An Online Algorithm for Support Vector Machine Based on Subgradient Projection

Hua Liu* . Zonghai Sun**

**College of Automation Science and Engineering,*
South China University of Technology
Guangzhou, China, (e-mail: liuhua105@163.com)
***College of Automation Science and Engineering,*
South China University of Technology
Guangzhou, China, (e-mail: sunzh@scut.edu.cn)

Abstract: This paper presents an online algorithm for support vector regression based on subgradient projection in reproducing kernel Hilbert spaces. Firstly, the paper choose the distance between f and the intersection of SK to characterize the empirical risk of support vector machine learning, and formulate a new expression of support vector regression sequentially. According to the optimization theory, the paper obtains the optimal solution of the new formulation of support vector regression by choosing the feasible clearance i.e. the difference between the primal objective function value and the dual objective function value to characterize the optimal solution of support vector machine. Secondly, the paper explains the selection of subgradient based on set theory and projection theory what is sensitive to the convergence of the algorithm. Finally, compared with projection adaptive natural gradient algorithm (PANG), the paper has verified that the accuracy of two algorithms are similar, however, the adaptive projected subgradient algorithm (APSA) trains much faster than the adaptive natural gradient algorithm by simulating the Mackey-Glass (MG) system and a class of nonlinear control system.

Keywords: machine learning, support vector regression, online algorithm, subgradient projection

1. INTRODUCTION

Support vector machine (SVM), based on the ideal of Structural Risk Minimization (SRM), is a powerful methodology in the field of machine learning (Vapnik 1995). It has shown good performance in pattern classification and regression fields, such as high generalization ability, global optimal solution and fast convergence speed. Online learning is an important learning scheme in which an unlimited stream of training data is arrived one example at a time, and can only be seen in a single pass. It is contrary to offline learning where you can get the whole training data at first. So it is essential to study the online algorithm for SVM as to applying SVM to the time-varying field which may has unlimited stream of training data, due to the unfitness of the offline algorithm in these fields.

So far, it has yielded some results in the field of online algorithm for SVM. Cauwenberghs and Poggio proposed the IDSVM algorithm, which is one of the earliest online algorithms for SVM (Cauwenberghs and Poggio 2001). Ma et al. proposed an accurate on-line support vector regression (AOSVR) algorithm which is incremental algorithm essentially (Ma, Theiler, Perkins 2003). Wang et al. had improved the incremental algorithm training speed by introducing a kernel function matrix cache (Wang, Pi and Sun 2004). All these incremental algorithms have high computational complexity, because they are essentially solving a quadratic optimization problem. In the worst case, the computational complexity of the incremental algorithm is O (L^3)*O (kernel), where L denotes the dimension of weight vector ω. Kuh.; Liu et al.; Lau et al. had realized the online least squares support vector machine learning algorithm by applying LS-SVM to the online algorithm respectively (Kuh 2002; Liu et al. 2003; Lau and Wu 2003). However, they don't have sparse solutions. The complexity of these algorithms is O (n^3), where n denotes the number of training data. Sun et al. proposed the projection adaptive natural gradient (PANG) online algorithm for SVR by introducing the natural gradient to SVR (Sun Shi 2010). The complexity of PANG is O (L^2+L), where L denotes the dimension of weight vector. It doesn't have sparse solutions either.

This paper has realized the adaptive projected subgradient algorithm (APSA) based on a new formulation of support vector regression what will be introduced later in this paper. Yamada and Ogura proposed the adaptive projected subgradient method (Yamada Ogura 2004), which is a natural extension of the Polyak's subgradient algorithm (Gubin et al. 1967). The convergence of APSA is sensitive to the selection of the subgradient. Compared to PANG, the algorithm realized in this paper has lower computational complexity however with the similar accuracy.

The rest of the paper is organized as follows. The preliminary of subgradient projection is present in section 2. Section 3 introduced a new formulation of support vector regression based on a new selection of measurement of empirical risk.

Section 4 formulates the adaptive projected subgradient algorithm for the new formulation of support vector regression in section 3. Section 5 gives the realization of the adaptive projected subgradient algorithm. The numerical simulation of MG system and a class of nonlinear system lies in section 6. Finally, conclusions are made in section 7. In the rest of the paper, the bold denotes vectors or matrixes, and the normal font denotes scalar.

2. SUBGRADIENT PROJECTION

2.1 Projection onto A Closed Convex Set

Let C be a nonempty closed convex set in H. For any $f \in H$, define the distance of f from the set C as: $d(f,C) = \inf_{g \in C} \|f-g\|$. For each point in H, there exists a unique point $f_* \in C$ is called the projection of f onto convex set C, and it denotes as $f_* = P_C(f)$. Note that if $f \in C$, then $f_* = f$, i.e., the projection is the point itself. Fig. 1 illustrates the geometry of the projection.

