Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. 6, No. 3, 2018, pp. 312-325

Solving optimal control problems by PSO-SVM

Elham Salehpour*
Department of Mathematics, Nowshahr branch,
Islamic Azad university, Nowshahr, Iran.
E-mail: salehpour.e61@gmail.com

Javad Vahidi
Iran University of Science and Technology,
Information Technology Faculty, Tehran, Iran.
E-mail: jvahidi@iust.ac.ir

Hssan Hosseinzadeh
Department of Mathematics,
University of Mazandaran, Babolsar, Iran.
E-mail: hossinzadeh.h@umz.ac.ir

Abstract The optimal control of problem is about finding a control law for a given system such that a certain optimality criterion is achieved. Methods of solving the optimal control problems are divided into direct methods and mediated methods (through other equations). In this paper, the PSO- SVM indirect method is used to solve a class of optimal control problems. In this paper, we try to determine the appropriate algorithm to improve our answers to problems.

Keywords. Particle Swarm Optimization, Support vector machines, Optimal control.
2010 Mathematics Subject Classification. 65L05, 34K06, 34K28.

1. Introduction

Theory of particle swarm optimization (PSO) has been growing rapidly. PSO has been used by many applications of several problems. The algorithm of PSO emulates from behavior of animals societies that don't have any leader in their group or swarm, such as bird flocking and fish schooling. Typically, a flock of animals that has no leader finds food randomly by following one of the members of the group that has the closest position with a food source (potential solution). The flocks achieve their best condition simultaneously through communication among members who already have a better situation. Animal which has a better condition will inform it to its flock and the others will move simultaneously to that place. This would happen repeatedly until the best conditions or a food source discovered. The process of PSO algorithm in finding optimal values follows the work of this animal society. Particle swarm optimization consists of a swarm of particles, where particle represent a potential solution. Recently, there have been several modifications from original PSO.

Received: 13 January 2017 ; Accepted: 12 May 2018.
∗ Corresponding author.

It was modified to accelerate the achieving of the best conditions. The development will provide new advantages and also the diversity of problems to be resolved. It is necessary to conduct studies on the development of PSO in order to recognize its development, advantages and disadvantages and the usefulness of this method to settle a problem. The theoretical tutorial of PSO is described in [1, 2, 3, 4] testing the data is simpler, and has a simpler structure in comparison with other evolutionary computations.

Recently, Support Vector Machines (SVMs) were introduced by [14, 19] for solving classification and nonlinear function estimation problems [6, 8, 9, 13]. Within this new approach the training problem is reformulated and represented in a way to obtain a (convex) quadratic programming (QP) problem. The solution to this QP problem is global and unique. In SVMs, it is possible to choose several types of kernel functions including linear, polynomial, RBFs, MLPs with one hidden layer and splines, as long as the Mercer condition is satisfied. Furthermore, bounds on the generalization error are available from statistical learning theory [7, 17, 18] which is expressed in terms of the VC (Vapnik- Chervonenkis) dimension. An upper bound on this VC dimension can be computed by solving another QP problem.

In this article, we mix SVM and PSO and with a new algorithm we study the problem of Narendra and Mukhopadhyay in 1997 which was solved by Suykens and his colleagues by LS-SVM [16]. Consider the following nonlinear system where $x_{1,k}$ and $x_{2,k}$ are the state variables and u_k is the control variable:

$$\begin{cases} x_{1,k+1} = 0.1x_{1,k} + 2\left(\frac{u_k + x_{2,k}}{1+(u_k+x_{2,k})^2}\right) \\ x_{2,k+1} = 0.1x_{2,k} + u_k\left(2 + \frac{u_k^2}{1+x_{1,k}^2+x_{2,k}^2}\right) \end{cases} \tag{1.1}$$

We are considering a new method in this paper to give a better answer.
This paper is organized as follows:
In section 2, the N-stage optimal control problem is presented.
In section 3, some works on support vector machines are reviewed.
In section 4, PSO algorithm is studied.
In section 5, optimal control by support vector machines is studied.
In section 6, combining particle swarm algorithm with support vector machine is discussed.
In section 7, an example is presented.

2. The N-stage optimal control problem

In the N-stage optimal control problem is given one aims at solving the following problem is given [10]

$$\begin{aligned} \min: \quad & \Psi_M(x_i, u_i) = \tau(x_{M+1}) + \sum_{i=1}^{M} f(x_i, u_i) \\ s.t \quad & x_{i+1} = \varphi(x_i, u_i), \qquad i = 1, \ldots, M. \end{aligned} \tag{2.1}$$

where τ and f are positive definite functions. We will consider the quadratic cost

$$f(x_i, u_i) = x_i^T P x_i + u_i^T Y u_i, \qquad \tau(x_{M+1}) = x_{M+1}^T P x_{M+1} \tag{2.2}$$

with $P = P^T > 0$, $Y = Y^T > 0$.

