Learning Polynomial Generating Rules using Support Vector Machines

Tamer Salman

Learning Polynomial Generating Rules using Support Vector Machines

Research Thesis

Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science

Tamer Salman

Submitted to the Senate of
the Technion — Israel Institute of Technology
Adar 5764 Haifa March 2004

The research was done under the supervision of Prof. Yoram Baram and Prof. Alfred Bruckstein in the Department of Computer Science.

I thank Prof. Yoram Baram and Prof. Alfred Bruckstein for their guidance and help during all stages of research.
I thank my friend Dr. Hashem Zoubi for the appreciated conversasions and help.
I also would like to thank my parents Layla and Milad, my brothers Dr. Hanna Salman, Dr. Zaher Salman and Iyas Salman , and for my wife Hanan for their support.

The generous financial help of the Technion Graduate School is gratefully acknowledged.

Dedicated to Layla and Milad

Contents

List of Figures

Abstract

Model selection using support vector machines (SVM) has been widely applied and investigated in the radial basis functions (RBF) kernels case. The time consumption has been reduced meaningfully using smooth upper bounds on the generalization ability. However, there have been no conclusions about the analytical structure of the rule that generates the data and labels it. In this work we present a simple model selection algorithm for polynomial kernel SVM that results in tuning the polynomial degree of the used kernel to the best fit value. We show that this procedure provides us with some understanding of the analytical behavior of the generating rule of the data, assuming it is polynomial. Although the procedure is a heavy time consumer and many improvements from the RBF SVM model selection cannot be applied to it, it can be very helpful in understanding "nature rules" and analyzing certain qualities and behaviors. We present the case of the 2D circle recognition among 2D geometrical shapes, and show that some conclusions can be reached concerning the properties of circles that separate them from other geometrical shapes. A training sample consists of feature vectors of various properties of 2D shapes. A feature selection procedure can be applied to this data, choosing a partial set of features. The remaining features will be the discriminating ones; the area and the perimeter of the shape. The system is trained with various oracles that are a result of different mistake factors. One of the oracles is an average vote of a number of people. We show that the resulting system's success rate is better than the best member of the oracle group, and that the model selection procedure yields a quadratic relation between the area and the perimeter of the shape, a known mathematical fact.

List of Symbols

x^i	:an input vector
x^i_j	:a component of the vector
d	:dimension of input space
y^i	:a classification value
l	:training set size
f	:a separating function
θ	:a free parameters vector
Θ	:the space of free parameters
$P(x, y)$	:a probability distribution function
R	:actual risk functional
R_{emp}	:empirical risk functional
w	:a weight vector
b	:a scalar threshold value
α	:Lagrange multipliers vector
α_i	:a Lagrange multiplier
ξ_i	:a slack variable
C	:apenalty factor
Φ	:mapping function to the feature space
R^D	:feature space
z_i	:a vector in feature space
K	:a kernel function
s^i	:a support vector in input space
g	:a real function from
σ	:Radial Basis Function parameter
$coef$	:a scalar coefficient of the polynomial kernel function
p	:polynomial kernel degree
h	:Vapnik-Chervonenkis (VC) dimension
ψ	:VC confidence
r	:radius of the smallest enclosing sphere of a set of points
m	:number of support vectors
Λ_p	:a set of constraing linear combinations
λ	:a weight factor
S_p	:p-span
S	:S-span

D	:diameter of the smallest enclosing sphere of a set of points
$c^{(,,\ldots,)}$	:a coefficient of the polynomial separator function
A	:area of a 2D shape
P	:perimeter of a 2D shape
ϵ	:inaccuracy scalar value
δ	:indecision region width
γ	:margin
SVM	:Support Vector Machine
RBF	:Radial Basis Function
SV	:Support Vector
ERM	:Empirical Risk minimization
VC	:Vapnik-Chervonenkis
KKT	:Karush-Kuhn-Tucker
LOO	:Leave One out
kPCA	:kernel based Principle Components Analysis

Chapter 1

Introduction

1.1 Background

Pattern recognition has been used in many fields and applied to a wide variety of real life problems through various algorithms and methodologies. The problem of pattern recognition, which is also called classification, is finding a function that classifies data generated from some unknown probability function using only a finite training set with known labeling.

Suppose we are given a sample set of l points, where each point is a vector in some d-dimensional space called the input space. The set of sample points is denoted by

$$\left\{x^i\right\}_{i=1}^{l}, x^i \in R^d$$

The set of points is labeled with classification values

$$\left\{y^i\right\}_{i=1}^{l}, y^i \in R^1$$

where x^i belongs to the class y^i.

In this context, we wish to find a function that classifies a new point x correctly with high probability. This function is usually picked out of a certain given family of functions:

$$f(x, \theta) : R^d \to R^1$$

where θ is a vector of the free parameters. Each assignment for this vector produces a certain function from the family.

In our work we will focus on the binary classification problem, which is the case with only two possible classes and ,hence, two possible classification values. Usually, in real life problems, a sample set of some objects, such as images, sound samples or the record of a patient is given,. In these cases a preprocessing phase is needed in order to reduce the data to a classification problem. This phase is called feature extraction. In feature extraction we try to transfer the object set into a set of numerical values that might be important for classifying a certain object. The final product of a training algorithm is the classifier, which can be treated as a black box that receives an object as input and outputs the estimated class to which the object belongs.

Many algorithms in statistical learning theory have been developed during the last forty years, beginning with Rosenblatt's Perceptron [1], through artificial neural networks [2][3], to support vector machines, which were developed by V. Vapnik [4]. The Support Vector Machines (SVM) algorithm is one of the most widely used algorithms in statistical learning for pattern recognition. It has been shown to be superior to many other algorithms. Since its introduction, it has been applied to many problems in pattern recognition [5][6][7][8][9][10]. These works focus on building certain machines for certain problems given a set of example patterns. These applications usually produce a specific support vector machine with specific settings and parameters. Hence, the result is a procedure of feature extraction and preprocessing followed by a certain learning machine.

In this work we suggest a different direction for the use of SVM that can be helpful for many applications in various fields of interest. We use it in a model selection procedure to understand the analytical behavior of rules that stand behind given data sets and recognition problems.

1.2 Thesis Outline

In the beginning we will present the main cornerstones of the Vapnik-Chervonenkis (VC) theory, which will help us introduce the SVM method for pattern recognition, discuss its abilities and explain its outstanding performance. Along with the theoretical grounds, we will show its experimental advantages. Then, we will present several estimators and upper bounds on the error rate of SV machines on random unseen data, which are called risk bound or bounds on the generalization error. Then we will present methods of parameter tuning, called model selection, that were used in previous works [11][12][13][14] for a certain kind of SV machines based on Gaussian kernel functions or radial basis functions (RBF). Following that, we will devise a model selection procedure for SV machines with polynomial kernel functions and discuss its ability to learn the analytical rule that generated the given data set under some assumptions. We will show that we can discover the true relation between the features of the data when the generator, which we call "nature rule", is assumed to be a polynomial relation between the features. Although we lose several advantages of model selection by RBF SVM, we will gain the possibility for better understanding of problems with polynomial nature.

Finally, we will focus on the following case for clarification: A person can identify whether a geometric shape is a circle or not. Different trials will yield different success rates. Often, a person cannot explain by what criteria he classified the data. To demonstrate this difficulty, we can check whether a person can guide another person in making a decision based only upon measuring various features of a given shape. In this work we give a method using outcomes of learning machines to analyze the rule generating the given data. In the case of the 2D circle recognition among 2D connected shapes, we learn the relation between the features of the shapes. The features of the data might contain redundancy that can be cancelled using a feature selection procedure. From mathematics, we know that feature selection for circle recognition will result with only the perimeter and the area of the shape. Our model selection will guide us to the conclusion that this relation is quadratic, a known fact derived from basic geometric rules.

Chapter 2

Support Vector Machines for Pattern Recognition

Designing a classifier is choosing a decision function out of a set of functions depending on some training set of observations. Each observation is a vector of features and an integer classification value. The training set is independently drawn and identically distributed according to some unknown probability distribution

$$P(x, y)$$

The classifier we seek is a mapping from the space of the feature vectors into the space of the classes. In binary classification, the classes are $+1$ and -1, thus it is defined as a function f of the input vector x with parameters vector $\theta \in \Theta$ as:

$$y = sgn\,(f(x, \theta))$$

$$f(x, \theta) : R^d \to R^1$$

A good classifier is one that produces, with high probability, correct classification results on unseen data. This measure is known as the classifier's generalization ability or risk. It's defined as:

$$R(\theta) = \int \frac{1}{2} \left| y - f(x, \theta) \right| dP(x, y) \tag{2.1}$$

Unfortunately, this expression cannot be evaluated, since the probability distribution $P(x, y)$ is unknown. Alternatively, one may approximate the risk by various upper bounds and estimators for the error rate on unseen data.

One method for estimating the actual risk of a classifier is the Empirical Risk Minimization (ERM) inductive principle. This principle states that the classifier's performance on the unseen data from an unknown probability distribution will be close to the risk of the classifier on the seen data - the training set. When given l training points $\{x^i\}_{i=1}^{l}$ labeled by $\{y^i\}_{i=1}^{l}$, the empirical risk is defined as:

$$R_{emp}(\theta) = \frac{1}{2l}\sum_{i=1}^{l} |y^i - f(x^i, \theta)| \tag{2.2}$$

A straightforward guess would probably yield this principle, since (2.2) is simply a discrete estimate of the actual risk given in (2.1), with the replacement of a continuous integral with a discrete summation upon the given patterns.

However, this principle is actually very problematic, since the estimates obtained by it are poor when the training set size is small. Furthermore, minimizing the empirical risk as a criterion for convergence within the procedure of building a classifier may cause the classifier to be a very complex separation surface that lacks any capability for generalization, even though it separates the training set perfectly. This phenomenon is known as overfitting.

Hence, a good minimization principle for the convergence of a classifier must be one that prevents overfitting by balancing the quantity of the empirical risk by another term to forbid complex classifier structures.

The Vapnik-Chervonenkis (VC) theory gives necessary and sufficient conditions for the consistency of the learning process as well as a capacity measure for the learning machine [4]. The measure for capacity is called VC dimension. It is defined as follows: [4][15]

Definition 2.0.1: *The VC dimension of a set of indicator functions $f(x, \theta), \theta \in \Theta$ is the maximum number h of vectors $x^1, x^2, ..., x^h$ that can be separated into two classes in all 2^h possible ways (shattered) using functions of the set. If for any n there exists a set of n vectors that can be shattered by the set of functions, then the VC dimension equals infinity.*

The VC dimension of a set of functions is a measure of the complexity of the set. The ERM principle is consistent with the generalization ability when the VC dimension is small enough. But, it is important to note that the VC dimension does not always give good estimates for the problem's complexity. [16] presents some examples where the VC dimension is not consistent with the resulting performances. Thus, neither infinite "capacity" guarantees poor performance nor small VC dimension guarantees good performance [16].

Consequently, designing a good classifier should depend on complexity measures as well as separation abilities. In the next section we present the main development procedure of a learning algorithm that implements the mentioned capacity control and balances between the separation abilities and the surface complexity.

2.1 Linear Support Vector Machines

In this section we present the linear support vector machines (SVM) method for pattern recognition, mainly according to [4] and [16]. In the first part we present the linear SVM for linearly separable data and in the second part for linearly non-separable data. In the next section we present non-linear SVM as an extension of the linear case by using a kernel function mapping the given data into a different space, called feature space. The linear algorithm for separation is then applied in feature space, but its pre-image in the original input space will not be linear and will depend on the applied kernel function.

SVM is an optimization algorithm of a convex quadratic function with linear constraints. The solution of the optimization problem in this case is always optimal, since the extremum points are global. Furthermore, the solution is also unique. An extensive explanation on these issues can be found in [16].

The Separable Case

Suppose that the training data and labeling are $\{x^i, y^i\}$, where $x^i \in R^d$, $y^i \in \{+1, -1\}$. A separating hyperplane in R^d can be represented by a vector $w \in R^d$ and a constant threshold b as follows:

$$< w, x > +b = 0 \tag{2.3}$$

where$< w, x >$ denotes the inner product of the two vectors.

The "margin" induced by the hyperplane on the training set is the sum of distances of the two closest points from both classes to the hyperplane.

Linear SVM finds the separating hyperplane with maximal margin such that each training point satisfies one of the following:

$$\left\langle w, x^i \right\rangle + b \geq 0 \qquad for\ y^i = +1 \tag{2.4}$$

$$\left\langle w, x^i \right\rangle + b \leq 0 \qquad for\ y^i = -1 \tag{2.5}$$

Thus, by normalization of (w, b) we can assure that for each point one of the following holds true as well:

$$\left\langle w, x^i \right\rangle + b \geq +1 \qquad for\ y^i = +1 \tag{2.6}$$

$$\left\langle w, x^i \right\rangle + b \leq -1 \qquad for\ y^i = -1 \tag{2.7}$$

which can be combined into

$$y^i \left(\left\langle w, x^i \right\rangle + b \right) - 1 \geq 0 \qquad \forall i \tag{2.8}$$

The equality in (2.8) holds true for points that lie on one of the two following hyperplanes:

$$L1 : \left\langle w, x^i \right\rangle + b = 1$$

or

$$L2 : \left\langle w, x^i \right\rangle + b = -1$$

Hence, the margin is

$$\frac{2}{\|w\|}$$

Maximal margin can be obtained by minimizing $\|w\|$, subject to the constraints (2.8):

minimize

$$\frac{1}{2}\|w\|^2$$

subject to

$$y^i \left(\left\langle w, x^i \right\rangle + b \right) - 1 \geq 0$$

Constrained optimization theory supplies a solution by introducing positive Lagrange multipliers $\alpha_i, i = 1, ..., l$ and transforming the problem to:

minimize

$$\frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i \left[y^i \left(\left\langle w, x^i \right\rangle + b \right) - 1 \right] \tag{2.9}$$

with respect to w and b and maximize it with respect to α.

(2.9) is convex, giving SVM an enormous advantage over other learning machines such as artificial neural networks concerning the globality and uniqueness of the solution. The solution of this optimization problem is its global minimum, whereas in neural networks many local minima may exist.

Demanding the Lagrangian derivatives, with respect to w and b to vanish at the solution gives:

$$w = \sum_{i=1}^{l} \alpha_i y^i x^i \tag{2.10}$$

$$\sum_{i=1}^{l} \alpha_i y^i = 0 \tag{2.11}$$

Substituting (2.10) and (2.11) in (2.9) produces the following optimization problem:

maximize

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y^i y^j \left\langle x^i, x^j \right\rangle \tag{2.12}$$

subject to

$$\alpha_i \geq 0, \quad \forall i$$

$$\sum_{i=1}^{l} \alpha_i y^i = 0$$

The solution in most cases is found numerically. An iterative algorithm finds an assignment α^* for the vector α, for which the Lagrangian is minimal. Each training point, with positive multiplier $\alpha_i^* > 0$ is called a support vector (SV). All other points do not affect the resulting solution and, hence, can be excluded in any future training.

The solution for b is given from constrained optimization theory. The Karush Kuhn Tucker (KKT) conditions supply the solution for b. The conditions for the problem at hand are:

$$\frac{\partial}{\partial w_\nu} L_{primal} = w_\nu - \sum_{i=1}^{l} \alpha_i y^i x^{i\nu} \quad = \quad 0 \tag{2.13}$$

$$\frac{\partial}{\partial b} L_{primal} = -\sum_{i=1}^{l} \alpha_i y^i \quad = \quad 0 \tag{2.14}$$

$$y^i(\left\langle w, x^i \right\rangle + b) - 1 \quad \geq \quad 0 \qquad i = 1, ..., l \tag{2.15}$$

$$\alpha_i \quad \geq \quad 0 \qquad i = 1, ..., l \tag{2.16}$$

$$\alpha_i(y^i(\left\langle w, x^i \right\rangle + b) - 1) \quad = \quad 0 \qquad i = 1, ..., l \tag{2.17}$$

KKT conditions are necessary and sufficient conditions for w, b and α to be a solution. Finding b becomes easy once we know w, for example,using the third inequality of KKT conditions for any support vector. We get:

$$b = y^i - \left\langle w, x^i \right\rangle$$

Finally, SVM classifies unseen patterns x by:

$$sgn\left(\langle w, x \rangle + b\right)$$

The Non-Separable Case

When the data at hand is not linearly separable into two classes, the above presented solution is not applicable, since the iterative solution for (2.12) will not converge and the multipliers α_i will increase to infinity. An additional mechanism must be presented to allow tolerance of classification errors [4]. The mechanism can be applied by introducing positive slack variables $\xi_1, ..., \xi_l$ into the constraints (2.6):

$$\left\langle w, x^i \right\rangle + b \quad \geq \quad +1 + \xi_i \qquad for\ y^i = +1 \tag{2.18}$$

$$\left\langle w, x^i \right\rangle + b \leq -1 - \xi_i \qquad for\ y^i = -1 \tag{2.19}$$

$$\xi_i \geq 0 \qquad \forall i \tag{2.20}$$

Balancing between the need to tolerate classification errors and the need to minimize the number of errors is applied by the addition of a cost function. The cost function is an upper bound to the number of classification errors amongst the given training patterns. Therefore, instead of minimizing

$$\|w\|^2/2$$

we seek to minimize

$$\|w\|^2/2 + C(\sum_{i=1}^{l} \xi_i)$$

where, C is a penalty factor. The new optimization problem becomes:

maximize

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y^i y^j \left\langle x^i, x^j \right\rangle \tag{2.21}$$

$subject\ to:$

$$0 \leq \alpha_i \leq C,\ \forall\ i \tag{2.22}$$

$$\sum_{i=1}^{l} \alpha_i y^i = 0 \tag{2.23}$$

The solution in this case is similar to the separable case's solution, which is given by (2.10) and the KKT conditions for b.

