SUPPORT VECTOR REGRESSION VIA *MATHEMATICA*

B. PALÁNCZ[1], L. VÖLGYESI[2,3] and Gy. POPPER[4]

[1] Department of Photogrammetry and Geoinformatics
[2] Department of Geodesy and Surveying
[3] Physical Geodesy and Geodynamic Research Group of the Hungarian Academy of Sciences
[4] Department of Structural Mechanics
Budapest University of Technology and Economics
H-1521 Budapest, Hungary
e-mail: palancz@epito.bme.hu

Abstract

In this tutorial type paper a *Mathematica* function for Support Vector Regression has been developed. Summarizing the main definitions and theorems of SVR, the detailed implementation steps of this function are presented and its application is illustrated by solving three 2D function approximation test problems, employing a stronger regularized universal Fourier and a wavelet kernel. In addition a real world regression problem, forecasting of the peak of flood-wave, is also solved. The numeric - symbolic results show how easily and effectively *Mathematica* can be used for solving SVR problems.

Keywords: Support Vector, Machine regression, software *Mathematica,* 2D function approximation

Introduction

In geoinformatics, it frequently happens that data collections are irregular and far from substantial for function approximation. This fact leads to overfitting, a poor data generalization, which means a good accuracy for training samples but bad fitting of the test data points [1]. To solve this problem one may use ridge regression or other regularization techniques modifying the object function in some ways [2, 3, 4]. However, in case of finding nonlinear mapping for input/output data of high dimensions, these methods usually result computational problem of NP complexity.

In the last few years, there have been very significant developments in the theoretical understanding of a relatively new family of algorithms; they present a series of useful features for classification as well as generalization of datasets [5]. Support Vector Machines (SVM) algorithms combine the simplicity and computational efficiency of linear algorithms, such as the perception algorithm or ridge regression, with the flexibility of nonlinear systems, like neural networks, and rigour of statistical approaches, as regularization methods in multivariate statistics [6]. As a result of the special way they represent functions, these algorithms typically reduce the learning step to a convex optimization problem that can always be solved in polynomial time, avoiding the problem of local minima typical of neural networks, decision trees and other nonlinear approaches [7].

Their foundation in the principles of statistical learning theory makes them remarkably resistant to overfitting especially in regimes where other methods are affected by the curse of dimensionality. It is for this reason that they have become popular in bioinformatics and text analysis. Another important feature for applications is that they can naturally accept input data that are not in the form of vectors, such as strings, trees and images [8, 9].

In this tutorial type article, it will be shown, that how easy to use this efficient technique for function approximation via *Mathematica*. The steps of implementation of support vector regression (SVR) are discussed and the application of this *Mathematica* function is illustrated by solving 2D approximation test problems with different type of kernels.

1. Kernels

1.1 Definition of kernel

Let *X* and *H* denote a linear vector space and a Hilbert space with scalar product $\langle , \rangle$. A continuous symmetric function $K : X \times X \to \mathbf{R}$ is said to be a kernel on *X* if there exists a map $\Phi : X \to H$ with

$$K(x,z) = \{\Phi(x), \Phi(z)\}$$

for all $x, z \in X$. We call Φ a feature map and *H* a feature space of *K*. For example a possible mapping is the following,

$$\{x_1, x_2\} \in \mathbf{R}^2 \to \Phi(x_1, x_2) = \{x_1^2, x_2^2, x_1 x_2\} \in H$$

Note that both *H* and Φ are far from being unique. However, for a given kernel there exists a unique Hilbert space of functions, named the *R*eproducing *K*ernel *H*ilbert *S*pace (*RKHS*).

1.2 Mercer's condition

Let *X* be a compact subset of $\mathbf{R}^n$. Suppose *K* is a continuous symmetric function for which

$$\int_{X \times X} K(x,z) f(x) f(z) dx dz$$

is positive, that is for all *f* in the space of the continuous functions on *X,* with infinite norm $\| \;\|_\infty$, namely $f \in C[X]$. Then for *K* exists, an *RKHS.*

1.3 Universal kernel

A function $f : X \to \mathbf{R}$ is induced by the kernel *K* if there exists an element $w \in H$ such that $f = \langle w, \Phi(.) \rangle$. For example,

$$f(x_1,x_2) = \langle \{w_1, w_2, w_3\}, \Phi(x_1,x_2) \rangle = \langle \{w_1, w_2, w_3\}, \{x_1^2, x_2^2, x_1 x_2\} \rangle = w_1 x_1^2 + w_2 x_2^2 + w_3 x_1 x_2$$

A continuous kernel is called universal if the set of all induced functions is dense in the space of continuous function on *X,* $C[X]$, which means that for all $g \in C[X]$ and $\in > 0$ there exists a function *f* induced by *K* with $\|f - g\|_\infty \leq \in$.

If K can be represented by a Taylor series, a necessary and sufficient condition for universality of K can be expressed with the help of the nonvanishing Taylor coefficients. A kernel is universal if and only if its *RKHS* is dense in $C\,[X]$ [10].

1.4 Some typical kernels

Here, we mention some typical kernels and display them in case of $X \subset \mathbf{R}^1$.

1.4.1 The Gaussian RBF universal kernel

The Gaussian Radial Basis Function universal kernel with $\beta > 0$ and all compact $X \subset \mathbf{R}^n$.

$$K(x,z) = \exp(-\beta(\|x-z\|)^2)$$

with $\beta = 0.1$

```
β= 0.1; K[x_,z_]:= Exp[-β(x-z)(x-z)];
Plot3D[K[x,z], {x,-10,10}, {z,-10,10},
  ViewPoint→{3.849, 6.909, 4.295}, PlotPoints→{30,30}];
```

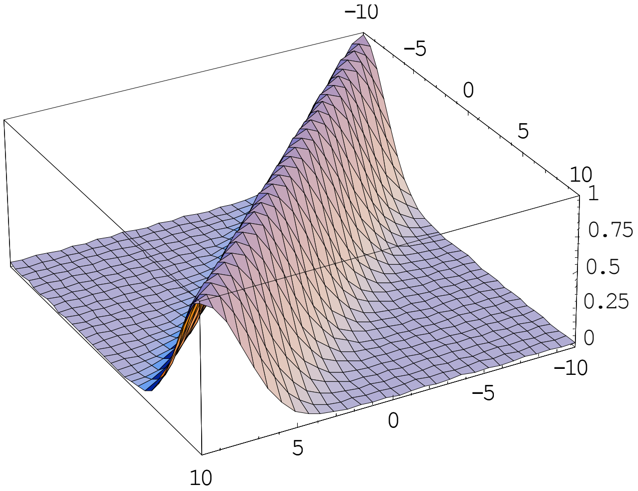

Fig.1. Gaussian RBF universal kernel with $\beta = 0.1$, in case of $x, z \in \mathbf{R}^1$

1.4.2 Polynomial kernel

Polynomial kernel of degree $d > 0$,

$$K(x,z) = \left(c + \langle x, z\rangle\right)^d$$

with

```
c=1; d=2; K[x_,z_]:= (c+xz)^d;
Plot3D[K[x,z], {x,0,10}, {z,-10,10}, PlotRange→All,
  ViewPoint→{-7.654, -1.091, 4.608}, PlotPoints→{30,30}];
```

