Robust Minimax Probability Machine Regression; CU-CS-952-03

Thomas Strohmann
University of Colorado Boulder

Gregory Grudic
University of Colorado Boulder

Follow this and additional works at: http://scholar.colorado.edu/csci_techreports

Recommended Citation
http://scholar.colorado.edu/csci_techreports/894
Robust Minimax Probability Machine Regression

Thomas R. Strohmann  
Gregory Z. Grudic

Department of Computer Science  
University of Colorado  
Boulder, CO 80309-0430, USA

Abstract

We formulate regression as maximizing the minimum probability (Ω) that the true regression function is within ±ϵ of the regression model. Our framework starts by posing regression as a binary classification problem, such that a solution to this single classification problem directly solves the original regression problem. Minimax probability machine classification (Lanckriet et al., 2002a) is used to solve the binary classification problem, resulting in a direct bound on the minimum probability Ω that the true regression function is within ±ϵ of the regression model. This minimax probability machine regression (MPMR) model assumes only that the mean and covariance matrix of the distribution that generated the regression data are known; no further assumptions on conditional distributions are required. Theory is formulated for determining when estimates of mean and covariance are accurate, thus implying robust estimates of the probability bound Ω. Conditions under which the MPMR regression surface is identical to a standard least squares regression surface are given, allowing direct generalization bounds to be easily calculated for any least squares regression model (which was previously possible only under very specific, often unrealistic, distributional assumptions). We further generalize these theoretical bounds to any superposition of basis functions regression model. Experimental evidence is given supporting these theoretical results.

Keywords: Minimax Probability Machine, Regression, Robust, Kernel Methods, Bounds, Distribution free

1. Introduction

In this paper we formulate the regression problem as one of maximizing the probability, denoted by Ω, that the predicted output will be within some margin ϵ of the true regression function.\(^1\) We show how to compute a direct estimate of this probability for any given margin ϵ > 0. Following the idea of the minimax probability machine for classification (MPMC) due to Lanckriet et al. (2002a), we make no detailed distributional assumptions, but obtain a probability Ω that is a lower bound for all possible distributions with a known mean and covariance matrix. In other words, we maximize the minimum probability of our regression model being within ±ϵ correct on test data for all possible distributions that

\(^1\) Throughout this paper, the phenomenon that generated the training data is referred to as the true regression function (or surface), while the model that is constructed from the training data is referred to as the regression model.
have the same mean and covariance matrix as the distribution that generated the training data. We term this type of regression model as a minimax probability machine regression (MPMR) (see Strohmann and Grudic, 2003).

Current practice for estimating how good a regression model is dictates that one has to either estimate the underlying distribution of the data or make Gaussian assumptions, both of which have their problems. Estimating high dimensional distributions from small samples is a very difficult problem that has not been solved satisfactory (see S.J. Raudys, 1991). Gaussian Processes obtain regression models within a Bayesian framework (see Williams and Rasmussen, 1995, Rasmussen, 1996, Gibbs, 1997a, Mackay, 1997). As the name indicates they put a prior on the function space that is a generalization of a Gaussian distribution for (a finite number of) random variables. Instead of mean and covariance, Gaussian Processes use a mean function and a covariance function to express priors on the function space. By adopting these restrictions it is possible to obtain confidence intervals for a given regression estimate. In practice, however, the assumption that data is generated from underlying Gaussians does not always hold which can make the Gaussian Process confidence intervals unrealistic. Similar problems with unrealistic bounds occur in such algorithms as the relevance vector machine (Tipping, 2000) where approximations, which may be unrealistic, are made in obtaining these. Whereas—as long as one can obtain accurate estimates of mean and covariance matrix—our distribution free algorithm will always yield a correct lower bound estimate, no matter what the underlying distribution is.

Because the accuracy of the proposed regression algorithm directly depends on the accuracy of mean and covariance estimates, we propose a theoretical framework for determining when these estimates are accurate. This in turn determines when the estimate of the bound \( \Omega \) is accurate. The general idea is to randomly discard a (small) subset of the training data and measure how sensitive the estimate of \( \Omega \) is when we use the remaining data to estimate mean and covariance matrix. This approach has some resemblance to the RANSAC (Random SAmpling and Consensus) method, originally developed by Fischler and Bolles (1981) for the task of image analysis.

To solve the regression problem, we reduce it to a binary classification problem by generating two classes that are obtained by shifting the dependent variable \( \pm \epsilon \) (see Figure 1). The regression surface is then interpreted as being the boundary that separates the two classes. Although any classifier can be used to solve the regression problem in this way, we propose to use MPMC (Lanckriet et al., 2002a) which gives a direct bound on \( \Omega \). One result of this is paper is a theorem stating that when MPMC is used to solve the regression problem as described above, the regression model is identical to the one obtained by standard least squares regression. The importance of this result is that it is now easy to calculate a generalization bound for the least squares method. This was previously only possible under strict, often unrealistic, distributional assumptions.

The paper is organized as follows: In section 2 we give a strict mathematical definition of our approach to the regression problem and describe our hypothesis space of functions for linear and nonlinear regression surfaces. Section 3 outlines the statistical foundation of the minimax probability machine, based on a theorem due to Marshall and Olkin (1960) that was later on extended by Popescu and Bertsimas (2001). We then show how any regression problem can be formulated as a binary classification problem by creating two symmetric classes: one has the output variable shifted by \( +\epsilon \), the other one has it shifted by \( -\epsilon \) (see
Figure 1). Following this scheme we conclude the section by deriving the linear MPMR

from the linear MPMC and demonstrating that due to the symmetry it is computationally
easier to find an MPM regression model than to find an MPM classifier. In section 4 we
describe how to obtain nonlinear regression surfaces by using any set of nonlinear basis
functions and then focus on using nonlinear kernel maps as a distance metric on pairs of
input vectors. The problem of overfitting the training data with highly sensitive nonlinear
kernel maps is addressed in section 5. There we propose a robustness measure \( R \) that reflects
how sensitive our estimate of \( \Omega \) is with respect to the estimates of mean and covariance
(which are the only estimates MPMR relies upon). Section 6 contains the experiments we
conduct with our learning algorithm on toy problems and real world data sets. We start
out with a simple toy example and show that for certain distributions the Gaussian Process
error bounds fail while the ones obtained by MPMR still hold. For the real world data sets
we focussed on investigating the accuracy of \( \Omega \) as a lower bound probability on \( \pm \epsilon \)-accurate
predictions and its relation to our robustness measure \( R \). The last section summarizes the
main contributions of this paper and points out some open research questions concerning
the MPMR framework.

A Matlab implementation of the MPMR algorithm can be obtained from:
http://nago.cs.colorado.edu/~strohman/papers.html

2. Regression Model

Formally, MPMR addresses the following problem: Let \( f^* : \mathbb{R}^d \rightarrow \mathbb{R} \) be some unknown
regression function (i.e. the phenomenon that generated the learning data), let the random
vector \( x \in \mathbb{R}^d \) be generated from some bounded distribution that has mean \( \bar{x} \) and covariance
\( \Sigma_x \)—which we will write from now on as \( x \sim (\bar{x}, \Sigma_x) \)—and let our training data be generated
according to:

\[
y = f^*(x) + \rho
\]
where $\rho$ is a noise term with an expected value of $E[\rho] = \bar{\rho} = 0$ and some finite variance $\text{Var}[\rho] = \sigma_\rho^2$. Given a hypothesis space $\mathcal{H}$ of functions from $\mathbb{R}^d$ to $\mathbb{R}$, we want to find a model $\hat{f} \in \mathcal{H}$ that maximizes the minimum probability of being $\pm \epsilon$ accurate. We define:

$$\Omega_f = \inf_{(x,y) \sim (\bar{x}, \bar{y}, \Sigma)} Pr\{|f(x) - y| < \epsilon\}$$  \hspace{1cm} (1)

where

$$\Sigma = \text{cov}(x_1, x_2, \ldots, x_d, y)$$

with $\Sigma \in \mathbb{R}^{(d+1) \times (d+1)}$ is the complete covariance matrix of the random vector $x$ and the random variable $y = f^*(x) + \rho$ (see equation (12) for details). The only assumptions we make about the first and second moments of the underlying distribution is that $\bar{x}, \bar{y}$ and $\Sigma$ are finite. No other distributional assumptions are made.