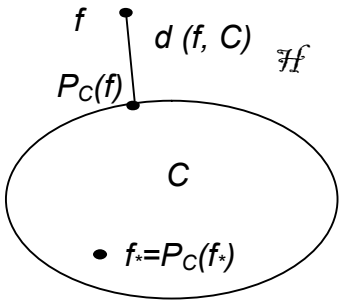

Fig. 1. The projection onto the closed convex set C

2.2 Adaptive Projected Subgradient Method

First, let us define the convex function. A function $\Phi : R^N \to R$ is said to be convex if $\Phi(vx+(1-v)y) \le v\Phi(x)+(1-v)\Phi(y)$, for all $(x,y) \in R^N \times R^N$, for all $v \in (0,1)$.

For a continuous convex function, the subgradient is a generalization of gradient (To be precise, subgradient is a generalization of Gâteaux differential.), and it always exists. In a differentiable case, the gradient $\Phi'(y)$ at an arbitrary point $y \in R^N$ is characterized as the unique vector satisfies $\langle x-y, \Phi'(y)\rangle + \Phi(y) \le \Phi(x)$, where $\langle \bullet, \bullet \rangle$ denotes the inner product in R^N, for all $x \in R^N$. In a nondifferentiable case, however, such a vector is not unique in general, and the set of such vectors $\partial\Phi(y) = \{a \in R^N \mid \langle x-y, a\rangle + \Phi(y) \le \Phi(x), \forall x \in R^N\} \neq \varnothing$ is called subdifferential of Φ at $y \in R^N$. Elements of the subdifferential $\partial\Phi(y)$ are called the subgradients of Φ at y.

Let $\Omega_n : H \to [0,\infty) (\forall n \in N)$ be a sequence of continuous convex functions and $K \subset H$ a nonempty closed convex set. For an arbitrarily given $u_0 \in K$, the sequence $(u_n)_{n \in N} \subset K$ generated by the adaptive projected subgradient method:

$$u_{n+1} = \begin{cases} P_K\left(u_n - \lambda_n \dfrac{\Omega_n(u_n)}{\left\|\Omega_n'(u_n)\right\|^2} \Omega_n'(u_n)\right) \text{ if } \Omega_n'(u_n) \neq 0 \\ u_n \qquad \text{otherwise} \end{cases} \tag{1}$$

Where $\Omega_n'(u_n) \in \partial\Omega_n(u_n)$ and $0 \le \lambda_n \le 2$. More details can be seen in the reference Yamada and Ogura, 2004.

3. SUPPORT VECTOR MACHINE

Suppose given the training data set $T = \{x_k, y_k\}_{k=1}^N$, where $x_k \in R^N$, $y_k \in R$, N is the size of the training data. For any integer $1 \le k \le N$, the functional relation between x_k and y_k by the following regression model:

$$y_k = f(x_k) = \omega^T \varphi(x_k) + b, k = 1, 2 \cdots, N \tag{2}$$

Where $\omega \in R^N$ denotes the weight vector, $b \in R$ denotes the offset.

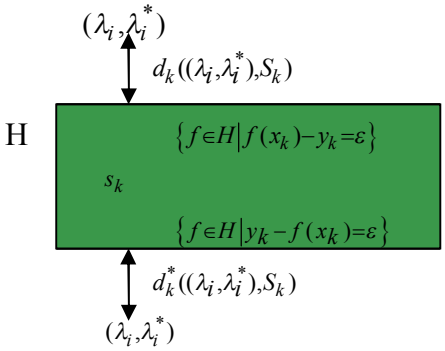

Fig. 2. S_k i.e. the set of all points that satisfy the bound ε between y_k and $f(x_k)$

In order to minimize the empirical risk, our goal becomes to find the function $f(\bullet)$ in (2), so that y_k is as close as possible to $f(x_k)$. Introducing the tolerance ε, for every sample point, the function $f(x_k)$ satisfies $S_k = \left\{f \in H \middle| \left|y_k - f(x_k)\right| \le \varepsilon\right\}, \forall k \in \{1, \cdots, N\}$ as depicted in figure 2. In other words, for each sample point $f(x_k)$ lies inside the hyperslab S_k. For all the sample points, our goal is to find a $f \in H$ lies in the intersection of all S_k, i.e.

$$f \in \bigcap_{k=1}^{N} S_k \tag{3}$$

Define the distances between f and hyperslab S_K as:

$$\begin{cases} d_k(f,S_k) = \max((\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x_k}) + b - y_k - \varepsilon)/\left\|\boldsymbol{\varphi}(\mathbf{x_k})\right\|, 0) \\ d_k^*(f,S_k) = \max((y_k - \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x_k}) - b - \varepsilon)/\left\|\boldsymbol{\varphi}(\mathbf{x_k})\right\|, 0) \end{cases} \tag{4}$$

Where $\left\|\boldsymbol{\varphi}(\mathbf{x_k})\right\|$ denotes the norm of $\boldsymbol{\varphi}(\mathbf{x_k})$ in Hilbert space H. The geometric meaning was illustrated in fig. 2.

