An Online Kernel Learning Algorithm based on Orthogonal Matching Pursuit

ShiLei Zhao
School of Software, Harbin University of Science and Technology, Harbin, china
Email: zhaosl1210@126.com

Peng Wu
College of Mechanical and Electrical Engineering, Northeast Forestry University, Harbin, china
Email: wupeng_001980@163.com

YuPeng Liu
School of Software, Harbin University of Science and Technology, Harbin, china

Abstract—Matching pursuit algorithms learn a function that is weighted sum of basis functions, by sequentially appending functions to an initially empty basis, to approximate a target function in the least-squares sense. Experimental result shows that it is an effective method, but the drawbacks are that this algorithm is not appropriate to online learning or estimating the strongly nonlinear functions. In this paper, we present a kind of online kernel learning algorithm based on orthogonal matching pursuit. The orthogonal matching pursuit is employed not only to guide our online learning algorithm to estimate the target function but also to keep control of the sparsity of the solution. And the introduction of “kernel trick” can effective reduce the error when it is used to estimate the nonlinear functions. At last, a kind of nonlinear two-dimensional “sinc” function is used to test our algorithm and the results are compared with the well-known SVMTorch on Support Vectors percent and root mean square error which approve that our online learning algorithm is effective.

Index Terms—orthogonal matching pursuit; kernel trick; online learning;

I. INTRODUCTION

Recently, there has been a renewed interest for Kernel-based methods, due in great part to the success of Support Vector Machine approach[1]. Support Vector Machine are kernel-based learning algorithm in which only a fraction of training samples are used in the solution, and where the objective of learning is to maximize a margin around the decision surface.

Kernel machines[2] are another class of learning algorithms utilizing kernels in order to produce non-linear versions of conventional linear learning algorithms. The basic idea behind kernel machines is kernel function, the kernel function used frequently is the Mercer kernels which applies to pairs of input vectors, can be interpreted as an inner product in a high dimensional Hilbert space (the feature space), thus allowing inner products in feature space to be computed without making direct reference to feature vectors. This idea, commonly known as the “kernel trick”, has been used extensively in recent years, most notably in classification and regression[3][4][5].

Online algorithms are useful in learning scenarios where input samples are observed sequentially, one at a time step. In such cases, there is a clear advantage to algorithms that do not need to relearn from scratch when new data arrives. An important requirement of an online algorithm is that its per-time-step computational burden should be as light as possible, for it is assumed that samples arrive at a constant rate.

This paper present a new online kernel learning algorithm based on orthogonal matching pursuit which considers not only the introduction of “kernel trick” to decrease the error of estimating nonlinear target functions but also the problem of the online computational burden. This paper organized as follows:

We first (section 2) give the introductions of the basic matching pursuit method which is always used in signal-processing community. It was a general, greedy, sparse function approximation scheme with the squared error loss, which iteratively adds new functions.

We then show (section 3) main steps of the OMP algorithm which is based on the matching pursuit, it has a character of sparsity which use the orthogonal basis of the dictionary collection, but it is still an offline algorithm.

In section 4, we describe online learning algorithm we proposed in detail whose name is “an online kernel learning algorithm based on the orthogonal matching pursuit”. We first introduce the “kernel trick” to the OMP algorithm, then we revise the OMP algorithm and make it appropriate to the online learning.

Finally, in section 5, we provide an experimental comparison between the online kernel learning algorithm based on the orthogonal matching pursuit and the well-known SVMTorch. The experimental results are that the online kernel learning algorithm based on the orthogonal matching pursuit can yield performance as well as the SVMTorch, but has a fewer support vectors.
II. BASIC MATCHING PURSUIT

In this section, we describe the basic Matching pursuit algorithm[6] briefly.