2.2 Non-Linear Support Vector Machines

Suppose we can map the data patterns from R^d into a high dimensional space $F \equiv R^D, D > d$ called feature space. The mapped points z^i will be related to x^i through a transformation function Φ as follows:

$$\Phi : R^d \rightarrow R^D \tag{2.24}$$

$$\Phi(x^i) = z^i \tag{2.25}$$

If we try to separate the set $\{z^i\}_{i=1}^{l}$ with linear SVM, the solution for w is:

$$w = \sum_{i=1}^{l} \alpha_i y^i \left\langle z^i, z^j \right\rangle = \sum_{i=1}^{l} \alpha_i y^i \left\langle \Phi(x^i), \Phi(x^j) \right\rangle \tag{2.26}$$

and b is found using the KKT conditions.

This mapping is highly not applicable and rather useless, since the transformation into a high dimensional space and the computations of all inner products are heavily time consuming. Furthermore, the resulting classifier will generalize poorly because of the high dimensionality of the feature space. In VC terms, the VC dimension of the set of functions will be $(D+1)$, which is too large. As a solution, the inner product of the mappings is replaced with a function of the two input vectors:

$$K(x^i, x^j) : R^d \times R^d \to R^1$$

such that

$$K(x^i, x^j) = \left\langle \Phi(x^i), \Phi(x^j) \right\rangle \tag{2.27}$$

Then, Instead of computing several inner products in the high dimensional space R^D we substitute the function K and all computations become less complex in R^d. Such function is called "kernel function".

Note that w is a vector in R^D and not in R^d. The solution for the linear case is a linear hyperplane with normal w. But the mapping Φ is nonlinear, therefore, the solution is a nonlinear surface in R^d, whose mapping by Φ is linear with a normal w in R^D.

The resulting classifier classifies new points x as:

$$sgn(f(x))$$

$$f(x) = \sum_{i=1}^{l} \alpha_i y^i K(s^i, x) + b$$

Of course, the only points that participate in the summation operation are the support vectors, and thus, we can write:

$$f(x) = \sum_{i=1}^{N_S} \alpha_i y^i K(s^i, x) \tag{2.28}$$

where $s^i : i = 1, ..., N_S$ are the support vectors and N_S is their number.

Finding the multipliers α_i is similar to the linear case:

maximize

$$L_{dual} = \sum_{i=1}^{l} \alpha_i \quad - \quad \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y^i y^j K(x^i, x^j) \tag{2.29}$$

$$subject\ to:$$

$$\alpha_i \geq 0, \ \forall\ i \tag{2.30}$$

$$\sum_{i=1}^{l} \alpha_i y^i = 0 \tag{2.31}$$

Analogously, in the non-separable case:

maximize

$$L_{dual} = \sum_{i=1}^{l} \alpha_i \quad - \quad \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y^i y^j K(x^i, x^j) \tag{2.32}$$

$$subject\ to:$$

$$0 \leq \alpha_i \quad \leq \quad C,\ \forall\ i \tag{2.33}$$

$$\sum_{i=1}^{l} \alpha_i y^i \quad = \quad 0 \tag{2.34}$$

The uniqueness and globality of the solution still hold, since the functional is still quadratic. However, many questions arise concerning the use of kernel functions, such as, (1) how do we find the functions Φ? (2) how do we know whether a function K is a kernel function? and (3) how do we know what is the function Φ when we are only given K?

There is no use for the function Φ in the algorithm and thus it's not needed. Moreover, there is no guarantee for the uniqueness of the mapping function Φ nor its dimensionality once the kernel function is given. A simple example can demonstrate the non-uniqueness of the mapping function. Given two dimensional data patterns [16]:

$$x^1 = \begin{pmatrix} x_1^1 \\ x_2^1 \end{pmatrix}$$

$$x^2 = \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix}$$

and a kernel function

$$K\left(x^1, x^2\right) = \left\langle x^1, x^2 \right\rangle^2$$

The mapping function could be either

$$\Phi_1 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix}$$

or

$$\Phi_2 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ x_1 x_2 \\ x_1 x_2 \\ x_2^2 \end{pmatrix}$$

as both of them satisfy

$$\left\langle \Phi_1 \begin{pmatrix} x_1^1 \\ x_2^1 \end{pmatrix}, \Phi_1 \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix} \right\rangle = \left\langle x^1, x^2 \right\rangle^2 = K\left(x^1, x^2\right) \tag{2.35}$$

and

$$\left\langle \Phi_2\begin{pmatrix} x_1^1 \\ x_2^1 \end{pmatrix}, \Phi_2\begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix} \right\rangle = \left\langle x^1, x^2 \right\rangle^2 = K\left(x^1, x^2\right) \tag{2.36}$$

However, we would have to be able to tell whether a function K is a kernel function, meaning is there any function Φ such that (2.27) holds true? The answer is called Mercer's condition, which states that there is an expansion for a function $K(x^1, x^2)$ in the form of (2.27) if and only if for all functions

$$g : R^d \to R^1$$

such that

$$\int_{-\infty}^{\infty} g^2(x)dx < \infty$$

the following holds:

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} K(x^1, x^2)g(x^1)g(x^2)dx^1dx^2 \geq 0$$

i.e. K is a positive semi-definite kernel.

Mercer's condition requires proof for every finite energy function and it is very difficult to figure out if there is an expansion of this sort. However, there are some known and widely used kernel functions that we will discuss.

Radial Basis Function (RBF) Kernels

The form of radial basis functions (RBF) we will present in this work is the Gaussian functions:

$$K(x^1, x^2) = e^{-\|x^1 - x^2\|^2/2\sigma^2} \tag{2.37}$$

Other forms of RBF kernel functions exist, such as the sub-linear RBF kernel [17].

These kernels have infinite VC dimension and their ability to shatter any set of points is obvious since the parameter σ can be chosen arbitrarily small and assure that each point is classified with its own Gaussian [16]. Nevertheless, they seem to generalize well when applied on various real life problems [5][6][10][7][8][9][18][19].

The Gaussian kernel is perhaps the leading kernel in SVM literature. We will discuss former works concerning this family of kernels and determine the reasons that make RBF SVM generalize well. A more general form of the RBF family [17] is given by:

$$K\left(x^1, x^2\right) = e^{-\rho d_{a,b,c}\left(x^1, x^2\right)} \tag{2.38}$$

$$d_{a,b,c}\left(x^1, x^2\right) = \left(\sum_i \left|\left(x_i^1\right)^a - \left(x_i^2\right)^a\right|^b\right)^c \tag{2.39}$$

Where b must satisfy $0 \leq b \leq 2$ in order to satisfy Mercer's condition. The value of ab provides different speeds of decaying, while the value of a alone is a simple nonlinear mapping of the input with no impact on Mercer's condition. Thus, the Gaussian kernel (2.37) is only a specific case ($a = 1, b = 2, c = 1$) and its free parameter of the is $\sigma \in R^{+}$.

Polynomial Kernels

Polynomial kernels have a variety of interesting features concerning their ability to generalize and the properties of the induced feature space. To investigate them we present two kinds of polynomial kernels. The first is the homogeneous polynomial kernels family:

$$K(x^1, x^2) = \left\langle x^1, x^2 \right\rangle^p \tag{2.40}$$

where the parameter is the polynomial degree p. The second family of polynomial kernels is the non-homogeneous polynomial kernels family:

$$K(x^1, x^2) = \left(\left\langle x^1, x^2 \right\rangle + coef \right)^p \tag{2.41}$$

where the parameters are the coefficient $coef$ and the polynomial degree p. In our experiments we will set $coef = 1$.

The complexity of this family can be estimated by its VC dimension. The VC dimension of the homogeneous polynomial kernel family is the dimension of the minimal embedding space + 1, and the dimension of the minimal embedding space is given through a theorem in [16] as

$$\binom{d+p-1}{p}$$

where d is the dimension of the input space. In addition we can also find an explicit mapping to the minimal embedding space of the kernel

$$K(x^s, x^t) = \left\langle x^s, x^t \right\rangle^p = (\sum_{i=1}^{d} x_i^s x_i^t)^p$$

Each dimension of the feature space is a term given by a combination of powers of $x_i^s x_i^t$ such that the sum of powers equals p. We can write the mapping function for a point x as:

$$\Phi_{r_1, r_2, .., r_d}(x) = \sqrt{(\frac{p!}{r_1! r_2! ... r_d!})} x_1^{r_1} x_2^{r_2} ... x_d^{r_d}, \quad \sum_{i=1}^{d} r_i = p_i, \; r_i \geq 0 \tag{2.42}$$

The same formulation can be written concerning the non-homogeneous polynomial kernels, and because of the constant coefficient we have all the terms of combinations of powers from 0 up to

p. In other words the mapping to the embedding space identified with the use of homogeneous polynomial kernels is

$$\Phi(x) = \Phi(x_1, ..., x_d) = \left\langle w, (x_1^p, ..., x_d^p, x_1 x_2^{p-1}, ..., x_1^{p-1} x_2, ...) \right\rangle \tag{2.43}$$

And the one identified with the use of non-homogeneous polynomial kernels is

$$\Phi(x) = \Phi(x_1, ..., x_d) = \left\langle w, (1, x_1, ..., x_d, x_1^2, ..., x_d^2, x_1 x_2, ..., ..., x_1^p, ...) \right\rangle \tag{2.44}$$

where in both functions one can eliminate symmetric terms to receive the minimal embedding space.

The Sigmoidal Kernels

This family of kernels is equivalent to a two layered neural network and consists of the increasing monotonic function $\tanh(\cdot)$. The kernel function is:

$$K(x^s, s^t) = \tanh\left(\kappa \left\langle x^s, x^t \right\rangle + \delta\right) \tag{2.45}$$

This function satisfies Mercer's condition only for certain values of κ and δ and the data $\|x\|^2$ [4]. This kernel family will not be discussed any further in this work.

2.3 Estimating Performance of SVM

There are many factors determining the quality of a classifier, such as the time consumption of learning and classification. The time consumption of learning can be ignored if the classifier is trained in off-line mode. The classification time in SVM (2.28) is only $O(N_S K)$, where K is the number of operations involved in computing $K(s^i, x)$. For example, RBF kernel (2.38) consumes $O(d)$ operations, where d is the dimension of the data patterns, hence, classification requires $O(N_S d)$ operations [16].

All the above considerations are much less important than the accuracy of the resulting classifier on unseen data, which is also known as the generalization ability of the classifier. This quantity measures the expected rate of error classifications for unseen data drawn from $P(x, y)$ and is given by equation (2.1). As mentioned previously, this quantity cannot be measured due to $P(x, y)$.

Performance estimators try to approximate or bound the expected error rate using the training data. The simplest and most direct estimator is the empirical risk (2.2) and is a simple discretization of the actual risk (2.1). The empirical risk converges to the actual risk asymptotically from beneath with some probability under some conditions [4]. In these cases, the empirical risk and the actual risk are equal for training size $l \to \infty$ with some probability [4]:

$$|R(\theta_l) - R_{emp}(\theta_l)| \xrightarrow[l\to\infty]{P} 0 \tag{2.46}$$

Measuring accuracy of a classifier using a limited set of data can be performed in various ways. The methods we present are well known in machine learning literature.

Validation

A simple method is measuring through validation, where the given data is split into two subsets, a training set and a validation set. The classifier is trained using the training set, then, estimated by testing with the validation set. This method's most obvious shortcoming is the trade-off between the accuracy of the classifier and the accuracy of the estimation.

k-fold Cross Validation

A more accurate, but time consuming method, is the k-fold cross validation, where the data is split into k subsets. Estimating performance of the classifier trained with a complete set of l patterns is performed by training k different classifiers. Each classifier is trained with $k-1$ subsets and tested on the remaining subset. The obtained results are then averaged. In this method the estimation is based on $\frac{k-l}{k}$ training points and $\frac{kl}{k} = l$ testing points over all validations.

Leave One Out

The last and most accurate estimation method is the leave one out (LOO) method, which is the extreme end of k-fold cross validation, where $k = l$. The data is exploited up to the point where estimation is based on $l-1$ training points in average and tested on l points overall.

The leave one out method for performance estimation is heavily time consuming and is not applicable in many cases. Following, is a description of the procedure:

The Leave one out estimation method:

Given: $\{x^i, y^i\}_{i=1}^{l}$

1. Initialize $X = \{x^i\}_{i=1}^{l}$

   $errors = 0$.

2. For $i = 1$ to l do

2.1. $Svm \leftarrow$ Train SVM with training set $X \setminus \{x^i\}$.

2.2. $errors \leftarrow errors + \frac{1}{2}\left(y^i - Svm\left(x^i\right)\right)$

3. $error\_rate \leftarrow errors/l$.

The time complexity can be reduced by testing only the points that were support vectors in the complete set, because they are the only points that might be misclassified. Other points do not affect the solution and, hence, will not be misclassified when trained without them.

Estimating the generalization ability of a classifier can also be done by bounding its risk [4]. There exist upper bounds as well as lower bounds, however, the latter are not as interesting, since we seek to estimate the worst case. Bounding the generalization rate of a learning process from above guarantees certain characteristics on the worst case performance when the bound is tight. Furthermore, we will see that loose bounds can also contribute knowledge concerning the resulting classifier.

We can divide the bounds on the generalization ability into two categories. The first are bounds that do not depend on the distribution of the given training data, but only on the rate of misclassification made on it, thus named distribution independent bounds. The second kind of bounds depends on the distribution of the training data, and produce tighter bounds, thus named distribution dependent bounds. The first category of bounds is much easier to compute since it only depends on the training set size and the number of errors made during the training process. The bounds of the second category use more information depending on the training points and naturally they are more complex to compute.

2.3.1 Distribution Independent Bounds

The empirical risk minimization (ERM) principle (2.2) is not always a good estimator of the actual risk (2.1) of the machine, yet the empirical term is important since we can bound the probability that the distance between the risk to the empirical risk is over some threshold [15]. According to [4][15], when given a set of functions that are totally bounded in a range $[A, B]$ with probability $1-\eta$ the following inequalities hold true:

$$R(\theta) \leq R_{emp}(\theta) + (B-A)\sqrt{\frac{h(\ln(2l/h)+1) - \ln(\eta/4)}{l}} \tag{2.47}$$

$$R(\theta) \geq R_{emp}(\theta) - (B-A)\sqrt{\frac{h(\ln(2l/h)+1) - \ln(\eta/4)}{l}} \tag{2.48}$$

where h is the VC dimension of the set of functions.

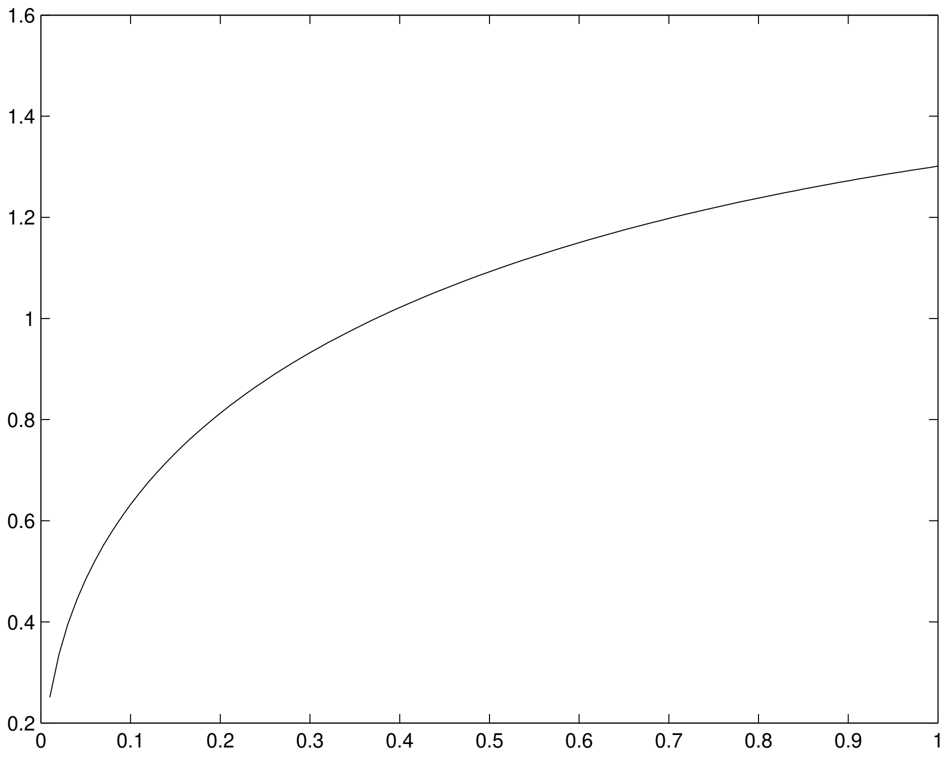

Figure 2.1: The VC confidence term as a function of h/l when $\eta = 0$.

The upper bound (2.47), which is also called the Vapnik-Chervonenkis (VC) bound, consists of two terms that make perfect sense in measuring the fitness of a classifier, the success rate on what we have seen and the ability to generalize well for the unseen. The first term is the empirical risk of the classifier and the second term is called the VC confidence. The VC confidence is proportional to the logarithm of the ratio between the VC dimension and the training set size l. In general, the VC bound is:

$$R(\alpha) \leq R_{emp}(\alpha) + \psi\left(\frac{h}{l}\right)$$

The VC bound is not tight, and is certainly not tight when the VC confidence is larger than one. When the VC dimension is smaller than the training set size, then the bound is not too loose (refer to figure (2.1)).

