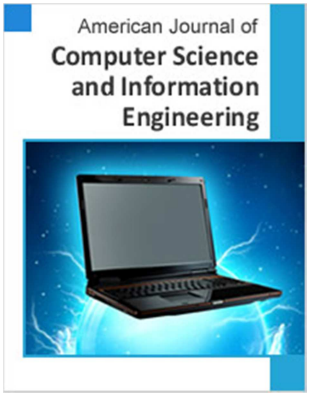
American Journal of Computer Science and Information Engineering
2016; 3(1): 1-6
http://www.aascit.org/journal/ajcsie
ISSN: 2381-1110 (Print); ISSN: 2381-1129 (Online)

American Association for Science and Technology



Keywords
Ordinary Differential Equation,
ϵ-Least Squares Support Vector Machine,
Quadratic Programming Problem,
Constrained Optimization Problem

Received: January 25, 2016
Accepted: February 20, 2016
Published: March 28, 2016

ϵ-Least Square Support Vector Method for Solving Differential Equations

Mojtaba Baymani[1, *], Omid Teymoori[1], Seyed GHasem Razavi[2]

[1]Department of Computer and Mathematics, Quchan University of Advanced Technology, Quchan, Iran
[2]Department of Mathematics, Ferdowsi University of Mashhad, Mashhad, Iran

Email address
m_baymani@qiet.ac.ir (M. Baymani), omid.teymoorikase@gmail.com (O. Teymoori), gh.razavi@ymail.com (S. G. Razavi)

[*]Corresponding author

Citation
Mojtaba Baymani, Omid Teymoori, Seyed GHasem Razavi. ϵ-Least Square Support Vector Method for Solving Differential Equations. *American Journal of Computer Science and Information Engineering*. Vol. 3, No. 1, 2016, pp. 1-6.

Abstract
In this paper, a new method based on ϵ-Least Square Support Vector Machines (ϵ-LSSVM's) is developed for obtaining the solution of the ordinary differential equations in an analytical function form. The approximate solution procedure is based upon forming of support vector machines (SVM's) whose parameters are adjusted to solve a quadratic programming problem. The details of the method are discussed, and the capabilities of the method are illustrated by solving some differential equations. The performance of the method and the accuracy of the results are evaluated by comparing with the available numerical and analytical solutions.

1. Introduction

We consider the m-th order linear ODE by initial conditions as follows:

$$y^{(m)}(t) - \sum_{i=0}^{m-1} f_i(t) y^{(m-i)}(t) = r(t), t \in [a, c] \quad (1)$$

$$y^{(i)}(a) = p_i, i = 0,1, \ldots\ldots, m-1 \quad (2)$$

where $r(t)$, $f_i(t)$ are the given functions, p_i is the given scalar and $y^{(i)}(t)$ denotes the i-th derivative of function y(t) with respect to t.

Nowadays, Many of the problems in studies fields; including engineering, medical sciences and medicine can be applied to a set of differential equations (DE's) through a process of mathematical modeling is reduced. Because in most cases it is not easy to get the exact solution of DE's, so numerical methods should be applied. There are a lot of mathematical methods to solve DE's. Most techniques offer a discrete solution such as finite difference (for example predictor-corrector, or Runge-Kutta methods) or a solution of limited differentiability (for example finite elements). These methods define a mesh (domain discretization) and functions are approximated locally ([1]-[2]).

Recently, researchers proposed some new methods that were based on artificial neural network (ANN) models. The ANN methods in comparison with other numerical methods have more advantages. The solution of DE's in ANN methods are differentiable and continuous ([3]-[5]). Lagaris et al. [3] used neural networks to solve ODE's and PDE's. They used multilayer perceptron in their network architecture. Malek et al. [4] proposed a novel hybrid method based on optimization techniques and neural networks methods for

the solution of high order ODE's. They offered a new solution method for the approximated solution of high order ODE's using innovative mathematical tools and neural-like systems of computation. Fasshauer [6] proposed a new unsupervised training method by using Radial Basis functions (RBF) to solve DE's. The methods based on genetic programming have also been proposed as well as methods that induce the underlying differential equation from experimental data. The technique of genetic programming is an optimization process based on the evolution of a large number of candidate solutions through genetic operations such as replication, cross over and mutation ([7]-[9]).

Recently Suykens and Mehrkanoon proposed approximating solutions to ODE's and PDE's using LSSVM's model. Also, they are extended them approaches in approximate solution to linear time varying descriptor systems ([13]-[14]).