The functions τ, f and φ are twice continuously differentiable. $x_i \in \mathbb{R}^n$ denotes the state vector, $u_i \in \mathbb{R}$ is the input of the system. This article is not limited to single-input systems.

In order to find the optimal control law, one constructs the Lagrangian

$$\mathcal{H}(x_i, u_i, \lambda_i) = \Psi_M(x_i, u_i) + \sum_{i=1}^{M} \lambda_i^T [x_{i+1} - \varphi(x_i + u_i)] \tag{2.3}$$

where Lagrange multipliers $\lambda_i \in \mathbb{R}^n$. The conditions for optimality are given [13]

$$\begin{cases} \frac{\partial \mathcal{H}_M}{\partial x_{M+1}} = \frac{\partial f}{\partial x_i} + \lambda_{i-1} - \left(\frac{\partial \varphi}{\partial x_i}\right)^T \lambda_i = 0, & i = 2, \dots M \\ \frac{\partial \mathcal{H}_M}{\partial x_{M+1}} = \frac{\partial r}{\partial x_{M+1}} + \lambda_M = 0, & \\ \frac{\partial \mathcal{H}_M}{\partial u_i} = \frac{\partial f}{\partial u_i} - \lambda_i^T \frac{\partial \varphi}{\partial u_i} = 0, & i = 1, \dots M \\ \frac{\partial \mathcal{H}_M}{\partial \lambda_i} = x_{i+1} - \varphi(x_i, u_i), & i = 1, \dots M \end{cases} \tag{2.4}$$

For the case of a quadratic cost function subject to linear system dynamics with infinite time horizon, the optimal control law can be represented by full static state feedback control. However, in general, the optimal control law cannot be represented by state feedback as the optimal control law may also depend on the co-state. Nevertheless, one is often interested in finding a suboptimal control strategy of this form. In the context of neural control [11, 12, 15].

$$u_i = k(x_i), \tag{2.5}$$

where k is a parametrized neural network architecture. In this paper, we will also consider a control law given in Eq.(2.5), to support vector machines. Because SVM methodology is not a parametric modeling approach, this is less straightforward in comparison with standard neural networks such as MLP's and RBF's.

3. The support vector machine

SVM algorithm was presented early in 1963 by Vladimir Vapnik. In 1995, he and his colleague extended the non-linear mode. SVM has very valuable properties which makes it suitable for pattern recognition.

SVM does not have local optimization problem in training. The category is built with maximum extension. The structure and topology are defined to optimize and the form of differential non-linear functions are calculated easily by using the concept of Hilbert spaces' domestic multiplication. This is a supervised learning method used for classification and regression. This is a relatively new method. Compared to the older methods, the new method has been shown to have a good performance in recent years.

3.1. **Reasons to use SVM.**

With a little precision in training algorithm, it seems that one of the problems of such systems is on the one hand the large number of samples for training and on the other hand the increased number of features to express the sample or in other words the increased dimension. So for having the desired output, we need a classifier which can support a large set of training data with a lot

of dimensions. Large dimensions in a classification create a lot of parameters (when dimensions are increased, the covariance matrix becomes larger and more parameters must be calculated) that is difficult to estimate.

3.2. **The users of SVM.** The SVM algorithm is classified as the pattern recognition algorithm. SVM algorithm can be used wherever detecting patterns or categories of objects in specific classes is needed including risk analysis system, control plane without a pilot, track deviation plane, route simulation, automatic guidance system for cars, quality in section systems, analysis of welding quality, quality forecast, quality analyzes computer, analysis of mill operations, chemical analysis of product design, analysis of maintenance, proposal, management and planning, chemical process, dynamic control system, the design of artificial limbs, optimizing the time of organ transplantation, reducing hospital costs, improving the quality of hospitals, testing the emergency room, oil and gas exploration, automatic route control devices, robots, cranes, visual systems, voice recognition, concisely speech, classifieds sound, market analysis, consulting systems, calculating the cost of inventory, concise information and images, automatic information services, customer payment processing systems, truck brake detection systems, vehicle scheduling, routing systems, classifieds charts customer/market, drug recognition, signature verification, the loan risk estimation, spectral identification, investment appraise land so on [14, 15, 16].

3.3. **Advantages and disadvantages.**

1: Designing categories with maximum extension
2: Reaching the global optimal of cost function
3: Automatic determination structure and optimal topology of classifier
4: Modeling the non-linear differentiation function by using non-linear kernel and concept of domestic multiplication in Hilbert spaces
5: Training is relatively simple
6: Unlike neural networks, it does not exist in local maximum
7: For high-dimensional data almost works good
8: Reconciliation between the complexity of product categories and the error is clearly controlled
9: Needs a good kernel function and select parameter C.