For any function $f \in \mathcal{H}$, the model $\hat{f}$ that we are looking for has to satisfy:

$$\Omega_{\hat{f}} \geq \Omega_f \text{ for all } f \in \mathcal{H}$$

For linear models, $\mathcal{H}$ contains all the functions that are linear combinations of the input vector:

$$f(x) = \sum_{j=1}^{d} \beta_j x_j + \beta_0 = \beta^T x + \beta_0 \quad (\text{where } \beta \in \mathbb{R}^d, \beta_0 \in \mathbb{R})$$  \hspace{1cm} (2)

For nonlinear models we consider the general basis function formulation:

$$f(x) = \sum_{i=1}^{k} \beta_i \Phi_i(x) + \beta_0$$  \hspace{1cm} (3)

In this paper we focus on the commonly used kernel representation $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ (see e.g. Smola and Schölkopf, 1998) where the hypothesis space consists of all linear combinations of kernel functions with the training inputs as their first arguments (i.e. $\Phi_i(x) = K(x_i, x)$):

$$f(x) = \sum_{i=1}^{N} \beta_i K(x_i, x) + \beta_0$$

where $x_i \in \mathbb{R}^d$ denotes the $i$th input vector of the training set $\Gamma = \{(x_1, y_1), \ldots, (x_N, y_N)\}$. In the nonlinear formulation we consider the distribution of the random vector of basis functions $\Phi = (\Phi_1, \Phi_2, \ldots, \Phi_N)$ to formulate the MPMR bound $\Omega$:

$$\Omega_f = \inf_{(\Phi,y) \sim (\bar{\Phi}, \bar{y}, \Sigma)} Pr\{|f(x) - y| < \epsilon\}$$

where

$$\Sigma = \text{cov}\left(y, \Phi_1, \Phi_2, \ldots, \Phi_N\right)$$

see equation (12) for details on calculating $\Sigma$. 

4
3. Linear Framework

In this section we will develop the linear MPMR by reducing the regression problem to a single (specially structured) binary classification problem. We will also show that the regression estimator of the linear MPMR is equivalent to the one obtained by the method of least squares. The terms random vector and class will be used interchangeably in a sense that all examples of a class are assumed to be generated from its associated random vector.

3.1 Minimax Probability Machine Classification (MPMC)

The minimax binary classifier for linear decision boundaries can be formulated as a hyperplane that maximizes the minimum probability of correctly classifying data points generated from the two underlying random vectors (Lanckriet et al., 2002a,b). Assume we have random vectors $\mathbf{u}$ and $\mathbf{v}$ that are drawn from two probability distributions characterized by $(\bar{\mathbf{u}}, \Sigma_u)$ and $(\bar{\mathbf{v}}, \Sigma_v)$, the linear MPMC algorithm solves the following optimization problem:

$$
\max_{\Omega, \mathbf{a} \neq \mathbf{0}, b} \Omega \text{ s.t. } \inf_{\mathbf{u} \sim (\bar{\mathbf{u}}, \Sigma_u)} P \{ \mathbf{a}^T \mathbf{u} \geq b \} \geq \Omega \land \inf_{\mathbf{v} \sim (\bar{\mathbf{v}}, \Sigma_v)} P \{ \mathbf{a}^T \mathbf{v} \leq b \} \geq \Omega \quad (4)
$$

A theorem due to Marshall and Olkin (1960) that was later on extended by Popescu and Bertsimas (2001) states a formula for the supremum probability that a random vector lies on one side of a hyperplane:

$$
\sup_{\mathbf{v} \sim (\bar{\mathbf{v}}, \Sigma_v)} P \{ \mathbf{a}^T \mathbf{v} \geq b \} = \frac{1}{1 + \delta^2}, \text{ with } \delta^2 = \inf_{\mathbf{a}^T \mathbf{w} \geq b} (\mathbf{w} - \bar{\mathbf{v}})^T \Sigma_v^{-1} (\mathbf{w} - \bar{\mathbf{v}})
$$

Where the infimum can be computed analytically from the hyperplane parameters $\mathbf{a}$ and $b$ and the estimates $\bar{\mathbf{v}}, \Sigma_v$ (Lanckriet et al., 2002b). This result can be used to simplify the optimization problem (4) to the following one:

$$
m = \left( \min_{\mathbf{a}} \sqrt{\mathbf{a}^T \Sigma_u \mathbf{a}} + \sqrt{\mathbf{a}^T \Sigma_v \mathbf{a}} \right) \quad \text{s.t. } \mathbf{a}^T (\bar{\mathbf{u}} - \bar{\mathbf{v}}) = 1 \quad (5)
$$

The offset $b$ of the hyperplane is uniquely determined for any $\mathbf{a}$:

$$
b = \mathbf{a}^T \bar{\mathbf{u}} - \frac{\sqrt{\mathbf{a}^T \Sigma_u \mathbf{a}}}{m} = \mathbf{a}^T \bar{\mathbf{v}} + \frac{\sqrt{\mathbf{a}^T \Sigma_v \mathbf{a}}}{m}
$$

Once a minimum value $m$ is obtained for (5), the probability bound $\Omega$ can be computed as:

$$
\Omega = \frac{1}{m^2 + 1}
$$

3.2 Formulating Regression as Classification

In a linear regression model we want to approximate $f^* : \mathbb{R}^d \rightarrow \mathbb{R}$ with a linear estimator, i.e. a function of the form:

$$\hat{y} = \hat{f}(\mathbf{x}) = \beta^T \mathbf{x} + \beta_0$$

To obtain a minimax regression model—i.e. one which maximizes the minimum probability that future points are predicted within $\pm \epsilon$ correctly—we use the MPMC framework in
the following way. Each training data point \((x_i, y_i)\) for \(i = 1, ..., N\) turns into two \(d + 1\)
dimensional vectors, one is labelled as class \(u\) (and has the \(y\) value shifted by \(+\epsilon\)) the other one as class \(v\) (and has the \(y\) value shifted by \(-\epsilon\), see also Figure 1):

\[
\begin{align*}
\mathbf{u}_i &= (y_i + \epsilon, x_{i1}, x_{i2}, ..., x_{id}) & i = 1, ..., N \\
\mathbf{v}_i &= (y_i - \epsilon, x_{i1}, x_{i2}, ..., x_{id}) & i = 1, ..., N
\end{align*}
\]

(7)

It is interesting to note that we could use any binary classification algorithm to solve this
artificial classification problem. The boundary obtained by the classifier turns directly into
the regression surface one wants to estimate. For example one could even use decision trees
as underlying classifier, determine where the decision tree changes its classification, and use
this boundary as the regression output.

In this paper we focus on the consequences of using MPMC as the underlying classifier for
the problem defined by (7). Once the hyperplane parameters \(a\) and \(b\) have been determined
by the MPMC, we use the classification boundary \(a^T z = b\) to predict the output \(\hat{y}\) for a
new input \(x\) (where \(z\) is defined as \(z = (\hat{y}, x_1, x_2, ..., x_d)\)):

\[
\begin{align*}
\mathbf{a}^T \mathbf{z} &= b & \text{(classification boundary)} \\
\Leftrightarrow \quad a_1 \hat{y} + a_2 x_1 + a_3 x_2 + ... + a_{d+1} x_d &= b \\
\Leftrightarrow \quad \hat{y} &= -\frac{a_2}{a_1} x_1 - \frac{a_3}{a_1} x_2 - ... - \frac{a_{d+1}}{a_1} x_d + \frac{b}{a_1} & (a_1 \neq 0) \\
\Leftrightarrow \quad \hat{y} &= \beta_1 x_1 + \beta_2 x_2 + ... + \beta_d x_d + \beta_0 = \beta^T \mathbf{x} + \beta_0 & \text{(regression function)}
\end{align*}
\]

(8)

The symmetric \(\pm \epsilon\) shift in the output direction has a number of useful properties. First
of all, the probability bound \(\Omega\) of correct classification is equivalent to the probability that
future predictions will be within \(\pm \epsilon\) accurate. Roughly speaking, if a hyperplane classifies
a point of class \(u\) correctly, then the corresponding \(y\) value can be at most \(\epsilon\) below the
hyperplane. Similarly, for points of class \(v\) that are classified correctly, the corresponding
\(y\) value can be at most \(\epsilon\) above the hyperplane (see the next section for a rigorous proof).
Second, the artificial classification problem has a much simpler structure than the general
classification problem. In particular, we find that \(\Sigma_u = \Sigma_v = \Sigma\) and \(\bar{u} - \bar{v} = (2\epsilon, 0, ..., 0)^T\).
This allows us to simplify (5) even further, we can minimize \(2 \sqrt{\mathbf{a}^T \Sigma \mathbf{a}}\) or equivalently \(\mathbf{a}^T \Sigma \mathbf{a}\)
with respect to \(\mathbf{a}\). The next section shows how to solve this minimization problem by just
solving one linear system.