We are given $l$ noisy observations $\{y_1, \ldots, y_l\}$ of a target function $f \in \mathbf{H}$ at points $\{x_1, \ldots, x_l\}$. We are also given a finite dictionary $D = \{d_1, \ldots, d_M\}$ of $\mathbf{M}$ functions in a Hilbert space $\mathbf{H}$, and we are interested in sparse approximations of $f$ that are expansions of the form:

$$f_N = \sum_{n=1}^{N} \alpha_n g_n$$  \hspace{1cm} (1)

Where $N$ is the number of basic functions in the expansion; $\{g_1, \ldots, g_N\} \subset D$ can be called the basic of the expansion; $\{\alpha_1, \ldots, \alpha_N\}$ is the set of corresponding coefficients of the expansion; $f_N$ designs an approximation of $f$ that uses exactly $N$ distinct basis functions taken from the dictionary.

For any function $f \in \mathbf{H}$, we will use $\hat{f}$ to represent the $l$ -dimensional vector that corresponds to the evaluation of $f$ on the $l$ training points:

$$\hat{f} = \left( f(x_1), \ldots, f(x_l) \right)$$  \hspace{1cm} (2)

where $\hat{y} = \{y_1, \ldots, y_l\}$ is the target vector; $\hat{R}_N = \hat{y} - \hat{f}_N$ is the residue; $\{\hat{h}_1, \hat{h}_2\}$ will be used to represent the usual dot product between vector $\hat{h}_1$ and $\hat{h}_2$; $\|\hat{h}\|$ will be used to represent the usual $L_2$ norm of a vector $\hat{h}$.

The algorithm described below use the dictionary functions as actual functions only when applying the learned approximation on new test data. During training, only the values at the training points are relevant, so that they can be understood as working entirely in an $l$ -dimensional vector space.

The basic $\{g_1, \ldots, g_N\} \subset \mathbf{D}$ and the corresponding coefficient $\{\alpha_1, \ldots, \alpha_N\} \subset \mathbf{R}^N$ are chosen such that they minimize the squared norm of the residue:

$$\|\hat{R}_N\|^2 = \|\hat{y} - \hat{f}_N\|^2 = \sum_{i=1}^{l} \left( \hat{y}_i - f_N(x_i) \right)^2$$  \hspace{1cm} (3)

It starts at stage 0 with $\hat{f}_0 = 0$, an recursively appends functions to an initially empty basis, at each stage $n$, trying to reduce the norm of residue:

$$\hat{R}_n = \hat{y} - \hat{f}_n$$  \hspace{1cm} (4)

Given $\hat{f}_n$, we can write:

$$\hat{f}_{n+1} = \hat{f}_n + \alpha_{n+1} \tilde{g}_{n+1}$$  \hspace{1cm} (5)

By searching for $\tilde{g}_{n+1} \in D$ and for $\alpha_{n+1} \in \mathbf{R}$ that minimize the residual error, i.e. the squared norm of the next residue:

$$\|\hat{R}_{n+1}\| = \|\hat{y} - \hat{f}_{n+1}\|^2 = \|\hat{y} - (\hat{f}_n + \alpha_{n+1} \tilde{g}_{n+1})\|^2$$  \hspace{1cm} (6)

Formally:

$$(g_{n+1}, \alpha_{n+1}) = \underset{\tilde{g} \in D, \alpha \in \mathbf{R}}{\text{argmin}} \|\hat{R}_n - \alpha \tilde{g}\|^2$$  \hspace{1cm} (7)

For any $g \in \mathbf{D}$, $\alpha$ that minimizes $\|\hat{R}_n - \alpha \tilde{g}\|^2$ is given by:

$$\frac{\partial}{\partial \alpha} \|\hat{R}_n - \alpha \tilde{g}\|^2 = 0$$

$$\Rightarrow -2 \langle \tilde{g}, \hat{R}_n \rangle + 2 \alpha \|\tilde{g}\|^2 = 0$$  \hspace{1cm} (8)

$$\Rightarrow \alpha = \frac{\langle \tilde{g}, \hat{R}_n \rangle}{\|\tilde{g}\|^2}$$