3.4. **Support vector machine by linear classifier with linearly separable data:**[14]**.** Suppose that we have the number of feature vectors or training patterns $\{x_1, \ldots, x_M\}$ and each of them is the m-dimensional feature vector and had labeled y_i, $y_i \in \{-1, +1\}$. The purpose of solving optimization problem is in two classes. Suppose that we separate two classes with the differentiate function $f(X)$ and space H with the following equation.

$$\begin{aligned} H &: wx + b = 0 \\ f(x) &= w^T x + b \end{aligned} \tag{3.1}$$

If two examples of two classes with the names of $(+1)$ and (-1) are expressed, these spaces can be considered as separate positive samples from negative samples.

So the problem becomes minimized for $\frac{1}{2}w^T w = \frac{1}{2}\|w\|^2$.

So actually design is a hyper plane classification with border optimal which will be as follows:

$$\begin{aligned} \min \quad & \frac{1}{2}\|w\|^2 \\ s.t \quad & y_i(WX_i + b) \geq 1, \qquad i = 1, 2, \ldots M. \end{aligned} \tag{3.2}$$

Clearly, we have:$w = [w_1, w_2, \ldots, w_m]^T$ and $\|w\|^2 = w^T w$ The above problem can be solved by quadratic programming (QP) optimization techniques. To solve it, this problem is written in the form of the following Lagrangian function and the Lagrange multipliers are obtained.

$$L(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{M} \alpha_i \{y_i(w^T x_i + b) - 1\}. \tag{3.3}$$

The problem should be minimized in relation to b, w and α_i variables should be maximized. We should solve this problem by eliminating w, b variables.

So the problem is a maximization problem. It will be based on α_i. we solve this problem. To be $L(w, b, \alpha)$ the answer of the question, this answer must be true in KKT conditions and in the parts of the answer, L derive in relation to (α, b, w) is equal to. By equating the derivative to zero, we get the following equation:

$$\begin{cases} \frac{L(w,b,\alpha)}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{M} \alpha_i y_i x_i, \\ \frac{L(w,b,\alpha)}{\partial b} = 0 \rightarrow w = \sum_{i=1}^{M} \alpha_i y_i = 0. \end{cases} \tag{3.4}$$

By replacing w in the eq.(3.3), dual optimization issue will occur.

This formula of SVM is called difficult margin SVM. Because conducts classification completely and without any breach.

After solving the dual optimization issue, we reach to the Lagrangian coefficients. In fact, each of the coefficients α_i is corresponding to one of the patterns x_i that corresponds to $\alpha_i > 0$ coefficients (positive) and are called support vectors sv_i. [19]

The weight vector and b are obtained from the following relationship:

$$\begin{cases} w = \sum_{i=1}^{M_{sv}} \alpha_i y_i sv_i, \\ b_j = y_j - \sum_{i=1}^{M_{sv}} \alpha_i y_i sv_i sv_j, \\ b = \frac{1}{M_{sv}} \sum_{j=1}^{M_{sv}} b_j. \end{cases}$$

Distinction function will be classified by an input x pattern which is as follows:

$$f(X) = sign\left(\sum_{i=1}^{M_{sv}} \alpha_i y_i X sv_i + b\right). \tag{3.5}$$

The linear classifier support vector machine with linearly in separable data (integral mode):

In the previous section, we assumed a linear separation training data. When data training does not mean linear separation, it enters into a different class. In fact, there was an error, there is no feasible solution. Here we apply the extension of support

vector machine for the inseparable state. Now, we consider the case that we are still looking for a linear separating hyper plane with the difference that the separating hyper plane does not fully apply in inequality. To check this state, we define a new variable called Deficiency variable ε_i (slack variable) which represented the deviation from inequality (3.2). So the purpose of the algorithm is the maximum margin to define the payment of the cost appropriate to the deviation of inequality (3.2). It is clear that as long as the total amount of variables increases, the more error will occur which is far from optimal. The mathematical expression is in the form of the following:

$$\begin{aligned} \min: \quad & Q(w,b,\varepsilon) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{M}\varepsilon_i^p, \\ s.t \quad & y_i(w^T x_i + b) \geq 1 - \varepsilon_i, \qquad \varepsilon_i \geq 0, \qquad i = 1,\ldots,M, \end{aligned} \tag{3.6}$$

where $\varepsilon = (\varepsilon_1,\ldots,\varepsilon_M)^T$ and c are margin parameters that determine the difference between the maximum margin and minimum error the classification.

