SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2016, Vol. 59 052103:1–052103:15
doi: 10.1007/s11432-016-5536-6

Learning capability of the truncated greedy algorithm

Lin XU[1], Shaobo LIN[2]* & Zongben XU[1]

[1]*Institute for Information and System Sciences, Xi'an Jiaotong University, Xi'an* 710049*, China;*
[2]*College of Mathematics and Information Science, Wenzhou University, Wenzhou* 325035*, China*

Received September 30, 2015; accepted November 6, 2015; published online April 8, 2016

Abstract Pure greedy algorithm (PGA), orthogonal greedy algorithm (OGA) and relaxed greedy algorithm (RGA) are three widely used greedy type algorithms in both nonlinear approximation and supervised learning. In this paper, we apply another variant of greedy-type algorithm, called the truncated greedy algorithm (TGA) in the realm of supervised learning and study its learning performance. We rigorously prove that TGA is better than PGA in the sense that TGA possesses the faster learning rate than PGA. Furthermore, in some special cases, we also prove that TGA outperforms OGA and RGA. All these theoretical assertions are verified by both toy simulations and real data experiments.

Keywords supervised learning, learning theory, generalization capability, greedy algorithm, truncated greedy algorithm

Citation Xu L, Lin S B, Xu Z B. Learning capability of the truncated greedy algorithm. Sci China Inf Sci, 2016, 59(5): 052103, doi: 10.1007/s11432-016-5536-6

1 Introduction

Greedy learning or, more specifically, applying greedy-type algorithms in machine learning has triggered considerable research activities in the past decades [1–7]. Greedy-type algorithms are stepwise inference processes that follow the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum. Compared with standard coefficient-based regularized methods in dictionary learning, greedy-type algorithms can greatly reduce the computational burden [1] when the capacity of the dictionary is large.

The study of greedy-type algorithms could date back to 1906, when Schmidt [8] provided the known Schmidt expansion for bilinear approximation. It was understood [8] (or [9, Page 306]) that the best bilinear approximation can be realized by the pure greedy algorithm (PGA). PGA possesses the dominance in computation, but the main defect is that, for a general dictionary, its convergence rate is far slower than the best nonlinear approximant [9–12] and consequently corrupts its learning performance, especially when atoms of the dictionary are highly correlated. Under this circumstance, several modified versions of PGA have been proposed to accelerate the approximation rate and consequently, to improve

* Corresponding author (email: sblin1983@gmail.com)

the generalization capability. Typical examples [1] include the orthogonal greedy algorithm (OGA) and relaxed greedy algorithm (RGA).

OGA [10] improves the approximation capability of PGA via building the best approximant from the sub-space spanned by all the selected atoms rather than only one atom, which is utilized in PGA. It can be found in [1] that the approximation rate of PGA can be essentially improved by this method. Particularly, if the target function satisfies a certain $\mathcal{L}_1$-sparsity constraint [10], OGA can reach the optimal approximation rate. RGA [10] promotes the approximation capability of PGA via introducing a relaxation operator on the deduced estimator in each greedy iteration. The core idea [13] is that if the approximation effect of the k-th iteration is not good, then the deduced estimator is regarded to be too aggressive and therefore needs relaxing within a certain extent. It was also proved in [1, 13] that RGA can boost convergence rate of PGA to the optimal approximation rate. Both OGA and RGA were successfully used in machine learning and an $\mathcal{O}(m^{-1/2}\log m)$ learning rate was deduced for them, where m is the number of samples. We refer the readers to [1–3, 5, 7] for more details of the OGA and RGA learning.

According to the well-known bias and variance balance principle in learning theory [14], the generalization error of a learning scheme can be divided into the approximation error (bias) and sample error (variance). Reducing either the bias or the variance can improve the generalization capability of this learning scheme. As far as the PGA is concerned, both RGA and OGA focus on reducing its bias and therefore successfully improve its learning performance. In the present paper, we drive a totally different direction to improve the learning capability of PGA. To this end, we borrow the idea from the "regularization boosting via truncation" from [4, 15] in statistics and the "coefficient-restriction greedy-type algorithm" [9, Subsection 6.7] in non-linear approximation to tackle supervised learning problems. For the sake of brevity, we rename the proposed algorithm as the truncated greedy algorithm, since it truncates the step size of PGA at a specified value in each greedy itetation to cut down the model complexity. The main novelty of this paper is to prove TGA can essentially improve the learning performance of PGA.

We present both theoretical analysis and experimental verification to illustrate the learning performance of TGA. Theoretically, we prove that TGA can cut down the model complexity ($\mathcal{L}^1$ norm) of PGA without sacrificing its approximation capability. Therefore, TGA can reduce the variance and consequently, the generalization error of the PGA learning. To be detailed, we prove that for some specified learning tasks, TGA attains a new "record" of greedy learning by omitting the logarithmic factor in the learning rates of the OGA and RGA learning. This means that, different from the traditional variants RGA and OGA, TGA provides a new competitive way to modify PGA. Experimentally, we employ both toy simulations and real data experiments to illustrate the outperformance of TGA. In the toy simulations, we show that in some standard simulation setting [16], TGA is comparable with the existing popular schemes such as the OGA, RGA, Lasso [17] and ridge regression [18]. Furthermore, we also find that for some specified learning tasks, TGA performs much better than PGA. In the real data experiments, we run TGA, RGA, OGA and PGA on five data sets with two types of dictionaries and show that TGA essentially outperforms PGA and is comparable with other two algorithms. All the experimental results are consistent with the theoretical assessments and therefore verify our assertions.