2.3.2 Distribution Dependent Bounds

A Bound from the Margin

We define a hyperplane

$$< w, x > + b = 0$$

as a γ-margin separating hyperplane if it classifies the given patterns as follows:

$$y = \begin{cases} 1 & if \ < w, x > +b \geq \gamma \\ -1 & if \ < w, x > +b < \gamma \end{cases}$$

The following theorem is then true according to [15]:

Theorem 2.3.1 *Suppose that X is all contained in the ball of radius r in R^d. Then the VC dimension of γ-margin separating hyperplane is upper bounded by the term*

$$min\left\{\left[r^2/\gamma^2\right], d\right\} + 1$$

and since the margin is $\|w\|^{-1}$, the bound can be written as:

$$min\left\{\left[r^2\|w\|^2\right], d\right\} + 1$$

We will refer to this bound as the margin-radius bound.

A Bound from Leave One Out

If we denote by $L(x^1, y^1, ..., x^l, y^l)$ the number of misclassification during the leave one out procedure then

$$E\left(p_{error}^{l-1}\right) = E\left(\frac{L(x^1, y^1, ..., x^l, y^l)}{l}\right) \tag{2.49}$$

Considering iteration i, if the point (x^i, y^i) is a support vector in the SVM trained with l points then it might be misclassified due to the fact that it lies on the boarder of the margin. However, if (x^i, y^i) is not a support vector then its removal will not affect the resulting decision surface and it will not be misclassified. Hence, we can bound the probability of test error by the following theorem [4]

Theorem 2.3.2 *If the training vectors are separated by the optimal hyperplane, then the expectation value of the probability of committing an error on a test example is bounded by the ratio of the expectation of the number of support vectors to the number of examples in the training set*

$$E\left(p_{error}^{l-1}\right) = E\left(\frac{L(x_1, y_1, ..., x_l, y_l)}{l}\right) \leq \frac{E\left[Number\ of\ Support\ Vectors\right]}{l}$$

The bound on the generalization ability was made even tighter by Vapnik [15] in the following theorem, which is a result of theorem (2.3.1) and theorem (2.3.2)

Theorem 2.3.3 *If training sets containing l examples are separated by the maximal margin hyperplanes, then the expectation (over training sets) of the probability of test error is bounded by the expectation of the minimum of three values: the ratio m/l, where m is the number of support vectors, the ratio $[r^2|w|^2]/l$, where r is the radius of the smallest sphere containing the data and $|w|^{-1}$ is the value of the margin, and the ratio d/l, where d is the dimensionality of the space.*

$$E\left(p_{error}^{l-1}\right) \leq E\left[\min\left(\frac{m}{l}, \frac{[r^2|w|^2]}{l}, \frac{d}{l}\right)\right]$$

This theorem gives multiple reasoning to learning machines that generalize well. The generalization is better when the compression of data is large$\left(\frac{m}{l}\right)$, when the margin is large $\left(\frac{[r^2|w|^2]}{l}\right)$ and when the dimension of the input space is small $\left(\frac{d}{l}\right)$. In many cases the second term can be a tighter bound on the generalization ability of a SVM, since the algorithm is shown to maximize the margin. However, computing this bound requires the value of the radius of the smallest enclosing sphere of the data, which can be found by solving an optimization problem much like SVM itself [16] (refer to appendix (B)).

A Bound from the Span

The bound given in theorem (2.3.3) was improved once more by Vapnik and Chapelle in [13] using a new geometric concept called the span of the support vectors. One has to define at first for any support vector x^p a set Λ_p as a constrained linear combinations of the remaining points $\{x^i\}_{i \neq p}$

$$\Lambda_p = \left\{ \sum_{i=1, i\neq p}^{n} \lambda_i x^i : \sum_{i=1, i\neq p}^{n} \lambda_i = 1 \; and \; \forall i \neq p \; \alpha_i^0 + y^i y^p \alpha_p^0 \lambda_i \geq 0 \right\}$$

where λ_i are real coefficients (that might be negative). Then the span of support vector x^p is defined as the distance between the support vector and the set Λ_p

$$S_p^2 = d^2(x^p, \Lambda_p) = \min_{x \in \Lambda_p} (x^p - x)^2$$

We can rewrite the span of support vector x^p by setting $\lambda_p = -1$

$$S_p^2 = \min \left\{ \left(\sum_{i=1}^{n} \lambda_i x^i \right)^2 : \lambda_p = -1, \sum_{i=1}^{n} \lambda_i = 0, \alpha_i^0 + y^i y^p \alpha_p^0 \lambda_i \geq 0 \right\}$$

and the S-Span is defined as

$$S = \max\left\{ d(x^1, \Lambda_1), ..., (x^n, \Lambda_n) \right\} = \max_p S_p$$

The same definitions hold for the non-separable case, where only support vectors with $\alpha_i : 0 < \alpha_i < C$ participate. It was proved in [13] that $S \leq D_{SV}$, where D_{SV} is the diameter of the smallest sphere containing the support vectors (the leave one out estimator is carried only over the support vectors). Following the above definition of the span the following theorem holds true

Theorem 2.3.4 *The expectation of the probability of error p_{error}^{l-1} for a SVM trained on training data of size $l-1$ has the bound*

$$Ep_{error}^{l-1} \leq E\left(\frac{S\max(D, 1/\sqrt{C})\sum_{i=1}^{n^*}\alpha_i^0 + m}{l}\right)$$

where α_i, $1 \leq i \leq n^*$ are multipliers of support vectors that satisfy $0 < \alpha_i < C$, and m is the number of the support vectors for which $\alpha_i = C$. S is the span of the support vectors, D is the diameter of the smallest sphere containing the training points and the multipliers $\alpha_1^0, ..., \alpha_n^0$ are considered for SVM trained on training sets of size l.

Since $S \leq D$, this bound is an improvement of the previous bound given in theorem (2.3.3). This can be seen in the separable case as follows: we have $m = 0$ since there are no bounded support vectors, and $n^* = n$, hence,

$$Ep_{error}^{l-1} \leq E\left(\frac{S\max(D, 1/\sqrt{C})\sum_{i=1}^{n^*}\alpha_i^0 + m}{l}\right) = E\left(\frac{SD\sigma_{i=1}^{n}\alpha_i^0}{l}\right) \leq E\left(\frac{D^2\sigma_{i=1}^{n}\alpha_i^0}{l}\right) = 4E\left(\frac{R^2|w|^2}{l}\right)$$

which means that the span bound is not worse than four times the margin radius bound.

2.4 Model Selection in SVM

A family of classifiers is complex when the set of free (tunable) parameter is large. Once the family of kernel functions is chosen for SVM, the only free parameters are the kernel parameter (parameters) and the capacity control parameter. Model selection is the procedure of seeking the optimal parameters for a classifier. An intuitive implementation is scanning a reasonable range of possible values. The outcome of the procedure is the best classifier depending on the parameter value. In this context, we view model selection as a search in some m-dimensional space, where m is the number of free parameters.

The optimal parameter (parameters) for a classifier is the one that yields the classifier with the best generalization ability. Good estimates of the classification error are time consuming, hence, usually replaced by upper bounds. There are many distribution dependent bounds on the test error of SVM. We use two bounds: The margin-radius bound and the span bound.

The margin-radius bound depends on the margin, which SVM maximizes and the radius of the smallest enclosing sphere in feature space, which depends on the data itself as much as on the kernel. The span bound is derived directly from the distribution of the training points in feature space. These bounds are good for model selection, since they behave similarly to the leave one out estimate of the test error [11][16][13][14]. We also support this observation in the next chapter, where we show a procedure that determines experimentally that these bounds are better than others.

2.4.1 Model Selection in RBF SVM

Let us assume that the only tunable parameter in RBF SVM is the Gaussian's width denoted by σ . The naive procedure of model selection in the range $[\sigma_i, \sigma_f]$ is as follows:

Naive model selection in RBF SVM

Given: A training set T and a validation set V.

1. Initialize: $\sigma \leftarrow \sigma_i$
   $minimal\_error\_rate \leftarrow 1$
   $\sigma^* \leftarrow \sigma_i$
2. While $\sigma \leq \sigma_f$
   2.1. $Svm \leftarrow$ Train a SVM with RBF kernel K_σ and training set T
   2.2. $error\_rate \leftarrow Svm(V)$
   2.3. if $error\_rate < minimal\_error\_rate$
      2.3.1. $minimal\_error\_rate \leftarrow error\_rate$
      2.3.2. $\sigma^* \leftarrow \sigma$
   2.4. $\sigma \leftarrow \sigma + \delta\sigma$
3. return σ^*.

The above procedure has two obvious shortcomings, which make it computationally infeasible. The first one is the validation of each classifier and the second is the number of classifiers to be trained. Additionally, it becomes infeasible when applied to tuning more than 2 parameters [14]. [11] and [14] address these problems only for the case of RBF SVM. The first work shows that the performance of RBF SVM is a smooth and continuous function of σ. Thus, for quick convergence of SVM with σ start with the final state of SVM with $\sigma - \delta\sigma$. Furthermore, use an error bound with a good model selection ability for measuring the SVM performance instead of a validation procedure to reduce the time complexity.

Considering all the above arguments, [11] suggests a better procedure for the parameter tuning of RBF SVM:

Model selection using a risk bound in RBF SVM

Given: A training set T and a bound $B(\cdot)$.

1. Initialize: $\sigma \leftarrow \sigma_i$

   $minimal\_error\_rate \leftarrow \infty$

   $\sigma^* \leftarrow \sigma_i$

2. While $\sigma \leq \sigma_f$

   2.1. $Svm \leftarrow$ Train a SVM with RBF kernel K_σ and training set T

   2.2. $error\_rate \leftarrow B(Svm)$

   2.3. if $error\_rate < minimal\_error\_rate$

      2.3.1. $minimal\_error\_rate \leftarrow error\_rate$

      2.3.2. $\sigma^* \leftarrow \sigma$

   2.4. $\sigma \leftarrow \sigma + \delta\sigma$

3. return σ^*.

Scanning a wide range $[\sigma_i, \sigma_f]$ can still be infeasible when tuning multiple parameters. [14] suggests a more general framework that relies on the smoothness and continuity of the performance as a function of σ. Compute the derivative of the bound as a function of each tunable parameter. It requires the bound to be smoothed in some cases (The span bound [14]). Then choose the parameters for which the derivatives vanish.

In this framework it is possible to tune multiple parameters simultaneously with low computational cost for more than 2 parameters. In RBF SVM, this is very helpful for determining the values of σ and C together in a gradient descent algorithm of the bound estimate in (σ,C)-space.

Chapter 3

Polynomial Model Selection for Learning Relations

3.1 Choosing the Best Bound

The computations and comparisons between various upper bounds we have shown in the previous chapter are not conclusive. We offer a mechanism for determining the best bound. However, this task is not simple and is related to the target of the needed estimation. For example, we might have a certain classifier (or classifiers) and we only need to estimate its performance. In this case we would want to use the tightest bound possible. On the other hand, in some situations we might want to choose a classifier out of a family of possible classifiers, which is the best. In the latter case we would want to use a bound that has the most similar behavior to the actual risk. The best classifier according to the bound is then, with high probability, close to the true best classifier.

Given a family of classifiers $\{\alpha\}$ and a training set $S = \{x^i, y^i\}_{i=1}^n$ we can measure the tightness of the bound as

$$\int_\alpha L\left(R_{actual}(S), R_{bound}(S)\right) d\alpha$$

where R_{actual} and R_{bound} are the actual risk and the bound estimation respectively. $L(\cdot,\cdot)$ is some loss function.

On the other hand, the model selection ability can be measured by

$$\int_\alpha L\left(\frac{d}{d\alpha}R_{actual}(S), \frac{d}{d\alpha}R_{bound}(S)\right) d\alpha$$

This can be generalized introducing a weight coefficient $\lambda \in [0,1]$ as

$$\lambda \int_\alpha L\left(R_{actual}(S), R_{bound}(S)\right) d\alpha + (1-\lambda)\int_\alpha L\left(\frac{d}{d\alpha}R_{actual}(S), \frac{d}{d\alpha}R_{bound}(S)\right) d\alpha$$

where the extreme values of λ measure only tightness ($\lambda = 1$) and only model selection ability ($\lambda = 0$).

The above computations give us estimations on how good is a bound in comparison to other bounds. Our experiments show that distribution dependent bounds that are more time consuming produce better results than distribution independent bounds. In most cases simpler bounds are looser and not as informative.

We note here that we used this procedure to determine the bound we apply before each experiment. The results show that distribution dependent bounds achieve minimal values of the functional. This formulation requires the computation of the leave one out error, which we want to avoid. Hence, our only conclusion from the obtained results is a strong support of the observation made in [14], which is that there is no favorable bound among the distribution dependent bounds we use.

3.2 Model Selection in Polynomial SVM

Suppose we are seeking the best choice of kernel when the given family is $k(x,y) = (< x, y > +1)^p$ and the range of kernel parameters from which we are seeking an optimal parameter is $[p_{min}, p_{max}]$. Choosing the best parameter requires to iterate on $p \in [p_{min}, p_{max}]$ and for each step to train an appropriate SVM with the specific parameter. A validation set or alternatively a risk bound is then used to evaluate the classifier's performance. Finally, the chosen parameter is the one that yields the best SVM.

The risk bounds are still helpful and support model selection independently from the kernel family for the same reasoning used in the RBF case. In addition, our experiments support this claim. The procedure is as follows:

Model selection using a risk bound in polynomial SVM

Given: A training set T and a bound $B(.)$.

1. Initialize: $p \leftarrow p_i$

    $total\_error\_rate \leftarrow 1$

    $p^* \leftarrow p_i$

2. While $p \leq p_f$

    2.1. $S \leftarrow$ Train a SVM with polynomial kernel K_p and training set T

    2.2. $error\_rate \leftarrow B(S)$

    2.3. if $error\_rate < total\_error\_rate$

        2.3.1. $total\_error\_rate \leftarrow error\_rate$

        2.3.2. $p^* \leftarrow p$

    2.4. $p \leftarrow p + 1$

3. return p^*.

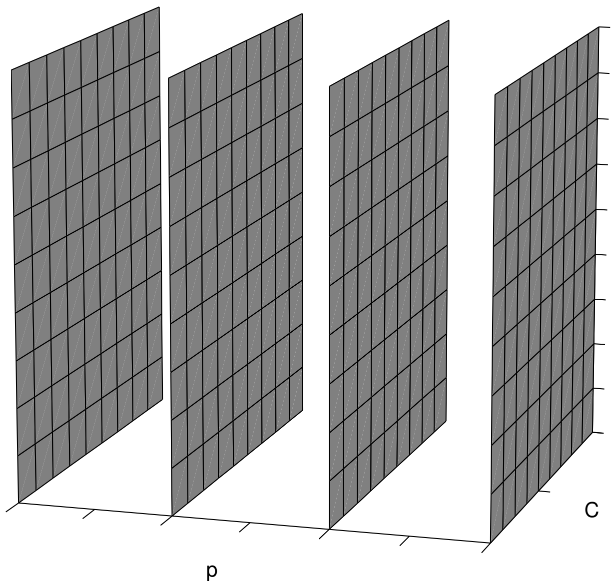

Figure 3.1: (p, C)-search-space, where p is polynomial degree and C is the penalty parameter.

Let us consider the case where the data is generated according to some unknown polynomial relation between its features. We can use the model selection procedure for relation prediction. However, it prevents us from some of the advantages described in [11] and [14] just for the reason that the parameter range is discrete. [14] shows that the performance of the produced classifier is a smooth function of two tunable parameters, which are the Gaussian width and the penalty factor (σ, C). In polynomial SVM, we perform a 2-dimensional search in the (p, C)-space. One of the parameters is discrete and the other is continuous. Consequently, we reduce the search to a 1-dimensional search in the discrete p-space when it is performed among the best results achieved in searches in C-space for each scanned value of p. Unfortunately, we are not able to reduce time consumption by using the final state of each SVM training as an initial state for the next SVM training. Figure (3.1) describes the $2D$ search space, and figure (3.2) shows the search procedure. The optimal values in each C search participate in the p search.

We now explore whether model selection can supply information concerning the relations between the features of given patterns. Assuming the rule that generated that data is polynomial, would polynomial model selection yield best results for the degree of that rule? Answering this question can be very helpful for discovering nature rules of polynomial qualities.

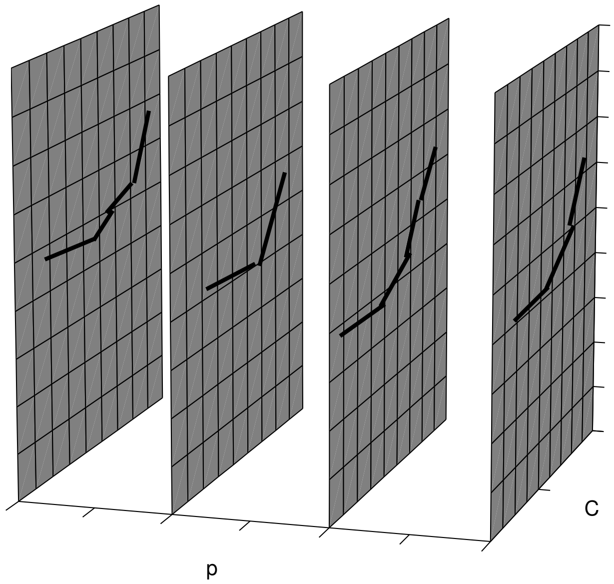

Figure 3.2: (p, C)-search space. two 1D searches. For each p, search best C, then chose best result.