So, the distance between f and $\bigcap_{k=1}^{N} S_k$ can be expressed as $\sum_{k=1\cdots N}\theta(k)(d_k(f,S_k)+d_k^*(f,S_k))$, where $\theta(k) = \dfrac{d_k(f,S_k)+d_k^*(f,S_k)}{\sum_{k=1\cdots N}(d_k(f,S_k)+d_k^*(f,S_k))}$, which characterize the empirical risk of the learning. In the support vector machine learning, minimize the following function L, based on the ideal of Structural Risk Minimization (SRM).

$$\min L = \frac{1}{2}\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega} + C\sum_{k=1\cdots N}\theta(k)(d_k(f,S_k)+d_k^*(f,S_k)) \tag{5}$$

$$\text{s.t.}\begin{cases} \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x_k}) + b - y_k \le \varepsilon + d_k(f,S_k)\left\|\boldsymbol{\varphi}(\mathbf{x_k})\right\| & k = 1\cdots N \\ y_k - \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x_k}) - b \le \varepsilon + d_k^*(f,S_k)\left\|\boldsymbol{\varphi}(\mathbf{x_k})\right\| & k = 1\cdots N \\ d_k(f,S_k),\ d_k^*(f,S_k) \ge 0 & k = 1\cdots N \end{cases} \tag{6}$$

The constant C>0 determines the trade-off between the flatness of y and the amount up to which deviations larger than ε are tolerated. The function L in (5) was called the primal objective function.

By introducing a dual set of variables, it can construct a Lagrange function from the primal objective function and the corresponding constraints. It has the dual objective function as follows:

$$\begin{aligned} \max L = {} & \frac{1}{2}\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega} + C\sum_{k=1\cdots N}\theta(k)(d_k(f,S_k)+d_k^*(f,S_k)) \\ & - \sum_{k=1\cdots N}\lambda_k(y_k - \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x_k}) - b + \varepsilon + d_k(f,S_k)\left\|\boldsymbol{\varphi}(\mathbf{x_k})\right\|) \\ & - \sum_{k=1\cdots N}\lambda_k^*(\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x_k}) + b - y_k + \varepsilon + d_k(f,S_k)\left\|\boldsymbol{\varphi}(\mathbf{x_k})\right\|) \\ & - \sum_{k=1\cdots N}(\alpha_k d_k(f,S_k) + \alpha_k^* d_k^*(f,S_k)) \end{aligned} \tag{7}$$

Setting the partial derivatives of L with respect to the primal variables $\boldsymbol{\omega}$, b, $d_k(f,S_k)$, $d_k^*(f,S_k)$ to zeros, it obtains:

$$\begin{cases} \boldsymbol{\omega} = \sum_{k=1\cdots N}(\lambda_k^* - \lambda_k)\boldsymbol{\varphi}(\mathbf{x_k}) \\ C\theta(k) = \alpha_k + \lambda_k\left\|\boldsymbol{\varphi}(\mathbf{x_k})\right\| \quad k = 1\cdots N \\ C\theta(k) = \alpha_k + \lambda_k^*\left\|\boldsymbol{\varphi}(\mathbf{x_k})\right\| \quad k = 1\cdots N \\ \sum_{k=1}^{N}\lambda_k = \sum_{k=1}^{N}\lambda_k^* \end{cases} \tag{8}$$

Where, $\alpha_k, \lambda_k, \alpha_k^*, \lambda_k^*$ denote Lagrangian multipliers. Substitute (8) into (7), the dual formulation of the primal problem (5), (6) can be expressed as:

$$\begin{aligned} \min L = {} & \frac{1}{2}(\boldsymbol{\lambda}^* - \boldsymbol{\lambda})^T K(\boldsymbol{\lambda}^* - \boldsymbol{\lambda}) \\ & - (\boldsymbol{\lambda}^* - \boldsymbol{\lambda})^T \boldsymbol{y} + \varepsilon(\boldsymbol{\lambda}^* - \boldsymbol{\lambda})^T \boldsymbol{r} \end{aligned} \tag{9}$$

$$\text{s.t.}\begin{cases} 0 \le \boldsymbol{\lambda} \le (\dfrac{C\theta(k)}{\left\|\boldsymbol{\varphi}(\boldsymbol{x_k})\right\|})\boldsymbol{r} \\ 0 \le \boldsymbol{\lambda}^* \le (\dfrac{C\theta(k)}{\left\|\boldsymbol{\varphi}(\boldsymbol{x_k})\right\|})\boldsymbol{r} \\ \sum_{k=1}^{N}\lambda_k = \sum_{k=1}^{N}\lambda_k^* \end{cases} \tag{10}$$

