KERNELS EVALUATION OF SVM-BASED ESTIMATORS FOR INVERSE SCATTERING PROBLEMS

Emanuela Bermani, Andrea Boni, Aliaksei Kerhet, Andrea Massa

2004

Technical Report # DIT-04-062

ν–based SVM Regression for Inverse Scattering Problems

Andrea Boni Emanuela Bermani Aliaksei Kerhet
Andrea Massa

Department of Information and Communication Technology
University of Trento
Via Sommarive, 14, 38050 Povo (Trento), Italy
andrea.boni@ing.unitn.it
emanuela.bermani@ing.unitn.it
kerhet@dit.unitn.it
andrea.massa@ing.unitn.it

Abstract

Buried object detection by means of microwave-based sensing techniques is faced in biomedical imaging, mine detection etc. Whereas conventional methods used for such a problem consist in solving nonlinear integral equations, this work considers a recently proposed approach [1] based on Support Vector Machines, the techniques that proved to be theoretically justified and effective in real world domains. Simulation is carried out on synthetic data generated by Finite Element code and a PML technique; noisy environments are considered as well. Results obtained for cases of polynomial and Gaussian kernels are presented and discussed.

Keywords: Support Vector Machines, Statistical Learning, Microwave Inverse Scattering, Model Selection.

1 Introduction

The relation between forward and inverse problems can be expressed in the following way: if for the forward problem one seeks a consequence of a cause, then inverse problem requires to restore a cause for an observed consequence. In particular, an inverse scattering problem requires the determination of unknown dielectric properties of scatterers from the scattered field information. Such a problem arises in various areas, such as biomedical imaging, geophysics, remote sensing, and non-destructive evaluation, when inner properties of a body are deduced from its exterior measurements.

The problem is normally formulated in terms of an integral equation, which is iteratively solved by means of generally nonlinear minimization techniques. High computational cost of this approach could lead to its impracticability when real-time performance is required. In addition, the problem is ill-posed, that is, a small error of measured data can bring to significant errors of estimated parameters.

However, there are circumstances when one has (sometimes restricted) amount of *a-priori* information about the problem in the form of cause-consequence pairs. Furthermore, in many situations one does not need to recover exhaustive electromagnetic properties of an object under analysis (relative permittivity and conductivity as functions of spatial coordinates), but only to estimate some of object's properties (e.g. scatterers presence or absence). Recovering exaustive properties in this case seems to be redundant. In other words,

> *when solving a given problem, try to avoid solving a more general problem as an intermediate step* [2].

Such circumstances give opportunity to solve the problem using *learning by examples* approaches, in particular such popular techniques as Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs). The advantages of the latter are: 1) one has to solve a constrained quadratic optimization problem (instead of multiextremal minimization for ANNs), 2) SVMs are based on Statistical Leaning Theory that gives the possibility to control the model's complexity and, hence, to control its generalization ability [2].

This paper considers recently proposed SVM-based approach to buried objects detection problem [1]. The approach performance is estimated and compared for two different SVM configurations: Gaussian kernel SVM and polynomial kernel SVM. The obtained results demonstrate that using polynomial kernel along with slightly sophisticated model selection criterion can deliver higher accuracy to the problem under consideration than the Gaussian kernel.

The initial data for training, model selection, and testing have been synthetically obtained by means of Finite Element code and a PML technique. Environments with a number of *signal-to-noise ratios* (SNRs) are considered.

The paper is organized as follows: Section 2 describes the geometry of the problem under consideration and presents its mathematical and statistical learning statements. Section 3 is devoted to brief introduction to the SVM regression technique. Section 4 discusses the problem of model selection. Section 5 deals with the description of simulation steps; the results are presented and discussed as well. In Section 6 conclusions from the obtained results are drawn, and future work directions are given.

2 Inverse Scattering Problem Posing

This Section briefly describes the geometry, the physics, and the statistical learning formulation of the problem under consideration (a two-dimensional half-space, see Fig. 1).

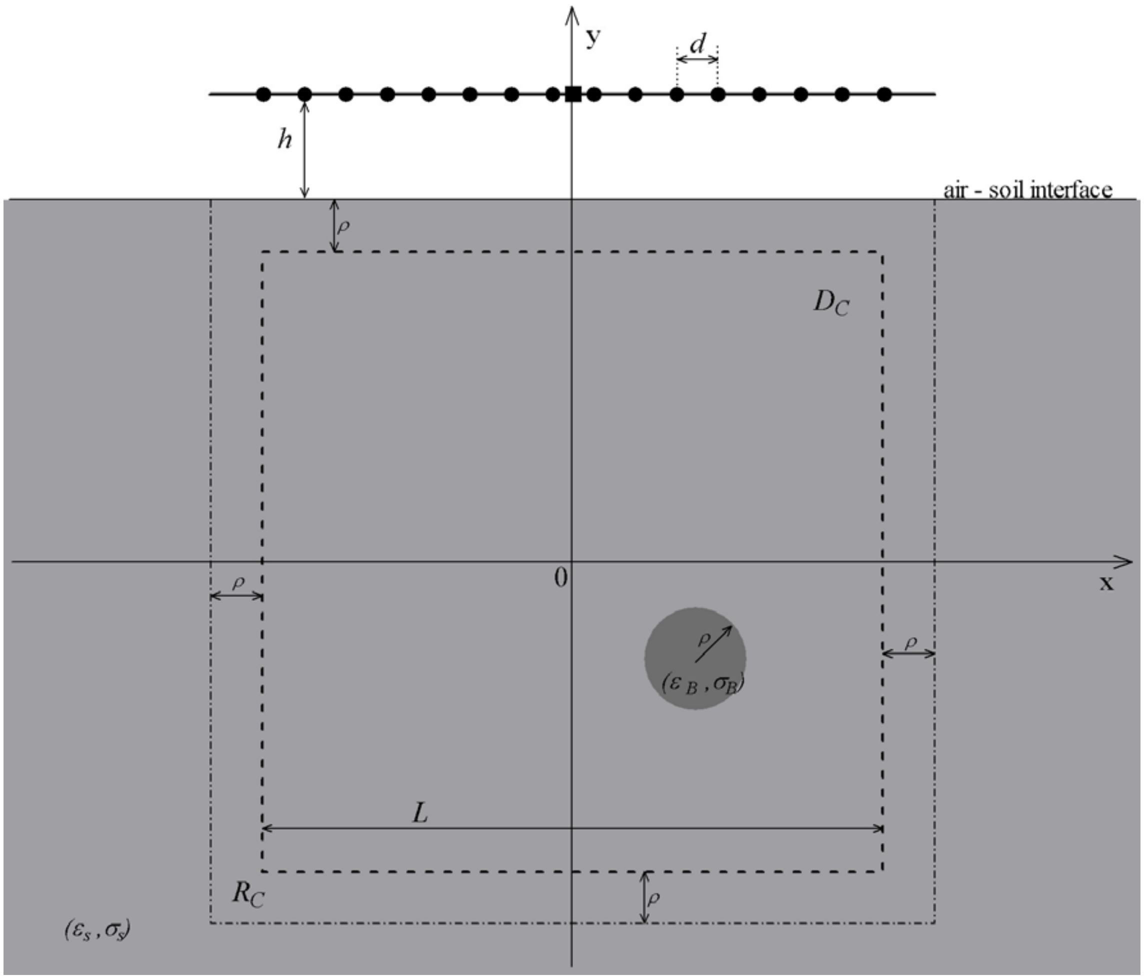


Figure 1: The geometry of the problem.

A homogeneous circular cylindrical scatterer with center coordinates (x_{act}, y_{act}) and radius ρ is buried into the homogeneous soil inside the square region R_C (chained line). Hence, the domain under consideration for the cylinder centers D_C is the square located inside R_C at a distance of ρ from its borders (dashed line). The coordinate origin is associated with the center of D_C.

Multiple transmitters/receivers with the coordinates (x_{tr}, y_{tr}), $tr = 1, \ldots, TR$ and (x_{rs}, y_{rs}), $rs = 1, \ldots, RS$ are located at the height h above the air-soil interface. The soil's and the scatterer's dielectric properties are given by complex constants $\tau_S = (\varepsilon_S - 1) - j\frac{\sigma_S}{2\pi f \varepsilon_0}$ and $\tau_B = (\varepsilon_B - 1) - j\frac{\sigma_B}{2\pi f \varepsilon_0}$ respectively.