To solve the equation, Lagrange multipliers are used

$$\begin{cases} Q(w,b,\varepsilon,\alpha,\beta) = \frac{\|w\|^2}{2} + C\sum_{i=1}^{M}\varepsilon_i - \alpha_i\left(y_i(w^T x_i + b) - 1 + \varepsilon_i\right) - \beta_i\varepsilon_i, \\ \frac{\partial Q(w,b,\varepsilon,\alpha,\beta)}{\partial w} = 0, \\ \frac{\partial Q(w,b,\varepsilon,\alpha,\beta)}{\partial b} = 0, \\ \frac{\partial Q(w,b,\varepsilon,\alpha,\beta)}{\partial \varepsilon} = 0, \\ \alpha_i\left(y_i(w^T x_i + b) - 1 + \varepsilon_i\right), \qquad i = 1,\ldots M, \\ \beta_i\varepsilon_i = 0, \qquad i = 1,\ldots M, \\ \alpha_i \geq 0, \beta_i \geq 0, \varepsilon_i \geq 0, \qquad i = 1,\ldots M. \end{cases}$$

Respectively, we take partial derivatives in relation to w, b, ε.

$$\begin{cases} w = \sum_{i=1}^{M}\alpha_i y_i x_i, \\ \sum_{i=1}^{M}\alpha_i y_i = 0, \\ \alpha_i + \beta_i = C, \qquad i = 1,\ldots,M. \end{cases}$$

Replace and dual is written as follows:

$$\begin{aligned} \max \quad & Q(\alpha) = \sum_{i=1}^{M}\alpha_i - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\alpha_i\alpha_j y_i y_j x_i^T x_j \\ s.t \quad & \sum_{i=1}^{M}\alpha_i y_i = 0; \qquad 0 \leq \alpha_i \leq C, \qquad i = 1,\ldots,M. \end{aligned} \tag{3.7}$$

This is the results that we mentioned below [20]

1: If $\alpha_i = 0$ then $\varepsilon_i = 0$ in this case x_i categorized.
2: If $0 < \alpha_i < C$ then $y_i(w^T x_i + b) - 1 + \varepsilon_i = 0$ and $\varepsilon_i = 0$ in this case $y_i(w^T x_i + b) = 1$ and x_i supportive vector. The support vectors with $0 < \alpha_i < C$ is called a support vector boundless.
3: If $\alpha_i = C$ then $y_i(w^T x_i + b) - 1 + \varepsilon_i = 0$ and $\varepsilon_i = 0$. So x_i is a support vector. The support vectors with $\alpha_i = C$ $\varepsilon_i \geq 0$, x_i is classified and if $\varepsilon_i \geq 0$, is a typed text or a website address or a translated document. It will not be classified.

4. Particle swarm optimization algorithm

Particle swarm optimization algorithm or PSO, also known as the swarm, is one of the most powerful and popular algorithms for optimization. This is mostly because the relatively high rate of convergence is used. These algorithms are a little old, but were successful in many application areas, older algorithms, such as genetic algorithms, surpass and were considered as the first choice.

PSO is one of the optimization methods inspired by nature which has been developed for solving numerical optimization with very large search space without having to inform the gradient of the objective function. This was invented the first time in 1995 by two people named Kennedy and Eberhart. At the beginning, it was used to simulate the mass flight of birds. However, after the initial simple algorithm it was observed it was actually a type of optimization algorithm and for this reasons, it can also be used to solve other optimization problems.

The algorithm is inspired by the lives of a group of animals, including insects (such as ants, bees, etc.), birds and fish. To solve an optimization problem, a population of candidate solutions moves accidentally in the domain of problem using a simple formula, and explores it with aim of finding the global optimal answers. In PSO algorithm, each of these candidate answers is called a particle, and each particle flies with one of the birds in a flock as a corresponding answer. PSO algorithm is similar to genetic algorithms [1, 3].

PSO algorithm is similar to genetic algorithm in the sense that a population of solutions is randomly generated by the algorithm and is looking for the answer by moving through the problem domain. However, unlike genetic algorithm, in the PSO algorithm a random velocity is assigned to each potential answer of optimization problem (in fact to each particle) in a way that in each repetition, each particle moves (or idiomatically flies) based on its velocity in the problem domain. Also, unlike genetic algorithm, in PSO algorithm, the best obtained answer for optimization problem must be stored by each of the particles (from the beginning of the program until the last repetition). Like genetic algorithm, PSO algorithm is also inherently suitable for solving the unconstrained maximization in continuous mode.

However, by making changes in the definition method of the objective function, it can be used for solving the optimization problems (including minimization or maximization) in constrained (continuous) mode. PSO algorithm does not need any combination of practical information from optimized function and only uses basic

mathematical operators in order to adjust it to the minimum parameters required. In this method, minimum parameters are needed for adjustment. Furthermore, the executive function of algorithm will not be lost when the dimensions of research space are developed. PSO method is one of the new species of evolutionary methods whose application potential in optimization problems with continuous functions has been proved. In this way, move toward the optimal point, based on two data sets is done. One of the best-point of information obtained from each of the initial population [2, 3].