Where $\mathbf{r}$ denotes the column vector of 1 of N dimension, $\boldsymbol{\lambda}, \boldsymbol{\lambda}^*, \boldsymbol{y}$ are column vectors construct by $\lambda_k, \lambda_k^*, y_k$ respectively. $\boldsymbol{K}(x_k, \boldsymbol{x}) = \boldsymbol{\varphi}(x_k)^T\boldsymbol{\varphi}(\boldsymbol{x})$ denotes kernel matrix which meets the Mercer's condition. Then, substituting (8) into (2), the output of SVM is:

$$f(x_k) = (\boldsymbol{\lambda}^* - \boldsymbol{\lambda})^T \boldsymbol{K}(x_k, \boldsymbol{x}) + b \tag{11}$$

4. SUPPORT VECTOR REGRESSION BASED ON SUBGRADIENT PROJECTION

From the last section, it shows that solving a support vector regression problem equals to solving a convex quadratic optimization problem (9) under the constrain (10). According to the optimization theory, the difference between the primal objective function value and the dual objective function value comes to zero when it reaches to optimum (Cristianini and Shawe-Taylor 2005). (5) subtract (7), it obtains the difference what called feasible clearance Θ:

$$\begin{aligned} \Theta = {} & \min L - \max L = \\ & \sum_{k=1\cdots N}\lambda_k(y_k - \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x_k}) - b + \varepsilon + d_k(f,S_k)\left\|\boldsymbol{\varphi}(\mathbf{x_k})\right\|) + \\ & \sum_{k=1\cdots N}\lambda_k^*(\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x_k}) + b - y_k + \varepsilon + d_k(f,S_k)\left\|\boldsymbol{\varphi}(\mathbf{x_k})\right\|) + \\ & \sum_{k=1\cdots N}(\alpha_k d_k(f,S_k) + \alpha_k^* d_k^*(f,S_k)) \end{aligned} \tag{12}$$

Observing (11), it is obviously that f is the function of $\boldsymbol{\lambda}, \boldsymbol{\lambda}^*$. So the corresponding Hilbert spaces H is constructed of $\boldsymbol{\lambda}, \boldsymbol{\lambda}^*$, and $\|\boldsymbol{\varphi}(\mathbf{x_k})\|$ becomes to $\|2\boldsymbol{K}(x, x_k)\|$. Considering (10), substituting (8) into (12), it establishes a new formulation of support vector regression:

$$\begin{aligned}\Theta = C\sum\nolimits_{k=1\cdots N}\theta(k)(d_k + d_k^*) + \varepsilon(\boldsymbol{\lambda}+\boldsymbol{\lambda}^*)^T\boldsymbol{r} \\ (\boldsymbol{\lambda}+\boldsymbol{\lambda}^*)^T\boldsymbol{y} + (\boldsymbol{\lambda}^*-\boldsymbol{\lambda})^T\boldsymbol{K}(\boldsymbol{\lambda}^*-\boldsymbol{\lambda})\end{aligned} \tag{13}$$

$$\text{S.t.}\begin{cases} 0 \le \boldsymbol{\lambda} \le (\dfrac{C\theta(k)}{\|\boldsymbol{\varphi}(\boldsymbol{x_k})\|})\boldsymbol{r} \\ 0 \le \boldsymbol{\lambda}^* \le (\dfrac{C\theta(k)}{\|\boldsymbol{\varphi}(\boldsymbol{x_k})\|})\boldsymbol{r} \\ \sum_{k=1}^{N}\lambda_k = \sum_{k=1}^{N}\lambda_k^* \end{cases} \tag{14}$$

So, the convex quadratic optimization problem is equivalent to find some $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}^*$ satisfy the convex constrain (14) and make (13) comes to zero asymptotically.