The rest of this paper is organized as follows. In Section 2, we give a brief introduction to PGA and TGA. In Section 3, we study the theoretical behaviors of TGA. In Section 4, we present a series of toy simulations and real data experiments to illustrate our theoretical assertions. In Section 5, we provide the proofs of the main results. In the last section, we draw a simple conclusion.

2 Truncated greedy learning

In this section, we introduce some concrete greedy learning schemes for regression. In a regression problem [14] with a covariate x on $X \subseteq \mathbb{R}^d$ and a real response variable $y \in Y \subseteq \mathbb{R}$, we observe m i.i.d. samples $\boldsymbol{z} = \{(x_i, y_i)\}_{i=1}^m$ from an unknown underlying distribution ρ. Without loss of generality, we always assume $Y \subseteq [-M, M]$, where $M < \infty$ is a positive real number. The aim is to find a function to

minimize the generalization error

$$\mathcal{E}(f) = \int (f(x) - y)^2 \mathrm{d}\rho.$$

The known regression function [14]

$$f_\rho(x) = \int_Y y \mathrm{d}\rho(y|x)$$

minimizes the generalization error. In such a setting, we are interested in finding a function $f_{\boldsymbol{z}}$ based on $\boldsymbol{z}$ such that $\mathcal{E}(f_{\boldsymbol{z}}) - \mathcal{E}(f_\rho)$ is small.

Let $\mathcal{D}_n := \{g_1, \ldots, g_n\}$ be the set of atoms (or dictionary) and define

$$\mathrm{span}(\mathcal{D}_n) = \left\{ \sum_{j=1}^n a_j g_j : g_j \in \mathcal{D}_n, a_j \in \mathbb{R} \right\}.$$

Define the empirical norm and empirical inner product by

$$\|f\|_m = \sqrt{\frac{1}{m}\sum_{i=1}^m f(x_i)^2} \quad \text{and} \quad \langle f, g\rangle_m = \frac{1}{m}\sum_{i=1}^m f(x_i)g(x_i),$$

respectively. We also define the empirical risk as

$$\mathcal{E}_{\boldsymbol{z}}(f) = \frac{1}{m}\sum_{i=1}^m (f(x_i) - y_i)^2.$$

Then the pure greedy algorithm (PGA) [10] can be described as Algorithm 1.

Algorithm 1 PGA learning

Step 1 (Initialization): Given data $\boldsymbol{z} = \{(x_i, y_i) : i = 1, \ldots, m\}$, dictionary $\mathcal{D}_n$, and $k^* \in \mathbb{N}$, $f_{\boldsymbol{z},0} = 0$, $r_{\boldsymbol{z},0} = y$, $k = 1$.
Step 2 (Projection of gradient to learner): Find $g_{\boldsymbol{z},k} \in \mathcal{D}_n$ such that

$$g_{\boldsymbol{z},k} = \arg\max_{g\in\mathcal{D}_n} |\langle r_{\boldsymbol{z},k-1}, g\rangle_m|,$$

where $r_{\boldsymbol{z},k-1}(\cdot) = y(\cdot) - f_{\boldsymbol{z},k-1}(\cdot)$ is the $(k-1)$-th residual, and $y(\cdot)$ satisfies $y(x_i) = y_i$.
Step 3 (Linear search): Find $a_{\boldsymbol{z},k} \in \mathbb{R}$ such that

$$a_{\boldsymbol{z},k} = \arg\min_{a\in\mathbb{R}} \|r_{\boldsymbol{z},k-1} - a g_{\boldsymbol{z},k}\|_m^2.$$

It is readily seen that $a_{\boldsymbol{z},k}$ is given explicitly by

$$a_{\boldsymbol{z},k} = \langle r_{\boldsymbol{z},k-1}, g_k\rangle_m.$$

Update $f_{\boldsymbol{z},k} = f_{\boldsymbol{z},k-1} + a_{\boldsymbol{z},k} g_{\boldsymbol{z},k}$.
Step 4 (Iteration): Increase k by one and repeat Step 2 to Step 4 until $k = k^*$. Output $f_{\boldsymbol{z},k^*}$.

Remark 1. Suppose the total iteration number is set as K, then the overall complexity of PGA learning is $\mathcal{O}(Kmn)$ and the memory required for the naive approach is $\mathcal{O}(mn)$.

Despite that PGA learning was proved to be consistent [1,10], which can be easily deduced by applying the method in [1] to [12, Theorem 1], a number of studies [10–13] also showed that its approximation rate is far slower than that of the best nonlinear approximant. The main reason is that the linear search in Algorithm 1 makes $f_{\boldsymbol{z},k+1}$ be not always the greediest one [19]. In particular, as shown in Figure 1, when $f_{\boldsymbol{z},k}$ walks along $g_{\boldsymbol{z},k}$'s direction to fit the residual $r_{\boldsymbol{z},k-1}$, after $a^t_{\boldsymbol{z},k} g_{\boldsymbol{z},k}$, there exists at least one atom $g_{\boldsymbol{z},k+1}$ which is more relevant to the updated residual (i.e., the angle $\alpha \leqslant \beta$) than $g_{\boldsymbol{z},k}$. Thus, after $a^t_{\boldsymbol{z},k} g_{\boldsymbol{z},k}$, continuing to walk along $g_{\boldsymbol{z},k}$ is no more the greediest choice (i.e., the truncated $(k+1)$-th residual $r^t_{\boldsymbol{z},k+1}$ is less than the original residual $r_{\boldsymbol{z},k+1}$). Under this circumstance, we suppose that if we truncate the linear search between $a^t_{\boldsymbol{z},k} g_{\boldsymbol{z},k}$ and $a_{\boldsymbol{z},k} g_{\boldsymbol{z},k}$, then the approximation capability of PGA at least does not degrade. This motivates our study of the truncated greedy algorithm (TGA).