3.3 Linear MPMR

Given an estimate \(\Sigma = \Sigma_u = \Sigma_v \in \mathbb{R}^{d+1 \times d+1}\) of the covariance matrix for the random
vector \((y, x)\) that generated the data, we can simplify (5):

\[
\min_{\mathbf{a}} \sqrt{\mathbf{a}^T \Sigma_u \mathbf{a}} + \sqrt{\mathbf{a}^T \Sigma_v \mathbf{a}} \quad s.t. \quad \mathbf{a}^T (\bar{u} - \bar{v}) = 1 \iff \min_{\mathbf{a}} 2 \sqrt{\mathbf{a}^T \Sigma \mathbf{a}} \quad s.t. \quad \mathbf{a}^T (2\epsilon, 0, ..., 0)^T = 1
\]

Since we are only interested in the minimum we can get rid of the square root and the factor
of 2. The goal of the linear MPMR can then be formulated as the solution to the following
constrained optimization problem for \(\mathbf{a} \in \mathbb{R}^{d+1}\):

\[
\min_{\mathbf{a}} \mathbf{a}^T \Sigma \mathbf{a} \quad s.t. \quad (2\epsilon, 0, ..., 0)\mathbf{a} = 1
\]

(9)
We use Lagrangian multipliers to turn the problem into an unconstrained optimization problem:

$$L(a, \lambda) = a^T \Sigma a + \lambda((2\varepsilon, 0, \ldots, 0)a - 1)$$

By setting the derivatives $\frac{\partial L(a, \lambda)}{\partial a} = 0$ and $\frac{\partial L(a, \lambda)}{\partial \lambda} = 0$ we obtain the following system of linear equations which describe a solution of (9):

$$2\Sigma a + (\lambda 2\varepsilon, 0, \ldots, 0)^T = 0$$

and

$$a_1 = \frac{1}{2\varepsilon}$$

where

$$\Sigma = \begin{pmatrix}
\sigma_{yy} & \sigma_{yx_1} & \cdots & \sigma_{yx_d} \\
\sigma_{xy_1} & \sigma_{x_1x_1} & \cdots & \sigma_{x_1x_d} \\
\cdots & \cdots & \cdots & \cdots \\
\sigma_{xy_d} & \sigma_{xdx_1} & \cdots & \sigma_{xdx_d}
\end{pmatrix}$$

is the full sample covariance matrix for the training data $\Gamma = \{(x_1, y_1), \ldots, (x_N, y_N)\}$. The entries of $\Sigma$ are estimated as follows:

$$\bar{x}_j = \frac{1}{N} \sum_{i=1}^{N} x_{ij}, \quad j = 1, \ldots, d$$

$$\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i$$

$$\sigma_{yy} = \frac{1}{N-1} \sum_{i=1}^{N} (y_i - \bar{y})^2$$

$$\sigma_{xy_j} = \frac{1}{N-1} \sum_{i=1}^{N} (x_{ij} - \bar{x}_j)(y_i - \bar{y}) = \sigma_{yx_i}, \quad j = 1, \ldots, d$$

$$\sigma_{x_jx_k} = \frac{1}{N-1} \sum_{i=1}^{N} (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k), \quad j = 1, \ldots, d; \quad k = 1, \ldots, d$$

The system (10) has $d + 2$ equations and $d + 2$ unknowns: $a_1, a_2, \ldots, a_{d+1}$ and the Lagrange multiplier $\lambda$ (also note that (11) guarantees the requirement $a_1 \neq 0$ of (8) for an arbitrary $\varepsilon > 0$). For any values $a_1, a_2, \ldots, a_{d+1}$, the first equation of (10) can always be solved by setting the free variable $\lambda$ appropriately:

$$\lambda = -\frac{2(\Sigma a)_1}{2\varepsilon}$$

We write out the other $d$ equations of (10) and omit the factor of 2, i.e.:

$$\begin{pmatrix}
\sigma_{x_1y} & \sigma_{x_1x_1} & \cdots & \sigma_{x_1x_d} \\
\sigma_{x_2y} & \sigma_{x_2x_1} & \cdots & \sigma_{x_2x_d} \\
\cdots & \cdots & \cdots & \cdots \\
\sigma_{xdy} & \sigma_{xdx_1} & \cdots & \sigma_{xdx_d}
\end{pmatrix}
\begin{pmatrix}
a_1 \\
a_2 \\
\cdots \\
a_{d+1}
\end{pmatrix} = 0$$

Recall that we are interested in the coefficients $\beta_j$ which can be written as $\beta_j = -\frac{a_{j+1}}{a_1}$ for $j = 1, \ldots, d$ (see (8)). Therefore we rearrange the system above in terms of $\beta_j$ (by multiplying out the $\sigma_{x_jy}a_1$ terms, bringing them to the other side, and dividing by $-a_1$):

$$\begin{pmatrix}
a_1\sigma_{x_1y} \\
a_1\sigma_{x_2y} \\
\cdots \\
a_1\sigma_{xdy}
\end{pmatrix} +
\begin{pmatrix}
\sigma_{x_1x_1} & \sigma_{x_1x_2} & \cdots & \sigma_{x_1x_d} \\
\sigma_{x_2x_1} & \sigma_{x_2x_2} & \cdots & \sigma_{x_2x_d} \\
\cdots & \cdots & \cdots & \cdots \\
\sigma_{xdx_1} & \sigma_{xdx_2} & \cdots & \sigma_{xdx_d}
\end{pmatrix}
\begin{pmatrix}
a_2 \\
a_3 \\
\cdots \\
a_{d+1}
\end{pmatrix} =
\begin{pmatrix}
0 \\
0 \\
\cdots \\
0
\end{pmatrix}$$
becomes
\[
\begin{pmatrix}
\sigma_{x_1x_1} & \sigma_{x_1x_2} & \cdots & \sigma_{x_1x_d} \\
\sigma_{x_2x_1} & \sigma_{x_2x_2} & \cdots & \sigma_{x_2x_d} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{x_dx_1} & \sigma_{x_dx_2} & \cdots & \sigma_{x_dx_d}
\end{pmatrix}
\begin{pmatrix}
a_2 \\
a_3 \\
\vdots \\
a_{d+1}
\end{pmatrix}
= \begin{pmatrix}
-a_1\sigma_{x_1y} \\
-a_1\sigma_{x_2y} \\
\vdots \\
-a_1\sigma_{xdy}
\end{pmatrix}
\]
and finally (divide by \(-a_1\) and substitute \(\beta_j = \frac{-a_{j+1}}{a_1}\) for \(j = 1, \ldots, d\))
\[
\begin{pmatrix}
\sigma_{x_1x_1} & \sigma_{x_1x_2} & \cdots & \sigma_{x_1x_d} \\
\sigma_{x_2x_1} & \sigma_{x_2x_2} & \cdots & \sigma_{x_2x_d} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{xdx_1} & \sigma_{xdx_2} & \cdots & \sigma_{xdx_d}
\end{pmatrix}
\begin{pmatrix}
\beta_1 \\
\beta_2 \\
\vdots \\
\beta_d
\end{pmatrix}
= \begin{pmatrix}
\sigma_{x_1y} \\
\sigma_{x_2y} \\
\vdots \\
\sigma_{xdy}
\end{pmatrix}
\]
(13)
One can show (see Appendix A) that this is exactly the same system which appears for solving linear least squares problems. Consequently the offset \(\beta_0\) should also be equivalent to the offset \(\beta_0 = \bar{y} - \beta^T \bar{x}\) of the least squares solution. To see why this is true, we start by looking at the offset \(b\) of the hyperplane:
\[
b = a^T \bar{u} - \frac{\sqrt{a^T \Sigma a}}{\sqrt{a^T \Sigma a + \sqrt{a^T \Sigma a}}} = \frac{1}{2\epsilon} (\bar{y} + \epsilon - \sum_{i=1}^{d} \beta_i \bar{x}_i) - \frac{1}{2\epsilon}
\]
Therefore, we have:
\[
\beta_0 = \frac{b}{a_1} = b \cdot 2\epsilon = \bar{y} - \sum_{j=1}^{d} \beta_j \bar{x}_j = \bar{y} - \beta^T \bar{x}
\]
(14)
From (13) and (14) we obtain our first theorem:

**Theorem 1** The linear MPMR algorithm computes the same parameters \(\beta_0, \beta_1, \beta_2, \ldots, \beta_d\) as the method of least squares.