Parameter	Value	Parameter	Value
λ	0.6 m	d	$\lambda/15$
L	λ	ε_S	8.0
ρ	$\lambda/12$	σ_S	0.025 S/m
h	$\lambda/6$	ε_B	5.0
TR	1, in the centre	σ_B	0.0 S/m
RS	16, equally spaced		

Table 1: Values of the parameters

The transmitter with coordinates (x_{tr}, y_{tr}) radiates monochromatic electromagnetic field with free-space wavelength λ in microwave range. The electric field collected at the point (x_{rs}, y_{rs}) is

$$\begin{aligned} &E^{tot}(x_{rs}, y_{rs}|x_{tr}, y_{tr}) = E^{inc}(x_{rs}, y_{rs}|x_{tr}, y_{tr}) + \\ &+k^2 \int_{R_C} E_S(x, y|x_{tr}, y_{tr}) \cdot G_S(x_{rs}, y_{rs}; x, y) \cdot \tau(x, y) dx\, dy, \end{aligned} \tag{1}$$

$$\text{where} \quad \tau(x, y) = \begin{cases} \tau_B & \text{if } \sqrt{(x - x_{act})^2 + (y - y_{act})^2} \le \rho \\ \tau_S & \text{for the rest of } R_C\ , \end{cases} \tag{2}$$

where $E^{inc}(x_{rs}, y_{rs}|x_{tr}, y_{tr})$ is the electric field collected at the point (x_{rs}, y_{rs}) in case of absence of the scatterer; $E_S(x, y|x_{tr}, y_{tr})$ is the electric field inside R_C in case of the scatterer's presence; $G_S(x_{rs}, y_{rs}; x, y)$ is the Sommerfeld-Green function for the half-space geometry (for details see [3, 1] and references therein).

The values of the different geometric and physics parameters assumed in given paper are summarized in Table 1.

Inverse scattering problem in this case consists in recovering the location of the scatterer's center on the basis of known values of

$$\begin{aligned} &E^{tot}(x_{rs}, y_{rs}|x_{tr}, y_{tr}), \\ &rs = 1, \ldots, RS \\ &tr = 1, \ldots, TR. \end{aligned} \tag{3}$$

In terms of statistical learning, (3) forms a vector of *features (inputs)*, while horizon and depth coordinates form a vector of *outcomes (outputs)*. The learning process consists in building a prediction model on the basis of the set of available observations *(examples)*. Example means a known input-output pair, and the set of such pairs used for building a prediction model is called *training set* Γ_{train}.

Thus, for the inverse scattering problem stated above, feature vector $\boldsymbol{\chi}$ consists of $N = 2{\cdot}TR{\cdot}RS$ scalar features (this follows from (3) after taking into consideration the fact that every $E^{tot}(x_{rs}, y_{rs}|x_{tr}, y_{tr})$ consists of real and imaginary parts). The output vector $\boldsymbol{\upsilon}$ is a 2-vector: $\boldsymbol{\upsilon} = (\upsilon^x,\ \upsilon^y)$, where υ^x and υ^y denote horizon and depth coordinates respectively. Let us denote the number of examples in Γ_{train} by l. In this case

$$\Gamma_{train} = \{(\boldsymbol{\chi}_i, \boldsymbol{\upsilon}_i),\ i = 1, \ldots, l\} \tag{4}$$

3 SVM Regression Formulation

Support Vector Machines (SVMs) [2, 4, 5] are learning by examples techniques introduced by V. Vapnik. Their advantage over other approaches like ANNs is due to such relevant aspects as 1) reduction of problem to solving constrained quadratic optimization problem (CQP) and 2) the solid Vapnik's Statistical Learning Theory basement that results in employment of Structural Risk Minimization (SRM) principle and Vapnik-Chervonenkis complexity measure [2]. Since SVMs are kernel methods, they represent input/output relation in form of linear combination of basis functions (*kernels*). This Section briefly introduces SVM regression approach.

Let us consider the inverse scattering problem stated in Section 2. Since SVM regression implies scalar outputs, the problem has been decomposed on recovering horison and depth coordinates respectively. This means reformulation of (4) in terms of two training sets

$$\begin{array}{l}\Gamma^{\upsilon}_{train} = \left\{(\boldsymbol{\chi}_i, \upsilon_i),\ i = 1, \ldots, l\right\},\\ \upsilon \in \{\upsilon^x,\ \upsilon^y\}\end{array} \tag{5}$$

and training two independent SVMs.

Let us suppose to have a non-linear transformation $\mathbf{\Phi} : \mathbb{R}^N \to \mathbb{F}$, mapping the input measures $\boldsymbol{\chi}$ into a new high-dimensional space $\mathbb{F}$. Such a transformation is necessary in order to better interpolate all the input samples characterized by a strong non-linearity. Consequently, the estimating function is

$$\hat{\upsilon} = \mathbf{w} \cdot \mathbf{\Phi}(\boldsymbol{\chi}) + b\,. \tag{6}$$

In order to fit this model to the available $\Gamma^{\upsilon}_{train}$ (that is, to define optimal values of $\mathbf{w}$ and b, which we denote by $\mathbf{w}^{opt}$ and b^{opt}), a *loss function* $LF(\upsilon,\ \hat{\upsilon})$ has to be introduced. This function evaluates discrepancy between original and predicted outputs. According to the Vapnik's theory a very robust loss function is the so-called ε-insensitive loss function, which is defined as follows [2]:

$$\left|\upsilon - \hat{\upsilon}\left(\boldsymbol{\chi}\right)\right|_{\varepsilon} = \max\left\{0,\ \left|\upsilon - \hat{\upsilon}\left(\boldsymbol{\chi}\right)\right| - \varepsilon\right\}. \tag{7}$$

The simplest and the most intuitive way of fitting model is based on *Empirical Risk Minimization* principle (ERM), that is

$$\left(\mathbf{w}^{opt}, b^{opt}\right) = \arg\min_{\mathbf{w},b} \sum_{i=1}^{l} \left|\upsilon_i - \hat{\upsilon}_i\left(\boldsymbol{\chi}\right)\right|_{\varepsilon}\ . \tag{8}$$

However, this principle does not take into consideration model's complexity, which has straightforward relation to model's generalization capacity [2]. Thus, SVM approach consists in minimizing the following functional:

$$\left(\mathbf{w}^{opt}, b^{opt}\right) = \arg\min_{\mathbf{w},b} \left[\frac{1}{2}\left\|\mathbf{w}\right\|^2 + C\sum_{i=1}^{l} \left|\upsilon_i - \hat{\upsilon}_i\left(\boldsymbol{\chi}\right)\right|_{\varepsilon}\right]\ , \tag{9}$$

which takes into consideration both empirical risk and the model's complexity. In other words, this functional minimizes a trade-off (tuned by the parameter C) between the model's complexity and the error on $\Gamma^{\upsilon}_{train}$.

The expression (9) can be rewritten as follows:

$$
\begin{array}{ll}
(\mathbf{w}^{opt}, b^{opt}) = \arg \min\limits_{\mathbf{w},b,\xi_i,\xi_i^*} & \left[\frac{1}{2} \|\mathbf{w}\|^2 + C \sum\limits_{i=1}^{l} (\xi_i + \xi_i^*) \right] \\
\text{subject to} & \left\{ \begin{array}{rcl} \upsilon_i - \mathbf{w} \cdot \mathbf{\Phi}(\boldsymbol{\chi}_i) - b & \leq & \varepsilon + \xi_i \\ \mathbf{w} \cdot \mathbf{\Phi}(\boldsymbol{\chi}_i) + b - \upsilon_i & \leq & \varepsilon + \xi_i^* \\ \xi_i,\ \xi_i^* & \geq & 0\,. \end{array} \right.
\end{array}
\tag{10}
$$

The above CQP, also called the *Primal*, is usually solved by using the Lagrange multipliers theory [6] in order to obtain the corresponding *Dual*:

$$
\begin{array}{ll}
\max\limits_{\alpha_i,\alpha_i^*} & -\frac{1}{2} \sum\limits_{i,j=1}^{l} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \mathbf{\Phi}(\boldsymbol{\chi}_i) \cdot \mathbf{\Phi}(\boldsymbol{\chi}_j) - \varepsilon \sum\limits_{i=1}^{l} (\alpha_i + \alpha_i^*) + \sum\limits_{i=1}^{l} \upsilon_i (\alpha_i - \alpha_i^*) \\
\text{subject to} & \left\{ \begin{array}{l} 0 \leq \alpha_i, \alpha_i^* \leq C \\ \sum\limits_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0\,. \end{array} \right.
\end{array}
\tag{11}
$$

Each of *dual variables* α_i, α_i^* is a Lagrange multiplier associated with the corresponding constraint.