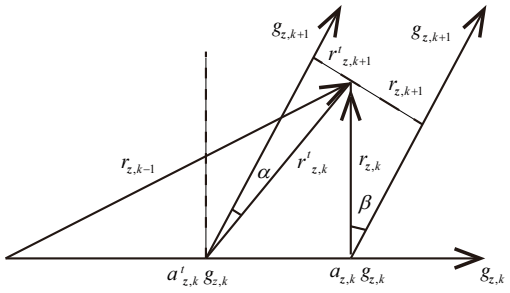

Figure 1 The drawback of PGA.

The core idea of TGA is to truncate the linear search in a bounded interval instead of the real axis in each iteration. That is, TGA introduces a restriction to the step size of linear search and consequently cuts down the model complexity. The following Algorithm 2 depicts the scenario of TGA.

Algorithm 2 TGA learning

Step 1 (Initialization): Given data $\boldsymbol{z} = \{(x_i, y_i) : i = 1, \ldots, m\}$, dictionary $\mathcal{D}_n$, a set of closed subsets $\Lambda_k = [-h_k, h_k]$, $k = 1, 2, \ldots$, and $k^* \in \mathbb{N}$, $f^t_{\boldsymbol{z},0} = 0$, $r^t_{\boldsymbol{z},0} = y$, $k = 1$.

Step 2 (Projection of gradient to learner): Find $g^t_{\boldsymbol{z},k} \in \mathcal{D}_n$ such that

$$g^t_{\boldsymbol{z},k} = \arg\max_{g\in\mathcal{D}_n} |\langle r^t_{\boldsymbol{z},k-1}, g\rangle_m|,$$

where $r^t_{\boldsymbol{z},k-1}(\cdot) = y(x) - f^t_{\boldsymbol{z},k-1}(x)$ is the $(k-1)$-th residual.

Step 3 (Linear search): Find $a_{\boldsymbol{z},k} \in \mathbb{R}$ such that

$$a^t_{\boldsymbol{z},k} = \arg\min_{a\in\boldsymbol{\Lambda}_k} \|r^t_{\boldsymbol{z},k-1} - a g^t_{\boldsymbol{z},k}\|^2_m.$$

That is,

$$a^t_{\boldsymbol{z},k} = \operatorname{sign}(\langle r^t_{\boldsymbol{z},k-1}, g^t_k\rangle_m) \min\{|\langle r^t_{\boldsymbol{z},k-1}, g^t_{\boldsymbol{z},k}\rangle_m|, h_k\},$$

where $y(\cdot)$ satisfies $y(x_i) = y_i$. Update $f^t_{\boldsymbol{z},k} = f^t_{\boldsymbol{z},k-1} + a^t_{\boldsymbol{z},k} g^t_{\boldsymbol{z},k}$.

Step 4 (Iteration): Increase k by one and repeat Step 2 to Step 4 until $k = k^*$. Output $f^t_{\boldsymbol{z},k^*}$.

Remark 2. In the Step 1 of Algorithm 2, h_k is referred as the truncated parameter, which is chosen via some parameter-selection methods generally. In the present paper, we just set $h_k \sim k^{-2/3}$ and verify its feasiblity.

Remark 3. If the truncated parameter is fixed (i.e., $h_k = k^{-2/3}$), then the overall complexity of TGA learning is $\mathcal{O}(Kmn)$, which is the same as that of PGA.

3 Theoretical behaviors of TGA

Before giving the main result, we need to introduce a few notations. Let $s \in (0, 1]$, and $\phi(x^*, x)$ be a continuous function on $X \times X$ such that for all $x^*, x, x' \in X$, there holds

$$|\phi(x^*, x) - \phi(x^*, x')| \leqslant C_s |x - x'|^s, \quad \max|\phi(x^*, x)| \leqslant 1. \tag{1}$$

Here C_s is a positive constant depending only on s. We remark that the first inequality in (1) is simply the Lipchitz continuity of ϕ with respect to x, and that the second inequality is essentially the boundedness of ϕ with respect to both x and x^*. The examples of such functions are abundant [20]. For instance, the widely used Gausssian kernel $G(x, y) = \exp\{-\frac{|x-y|^2}{a}\}$ with $a > 0$ fulfills the assumption (1) with $s = 1$. For a given set of n distinct points $t_1, \ldots, t_n \in X$, let

$$\mathcal{D}_n = \{g_1, \ldots, g_n\} := \{\phi(t_i, \cdot), i = 1, 2, \ldots, n\}, \tag{2}$$

and $\mathcal{D} := \{\phi(x, \cdot) : x \in X\}$.

Let

$$\mathcal{L}_1(\mathcal{D}) := \left\{ f : f = \sum_{g\in\mathcal{D}} a_g g, \|f\|_{\mathcal{L}_1(\mathcal{D})} < \infty \right\}$$

endowed with the norm

$$\|f\|_{\mathcal{L}_1(\mathcal{D})} := \inf \left\{ \sum_{g\in\mathcal{D}} |a_g| : f = \sum_{g\in\mathcal{D}} a_g g \right\}.$$

According to (1), we have $\|g\| \leqslant 1$ for all $g \in \mathcal{D}$, where $\|\cdot\|$ denotes the uniform norm for the continuous function space $C(X)$. For $r > 0$, the space $\mathcal{L}^r_{\mathcal{D}_n}$ is defined to be the set of all functions f such that, there exists $h \in \mathrm{span}\{\mathcal{D}_n\}$ such that

$$\|h\|_{\mathcal{L}_1(\mathcal{D})} \leqslant \mathcal{B} \text{ and } \|f - h\| \leqslant \mathcal{B}n^{-r}. \tag{3}$$

The infimum of all $\mathcal{B}$ defines a norm for f on $\mathcal{L}^r_{\mathcal{D}_n}$. It follows from [1] that Eq. (3) defines an interpolation space which has been widely used in nonlinear approximation and greedy learning [1, 5, 7, 13].