**Proof.** Derivation of (13) and (14). □

We have shown that for the linear case the regression estimator we obtain from MPMR is the same one gets from least squares. The MPMR approach, however, allows us to compute a distribution free lower bound \(\Omega\) on the probability that future data points are predicted within \(\pm \epsilon\) correctly. We don’t know of any other formulation that yields a bound like that for the commonly used least squares regression. To actually compute the bound \(\Omega\), we estimate the covariance matrix \(\Sigma\) from the training data and solve (13) for \(\beta\). Then, for any given margin \(\epsilon > 0\) we set \(a_1 = \frac{1}{2\epsilon}\), \(a_{j+1} = -\beta_j a_1\) \((j = 1, \ldots, d)\) and calculate:
\[
\Omega = \frac{1}{4a^T \Sigma a + 1}
\]
(15)
The following theorem establishes that the \(\Omega\) calculated in (15) is a lower bound on the probability of future predictions being \(\pm \epsilon\) accurate (if we have perfect estimates for \(\bar{x}, \bar{y}, \Sigma\)).
Theorem 2. For any distributions $\Lambda_X$ with $E[\Lambda_X] = \bar{x}, V[\Lambda_X] = \Sigma_X$ and $\Lambda_\rho$ with $E[\Lambda_\rho] = 0, V[\Lambda_\rho] = \sigma^2$ (where $\bar{x}, \Sigma_X, \sigma$ are finite) and $x = (x_1, \ldots, x_d) \sim \Lambda_X, \rho \sim \Lambda_\rho$ let $y = f^*(x) + \rho$ be the (noisy) output of the true regression function. Assume perfect knowledge of the statistics $\bar{x}, \bar{y}, \Sigma$ (where $\Sigma$ is defined in (1)). Then the MPMR model $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ satisfies:

$$\inf_{(x,y) \sim (\bar{x},\bar{y},\Sigma)} \Pr\{|\hat{f}(x) - y| < \epsilon\} = \Omega$$

Proof. For any $d+1$ dimensional vector $z = (y, x_1, \ldots, x_d)$ we have:

- $z$ is classified as class $u \Leftrightarrow y > \hat{f}(x)$
- $z$ is classified as class $v \Leftrightarrow y < \hat{f}(x)$

(this follows directly from the construction of the $d+1$ dimensional hyperplane).

Now look at the random variables $z_{+\epsilon} = (y + \epsilon, x)$ and $z_{-\epsilon} = (y - \epsilon, x)$. The distribution of $z_{+\epsilon}$ satisfies $z_{+\epsilon} \sim ((\bar{y} + \epsilon, \bar{x}), \Omega)$ which has the same mean and covariance matrix as our (artificial) class $u \sim (\bar{u}, \Sigma_u)$. Similarly, we find that $z_{-\epsilon} \sim (\bar{v}, \Sigma_v)$. The MPM classifier guarantees that $\Omega$ is a lower bound of correct classification. Formally, correct classification is the probabilistic event that $z_{+\epsilon}$ is classified as $u$ and that at the same time $z_{-\epsilon}$ is classified as $v$. If we use the two equivalences above we obtain:

$$\Pr\{z_{+\epsilon} \text{ classified as } u \land z_{-\epsilon} \text{ classified as } v\} = \Pr\{y + \epsilon > \hat{f}(x) \land y - \epsilon < \hat{f}(x)\} = \Pr\{|\hat{f}(x) - y| < \epsilon\}$$

Since $\Omega$ is the infimum probability over all possible distributions for $z_{+\epsilon}$ and $z_{-\epsilon}$ it is also the infimum probability over the equivalent distributions for $x$ and $y$. This completes the proof that $\Omega$ is a lower bound probability for the MPMR model being $\pm \epsilon$ accurate. □

It is interesting to note that once we have calculated a model we can also apply (15) in the other direction, i.e. for a given probability $\Omega \in [0, 1]$ we can compute the margin $\epsilon$ such that $\Omega$ will be a lower bound on the probability of test points being $\pm \epsilon$ accurate. Simple algebra shows that $4a^T \Sigma a = (\sigma_{yy} - \sum_{j=1}^d \beta_j \sigma_{yx_j})/\epsilon^2$;

$$4(a_1, a_2, \ldots, a_{d+1}) \begin{pmatrix} \sigma_{yy} & \sigma_{yx_1} & \cdots & \sigma_{yx_d} \\ \sigma_{x_1y} & \sigma_{x_1x_1} & \cdots & \sigma_{x_1x_d} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_{x_dy} & \sigma_{x_dx_1} & \cdots & \sigma_{x_dx_d} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{d+1} \end{pmatrix}$$

We substitute $a_1 = \frac{1}{2\epsilon}$ and $a_{j+1} = -\beta_j * a_1$ for $j = 1, \ldots, d$:

$$4 \begin{pmatrix} 1/2\epsilon, -\beta_1/2\epsilon, \ldots, -\beta_d/2\epsilon \end{pmatrix} \begin{pmatrix} \sigma_{yy} & \sigma_{yx_1} & \cdots & \sigma_{yx_d} \\ \sigma_{x_1y} & \sigma_{x_1x_1} & \cdots & \sigma_{x_1x_d} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_{x_dy} & \sigma_{x_dx_1} & \cdots & \sigma_{x_dx_d} \end{pmatrix} \begin{pmatrix} 1/2\epsilon \\ -\beta_1/2\epsilon \\ \cdots \\ -\beta_d/2\epsilon \end{pmatrix}$$

We factor our $1/2\epsilon$ from each of the two vectors (the 4 cancels out with $(\frac{1}{2})^2$):

$$\frac{1}{\epsilon^2} \begin{pmatrix} 1, -\beta_1, \ldots, -\beta_d \end{pmatrix} \begin{pmatrix} \sigma_{yy} & \sigma_{yx_1} & \cdots & \sigma_{yx_d} \\ \sigma_{x_1y} & \sigma_{x_1x_1} & \cdots & \sigma_{x_1x_d} \\ \cdots & \cdots & \cdots & \cdots \\ \sigma_{x_dy} & \sigma_{x_dx_1} & \cdots & \sigma_{x_dx_d} \end{pmatrix} \begin{pmatrix} 1 \\ -\beta_1 \\ \cdots \\ -\beta_d \end{pmatrix}$$
We do the matrix vector multiplication on the right side. Note that \( \sum_{j=1}^{d} \sigma_{x_j} \beta_j = \sigma_{x_i} \) \( (i = 1, ..., d) \) from (13):

\[
\frac{1}{\epsilon^2} (1, -\beta_1, \ldots, -\beta_d) \begin{pmatrix}
\sigma_{yy} - \sum_{j=1}^{d} \sigma_{yx_j} \beta_j \\
\sigma_{x_1y} - \sigma_{x_1y} \\
\sigma_{x_{1}x_1} - \sigma_{x_{1}x_1} \\
\vdots \\
\sigma_{x_dy} - \sigma_{x_dy}
\end{pmatrix}
\]

The last \( d \) components of the column vector are zero, hence the expression simplifies to:

\[
\frac{1}{\epsilon^2} \left( \sigma_{yy} - \sum_{j=1}^{d} \beta_j \sigma_{yx_j} \right)
\]

The numerator \( \nu = \sigma_{yy} - \sum_{j=1}^{d} \beta_j \sigma_{yx_j} \) establishes the relation between \( \Omega \) and \( \epsilon \). It is important to note that the lower bound \( \Omega \) is also valid for linear and nonlinear (see section 4.1) regression models generated from other learning algorithms. However, for other models \( \beta \) may not satisfy (13), i.e. the last \( d \) components don’t zero-out and we have to compute the whole expression:

\[
\tilde{\nu} = (1, -\beta_1, \ldots, -\beta_d) \begin{pmatrix}
\sigma_{yy} & \sigma_{yx_1} & \cdots & \sigma_{yx_d} \\
\sigma_{x_1y} & \sigma_{x_1x_1} & \cdots & \sigma_{x_1x_d} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{x_{d}y} & \sigma_{x_{d}x_1} & \cdots & \sigma_{x_{d}x_d}
\end{pmatrix} \begin{pmatrix}
1 \\
-\beta_1 \\
\cdots \\
-\beta_d
\end{pmatrix}
\]

(16)

If we set \( \beta_0 \) according to (14) we can, for example, compute distribution free error bounds of an SVM for any given confidence \( \Omega \). We summarize these results in the following theorem:

**Theorem 3** For any linear regression model \( f(x) = \sum_{j=1}^{d} \beta_j x_j + \beta_0 \) we have the following relationships between the distribution free lower bound probability \( \Omega \) and the margin size \( \epsilon \):

\[
\Omega = \frac{1}{\tilde{\nu}/\epsilon^2 + 1} \\
\epsilon = \sqrt{\frac{\tilde{\nu} \cdot \Omega}{1 - \Omega}}
\]

where \( \tilde{\nu} \) is defined in (16).