3.3 Predicting the Degree of the Relations

Mapping the generating polynomial into the appropriate feature space using a polynomial kernel with its degree will make at least one of its projections linear. Thus, data generated by this rule is linearly separable in this space. We use this observation as a basis for detecting the relation's degree. Theoretically, using the kernel

$$K(x, y) = (< x, y > +1)^p$$

yields a surface of the form

$$f(x) = \sum_{i=1}^{\#SV} \alpha_i y^i K(x, s^i) + b = 0$$

which is a polynomial equation of degree p with given support vectors, where $\#SV$ is the number of the support vectors. The separating hyperplane is

$$f(x) = f(x_1, x_2, ..., x_n) = c^{(p,0,0,...,0)} (x_1)^p + c^{(p-1,1,0,...,0)} (x_1)^{p-1} x_2 + ... + c^{(1,1,...,1)} x_1 x_2 ... x_n + ... + c^{(0,0,...,0)} = 0 \quad (3.1)$$

where $c^{(\cdot)}$ are constants depending on $\{\alpha_i\}$, $\{y^i\}$ and polynomialy on the set of support vectors $\{s^i\}$.

Suppose we are given a sample set that was generated by an unknown polynomial relation between the features of the patterns. Polynomial kernel SVM with a degree lower than the generator's degree results in low performance. The separation of that SVM is not able to express the generating rule because the relations between the features are of a higher degree than the used in (3.1). On the other hand, a polynomial kernel SVM with a degree that is higher than the generator's degree produces a complex separating surface and the generalization ability decreases. Thus, the optimal SVM should use the polynomial kernel with the generator's degree. Its ability depends on the freedom of choice of the constants $c^{(\cdot,\cdot\cdot\cdot,\cdot)}$. Hence, the best results of polynomial SV machines reveal the degree of the generator. This supplies a method for discovering the polynomial degree, when given only a sample set generated according to this relation.

Of course, these assumptions sound valid and logical only for the hard margin SVM, where the resulting separator leaves no misclassified training points. In real life problems this assumption will not hold. We suggest using a penalty factor C that is large enough. Determining whether C is large enough is done by one of three options: An experimental search, choosing very large values, or demanding the derivative of a smooth bound as a function of C to vanish [14].

We have performed several experiments on synthetic data that demonstrate the search for the polynomial degree of the generating rule. The expected results can be seen on various experiments with different training sets generated according to rules of increasing degrees 2,3 and 4:

$$x_1^2 + x_2^2 = 1 \tag{3.2}$$

$$0.1x_1^3 + 0.1x_2^3 + x_1^2x_2 = 0.1 \tag{3.3}$$

$$x_1^4 +_2^4 +x_1x_2^3 = 3 \tag{3.4}$$

Each training set contains 3000 2-dimensional points labeled according to the appropriate generating rule. Subsets of 300 points with the true separators are plotted in figures (3.3), (3.4) and (3.5) respectively.

In each experiment, we divide the 3000 points data set into 10 different subsets of 300 points, train an SVM with each one of the first 9 subsets alone and test on the last subset. The results are then averaged over the 9 training procedures. The penalty parameter is tuned to $C = 1000$ and it was seen experimentally to be large enough.

The results of the model selection procedures are plotted in figures (3.6), (3.7) and (3.8) respectively. One can see that in each experiment the minimal values of the bound as well as

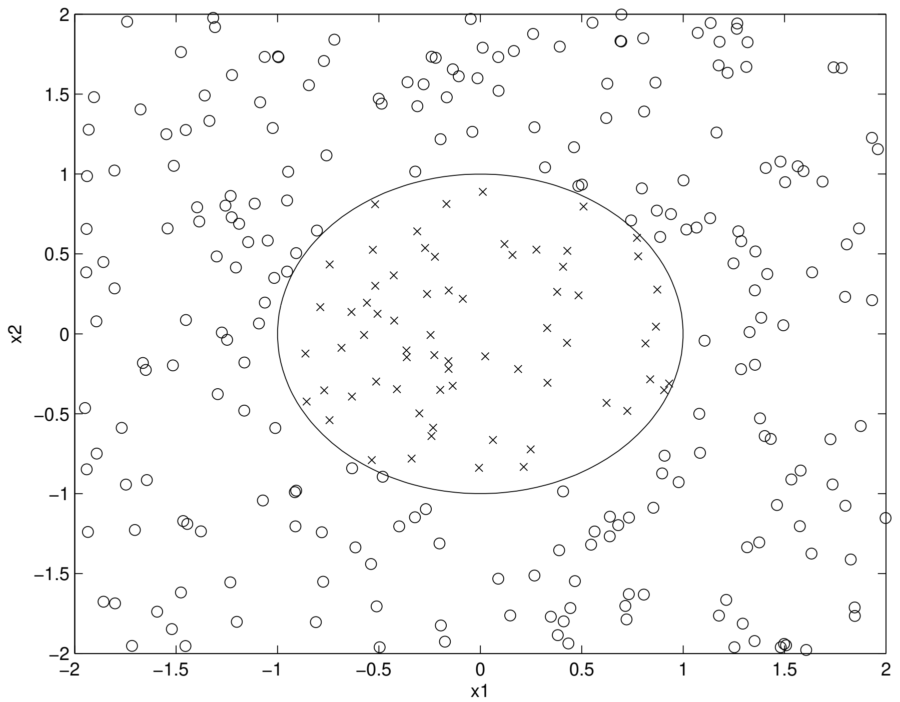

Figure 3.3: 2D set of 300 points generated according to equation (3.2).

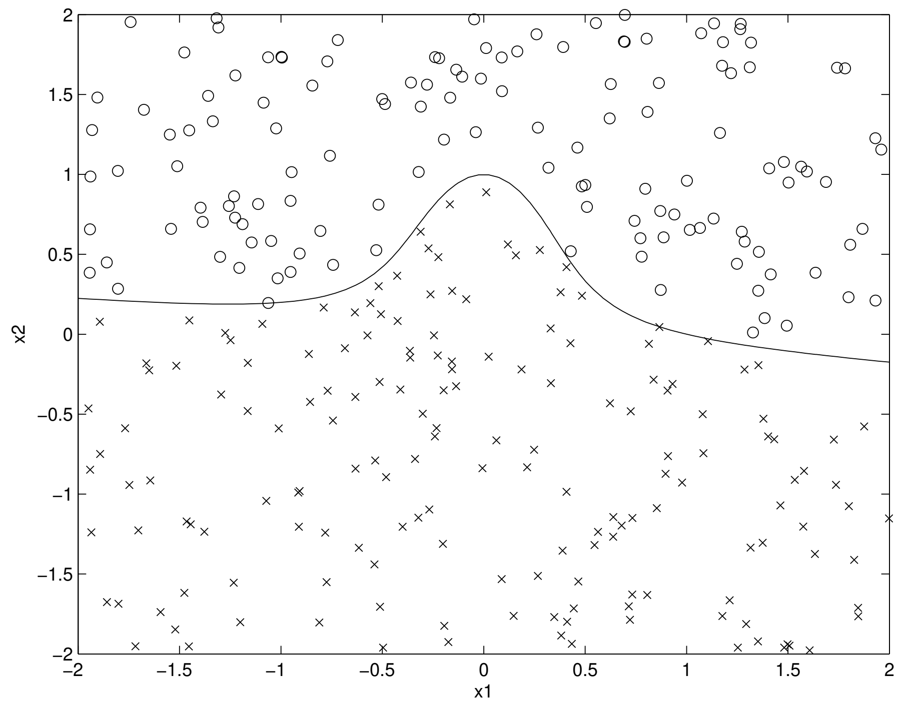

Figure 3.4: 2D set of 300 points generated according to equation (3.3).

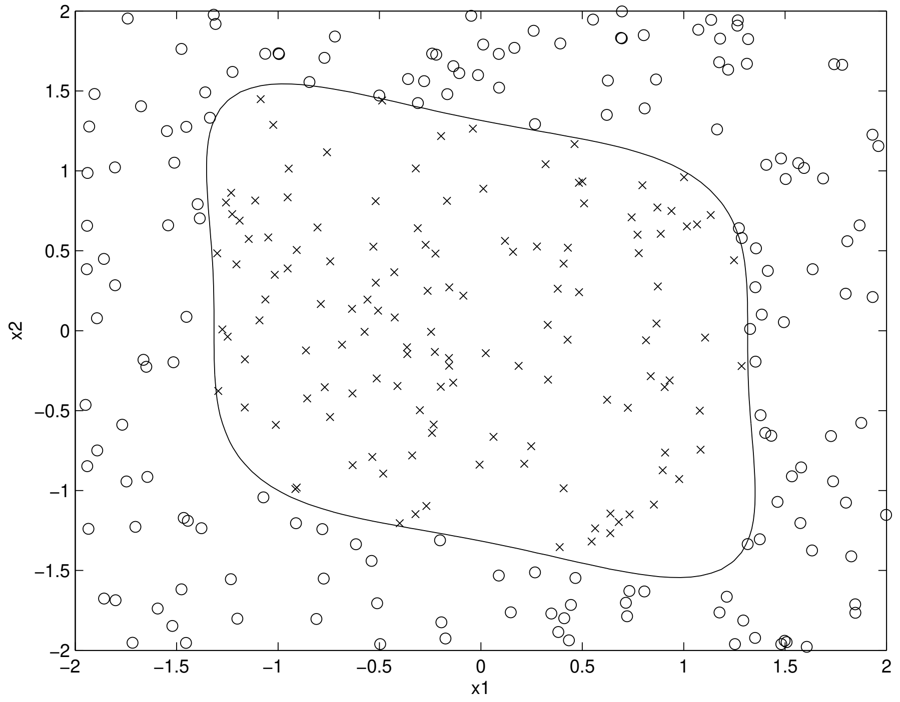

Figure 3.5: 2D set of 300 points generated according to equation (3.4).

leave one out errors are achieved for the appropriate polynomial degree. Hence, we successfully predict the degree of the true generator.

Furthermore, we have experimented with real life data and performed model selection on it. The "Letter Image Recognition Data", a 16-dimensional data set of features of images of letters from A to Z in a 4x4 pixels images. We perform a one-against-all testing on a 20000 points set by dividing it to 10 subsets, training a SVM on each subset from the first 9 and testing with the last subset. The letter we try to separate from the others is "A". The results are shown in figure (3.9), where we plot the natural logarithm of the leave-one-out error (dashed line) and the natural logarithm of the margin-radius bound divided by 1000 (solid line). The results show that the best performance was achieved with a polynomial kernel of degree 3 and was correctly predicted by the margin-radius bound.

The letter recognition experiment results do not determine that the relation between the features that separates the letter "A" from all other letters are polynomial of degree 3. The assumption that the relation is polynomial is probably incorrect in the first place. However, we learn that the best polynomial approximation of the separator between "A" and the rest of the alphabet is of degree 3.

In section (3.5) we show a case were we know for sure that the relation between the features of the patterns is polynomial.

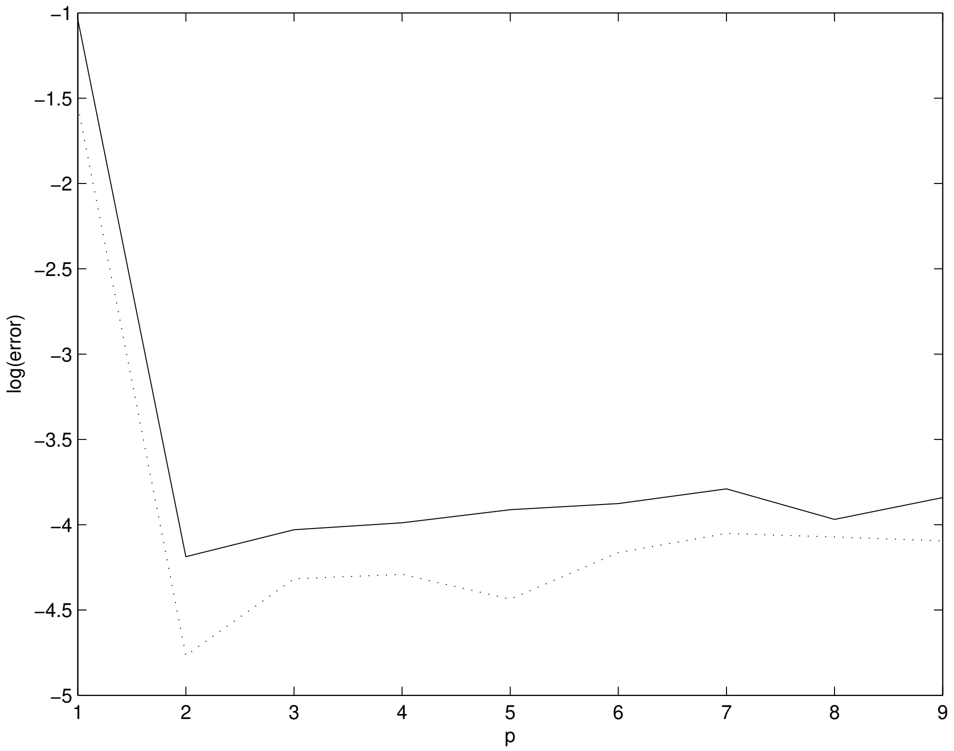

Figure 3.6: Natural logarithm of span bound error (solid) and leave one out error (dashed) for data generated by equation (3.2) vs. polynomial kernel degree. Best result achieved for $p = 2$.

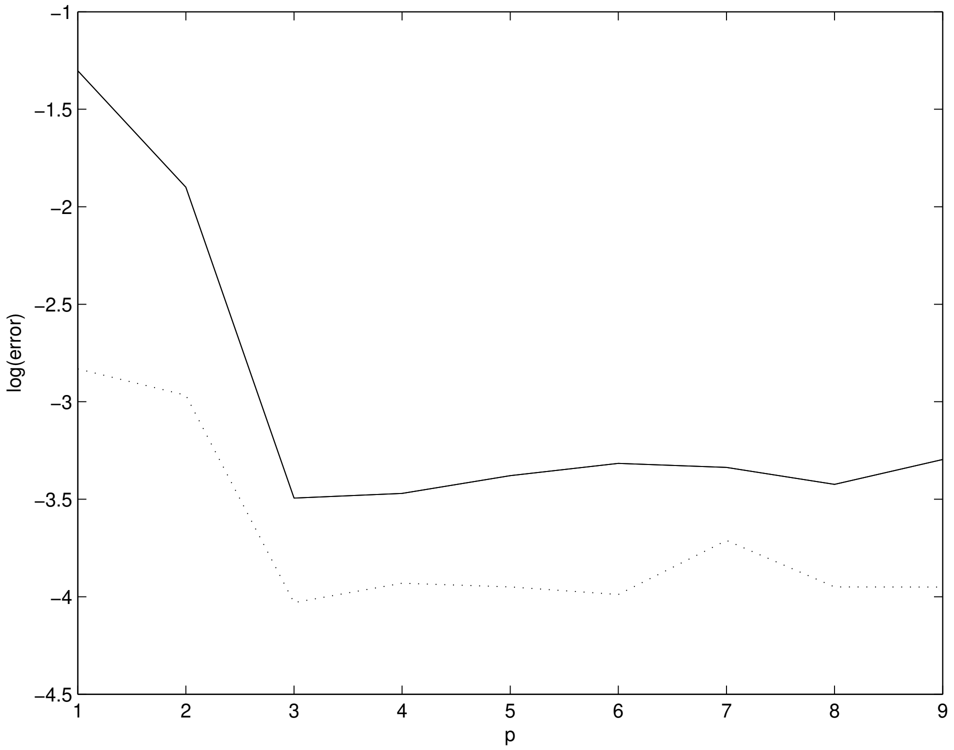

Figure 3.7: Natural logarithm of span bound error (solid) and leave one out error (dashed) for data generated by equation (3.3) vs. polynomial kernel degree. Best result achieved for $p = 3$.

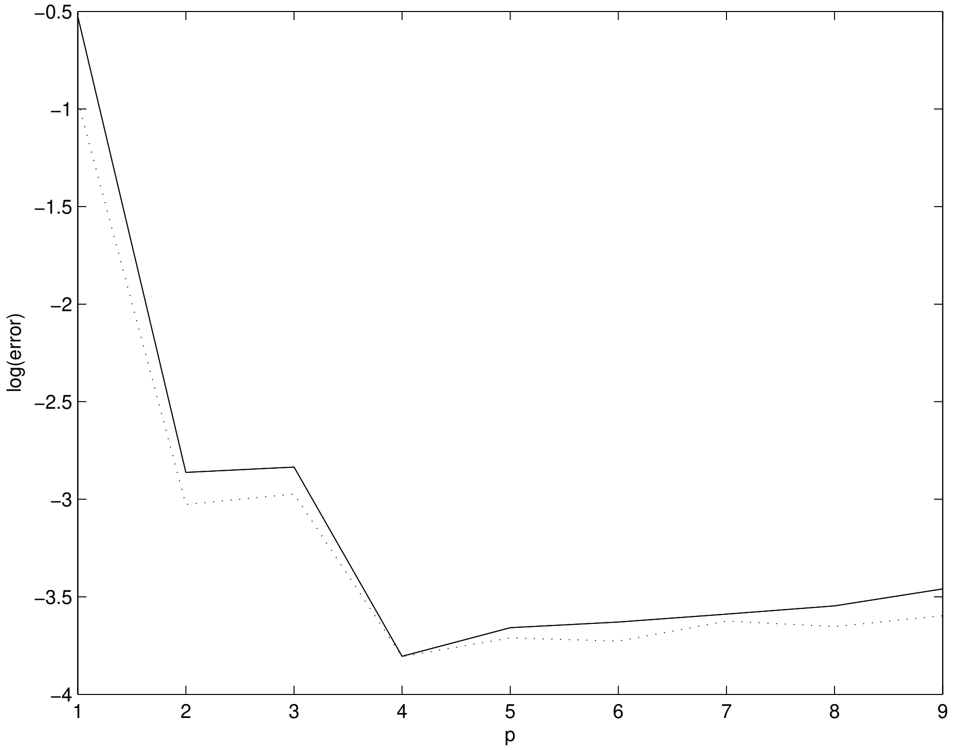

Figure 3.8: Natural logarithm of span bound error (solid) and leave one out error (dashed) for data generated by equation (3.4) vs. polynomial kernel degree. Best result achieved for $p = 4$.