In this paper, we introduce a new method based on SVM's for solving DE's. SVM's are very popular methods in machine learning which solving pattern recognition and function estimation problems ([10]-[12]). We will utilizing ϵ-LSSVM [15] method for solving ODEs. In last section will show that our results have better accuracy and loss error.

2. ϵ-LSSVM for Regression

We consider a given training set $\{x_i, y_i\}_{i=1}^{N}$ with input data $x_i \in \mathbb{R}$ and output data$y_i \in \mathbb{R}$. The regression problem is to estimate a model of the form $\bar{y}(t) = w.\phi(t) + b$. The primal problem of ϵ-LSSVM regression is follow:

$$\begin{aligned} &\min \ \frac{1}{2} \| w \|^2 + \frac{C}{2}\sum_{i=1}^{l}(\xi_i^2 + \eta_i^2) \\ &s.t. \ y_i - w.\phi(x_i) - b \le \epsilon + \xi_i, \\ &-y_i + w.\phi(x_i) + b \le \epsilon + \eta_i \end{aligned} \tag{3}$$

According to [15], the dual problem of (3) is

$$\begin{aligned} &\min \ \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\beta_i - \alpha_i)(\beta_j - \alpha_j)\left(K(x_i, x_j) + \frac{\delta_{ij}}{C}\right) + \epsilon\sum_{i=1}^{l}(\beta_i + \alpha_i) - \sum_{i=1}^{l} y_i(\beta_i - \alpha_i) \\ &s.t. \sum_{i=1}^{l}(\beta_i - \alpha_i) = 0, \\ &\beta_i \ge 0, \alpha_i \ge 0, i = 1,2,\dots,l \end{aligned} \tag{4}$$

where $K(x, y) = \phi(x).\phi(y)$ is a given positive definite function (kernel function). We use the following differential operators in next section which employed in [13]:

$$\nabla_n^m \equiv \frac{\partial^{n+m}}{\partial^n \partial^m},$$

$$[\phi^{(n)}(x)].\phi^{(m)}(y) = \nabla_n^m[\phi(x).\phi(y)] = \nabla_n^m[K(x, y)],$$

$$\nabla_1^0[K(x, y)] = \phi^{(1)}(x).\phi(y),$$

$$\nabla_0^1[K(x, y)] = \phi(x).\phi^{(1)}(y),$$

$$\nabla_0^0[K(x, y)] = \phi(x).\phi(y) = K(x, y),$$

$$\nabla_1^1[K(x, y)] = \phi^{(1)}(x).\phi^{(1)}(y).$$

3. Description of the Method

To obtain a solution of (1) the collocation method is adopted which assumes a discretization of the interval $[a, c]$ into a set of collocation points (training points)$\Delta = \{a = t_1 \le t_2 \le t_3 \le \cdots \le t_N = c\}$. Let $\hat{y}(t) = w.\phi(t) + b$ denote the approximate solution to (1), with adjustable parameters w and b, the problem (1) is transformed to the following quadratic programming problem:

$$\min \ \frac{1}{2} \| w \|^2 + \frac{C}{2}\sum_{i=2}^{N}(\xi_i^2 + \eta_i^2)$$

$$\text{s.t. } w(\phi^{(m)}(t_i) - \sum_{k=1}^{m} f_k(t_i)\phi^{(m-k)}(t_i)) - f_m(t_i)b - r(t_i) \le \epsilon + \xi_i, i = 2,3,\dots,N$$

$$-w(\phi^{(m)}(t_i) - \sum_{k=1}^{m} f_k(t_i)\phi^{(m-k)}(t_i)) + f_m(t_i)b + r(t_i)$$

$$\begin{aligned} &\leq \epsilon + \eta_i, i = 2,3, \dots, N \\ w.\phi(t_1) + b &= p_1 \\ w.\phi^{(i-1)}(t_1) &= p_i, i = 2,3, \dots, m \end{aligned} \tag{5}$$

In practice, the quadratic programming problem (5) is solved via its dual.