An explanation on the PSO algorithm:

First, in the search space, a number of points are selected as the initial population. Points in different categories are based on Euclidean distance. For example i category includes three searching factors. The function value of each available factor in the search space is calculated and in each category, depending on the intended purpose, it is determined in which spot the value is minimized or maximized. Therefore, the best spot is identified for each category.

On the other hand, by accessing the previous information, each factor is able to locate the best spot that has been discovered so far. Thus, the optimal point information of each category and agent is specified. The first knowledge concerns the global optimal point in each group and the second knowledge concerns the local optimal point.

With this information, the motion vector is given to each factor. This method is not affected much by the size and nonlinear problem and has good results in static environments, noise and environments which are continuously changing. Simplicity of implementation, lack of commitment to the continuity of the objective function and the ability to adapt to a dynamic environment makes the algorithm useful in many different areas. Accordingly, it can be concluded that the targeted nature of the behavior of particles in PSO method is based on two principles [4].

These two principles are:

i . Individual knowledge: Each individual moves to the best of his knowledge to gain new knowledge.

ii . Social science: The person in terms of his relationship with the community uses the best information for continuing the movement.

In Article ii, Individual relationship with the community is important to the topology structure of the community, based on this, different topologies are defined for the society. This algorithm has advantages and disadvantages which are mentioned below [3, 4].

Advantages algorithm include the following:

1: This algorithm is compared to other less regulated parameters optimized algorithms.
2: Implementation is easy and it has simple concepts.
3: It can be used effectively for various issues.
4: It is also used for discrete states and the continuum concepts.
5: The algorithm's performance will not disappear with the growth of research space.

6: The above algorithm optimization function application does not require any combination of information and only uses basic math operators.
7: It does not require heavy mathematical operations such as gradients.
8: It is a population-based approach.
9: The partnership uses particles.

According to mathematic law, in order to search for new solutions, the particle swarm optimization is valued randomly in search and movement space through D space. Assume that x_k^i and ν_k^i are respectively situation and particle speed of i in search space in k repetition then speed and situation of these particles will be updated in repetition $(k+1)$ therefore, we will have following equations:

$$\begin{cases} v_{i+1}^T = w.v_k^T + c_1.r_1(p_k^i - x_k^i) + c_2.r_2.(p_k^g - x_k^i), \\ x_{k+1}^i = x_k^i + v_{k+1}^i. \end{cases}$$

where r_1 and r_2 are the random number between 0 and 1 and c_1 and c_2 are fixed, p_k^i shows the best situation of i particle and p_k^g relates to the best situation in the swap to k repetition.

One of the principle steps of particle swap optimization can be summarized as a pseudo code in algorithm 1.

Algorithm 1: Pseudo code of particle swarm optimization (PSO).

1: Objective function: $f(x), x = (x_1, x_2, , x_D)$;
2: Initialize particle position and velocity for each particle and set $k = 1$;
3: Initialize the particle's best known position
to its initial position i.e. $P_k^i = X_k^i$; 4: do;
5: Update the best known position (P_k^i) of each particle
and swarm's best known position (P_k^g);
6: Calculate particle velocity according to the velocity equation;
7: Update particle position according to the position equation;
8: While maximum iterations or minimum error criteria is not attained.

5. Optimal control by support vector machines

Now, we are concerned with putting the training data $\{x_i, y_i\}_{i=1}^p$ in equation to the state space and action space in Eq. (2.1), $\{x_i, u_i\}_{i=1}^M$. We state the following N-stage optimal control problem:

$$\begin{aligned} \min \quad & \Psi(x_i, u_i, w, e_i) = \Psi_N(x_i, u_i) + \frac{1}{2} w^T w + \frac{1}{2} \sum_{i=1}^{M} e_i^2, \\ s.t \quad & x_{i+1} = \varphi(x_i, u_i), \qquad i = 1, \ldots, M, \end{aligned} \tag{5.1}$$

and the control law

$$u_i = w^T g(x_i) + e_i, \qquad i = 1, \ldots, M, \tag{5.2}$$

where Ψ_M is defined in Eq. (2.1), $w \in \mathbb{R}^{n_f}$, $f, g : \mathbb{R} \to \mathbb{R}^{n_f}$ and n_f is the number of hidden units of g. The actual control signal applied to be $w^T g(x_i)$. In the linear control case has $g(x_i) = x_i$ with $n_f = n$.