**Proof.** Combining (15) and (16) we conclude that we can write \( \Omega \) as:

\[
\Omega = \frac{1}{4a^T \Sigma a + 1} = \frac{1}{\tilde{\nu}/\epsilon^2 + 1}
\]

for any basis function regression model (which completes the first part of the proof). If we solve this equation for \( \epsilon \) we obtain:

\[
\Omega = \frac{1}{\tilde{\nu}/\epsilon^2 + 1} \\
\Leftrightarrow (\tilde{\nu}/\epsilon^2 + 1)\Omega = 1 \\
\Leftrightarrow (\tilde{\nu} + \epsilon^2)\Omega = \epsilon^2 \\
\Leftrightarrow \tilde{\nu} \Omega = \epsilon^2(1 - \Omega) \\
\Leftrightarrow \epsilon = \sqrt{\frac{\tilde{\nu} \cdot \Omega}{1 - \Omega}} \quad \square
\]
4. Nonlinear Framework

We now extend the above linear regression formulation to nonlinear regression using the basis function formulation:

\[ f(x) = \sum_{i=1}^{k} \beta_i \Phi_i(x) + \beta_0 \]

where the basis functions \( \Phi_i : \mathbb{R}^d \rightarrow \mathbb{R} \) can be arbitrary nonlinear mappings.

4.1 Kernel Maps as Features

In this paper we focus our analysis and experiments on Mercer kernels and use the kernel maps \( K(x_i, \cdot) \) that are induced by the training examples as a basis. Instead of \( d \) features each input vector is now represented by \( N \) features; the kernel map evaluated at all of the other training inputs (including itself).

\[ f(x) = \sum_{i=1}^{N} \beta_i K(x_i, x) + \beta_0 \]

We then solve the regression problem with the linear MPMR, i.e. we minimize the following least squares expression:

\[ \min_{\beta} \sum_{i=1}^{N} (y_i - (\sum_{j=1}^{N} \beta_j z_{ij} + \beta_0))^2 \]

where \( z_{ij} = K(x_i, x_j) \), \( i = 1, ..., N; j = 1, ..., N \)

A new input \( x \) is then evaluated as:

\[ \hat{y} = \hat{f}(x) = \sum_{i=1}^{N} \beta_i K(x_i, x) + \beta_0 \]

Commonly used kernel functions include the linear \( K(s, t) = s^T v \), the polynomial \( K(s, t) = (s^T v + 1)^p \) and the Gaussian kernel \( K(s, t) = exp(-\frac{(s-t)^T(s-t)}{2\sigma^2}) \) (see for example Schölkopf and Smola, 2002). Note that the learning algorithm now has to solve an \( N \)-by-\( N \) system instead of a \( d \)-by-\( d \) system:

\[
\begin{pmatrix}
\sigma_{z_1z_1} & \sigma_{z_1z_2} & \cdots & \sigma_{z_1z_N} \\
\sigma_{z_2z_1} & \sigma_{z_2z_2} & \cdots & \sigma_{z_2z_N} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{z_Nz_1} & \sigma_{z_Nz_2} & \cdots & \sigma_{z_Nz_N}
\end{pmatrix}
\begin{pmatrix}
\beta_1 \\
\beta_2 \\
\vdots \\
\beta_N
\end{pmatrix}
= 
\begin{pmatrix}
\sigma_{z_1y} \\
\sigma_{z_2y} \\
\vdots \\
\sigma_{z_Ny}
\end{pmatrix}
\]

and setting

\[ \beta_0 = \bar{y} - \sum_{i=1}^{N} \beta_j \bar{z}_j \]
where
\[
\hat{y} = \bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i, \\
\bar{z}_j = \frac{1}{N} \sum_{i=1}^{N} z_{ij}, \\
\sigma_{z_{ij}} = \frac{1}{N-1} \sum_{i=1}^{N} (z_{ij} - \bar{z}_j)(y_i - \hat{y}) = \sigma_{yz}, \\
\sigma_{z_{ik}} = \frac{1}{N-1} \sum_{i=1}^{N} (z_{ij} - \bar{z}_j)(z_{ik} - \bar{z}_k) \quad j = 1, ..., N; \quad k = 1, ..., N.
\]

The proof of Theorem 3 generalizes in a straightforward way to the basis function framework:

**Theorem 4** For any basis function regression model \( f(x) = \sum_{i=1}^{N} \beta_i \Phi_i(x) + \beta_0 \) we have the following relationships between the distribution free lower bound probability \( \Omega \) and the margin size \( \epsilon \):

\[
\Omega = \frac{1}{\hat{\nu}/\epsilon^2 + 1}, \\
\epsilon = \sqrt{\frac{\hat{\nu} \Omega}{1 - \Omega}}
\]

where
\[
\hat{\nu} = (1, -\beta_1, \ldots, -\beta_N) \begin{pmatrix}
\sigma_{yy} & \sigma_{yz_1} & \cdots & \sigma_{yz_N} \\
\sigma_{zy_1} & \sigma_{zz_1} & \cdots & \sigma_{zz_d} \\
\cdots & \cdots & \cdots & \cdots \\
\sigma_{z_Ny} & \sigma_{z_Nz_1} & \cdots & \sigma_{z_Nz_N}
\end{pmatrix} \begin{pmatrix} 1 \\ -\beta_1 \\ \cdots \\ -\beta_N \end{pmatrix}
\]

**Proof.** Analogous to proof of Theorem 3. \( \square \)

### 4.2 A Different Approach

In previous work we proposed a different nonlinear MPMR formulation which is based on the nonlinear MPM for classification by Lanckriet et al. (2002a). Instead of using kernel maps as features, this formulation finds a minimax hyperplane in an higher dimensional space. Like kernelized support vector machines, one considers the (implicit) mapping \( \varphi : \mathbb{R}^d \rightarrow \mathbb{R}^h \) and its dot products \( \varphi(x_i)^T \varphi(x_j) =: K(x_i, x_j) \). This MPMR algorithm (we call it kernel-MPMR since it requires Mercer kernels) was also obtained by using the MPM classifier on an artificial data set where the dependent variable, shifted by \( \pm \epsilon \), is added as first component to the input vector (yielding the \( d+1 \)-dimensional vector \( (y, x_1, \ldots, x_d) \)). The kernel-MPMR model looks like:

\[
\sum_{i=1}^{2N} \gamma_i K(z_i, z) + \gamma_0 = 0
\]

where \( z_i = (y_i + \epsilon, x_{i1}, \ldots, x_{id}) \) for \( i = 1, \ldots, N \) and \( z_i = (y_{i-N} - \epsilon, x_{i-N,1}, \ldots, x_{i-N,d}) \) for \( i = N + 1, \ldots, 2N \). Evaluating a new point \( x \) amounts to plugging in \( z = (y, x) \) into (18) and solving the equation for \( y \). For an arbitrary kernel this would imply that we have to deal with a nonlinear equation which is at best computationally expensive and at worst not solvable at all. Therefore, the kernel is restricted to be \textit{output-linear}, i.e. we require:

\[
K((s_1, s_2, \ldots, s_{d+1}), (t_1, t_2, \ldots, t_{d+1})) = s_1 t_1 + K'(s_2, \ldots, s_{d+1}, t_2, \ldots, t_{d+1})
\]

The theory of positive definite kernels (see for example Schölkopf and Smola, 2002) states that positive definite kernels are closed under summation, which means that we could use
an arbitrary positive kernel for $K'$ and obtain a valid kernel $K$ (of course the linear output kernel is also positive definite). By using a kernel $K$ of the form (19), we can analytically solve (18) for $y$ and obtain the regression model:

$$y = (-2\epsilon) \left[ \left( \sum_{i=1}^{N} (\gamma_{i} + \gamma_{i+N})K'(x_{i}, x) \right) + \gamma_{0} \right]$$

If we compare the two different MPM regression algorithms we note that the computational cost of learning an kernel-MPMR model is bigger because instead of an $N \times N$ system we have to solve an $2N \times 2N$ system to obtain the coefficients $\gamma$. For MPMR we are not restricted to Mercer kernels but can use any set $\Phi_{i}$ of nonlinear basis functions. It is an open question as to whether MPMR and kernel-PMMR are equivalent if we use Mercer kernels for the MPMR.