Lemma: The solution to (5) is obtained by solving the following the quadratic programming problem:

$$\begin{aligned} &\min \frac{1}{2} X^T H X + FX \\ &s.t. DX = 0, \\ &AX \leq 0 \end{aligned} \tag{6}$$

where

$$A = \begin{bmatrix} -I_{N-1\times N-1} & -I_{N-1\times N-1} & 0_{N-1\times m} \\ 0_{N-1\times N-1} & -I_{N-1\times N-1} & 0_{N-1\times m} \end{bmatrix}$$

$$D = [f_m(t)_{N-1\times 1} \quad 0_{N-1\times 1} \quad -e_{1_{m\times 1}}], F = [\epsilon e - r(t)_{N-1\times 1} \quad 2\epsilon e_{N-1\times 1} \quad -p_{m\times 1}]$$

$$H = \begin{bmatrix} \kappa_{N-1\times N-1} & 0_{N-1\times N-1} & \Omega_{N-1\times m} \\ 0_{N-1\times N-1} & 0_{N-1\times N-1} & 0_{N-1\times m} \\ \Omega_{m\times N-1} & 0_{m\times N-1} & \Delta_{m\times m} \end{bmatrix}, X = \begin{bmatrix} \lambda \\ \alpha \\ \gamma \end{bmatrix}$$

where, $e = (1,1, \dots,1)$ and $e_1 = (1,0, \dots,0)$.

Proof: The Lagrangian of the constrained optimization problem (5) becomes

$$\begin{aligned} L(w,b,\xi,\eta,\alpha,\beta,\gamma,\gamma_1) &= \frac{1}{2} \| w \|^2 + \frac{C}{2} \sum_{i=2}^{N} (\xi_i^2 + \eta_i^2) \\ &+ \sum_{i=2}^{N} \alpha_i \left(w(\phi^{(m)}(t_i) - \sum_{k=1}^{m} f_k(t_i) \phi^{(m-k)}(t_i) \right) - f_m(t_i) b - r(t_i) - \epsilon - \xi_i) \\ &+ \sum_{i=2}^{N} \beta_i \left(-w(\phi^{(m)}(t_i) - \sum_{k=1}^{m} f_k(t_i) \phi^{(m-k)}(t_i) \right) + f_m(t_i) b + r(t_i) - \epsilon - \eta_i) \\ &- \sum_{i=2}^{m} \gamma_i [w.\phi^{(i-1)}(t_1) - p_i] - \gamma_1 (w.\phi(t_1) + b - p_1) \end{aligned}$$

where α_i, β_i and γ_i for $i = 2,3, \dots, N$ are Lagrange multipliers and ξ_i and η_i are slack variables. Then, optimality conditions are as follows,

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=2}^{N} (\beta_i - \alpha_i)(\phi^{(m)}(t_i) - \sum_{k=1}^{m} f_k(t_i) \phi^{(m-k)}(t_i)) + \sum_{i=1}^{m} \gamma_i \phi^{(i-1)}(t_1)$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=2}^{N} (\beta_i - \alpha_i) f_m(t_i) - \gamma_1 = 0$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \xi_i = \frac{\alpha_i}{C}, i = 2, \dots, N$$

$$\frac{\partial L}{\partial \eta_i} = 0 \Rightarrow \eta_i = \frac{\beta_i}{C}, i = 2, \dots, N$$

We are imposing $\lambda_i = \beta_i - \alpha_i$ and then dual of (6) are as follows:

$$\min \frac{1}{2} [\sum_{i=2}^{N} \sum_{j=2}^{N} \lambda_i \lambda_j ([\nabla_m^m K](t_i, t_j) - \sum_{k=1}^{m} f_k(t_i) [\nabla_{m-k}^m K](t_i, t_j) - \sum_{k=1}^{m} f_k(t_j) [\nabla_m^{m-k} K](t_i, t_j) + \sum_{k=1}^{m} \sum_{k=1}^{m} [\nabla_{m-k}^{m-k} K](t_i, t_j))$$

$$+\sum_{i=1}^{m}\sum_{j=2}^{N}\gamma_i\lambda_j([\nabla_{j-1}^{m}K](t_1,t_j)-\sum_{k=1}^{m}f_k(t_j)[\nabla_{j-1}^{m-k}K](t_1,t_j))$$

$$+\sum_{i=2}^{N}\sum_{j=1}^{m}\lambda_i\gamma_j([\nabla_{m}^{j-1}K](t_i,t_1)-\sum_{k=1}^{m}f_k(t_i)[\nabla_{m-k}^{j-1}K](t_i,t_1))$$

$$+\sum_{i=1}^{m}\sum_{j=1}^{m}\gamma_i\gamma_j[\nabla_{i-1}^{j-1}K](t_1,t_1)]+\sum_{i=2}^{N}\lambda_i(\varepsilon e_i-r(t_i))+\sum_{i=2}^{N}2\varepsilon\alpha_i$$