Then $\mathbf{w}^{opt}$ is calculated as a linear combination of transformed input vectors from $\Gamma_{train}^{\upsilon}$:

$$
\mathbf{w}^{opt} = \sum_{i=1}^{l} (\alpha_i^{opt} - \alpha_i^{*\,opt}) \mathbf{\Phi}(\boldsymbol{\chi}_i)\,,
\tag{12}
$$

α_i^{opt} and $\alpha_i^{*\,opt}$ being the optimal α_i and α_i^* for (11). Thus, the dual formulation allows to write $\hat{\upsilon}$ in terms of dual variables:

$$
\hat{\upsilon}(\boldsymbol{\chi}) = \sum_{i=1}^{l} (\alpha_i^{opt} - \alpha_i^{*\,opt}) \mathbf{\Phi}(\boldsymbol{\chi}_i) \cdot \mathbf{\Phi}(\boldsymbol{\chi}) + b^{opt}\,.
\tag{13}
$$

Furthermore, the dual allows us to use the kernel trick as well. To this end, the kernel function is introduced:

$$
k(\boldsymbol{\chi}_i, \boldsymbol{\chi}_j) = \mathbf{\Phi}(\boldsymbol{\chi}_i) \cdot \mathbf{\Phi}(\boldsymbol{\chi}_j),
\tag{14}
$$

which allows to avoid the explicit handling $\mathbf{\Phi}$. The theory of kernels, that is, the conditions under which equation (14) holds, is known since the beginning of the last century thanks to the Mercer's theorem [2], and has been applied to pattern recognition tasks since the '60s [7], but only recently its connection with learning machines has been well formalized [5]. Since the seminal works on kernel functions, many kernels have been found which satisfy the Mercer's theorem; we recall the linear, the Gaussian and the polynomial kernels:

$$
\begin{array}{rcl}
k(\boldsymbol{\chi}_i, \boldsymbol{\chi}_j) & = & \boldsymbol{\chi}_i \cdot \boldsymbol{\chi}_j \\
k(\boldsymbol{\chi}_i, \boldsymbol{\chi}_j) & = & \exp\left(-\gamma \|\boldsymbol{\chi}_i - \boldsymbol{\chi}_j\|\right) \\
k(\boldsymbol{\chi}_i, \boldsymbol{\chi}_j) & = & \left(\delta + \gamma \cdot \boldsymbol{\chi}_i \cdot \boldsymbol{\chi}_j\right)^p\,.
\end{array}
\tag{15}
$$

As far as the kernel satisfies the Mercer's theorem, the CQP (11) can be efficiently solved [8, 9].

As a final remark, let us note that the parameter b can be computed by exploiting the *Karush–Khun–Tucker* (KKT) conditions [5]. In particular according to KKT, at the solution point the product between dual variables and constraints must vanish:

$$\begin{array}{rcl} \alpha_i^{opt}\left(\varepsilon + \xi_i - \upsilon_i + \mathbf{w}^{opt} \cdot \mathbf{\Phi}(\boldsymbol{\chi}_i) + b^{opt}\right) & = & 0 \\ \alpha_i^{*\,opt}\left(\varepsilon + \xi_i^* + \upsilon_i - \mathbf{w}^{opt} \cdot \mathbf{\Phi}(\boldsymbol{\chi}_i) - b^{opt}\right) & = & 0\,. \end{array} \tag{16}$$

This allows one to write (see [5] for details):

$$\begin{array}{l} b^{opt} = \upsilon_i - \mathbf{w}^{opt} \cdot \mathbf{\Phi}(\boldsymbol{\chi}_i) - \varepsilon, \quad \alpha_i^{opt} \in (0, C) \\ b^{opt} = \upsilon_i - \mathbf{w}^{opt} \cdot \mathbf{\Phi}(\boldsymbol{\chi}_i) + \varepsilon, \quad \alpha_i^{*\,opt} \in (0, C)\,. \end{array} \tag{17}$$

The approach described so far, the ε-based SVM for regression (ε-SVMR), is the algorithm to be used when the desired accuracy of our estimation is known a priori. However, in the case of the inverse scattering problem discussed here one needs the estimate to be as accurate as possible, without having to fix a priori a given level of accuracy. To this aim, one could use a modification of ε-SVMR, called ν-based SVM for regression (ν–SVMR) [10, 5]. The main concept of ν-SVMR can be summarized in the following way: for each $\boldsymbol{\chi}_i$ we accept an error ε; all errors above ε are stored in slack variables ξ_i or ξ_i^*, which are inserted in the global cost function and penalized by the constant C; the value of ε, in its turn, is traded-off against the model's complexity and slack variables by a parameter $\nu \geq 0$. The final Primal CQP problem is

$$\begin{array}{ll} (\mathbf{w}^{opt}, b^{opt}) = \arg \min\limits_{\mathbf{w}, b, \xi_i, \xi_i^*, \varepsilon} & \left[\frac{1}{2}\|\mathbf{w}\|^2 + C\left(l\nu\varepsilon + \sum\limits_{i=1}^{l}(\xi_i + \xi_i^*)\right)\right] \\ \text{subject to} & \left\{ \begin{array}{rcl} \upsilon_i - \mathbf{w} \cdot \mathbf{\Phi}(\boldsymbol{\chi}_i) - b & \leq & \varepsilon + \xi_i \\ \mathbf{w} \cdot \mathbf{\Phi}(\boldsymbol{\chi}_i) + b - \upsilon_i & \leq & \varepsilon + \xi_i^* \\ \xi_i,\ \xi_i^* & \geq & 0 \\ \varepsilon & \geq & 0\,. \end{array} \right. \end{array} \tag{18}$$

After several mathematical steps, the following dual CQP is obtained:

$$\begin{array}{ll} \max\limits_{\alpha_i, \alpha_i^*} & -\frac{1}{2}\sum\limits_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(\boldsymbol{\chi}_i, \boldsymbol{\chi}_j) + \sum\limits_{i=1}^{l}\upsilon_i(\alpha_i - \alpha_i^*) \\ \text{subject to} & \left\{ \begin{array}{l} 0 \leq \alpha_i, \alpha_i^* \leq C \\ \sum\limits_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \\ \sum\limits_{i=1}^{l}(\alpha_i + \alpha_i^*) \leq Cl\nu\,. \end{array} \right. \end{array} \tag{19}$$

Thanks to the presence of ν, ν-SVMR automatically computes ε. It has been shown that ν has several important properties; among others, the most important is that $\nu \in [0, 1]$ is an upper bound on the fraction of training points lying outside the ε-tube.

In this work LIBSVM software [11] has been applied to implement SVM technique. ν-SVM regression based on Gaussian and polynomial kernel functions (15) have been considered.

4 Parameters, hypreparameters, and model selection

One should notice that the procedure described in Section 3 finds optimal decision function (6) while values of the parameters γ, δ, p, C, ν are supposed to be already predefined (we will refer to these parameters as to *hyperparameters*). Consequently, the additional problem arises: finding such values of hyperparameters, which would afford SVM with possibly lower *generalization error (actual risk)* [2]. To this end one tries to minimize the function

$$G(h_1, h_2, \ldots, h_n) \tag{20}$$

that evaluates the generalization error of SVM with hyperparameters $(h_1, h_2, \ldots h_n)$, n being the number of the hyperparameters.

There is a number of approaches used for evaluating generalization error [12, 13]. The one used in this work is so-called *validation set* approach: the generalization error for the SVM that corresponds to the hyperparameters $(h_1, h_2, \ldots h_n)$ is evaluated by means of *mean square error* (MSE)

$$MSE(h_1, h_2, \ldots, h_n) \tag{21}$$

reached on the *validation set* Γ^{υ}_{val}. The structure of this set is the same as the structure of $\Gamma^{\upsilon}_{train}$ (5). It consists of input-output pairs, where input vector consists of N scalars describing the electromagnetic field, and output is horizon or depth scatterer's coordinate:

$$\begin{array}{l} \Gamma^{\upsilon}_{val} = \left\{ (\boldsymbol{\chi}_i, \upsilon_i),\ i = 1, \ldots, l^{val} \right\}, \\ \upsilon \in \{\upsilon^x,\ \upsilon^y\}. \end{array} \tag{22}$$

However, calculating (21) in some point $(h_1, h_2, \ldots, h_n)$ of hyperparameters' space means SVM training (i.e. solving CQP) and testing (on validation set), i.e. has high computational cost. Nevertheless, one can define a reasonable set of values

$$SET_i, \qquad i = 1, 2, \ldots, n \tag{23}$$

for every hyperparameter and minimize (21) on cartesian product of these sets. The values of hyperparameters obtained in such a way will be referred to as *suboptimal values of hyperparameters.*

The case of polynomial kernel is complicated by the fact that $n = 5$ (γ, δ, p, C, ν) instead of $n = 3$ (γ, C, ν) for the case of Gaussian kernel (polynomial kernel has more "degrees of freedom").

Sets of hyperparameter values used in case of Gaussian kernel and the obtained suboptimal values of hyperparameters are cited in Table 4 and Table 5 respectively.

The following approach has been considered to deal with polynomial kernel. At the beginning the sets (23) for ν and C have been chosen equal to ones for the case of Gaussian kernel SVM (see Table 4), the set for p has been chosen as follows: $SET_p = \{1, 2, 3, 4, 5\}$, γ and δ have been fixed by default LIBSVM values: $\gamma = 1/N$, $\delta = 0$. On the basis of obtained MSE values graphs

$$MSE(h_i^*) = MSE(h_1^{*sub}, \ldots, h_{i-1}^{*sub}, h_i^*, h_{i+1}^{*sub}, \ldots, h_m^{*sub}), i = 1, \ldots, m \tag{24}$$

have been plotted for both horizon and depth recovery cases. Here

- m is the number of hyperparameters that are not fixed;

- $h_i^* \in SET_i$;

- h_i^{*sub} is the found subopotimal value of the hyperparameter h_i.