5. Robust Estimates

The MPMR learning algorithm estimates the covariance matrix from the training data. Of course, these statistical estimates will be different from the true values and thus the regression surface itself and the lower bound probability $\Omega$ may be inaccurate. Especially for the nonlinear case the problem of overfitting the training data with highly sensitive kernel functions arises, so our goal is to find kernels that produce robust estimates. One way of dealing with the problem of selecting the right kernel (i.e. model) is to do cross validation on the training data and pick the kernel which yields the best result. In this paper we explore a different approach which is computationally much more efficient. The idea is to randomly select a fraction $\alpha$ of rows that will be left out when calculating the covariance matrix (note that we measure the covariance columnwise, i.e. deleting rows will not affect the dimension of the covariance matrix). If the covariance matrix of the remaining training data is "reasonably close" to the covariance matrix of all the training data, we consider our model to be robust.

We now define in mathematical terms what "reasonably close" means. The objective of the MPMR algorithm is to maximize the lower bound probability $\Omega$ which is determined by the term $a^{T}\Sigma a$. Therefore, we measure the sensitivity of an MPMR model by looking at the sensitivity in $a^{T}\Sigma a$. We call a model $\alpha$-robust if the relative error for this expression is $\leq \alpha$ when leaving a fraction of $\alpha$ points out for computing the covariance matrix:

**Definition**

- $\Sigma_{s}$ - true covariance matrix
- $\Sigma$ - cov. matrix of all $N$ examples
- $\Sigma_{\alpha}$ - cov. matrix of $(1-\alpha)N$ examples

\[ a_{s} \text{ minimizes } a_{s}^{T}\Sigma_{s}a_{s} \text{ s.t. } a_{s1} = \frac{1}{2\epsilon} \]

\[ a \text{ minimizes } a^{T}\Sigma a \text{ s.t. } a_{1} = \frac{1}{2\epsilon} \]

\[ a_{\alpha} \text{ minimizes } a_{\alpha}^{T}\Sigma_{\alpha}a_{\alpha} \text{ s.t. } a_{\alpha1} = \frac{1}{2\epsilon} \]

A model is $\alpha$-robust $\iff \frac{a_{\alpha}^{T}\Sigma_{\alpha}a_{\alpha} - a_{s}^{T}\Sigma_{s}a_{s}}{a_{\alpha}^{T}\Sigma_{\alpha}a_{\alpha}} \leq \alpha$

We can not apply this definition directly since we don’t know the true covariance matrix $\Sigma_{s}$ but we will prove that under the following two assumptions we can obtain a formula that
determine the sensitivity of \( \Omega \), Lanckriet et al. (2002b) use sampling methods to determine the plug-in estimates derived from the training data. Both approaches rely on sampling does not involve \( \Sigma \).

1) \( a^T \Sigma a \leq a^T \Sigma a \) (whole model is more accurate than \((1 - \alpha)N\) model)

2) \( \frac{a^T \Sigma a}{a^T \Sigma a} \geq \frac{a^T \Sigma a}{a^T \Sigma a} \) (for \( \alpha \) small \( a \) is closer to \( a \) than \( a \) is to \( a \))

Under these assumptions the following theorem holds:

**Theorem 5** If a model is \( \alpha \)-robust then the ratio \( \alpha : \frac{a^T \Sigma a - a^T \Sigma a}{a^T \Sigma a} \geq 1 \). In formulas:

\[
\frac{a^T \Sigma a - a^T \Sigma a}{a^T \Sigma a} \leq \alpha \Rightarrow R = \alpha : \frac{a^T \Sigma a - a^T \Sigma a}{a^T \Sigma a} \geq 1
\]  

(20)

**Proof of Theorem:**

from 2) \( \frac{a^T \Sigma a}{a^T \Sigma a} \geq \frac{a^T \Sigma a}{a^T \Sigma a} \)

from 1) \( a^T \Sigma a \leq a^T \Sigma a \Rightarrow \frac{a^T \Sigma a}{a^T \Sigma a} \leq \frac{a^T \Sigma a}{a^T \Sigma a} \)

from premise) \( \frac{a^T \Sigma a - a^T \Sigma a}{a^T \Sigma a} \leq \alpha \Leftrightarrow \frac{a^T \Sigma a - a^T \Sigma a}{a^T \Sigma a} \geq 1 - \alpha \)

combine inequalities:

\[
\frac{a^T \Sigma a}{a^T \Sigma a} \geq \frac{a^T \Sigma a}{a^T \Sigma a} \geq \frac{a^T \Sigma a}{a^T \Sigma a} \geq 1 - \alpha
\]

\[
\frac{a^T \Sigma a}{a^T \Sigma a} \geq 1 - \alpha \Leftrightarrow 1 - \frac{a^T \Sigma a}{a^T \Sigma a} \leq \alpha \Leftrightarrow \alpha \geq \frac{a^T \Sigma a - a^T \Sigma a}{a^T \Sigma a} \Leftrightarrow \alpha : \frac{a^T \Sigma a - a^T \Sigma a}{a^T \Sigma a} \geq 1
\]

\( \Box \)

We refer to the number \( R = \alpha : \frac{a^T \Sigma a - a^T \Sigma a}{a^T \Sigma a} \) as robustness measure or confidence measure.

In our experiments we apply the theorem to show that a model is not \( \alpha \)-robust. A model where we find that \( R < 1 \) will be rejected because of too much sensitivity in \( \Omega \).

A different approach for constructing a robust minimax probability machine has been investigated by Lanckriet et al. (2002b) for the task of classification. There the main idea is to put some error bounds on the plug-in estimates of mean and covariance. Instead of a single mean, the robust classifier now considers an ellipsoid of means which is centered around the estimated mean and bound in size by an error bound parameter \( \nu \). Similarly, instead of a single covariance matrix one now has a matrix ball, where each matrix is at most \( \rho \) away (measured in Frobenius norm) from the estimated covariance matrix. When a robust model is calculated, the lower bound probability \( \Omega \) (for correct classification) decreases when either \( \nu \) or \( \rho \) increase. The model itself is independent of \( \nu \), but it can depend on \( \rho \) and a small toy example shows (see Lanckriet et al., 2002b, p. 578) that a robust model \( (\rho > 0) \) can yield a better performance than a simple one (where \( \rho = 0 \)).

From a theoretical point of view the two approaches differ in a way that our method is output oriented (it considers the sensitivity of \( \Omega \) which is the objective that an MPM maximizes) whereas the other method is more input oriented (it considers the sensitivity of the plug-in estimates derived from the training data). Both approaches rely on sampling methods to make robust estimates: We sample a fraction \((1 - \alpha)\) percent) of examples to determine the sensitivity of \( \Omega \), Lanckriet et al. (2002b) use sampling methods to determine reasonable values for the parameters \( \nu \) and \( \rho \). Our method, however, has the advantage
that it provides a straightforward way to do model selection: choose the model that has a maximal $\Omega$ while satisfying $R > 1$. It is not immediately clear how a strategy for model selection can be implemented with the approach of Lanckriet et al. (2002b). It is an open question as to which practical applications either approach is preferable to the other.