$$s.t.\sum_{i=2}^{N}\lambda_i f_m(t_i)-\gamma_1=0, \tag{7}$$

$$\lambda_i+\alpha_i\geq 0,\alpha_i\geq 0,i=2,\dots,N$$

If using the following matrix form for problem (7), we can give the Dual problem (6),

$$\kappa_{i,j}=[\nabla_m^m K](t_i,t_j)-\sum_{k=1}^{m}f_k(t_i)[\nabla_{m-k}^{m}K](t_i,t_j)-\sum_{k=1}^{m}f_k(t_j)[\nabla_{m}^{m-k}K](t_i,t_j)$$

$$+\sum_{k=1}^{m}\sum_{k=1}^{m}f_k(t_i)f_k(t_j)[\nabla_{m-k}^{m-k}K](t_i,t_j),i,j=2,\dots,N$$

$$\Omega_{1,k}=[\nabla_{k-1}^{m}K](t_1,t_j)-\sum_{k=1}^{m}f_k(t_j)[\nabla_{k-1}^{m-k}K](t_1,t_j),j=2,3,\dots,N$$

$$\Delta_{ij}=[\nabla_{i-1}^{j-1}K](t_1,t_1),i,j=1,\dots,m.$$

The obtained approximation solution is as follows,

$$\hat{y}(t)=\sum_{i=2}^{N}\lambda_i\left([\nabla_m^0K](t_i,t)-\sum_{k=1}^{m}f_k(t_i)[\nabla_{m-k}^{0}K](t_i,t)\right)+\sum_{j=1}^{m}\gamma_j[\nabla_{j-1}^{0}K](t_1,t)+b$$

4. Numerical Results

In this section, we used the performance of the approach methods on two problems, first order and second order IVP. Therefore we solved two problems with ϵ-LSSVM model and then compared by SVM and LSSVM models [13] to shows that which of them are better. For all the problems, the RBF kernel is used ($K(x,y)=\exp(-\frac{(x-y)^2}{\sigma^2})$). Also, we use the set of midpoints of training points ($\frac{(t_i+t_{i+1})}{2},i=1,2,\dots,N-1$) as test point and compute the mean squared error (MSE) for the validation:

$$MSE(z_i)=y(z_i)-\hat{y}(z_i)$$

MATLAB 2015a is used to implement the code and all computational were carried out on a windows 8 system with Intel Pentium- ULV, 1.8 GHz CPU and4.00 GB RAM.

Example 1. Consider the following first order IVP [5]:

$$y'(t)+2y(t)=\sin(t),y(0)=1,t\in[0,10]$$

Exact solution of this example is

$$y(t)=\frac{6}{5}\exp(-2t)+\frac{2}{5}\sin(t)-\frac{1}{5}\cos(t)$$

The approximate solution by the proposed ϵ-LSSVM method are compared with the analytic solution; also we are compared numerical results of ϵ-LSSVM, SVM and LSSVM methods with together where are shown in Table 1. In this example, we suppose $\sigma=5$, $C=10^4$ and $\epsilon=10^{-5}$ for different amounts N. The results of this example shows that ϵ-LSSVM method for any N is the best method to relieve high accuracy and SVM model is not good; because, for $N>25$ give to us the high error amounts and is not suitable. Also in Fig. 1 the error function $y(t)-\hat{y}(t)$ for $N=50$ and 100 is depicted which shows the solution is very accurate

Table 1. *The comparison of our proposed method with SVM and LSSVM methods for example 1.*

N	Error	SVM	LSSVM	ϵ- LSSVM
25	MSE(Training)	1.18×10^{-5}	8.64×10^{-5}	7.81×10^{-7}
	MSE(Test)	1.57×10^{-4}	1.24×10^{-4}	1.36×10^{-6}
50	MSE(Training)	-	1.22×10^{-5}	4.60×10^{-10}
	MSE(Test)	-	1.33×10^{-5}	5.81×10^{-10}
75	MSE(Training)	-	5.08×10^{-6}	1.90×10^{-11}
	MSE(Test)	-	5.21×10^{-6}	2.14×10^{-11}
100	MSE(Training)	-	3.09×10^{-6}	4.30×10^{-12}
	MSE(Test)	-	3.12×10^{-6}	4.52×10^{-12}

Example 2. Consider the following second order IVP [3]:

$$y''(t)+\frac{1}{5}y'(t)+y(t)=\frac{-1}{5}\exp\left(-\frac{t}{5}\right)\cos(t), y(0)=0, y'(0)=1, t\in[0,5]$$