These graphs reflect the behavior of (21); in fact they are formed by the points that belong to cross-sections of surface defined by (21). This gives the opportunity to correct SET_i: to remove some elements from a region of h_i where MSE is large or on the contrary to add some new elements from a region of h_i where MSE has appeared to be small.

Such a correction allows to restart searching suboptimal values of hyperparameters on the corrected sets (23). Thus, searching suboptimal values of hyperparameters can be repeated iteratively.

5 Simulation

This Section describes the simulation steps (Sections 5.2), presents the obtained results (Sections 5.3 and 5.4). Section 5.5 is devoted to discussion of the results.

The following workstations have been used for the simulation:

- **Workstation 1**
  - 733 MHz Intel Pentium III CPU
  - 128 MBytes RAM
  - Linux Mandrake 7.1 operation system
- **Workstation 2**
  - 1700 MHz Intel Pentium IV CPU
  - 256 MBytes RAM
  - Linux Mandrake 7.1 operation system

5.1 Datasets

So far, we have already introduced training set Γ^v_{train} (5) and validation set Γ^v_{val} (22). The third set used in simulation is *test set* Γ^v_{test}. Its aim is to provide the data for calculating the prediction error of the model found by model selection procedure (Section 4). Γ^v_{test} has the same structure as Γ^v_{train} and Γ^v_{val}:

$$\begin{array}{l} \Gamma^v_{test} = \{(\boldsymbol{\chi}_i, v_i),\ i = 1, \ldots, l^{test}\}, \\ v \in \{v^x,\ v^y\}. \end{array} \tag{25}$$

Noise distortion of the scattered signals received by antennas has been considered and modeled as well (additive Gaussian noise). Thus, *ultima analysi* for each of horizon and depth recovery problems 7 triplets of datasets have been generated:

$$\begin{array}{lcl} (\Gamma^{v,SNR}_{train}, & \Gamma^{v,SNR}_{val}, & \Gamma^{v,SNR}_{test}) \\ v & \in & \{v^x,\ v^y\} \\ SNR & \in & \{5 \text{ dB}, 10 \text{ dB}, 20 \text{ dB}, 35 \text{ dB}, 50 \text{ dB}, 100 \text{ dB}, \text{noiseless}\}. \end{array} \tag{26}$$

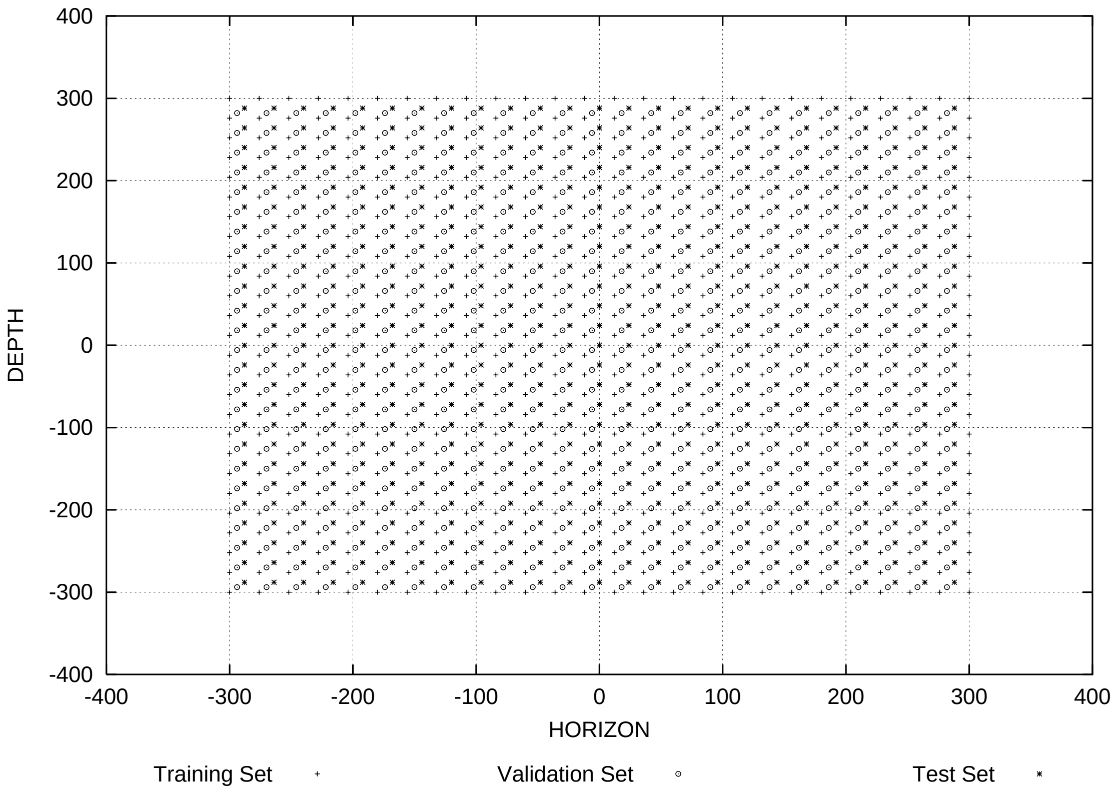


Figure 2: Training, validation and test sets' domains

To this end, Finite Element code and a PML technique have been applied for the problem stated in Section 2.

The cylinder's center positions used to form $\Gamma_{train}^{v,SNR}$, $\Gamma_{val}^{v,SNR}$, and $\Gamma_{test}^{v,SNR}$ are indicated in Figure 2. $\Gamma_{train}^{v,SNR}$ consists of $l = 676$ input-output pairs, whereas each of $\Gamma_{val}^{v,SNR}$ and $\Gamma_{test}^{v,SNR}$ consists of $l^{val} = l^{test} = 625$ pairs.

5.2 Simulation steps

In general the simulation can be represented as the consequence of the following phases:

1. Normalization phase
2. Phase of searching suboptimal values for hyperparameters
   (a) assignment of values for hyperparameters
   (b) SVM training phase
   (c) model selection phase
3. SVM test phase
4. Denormalization phase
5. Errors calculation phase

Below each phase is described in more details.

1. Normalization:

```
for $SNR \in \{5$ dB ,10 dB, 20 dB, 35 dB, 50 dB, 100 dB, noiseless$\}$
  for $\upsilon \in \{\upsilon^x,\ \upsilon^y\}$
    Output values $\upsilon$ and input vectors' components $(\boldsymbol{\chi}_i)_j$ of $\Gamma_{train}^{\upsilon,SNR}$ are linearly transformed
      $\begin{array}{l} \upsilon'_i = a^{\upsilon,SNR} \cdot \upsilon_i + b^{\upsilon,SNR} \\ (\boldsymbol{\chi}'_i)_j = c^{SNR} \cdot (\boldsymbol{\chi}_i)_j + d^{SNR} \\ i = 1, \dots, l \\ j = 1, \dots, N \end{array} \qquad (27)$
    in such a way that
      $\begin{array}{l} \min_i(\upsilon'_i) = \min_{i,j}(\boldsymbol{\chi}'_i)_j = -1 \\ \max_i(\upsilon'_i) = \max_{i,j}(\boldsymbol{\chi}'_i)_j = 1. \end{array} \qquad (28)$
    Then obtained normalization coefficients $a^{\upsilon,SNR}$, $b^{\upsilon,SNR}$, $c^{SNR}$, $d^{SNR}$ are used for the linear normalization of $\Gamma_{val}^{\upsilon,SNR}$ and $\Gamma_{test}^{\upsilon,SNR}$. This phase is useful for numerical reasons and strongly recommended [14].
  endfor
endfor
```

2. Searching suboptimal values for hyperparameters:

(2a) assignment of values for hyperparameters: Set of values (23) for any SVM hyperparameter is assigned. Cartesian product of these sets $SET_1 \times \cdots \times SET_n$ defines the set of SVMs that are considered on phases (2b) and (2c).