6. Experimental Results

In this section we present the results of the robust MPMR for toy problems and real world regression benchmarks. For all of the data sets we used a Gaussian kernel $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$ to obtain a nonlinear regression surface. We plot the behavior of $\Omega$ and $R$ for a range of kernel parameters $\sigma$ which is centered around $\bar{\sigma}$ with $2\bar{\sigma}^2 = 0.3 \cdot d$ where $d$ is the input dimension of the data set. Each step, the parameter $\sigma$ is increased by a factor of 1.1. For all our experiments we used $\alpha = 0.1$ as the fraction of points that are left out for the estimation of $R$. To quantify how accurate our lower bound estimate $\Omega$ is, we compute the number of test points that are within $\pm \epsilon$ correct ($\Omega^T$) for each test run. The common behavior in all data sets is that small values for $\sigma$ (which yield bad models due to overfitting) have a tiny value for $R$, i.e. the corresponding model will be rejected. As $\sigma$ increases, we obtain more robust estimates of the covariance matrix and therefore more robust MPMR models that will not be rejected. The point of interest is where the robustness measure $R$ crosses 1 because this will be the place where we become confident in our model and our $\Omega$ bound.

6.1 Toy Example

We used the sinc function $x \mapsto \sin(\pi x)/(\pi x)$ as a toy example for our learning algorithm. A noise term drawn from a Gaussian distribution with mean 0 and variance $\sigma^2 = 0.1$ was added to the examples. Both our training and test set consisted of $N_{train} = N_{test} = 200$ examples (see Figure 3a). We tried 30 different values for $\sigma$ and averaged over 100 test runs for each $\sigma$. For both a fixed $\Omega$ and a fixed $\epsilon$ we observed that $\Omega$ became a true lower bound on $\Omega^T$ right at the point where our confidence measure crossed 1 (see Figures 2a,2b). This toy experiment showed that the robustness ratio $R$ is suitable for measuring the robustness of $\Omega$ estimates.

6.1.1 Comparison to Gaussian Processes

In this section we also use the sinc function as toy data but add a noise variable $\rho$ that is not drawn from a (single) Gaussian distribution. Instead, we generate a distribution from two separate Gaussians: With probability $p_1 = 0.5$, $\rho$ is drawn from $\mathcal{N}(\mu, 1)$ and with probability $p_2 = 0.5$, $\rho$ is drawn from $\mathcal{N}(-\mu, 1)$ (See figure 3b). We tested 30 values for $\mu$ in the range $\mu = 0.1, 0.2, \ldots, 3.0$. Both the training and test set had $N_{train} = N_{test} = 100$ examples. The Gaussian Process implementation we used is provided by Gibbs (1997b). We calculated the error bounds of the Gaussian Process for $\sigma = 1$ standard deviation. In the MPMR framework this corresponds to a value of $\Omega = 0.6829$. By using (17) we computed
the MPMR error bounds for each distribution. As expected, the Gaussian Process error bounds $\epsilon_{GP}$ were tighter (i.e. more optimistic) than the worst case MPMR error bounds $\epsilon_{MPMR}$ (see Figure 4a).

For small values of $\mu$ where the noise distribution is still close to a (single) Gaussian the Gaussian Process error bounds are still accurate, i.e. they match to the measured percentage of $\pm \epsilon$ accurate predictions. As $\mu$ increases and the noise distribution becomes less Gaussian, the Gaussian Process error bounds are no longer a good estimate on the prediction accuracy. The MPMR error bounds, however, are valid for all possible values of $\mu$ (See figure 4b):

Figure 2: Toy example

Figure 3:

a) sinc function with Gaussian noise

b) "Two Gaussian" noise for $\mu = 2
6.2 Boston Housing

The first real world data set we experimented with is the Boston Housing data which consists of $N = 506$ examples, each with $d = 13$ real valued inputs and one real valued output. We divided the data randomly into a training set of size $N_{\text{train}} = 481$ and a test set of size $N_{\text{test}} = 25$. We tried 75 different values for $\sigma$, and for each of these we averaged our results over 100 runs. For the first plot (Figure 5a) we fixed $\Omega = 0.5$ and calculated the corresponding $\epsilon$ for the model. Additionally, we plotted the confidence measure $R$ and the percentage $\Omega^T$ of test points within $\pm \epsilon$ of our regression surface. As the number $R$ became bigger than 1, we noticed that $\Omega$ was a true lower bound, i.e. $\Omega < \Omega^T$. The second plot (Figure 5b) has a fixed $\epsilon$ which was set to the value of $\epsilon$ where $R$ became greater than 1 in the first plot. We recomputed $\Omega$ and $\Omega^T$ and saw as before that $\Omega < \Omega^T$ for $R > 1$. It is also remarkable that the best regression model (i.e. the one that maximizes $\Omega^T$ for a given $\epsilon$) is right at the point where $R$ crosses 1. The next plot (Figure 5c) shows the percentage of "bad models" (where $\Omega > \Omega^T$) which is below 10% for $R > 1$ and approaches zero as $R$ increases. This shows that the more robust a model is in terms of $R$ the more unlikely it is for $\Omega$ not to be a true lower bound on the probability of $\pm \epsilon$ accurate prediction. The fourth plot (Figure 5d) shows false positives and false negatives for the robustness measure $R$ itself. Note that we have a high false negative rate shortly before $R$ crosses 1. The reason for this effect is that a model with, say $R = 0.8$, will be rejected, but $\Omega < \Omega^T$ may still hold because $\Omega$ is a lower bound for all possible distributions, i.e. for specific distributions $\Omega$ is in general a pessimistic estimate. False positives are rare: when we accept a model because of $R > 1$ it is very unlikely ($< 5\%$) that the model will be inaccurate in a sense that $\Omega^T$ is smaller than $\Omega$. Our last plot (Figure 5e) shows the connection of our regression approach to the more commonly used one which tries to minimize the (normed) mean squared error. We found that the normed MSE $\frac{\text{MSE}_{\text{model}}}{\text{MSE}_{\text{mean}}}$ has a minimum right at the point where $R$ crosses 1 and $\Omega^T$ had a maximum in the second plot.
a) for $R > 1$ we have $\Omega^T > \Omega$, i.e. $\Omega$ is a lower bound

b) for $\epsilon = \text{const}$ the maximal $\Omega^T$ is around $R = 1$

c) for $R > 1$ the number of "bad models" < 10%

d) many false negatives ($\Omega$ is worst case bound)

e) minimum MSE corresponds to maximum $\Omega^T$

Figure 5: Boston Housing Results

6.3 Abalone

The ABALONE data set contains $N = 4177$ examples, each with 8 inputs and one realvalued output. One of the input features is not a real number but can take on one of three distinct values. We split up the discrete-valued feature into 3 real valued features, setting one component to 1 and the other two to 0, which increases the input dimension to $d = 10$. The data was then divided randomly into $N_{\text{train}} = 1000$ training examples and $N_{\text{test}} = 3177$ test examples. As in the Boston experiment we went over a range of kernel parameters (here 30) and averaged the results over a number of test runs (here 25). Again, we plotted $\epsilon$ for a constant $\Omega$, $\Omega$ for a constant $\epsilon$, the "bad models", false positives/negatives for $R$, and the normed mean squared error (see Figure 6). The overall behavior was the same that we observed for the Boston Housing data: Models with a confidence measure of $R > 1$ yielded accurate lower bounds $\Omega$. The false positive and false negative rates were more extreme for this data set: We didn’t get false positives anymore but on the other side the false negative percentage was very high for $R$ slightly less than 1. Also, models with a very insensitive kernel parameter tended to be about as good as models with a kernel parameter in the range of $R \approx 1$. We ascribe this effect to the high noise level of the Abalone data set which puts a bias towards insensitive models.
6.4 Kinematic Data

The KIN8NM data set consists of $N = 8192$ examples, each with 8 real valued inputs and one real valued output. We used the same parameters as in the ABALONE data set: $N_{\text{train}} = 1000$ training examples, a range of 30 kernel parameters and 25 test runs to average the results for each kernel parameter. As in the previous experiments, the robustness measure $R$ provided a good estimate for the point where $\Omega$ becomes a lower bound on $\Omega^T$. For this data set, however, the maximum $\Omega^T$ and the minimum MSE were not at the point where $R$ crossed 1, but occurred at a point where $R$ was still less than one (See Figure 7).