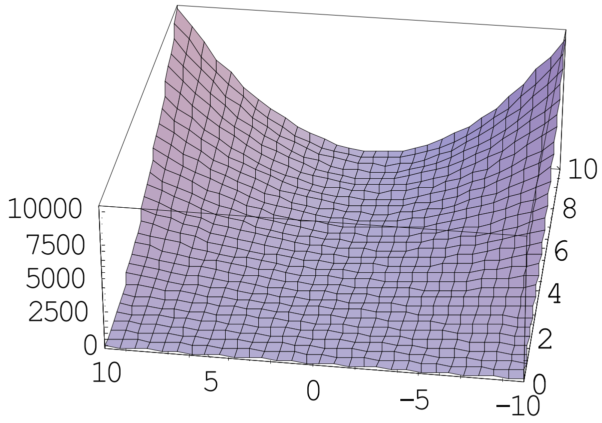

Fig. 2. Polynomial kernel, with $c = 1$, and $d = 2$, in case of $x, z \in \mathbf{R}^1$

1.4.3 Vovk's real infinity polynomial universal kernel

Vovk's real infinity polynomial universal kernel, for $d > 0$ and all compact $X \subset \left\{ x \in R^n : \|x\| < 1 \right\}$,

$$K(x,z) = \left(1 - \langle x,z \rangle\right)^{-d}$$

with

```
d=1; K[x_,z_]:= (1-xz)^-d;
Plot3D[K[x,z], {x,-0.95,0.95}, {z,-0.95,0.95}, PlotRange→All,
     ViewPoint→{5.906, 4.688, 4.913}, PlotPoints→{30,30}];
```

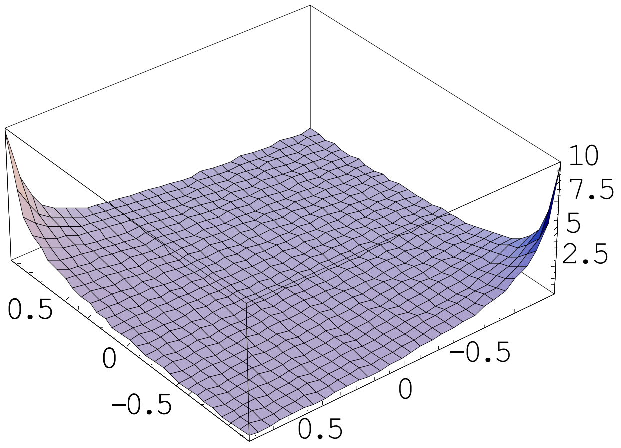

Fig. 3. Vovk 's real infinity polynomial universal kernel, with $d = 1$, in case of $x, z \in \mathbf{R}^1$

1.4.4 Stronger regularized universal Fourier kernel

The stronger regularized universal Fourier kernel with $0 < q < 1$ and all compact $X \subset [0, 2\pi]^n$

$$K(x,z)=\prod_{i=1}^{n}\frac{1-q^2}{2(1-2q\cos[x_i-z_i]+q^2)}$$

with

```
q=0.5;  K[x_,z_]:= (1-q^2)/2(1-2q Cos[x-z]+q^2);
Plot3D[K[x,z], {x,0,2π}, {z,0,2π},
     ViewPoint→{3.849, 6.909, 4.295}, PlotPoints→{30,30}];
```

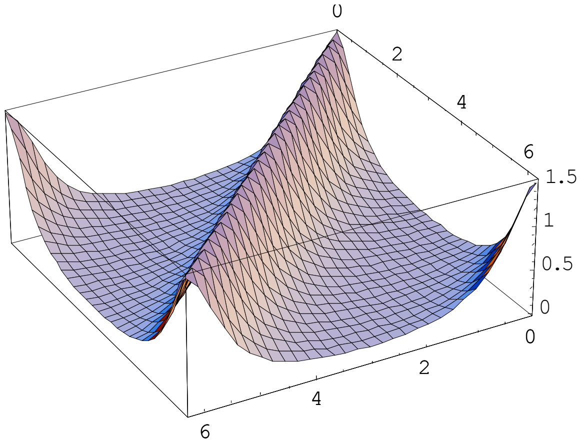

Fig. 4. The stronger regularized universal Fourier kernel with $q = 0.5$, in case of $x, z \in \mathbf{R}^1$

1.4.5 Wavelet kernel

Wavelet kernel with $a \in \mathbf{R}^1$ and all compact $X \subset \mathbf{R}^n$,

$$K(x,z)=\prod_{i=1}^{n}\left(\cos\left[1.75\frac{x_i-z_i}{a}\right]\exp\left[-\frac{(x_i-z_i)^2}{2a^2}\right]\right)$$

with

```
a=4; K[x_,z_]:= Cos[1.75(x-z)/a] exp[-(x-z)^2/2a^2];
Plot3D[K[x,z], {x,-10,10}, {z,-10,10},
   ViewPoint→{4.172, 5.346, 5.917}, PlotPoints→{30,30}];
```

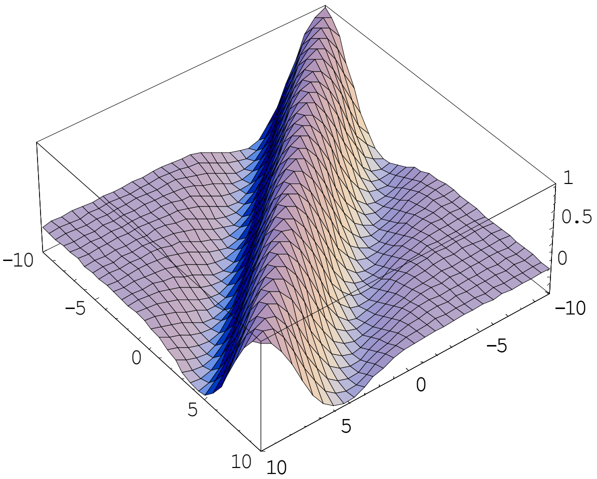

Fig. 5. Wavelet kernel with $a = 4$, in case of $x, z \in \mathbf{R}$

1.5 Making kernels from kernels

The following proposition can be viewed as shown that kernels satisfy a number of closure properties, allowing us to create more complicated kernels from simple building blocks.

Let K_1 and K_2 be kernels over $X \times X, x, z \in X \subseteq \mathbf{R}^n$, $a \in \mathbf{R}^+$ and *f* a real-valued function on *X*, in addition exist an other function $\Phi : X \to R^m$, with K_3 over $\mathbf{R}^m \times \mathbf{R}^m$, then the following functions are kernels.

$$K(x,z) = K_1(x,z) + K_2(x,z)$$
$$K(x,z) = a\ K_1(x,z)$$
$$K(x,z) = K_1(x,z)K_2(x,z)$$
$$K(x,z) = f(x)f(z)$$
$$K(x,z) = K_3(\Phi(x), \Phi(z))$$

In addition, if *p*(x) a polynomial with positive coefficients, then

$$K(x,z) = p\ K_1(x,z)$$
$$K(x,z) = \exp(K(x,z))$$

2. Support vector machine for regression

2.1 $\in$-insensitive loss function

The problem of regression is that of finding a function which approximates mapping from an input domain to the real numbers based on a training sample. We refer to the difference between the hypothesis output and its training value as the residual of the output, an indication of the accuracy of the fit at this point. We must decide how to measure the importance of this accuracy, as small residuals may be inevitable while we wish to avoid large ones. The loss function determines this measure. Each choice of loss function will result in a different overall strategy for performing regression. For example least square regression uses the sum of the squares of the residuals.