3.4 Convergence of the Separator's Coefficients

We have seen in the previous section that we can learn the degree of the polynomial generating rule that labels the data patterns at hand. Subsequently, we can use the general formula for a polynomial equation of the found degree and estimate the coefficients. Unfortunately, the parameters $c^{(\cdot,\cdot,\ldots,\cdot)}$ are not free, since they depend on the support vectors, hence the resulting separating hyperplane does not converge to the true separating rule. The existing support vectors do not allow the freedom for some coefficients to converge. Nevertheless, some observations can be made regarding this issue. For this purpose we refer to the previous experiments, where the generating degree was found, and analyze the behavior of the coefficients of the resulting monomials. We normalize the different separators produces by each training as follows:

$$c^{(a_1,a_2,\ldots,a_h)} = \frac{c^{(a_1,a_2,\ldots,a_h)}}{\sum c^{(\cdot,\cdot,\ldots,\cdot)}} \tag{3.5}$$

The normalized separators become comparable as the coefficients are mapped to an appropriate unit ball in feature space.

We use the data sets from equations (3.2) and (3.4). For each data set we use the polynomial kernel SVM with the best degree found and train multiple SV machines with increasing training

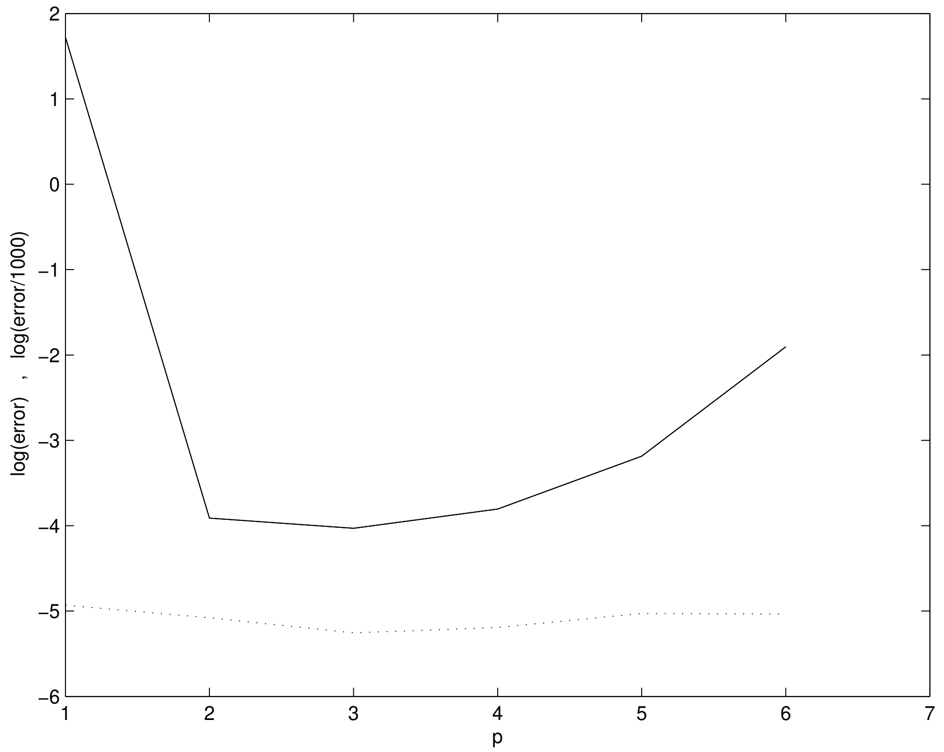

Figure 3.9: Natural logarithm of leave one out error (dashed line) and natural logarithm of margin-radius bound divided by 1000 (solid line) vs. polynomial kernel degree for letter image recognition ("A" against all) on average of 9 training sets of 2000 points and one test set of 2000.

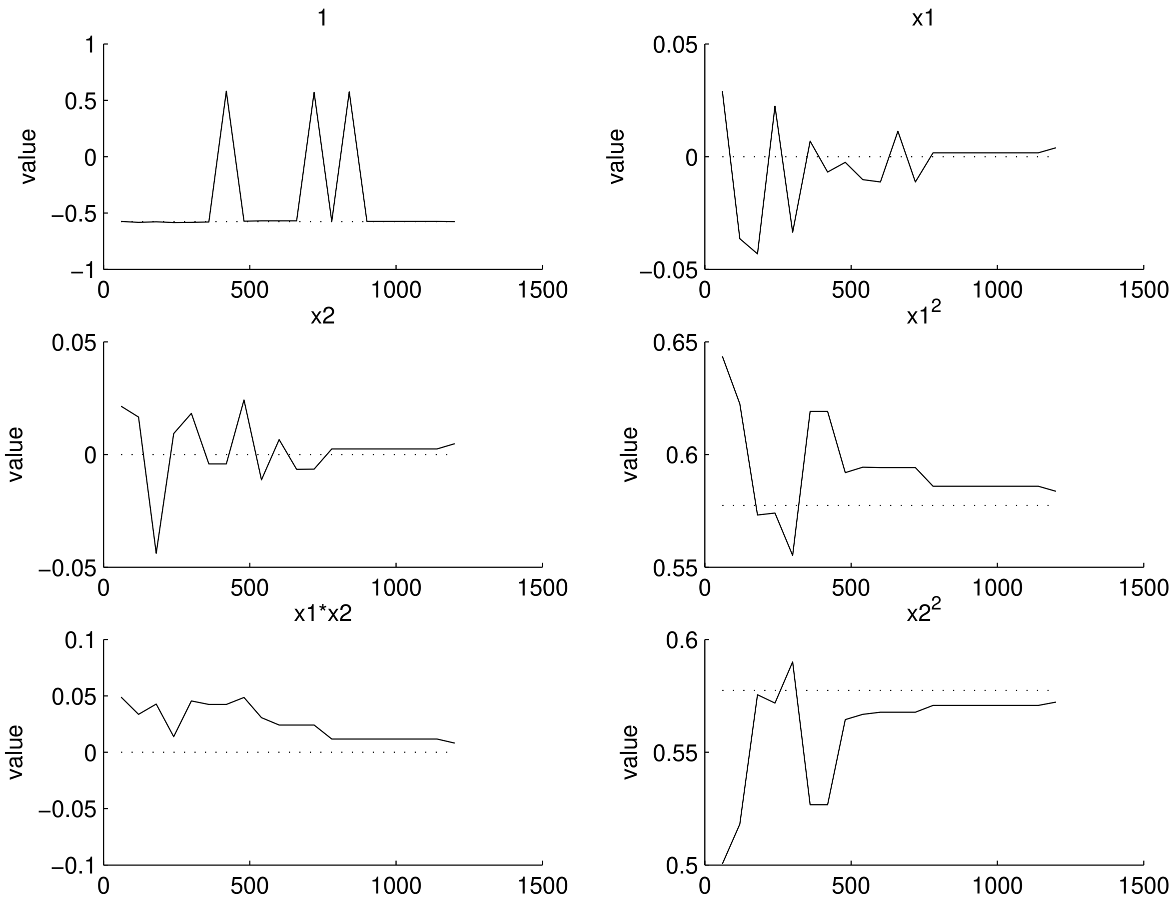

Figure 3.10: Coefficients of the resulting separator for different trainings of polynomial SVMs with $p = 2$ on nested subsets (solid line) compared to the values of the generator function (3.2).

sets. We start with a training set containing 60 data points, and each step we add another set of 60 points. In each training we normalize the resulting separating hyperplane according to (3.5) and plot the coefficients of each monomial.

The results are plotted in figure (3.10) for the first experiment and in figure (3.11) for the second experiment. In each graph we have the value of the coefficient of some monomial (solid line) and the true value for this coefficient used in the generating rule (dashed line). For example, the monomials for the second degree are $1, x_1, x_2, x_1^2, x_1x_2, x_2^2$. The results are very interesting. SV machines built with nested training sets tend to converge to the true coefficients of the generating rule and they tend to not make the separating hyperplane worse.

We notice that, excluding some singular cases, it is possible to find a good approximation of the generating rule by training a few SVMs with different number of points after discovering the polynomial degree via model selection when trained with some minimal training set sizes. Furthermore, the singular cases can be dealt with easily by checking the values for a few nested training sets and not to rely on one experiment.

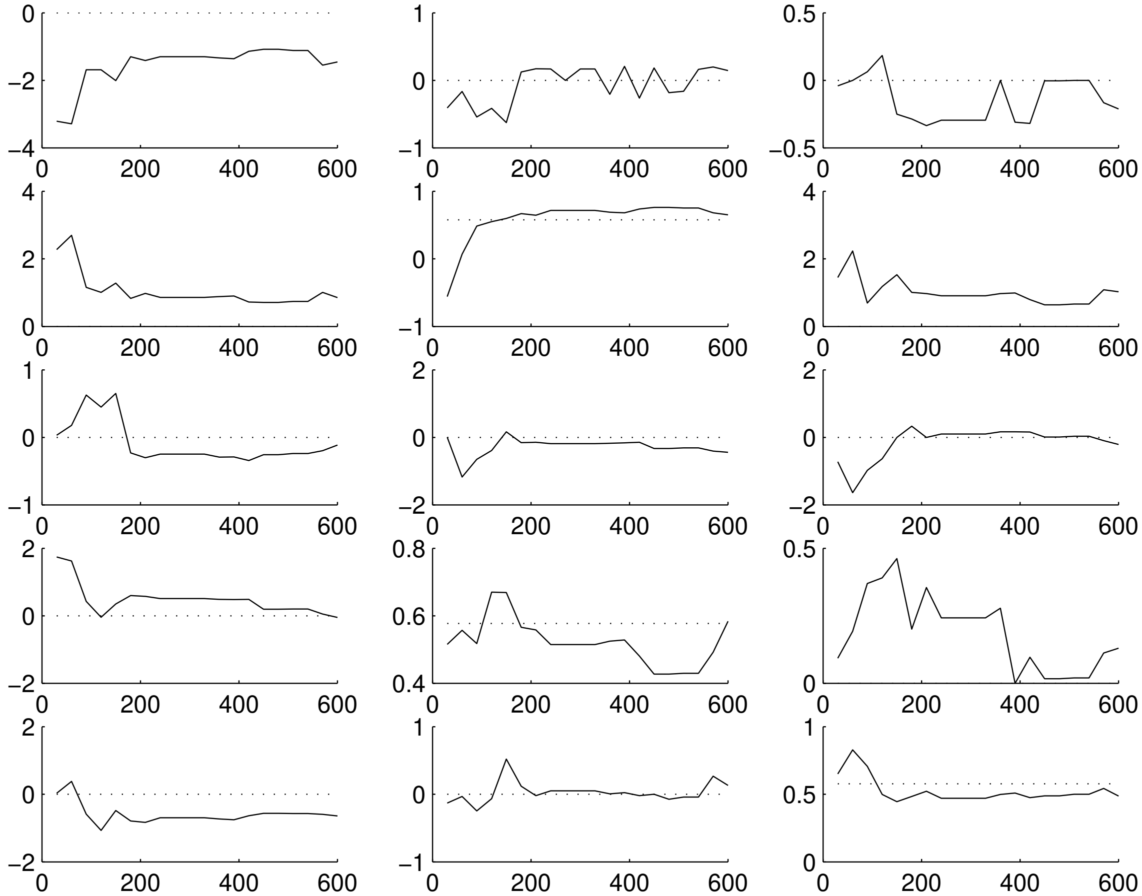

Figure 3.11: Coefficients of the resulting separator for different trainings of polynomial SVMs with $p = 4$ on nested subsets (solid line) compared to the values of the generator function (3.4).

Alternatively, once the correct polynomial degree is found, one can apply kernel principle component analysis (kPCA) [20] and find the relative contribution of each monomial to the separator.

3.5 2D Circle Recognition

The use of polynomial SVM model selection that we suggested in the previous sections is most fruitful when applied to problems that are known to be of polynomial nature. We are able to identify the degree of the relation between the features as well as the tendency of the coefficients to the true weights of the monomials in that relation.

The task we have chosen is the 2D circle recognition problem from the field of shape prototyping. The example deals with 2-dimensional data that is classified into two classes: circles and non-circles. For this purpose, suppose the shapes are preprocessed into vectors of various features we think might be relevant to the task. The irrelevant features are then excluded by a feature selection procedure. In this case, we know that the relevant features are the perimeter, denoted by P and the area, denoted by A, of a shape. Geometrically we know that the following properties hold true for any circle and only for circles:

$$A = \pi R^2$$
$$P = 2\pi R$$
$$\frac{A}{P^2} = \frac{1}{4\pi}$$
$$A = \frac{1}{4\pi} P^2$$

any other shape satisfies:

$$\frac{A}{P^2} < \frac{1}{4\pi}$$

This means there is a polynomial relation between the features of a 2D connected shape that separates circles from any other shape.

Solving this problem requires an oracle that labels the given shapes. The direct way is to give exact labeling to shapes using exact computations. However, for some real life problems this might be impossible, because sometimes the labeling of observations is inaccurate. In other cases the observations might be noisy. We present solutions for four different problems:

1. The observations are labeled with constant additive noise.
2. The observations are labeled with constant multiplicative noise.
3. The observations are labeled without noise.

4. The observations are labeled based on average human decision.

In the next four sub-sections we present the results of the four cases.

3.5.1 Case I - The Additive Constant Mistake Factor

If we assume that the oracle that labeled the training patterns is an expert who has a certain amount of constant inaccuracy factor, which we denote ϵ, then we can define a circle according to that expert to satisfy:

$$-\epsilon \leq A - \frac{1}{4\pi}P^2 \leq \epsilon$$

Geometrically, it is known that the upper bound is irrelevant since there are no connected geometric 2D shapes that satisfy:

$$A > \frac{1}{4\pi}P^2$$

Hence, a circle will be defined to satisfy:

$$-\epsilon \leq A - \frac{1}{4\pi}P^2 \leq 0$$

In addition, we assume that the oracle cannot classify shapes that fall in a margin defined by a positive constant δ:

$$-\epsilon - \delta \leq A - \frac{1}{4\pi}P^2 \leq -\epsilon$$

Consequently, the generator of shapes generates two classes, circles and non-circles according to the following rule:

$$\begin{aligned} Circles: \ & -\epsilon < A - \frac{1}{4\pi}P^2 < 0 \\ Non-Circles: \ & A - \frac{1}{4\pi}P^2 \leq -\epsilon - \delta \end{aligned}$$

The inaccuracy of the data generator in this case is insensitive to the size of the shape, as depicted in figure (3.12). Shapes that fall between the two dashed lines are not labeled and not used for training. Other remaining data patterns are used to train the classifier. Figure (3.13) depicts a randomly generated set of 300 shapes.

The result of polynomial model selection is shown in figure (3.14), were the span bound is plotted vs. the kernel parameter p. The minimal error estimation is achieved for $p = 2$,

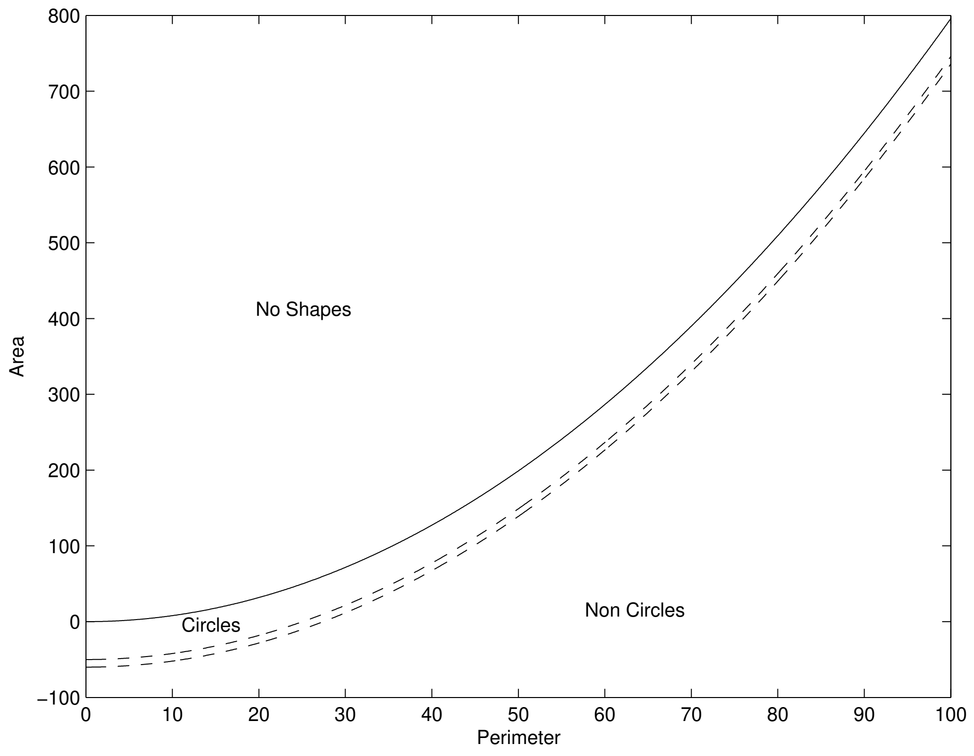

Figure 3.12: The classification regions of a generator with a constant additive mistake factor. The shapes between the two dashed borders are unlabeled and will be ignored.