For this optimal value of $\alpha$, we have:

$$\|\hat{R}_n - \alpha \tilde{g}\|^2 = \|\hat{R}_n - \frac{\langle \tilde{g}, \hat{R}_n \rangle}{\|\tilde{g}\|^2} \tilde{g}\|^2$$

$$= \|\hat{R}_n\|^2 - 2 \frac{\langle \tilde{g}, \hat{R}_n \rangle}{\|\tilde{g}\|^2} + \left( \frac{\langle \tilde{g}, \hat{R}_n \rangle}{\|\tilde{g}\|^2} \right)^2$$  \hspace{1cm} (9)

So, $\tilde{g} \in \mathbf{D}$ that minimizes expression (7) is the one that minimize expression (9), which corresponds to $\hat{R}_n = \frac{\langle \tilde{g}, \hat{R}_n \rangle}{\|\tilde{g}\|^2}$. In other words, it is function in the dictionary whose corresponding vector is “most collinear” with the current residue.
In summary, $g_{n+1}$ that minimize expression (7) is the one that maximizes $\frac{\langle f, g_{n+1} \rangle}{\| g_{n+1} \|^2}$ and the corresponding is:

$$\alpha_{n+1} = \frac{\langle f, g_{n+1} \rangle}{\| g_{n+1} \|^2}.$$  

(10)

In this algorithm, we have to choose an appropriate criterion to decide when to stop adding new functions to the expansion. The algorithm is usually stopped when the reconstruction error $\|R_k\|$ goes below a predefined given threshold.

III THE DESCRIPTION OF ONLINE LEARNING ALGORITHM

A. Orthogonal Matching Pursuit

In this part, we will introduce a kind of orthogonal matching pursuit algorithm which is much like the basic matching pursuit. And one advantage of this algorithm is that it has a character of sparsity because this algorithm use the orthogonal basis of the dictionary collection as the basic functions which can effective decrease the numbers of the basic functions. The description of the algorithm is as follows:

Given a collection of vector of vectors $D = \{x_i\}$ in Hilbert space $H$, let us define: $V = \text{Span}\{x_i\}$ and $W = V^\perp$, $W \subset H$. We shall refer to as $D$ a dictionary, and will assume the vectors $x_n$ are normalized ($\|x_n\| = 1$). Basic matching pursuit proposed an iterative algorithm that they termed matching pursuit (MP) to construct representations of the form:

$$P_V f = \sum_n a_n x_n$$  

(11)

where $P_V$ is the orthogonal projection operator onto $V$. Each iteration of the MP algorithm results in an intermediate representation of the form:

$$f = \sum_{i=1}^k a_i x_{n_i} + R_k f = f_k + R_k f$$  

(12)

Where $f_k$ is the current approximation of $f$, and $R_k f$ is the current residual (error); The collection $D_k$ which is composed of $x_{n_i}, i = 1, 2, \ldots, k$ is called the representation collection of functions to be estimated. Using initial values of $R_0 f = f$, $R_0 f = f$ and $k = 1$, the OMP(Orthogonal Matching Pursuit) algorithm is comprised of the following steps [7]:

<table>
<thead>
<tr>
<th>Initialization: $f_0 = 0, R_0 f = f, D_0 = {}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_0 = 0, a_0 = 0, k = 0$</td>
</tr>
</tbody>
</table>

(I) Compute $\{\langle R_k f, x_{n_i} \rangle; x_{n_i} \in D \setminus D_k\}$

(II) Find $x_{n_{k+1}} \in D \setminus D_k$ such that

$$\|\langle R_k f, x_{n_{k+1}} \rangle\| \geq \beta \sup_j \|\langle R_k f, x_j \rangle\|,$$

where $1 \geq \beta > 0$

(III) If $\|\langle R_k f, x_{n_{k+1}} \rangle\| < \delta$ ($\delta > 0$), then stop;

(IV) Recorder the dictionary $D$, by applying the permutation $k + 1 \leftrightarrow n_{k+1}$