Let $\pi_M t$ denote the clipped value of t at $\pm M$, that is, $\pi_M t := \min\{M, |t|\}\mathrm{sgn}(t)$. Then it is obvious [21] that, for all $y \in [-M, M]$, there holds

$$\mathcal{E}(\pi_M f) - \mathcal{E}(f_\rho) \leqslant \mathcal{E}(f) - \mathcal{E}(f_\rho).$$

Actually, the clipped technique has been widely used in the greedy learning [1, 7]. The following theorem is the main result of this section, whose proof will be postponed to Section 5.

Theorem 1. Let $\delta \in (0, 1)$, $\mathcal{D}_n$ and $f^t_{\boldsymbol{z},k}$ be defined in (2) and Algorithm 2 with $h_k \sim k^{-2/3}$, respectively. If $f_\rho \in \mathcal{L}^r_{\mathcal{D}_n}$ and (1) holds, then with confidence at least $1 - \delta$,

$$\mathcal{E}(\pi_M f^t_{\boldsymbol{z},k}) - \mathcal{E}(f_\rho) \leqslant C\mathcal{B}^2 \left(\log \frac{2}{\delta} k^{\frac{d}{3d+3s}} m^{-\frac{2s+d}{2d+2s}} + k^{-\frac{1}{3}} + n^{-2r} \right),$$

where C is a constant depending only on ϕ, d and M.

Let us give some remarks of Theorem 1. If the size of dictionary, n, is sufficiently large and $k \sim m^{\frac{6s+3d}{4d+2s}}$, then we can deduce a learning rate of $f^t_{\boldsymbol{z},k}$ asymptotically as $\mathcal{O}(m^{-\frac{2s+d}{4d+2s}})$. Especially, when $s = d = 1$ (i.e., for fitting one dimension data), we deduce a learning rate of $f^t_{\boldsymbol{z},k}$ asymptotically as $\mathcal{O}(m^{-1/2})$, which is the new "record" of greedy learning by omitting the logarithmic factor in the learning rate of the OGA and RGA learning [1, Corollary 3.7]. For more general cases, the learning rate $\mathcal{O}(m^{-\frac{2s+d}{4d+2s}})$ is better than $\mathcal{O}(m^{-1/4})$ for arbitrary d and s, which is the existing optimal learning rate for the PGA learning [1)].

We should also present the fact that the $\mathcal{L}_1$ norm of the estimate $f^t_{\boldsymbol{z},k}$ deduced from TGA is smaller than that of PGA. This is the main reason why TGA can improve the learning capability of PGA. It can be found in Algorithm 2 that in each step, the step size of TGA is not greater than $h_k \sim k^{-2/3}$. Thus, the $\mathcal{L}_1$ norm of the final estimate of TGA is at most $\mathcal{O}(k^{1/3})$. For PGA, as there is no restriction to the step size, we can deduce the $\mathcal{L}_1$ norm of its estimate $f_{\mathbf{z},k}$ by the Hölder inequality and the following inequality [12, Theorem 1]:

$$\|f - f_{\boldsymbol{z},k}\|^2_m \leqslant \|f - h\|^2_m + \frac{27}{2}\|h\|^2_{\mathcal{L}_1(\mathcal{D}_n)} k^{-1/3+\varepsilon},\ k = 1, 2, \ldots, \tag{4}$$

with $\varepsilon \in (0, 1/6)$. Due to (1) and the triangle inequality, we can derive

$$|a_{\boldsymbol{z},k}| = |\langle r_{\boldsymbol{z},k-1}, g_k\rangle_m| \leqslant \|f - f_{\boldsymbol{z},k-1}\|_m.$$

It then follows from (4) that

$$|a_{\boldsymbol{z},k}| \leqslant \|f - f_{\boldsymbol{z},k-1}\|_m \leqslant \|f - h\|_m + \frac{3\sqrt{6}}{2}\|h\|_{\mathcal{L}_1(\mathcal{D}_n)} k^{-1/6}.$$

1) This rate can be deduced directly by combining the methods in the proof of Theorem 3.1 in [1] and Theorem 1 in [12].

Hence, the best case in PGA is that the $\mathcal{L}_1$ norm of the final estimate satisfies $\sum_{i=1}^{k}|a_{\boldsymbol{z},k}| = \mathcal{O}(k^{5/6})$, which is larger than that of TGA. Although, the estimate of the $\mathcal{L}_1$ norm of PGA can be possibly reduced within a certain extent by using a tight approximation error as $\mathcal{O}(k^{-0.182})$ [9], to the best of our knowledge, its value can not achieve $\mathcal{O}(k^{1/3})$.

It can be found in [1, 7] that both OGA and RGA can reach the learning rate as $\mathcal{O}(m^{-1/2})$ (up to a logarithmic factor), but the learning rate built in Theorem 1 is slower than $\mathcal{O}(m^{-1/2})$, expect for some special cases. We conjecture that via suitably selected h_k (may be smaller than the specified condition: $h_k = \mathcal{O}(k^{-2/3})$, TGA can also reach this bound. This may depend on some more delicate tools in statistics and approximation theory.