In the following, we will use RBF kernels. In the support vector method, applying the original Mercer's condition is essential such that one does not have to construct g. Although the use of this kernel function will be proposed, it can assess g explicitly, similar to the multilayer perceptron classifiers which is shown [9]. For the RBF case, Eq. (5.2) converts then

$$u_i = \sum_{r=1}^{n_f} w_r \exp\left(-\frac{1}{\beta^2}\|x_i - a_r\|_2^2\right) + e_i, \tag{5.3}$$

where $a_r \in \mathbb{R}^n$ selection centers and β selection is constant. For example the set $\{a_r\}_{r=1}^{n_f}$ equal to $\{x_i\}_{i=1}^{M}$. In standard SVM theory for static function estimation problems β can be selected so as to minimize an upper bound on the generalization error. These bounds are not applicable in the context of SVM control due to the fact that the input patterns to the activation function are not independent from each other. Hence β should be chosen as hoc or could be taken as an additional unknown within the cost function.

In order to find the optimal control law we construct the Lagrangian

$$\begin{aligned}\Psi(x_i, u_i, w, e_i, \lambda_i, \gamma_i) &= \Psi_M(x_i, u_i) + \frac{1}{2}w^T w + \\ &\quad \frac{1}{2}\sum_{i=1}^{M} e_i^2 + \sum_{i=1}^{M} \lambda_i^T [u_i - w^T g(x_i) - e_i].\end{aligned}$$

The conditions for optimality are

$$\begin{cases} \frac{\partial \mathcal{H}}{\partial x_i} = \frac{\partial f}{\partial x_i} + \lambda_{i-1} - \left(\frac{\partial \varphi}{\partial x_i}\right)^T \lambda_i - \gamma_i \frac{\partial}{\partial x_i}[w^T g(x_i)] = 0, & i = 2, \ldots, M, \\ \frac{\partial \mathcal{H}}{\partial x_{M+1}} = \frac{\partial r}{\partial x_{M+1}} + \lambda_M = 0, & \\ \frac{\partial \mathcal{H}}{\partial u_i} = \frac{\partial f}{\partial u_i} - \lambda_i^T \frac{\partial \varphi}{\partial u_i} + \gamma_i = 0, & i = 1, \ldots, M, \\ \frac{\partial \mathcal{H}}{\partial w} = w - \sum_{i=1}^{M} \gamma_i g(x_i) = 0, & \\ \frac{\partial \mathcal{H}}{\partial e_i} = \alpha e_i - \gamma_i = 0, & i = 1, \ldots, M, \\ \frac{\partial \mathcal{H}}{\partial \lambda_i} = x_{i+1} - \varphi(x_i, u_i), & i = 1, \ldots, M, \\ \frac{\partial \mathcal{H}}{\partial \gamma_i} = u_i - w^T g(x_i) - e_i = 0, & i = 1, \ldots, M. \end{cases}$$

For RBF kernels

$$\frac{\partial}{\partial x_i}[w^T g(x_i)] = -2\sum_r w_r \exp(-\frac{1}{\beta^2}\|x_i - a_r\|_2^2)\frac{x_i - a_r}{\beta^2}. \tag{5.4}$$

The set of nonlinear

$$G_1 = (x_i, x_{M+1}, u_i, w, e_i, \lambda_i, \gamma_i) = 0, \qquad i = 1, , \ldots, M. \tag{5.5}$$

With given x_1 and can numerically solve the unknown variables. In Appendix A a formulation without e_i variables is given. In comparison with function estimation or classification problems, one looses the advantage of solving a quadratic program or a linear least squares problem. Nevertheless, one is still able to exploit some of the interesting SVM features.

As in SVM theory, Mercer's condition can be replaced $w = \sum_i \gamma_i g(x_i)$. in the equations.

For kernels satisfying Mercer's condition

$$K(x_i, x_j) = g(x_i)^T g(x_j). \tag{5.6}$$

For RBF kernels [5, 7, 17]

$$K(x_i, x_t) \exp(-\vartheta \|x_i - x_t\|_2^2), \tag{5.7}$$

ν is a positive real constant

$$G_2 = (x_i, x_{M+1}, u_i, \lambda_i, \gamma_i) = 0, \qquad i = 1, , \ldots, M. \tag{5.8}$$

By Mercer's condition have [6]

$$\begin{cases} \frac{\partial f}{\partial x_i} + \lambda_{i-1} - (\frac{\partial \varphi}{\partial x_i})^T \lambda_i - \gamma_i \sum_{t=1}^{M} \gamma_t \frac{\partial K(x_i, x_t}{\partial x_i} = 0, & i = 2, \ldots, M, \\ \frac{\partial r}{\partial x_{M+1}} + \lambda_M = 0, & \\ \frac{\partial f}{\partial u_i} - \lambda_i^T \frac{\partial \varphi}{\partial u_i} + \gamma_i = 0, & i = 1, \ldots, M, \\ x_{i+1} - \varphi(x_i, u_i) = 0, & i = 1, \ldots, M, \\ u_i - \sum_{t=1}^{M} \alpha_t K(x_t, x_i) - \frac{\gamma_i}{\alpha} = 0, & i = 1, \ldots, M. \end{cases} \tag{5.9}$$