Introducing the adaptive projected subgradient method in (1), it obtains $\boldsymbol{\lambda}_{i+1}, \boldsymbol{\lambda}_{i+1}^*$ in i-th iteration are:

$$\boldsymbol{\lambda}_{i+1} = \begin{cases} \boldsymbol{\lambda}_i & if\ \partial_{\boldsymbol{\lambda}_i}\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*) = 0 \\ P_K(\boldsymbol{\lambda}_i - \mu_i \dfrac{\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*)}{\left\|\partial_{\boldsymbol{\lambda}_i}\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*)\right\|^2}\partial_{\boldsymbol{\lambda}_i}\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*)) & otherwise \end{cases} \tag{15}$$

$$\boldsymbol{\lambda}_{i+1}^* = \begin{cases} \boldsymbol{\lambda}_{i+1}^* & if\ \partial_{\boldsymbol{\lambda}_i^*}\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*) = 0 \\ P_K(\boldsymbol{\lambda}_i^* - \mu_i^* \dfrac{\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*)}{\left\|\partial_{\boldsymbol{\lambda}_i^*}\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*)\right\|^2}\partial_{\boldsymbol{\lambda}_i^*}\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*)) & otherwise \end{cases} \tag{16}$$

Where $\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*)$ denotes the value of function (13) in i-th iteration, $\partial_{\boldsymbol{\lambda}_i}\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*)$, $\partial_{\boldsymbol{\lambda}_i^*}\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*)$ denote the subgradient of $\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*)$. $\mu_i, \mu_i^* \in [0, 2]$, K denotes the constrain (14).

By examining (15) and (16), it shows that the selection of subgradient will directly affect the convergence speed of the optimization process. So there is a critical issue: how to choose the subgradient?

Considering $d_k(f, S_k) * d_k^*(f, S_k) = 0, \boldsymbol{\lambda}_i^T\boldsymbol{\lambda}_i^* = 0$, the partial derivatives of Θ_i with respect to $\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*$ are:

$$\begin{aligned}\partial_{\boldsymbol{\lambda}_i}\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*) = C\partial_{\boldsymbol{\lambda}_i}\sum_{k=1\cdots N}\theta(k)(d_k + d_k^*) + \varepsilon\boldsymbol{R} + \boldsymbol{Y} + \\ 2\boldsymbol{K}(\boldsymbol{\lambda}_i^* - \boldsymbol{\lambda}_i)\end{aligned} \tag{17}$$

$$\begin{aligned}\partial_{\boldsymbol{\lambda}_i^*}\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*) = C\partial_{\boldsymbol{\lambda}_i^*}\sum_{k=1\cdots N}\theta(k)(d_k + d_k^*) + \varepsilon\boldsymbol{R}^* + \boldsymbol{Y} + \\ 2\boldsymbol{K}(\boldsymbol{\lambda}_i^* - \boldsymbol{\lambda}_i)\end{aligned} \tag{18}$$

Where $\partial_{\boldsymbol{\lambda}_i}\sum_{k=1\cdots N}\theta(k)(d_k + d_k^*)$, $\partial_{\boldsymbol{\lambda}_i^*}\sum_{k=1\cdots N}\theta(k)(d_k + d_k^*)$ denote the partial derivatives of $\sum_{k=1\cdots N}\theta(k)(d_k + d_k^*)$ with respect to $\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*$. **R**, **R***, **Y**, and **Y*** denote the column vectors which composed of R_k, R_k^*, Y_k, Y_k^* respectively, and they are defined as follows $R_k = \begin{cases} 0 & if\ \lambda_k = 0 \\ 1 & else \end{cases}$,

$R_k^* = \begin{cases} 0 & if\ \lambda_k^* = 0 \\ 1 & else \end{cases}$, $Y_k = \begin{cases} 0 & if\ \lambda_k = 0 \\ 1 & else \end{cases}$, $Y_k^* = \begin{cases} 0 & if\ \lambda_k^* = 0 \\ 1 & else \end{cases}$,

Considering the definition of $d_k(f, S_k)$ and $d_k^*(f, S_k)$ in (4) and the projection theory, select $\partial_{\boldsymbol{\lambda}_i}\sum_{k=1\cdots N}\theta(k)(d_k + d_k^*)$ and $\partial_{\boldsymbol{\lambda}_i^*}\sum_{k=1\cdots N}\theta(k)(d_k + d_k^*)$ as follows:

$$\partial_{\boldsymbol{\lambda}_i}\sum_{k=1\cdots N}\theta(k)(d_k + d_k^*) = \begin{cases} 0 & if\ \sum_{k=1\cdots N}d_k + d_k^* = 0 \\ \sum_{k=1}^{N}\theta(k)(\dfrac{d_k^* - d_k}{\|2\boldsymbol{K}(\boldsymbol{x}, x_k)\|}\boldsymbol{K}(\boldsymbol{x}, x_k)) & otherwise \end{cases} \tag{19}$$

$$\partial_{\boldsymbol{\lambda}_i^*}\sum_{k=1\cdots N}\theta(k)(d_k + d_k^*) = \begin{cases} 0 & if\ \sum_{k=1\cdots N}d_k + d_k^* = 0 \\ \sum_{k=1}^{N}\theta(k)(\dfrac{d_k - d_k^*}{\|2\boldsymbol{K}(\boldsymbol{x}, x_k)\|}\boldsymbol{K}(\boldsymbol{x}, x_k)) & otherwise \end{cases} \tag{20}$$