4 Numerical results

In this section, a series of toy simulations and real data experiments are carried out to verify our theoretical assertions. All numerical studies are implemented by using MATLAB R2014a on a Windows personal computer with Core(TM) i7-3770 3.40 GHz CPUs and RAM 4.00 GB, and the output results are averaged based on 20 independent trails for each simulation.

4.1 Simulation experiments

In this part, three groups of toy simulations are carried out. In the first two groups, we compare TGA with other popular schemes such as PGA, RGA, OGA, LASSO, and ridge regression (RR) in the same supervised learning problems. In the third group, we only compare the performance of TGA with that of PGA to highlight the outperformance of TGA over PGA. We generate the data sets from the following model [16]:

$$y = m(x) + \varepsilon,$$

where ε is the Gaussian noise $N(0, \sigma^2)$ and independent of x and x is uniformly distributed on $[-2, 2]^d$ with $d \in \{1, 10\}$. Three typical regression functions are considered

$$m_1(x) = 0.1\mathrm{e}^{-\|x-t_1\|^2} + 0.2\mathrm{e}^{-\|x-t_2\|^2} + 0.3\mathrm{e}^{-\|x-t_3\|^2} + 0.4\mathrm{e}^{-\|x-t_4\|^2}$$

with t_i being drawn according to the uniform distribution in $[-2, 2]$,

$$m_2(x) = \cos(x) + 2\cos(17 \times x) + 0.7\cos(101 \times x),$$

and

$$m_3(x_1, \ldots, x_{10}) = \sum_{j=1}^{10}(-1)^{j-1}x_j \sin({x_j}^2).$$

For each regression function, we first generate a training set of size $m = 1000$ and an independent test set, including $m' = 1000$ noiseless observations, and then evaluate the generalization capability of each algorithm in terms of the root mean squared error (RMSE). We adopt three different types of function sets to build the dictionaries. For regression function m_1, we use the dictionary as the set of functions of the form $\mathcal{D}_n^1 = (1, \cos(\pi x), \cos(2\pi x), \ldots, \cos(n\pi x))$ with $n = 500$. As far as m_2 is concerned, we employ $\mathcal{D}_n^2 = \mathrm{e}^{-\|x-\xi_i\|^2}$, $n = 1000$ to be the dictionary, where the centers ξ_i are 1000 points drawn according to the uniform distribution in $[-2, 2]$. Turning to the regression function m_3, we utilize the CART [22] (with the number of splits $J = 4$) to build up the dictionary.

It is known that the greedy-type algorithm requires specification of the number of iterations. A suitable value of the number of iterations can range from a few dozen to several thousand, depending on the algorithm itself and which data set we used. Since we mainly focus on the comparisons of the learning performances among the aforementioned algorithms, a preferable way is to select the theoretically optimal number of iterations via the test data set. More precisely, we select the number of iterations, $k^* \in [0, 10000]$, as the best one according to the test data set directly. Furthermore, for the additional

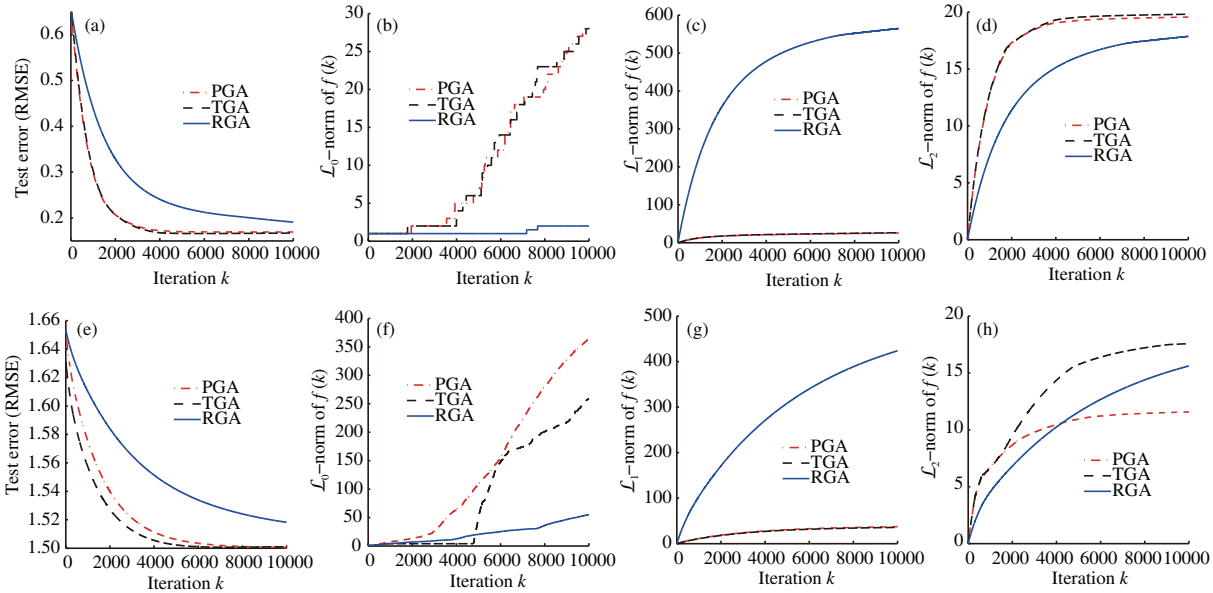

Figure 2 (Color online) Relations between the test error, $\mathcal{L}_0$, $\mathcal{L}_1$, $\mathcal{L}_2$ norms of the PGA (dotted line), TGA (septal line), RGA (solid line) estimates and the number of iterations for m_1 and m_2, where (a), (b), (c) and (d) are the empirical results of m_1 and (e), (f), (g) and (h) are the results of m_2.

introduced parameter, we set the relaxation parameter in RGA to be $\alpha_k = 1 - 1/k$ as in [1], and the truncated parameter in TGA to be $h_k = k^{-2/3}$. Concerning the regularized parameter in RR and LASSO, we logarithmically draw 20 equally spaced values of λ between 10^{-5} to 10^3.