7. Conclusion & Future Work

We have presented the MPMR framework as a new approach to address regression problems where models are represented as nonlinear (or linear) basis function expansions (see (2) and (3)). The objective of MPMR is to find a model that maximizes the worst case probability that future data is predicted accurately within some $\pm \epsilon$ bound. Also, we introduced a general approach to reduce any regression problem to a binary classification problem by shifting the dependent variable both by $+\epsilon$ and by $-\epsilon$. In this paper, we used MPMC as underlying classifier which gives us a distribution free probability bound $\Omega$, and therefore a regression algorithm that can estimate its ability to make $\pm \epsilon$ correct predictions. Remarkably, the resulting MPMR is equivalent to the method of least squares which is a standard technique in statistical analysis. One consequence of our work is that it is now possible to
calculate distribution free confidence intervals for least squares solutions (formerly this was only feasible under Gaussian or other specific, often unrealistic, distributional assumptions).

For nonlinear problems we used Mercer kernels as basis functions for our regression model, however, our formulation is not limited to Mercer kernels but works with arbitrary basis functions. Typically, there are kernel parameters (e.g. the width $\sigma$ of the Gaussian kernel) that need to be adjusted properly to obtain good models. A common technique to find these parameters is to do 10-fold cross validation which is computationally expensive. Our approach, however, does not involve cross validation but is based on a robustness measure $R$. $R$ is calculated by sampling parts of the training data and looking on the effect on $\Omega$—the objective of the MPMR. This is computationally more efficient since $N$-fold cross validation requires to build $N$ models whereas we only need to build two models. Experiments with real world data sets have shown that the robustness measure works well for filtering out models that overfit the data and consequently violate the property of $\Omega$ as a true lower bound. Empirical evidence is given showing that $R$ can be used to find optimal kernel parameters and therefore is suitable for model selection.

Future research on MPMR will focus on investigating the possibility of doing model selection solely based on $R$ without any cross validation. As a consequence of Theorem 4 (Section 4.1), we are looking into calculating lower bounds $\Omega$ for basis function regression models that are coming from other algorithms (e.g. the support vector machine regression described in Smola and Schölkopf (1998)). In combination with the robustness measure $R$ (and possibly a model selection criterion) this idea could help to improve the accuracy of a whole class of regression algorithms. Also, the current implementation requires solving an
Robust Minimax Probability Machine Regression

$N \times N$ linear system of equations (complexity $O(N^3)$) and therefore is prohibitive for large $N$ (where $N$ is the number of training examples). As a result we are currently working on Sparse MPMR.

Finally, we note that for $N \to \infty$ MPMR converges to the true $\Omega$ (and the optimal model) since the estimates of mean and covariance matrix become more and more accurate as the number of training examples ($N$) increases. We are currently looking into the rate of convergence for $\Omega$ as $N$ increases, which should be directly related to the rate of convergence for the mean and covariance estimates.

A Matlab implementation of the MPMR algorithm can be obtained from:
http://nago.cs.colorado.edu/~strohman/papers.html

8. Acknowledgements

We are grateful to Mark Gibbs for providing an implementation to do regression estimation with Gaussian Processes.

9. Appendix A

The least squares problem can be formulated as the following linear system (see for example Wakcerly et al., 2002, chap. 11):

$$X^TX\beta = X^TY$$

where

$$X = \begin{pmatrix}
1 & x_{11} & \cdots & x_{1d} \\
1 & x_{21} & \cdots & x_{2d} \\
\vdots & \vdots & \ddots & \vdots \\
1 & x_{N1} & \cdots & x_{Nd}
\end{pmatrix} \quad Y = \begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N
\end{pmatrix}$$

If we write out (21) we obtain (from now on all sums are from $i = 1$ to $N$):

$$\begin{pmatrix}
N & \sum x_{i1} & \cdots & \sum x_{id} \\
\sum x_{1i} & \sum x_{11} & \cdots & \sum x_{1d} \\
\vdots & \vdots & \ddots & \vdots \\
\sum x_{Ni} & \sum x_{Ni} & \cdots & \sum x_{Nd}
\end{pmatrix} \begin{pmatrix}
\beta_0 \\
\beta_1 \\
\vdots \\
\beta_d
\end{pmatrix} = \begin{pmatrix}
\sum y_i \\
\sum x_{i1}y_i \\
\vdots \\
\sum x_{id}y_i
\end{pmatrix}$$

Now we write all the sums in terms of means and covariances:

$$\begin{align*}
\sum y_i &= N\bar{y} \\
\sum x_{ij} &= Nx_j & j = 1, \ldots, d \\
\sum x_{ij}y_i &= (N-1)\sigma_{x_jy} + N\bar{x}_j\bar{y} & j = 1, \ldots, d \\
\sum x_{ij}x_{ik} &= (N-1)\sigma_{x_jx_k} + N\bar{x}_j\bar{x}_k & j = 1, \ldots, d; k = 1, \ldots, d
\end{align*}$$

where the last two lines are obtained by noting that:

$$\begin{align*}
\sigma_{x_jx_k} &= \frac{1}{N-1} \sum (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k) \\
\sigma_{x_jx_k} &= \frac{1}{N-1} \left( \sum x_{ij}x_{ik} - \sum x_{ij}\bar{x}_k - \sum x_{ik}\bar{x}_j + \sum \bar{x}_j\bar{x}_k \right) \\
\sigma_{x_jx_k} &= \frac{1}{N-1} \left( \sum x_{ij} - N\bar{x}_j\bar{x}_k - N\bar{x}_j\bar{x}_k + N\bar{x}_j\bar{x}_k \right) \\
(N-1)\sigma_{x_jx_k} &= \sum x_{ij}x_{ik} - N\bar{x}_j\bar{x}_k \\
\sum x_{ij}x_{ik} &= (N-1)\sigma_{x_jx_k} + N\bar{x}_j\bar{x}_k
\end{align*}$$
If we plug these expressions into the linear system above we have:

\[
\begin{pmatrix}
N
N x_1 \\
N x_1 (N-1) \sigma_{x_1 x_1} + N \bar{x}_1 \bar{x}_1 \\
\vdots \\
N x_d (N-1) \sigma_{x_d x_1} + N \bar{x}_d \bar{x}_1 \\
\end{pmatrix}
\begin{pmatrix}
N \bar{x}_1 \\
(N-1) \sigma_{x_1 x_1} + N \bar{x}_1 \bar{x}_1 \\
\vdots \\
(N-1) \sigma_{x_d x_1} + N \bar{x}_d \bar{x}_1 \\
\end{pmatrix}
\begin{pmatrix}
\beta_0 \\
\beta_1 \\
\vdots \\
\beta_d \\
\end{pmatrix}
= 
\begin{pmatrix}
N \bar{y} \\
(N-1) \sigma_{x_1 y} + N \bar{x}_1 \bar{y} \\
\vdots \\
(N-1) \sigma_{x_d y} + N \bar{x}_d \bar{y} \\
\end{pmatrix}
\]

If we look at the first equation we retrieve the familiar formula for \( \beta_0 \):

\[
N \beta_0 + \sum_{j=1}^{d} N x_j \beta_j = N \bar{y} 
\Leftrightarrow \beta_0 = \bar{y} - \sum_{j=1}^{d} \beta_j x_j
\]

Now look at the \( j+1 \)th \((j = 1, \ldots, d)\) equation which has \( x_j \) as the common variable:

\[
N \bar{x}_j \beta_0 + \sum_{k=1}^{d} \beta_k [(N-1) \sigma_{x_j x_k} + N \bar{x}_j \bar{x}_k] = (N-1) \sigma_{x_j y} + N \bar{x}_j \bar{y}
\Leftrightarrow N \bar{x}_j (\bar{y} - \sum_{k=1}^{d} \beta_k \bar{x}_k) + N \sum_{k=1}^{d} \beta_k \bar{x}_j \bar{x}_k + (N-1) \sum_{k=1}^{d} \beta_k \sigma_{x_j x_k} = (N-1) \sigma_{x_j y} + N \bar{x}_j \bar{y}
\Leftrightarrow N \bar{x}_j \bar{y} + (N-1) \sum_{k=1}^{d} \beta_k \sigma_{x_j x_k} = (N-1) \sigma_{x_j y} + N \bar{x}_j \bar{y}
\Leftrightarrow (N-1) \sum_{k=1}^{d} \beta_k \sigma_{x_j x_k} = (N-1) \sigma_{x_j y}
\Leftrightarrow \sum_{k=1}^{d} \beta_k \sigma_{x_j x_k} = \sigma_{x_j y}
\]

The last expression matches exactly the \( j \)th equation of (13). Since this is true for all \( j = 1, \ldots, d \) we have shown that (13) / (14) describe the same system that appears for Least Squares problems. \( \Box \)

References