(2b) SVM training phase:

```
for $SNR \in \{5$ dB ,10 dB, 20 dB, 35 dB, 50 dB, 100 dB, noiseless$\}$
  for $\upsilon \in \{\upsilon^x,\ \upsilon^y\}$
    Set of SVMs with hyperparameters defined on phase (2a) is trained on normalized $\Gamma_{train}^{\upsilon,SNR}$.
  endfor
endfor
```

(2c) model selection phase:

```
for $SNR \in \{5$ dB ,10 dB, 20 dB, 35 dB, 50 dB, 100 dB, noiseless$\}$
  for $\upsilon \in \{\upsilon^x,\ \upsilon^y\}$
    Set of SVMs trained on normalized $\Gamma_{train}^{\upsilon,SNR}$ on phase (2b) is tested on normalized $\Gamma_{val}^{\upsilon,SNR}$. The SVM that has delivered the lowest value of mean square error (MSE) is selected. Values of hyperparameters of this SVM represent suboptimal values of hyperparameters for given $SNR$ and $\upsilon$ values.
  endfor
endfor
```

3. SVM test phase:

```
for $SNR \in \{5$ dB ,10 dB, 20 dB, 35 dB, 50 dB, 100 dB, noiseless$\}$
    for $\upsilon \in \{\upsilon^x,\ \upsilon^y\}$
        SVM with suboptimal values of hyperparameters is tested on normalized $\Gamma_{test}^{\upsilon,SNR}$.
    endfor
endfor
```

4. Denormalization phase:

```
for $SNR \in \{5$ dB ,10 dB, 20 dB, 35 dB, 50 dB, 100 dB, noiseless$\}$
    for $\upsilon \in \{\upsilon^x,\ \upsilon^y\}$
        Coordinate values predicted on the previews phase are denormalized using normalization coefficients $a^{\upsilon,SNR}$, $b^{\upsilon,SNR}$ obtained on phase (1).
    endfor
endfor
```

5. Calculation of errors:

```
for $SNR \in \{5$ dB ,10 dB, 20 dB, 35 dB, 50 dB, 100 dB, noiseless$\}$
    Real and predicted coordinate values are used for calculating *local average error* according to the next definition [3]:
```

$$
\begin{array}{ll}
\zeta_x^u = \frac{\left|x_{act}^u - \frac{1}{V(u)}\sum_{v(u)=1}^{V(u)} x_{rec}^{v(u)}\right|}{d_{max}} & u = 1,\ldots,U \\
\zeta_y^v = \frac{\left|y_{act}^v - \frac{1}{U(v)}\sum_{u(v)=1}^{U(v)} y_{rec}^{u(v)}\right|}{d_{max}} & v = 1,\ldots,V
\end{array}
\qquad (29)
$$

```
    Here $u$ and $v$ define respectively horizontal and vertical position on the grid formed by the cylinder's center coordinates of the test set (Figure 2). $v(u)$ represents possible vertical positions for horizontal position defined by $u$; $V(u)$ is the number of such positions. Similarly $u(v)$ represents possible horizontal positions for vertical position defined by $v$; $U(v)$ is the number of such positions. For the given test set $V(u)$ and $U(v)$ are constant values equal to 25. $x_{act}^u$ and $y_{act}^v$ are actual values of horizon and depth coordinates for the position on the grid defined by $u$ and $v$ respectively. $x_{rec}^{v(u)}$ is the recovered horizon value for the position on the grid defined by $(u, v(u))$. Similarly, $y_{rec}^{u(v)}$ is the recovered depth value for the position on the grid defined by $(u(v), v)$. $d_{max} = L_S$, see Section 2.
endfor
```

5.3 Polynomial kernel, γ and δ fixed by default values

For this simulation the number of corrections of (23) (see Section 4) has been limited by 2. The obtained sets (which give rise to 1200 different combinations) are collected in Table 6. Graphs defined by (24) are presented on Figure 3. Obtained suboptimal values are cited in Table 7. Finally Figure 4 and Figure 5 demonstrate the obtained local average error (29) for horizon and depth recovery problems.

The simulation has been performed on Workstation 1 and has taken approximately 5 days without consideration of time spent to form sets mentioned in Table 6. The major part of time (approximately 4 days) fell to SVM training phase. In its turn, majority of training phase time fell to SNR = 5 dB (approximately 2 days). It has been also noticed that within the bounds of every particular SNR value the main time expenses fell to training SVMs with large values of C ($C = 10^5$, $C = 10^6$). Table 2 and Table 3 describe dependence of training and test phase times on values of SNR and C. $C = 10^5$, $C = 10^6$ are not cited in Table 3 because of requiring relatively great time expenses.

SNR (dB)	training time (sec)	test time (sec)
5	5.89	1.68
10	3.25	1.74
20	3.26	1.81
35	2.69	1.82

Table 2: Dependence of (horizon+depth) training and test phase times on SNR; $p = 3$, $\nu = 0.6$, $C = 1000$

C	training time (sec)	test time (sec)
10^{-1}	2.24	1.65
1	2.37	1.79
10^2	2.67	1.59
10^3	5.89	1.68
10^4	29.43	1.61

Table 3: Dependence of (horizon+depth) training and test phase times on C; $p = 3$, $\nu = 0.6$, SNR = 5 dB

5.4 Polynomial kernel, various values of γ

This simulation concerns removing limitation on fixing γ. To approximately estimate amount of time needed for its execution, two *horizon+depth* SVM pairs ($p = 5$, $\nu = 0.7$, $C = 10^6$, $\delta = default$) have been trained on $\Gamma_{train}^{v^x,\,100\ dB}$ and $\Gamma_{train}^{v^y,\,100\ dB}$: with default γ (1/32) and $\gamma = 0.9$. Training has taken 17 seconds and 17 minutes respectively. Several other values of γ have just reconfirmed the trend: the greater the γ the more time expensive training phase. It has been decided to use three different γ values: 0.005 (less than default γ), 0.05 and 0.1 (greater than default γ). Sets of values for p, ν and C have been formed on the basis of the values noted in Table 7 with some changes. These sets (which give rise to 288 different combinations) and suboptimal values are collected in Table 8 and Table 9 respectively. $C = 10^6$ has not been considered because of high computational expenses.

Execution of this simulation has been started on Workstation 1. In 5 days training phase has been in progress. Therefore it has been decided to use additionally Workstation 2. In 2 days training phase has been finished (thus, Workstation 1 and Workstation 2 have been working for 7 and 2 days respectively). All the consequent simulation phases (model selection, testing, etc.) have been completed on Workstation 2 and have taken less than 1 day.

Thus given simulation has consumed more time than the previous one despite the fact of decreasing the number of combinations by approximately 4 times (288 instead 1200). Figure 6 and Figure 7 demonstrate obtained local average error (29) for horizon and depth recovery problems.

5.5 Discussions

Training phases for simulations described in Sections 5.3 and 5.4 have taken more than 4 and 7 days respectively. Therefore, the considered approach can not be recommended for problems where learning machine is supposed to be retrained often. On the other hand, test phase takes less than 2 seconds for $l^{test} = 625$ samples, that is less than $3.2 \cdot 10^{-3}$ seconds per sample. This means possibility of scatterer detection on the run.

Figure 3 demonstrates that generalization performance of SVM in case of horizon recovery does not significantly depend on degree of polynomial kernel starting from $p = 2$ (Fig. 3(a)). The same relates to dependence on ν except SNR=5 dB (Fig. 3(e)), however slight trend to decreasing MSE when ν increases up to 0.5 can be traced for high SNR values. For depth recovery (Fig. 3(c) and 3(f)) trends are more traceable. Namely, optimal degree values are almost always 2, 3 or 4. Then MSE significantly decreases for high SNR values when ν increases up to 0.5.

Dependence on C is the same for both depth and horizon recovery (except 5 dB, 10 dB and 20 dB horizon recovery): MSE decreases when C increases. Probable explanation of this fact is high similarity of Γ^v_{train} and Γ^v_{val} (Figure 2). Thus, for every point from Γ^v_{val} close point from Γ^v_{train} exists. This means that as long as SNR value is small, SVM during model selection phase is tested on almost the same data that has been used for training. Consequently, it seems difficult to evaluate generalization performance in this situation. This remains solving the ERM problem (8) obtained from (9) by assuming exactly $C = \infty$. As a conclusion, redefining of datasets in more disorderly way is expected to be reasonable.

According to Figures 4, 5, 6 and 7 polynomial kernels have demonstrated better performance. However it is worth to notice that in case of Gaussian kernels iterative procedure of searching suboptimal values for hyperparameters has not been performed. This fact hamper in rigorous comparison of two kernels performances. Nevertheless, for low SNR values results on Figure 4 and Figure 5 are quite similar for both kernels.

Results obtained in Section 5.4 generally exceed the ones for Section 5.3.

The given work represents the starting point in the sense that only one transmitter have been used. The obtained results distinctly demonstrate the significant increase of horizon's local average error as the horizontal distance from the transmitter increases. Thus, the main direction the future work will press towards is considering the polynomial (and possibly some other) kernel performance for the model with multiple transmitters [15]. Let us note two alternative approaches: 1) multiple transmitters operate in sequence, the the feature vectors' dimensionality is $N = 2 \cdot TR \cdot RS$ (see Section 2); 2) multiple transmitters operate simultaneously, thus the the feature vectors' dimensionality is still $N = 2 \cdot RS$.