Substituting (19) and (20) into (17) and (18), it has:

$$\partial_{\boldsymbol{\lambda}_i}\Theta_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i^*) = \begin{cases} \varepsilon\boldsymbol{R} + \boldsymbol{Y} + 2\boldsymbol{K}(\boldsymbol{\lambda}_i^* - \boldsymbol{\lambda}_i) & if\ \sum_{k=1}^{N}d_k + d_k^* = 0 \\ \varepsilon\boldsymbol{R} + \boldsymbol{Y} + 2\boldsymbol{K}(\boldsymbol{\lambda}_i^* - \boldsymbol{\lambda}_i) + \sum_{k=1}^{N}\theta(k)(\dfrac{d_k^* - d_k}{\|2\boldsymbol{K}(\boldsymbol{x}, x_k)\|}\boldsymbol{K}(\boldsymbol{x}, x_k)) & others \end{cases} \tag{21}$$

$$\partial_{\lambda_i^*}\Theta_i(\lambda_i,\lambda_i^*)=$$

$$\begin{cases} \varepsilon \boldsymbol{R}^* - \boldsymbol{Y}^* + 2\boldsymbol{K}(\lambda_i - \lambda_i^*) & if \ \sum_{k=1}^{N} d_k + d_k^* = 0 \\ \varepsilon \boldsymbol{R}^* - \boldsymbol{Y}^* + 2\boldsymbol{K}(\lambda_i - \lambda_i^*) + \sum_{k=1}^{N} \theta(k)\left(\frac{d_k - d_k^*}{\left\|2\boldsymbol{K}(\boldsymbol{x},x_k)\right\|}\boldsymbol{K}(\boldsymbol{x},x_k)\right) & others \end{cases} \quad (22)$$

Considering the constrain $(\lambda)^T \boldsymbol{r} = (\lambda^*)^T \boldsymbol{r}$, it obtains,

$$(\lambda_i - \mu_i \frac{\Theta_i}{\left\|\partial_{\lambda_i}\Theta_i\right\|^2}\partial_{\lambda_i}\Theta_i)^T \boldsymbol{r} = (\lambda_i^* - \mu_i^* \frac{\Theta_i}{\left\|\partial_{\lambda_i^*}\Theta_i\right\|^2}\partial_{\lambda_i^*}\Theta_i)^T \boldsymbol{r} \quad (23)$$

Simplifying (23),

$$\mu_i^* = \frac{\mu_i((\partial_{\lambda_i}\Theta_i)^T \boldsymbol{r})(\left\|\partial_{\lambda_i^*}\Theta_i\right\|)^2}{(\left\|\partial_{\lambda_i}\Theta_i\right\|)^2((\partial_{\lambda_i^*}\Theta_i)^T)\boldsymbol{r}} \quad \mu_i \in (0,2] \quad (24)$$

Substituting (21), (22) and (24) into (15) and (16) respectively, it obtains the complete iterative formula. The convergence of the algorithm has no difference from (1), and it was proofed in the reference of Yamada and Ogura 2004.

5. THE REALIZATION OF ADAPTIVE PROJECTED SUBGRADIENT ALGORITHM

The dimension of kernel matrix K increases with the increasing of sample points. When it reaches to a certain amount, the kernel matrix K will be not calculated, the online learning algorithm will not be realized. So, choose an integer l, assuming that l-dimensional K matrix can represent all the information of the solving problem. Assuming denotes the index set. The steps of the algorithm implementation are given as follows.

Algorithm:

- Starting time index i=1, select ε, C, integer l and kernel function K; initialize $d_k, d_k^* = 0 \ k \in \Gamma_i$, select initial feasible solutions λ_1, λ_1^*;
- Repeat over the time index i as follows
  - If $i \le l$, set the index set $\Gamma_i = \{1, 2, \cdots, i\}$; calculate K by sample data $\{(x_1, y_1), (x_2, y_2) \ (x_i, y_i)\}$, calculate $d_k, d_k^*, \Theta_i, \Theta_i^{'}, \mu_i, \mu_i^*$; calculate $\lambda_{i+1}, \lambda_{i+1}^*$ via (4), (13), (15), (16), (21), (22) and (24);
  - Else, set the index set $\Gamma_i = \{i - l + 1, \cdots, i\}$; calculate K by sample data $\{(x_{i-l+1}, y_{i-l+1}) \ (x_i, y_i)\}$, calculate $d_k, d_k^*, \Theta_i, \Theta_i^{'}, \mu_i, \mu_i^*$; calculate $\lambda_{i+1}, \lambda_{i+1}^*$ via (4), (13), (15), (16), (21), (22) and (24).

