Processing Systems 4*, pages 928–935, San Mateo, CA. Morgan Kaufman.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In Hanson, S., Cowan, J., and Giles, C., editors, *Advances in Neural Information Processing Systems 4*, pages 950–957, San Mateo, CA. Morgan Kaufman.

Le Cun, Y., Denjer, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. (1990). Handwritten digit recognition with a backpropagation network. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2*, pages 396–404, San Mateo, CA. Morgan Kaufmann.

Minsky, M. and Papert, S. (1969). *Perceptrons*. MIT Press, Cambridge, MA.

Osuna, E., Freund, R., and Girosi, F. (1997). An improved training algorithm for support vector machines. In *Proc. of IEEE NNSP 97*, Amelia Island.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge, MA.

Vapnik, V. N. (1992). Principles of risk minimization for learning theory. In Hanson, S., Cowan, J., and Giles, C., editors, *Advances in Neural Information Processing Systems 4*, pages 831–838, San Mateo, CA. Morgan Kaufman.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Widrow, B. and Hoff, M. (1960). Adaptive switching circuits. In *IRE WESCON Convention Record, Part4*, pages 96–104.