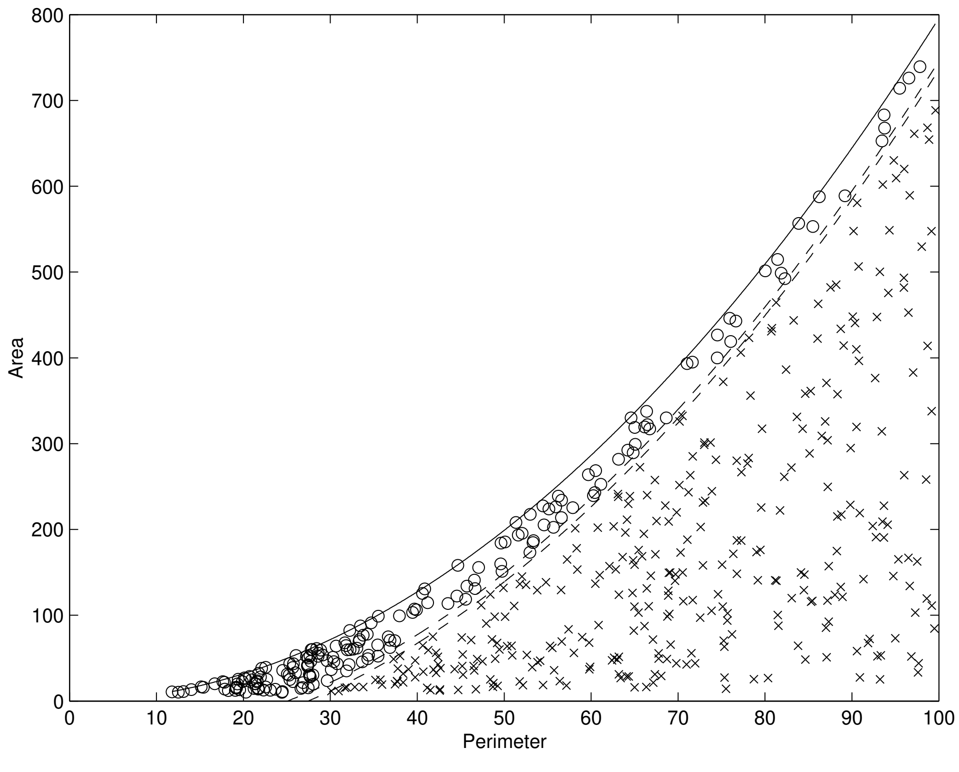

Figure 3.13: A randomly generated set of shapes (perimeter and area) according to a generator with an constant additive mistake factor.

predicting that circle recognition depends on a quadratic relation between the perimeter and the area.

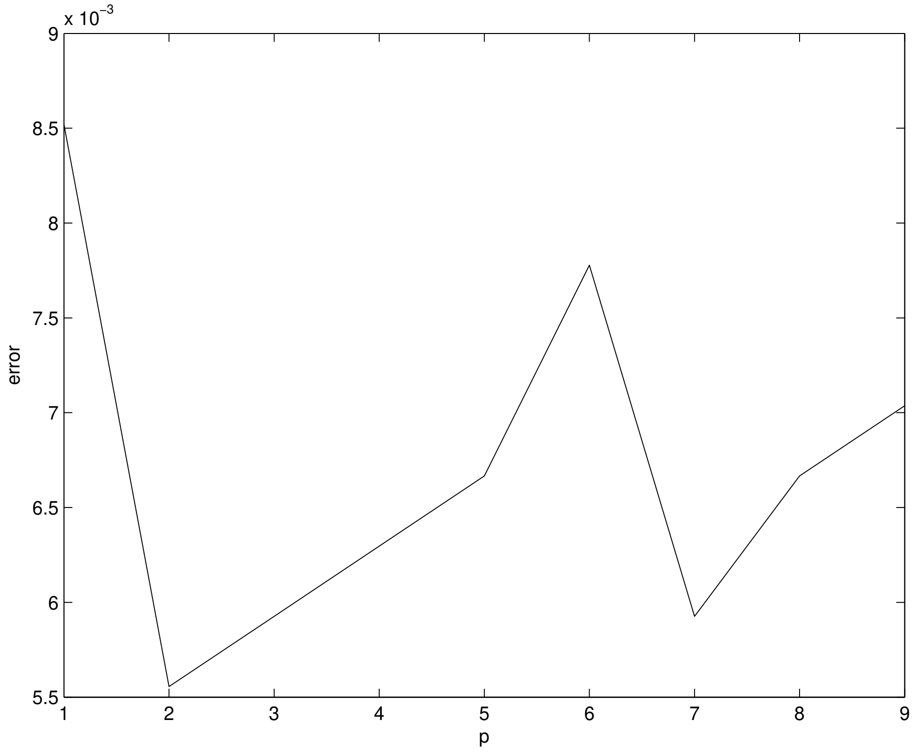

Figure 3.14: The span bound vs. the polynomial kernel degree for the 2D circles recognition problem with a constant additive mistake factor. The minimum of the span bound is achieved for a polynomial degree 2

3.5.2 Case II - The Multiplicative Constant Mistake Factor

In case the oracle has a multiplicative constant noise in decisions, the inaccuracy is sensitive to the size of shapes. For larger shapes the false positive region increases. If we denote the inaccuracy factor $\epsilon < 1$, then:

$$A = \frac{1}{4\pi} P^2 \epsilon$$

In addition, we assume that the oracle cannot classify shapes that fall in the region

$$\frac{1}{4\pi} P^2 \epsilon \delta \leq A \leq \frac{1}{4\pi} P^2 \epsilon$$

where $\delta < 1$. Consequently, the generator of shapes generates two classes according to the following rule:

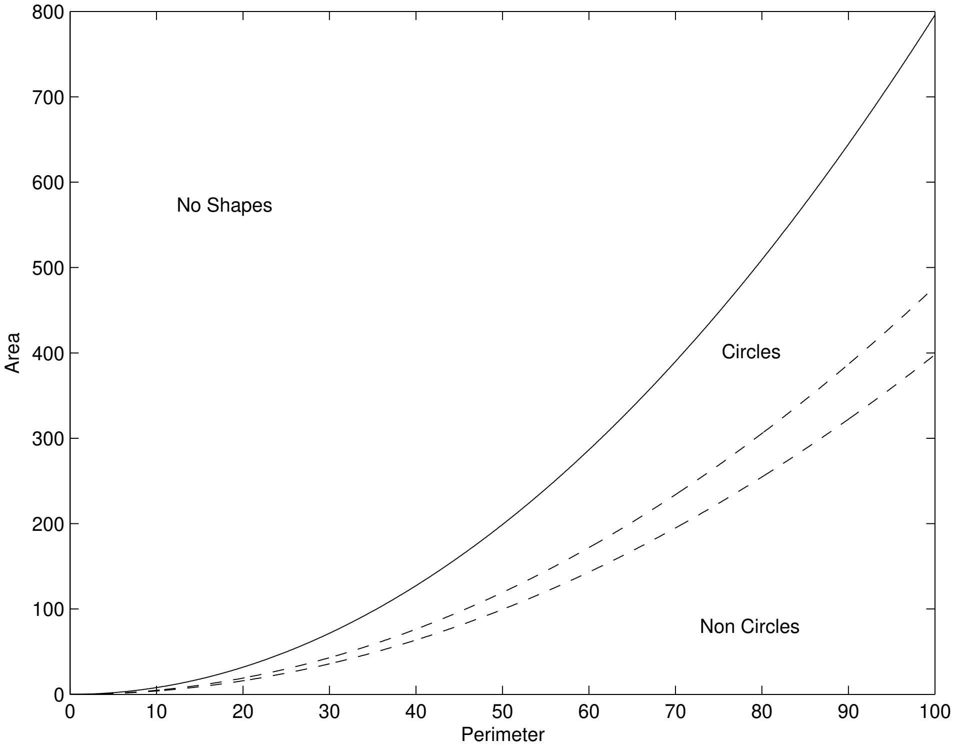

Figure 3.15: The classification regions of a generator with a constant multiplicative mistake factor. The shapes between the two dashed borders are unlabeled and will be ignored.

$$Circles: \ \frac{1}{4\pi}P^2\epsilon \leq A \leq \frac{1}{4\pi}P^2$$
$$Non-Circles: \ A \leq \frac{1}{4\pi}P^2\epsilon\delta$$

Figure (3.15) describes such a generator. Once again, the experiment was carried out on a randomly generated set (figure (3.16)) according to the new generator. The results are shown in figure (3.17) and we can see that the minimum of the span bound estimation is achieved for $p = 2$, predicting the same quadratic relation.

3.5.3 Case III - The Exact Oracle

The exact oracle is a very important case. The oracle makes no mistakes, such as a computer or any perfect measuring devise. If the relation between the area of the shape and its square perimeter is $1/4\pi$ the oracle classifies it as a circle otherwise it classifies it as a non-circle. The generator of shapes generates two classes according to the rule:

$$Circles: \ A = \frac{1}{4\pi}P^2$$

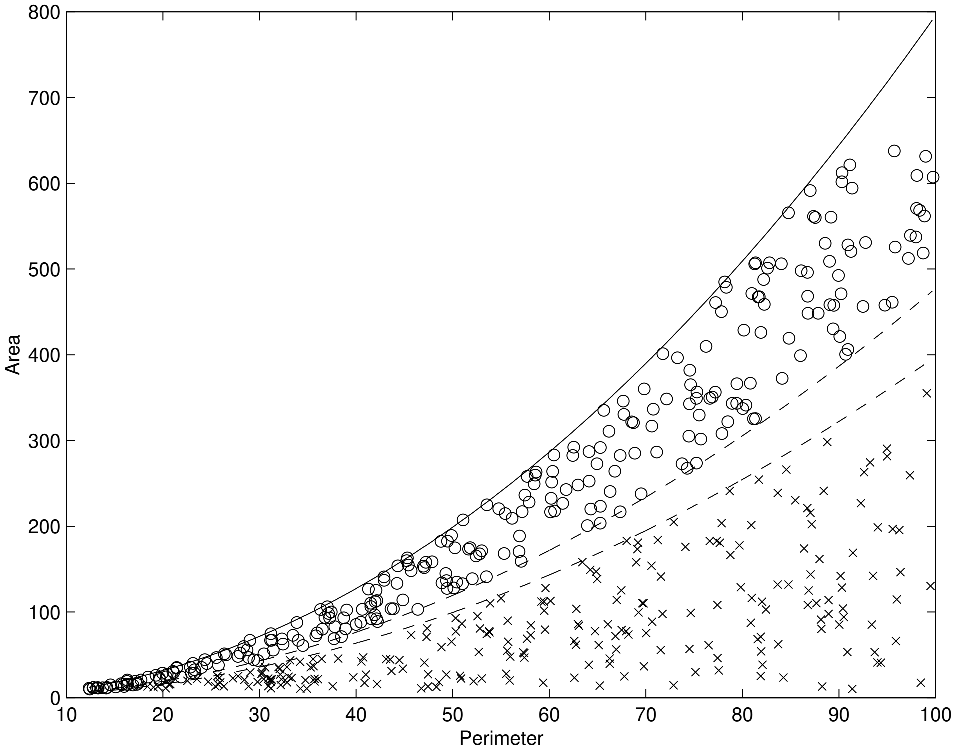

Figure 3.16: A randomly generated set of shapes (perimeter and area) according to a generator with a constant multiplicative mistake factor.

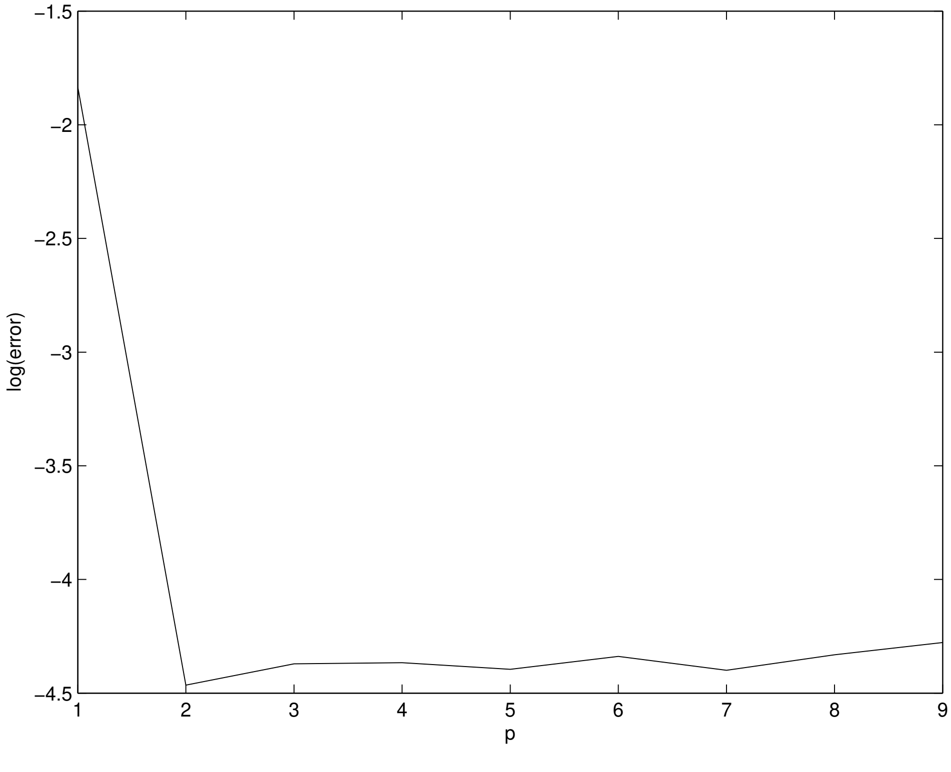

Figure 3.17: The span bound vs. the polynomial kernel degree for the 2D circles recognition problem with a constant multiplicative mistake factor. The minimum of the span bound estimation is achieved for a polynomial degree 2

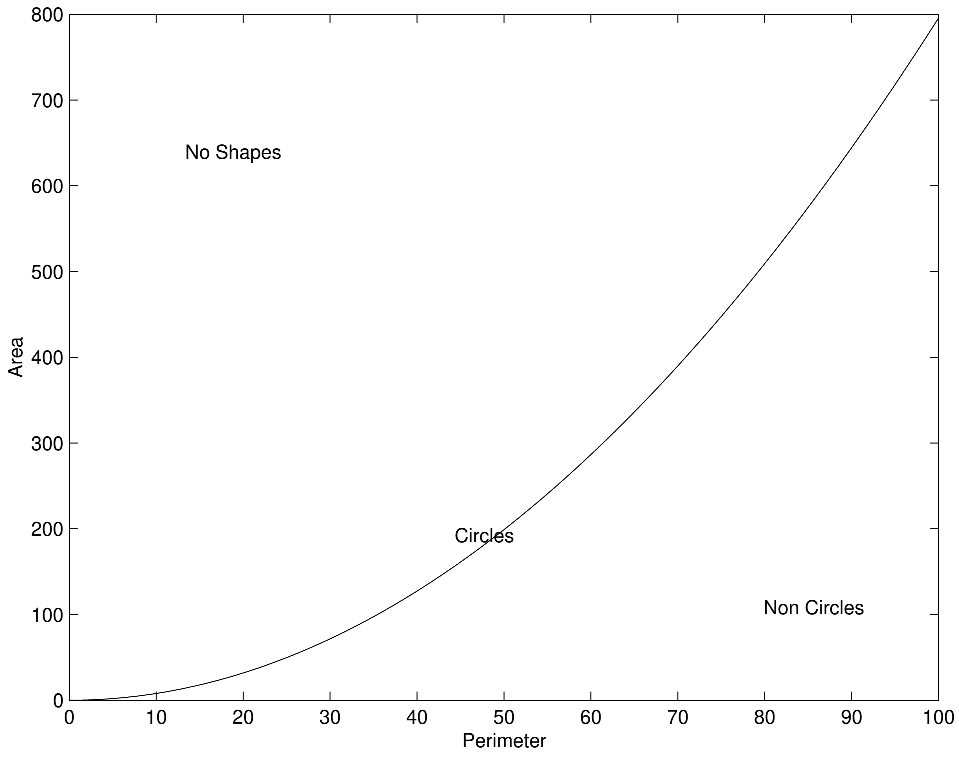

Figure 3.18: The classification regions of a generator with a constant multiplicative mistake factor.

$$Non - Circles: \; A < \frac{1}{4\pi} P^2 \tag{3.6}$$

The generator is described in figure (3.18), where there is no uncertainty region. A randomly generated set of shapes according to the above generator is shown in (3.19). Again, as expected, the best choice of the kernel is the polynomial kernel with $p = 2$ according to the span bound estimation as shown in figure (3.20).