The exact solution of this example is

$$y(t)=\exp\left(-\frac{t}{5}\right)\sin(t)$$

Table 2. *The comparison of our proposed method with SVM and LSSVM methods for example2.*

N	Error	SVM	LSSVM	ϵ- LSSVM
20	MSE(Training)	9.86×10^{-5}	9.85×10^{-5}	9.82×10^{-5}
	MSE(Test)	1.01×10^{-4}	1.01×10^{-4}	1.00×10^{-4}
25	MSE(Training)	9.17×10^{-5}	1.03×10^{-4}	9.57×10^{-5}
	MSE(Test)	9.35×10^{-5}	1.05×10^{-4}	9.77×10^{-5}
50	MSE(Training)	-	1.16×10^{-7}	6.61×10^{-9}
	MSE(Test)	-	1.17×10^{-7}	6.68×10^{-9}
75	MSE(Training)	-	6.74×10^{-9}	1.37×10^{-10}
	MSE(Test)	-	6.78×10^{-9}	1.66×10^{-10}

The approximate solution by the proposed ϵ-LSSVM method are compared with the analytic solution; also we are compared numerical results of ϵ-LSSVM, SVM and LSSVM methods with together where are shown in Table 2. In this example, we suppose $\sigma=0.5$, $C=10^4$ and $\epsilon=10^{-5}$ for different amounts N. This Table shows that ϵ-LSSVM method for any N is the best method to relieve high accuracy and SVM model is not good; beacuse, for $N>25$ give to us the high error amounts and is not suitable. Also, error amounts for any N in LSSVM is losser ϵ-LSSVM. But, for $N=20$ results not continues an antiseptic process. In Figures 2 the error function $y(t)-\hat{y}(t)$ is depicted which shows the approximate solution convergence to exact solution. Also, In Figures 3 the error function $y'(t)-\widehat{y'}(t)$ is depicted which shows the approximate solution is differentiable.

5. Conclusion

In this paper a new method based on ϵ-LSSVM has been applied to find solutions for m-th order differential equations with initial conditions. The solution via ϵ-LSSVM method is a differentiable, closed analytic form easily used in any subsequent calculation. The neural network here allows us to obtain the solution of differential equations starting from training data sets and refined it without wasting memory space and therefore reducing the complexity of the problem. If we compare the results of the numerical methods with our methods, we see that our method has some small error. Other advantage of this method, the solution of differential equation is available for each arbitrary point in training interval (even between training points).

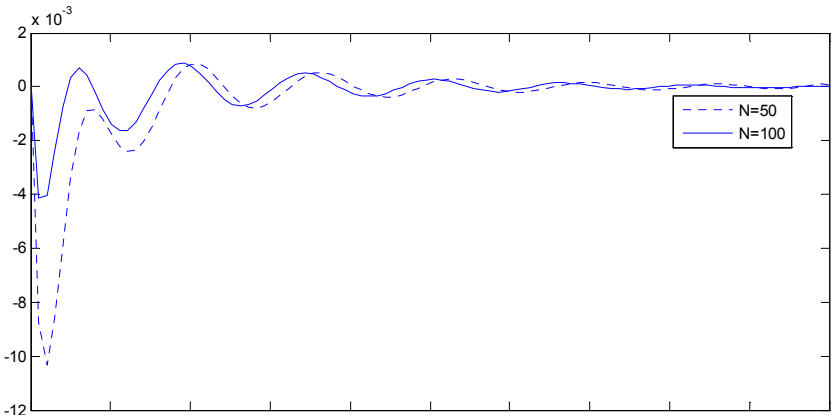


Fig. 1. *The error function $y(t)-\hat{y}(t)$ for example1 when [0; 10] is discretized into 50 and 100 equal parts.*

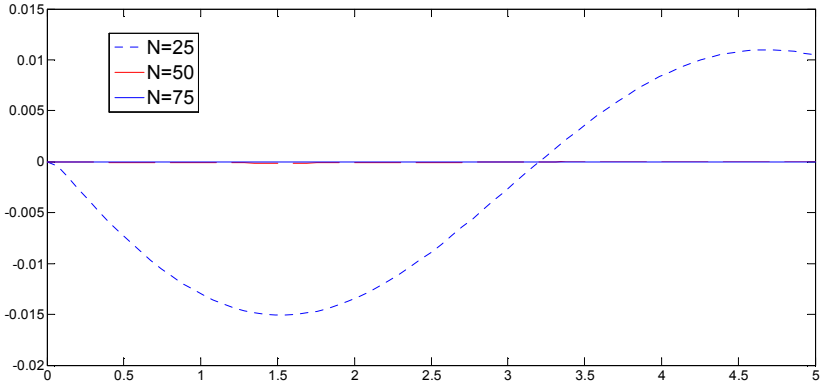


Fig. 2. *The error function $y(t)-\hat{y}(t)$ for example 2 when [0; 5] is discretized into 25, 50 and 75 equal parts.*