6. NUMERICAL SIMULATION

In this section, we give two simple examples to illustrate the effectiveness of the proposed online algorithm. One is Mackey-Glass (MG) system, the other is a simple class of nonlinear system. Compared with the projection adaptive natural gradient algorithm, it is easy to find that the proposed algorithm has better performance.

6.1 Simulation of Mackey-Glass System

The MG system is a blood cell regulation model established in 1977 by Mackey and Glass. It is a chaos system described as:

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1 + x^{10}(t-\tau)} - bx(t)$$

The embedded dimensions are $x(t)$ and $x(t-\tau)$. In the simulation set a=0.2, b=0.1, $\tau = 17$, $\Delta t = 0.1$, $t \in (0, 200]$. Select a Gaussian function with kernel parameter $\sigma = 0.065$ as the kernel function. Compared with the projection adaptive NG algorithm, the mean square errors (MSE) of the last 200 sample points and the mean learning time per iteration were calculated in the simulation. All the results are shown in table 1.

Table 1. The result of simulation of MG system

Algorithm / result	PANG	**APSA**
MSE	2.8101e-05	**8.5686e-06**
Mean learning time(s)	0.03200	**0.00636**
Parameters: $\varepsilon = 0.001$, C = 0.4, $\sigma = 0.065$		

In table 1, it shows that the MES of APSA is less than that of PANG. The mean learning time of APSA is much less than that of PANG either. The regression errors are shown in fig. 3. It can be seen that the errors of adaptive projected subgradient algorithm are less than that of projection adaptive NG algorithm.

6.2 Simulation of A Simple Class of Nonlinear System

As to comparing with APSA, the same example as in the reference of Sun et al. 2011 was used in this paper. Given the following single-input single-output nonlinear system:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_1^2 + 0.15u^3 + 0.1(1 + x_2^2)u + \sin(0.1u) \\ y = x_1 \end{cases}$$

The purpose is to make y track the object trajectory $y_d = \sin t + \cos(0.5t)$. Give the initial state of the system $x(0) = [0.2, 0.2]^T$.

In the simulation, select a Gaussian function with kernel parameter $\sigma = 2$ as the kernel function, select $p_1 = 12, p_2 = 7$. Select initial solution $\boldsymbol{\lambda}_1 = [0.15, 0]^T$, $\boldsymbol{\lambda}_1^* = [0, 0.15]^T$. The tracking errors whose time index is from 1000 to 4000 are shown in fig. 4. It shows that the tracking error of adaptive projected subgradient algorithm is a little less than that of projection adaptive NG algorithm. The mean square errors (MSE) of the last 1000 sample points and the mean learning time per iteration were calculated in the simulation. The results are shown in table 2.

Table 2. The results of simulation of nonlinear system

Algorithm / result	*PANG*	***APSA***
MSE	7.2934e-06	**4.7731e-06**
Mean learning time(s)	0.00931	**0.00148**
Parameters: $\varepsilon = 0.006, C = 0.25, \sigma = 2$		

From table 2, it's easy to see that the MSE of adaptive projected subgradient algorithm is less than that of projection adaptive NG algorithm. The mean learning time of adaptive projected subgradient algorithm is much less than that of the projection adaptive NG algorithm either. It can be seen that the computation complexity of the adaptive projected subgradient algorithm is dominated by (15) and (16), and the complexity of the two equations is O (L^2). Compared with the results in the reference of Sun and Shi 2010, it is easy to explain the result of mean learning time in table 1 and table 2.

7. CONCLUSIONS

This paper has realized the adaptive projected subgradient algorithm for a new formulation of support vector regression. Firstly the article formulates a new expression of support vector regression by choosing the distance between f and the intersection of S_K to characterize the empirical risk of support vector machine learning. Then, it obtains the optimal solution of SVR by minimizing the feasible clearance. Secondly, it explained the selection of subgradient, and obtained the realization of APSA for the new formulation of support vector regression. Finally, the adaptive projected subgradient algorithm is applied to the MG system and a class of nonlinear system in simulation. Compared with projection adaptive natural gradient algorithm, the accuracy of two algorithms are similar, however, the adaptive projected subgradient algorithm is training much faster than the projection adaptive natural gradient algorithm. Further, improving the accuracy of the algorithm by introducing some more effective adaptive learning strategies is considerable. Moreover the robustness of the algorithm should be well studied.