3.5.4 Case IV - Circle Recognition from Human Prediction

In this sub-section we show that we can also learn the relation between the perimeter and the area in circles from a training set labeled by the average human eye. This task is much closer to real life, where shapes are classified based on human observations. The collected data is fed into a learning system. The learning system then builds a model for classifying $2D$ circles. We show that this model can achieve better results than human observations for future data as well as understanding the relation between the feature.

For this task we have created 100 shapes of circles and ellipses (see figure (3.21)) as a training set. We tested 30 people that do not have any known sight problems. The average of their prediction was taken as the correct labeling. Then, polynomial model selection learned

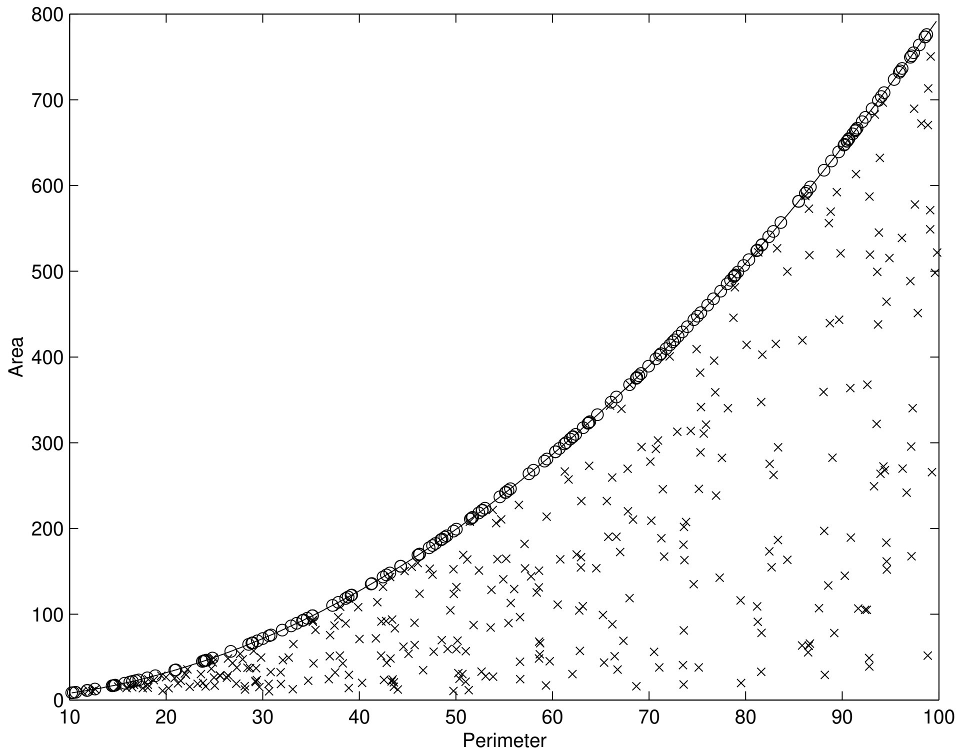

Figure 3.19: A randomly generated set of shapes (perimeter and area) according to an exact generator.

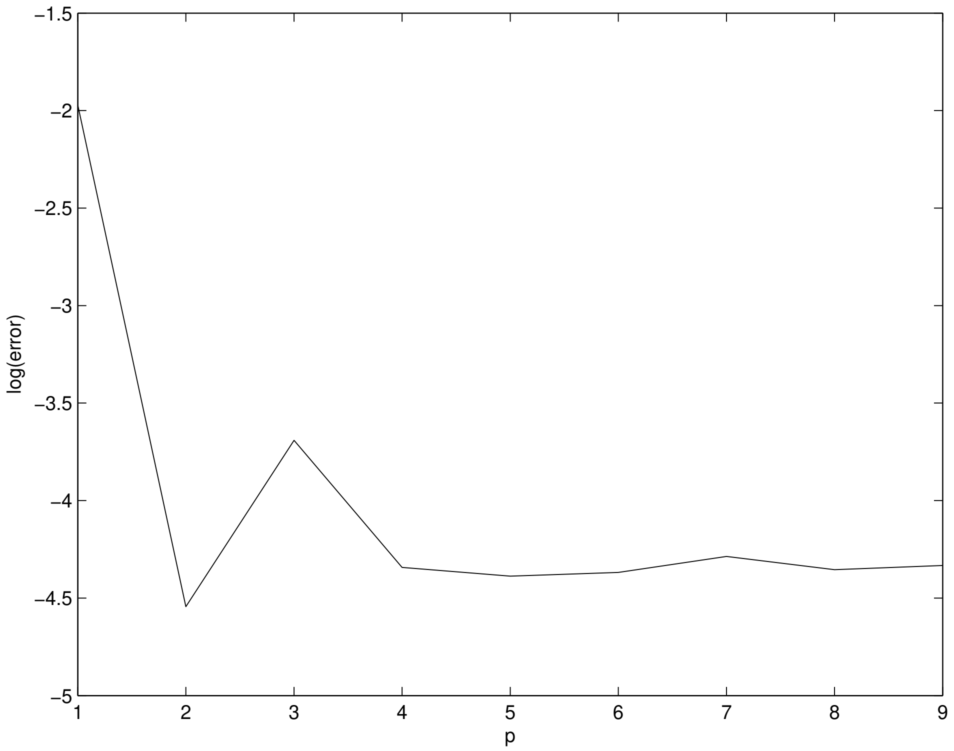

Figure 3.20: The span bound vs. the polynomial kernel degree for the 2D circles recognition problem with an exact oracle. The minimum of the span bound estimation is achieved for a polynomial degree 2

the degree of the relation between the feature vectors.

The span bound predicted that the relation between the area and perimeter is a polynomial relation of degree 2 (see figure (3.22)).

We succeeded in outperforming the human decision with a good choice of SVM and were able to learn the correct relational characteristics of the problem, even though the labeling was inaccurate with unknown noise.

After we have found the degree of the relation between features of the given shapes, we would like to know which monomials are really active and might affect the decision. The simplest way is to find the relative weights of the normalized separator after training with a few nested subsets of the training data. As we have shown in the previous section we can plot the monomial coefficient values. These values are shown in figure (3.23). We can see that if we consider only the two most effective monomials, we find that they are the area and the square of the perimeter of the shape. Their relative weight, even after training only with 90 points, is very close to the true weight, $1/4\pi$.

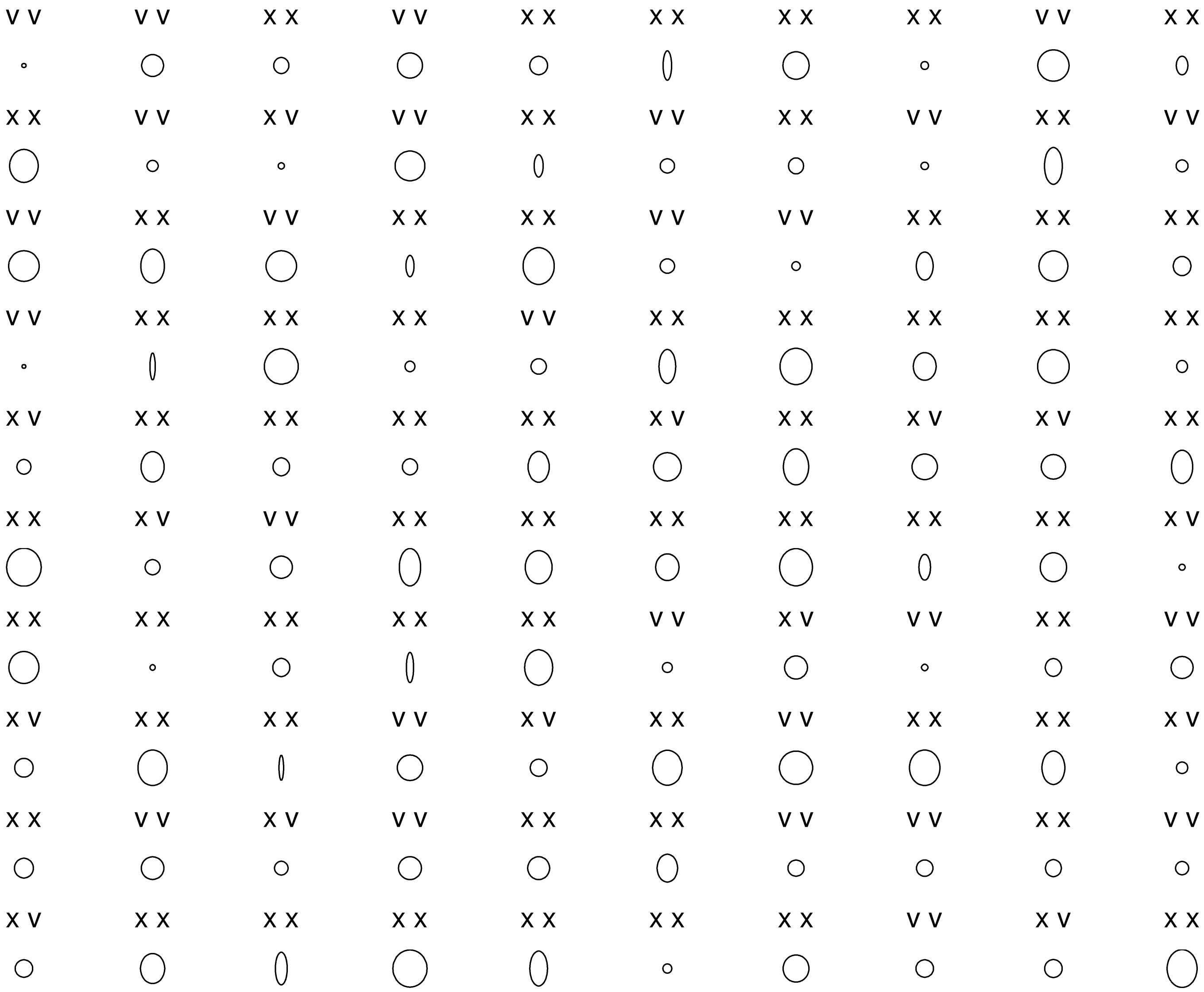

Figure 3.21: The training set that was labeled by humans. Each shape is marked with two marks, the first one is the correct labeling and the second one is the averaged labeling of humans. V stands for circles and X stands for non-circles. The group of correct circles is a subgroup of the circles according to humans decision.

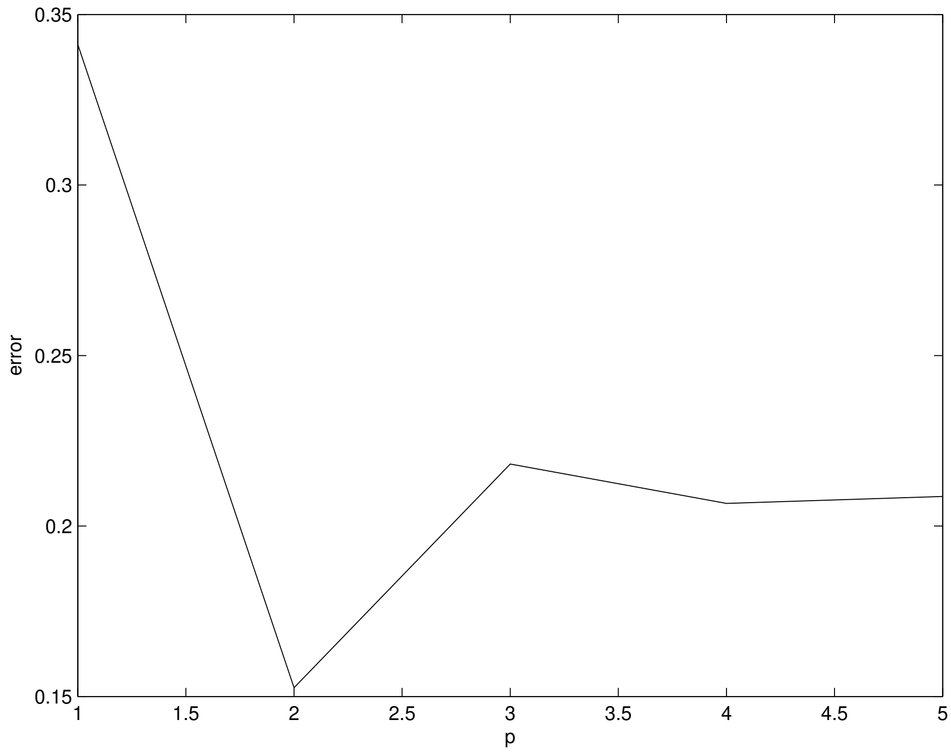

Figure 3.22: The span bound vs. the polynomial kernel degree for the 2-D circles recognition problem with a human labeling as oracle. We can see that the minimum of the span bound is reached for a polynomial degree 2

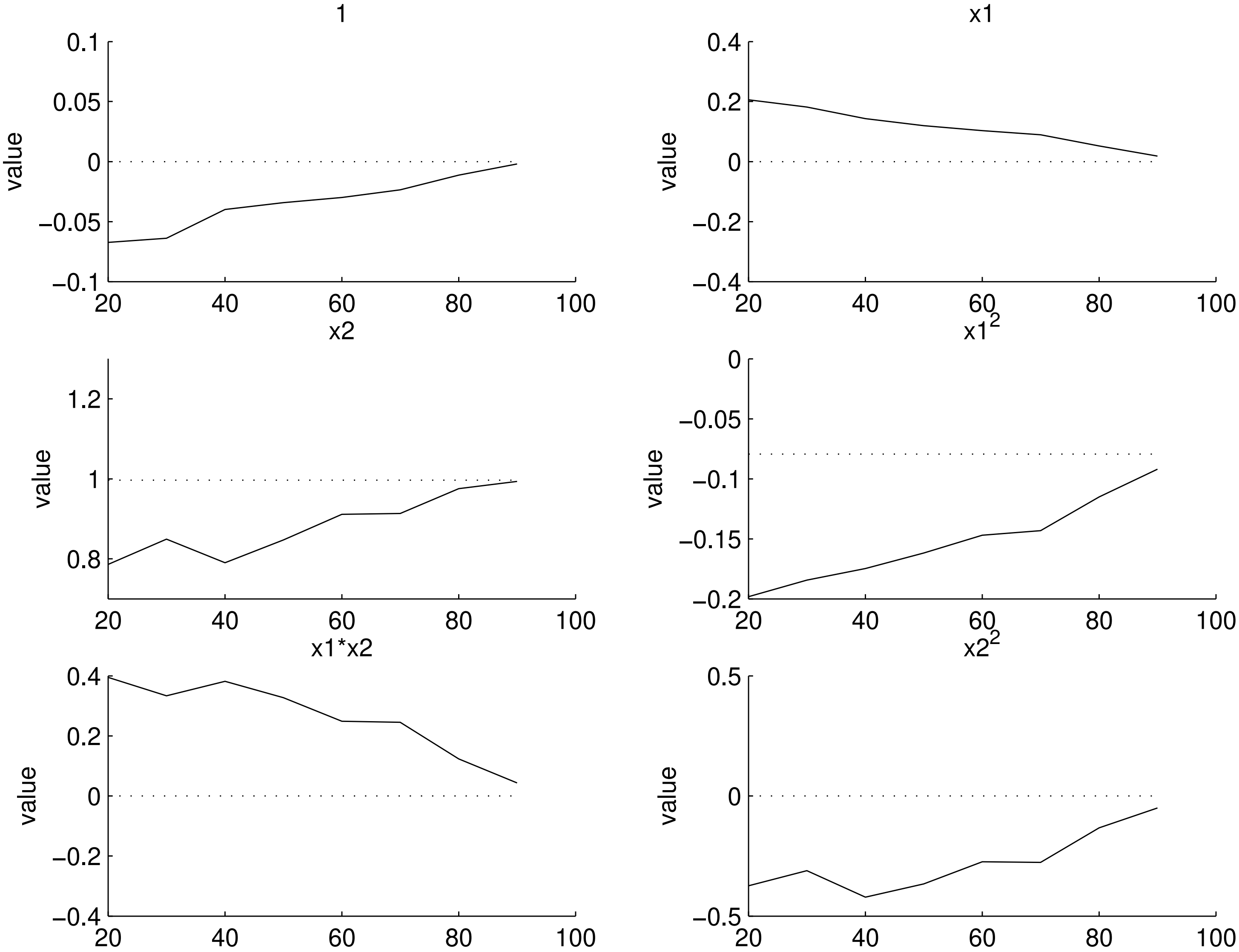

Figure 3.23: Coefficients of the resulting separator for different trainings of polynomial SVMs with $p = 2$ on nested subsets (solid line) compared to the values of the generator of circles.

Chapter 4

Conclusions

In statistical learning theory, a training set is drawn from an unknown probability distribution. The distribution of this set is a generator that obeys certain rules and constraints, which usually remain undiscovered after the learning tasks. However, we might be interested in these rules, which we call "nature rules", for explaining certain observed phenomena and unexplained behaviors. In this work we show that using polynomial kernel SVM in model selection might unveil certain information concerning the nature rule under some assumptions.

We assume that the rule that generated the data is polynomial, or can be approximated and explained as a polynomial rule. Hence, the method we present enables us to learn the relations between features under polynomial rules. It relies only on the given data set and reveals the maximal polynomial degree of the generator, based on explanations from VC theory. The degree of the generator satisfies optimality and balance between complexity and separation.

Furthermore, we show that we might be able to learn more about the generating rule after finding the maximal polynomial degree. From this view point we can formulate a general structure of the rule as multiple relations between the features of the data in monomial terms. Using this structure, we present nested subsets of the data for learning the tendency of the weights of each monomial. We present experiments that show that the weights converge to the true weights of the relation in the nature rule.