(V) Compute $\{b_n^k\}_{n=1}^{k}$ such that:

$$x_{n_{k+1}} = \sum_{n=1}^{k} b_n^k x_n + \gamma_k$$

and $\langle \gamma_k, x_n \rangle = 0, n = 1, \ldots, k$

(VI) Set $\alpha_k = \frac{\langle R_k f, x_{n_{k+1}} \rangle}{\|\gamma_k\|^2} = \frac{\langle R_k f, x_{n_{k+1}} \rangle}{\|x_{n_{k+1}}\|^2}$

$$\alpha_{n_{k+1}} = \alpha_n - a_k b_n^k, n = 1, \ldots, k$$

And update the model: $f_{k+1} = \sum_{n=1}^{k} \alpha_n x_{n_i}$,

$$R_{k+1} f = f - f_k$$

$$D_{k+1} = D_k \cup \{x_{n_{k+1}}\}$$

(VII) Set $k \leftarrow k + 1$ and repeat the steps (I)–(VII)

The $b^k_n$ in the Auxiliary formula

$$x_{k+1} = \sum_{n=1}^{k} b_n^k x_n + \gamma_k$$  

(13)

is a vector which is a solution of the equation

$$\sum_{n=1}^{k} b_n^k x_n = P_V x_{k+1}, \quad \gamma_k = P_{V^\perp} x_{k+1}$$  

(14)

In this formula, the superscript $k$ of $b^k_n$ represent the times of the iterations and the subscript $n$ of $b^k_n$ represent that $b^k_n$ is the $n$th element.

And $x_{k+1} \in D_{k+1}$ is one solution vector which maximize the formula:

$$\|\langle R_k f, x_{k+1} \rangle\|$$  

(15)
But, in many cases, we can only find a vector close to maximizing the function [8]:
\[
\langle R_k f, x_k \rangle \geq \beta \sup_j \langle R_k f, x_j \rangle
\]  \hspace{1cm} (16)

Where \( \beta \in (0, 1] \) is a coefficient.

B. Online Kernel Learning Algorithm based on Orthogonal Matching Pursuit

From the above description, we have known the OMP offline estimation algorithm, but there are still two problems in it. One is that the offline algorithm does not satisfy the request of online learning. The other is that OMP is based on the least-square which can not estimate the target function effectively when the target function has a character of strongly nonlinear. So, in the following part, we shall take the two problems into account.

One major difference of the online learning algorithm with the offline learning algorithm is that the learning samples arrive one by one and at a certain rate in the online learning algorithm, and the online algorithm learn the model step by step and there does not exist the known collection in the online learning algorithm. This problem can be solved by the orthogonal matching pursuit algorithm which could guide the online learning.

Another difference is that the online computing burden at every time step is very important to online learning algorithm. Consider it from another side, the computing burden problem can be seen as problem of the size of representation collection \( \text{D}_k \). The size of collection \( \text{D}_k \) should be small as it can, which could effectively lighten the computing burden. Fortunately, the OMP algorithm has provided the solution to this problem. So, we will follow this to develop our algorithm.

In this part, we would introduce “kernel trick” into the OMP method and modify the OMP to fit to the online learning.

First, consider a nonlinear mapping \( \phi: \mathbb{R}^n \rightarrow \mathbb{R}^s \) from the input space to some high-dimensional feature space and kernel function is defined by \( K(x_i, x_j) \).

Usually, the linear regressor can be represented as:
\[
f(x) = \langle \omega, \phi(x) \rangle + b
\]

Where \( \omega \) is the weight vector, \( \phi(x) \) is the feature vector after mapping, \( b \) is the bias of the equation.