For RBF kernels

$$\frac{\partial K(x_i, x_t)}{\partial x_i} = -2\vartheta(x_i - x_t) \exp(-\vartheta \|x_i - x_t|_2^2). \tag{5.10}$$

The actual control signal applied to the plant becomes

$$u_i = \sum_{t=1}^{M} \gamma_t K(x_t, x_i), \tag{5.11}$$

where $\{x_t\}_{t=1}^{M}$ and $\{\gamma_t\}_{t=1}^{M}$ are obtained from solving the set of nonlinear Eq. (5.9) and x_i is the actual state vector at time i. The data $\{x_t\}_{t=1}^{M}$ are used as support vector data for the control signal. Furthermore, note that e_i was considered equal to zero in Eq. (5.9).

6. Combine PSO with SVM

According to the algorithm PSO, each particle has two elements: factors position and the objective function. In the proposed algorithm, the position of each particle is the starting point. We're looking for an optimal position on the issue. To obtain value of objective function using the combination method of optimal control and support vector machine, the unknown factors and coefficients are calculated. Then by replacing it in the objective function, the costs are calculated. With the help of algorithm PSO, these steps are calculated for each particle, and finally the optimum starting point is achieved. As a result, the optimal starting point is placed in combination of SVM and optimal control, Finally, we obtain the optimal starting point. To solve the problem of optimal control using SVM, numerical methods using FMINCON function are used in MATLAB software.

Algorithm 2: Combine PSO with SVM.

1: Objective function: Optimal Control By LSSVM, $x = x_1, x_2, \ldots, x_D$;
2: Initialize particle position and velocity for each particle and set $k = 1$.
3: Initialize the particle's best known position
to its initial position i.e. $P_k^i = X_k^i$.
4: Do
5: Update the best known position P_k^i of each particle and
swarm's best known position P_k^g.
6: Calculate particle velocity according to the velocity equation.
7: Update particle position according to the position equation.
8: While maximum iterations or minimum error criteria is not attained.

7. Simulation example

Consider the following nonlinear system: [16]

$$\begin{cases} x_{1,i+1} = 0.1x_{1,i} + 2\left(\frac{u_i + x_{2,i}}{1+(u_i+x_{2,i})^2}\right), \\ x_{2,i+1} = 0.1x_{2,i} + u_i\left(2 + \frac{u_i^2}{1+x_{1,i}^2+x_{2,i}^2}\right). \end{cases} \tag{7.1}$$

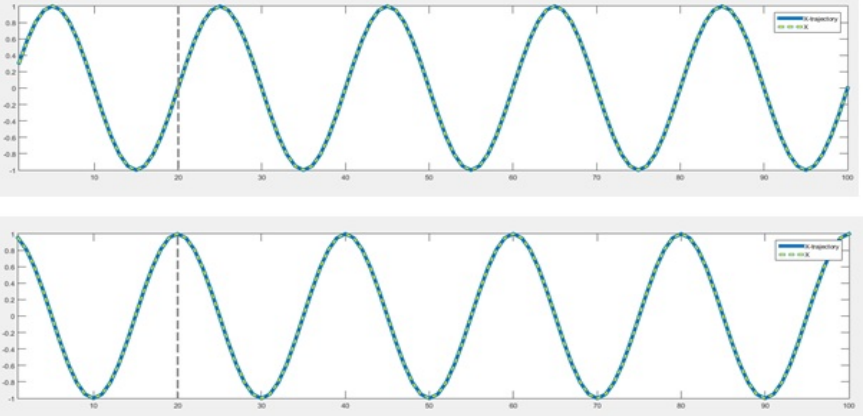
We consider a state vector tracking problem with

$$\begin{cases} f(x_i, u_i) = (x_i - x_i^r)^T P(x_i - x_i^r) + u_i^T Y u_i, \\ \gamma(x_{N+1}) = (x_{N+1} - x_{N+1}^r)^T P(x_{N+1} - x_{N+1}^r), \end{cases} \tag{7.2}$$

where x_i^r is the reference trajectory. We aim at following these steps the first state variable and choose $P = diag\{1, 0.001\}$, $R = 1$ for $x_i^r = \left[\sin(\frac{2\pi k}{20}); \cos(\frac{2\pi k}{20})\right]$ with $i = 1, \ldots, N$ and $N = 20$. The given initial state is $x_1 = [0; 0]$. As control law is taken $w^T g([x_i; x_i^r])$. In Figure 1, simulation results for method (5.9) show that Mercer's conditions are useful. The set of nonlinear equations was solved using Matlab's optimization toolbox (function leastsq) with unknowns $x_i, u_i, \lambda_i, \gamma, \beta$ with $\alpha = 100$.