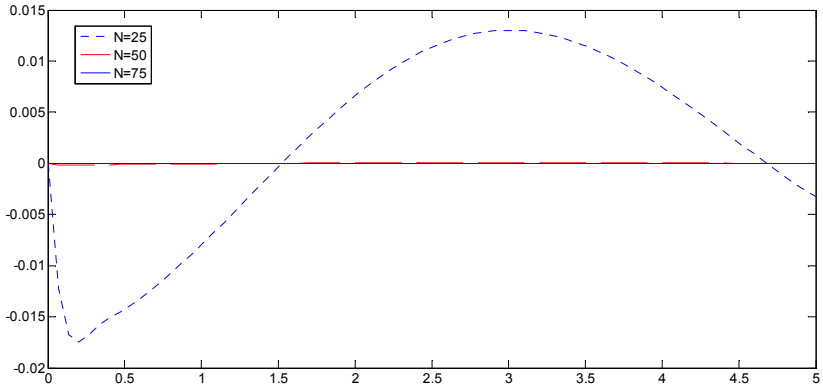


Fig. 3. *The error function* $y'(t) - \widehat{y}'(t)$ *for example2 when [0; 5] is discretized into 25, 50 and 75 equal parts.*

Acknowledgements

This work was supported by Quchan University of Advanced Technology under grant number 1024.

References

[1] S. D. Conte, C. de Boor, "Elementary numerical analysis, an algorithmic approach", Third Edition, 1980.

[2] D. R. Kincaid, E. W. Cheney, "Numerical analysis mathematics of scientific computing",third ed., Brooks/Cole, Pasific Grove CA, 2002.

[3] I. L. Lagaris, A. Likas, D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations",IEEE Transactions on Neural Networks, 9(5), 987-1000(1998).

[4] A. Malek, R. Shekari Beidokhti, Numerical solution for high order differential equations using a hybrid neural network optimization method, Applied Mathematics and Computation 183, 260-271(2006).

[5] H. S. Yazdi, M. Pakdaman, H. Modaghegh, Unsupervised kernel least square algorithm for solving Ordinary Differential Equations, Neurocomputing 74, 2062-2071, (2011).

[6] G. E. Fasshauer, Solving Differential Equations with Radial Basis Functions: Multilevel Methods and Smoothing, Advances in Computational Mathematics, 11(2), 139-159(1999).

[7] G. Burgess, Finding Approximate Analytic Solutions to Differential Equations Using Genetic Programming, Surveillance Systems Division, Electronics and Surveillance Research Laboratory, Department of Defense, Australia, 1999.

[8] I. G. Tsulos, I. E. Lagaris, Solving differential equations with genetic programming", Genetic Programming Evolable Machines, 7, 33-54(2006).

[9] J. R. Koza, Genetic Programming: On the programming of Computer by Means of Natural Selection. MIT Press: Cambridge, MA, 1992.

[10] V. Vapnik, "Statistical learning theory", New York: Wiley (1998).

[11] V. Vapnik, "The natural of statistical learning theory", Springer New York (1995).

[12] Y. Xu, X. Pan, Structural least square twin support vector machine for classification, Springer Science and Business Media New York, 42, 527-536(2015).

[13] S. Mehrkanoon, T. Flack and J. A. K. Suykens, Approximate solutions to ordinary differential equations using least squares support vector machines, IEEE Transactions on Neural Networks and Learning Systems 23, 1356-1367(2012).

[14] S. Mehrkanoon, J. A. K. Suykens, LS-SVM approximate solution to linear time varying descriptor systems", Automatica, 48, 2502-2511(2012).

[15] Y. J. Tian, X. C. Ju, Z. Qi and Y. Shi, Efficient sparse least squares support vector machines for pattern classification, Computers and Mathematics with Applications 28(6), (2013).

[16] J. A. K. Suykens and J. Vandewalle, Least squares support vector machine classifier, Neural Processing Letters 9(3), 293-300(1999).

[17] D. Tax and R. Duin, Support vector domain description, Pattern Recognition Letter 20, 1191-1199(1999).

[18] D. Tax and R. Duin, Support vector data description", Machine Learning 54, 45-66 (2004).