In order to transform the linear regressor to the form which can be represented by matching pursuit algorithm, We can redefine \( \omega \) and \( \phi(x) \) by \( \bar{\phi} = (\phi^T, \lambda)^T \) and \( \bar{\omega} = (\omega, b/\lambda)^T \), the weight vector can absorb the bias \( b \), then the linear regressor can be rewritten:
\[
f(x) = \langle \omega, \phi(x) \rangle
\]

After introducing the “kernel trick” into the linear regressor, it can be represented as:
\[
f(x) = \sum_{n=1}^k \alpha^n \langle x_n, x \rangle
\]

Then, we would introduce the “kernel trick” into the OMP method. For \( b^{k+1} \) in the Auxiliary formula, we can rewrite it as follow:
\[
k_{k+1} = A_k b^{k+1}
\]  \hspace{1cm} (17)

Where \( b^{k+1} = [h_{k+1}^T, K_{k+1} h_{k+1}]^T \) and
\[
k_{k+1} = \left[ \langle x_{k+1}, x_1 \rangle, K(x_{k+1}, x_2), \ldots, K(x_{k+1}, x_n) \right]^T
\]
\[
A_k = \begin{bmatrix}
K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_n) \\
K(x_2, x_1) & K(x_2, x_2) & \cdots & K(x_2, x_n) \\
\vdots & \vdots & \ddots & \vdots \\
K(x_n, x_1) & K(x_n, x_2) & \cdots & K(x_n, x_n)
\end{bmatrix}
\]

So, the solution can be represented as:
\[
b^{k+1} = A_k^{-1} k_{k+1}
\]

\( s \) in the above formula is the size of the collection \( \text{D}_k \) at time step \( k \). Usually, \( s \) is not equal to \( k \), because we can not add observation samples at all time steps into the collection \( \text{D}_k \). So, \( s \neq k \).

Now, we shall describe the method which the OMP used to decrease the size of \( \text{D}_k \) --using the orthogonal basis of the collection to represent the collection \( \text{D}_k \).

As we say, due to the computing burden problem, the online algorithm can not add all the observation samples arrived into the collection \( \text{D}_k \) at a certain rate into the collection \( \text{D}_k \) or not, the algorithm can be separated into two cases naturally.

One is that a new sample needs to be added to the collection \( \text{D}_k \), it is to say, the element number of the collection \( \text{D}_k \) needs to be increased.

But, how to know a new sample should be added into the collection \( \text{D}_k \) ? We can concludes as follows [9][10]:

A new sample \( x_{k+1} \) is mapping by \( \phi \) and can be represented as \( \phi(x_{k+1}) \) after mapping. When the mapped sample \( \phi(x_{k+1}) \) can be represented by the vectors \( \{ \phi(x_1), \ldots, \phi(x_n) \} \) in the collection \( \text{D}_k \)
( \varphi(x_1),..., \varphi(x_s) \) and \( \varphi(x_{k+1}) \) has a certain kind of linear relationship. The variable \( \gamma_k \)

\[ \|\gamma_k\|^2 = K(x_{k+1}, x_{k+1}) - (k_{k+1})^T b_{k+1} \]

can represent this kind of linear degree, and the \( \gamma_k \) is closer to 0, the linear degree is stronger. Where
\[ k_{k+1} = \begin{bmatrix} K(x_{k+1}, x_1), K(x_{k+1}, x_s) \end{bmatrix}^T, \]

One case is that a new mapped sample \( x_{k+1} \) can be represent as \( \varphi(x_{k+1}) \). When \( \varphi(x_{k+1}) \) cannot be represented by the vectors \( \{ \varphi(x_1), ..., \varphi(x_s) \} \) in the collection \( D_k \). That is when the \( \{ \langle \gamma_k, \varphi(x_n) \rangle \} = 0 \)
\( \gamma_k \neq 0 \), the new coefficient vector can be computed as follows:

\[ \alpha^{k}_{n+1} = \alpha_{n} - \alpha_{k} b_{n} \]

Where \( b_{n} = K(x_{k+1}, x_1) \) and \( \alpha_{k} \) is the coefficient vector at the time step \( k \).

Another case is that \( \gamma_k \) close to 0 which means the \( \varphi(x_{k+1}) \) can be represented by \( \{ \varphi(x_1), ..., \varphi(x_s) \} \).