Although several different approaches are possible, we will provide an analysis for generalization of regression by introducing a threshold test accuracy θ, beyond which we consider a mistake to have been made. We therefore aim to provide a bound on the probability that a randomly drawn test point will have accuracy less than θ. If we access the training set performance using the same θ, we are effectively using the real - valued regressors as classifiers and the worst case lower bounds apply. What we must do in order to make use of dimension free bounds is to allow a margin in the regression accuracy that corresponds to the margin of a classifier. We will use the symbol γ to denote the margin, which measures the amount by which the training and test set accuracy can differ. It should be emphasized that we are therefore using a different loss function during training and testing, where γ measures the discrepancy between the two losses, implying that training point counts as mistake if its accuracy less than $\theta - \gamma$. One way of visualizing this method of assessing performance is to consider a band of size $\pm(\theta-\gamma)$ around the hypothesis function any training points lying outside this band are considered to be training mistakes. Test points count as mistakes only if they lie outside the wider band $\pm\theta$.

The linear $\in$-insensitive loss function $L^{\in}(x,y,f)$ is defined by

$$L^{\in}(x,y,f)=|y-f(x)|_{\in}=\max\big(0,|y-f(x)|-\in\big)$$

where *f* is a real-valued function on a domain $X, x\in X$ and $y\in\mathbf{R}$. Similarly the quadratic $\in$-insensitive loss is given by

$$L_2^{\in}(x,y,f)=\big(|y-f(x)|_{\in}\big)^2$$

2.2 Support Vector Regression

SVR uses an admissible kernel, which satisfies the Mercer 's condition to map the data in input space to a high dimensional feature space in which we can process a regression problem in linear form. Let $x\in R^n$ and $y\in\mathbf{R}$, where $\mathbf{R}^n$ represents input space. By some nonlinear mapping Φ, the vector *x* is mapped into a feature space in which a linear regressor function is defined,

$$y=f(x,w)=\langle w,\,\Phi(x)+b\rangle$$

We seek to estimate this *f* function based on independent uniformly distributed data $\{\{x_1,y_1\},...,\{x_m,y_m\}\}$, by finding *w* which minimizing the quadratic $\in$-insensitive losses, with $\in=\theta-\gamma$ namely the following function should be minimize [11]

$$c\sum_{i=1}^{m}L_2^{\in}(x_i,y_i,f)+\frac{1}{2}\|w\|^2\rightarrow\min$$

where *w* is weight vector and *c* is a constant parameter.

Considering dual representation of a linear regressor, $f(x)$ can be expressed as [12],

$$f(x)=\sum_{i=1}^{m}\beta_i y_i\langle\Phi(x_i),\Phi(x)\rangle+b$$

which means that the regressor can be expressed as a linear combination of the training points. Consequently using an admissible kernel, a kernel satisfying the Mercer' s condition, we get

$$f(x)=\sum_{i=1}^{m}\beta_i y_i K(x_i,x)+b=\sum_{i=1}^{m}\alpha_i K(x_i,x)+b\ .$$

By using Lagrange multiplier techniques, the minimization problem leads to the following dual optimization problem [12],

$$\text{maximize}\quad W(\alpha)=\sum_{i=1}^{m}y_i\alpha_i-\in\sum_{i=1}^{m}|\alpha_i|-\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j\left(K(x_i,x_j)+\frac{1}{c}\delta_{ij}\right)$$

$$\text{subject to} \quad \sum_{i=1}^{m} \alpha_i = 0$$

Let

$$f(x) = \sum_{i=1}^{m} \alpha_i^* K(x_i, x) + b^*$$

where α^* is the solution of the quadratic optimization problem and b^* is chosen so that $f(x_i) = y_i - \in - \frac{\alpha_i^*}{c}$ for any $\alpha_i^* > 0$.

For samples are inside the $\in$-tube, $\{\, x_i : \left| f(x_i) - y_i \right| \,\} < \in$, the corresponding α_i^* is zero. It means we do not need these samples to describe the weight vector *w*. Consequently

$$f(x) = \sum_{i \in SV} \alpha_i^* K(x_i, x) + b^*$$

where

$$SV = \left\{\, i : \left| f(x_i) - y_i \right| \geq \in OR\ \alpha_i^* > 0 \,\right\}$$

These x_i sample vectors $\left\{\, x_i : i \in SV \,\right\}$ that come with nonvanishing coefficients α_i^* are called support vectors.

2.3 The main features of SVR

- Application of nonlinear mapping of data from input space into a feature space, in which linear regressor (machine or learning machine) is used. The dual representation of linear regressor leads to the employment of kernel function.
- Quadratic $\in$-insensitive loss function is used as objective function with regularization term, *c* for estimation of the weight vector in the kernel representation of the regressor function f, ensuring $\in$ accuracy, and leading the a part of samples, called support vector, only which influence the weight vector.

3. Implementation of SVR in *Mathematica*

3.1 Steps of implementation

The dual optimization problem can be solved conveniently using *Mathematica*. In this section, the steps of the implementation of SVR algorithm are shown by solving a function approximation problem [13]. The function to be approximated is,

```
z[{x_,y_}]:= (x^2-y^2) Sin[0.5x]
```

Let us display it

```
p1= Plot3D[z[{x,y}], {x,-10,10}, {y,-10,10}, PlotPoints→{30,30}];
```

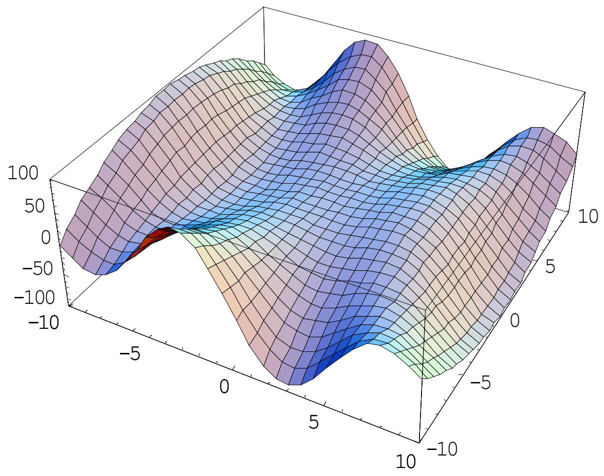

Fig. 6. Function to be approximated

A training set can be created in form $\{\{x_i, y_i\}, z\{x_i, y_i\}\}$

```
data= Table[{{x,y}, z[{x,y}]}, {x,-10,10,2.5}, {y,-10,10,2.5}];
```

Let us separate the dependent and independent variables,

```
{xym, zm}= Transpose[Flatten[data,1]];
```

Wavelet kernel with parameter $a = 4$, in case of dimension $n = 2$

```
n= 2; a= 4;
```

$$\texttt{K[u\_,v\_]:=}\prod_{i=1}^{n}\texttt{(Cos[1.75(u[[i]]-v[[i]])/a] Exp[-(u[[i]]-v[[i]])}^{2}\texttt{/2a}^{2}\texttt{])}$$

The number of the data pairs in the training set, *m* is

```
m= Length[xym]
81
```

Create the objective function $W(\alpha)$ to be maximized, with parameters

```
∈ =0.0025; c=200.;
```

First, we prepare a matrix *M*

```
M= (Table[N[K[xym[[i]], xym[[j]]]], {i,1,m}, {j,1,m}] +
   (1/c)IdentityMatrix[m]);
```

then the objective function can be expressed as,

$$\texttt{W=}\sum_{i=1}^{m}\alpha_i\ \texttt{zm[[i]] -}\in\sum_{i=1}^{m}\texttt{Abs[}\alpha_i\texttt{] - (1/2)}\sum_{i=1}^{m}\sum_{j=1}^{m}\texttt{(}\alpha_i\,\alpha_j\ \texttt{M[[i, j]]);}$$

The constrains for the unknown variables are

$$\texttt{q= Apply[And, Join[Table[-c <}\ \alpha_i \le \texttt{c, \{i, 1, m\}], \{(}\sum_{i=1}^{m}\alpha_i\texttt{=0)\}]];}$$

However the maximization problem is a convex quadratic problem, from practical reasons to maximize the objective function the built in function *NMaximize* is applied. *NMaximize* implements several algorithms for finding constrained global optima. The methods are flexible enough to cope with functions that are not differentiable or continuous, and are not easily trapped by local optima.