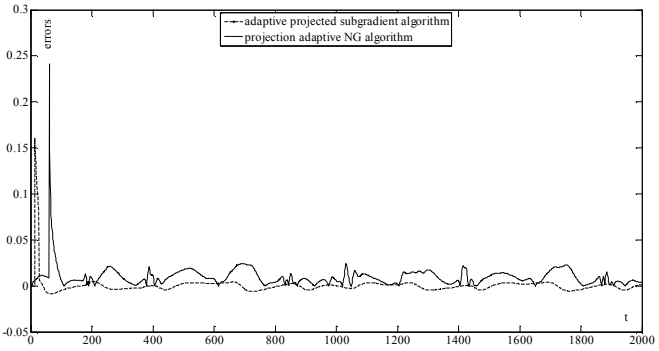

Fig. 3. Learning errors of APSA and PANG

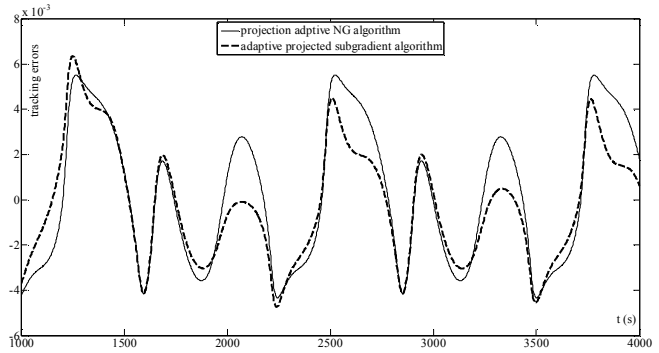

Fig. 4. Tacking errors of PANG and APSA, the time index is from 1000 to 4000.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation of China under grant NO.60704012, NO.60574019 and the Science Foundation of Guangdong Province under grant NO.06300232, and the Fundamental Research Funds for the Central Universities, SCUT NO.2009zm0161.

REFERENCES

Cauwenberghs, G.,Poggio, T. (2001). Incremental and Decremental *Support Vector Machine Learning*, Vol. 44, No. 13, p 409-415.

Cristianini, N., Shawe-Taylor, J. (2005). An *Introduction to Support Vector Machines and another Kernel-based Learning Method*, China Machine Press, Peking, China.

Gubin, L.G., Polyak, B.T., Raik, E.V. (1967). The Method of Projections for Finding the Common Point of Convex Sets. *USSR Computational Mathematics and Mathematical Physics*, Vol. 7, No.6, p 1-24.

Kuh, A. (2002). Adaptive Least Square Kernel Algorithms and Applications. In Proceedings of International Joint Conference on Neural Networks, p 2104-2107.

Lau, K.W., Wu, Q.H. (2003). Online Training of Support Vector Classifier, *Pattern Recognition*, Vol. 36, p 1913-1920.

Liu, J.H., Chen, J.P., et al. (2003). On-line LS-SVM for Function Estimation and Classification, *Journal of University of Science and Technology Beijing*, Vol. 10, No. 5, p 73-77.

Ma, J.Sh., Theiler, J., Perkins, S. (2003). Accurate on-line Support Vector Regression, *Neural Computation,* Vol. 15, No. 11, p 2683-2703.

Smola, A., Scholkopf, B. (2004). A Tutorial on Support Vector Regression, *Statistics and Computing*, Vol. 14, No. 8, p 199-222.

Sun, Z.H., Shi, B.H. (2010). The Projection Adaptive Natural Gradient Online Algorithm for SVM, In Proceedings of the 29th Chinese Control Conference.

Sun, Z.H., Hu, M., Liu, H. (2011) Adaptive Control of Nonlinear System Based On SVM Online Algorithm, Submit to the 30th Chinese Control Conference.

Theodoridis, S., Slavakis, K., Yamada, I. (2011). Adaptive Learning in AWorld of Projections, IEEE signal processing magazine, Vol. 28, No. 1, p 97-123.

Vapnik, V.N. (1995) *The Nature of Statistical Learning Theory*. Springer Verlag, New York.

Wang, H., Pi, D.Y., Sun, Y.X. (2004). On-line Support Vector Machine Training Algorithm and its Application, *Journal of Zhejiang University (Engineering Science),* Vol. 38, No.12, p 1642-1645.

Yamada, I., Ogura, N. (2004). Adaptive Projected Subgradient Method for Asymptotic Minimization of Sequence of Nonnegative Convex Functions, *Numerical Functional Analysis and Optimization*, Vol. 25, No. 7 & 8, p 593-617.

Yukawa, M., Yamada, I. (2009). A Unified View of Adaptive Variable-Metric Projection Algorithms, *EURASIP Journal on Advances in Signal Processing*, Vol. 2009 January 2009.