The last step in this procedure reveals the most important components in feature space using a specific kernel function. It can be replaced by a kernel principle component analysis (kPCA) [20]. So, we can use polynomial model selection to discover the degree of the wanted polynomial and then apply kPCA using the kernel with that degree to obtain the weight of each monomial.

This work exploits the shortcomings of using model selection on polynomial kernel SV machines. The tunable parameter is discrete, ruling out the possibility of reaching the optimal value without an exhaustive search as in [14] or even a fast search as in [11], where both methods depend on the continuity of RBF kernel parameter and the smoothness of the bound estimates. In addition, the discrete range of values contributes to the reliability of the result.

We tested the method on synthetic data generated from polynomial rules of different degrees. The results were very encouraging, as the bounds were informative and gave the correct parameter values. We proceeded to focus on the 2D circles recognition and tested the method extensively on that case. We applied the method to noisy and noiseless labeling. We found from that the relation between the perimeter and the area in circles is quadratic. But, most importantly, we found that when the teacher is the average human, the method is able to find a machine that improved the human decision and identified the relation between the features. This result is very interesting, since the mistake and uncertainty factors of human decisions depend on several unknown sight and psychological aspects that we cannot quantify. Modeling the human decision and explaining nature rules correctly is very encouraging for identifying unknown rules from inaccurate observations. In general, the procedure might be computationally heavy for some large scale problems, however, we have shown that it can be very helpful for identifying unknown rules by testing a finite set of observations.

The drawback of the method is the fact that the parameter space is discrete, and consequently, the performance estimation cannot be smooth. This prevents deriving bounds and faster convergence. On the other hand it motivates extending the family of polynomials to a spurious family with a continuous parameter space. A possible scan of this parameter, followed by discretization of the result, can be investigated. This will also require finding a smooth bound for a continuous range of the polynomial degrees, which may exist in a complex space.

Appendix A

The Kernel AdaTron Algorithm

The kernel AdaTron algorithm for support vector classification was first introduced by [21] based on the basic AdaTron algorithm from the statistical mechanics literature [22]. The support vector problem is realized through the maximization of equation (2.29).

Imposing the constraints on the problem yields the following:

maximize

$$L_{dual} = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y^i y^j K(x^i, x^j) - \lambda \sum_{i=1}^{l} \alpha_i y^i \tag{A.1}$$

The Algorithm proposes an iterative gradient ascent steps towards a stable solution that maximizes equation (A.1) as a function of $\{\alpha_i\}_{i=1}^{l}$. The step in the direction of the gradient is calculated as follows

$$\frac{\partial L_{dual}}{\partial \alpha_i} = 1 - \frac{1}{2} y^i \sum_{j=1}^{l} \alpha_j y^j K(x^i, x^j) - \lambda y^i$$

The condition $\alpha_i \geq 0$ in imposed in the algorithm, where negative α_i are made zero. In some cases the condition $\sum_{i=1}^{l} \alpha_i y^i = 0$ is not imposed and thus the last term of the gradient λy^i will be dropped. This will cause the solution to pass through the origin in feature space. Following, we give both versions of the kernel AdaTron algorithm, the unbiased version and the biased version, where in the biased version λ is adjusted on the gradient by

$$\frac{\partial L_{dual}}{\partial \lambda} = \sum_{i=1}^{l} \alpha_i y^i$$

The Kernel AdaTron Algorithm with no Bias:

1. Initialize $\alpha_i^0 = 0$.
2. For $t = 1$ to t_{max} do
   2.1. For $i = 1$ to l do
      2.1.1. $z^i \leftarrow \sum_{j=1}^{l} \alpha_i^{t-1} y^j K(x^i, x^j)$
      2.1.2. $\delta\alpha_i^t = \eta(1 - y^i z^i)$

2.1.3. $\alpha_i^t \leftarrow \alpha_i^{t-1} + \delta\alpha_i^t$

2.1.4. If $\alpha_i^t < 0$ then

$\alpha_i^t \leftarrow 0$

2.2. $\gamma \leftarrow \frac{1}{2}\left(\min_{\{i/y^i=+1\}}(z^i) - \max_{\{i/y^i=-1\}}(z^i)\right)$

2.3. If $\gamma \approx 1$ then

Goto step (3)

3. Stop.

The Kernel AdaTron Algorithm with Bias

1. Initialize $\alpha_i^0 = 0$.
2. $\lambda^0 = \mu$
3. $\lambda^1 = -\mu$
4. For $t = 1$ to t_{max} do

4.1. If $t > 1$ then

$\lambda^t = \lambda^{t-1} - w^{t-1}\left(\frac{\lambda^{t-1}-\lambda^{t-2}}{w^{t-1}-w^{t-2}}\right)$

4.2. For $i = 1$ to l do

4.2.1. $z^i \leftarrow \sum_{j=1}^{l} \alpha_i^{t-1} y^j K(x^i, x^j)$

4.2.2. $\delta\alpha_i^t = \eta(1 - y^i z^i)$

4.2.3. $\alpha_i^t \leftarrow \alpha_i^{t-1} + \delta\alpha_i^t$

4.2.4. If $\alpha_i^t < 0$ then

$\alpha_i^t \leftarrow 0$

4.3. $w^t \leftarrow \sum_{j=1}^{l} \alpha_j^t y^j$

4.4. $\gamma \leftarrow \frac{1}{2}\left(\min_{\{i/y^i=+1\}}(z^i) - \max_{\{i/y^i=-1\}}(z^i)\right)$

4.5. If $\gamma \approx 1$ then

Goto step (5)

5. Stop.

Both versions of the algorithm can be slightly changed and adapted to the soft margin SVM to tolerate classification errors by adding an upper bound to α_i imposing $0 \leq \alpha_i \leq C$. Following the soft margin adaption of both versions:

The Kernel AdaTron Algorithm with no Bias and Soft Margin:

1. Initialize $\alpha_i^0 = 0$.
2. For $t = 1$ to t_{max} do

2.1. For $i = 1$ to l do

2.1.1. $z^i \leftarrow \sum_{j=1}^{l} \alpha_i^{t-1} y^j K(x^i, x^j)$

2.1.2. $\delta\alpha_i^t = \eta(1 - y^i z^i)$

2.1.3. $\alpha_i^t \leftarrow \alpha_i^{t-1} + \delta\alpha_i^t$

2.1.4. If $\alpha_i^t < 0$ then

$\alpha_i^t \leftarrow 0$

2.1.5. If $\alpha_i^t > C$ then

$\alpha_i^t \leftarrow C$

2.2. $\gamma \leftarrow \frac{1}{2}\left(\min_{\{i/y^i=+1,\alpha_i<C\}}(z^i) - \max_{\{i/y^i=-1,\alpha_i<C\}}(z^i)\right)$

2.3. If $\gamma \approx 1$ then

Goto step (3)

3. Stop.

The Kernel AdaTron Algorithm with Bias and Soft Margin

1. Initialize $\alpha_i^0 = 0$.
2. $\lambda^0 = \mu$
3. $\lambda^1 = -\mu$
4. For $t = 1$ to t_{max} do

4.1. If $t > 1$ then

$\lambda^t = \lambda^{t-1} - w^{t-1}\left(\frac{\lambda^{t-1}-\lambda^{t-2}}{w^{t-1}-w^{t-2}}\right)$

4.2. For $i = 1$ to l do

4.2.1. $z^i \leftarrow \sum_{j=1}^{l} \alpha_i^{t-1} y^j K(x^i, x^j)$

4.2.2. $\delta\alpha_i^t = \eta(1 - y^i z^i)$

4.2.3. $\alpha_i^t \leftarrow \alpha_i^{t-1} + \delta\alpha_i^t$

4.2.4. If $\alpha_i^t < 0$ then

$\alpha_i^t \leftarrow 0$

4.2.5. If $\alpha_i^t > C$ then

$\alpha_i^t \leftarrow C$

4.3. $w^t \leftarrow \sum_{j=1}^{l} \alpha_j^t y^j$

4.4. $\gamma \leftarrow \frac{1}{2}\left(\min_{\{i/y^i=+1,\alpha_i<C\}}(z^i) - \max_{\{i/y^i=-1,\alpha_i<C\}}(z^i)\right)$

4.5. If $\gamma \approx 1$ then

Goto step (5)

5. Stop.

Theory provides adequate choices for the learning rate parameter η and experiments in [21] demonstrate the dependency of the choice of η on the kernel function. It has been proved [23][4] that the algorithm converges when

$$0 < \eta < \frac{2}{K\left(x^i, x^i\right)}$$

For Gaussian kernels $K\left(x^i, x^i\right) = 1$, hence,

$$0 < \eta < 2$$

and for polynomial kernels

$$0 < \eta < \frac{2}{(\|x^i\|^2 + 1)^d}$$

for each pattern (x^i, y^i).

The proof is completed by showing that the Lagrangian is upper bounded by

$$L(\alpha^*) = \frac{1}{2\gamma^2} = \frac{\|w\|^2}{2} = \frac{\sum_{i=1}^{l} \alpha_i^*}{2}$$

where γ is the margin.

Appendix B

Computing Error Bounds

Margin Radius Bound

After training the SVM classifier we obtain the value of the margin $\gamma = |w|^{-1}$. Thus we only need to compute the radius of the smallest enclosing sphere in feature space. The technique was originally suggested in Vapnik [4] and illustrated in [16]. It was also used by Tax, Duin and Campbell as a technique for novelty detection [24][25][26][27][28], where one tries to bound a region of a given sample set and be able to detect abnormal test patterns.

In RBF kernels, such as the Gaussian (2.37) the case is very simple, and perhaps it is preferable for experiments and evaluations such as in [29][11], since the mapped data will all fall inside a sphere of radius 1 in feature space, and thus ,$r = 1$ is satisfying.

In other cases the radius can be found as the maximal distance between a pattern and the center of mass over all the given patterns. The minimal radius possible is the wanted radius. This is formalized as follows:

minimize r^2 subject to

$$\|\Phi(x^i) - C\|^2 \leq r^2 \ \forall i$$

where C is the center of the sphere, which is also unknown. We introduce Lagrange multipliers and minimize

$$L_P = r^2 - \sum_{i=1}^{l} \alpha_i \left(r^2 - \|\Phi(x^i) - C\|^2 \right)$$

subject to

$$\alpha_i \geq 0, \ \forall \ i$$
$$\sum_{i=1}^{l} \alpha_i = 1$$

Then, we find C at the solution by requiring that the derivative of the Lagrangian with respect to C vanishes:

$$\frac{\partial L_P}{\partial C} = -\sum_{i=1}^{l} \alpha_i \Phi(x^i) C + \sum_{i=1}^{l} \alpha_i C^2 = 0$$

At the solution $\sum_{i=1}^{l} \alpha_i = 1$, thus

$$C = \sum_{i=1}^{l} \alpha_i \Phi(x^i)$$

Substitute C in the Lagrangian yields

$$L_P = \sum_{i=1}^{l} \alpha_i \left(\|\Phi(x_i) - \sum_{i=1}^{l} \alpha_i \Phi(x_i)\|^2 \right) = \sum_{i=1}^{l} \alpha_i K(x_i, x_i) - \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j K(x_i, x_j)$$

and requiring that the derivatives with respect to α_i vanish

$$\frac{\partial L_P}{\partial \alpha_i} = K(x^i, x^i) - \sum_{j=1}^{l} \alpha_j K(x^i, x^j)$$

The solution r, can now be found by the substitution of the solution α_i^*, which can be computed by an iterative algorithm much like the kernel AdaTron with an iteration step of

$$\eta \left[K(x^i, x^i) - \sum_{j=1}^{l} \alpha_j K(x^i, x^j) \right]$$

The Algorithm for Finding the Radius of the Smallest Enclosing Sphere of a Sample Set in Feature Space:

1. Initialize $\alpha_i^0 = 0$.
2. $\lambda^0 = \mu$
3. $\lambda^1 = -\mu$
4. For $t = 1$ to t_{max} do
   - 4.1. If $t > 1$ then
     $\lambda^t = \lambda^{t-1} - w^{t-1} \left(\frac{\lambda^{t-1} - \lambda^{t-2}}{w^{t-1} - w^{t-2}} \right)$
   - 4.2. For $i = 1$ to l do
     - 4.2.1. $z^i \leftarrow \sum_{j=1}^{l} \alpha_i^{t-1} K(x^i, x^j)$
     - 4.2.2. $\delta\alpha_i^t = \eta(K(x^i, x^i) - z^i)$
     - 4.2.3. $\alpha_i^t \leftarrow \alpha_i^{t-1} + \delta\alpha_i^t$
     - 4.2.4. If $\alpha_i^t < 0$ then
       $\alpha_i^t \leftarrow 0$
   - 4.3. $w^t \leftarrow \sum_{j=1}^{l} \alpha_j^t$
5. Stop.

Span Bound

It was proved in [13] that under the assumption that the support vectors remain the same during the leave one out procedure

$$y^p\left(f^0\left(x^p\right) - f^p\left(x^p\right)\right) = \alpha_p^0 S_p^2$$

where f^0 is the classifier trained on the complete set and f^p is the classifier trained after removing x^p. The span bound then becomes

$$\frac{1}{l}\sum_{p=1}^{l}\left(\alpha_p S_p^2 - 1\right)$$

and the span S_p is computed as follows

$$S_p^2 = \frac{1}{\left(\widetilde{K}_{SV}^{-1}\right)_{pp}}$$

where $\widetilde{K}_{SV}$ is the kernel matrix of the support vectors. This bound was approximated to a smooth function [14]:

$$S_p^2 = \frac{1}{\left(\widetilde{K}_{SV} + D\right)_{pp}^{-1}} - D_{pp}$$

where D is diagonal with elements $D_{ii} = \eta/\alpha_i^0$ and $D_{\#SV+1,\#SV+1} = 0$ and η is a constant to be chosen experimentally.

The computation complexity required to evaluate the bound is the complexity of a matrix inversion, which is not larger the solving SVM.

Bibliography

[1] F. Rosenblatt. *Principles of Neurodynamics.* Spartam Books New York, 1962.

[2] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by backpropagation errors. *Nature*, 323:533–536, 1986.

[3] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error backpropagation. In James L. McClelland and David E. Rumelhart, editors, *Parallel Distributed Processing*, pages 318–362. MIT Press, MIT, 1987.

[4] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag New York, 1995.

[5] E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. *A.I. Memo 1602: MIT Artificial Intellegence Laboratory*, 1996.

[6] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. pages 130–136, 1997. Computer Vision and Pattern Recognition, Puerto Rico, June 16–20.

[7] T. Joachims. Text categorization with support vector machines. November 1998. Technical report LS-8 Report 23, University of Durtmond.

[8] C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable pedestrian detection system. pages 241–246, 1998. In IEEE Conference on Intelligent Vehicles.

[9] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. 1998. The International Conference on Computer Vision, Bombay, India.

[10] T. Evgeniou, M. Pontil, C. Papageorgiou, and T. Poggio. Image representations for object detection using kernel classifiers. 2000. The International Conference on Computer Vision.

[11] N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. 1998. NeuroCOLT Technical Report NC-TR-98-017, Royal Holloway College, University of London, UK.

[12] O. Chapelle and V. Vapnik. Model selection for support vector machines. 1999.

[13] V.N. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9), 2000.

[14] V.N. Vapnik, O. Chapelle, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

[15] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag New York, 2 edition, 2000.

[16] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):1–47, 1998.

[17] O. Chapelle, P. Haffner, and V.N. Vapnik. Svms for histogram-based image classification. *IEEE Transaction on Neural Networks*, 9, 1999.

[18] C. Papageorgiou and T. Poggio. A pattern classification approach to dynamical object detection. pages 1223–1228, 1999. The International Conference on Computer Vision.

[19] C. Papageorgiou and T. Poggio. Trainable pedestrian detection. 1999. The International Conference on Computer Vision.

[20] B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998. Technical Report No. 44, 1996, Max Planck Institut fr biologische Kybernetik, Tbingen.

[21] T.-T. Frieß, N. Cristianini, and C. Campbell. The kernel adatron algorithm: A fast and simple learning procedure for support vector machines. 1998. In 15th Intl. Conf. Machine Learning. Morgan Kaufmann Publishers.

[22] J. K. Anlauf and M. Biehl. The adatron: an adaptative perceptron algorithm. *Europhysics letter*, 10:687–692, 1989.

[23] P. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*. Cambridge, Mass: MIT Press, 1998.

[24] D. Tax, A. Ypma, and R. Duin. Support vector data description applied to machine vibration analysis. In M. Boasson, J. Kaandorp, J.Tonino, and M. Vosselman, editors, *Proc. 5th Annual Conference of the Advanced School for Computing and Imaging (Heijen, NL, June 15-17)*, pages 398–405. 1999.

[25] D. Tax and R. Duin. Data domain description using support vectors. In M. Verleysen, editor, *Proceedings of the European Symposium on Artificial Neural Networks*, pages 251–256. 1999.

[26] C. Campbell. Algorithmic approaches to training support vector machnies: A survey. *European Symposium on Artificial Neural Networks*, 2000.

[27] C. Campbell. An introduction to kernel methods. In R.J. Howlett and L.C. Jain, editors, *Radial Basis Function Networks: Design and Application*. Springer Verlag, Berlin, 2000.

[28] C. Campbell and K. P. Bennett. A linear programming approach to novelty detection. *Advances in Neural Information Processing Systems*, 14, 2000.

[29] B. Scholkopf, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Generalization bounds via eigenvalues of the gram matrix. 1999. Technical Report 99-035, NeuroCOLT.