For m_1 and m_2, Figure 2 illustrates how the test error ((a) and (e)), $\mathcal{L}_0$ norm of the estimate ((b) and (f)), $\mathcal{L}_1$ norm of the estimate ((c) and (g)) and $\mathcal{L}_2$ norm of the estimate ((d) and (h)) vary with the number of iterations in PGA, TGA and RGA. The results imply that for arbitrary iteration number, both the test error and $\mathcal{L}_1$ norm of the TGA estimate are slightly smaller than those of PGA and RGA[2)]. The aim to introduce such simulations with such a weak implication is to clarify that in the general case, TGA is a reliable choice as other popular learning schemes.

Table 1 summarizes the expected performance based on 20 independent simulations of the TGA, PGA, RGA, OGA, RR and LASSO for m_1 and m_2. OptRMSETest denotes the theoretically optimal mean rooted square test error (the standard errors of the OptRMSETest are also recorded in the parentheses) and OptParameter denotes the theoretically optimal parameter such as the number of iterations, k^*, for greedy-type algorithms TGA, PGA, RGA, OGA and the regularized parameter λ^* for RR and LASSO. $\mathcal{L}_0$ norm (or sparsity) depicts the average number of atoms in the dictionary used to construct the final estimate. $\mathcal{L}_1$ norm and $\mathcal{L}_2$ norm show the corresponding average magnitudes of the coefficient used to construct the final estimate. Time represents the average training time (in seconds) of running one simulation.

From the above simulations, we can summarize the following conclusion. Firstly, just as Table 1 described, greedy learning algorithms (TGA, PGA, RGA, OGA) possess charming generalization capability with less computation time than coefficient-based regularized method (RR, LASSO), especially in large capacity ($n = 1000$) dictionary learning. Secondly, there also exists distinct difference among TGA, PGA, RGA and OGA. At first, OGA can finish the learning with a small number of iterations. However, the generalization capability of OGA is very sensitive and the computational complexity of OGA in each iteration is much more than other greedy-type algorithms. This inevitably leads to a relatively high computational burden of OGA. Secondly, compared with PGA and RGA, TGA always possesses a good structure. To be detailed, all the $\mathcal{L}_0$ norm, $\mathcal{L}_1$ norm and $\mathcal{L}_2$ norm of TGA are relatively small, which is not witnessed in PGA (with a large $\mathcal{L}_0$ norm) and RGA (with a large $\mathcal{L}_1$ norm). Summarily, if we

2) As OGA is sensitive to iteration k, we cannot plot its curve in the scale of Figure 2 and thus omit it.

Table 1 Comparisons of TGA, PGA, RGA, OGA, RR and LASSO on simulated regression data sets for m_1 and m_2

Methods	OptRMSETest	OptParameter	$\mathcal{L}_0$ norm (Sparsity)	$\mathcal{L}_1$ norm	$\mathcal{L}_2$ norm	Time
Regression function m_1, dictionary $\mathcal{D}_n^1$, noise level $\sigma^2 = 0.1$						
TGA	0.1661(0.0031)	5790	13.3	23.1	19.6	19.5
PGA	0.1694(0.0036)	8631	23	24.8	19.6	19.7
RGA	0.1910(0.0051)	10000	2	564.8	17.9	19.9
OGA	0.1609(0.0039)	2	2	23.6	20.2	64.3
RR	0.3509(0.0014)	0.1624	500	175.4	17.6	20.1
LASSO	0.2610(0.0063)	0.1593	350.4	128.0	21.5	22.1
Regression function m_2, dictionary $\mathcal{D}_n^2$, noise level $\sigma^2 = 0.1$						
TGA	1.5000(0.0171)	7325	180	33.5	16.9	50.5
PGA	1.5005(0.0189)	9998	364.6	37.3	11.6	61.3
RGA	1.5182(0.0214)	10000	55	423.8	15.6	69.1
OGA	1.5023(0.0159)	7	7	49.8	21.2	167.8
RR	1.5235(0.0121)	1e−5	1000	1.1e+04	467.9	170.1
LASSO	1.5011(0.0152)	0.0121	372	1.0e+03	364.3	338.3

Table 2 Expected performance comparison of TGA and PGA on simulated regression data sets for m_3 (10-dimension cases and multiple noise level)

Methods	OptRMSETest	$\mathcal{L}_1$ norm	$\mathcal{L}_2$ norm
Regression function m_3, dictionary CART			
Noise level $\sigma^2 = 0$			
TGA	1.8388(0.1102)	16.0	1.3
PGA	2.3393(0.1112)	299.1	17.2
Noise level $\sigma^2 = 0.3$			
TGA	1.8380(0.0830)	16.1	1.3
PGA	2.4051(0.1112)	298.9	17.3
Noise level $\sigma^2 = 0.6$			
TGA	1.9628(0.0853)	16.0	1.3
PGA	2.4350(0.0836)	299.0	17.3
Noise level $\sigma^2 = 1$			
TGA	2.1575(0.0891)	16.0	1.3
PGA	2.6583(0.1103)	298.6	17.3

consider the computational burden, structures and test error, simultaneously, TGA is a preferable choice among all the aforementioned schemes.