The unknowns were randomly initialized with zero mean and standard deviation 0.3. The plots show the simulation results for the closed-loop system $\hat{x}_{i+1} = \varphi\left(\hat{x}_i, \sum_{t=1}^{N} \alpha_t K(x_t, \hat{x}_i)\right)$ with RBF kernel. The controller is generalizing well with respect to other initial conditions where the origin (for which it has been trained) and time horizon for $N = 20$. The method (5.10) without the use of Mercer's condition gave similar results by taking the centers $\{c_r\}$ equal to $\{x_i\}$.

FIGURE 1. optimal control by SVM-PSO for Eq (7.1)



8. Conclusion

In this paper we introduced the use of combine support vector machines with particle swarm algorithm for solving optimal control problems. It is shown that this method is valuable in order to solve up timing problems of engineering design.

References

[1] B. Akay and D. Karaboga, *Artificial bee colony algorithm for large-scale problems and engineering design optimization*, Journal of Intelligent Manufacturing., *23* (2012), 1001–1014.
[2] J. S. Arora, *Introduction to Optimum Design*, McGraw-Hill, New York, 1989.
[3] A. Kaveh and S. Talatahari, *Engineering optimization with hybrid particle swarm and ant colony optimization*, Asian J. Civil Eng.(Build.Hous.), *10* (2009), 611–628.
[4] J. Kennedy and R. C. Eberhart, *Particle swarm optimization*, in: IEEE International Conference on Neural Networks, Piscataway, NJ, Seoul, Korea, IV, 1995, 1942–1948.
[5] B. Scholkopf, K. K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, *Comparing support vector machines with Gaussian kernels to radial basis function classifiers*, IEEE Transactions on Signal Processing., *45*(11) (1997), 2758–2765.
[6] B. Scholkopf, C. J. C. Burges, and A. J. Smola, *Advances in kernel methods Support vector learning*, Cambridge, MA: MIT Press, 1999.
[7] A. Smola and B. Scholkopf, *On a kernel-based method for pattern recognition, regression, approximation and operator inversion*, Algorithmica., *22* (1998), 211–231.

[8] A. Smola, B. Scholkopf, and K. R. Muller, *The connection between regularization operators and support vector kernels*, Neural Networks., *11*(4) (1998), 637–649.
[9] J. A. K. Suykens and J. Vandewalle, *Training multilayer perceptron classifiers based on a modified support vector method*, IEEE Transactions on Neural Networks, *10*(4) (1999), 907–911.
[10] J. A. K. Suykens and J. Vandewalle, *Least squares support vector machine classifiers*, Neural Processing Letters., 9(3) (1999), 293-300.
[11] J. A. K. Suykens, B. De Moor, and J. Vandewalle, *Static and dynamic stabilizing neural controllers*, applicable to transition between equilibrium points. Neural Networks, *7*(5), (1994). 819–831.
[12] J. A. K. Suykens, B. De Moor, and J. Vandewalle, *NLq theory: a neural control framework with global asymptotic stability criteria*, Neural Networks, *10*(4) (1997), 615–637.
[13] J. A. K. Suykens, L. Lukas, and J. Vandewalle, *Sparse approximation using least squares support vector machines*, IEEE International Symposium on Circuits and Systems ISCAS 2000, Geneva, Switzerland, 2000, 28–31.
[14] J. A. K. Suykens, L. Lukas, and J. Vandewalle, *Sparse Least Squares Support Vector Machine Classifiers*, 8th European Symposium on Artificial Neural Networks ESANN 2000, Bruges Belgium, 2000, 37–42.
[15] J. A. K. Suykens, J. Vandewalle, and B. De Moor, *Artificial neural networks for modelling and control of non-linear systems*, Boston: Kluwer Academic, 1996.
[16] J. A. K. Suykens, J. Vandewalle, and B. De Moor, *Optimal control by least squares support vector machines*, Neural Networks, *14*(1) (2001), 23–35.
[17] V. Vapnik, *The nature of statistical learning theory*, New York: Springer., 2000.
[18] V. Vapnik, *Statistical learning theory*, New York: Wiley, 1998.
[19] V. Vapnik, *The support vector method of function estimation*, In J. A. K. Suykens and J. Vandewalle, Nonlinear modeling: advanced blackbox techniques. Boston: Kluwer Academic, 1998, 55-85.
[20] V. Vapnik, S. Golowich, and A. Smola, *Support vector method for function approximation, regression estimation and signal processing*, Advances in Neural Information Processing Systems, *9* (1996), 281–287.