Possible settings for the *Method* option include *NelderMead*, *DifferentialEvolution*, *SimulatedAnnealing* and *RandomSearch*.

Here we use *DifferentialEvolution*, which is a genetic algorithm that maintains a population of specimens, $x_1, ..., x_n$, represented as vectors of real numbers ("genes"). Every iteration, each x_i chooses random integers *a*, *b* and *c* and constructs the mate $y_i = x_i + \gamma(x_a + (x_b - x_c))$, where γ is the value of *ScalingFactor*. Then x_i is mated with y_i according to the value of *CrossProbability*, giving us the child z_i. At this point x_i competes against z_i for the position of x_i in the population. The default value of *SearchPoints* is Automatic, which is *Min[10*d,50]*, where *d* is the number of variables.

We need the list of unknown variables α,

```
vars= Table[αI, {i,1,m}];
```

Then the solution of the maximization problem is,

```
sol= NMaximize[{W,g}, vars, Method→DifferentialEvolution]
{60026.6, {α1→56.8943, α2→52.1366, ……, α81→-56.8928}}
```

The consistency of this solution can be checked by computing values of *b* for every data points. Theoretically, these values should be same for any data points however, this is only approximately true.

```
                          m
bdata= Table[(zm[[j]] -∑  αi K[xym[[j]],xym[[i]]] -∈ -αj /c)/.
                         i=1
        sol [[2]], {j,1,m}]
{0.000108394, 0.0000740285, ……, -0.00474002}
```

The value of *b* can be chosen as the average of these values

```
b= Apply[Plus,bdata]/m
-0.00248556
```

Then the regressor function is,

```
                     m
f[{x_, y_}]:= (∑  αi K[{x,y}, xym[[i]]] + b)/.sol [[2]];
                    i=1
```

The result in symbolic form is,

```
Short[ f [{x, y}], 5}
```

$$-0.00248556\ -56.8928\ e^{-0.03125(-10.+x)^2-0.03125(<<1>><<1>>)^2}$$
$$\mathrm{Cos}[0.4375(-10.+x)]\ \mathrm{Cos}[0.4375(-10.+y)]+$$
$$<<116>>+<<19>>+<<3>>-55.9222<<3>>+$$
$$56.8943\ e^{-0.03125(10.+x)^2-0.03125(10.+y)^2}$$
$$\mathrm{Cos}[0.4375(10.+x)]\ \mathrm{Cos}[0.4375(10.+y)]$$

This function can be compared with the original one,

```
p1= Plot3D[f[{x,y}], {x,-10,10}, {y,-10,10}, PlotPoints→{30,30}],
  DisplaFunction→Identity];
Show[GraphicsArray[{p1, p2}], DisplayFunction→ $DisplayFunction];
```

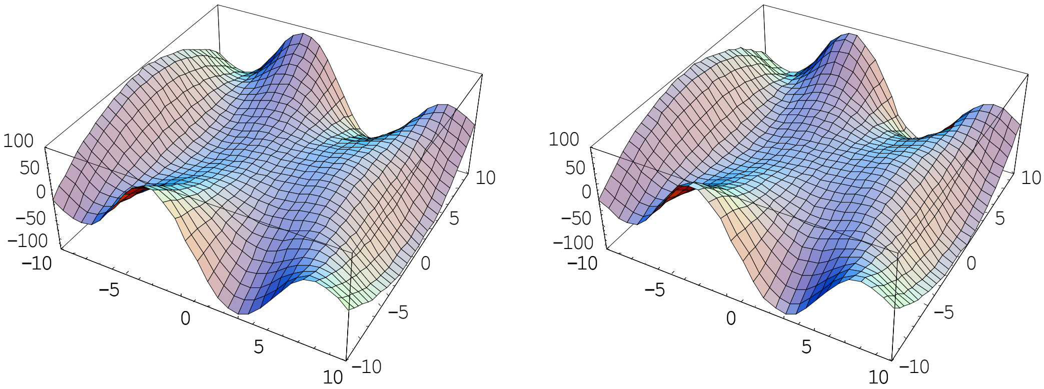

Fig. 7. Original surface (to left) and the approximated surface using SVR with wavelet kernel

These figures indicate a very good approximation.

3.2 Mathematica modul for SVR

These steps can be collected in a module, where the vector *xm* contains the input vectors and the vector *ym* contains the corresponding scalar output values,

```
SupportVectorRegression[{xm_, ym_,}, K_,∈_,c_]:= Module[
   {m, n, M, i, j, W, g, vars, sol, b},
   m= Length[xm]; n = Length[xm[[1]]];
   M= Table[K[xm[[i]], xm[[j]]], {i,1,m},{j,1,m}]
     +(1/c)IdentityMatrix[m];
   W=\sum_{i=1}^{m} α_i ym[[i]] -∈\sum_{i=1}^{m} Abs[α_i] -(1/2) \sum_{i=1}^{m} \sum_{j=1}^{m} (α_i α_j M[[i,j]]);
   g= Apply[And, Join[Table[-c<α_i≤c, {i,1,m}], {(\sum_{i=1}^{m} α_i==0)}]];
   vars= Table[α_i,{i,1,m}];
   sol= NMaximize[{W,g}, vars, Method→DifferentialEvolution][[2]];
   b= (1/m) Apply[Plus, Table[(ym[[j]] -\sum_{i=1}^{m} α_i K[xm[[i]],
       xm[[j]]] -∈ -(α_i/c))/.sol, {j,1,m}]];
   {(\sum_{i=1}^{m} α_i K[xm[[i]], Table[x_j, {j,1,n}]] +b)/.sol, vars /.sol}];
```

The module outputs are the regression function in symbolic form and the values of α 's. To test our module, let us solve another problem, [14]. The following function should be approximated in [-5, 5] × [-5, 5],

```
z[{x_,y_}]:= Sin[\sqrt{x^2+y^2} ] / \sqrt{x^2+y^2}
```

using a stronger regularized universal Fourier kernel with parameter, $q = 0.25$

```
q= 0.25;
K[u_,v_]:= \prod_{i=1}^{n} (1-q^2) / (2(1 -2q Cos[u[[i]] -v[[i]]] +q^2))
```

Because this kernel is valid for $x, y \in X \subset (0, 2\pi)^n$, therefore the variables of *x, y* should be scaled,

```
z[{x_,y_}]:= (Sin[√((5x/π−5)^2 +(5y/π−5)^2) ] /  √((5x/π−5)^2 +(5y/π−5)^2)
```

Let us display this function,

```
p1= Plot3D[ z[{x,y}], {x,0,2π}, {y,0,2π}, PlotPoints→{30,30}],
```