6 Conclusions

In this paper buried object detection problem has been reformulated as regression estimation and solved by means of *learning by examples* methodology, namely by means of ν-SVM regression technique. Simulation has been carried out on synthetic data generated by Finite Element code and a PML technique; noisy environments have been considered as well. Two different types of kernel functions have been considered. Though time required to SVM training can be tremendous, test phase takes less than 2 seconds for $l^{test} = 625$ samples, that is less than $3.2 \cdot 10^{-3}$ seconds per sample. This implies possibility of scatterer detection on the run.

It has been shown that using polynomial kernel along with iterative model selection phase allows to outperform Gaussian kernel.

The main direction of the future work will press towards considering the polynomial (and possibly some other) kernel performance for the model with multiple transmitters [15]. One of other directions is model selection techniques. So far, validation set method have been used for model selection, which is reasonable when one is in data-reach situation (like synthetic data). However, this method is unsuitable on real world domain, where only limited amount of data is available. Considering other model selection methods (e.g. *k-fold cross-validation* or *maximum discrepancy* criterion [13]), which are better meet the specificity of limited amount of data, will allow to more realistically evaluate the proposed approach.

References

[1] Bermani, E., Boni, A., Caorsi, S., Massa, A., "An Innovative Real-Time Technique for Buried Object Detection," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 41, No. 4, 927–931, 2003.

[2] Vapnik, V. N., *The Nature of Statistical Learning Theory*, Statistics for Engineering and Information Science, Springer Verlag, 2nd edition, 1999.

[3] Caorsi, S., Anguita, D., Bermani, E., Boni, A., Donelli, M., "A comparative study of nn and svm-based electromagnetic inverse scattering approaches to on-line detection of buried objects," *ACES journal*, Vol. 18, No. 2, 2003.

[4] Cristianini, N., Shawe–Taylor, J., *An Introduction to Support Vector Machines*, Cambridge University Press, 2000.

[5] Schölkopf, B., Smola, A. J., *Learning with Kernels*, MIT Press, Cambridge, MA, 2002.

[6] Bertsekas, D. P., *Constrained Optimization and Lagrange Multipliers*, Academic Press, New York, 1982.

[7] Aizerman, M. A., Braverman, E. M., Rozonoer, L. I., "Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning," *Automation and Remote Control*, Vol. 25, 821–837, 1964.

[8] Platt, J., "Fast Training of Support Vector Machines Using Sequential Minimal Optimization", in B. Scölkopf, C. Burges, A. Smola (Eds.), *Advances in Kernel Methods – Support Vector Learning*, MIT Press, 1999.

[9] Lin, C.-J., "Asymptotic Convergence of an SMO Algorithm Without Any Assumptions," *IEEE Trans. on Neural Networks*, Vol. 13, No. 1, 248–250, Jan. 2002.

[10] Smola, A., Schölkopf, B., Williamson, R., Bartlett, P., "New support vector algorithms," *Neural Computation*, Vol. 12, No. 5, 1207–1245, May 2000.

[11] Chang, C.-C., Lin, Ch.-J., *Libsvm: a Library for Support Vector Machines*, May 2003.

[12] Hastie, T., Tibshirani, R., Friedman, J., *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*, Springer, New York, 2001.

[13] Anguita, D., Ridella, S., Rivieccio, F., Zunino, R., "Hyperparameter design criteria for support vector classifiers," *Neurocomputing*, Vol. 55, 109–134, Sept. 2003.

[14] Hsu, Ch.-W., Chang, Ch.-Ch., Lin, Ch.-J., *A Practical Guide to Support Vector Classification*, Department of Computer Science and Information Engineering, National Taiwan University, July 2003.

[15] Bermani, E., Boni, A., Caorsi, S., Donelli, M., Massa, A., "A Multi-Source Strategy Based on a Learning-by-Examples Technique for Buried Object Detection," *PIER Jounal*, Vol. 48, 185–200, 2004.

HYPREPARAMETER	SET OF VALUES
ν	0.4 0.6 0.8
C	10^{-1} 1 10 10^2
γ	0.2 0.4 0.6 0.8 1

Table 4: Values of hyperparameters: Gaussian kernel

Horizon SVM			
SNR, dB	γ	ν	C
5	0.2	0.8	10^{-1}
10	0.8	0.4	10
20	0.2	0.8	10^{-1}
35	1	0.6	10^2
50	0.2	0.8	10^{-1}
100	1	0.6	10^2
noiseless	1	0.6	10^2

Depth SVM			
SNR, dB	γ	ν	C
5	0.2	0.4	10^{-1}
10	1	0.8	10^{-1}
20	1	0.8	10^2
35	1	0.8	10^2
50	1	0.8	10^2
100	1	0.8	10^2
noiseless	1	0.8	10^2

Table 5: Suboptimal values of hyperparameters: Gaussian kernel

HYPREPARAMETER	SET OF VALUES
p	1 2 3 4 5 6 7 8
ν	0.01 0.02 0.03 0.04 0.05 0.06 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
C	10^{-3} 10^{-2} 10^{-1} 1 10 10^2 10^3 10^4 10^5 10^6
δ	default (0)
γ	default ($1/k$)

Table 6: Values of hyperparameters: polynomial kernel, γ and δ are fixed

Horizon SVM				
SNR, dB	p	ν	C	MSE
5	4	0.5	10^{-1}	0.3252
10	4	0.03	10^3	0.2984
20	2	0.1	10^2	0.3319
35	4	0.6	10^6	0.2883
50	5	0.5	10^6	0.3059
100	5	0.7	10^6	0.2600
noiseless	5	0.6	10^6	0.2606
Depth SVM				
SNR, dB	p	ν	C	MSE
5	2	0.6	10^6	0.2942
10	2	0.6	10^6	0.2617
20	3	0.8	10^2	0.1887
35	4	0.7	10^6	0.1654
50	5	0.7	10^6	0.1609
100	4	0.6	10^6	0.1559
noiseless	4	0.6	10^6	0.1579

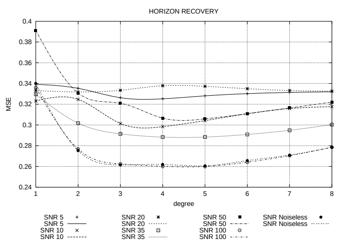
Table 7: Suboptimal values of hyperparameters: polynomial kernel, γ and δ are fixed

HYPREPARAMETER	SET OF VALUES
p	2 3 4 5
ν	0.01 0.03 0.1 0.3 0.6 0.7
C	10^{-1} 10^{2} 10^{3} 10^{5}
δ	default (0)
γ	0.005 0.05 0.1

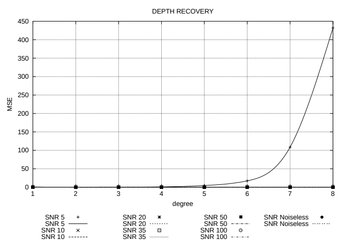
Table 8: Values of hyperparameters: polynomial kernel, δ is fixed

Horizon SVM					
SNR, dB	p	ν	C	γ	MSE
5	4	0.7	10^2	0.005	0.3257
10	3	0.03	10^2	0.1	0.2774
20	5	0.7	10^5	0.1	0.2973
35	4	0.6	10^5	0.1	0.2749
50	4	0.3	10^5	0.1	0.2412
100	4	0.3	10^5	0.1	0.2210
Depth SVM					
SNR, dB	p	ν	C	γ	MSE
5	3	0.6	10^5	0.1	0.2743
10	4	0.6	10^5	0.1	0.2524
20	4	0.3	10^5	0.1	0.1480
35	3	0.7	10^5	0.1	0.1462
50	4	0.7	10^5	0.1	0.1336
100	4	0.6	10^5	0.1	0.1357

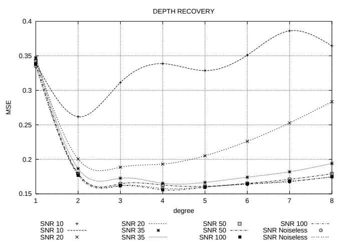
Table 9: Suboptimal values of hyperparameters:polynomial SVM, δ is fixed



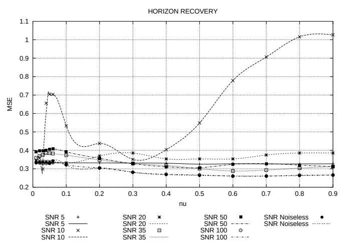
(a)



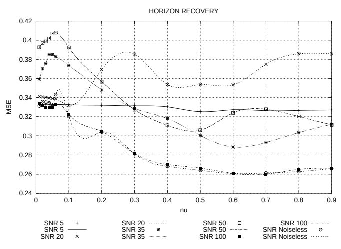
(b)



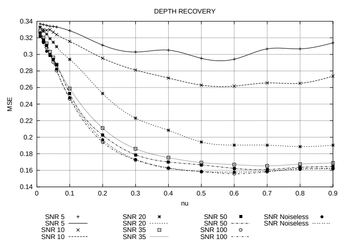
(c)