That is \( \gamma_k \approx 0 \), according to the back-fitting algorithm of the online learning methods in reference[9], we need to upgrade all the elements of the coefficient vector, it can be write as follows:

\[ M_{k+1}^T M_{k+1} A_{k+1} (\alpha^{k+1} - \alpha^k) = y_{k+1} - k_{k+1}^T b_{k+1} \]  

(18)

Where \( M_{k+1} = b_{k+1} \).

For further lightening the online computing burden in every time step, we give the iterative formulas which used to compute the inverse matrix of kernel matrix in the above two cases.

(1) A new sample needs not to be added into the collection \( D_k \), it is to say, the new mapping sample \( \varphi(x_{k+1}) \) can be represented by the vectors \( \{ \varphi(x_1), ..., \varphi(x_s) \} \) in the collection \( D_k \), there is no need to upgrade the collection \( D_k \). Just let \( A_{k+1} = A_k \).

(2) A new sample needs to be added into the collection \( D_k \), that is to say, \( A_{k+1} \neq A_k \). Using the following formula, we do not need to compute the inverse matrix of the matrix \( A_{k+1} \) and just compute \( A_{k+1}^{-1} \) iteratively which can be computed easily. This can be represented as follows:

\[ A_{k+1} = \begin{bmatrix} A_k & k_{k+1} \end{bmatrix} \]

\[ A_{k+1}^{-1} = \begin{bmatrix} \delta_{k+1} A_k^{-1} + h_{k+1} h_{k+1}^T - h_{k+1} h_{k+1}^T 1 \\
- h_{k+1} h_{k+1}^T 1 
\end{bmatrix} \]  

(20)

where \( \delta_{k+1} = b_{k+1}^T - k_{k+1} b_{k+1}^T \), \( h_{k+1} = A_1^{-1} k_{k+1} \).

In summary, the online learning algorithm based on “kernel trick” and the OMP algorithm is comprised of some steps as follows.

choose the appropriate parameters:
\( \nu, \delta \ (\delta > 0) \)

Initialization: \( A_1 = \begin{bmatrix} K(x_1, x_1) \end{bmatrix} \),
\( \alpha_1 = \begin{bmatrix} y_1/k_0^T \end{bmatrix} \)

(1) Obtain new samples \( \{ x_{k+1}, y_{k+1} \} \)

(2) If \( \langle R_k f, x_{k+1} \rangle < \nu \):

else:

compute the coefficient \( b_{k+1} \) using the following formula, \( b_{k+1} = (A_k)^{-1} k_{k+1} \),
\( n = 1,..., s \), then compute the \( \|\gamma_k\|^2 \) using the formula:
\[ \|\gamma_k\|^2 = K(x_{k+1}, x_{k+1}) - (k_{k+1})^T b_{k+1}^T \]
If \( \|\gamma_k\|^2 < \delta \):

Use the formula (19) to compute the \( \alpha^{k+1} \), and upgrade the
\( (A_{k+1})^{-1} = (A_k)^{-1} \), and let \( D_{k+1} = D_k \)
else:

compute the \( \alpha^{k+1} \) using the formula (18) and upgrade the \( (A_{k+1})^{-1} \) using the formula (20), and let \( D_{k+1} = D_k \cup \{ x_{k+1} \} \).
C. The Convergence of Our Algorithm

As in the case of OMP, convergence of our algorithm relies on an energy conservation equation that now takes
the form $\left\|R_k f\right\|^2 = \left\|R_{k+1} f\right\|^2 + \left(\left\langle R_k f, x_{k+1} \right\rangle\right)^2$. Our
algorithm has the same convergence properties with the OMP algorithm. The proof of the convergence parallels
the proof of the convergence properties in [7].

IV. EXPERIMENT

In this section, the algorithm we proposed will be tested in the MATLAB environment to evaluate the
performance of it.

We select the two-dimensional “sinc” function

$$y = \frac{\sin x}{x_1} + \frac{x_2}{10}$$

as the target function of performance test, and the proposed algorithm would be compared with well-known
SVMTorch algorithm.