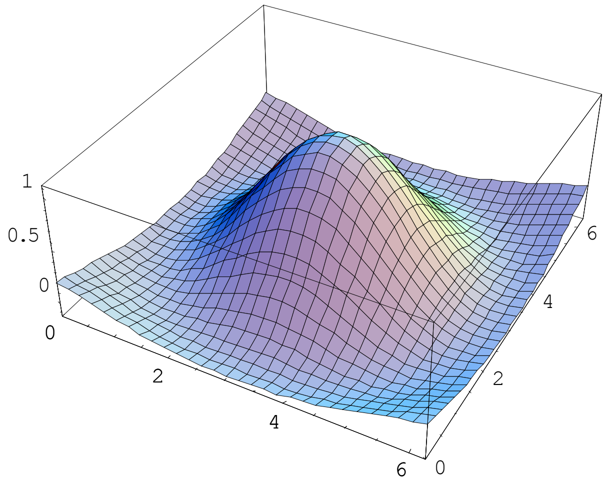

Fig. 8. Function to be approximated

Generating data,

```
data= Table[{{x,y}, z[{x,y}]}, {x,0,2π,2π/9}, {y,0,2π,2π/9}]//N;
```

then calling the SVR module,

```
F= SupportVectorRegression[data, K, ∈, c];
```

Symbolic form of the result,

```
Short[F,18]
0.0253161-
 0.112811/((1.0625-0.5Cos[0.-x1])(1.0625-0.5Cos[0.-x2]))+
 0.153849/((1.0625-0.5Cos[0.698132-x1])(1.0625-0.5Cos[0.-x2]))+
 0.0768015/((1.0625-0.5Cos[1.39626-x1])(1.0625-0.5Cos[0.-x2]))+
 <<139>>+
 0.0121515/((1.0625-0.5Cos[4.18879-x1])(1.0625-0.5Cos[6.28319-x2]))+
 0.0749787/((1.0625-0.5Cos[4.88692-x1])(1.0625-0.5Cos[6.28319-x2]))+
 0.145839/((1.0625-0.5Cos[5.58505-x1])(1.0625-0.5Cos[6.28319-x2]))+
 0.101006/((1.0625-0.5Cos[6.28319-x1])(1.0625-0.5Cos[6.28319-x2]))
```

The surface approximation can be qualified by comparing it with the original one,

```
p2= Plot3D[ F, {x1,0,2π},  {x2,0,2π},  PlotPoints→{30,30}],
DisplaFunction→Identity];
Show[GraphicsArray[{p1,p2}], DisplayFunction→ $DisplayFunction];
```

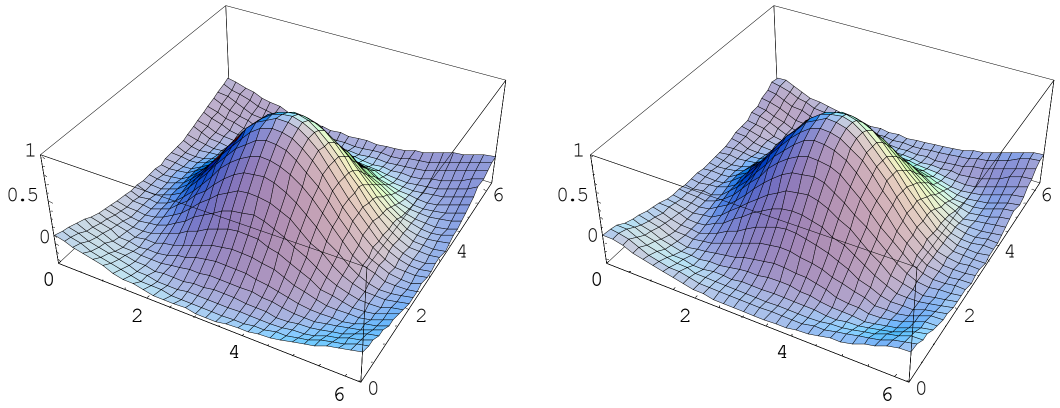

Fig. 9. Original surface (to left) and the approximated surface using SVR with stronger regularized universal Fourier kernel

The third test problem is a well-known benchmark, the approximation of the following function from noisy training samples [15],

```
z[{x_,y_}]:= 1.9(1.35 +e^x Sin[13(x -0.6)^2] e^-y Sin[7y]);
```

Let us display it

```
p1= Plot3D[z[{x,y}],{x,0,1},{y,0,1},PlotPoints→{30,30},
    Shading→False, ViewPoint→{-1.097, -1.601, 0.963}];
```

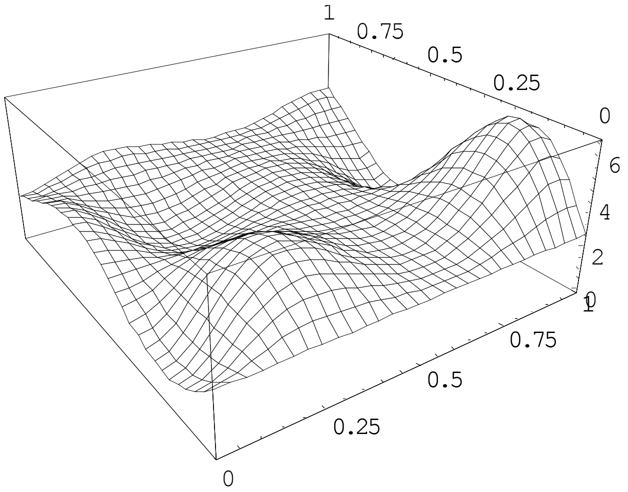

Fig. 10. Function to be approximated

Noise of normal distribution $N(\mu, \sigma)$, with $\mu = 0$ and $\sigma = 0.15$ was added to the function.

```
<<Statistics`ContinuousDistributions`
zn[{x_,y_}]:= z[{x,y}] +Random[NormalDistribution[0.,0.15]]
```

A training set can be created in form $\{\{x_i, y_i, z(x_i, y_i)\}$

```
data= Flatten[Table[N[{i,j,zn[{i,j}]}],{i,0,1, 1/9},{j,0,1, 1/9}],1];
<<Graphics`Graphics3D`
p2=ScatterPlot3D[data,BoxRatios→{1,1,1},PlotStyle→PointSize[0.019],
ViewPoint→{-1.014, -1.580, 1.237}, DisplayFunction→Identity];
Show[{p1,p2}, DisplayFunction→$DisplayFunction];
```

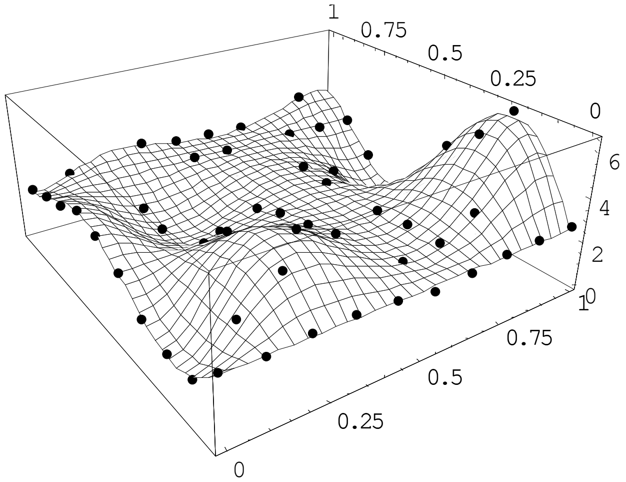

Fig. 11. Function to be approximated and the noisy training samples

Let us separate the dependent and independent variables,

```
{xym,zm}= Transpose[Map[{{#[[1]], #[[2]]}, #[[3]]}&, data]];
```

Wavelet kernel with parameter $a = 0.23$, in case of dimension $n = 2$

```
n= 2; a= 0.23;
K[u_,v_]:= ∏_{i=1}^{n} (Cos[1.75(u[[i]]-v[[i]])/a] Exp[-(u[[i]]-v[[i]])^2 /2a^2])
```