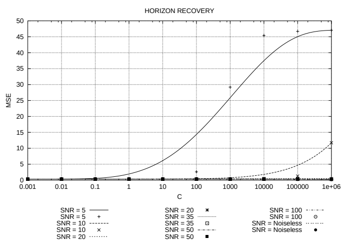
(d)



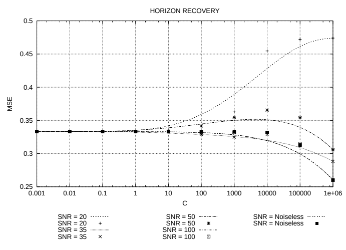
(e)



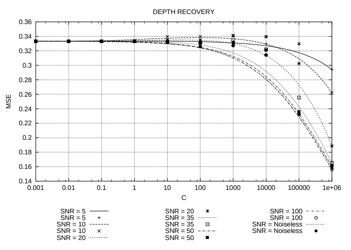
(f)



(g)

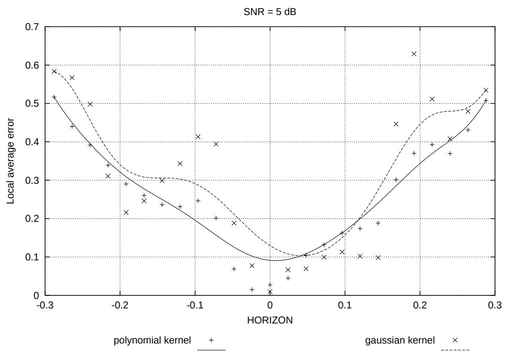


(h)

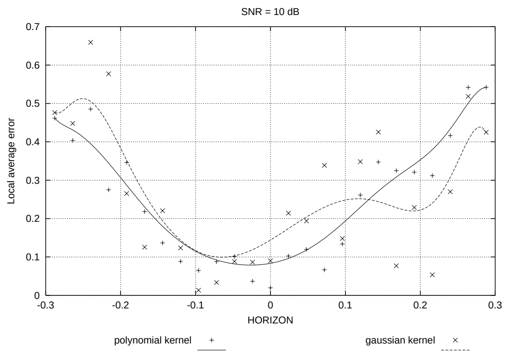


(i)

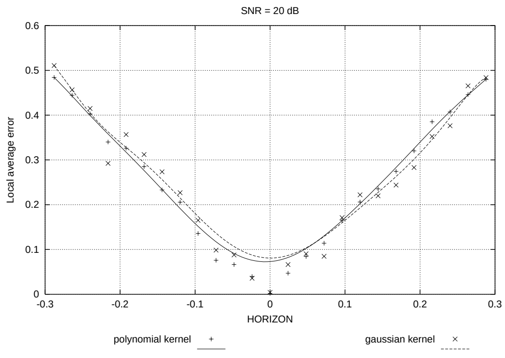
Figure 3: Mean Square Error on validation set: polynomial kernel, γ and δ are fixed



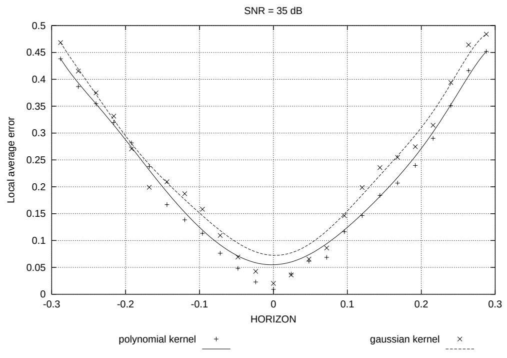
(a)



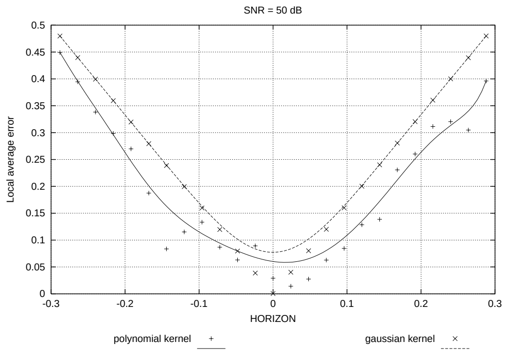
(b)



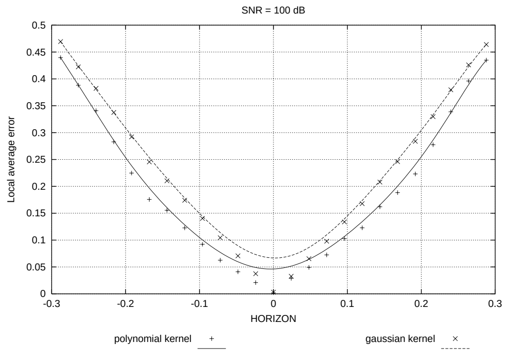
(c)



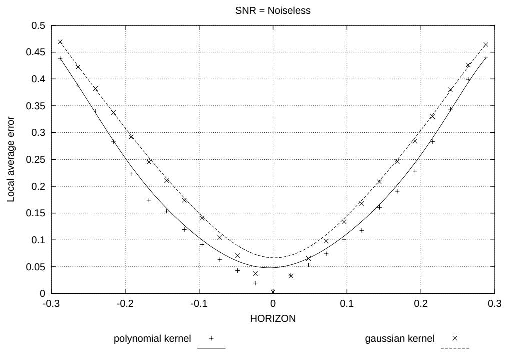
(d)



(e)

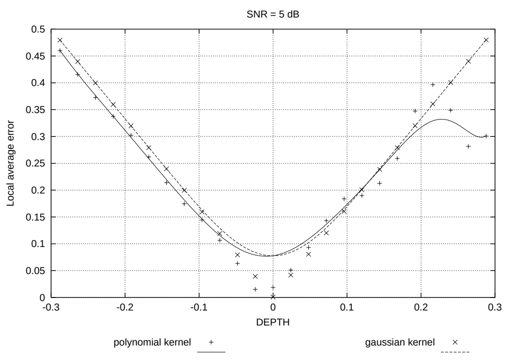


(f)

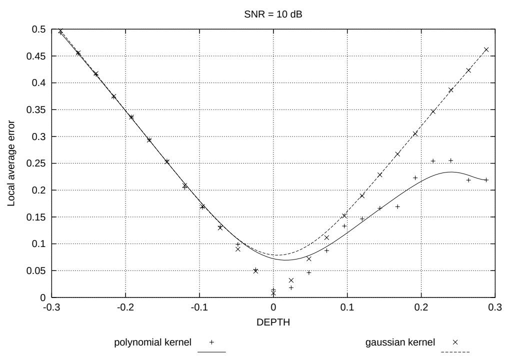


(g)

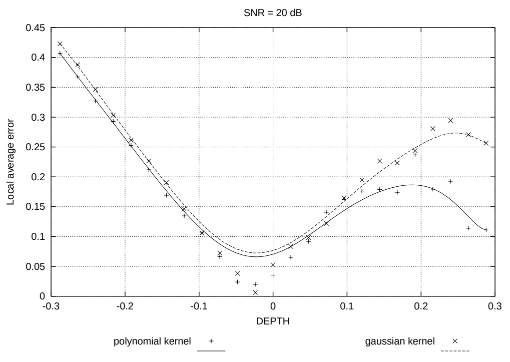
Figure 4: Local average error of horizon recovery: Gaussian kernel versus polynomial kernel (γ and δ are fixed)



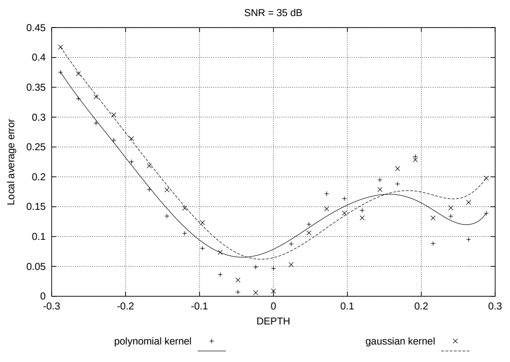
(a)



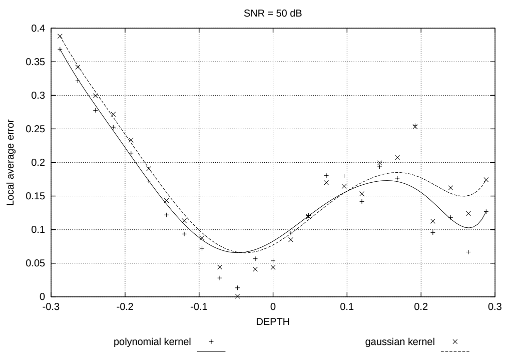
(b)