In the previous simulations, we do not highlight the outperformance of TGA over PGA in terms of the generalization capability, as shown in Theorem 1. And in the third toy simulation, we only compare PGA with TGA to illustrate that TGA can bring a significant performance improvement for some specified learning tasks. Table 2 records the comparison of the TGA and PGA on simulated data sets m_3. From the results, we can distinctly detect that, the performance of TGA surpasses that of PGA with a large extent. Here we do not compare the performance of TGA with other greedy learning schemes including RGA and OGA, because the main purpose of this simulation is to highlight the outperformance of the proposed TGA over the PGA for more general cases (i.e., for fitting 10-dimension data).

4.2 Real data experiments

We have verified that TGA outperforms PGA on the 2+4 = 6 different distributions in the previous toy simulations. We now further empirically compare the learning performances of TGA, PGA, RGA and OGA on five real data sets.

The first dataset is the Diabetes data set [19]. This data set contains 442 diabetes patients that were measured on ten independent variables, i.e., age, sex, body mass index, etc., and one response variable, i.e., a measure of disease progression. The second one is the Prostate cancer data set derived from a

Table 3 Performance comparison results of TGA and PGA on aforementioned five real data sets

Methods	Datasets				
	Diabetes	Prostate	Housing	CCS	Abalone
			Decision stumps		
TGA	56.5549(1.0703)	0.2643(0.0417)	4.2565(0.4629)	5.9421(0.2456)	2.2589(0.0419)
PGA	60.2129(1.7172)	0.3517(0.0823)	4.4683(0.3715)	6.2140(0.2651)	2.3691(0.0399)
RGA	55.4940(1.1551)	0.2449(0.0555)	4.1887(0.4344)	5.5895(0.2038)	2.1982(0.0502)
OGA	59.6912(1.4950)	0.3567(0.0541)	4.6887(0.3132)	5.6407(0.1572)	2.2720(0.0632)
			Vanilla neural networks		
TGA	57.8423(2.2464)	0.5403(0.1479)	4.3866(0.8436)	6.6445(0.2740)	2.1119(0.0826)
PGA	62.1425(2.2133)	0.6715(0.1176)	5.1566(0.7436)	6.7522(0.2750)	2.2119(0.0626)
RGA	58.0272(2.3606)	0.5576(0.1204)	4.2174(0.3211)	6.5948(0.3903)	2.1190(0.0466)
OGA	59.1272(3.1780)	0.6742(0.1162)	5.1483(0.6048)	6.5454(0.3231)	2.1391(0.0414)

study of prostate cancer by Blake et al. [23]. The data set consists of the medical records of 97 patients who were about to receive a radical prostatectomy. The predictors are eight clinical measures, i.e., cancer volume, prostate weight, age, etc., and one response variable, i.e., the logarithm of prostate-specific antigen. The third one is the Boston Housing data set created from a housing values survey in suburbs of Boston by Harrison [24]. This data set contains 506 instances which include thirteen attributions, i.e., per capita crime rate by town, proportion of non-retail business acres per town, average number of rooms per dwelling, etc., and one response variable, i.e., median value of owner-occupied homes. The fourth one is the concrete compressive strength (CCS) data set created from [25]. The data set contains 1030 instances including eight quantitative independent variables, i.e., age and ingredients, etc., and one dependent variable, i.e., quantitative concrete compressive strength. The fifth one is the Abalone data set, which comes from an original study in [26] for predicting the age of abalone from physical measurements. The data set contains 4177 instances which were measured on eight independent variables, i.e., length, sex, height, etc., and one response variable, i.e., the number of rings.

Similarly, we randomly divide all the real data sets into two disjoint equal parts. The first half serves as the training set and the second half serves as the test set. We also utilize the Z-score standardization method [27] to normalize the data sets, in order to avoid the error caused by considerable magnitude difference among data dimensions. For each real data experiment, dictionary (or the set of atoms) is firstly changed to the decision stumps (specifying one split of each tree, $J = 1$) corresponding to an additive model with only main effects. And then, the dictionary is changed to the Vanilla neural networks to further show that the proposed approach also can boost the performance of neural networks. The neural networks are set with one sigmoid hidden layer, where the input units equal to the dimension of samples, the hidden units are set to 5 and the output units (with affine transformation) are the same as the dimension of the labels. Back-propagation (BP) algorithm is employed to train each neural network.

Table 3 documents the empirical performance comparison results of TGA, PGA, RGA and OGA on five real data sets, respectively. It can be easily observed that, except OGA, which is sensitive to noise, TGA and RGA outperform the PGA to a large extent. Furthermore, TGA at least performs as the second best algorithm among all the greedy learning schemes. Thus, the results of real data coincide with the toy simulations and therefore, experimentally verify our theoretical assertions.

5 Proof of Theorem 1

In this section, we provide the proof of Theorem 1. The proof is divided into five parts, which contains the error decomposition strategy, approximation error estimate, sample error estimate, hypothesis error estimate and learning rate analysis. The methodology of our proof is the same as that of [5] and the main tool is borrowed from [15].