The number of the data pairs in the training set, *m* is

```
m= Length[xym]
100
```

Create the objective function $W(\alpha)$ to be maximized, with parameters

```
∈ = 0.0025; c= 200.;
F= SupportVectorRegression[{xym,zm},K,∈,c];
```

Symbolic form of the result,

```
Short[F[[1]],18]
2.65449+
 6.64787 e^(-9.4518(0.-x1)^2-9.4518(0.-x2)^2) Cos[7.6087(0.-x_1)] Cos[7.6087(0.-x_2)]-
 7.97406 e^(-9.4518(0.111111-x1)^2-9.4518(0.-x2)^2) Cos[7.6087 (0.111111 -x_1)]
   Cos[7.6087(0.-x_2)]+
 <<144>>+
 7.14558 e^(-9.4518(0.888889-x1)^2-9.4518(1.-x2)^2) Cos[7.6087 (0.888889-x_1)]
   Cos[7.6087(1.-x_2)]-
 1.32496 e^(-9.4518(1. -x1)^2-9.4518(1.-x2)^2) Cos[7.6087(1.-x_1)] Cos[7.6087 (1.-x_2)]
```

```
p3= Plot3D[F[[1]], {x_1, 0,1}, { x_2 ,0,1}, PlotPoints→{30,30},
  ViewPoint→{-1.014, -1.580, 1.237}, DisplayFunction→Identity];
Show[GraphicsArray[{p1,p3}], DisplayFunction→$DisplayFunction];
```

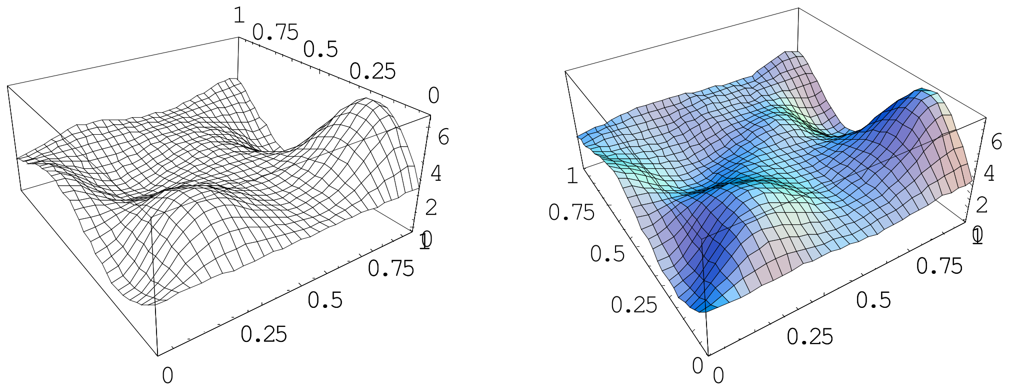

Fig. 12. Original surface (to left) and the approximated surface using SVR with wavelet kernel

These figures indicate a good approximation.

4. Engineering application

4.1 Problem definition

We should like to develop a statistical model for forecasting the peak of the flood-wave of the Danube at Budapest. The following 24 data triplets registered from the year of 1896 up to 1955 available in [16],

Table 1. Model variables

x	y	z
rainfall	water-level at the beginning of raining	peak of water-level
in mm	in cm	in cm

The measured data are,

```
data = {{58, 405, 590}, {52, 450, 660}, {133, 350, 780},
{179, 285, 770}, {98, 330, 710}, {72, 400, 640}, {72, 550, 670},
{43, 480, 520}, {62, 450, 660}, {67, 610, 690}, {64, 380, 500},
{33, 460, 460}, {57, 425, 610}, {62, 560, 710}, {54, 420, 620},
{48, 620, 660}, {86, 390, 620}, {74, 350, 590}, {95, 570, 740},
{44, 710, 730}, {77, 580, 720}, {46, 700, 640}, {123, 560, 805},
{62, 430, 673}};
```

The model $z = f(x, y)$ should generalizes these data and ensures forecasting z from measured x and y values. Now we have a real regression problem, because the data are not on a smooth surface,

```
<<Graphics`Graphics3D`
p0= ScatterPlot3D[data,BoxRatios→{1,1,1},PlotStyle→PointSize[0.03],
    AxesLabel→{"Rainfall ","Starting Level","Peak Level"}];
```

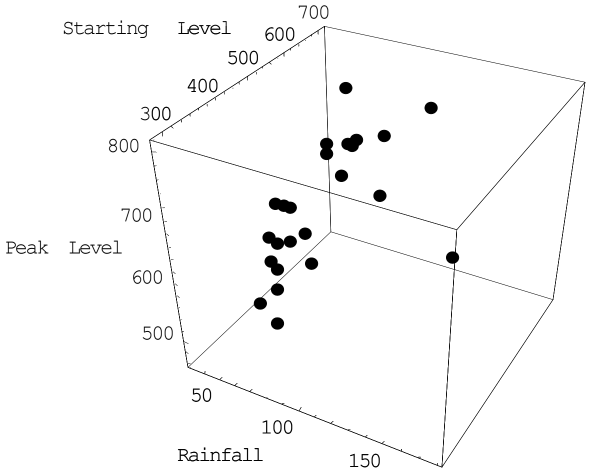

Fig. 13. Triplets of measurement

A general model means that the set of input/output relationships, derivate from the training set, apply equally well to new sets of data from the same problem not included in the training set. The main goal is thus the generalization to new data of the relationships learned on the training set.

In order to test the generality of our model to be developed, we divide the data into a training set and a test set. The number of data is

```
ndata= Length[data]
24
```

Let consider every fourth data as test data

```
dataV= Table[data[[i]], {i,1,ndata,4}]
{{58, 405, 590}, {98, 330, 710}, {62, 450, 660},
{57, 425, 610}, {86, 390, 620}, {77, 580, 720}}
```

The remaining elements are for training

```
dataT= Complement[data,dataV]
{{33, 460, 460}, {43, 480, 520}, {44, 710, 730}, {46, 700, 640},
{48, 620, 660}, {52, 450, 660}, {54, 420, 620}, {62, 430, 673},
{62, 560, 710}, {64, 380, 500}, {67, 610, 690}, {72, 400, 640},
{72, 550, 670}, {74, 350, 590}, {95, 570, 740}, {123, 560, 805},
{133, 350, 780}, {179, 285, 770}}
```

Let us display the training and test set

```
p1= ScatterPlot3D[dataT, BoxRatios→{1,1,1},
   PlotStyle→{RGBColor[0,0,1], PointSize[0.03]},
   AxesLabel→{"Rainfall ","Starting Level","Peak Level"},
   DisplayFunction->Identity];
p2= ScatterPlot3D[dataV,BoxRatios→{1,1,1},
   PlotStyle→{RGBColor[1,0,0], PointSize[0.03]},
   AxesLabel→{"Rainfall ","Starting Level","Peak Level"},
   DisplayFunction→Identity];
Show[GraphicsArray[{p1,p2}], DisplayFunction→$DisplayFunction];
```