In the first experiment, we will select random training samples of 300 noise-free points with uniformly
distributed on the domain $[-10,10]^2$, and the testing
was performed on another set of random samples of
1000 noise-free points on the same domain with the
training set. The parameters of our algorithm is $\delta = 0.0001$, $C = 1000$, $\nu = 0.001$, Gaussian kernel
is used for both algorithm and the bandwidth parameters
are $\sigma = 4.25$ for both algorithms, $\varepsilon$-insensitive of
SVMTorch select $\varepsilon = 0.001$, and all other parameters
of SVMTorch are at their default values. Estimation
results of both algorithms are as follows:

![Figure 1: Estimation result of our algorithm](image1)

![Figure 2: Estimation result of SVMTorch](image2)

The following table 1 is the comparison results of the
two algorithms in the first experiment.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>Training r.m.s.e</th>
<th>Number of SVs</th>
<th>Maximal testing error</th>
<th>Testing r.m.s.e</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMTorch</td>
<td>0.0029</td>
<td>199</td>
<td>0.04091</td>
<td>0.0049</td>
</tr>
<tr>
<td>Our algorithm</td>
<td>0.0014</td>
<td>84</td>
<td>0.0568</td>
<td>0.0042</td>
</tr>
</tbody>
</table>

From the results of the first experiments, we can conclude that both algorithms can estimate the two-
dimensional “sinc” function efficiently, and the errors of the estimation of both algorithms have little difference.

To further test the performance, we do another
experiment to compare the two algorithms on the percentage of Support Vectors(SVs) and the r.m.s.e.

Testing function is still the two-dimensional “sinc”
function $y = \frac{\sin x}{x_1} + \frac{x_2}{10}$, but the training data is
 corrupted by Normal distribution noise $\xi$, with $E\xi = 0$, $E\xi^2 = 0.01$, and the learning was
performed on an independent random samples generated
on the interval $[-10,10]^2$ and the number of the training
samples are from 10 to 30000. Testing is still performed
on a random set of samples of 1000 noise free points.

The kernel functions of both algorithms are still
Gaussian function with standard deviation $\sigma = 4.25$.
The SVMTorch parameters are $C = 10^7$ and $\varepsilon = 0.1$, while the parameter of our algorithm is $\delta = 0.001$, $\nu = 0.1$.

We plotted the generalization error and the number of
support vectors as a percentage of the training set. The horizontal axis is scaled logarithmically (based 10). In the
generalization error graph, we use a similar scale for the vertical axis, while on the SV percentage graph we use a linear scale. The comparison results of the two algorithms on percentage of SVs are shown in Figure 3. And the comparison results of the two algorithms on the root mean square error are shown in Figure 4.

As shown in Figure 3 and figure 4, we can conclude that as the training samples increases, a corresponding reduction in test r.m.s.e, and our algorithm have no significant difference with SVMTorch in prediction accuracy, but our algorithm has less support vector than SVMTorch.

V. CONCLUSION

We have shown how the OMP works and how the OPM was extended to a kind of online kernel learning algorithm based on orthogonal matching pursuit. And we also provided experimental evidence that the online learning algorithm we proposed is effective. We compare our algorithm with the well-known SVMTorch on SV percent and r.m.s.e which shown that the r.m.s.e. of the algorithm we proposed in estimating the two-dimensional “sinc” function has no significant difference with the SVMTorch, but the percentage of SVs is greatly less than the SVMTorch.

Shilei Zhao Harbin city, China. Birth date: December, 1979. is a lecturer in School of Software, Harbin University of Science and Technology. He is a Ph.D. and graduated from College of Astronaut, Harbin institute of Technology. And research interests on SVM and kernel machine.

Peng Wu Harbin city, China. Birth date: July, 1980, is a lecturer in lecturer in College of Mechanical and Electrical Engineering, Northeast Forestry University. He is a Ph.D. and graduated from College of Astronaut, Harbin institute of Technology.

YuPeng Liu Harbin city, China. is a lecturer in lecturer in School of Software, Harbin University of Science and Technology. He is a Ph.D. and graduated from Harbin institute of Technology.

REFERENCE