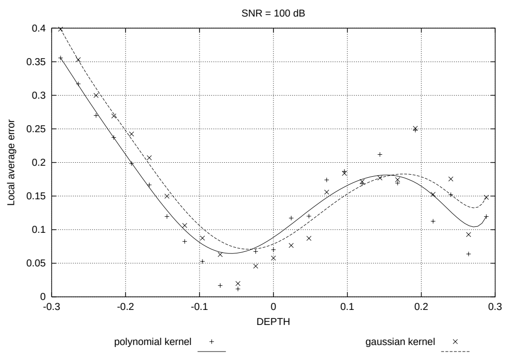
(c)



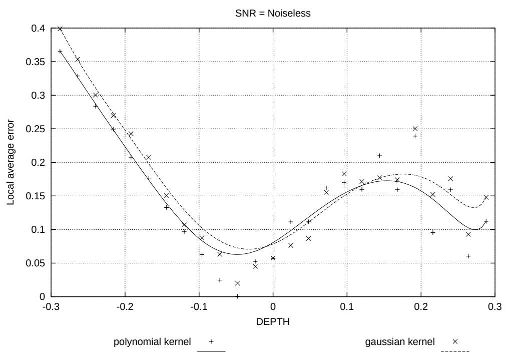
(d)



(e)

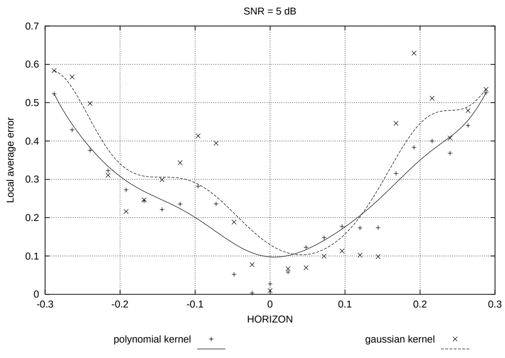


(f)

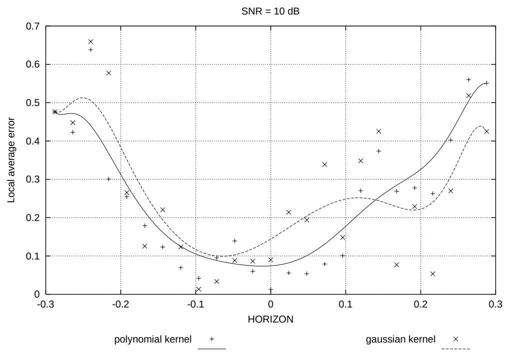


(g)

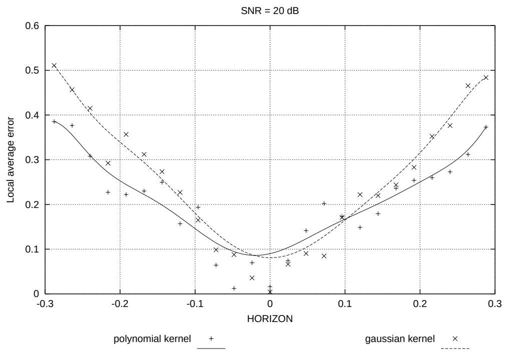
Figure 5: Local average error of depth recovery: Gaussian kernel versus polynomial kernel (γ and δ are fixed)



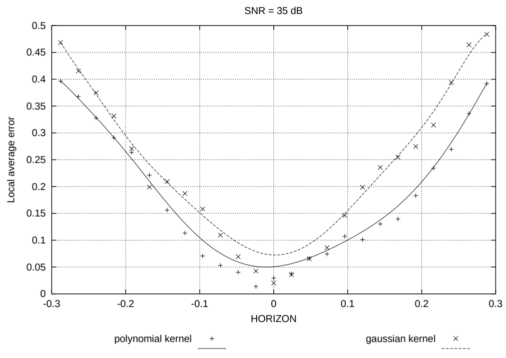
(a)



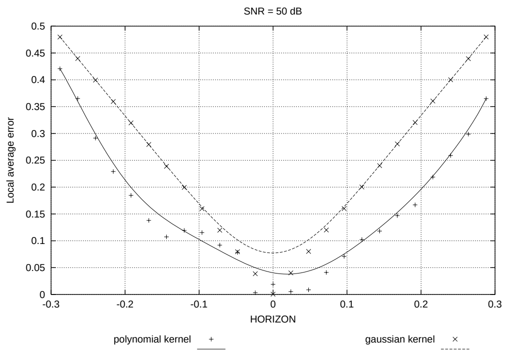
(b)



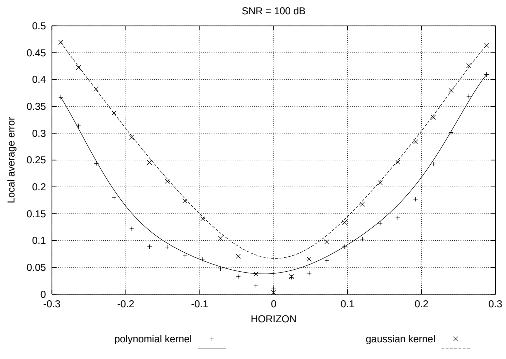
(c)



(d)

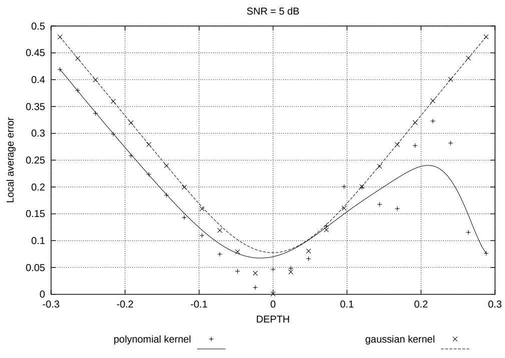


(e)

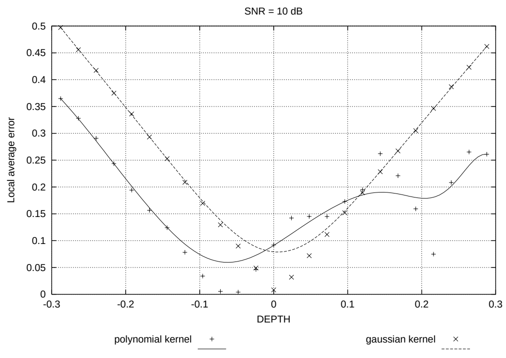


(f)

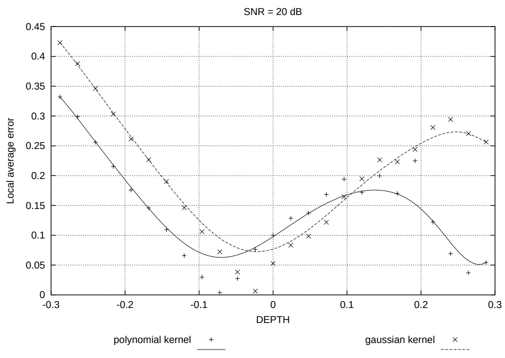
Figure 6: Local average error of horizon recovery: Gaussian kernel versus polynomial kernel (δ is fixed)



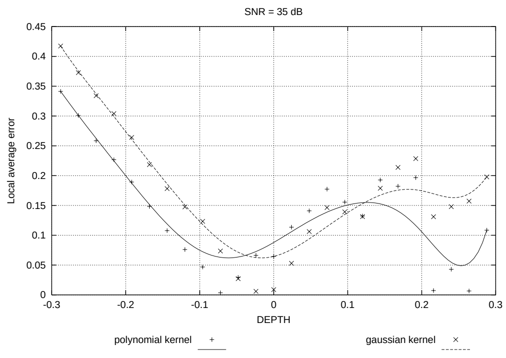
(a)



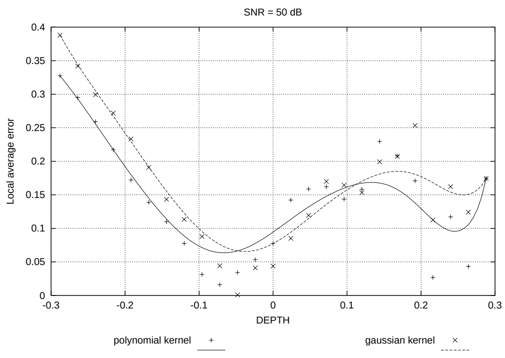
(b)



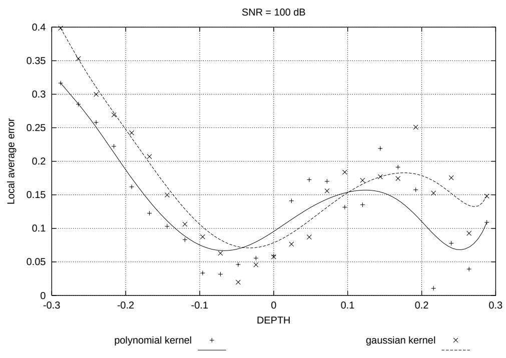
(c)



(d)



(e)



(f)

Figure 7: Local average error of depth recovery: Gaussian kernel versus polynomial kernel (δ is fixed)