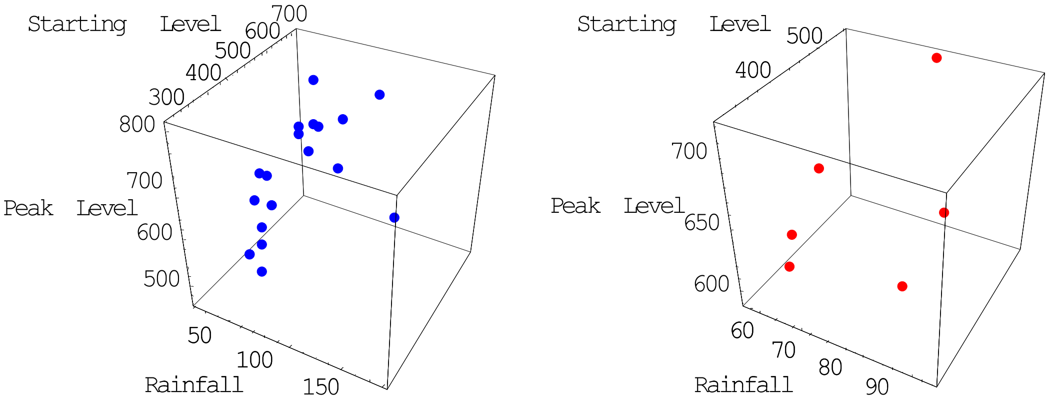

Fig. 14. Traning data (to left) and test data

Preparation of these sets for learning and testing

```
xym= Map[Drop[#,-1]&, data];
zm= Map[Take[#,{3,3}]&, data]//Flatten;
xymT= Map[Drop[#,-1]&, dataT];
zmT= Map[Take[#,{3,3}]&, dataT]//Flatten;
```

4.2 Solution with SVR

Employing wavelet kernel with parameter $a = 0.3$

```
n= 2; a= 0.3;
              n
K[u_,v_]:=    ∏   (Cos[1.75(u[[i]]-v[[i]])/a] Exp[-(u[[i]]-v[[i]])^2 /2a^2])
             i=1
```

and with the following parameters for SVR

```
∊ = 0.025; c= 200.;
F= SupportVectorRegression[{xymT,zmT},K,∊,c];
```

The regressor function is,

```
Short[F[[1]],15]
658.753+
 110.65 e^(-5.55556(179.-x1)^2-5.55556(285.-x2)^2) Cos[5.83333(179-x1)]
   Cos[5.83333(285-x2)]-
 68.4051 e^(-5.55556(74.-x1)^2-5.55556(350.-x2)^2) Cos[5.83333(74-x1)]
   Cos[5.83333(350-x2)]+
 <<18>>+
 1.19631 e^(-5.55556(48.-x1)^2-5.55556(620.-x2)^2) Cos[5.83333(48-x1)]
   Cos[5.83333(620-x2)]-
 18.6538 e^(-5.55556(46.-x1)^2-5.55556(700.-x2)^2) Cos[5.83333(46-x1)]
   Cos[5.83333(700-x2)]+
 70.8486 e^(-5.55556(44.-x1)^2-5.55556(710.-x2)^2) Cos[5.83333(44-x1)]
   Cos[5.83333(710-x2)]
```

```
f[{x_,y_}]= F[[1]]/.{x1→x, x2→y};
```

Let us display the relative error of the approximation on the whole data set

```
<< Graphics`Graphics`
BarChart[((Abs[zm-Map[f[#]&,xym]])/zm)100,
Ticks→{{1,5,9,13,17,21,25}, {0.,10,20}}, PlotRange→{0,20},
  AspectRatio→0.5];
```

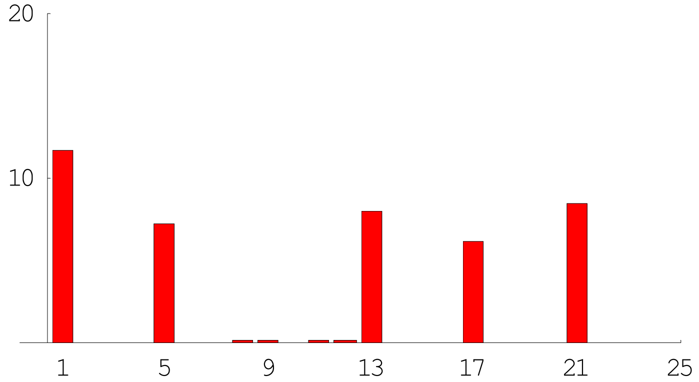

Fig. 15. Relative approximation error in percent on the whole data set for ∈ = 0.025

The figure shows, that on most of the test data, which were not included in the training, the error is considerably high, while on the elements of the training set the error is negligible. The maximum error is

```
Max[((Abs[zm-Map[f[#]&,xym]])/zm)100]
11.653
```

and its standard deviation is

```
StandardDeviation[((Abs[zm-Map[f[#]&,xym]])/zm)100]
3.52979
```

Let us see the values of α 's,

```
F[[2]]
{-197.758, -138.057, 70.8485, -18.654, 1.19758, 1.19684, -38.5545,
14.1321, 50.948, -157.957, 31.0475, -18.654, 11.1469, -68.4052,
80.7987, 145.475, 120.6, 110.649}
```

As we know, the input vector *xymT_i* is a support vector, if the corresponding $\alpha_i \neq 0$. We shall consider $\alpha_i \neq 0$ if its absolute value is greater than 10^{-3}. Then the support vectors can be selected in the following way,

```
supportvectors= Extract[xymT,Position[F[[2]],_?(Abs[#]>10^-3 &)]]
{{33, 460}, {43, 480}, {44, 710}, {46, 700}, {48, 620}, {52, 450},
{54, 420}, {62, 430}, {62, 560}, {64, 380}, {67, 610}, {72, 400},
{72, 550}, {74, 350}, {95, 570}, {123, 560}, {133, 350}, {179, 285}}
```

Now, all of the training vectors are support vector. It is easy to visualize them,

```
<<Graphics`PlotField`
ListPlotVectorField[Map[{{0,0},#}&,supportvectors],
  AspectRatio→0.5];
```

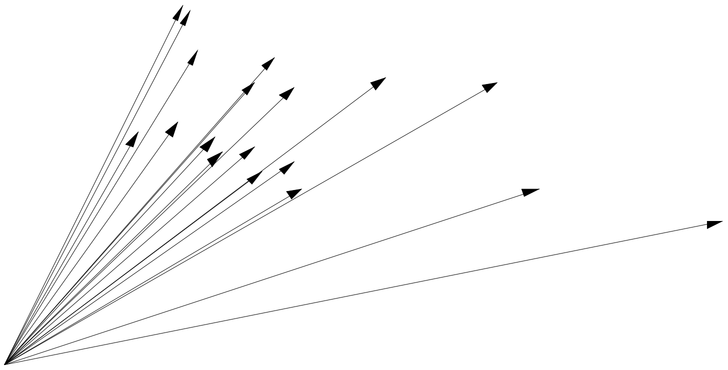

Fig. 16. All training vector are support vector in case of ∈ = 0.25

In order to eliminate the approximation error on the test set, to improve the generalization of our model, the parameter ∈ should be increased. Now let us carry out the computation with ∈ = 10.

```
∈= 10.;
F= SupportVectorRegression[{xymT,zmT},K,∈,c];
```

The regressor function is,

```
Short[F[[1]],15]
648.778 +
 102.114 e^(-5.55556(179-x1)^2-5.55556(285-x2)^2) Cos[5.83333(179-x1)]
   Cos[5.83333(285-x2)]-
 57.0896 e^(-5.55556(74-x1)^2-5.55556(350-x2)^2) Cos[5.83333(74-x1)]
   Cos[5.83333(350-x2)]+
 <<23>>+
 62.3134 e^(-5.55556(44-x1)^2-5.55556(710-x2)^2) Cos[5.83333(44-x1)]
   Cos[5.83333(710-x2)]
f[{x_,y_}]= F[[1]]/.{x1 -> x, x2 -> y};
```

Let us display again the relative error of the approximation on the whole data set

```
BarChart[((Abs[zm-Map[f[#]&,xym]])/zm)100,
  Ticks→{{1,5,9,13,17,21,25},{0.,10,20,30}},PlotRange→{0,20},
  AspectRatio→0.5];
```

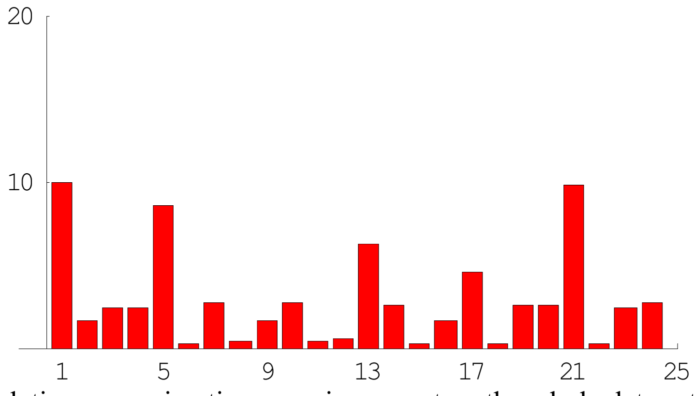

Fig. 17. Relative approximation error in percent on the whole data set for ∈ = 10

This figure shows that although the maximum of the approximation error decreased just a little, the error distribution is tending to a more uniform one, which indicates higher statistical confidence of the model.

```
Max[((Abs[zm-Map[f[#]&,xym]])/zm)100]
```

```
9.96234
```

and its standard deviation is

```
StandardDeviation[((Abs[zm-Map[f[#]&,xym]])/zm) 100]
2.93459
```

Let us see the values of α 's,

```
F[[2]]
{-186.444, -126.743, 62.3098, -7.33528, 0.000144156, -3.39837×10^-6,
-27.2438, 5.59695, 42.4118, -146.645, 22.516, -7.34333, 2.61709,
-57.0875, 72.2595, 136.947, 112.072, 102.11}
```

Now, we have two small α 's indicating less support vector as before

```
supportvectors= Extract[xymT,Position[F[[2]],_?(Abs[#]>10^-3 &)]];
ListPlotVectorField[Map[{{0,0},#}&,supportvectors],
  AspectRatio->0.5];
```

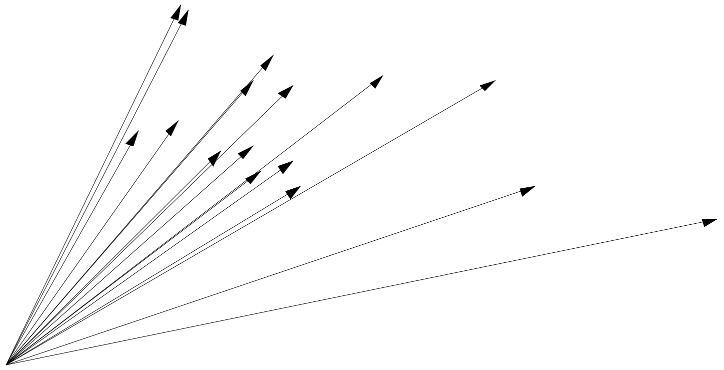

Fig. 18. Support vectors in case of ∈ = 10

5 Conclusions

Support vector regression method has been implemented in *Mathematica* code as a simple function and tested on four different problems. The solutions of them, especially the last one, a real world civil engineering problem, forecasting of the peak of a floodwave, clearly demonstrated the generalization ability of SVR as well as its robustnees.

Additional computations were also carried out with polynomial regression with different orders (2, 3 and 4) and RBF neural network with different number of neurons (3, 5 and 10). In case of these methods the maximum of the relative error was always considerably higher than that of SVR and the phenomena of overfitting always took place with increasing number of model parameters.

The *Mathematica* notebook version of this paper is available in [17].

Acknowledgements

Our investigations are supported by the National Scientific Research Found (OTKA T-037929 and T-037880). The authors wish to thank professor D. Holnapy for his valuable comments.

References

[1] Tzafestas S G, Dalianis P J, Anthopoulos G (1996): On the overtraining phenomenon of backpropagation neural networks, *Mathematics and Computers in Simulation*, Vol. 40. pp. 507 - 521.

[2] Neumaier A: Solving ill-conditioned and singular linear systems: a tutorial on regularization, WWW:http://solo.cma.univie.ac.at/~neum/

[3] Friess T T, Harrison R F (1999): A kernel based Adaline or function approximation. *Intelligent Data Analysis*, Vol. 3. pp. 307 - 313.

[4] Burger M, Neubauer A (2003): Analysis of Tikhonov regularization for function approximation by neural network. *Neural Networks*, Vol.16. pp.79 - 90.

[5] Berthold M, Hand D J /Eds./ (2003): Intelligent Data Analysis, An Introduction. Springer Verlag.

[6] Burgers C J C (1998): A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery 2,* pp. 121 - 167.

[7] Hearst M A (1998): Support Vector Machines. *IEEE Intelligent Systems*, pp. 18 -28, July/August, 1998.

[8] Kim K I, Jung K, Park S H, Kim H J (2002): Support Vector Machines for Texture Classification. *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. 24. No.11. pp. 1542 – 1550.

[9] Genov R, Cauwenberghs G, Keltron (2003): Support Vector "Machine" in Silicon. *IEEE Trans. Neural Networks,* Vol. 14. No.5. pp.1426 -1433.

[10] Steinwart I: On the Optimal Parameter Choice for ν - Support Vector Machines, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 25.

[11] Schölkopf B, Smola A J (1998): A Tutorial on Support Vector Regression. *NeuroCOLT2 Technical Report Series*, NC2 - TR - 1998 - 030.

[12] Cristianini N, Shawe-Taylor J (2003): An introduction to Support Vector Machines and other kernel - based learning methods. *Cambridge, University Press*.

[13] Zhang L, Zhou W, Jiao L (2004): Wavelet Support Vector Machine. *IEEE Trans. Systems, Man and Cybernetics - Part B: Cybernetics,* Vol. 4. No.1. pp. 34 -39.

[14] Hong X, Sharkey P M, Warwick. *A* (2003): Robust Nonlinear Identification Algorithm Using PRESS Statistic and Forward Regression*.* *IEEE Trans. Neural Networks,* Vol. 14. No. 2., pp.454 - 458.

[15] Cherkassy V, Gehring D, Mulier F (1996): Comparison of adaptive methods for function estimation from samples. *IEEE Trans. Neural Networks* 7, pp. 969-984.

[16] Szesztay, K (1961): Einige Methoden der Vorhersage der Abflussverhältnisse, *Vizgazdálkodási Tudományos Kutató intézet*.

[17] Paláncz B: Electronic version of Support Vector Regression via *Mathematica*, http://library.wolfram.com/infocenter/MathSource/5270/

* * *

Paláncz B, Völgyesi L, Popper Gy. (2005) Support Vector Regression via Mathematica. Periodica Polytechnica Civ. Eng, Vol. 49, Nr. 1. pp. 59-84.

Dr. Lajos VÖLGYESI, Department of Geodesy and Surveying, Budapest University of Technology and Economics, H-1521 Budapest, Hungary, Műegyetem rkp. 3.
Web: http://sci.fgt.bme.hu/volgyesi   E-mail: volgyesi@eik.bme.hu