Error decomposition. In order to give an error decomposition strategy for $\mathcal{E}(f^t_{\boldsymbol{z},k})-\mathcal{E}(f_\rho)$, we need to construct a function $f^*_k \in \mathrm{span}(D_n)$ as follows. Since $f_\rho \in \mathcal{L}^r_{\mathcal{D}_n}$, there exists a $h_\rho := \sum_{i=1}^n a_i g_i \in \mathrm{Span}(\mathcal{D}_n)$

such that

$$\|h_\rho\|_{\mathcal{L}_1(\mathcal{D})} \leqslant \mathcal{B} \text{ and } \|f_\rho - h_\rho\| \leqslant \mathcal{B}n^{-r}. \tag{5}$$

Define $f_0^* = 0$, and

$$f_k^* = \left(1 - \frac{1}{k}\right) f_{k-1}^* + \text{sign}\left(\left\langle h_\rho - \left(1 - \frac{1}{k}\right) f_{k-1}^*, g_k^* \right\rangle_\rho\right) \min\left\{\left|\left\langle h_\rho - \left(1 - \frac{1}{k}\right) f_{k-1}^*, g_k^* \right\rangle_\rho\right|, \frac{\mathcal{B}}{k}\right\} g_k^*, \tag{6}$$

where

$$g_k^* := \arg\max_{g \in \mathcal{D}_n} \left\langle h_\rho - \left(1 - \frac{1}{k}\right) f_{k-1}^*, g \right\rangle_\rho.$$

Let $f_{\boldsymbol{z},k}^t$ and f_k^* be defined as in Algorithm 2 and (6), respectively, then we have

$$\mathcal{E}(\pi_M f_{\boldsymbol{z},k}^t) - \mathcal{E}(f_\rho) \leqslant \mathcal{E}(f_k^*) - \mathcal{E}(f_\rho) + \mathcal{E}_{\boldsymbol{z}}(\pi_M f_{\boldsymbol{z},k}^t) - \mathcal{E}_{\mathbf{z}}(f_k^*) + \mathcal{E}_{\boldsymbol{z}}(f_k^*) - \mathcal{E}(f_k^*) + \mathcal{E}(\pi_M f_{\boldsymbol{z},k}^t) - \mathcal{E}_{\boldsymbol{z}}(\pi_M f_{\boldsymbol{z},k}^t).$$

Upon making the short hand notations

$$\mathcal{D}(k) := \mathcal{E}(f_k^*) - \mathcal{E}(f_\rho),$$

$$\mathcal{S}(\boldsymbol{z}, k, \delta) := \mathcal{E}_{\boldsymbol{z}}(f_k^*) - \mathcal{E}(f_k^*) + \mathcal{E}(\pi_M f_{\boldsymbol{z},k}^t) - \mathcal{E}_{\boldsymbol{z}}(\pi_M f_{\boldsymbol{z},k}^t),$$

and

$$\mathcal{P}(\boldsymbol{z}, k, \delta) := \mathcal{E}_{\boldsymbol{z}}(\pi_M f_{\boldsymbol{z},k}^t) - \mathcal{E}_{\boldsymbol{z}}(f_k^*)$$

respectively for the approximation error, the sample error and the hypothesis error, we have

$$\mathcal{E}(\pi_M f_{\boldsymbol{z},k}^t) - \mathcal{E}(f_\rho) = \mathcal{D}(k) + \mathcal{S}(\boldsymbol{z}, k, \delta) + \mathcal{P}(\boldsymbol{z}, k, \delta). \tag{7}$$

At first, we give an upper bound estimate for $\mathcal{D}(k)$, which can be easily deduced from [5, Proposition 1] and [13, Theorem 1.2].

Lemma 1. Let f_k^* be defined in (6). If $f_\rho \in \mathcal{L}_{\mathcal{D}_n}^r$, then

$$\mathcal{D}(k) \leqslant \mathcal{B}^2(k^{-1/2} + n^{-r})^2. \tag{8}$$

Now we turn to bound the sample error. Upon using the short hand notations

$$S_1(\boldsymbol{z}, k) := \{\mathcal{E}_{\boldsymbol{z}}(f_k^*) - \mathcal{E}_{\boldsymbol{z}}(f_\rho)\} - \{\mathcal{E}(f_k^*) - \mathcal{E}(f_\rho)\}$$

and

$$S_2(\boldsymbol{z}, k) := \{\mathcal{E}(\pi_M f_{\boldsymbol{z},k}^t) - \mathcal{E}(f_\rho)\} - \{\mathcal{E}_{\boldsymbol{z}}(\pi_M f_{\boldsymbol{z},k}^t) - \mathcal{E}_{\boldsymbol{z}}(f_\rho)\},$$

we write

$$\mathcal{S}(\boldsymbol{z}, k) = \mathcal{S}_1(\boldsymbol{z}, k) + \mathcal{S}_2(\boldsymbol{z}, k). \tag{9}$$

To bound $S_1(\boldsymbol{z}, k)$, we only need to use the [5, Proposition 2] by noticing $\|f_k^*\|_{\mathcal{L}_1(\mathcal{D})} \leqslant \mathcal{B}$.

Lemma 2. For any $0 < \delta < 1$, with confidence $1 - \frac{\delta}{2}$,

$$S_1(\boldsymbol{z}, k) \leqslant \frac{7(3M + \mathcal{B}\log\frac{4}{\delta})}{3m} + \frac{1}{2}\mathcal{D}(k).$$

In order to bound $\mathcal{S}_2(\boldsymbol{z}, k)$, we need the concept of an empirical covering number.

Definition 1. Let $(\mathcal{M}, d)$ be a pseudo-metric space and $T \subset \mathcal{M}$ a subset. For every $\varepsilon > 0$, the covering number $\mathcal{N}(T, \varepsilon, d)$ of T with respect to ε and d is defined as the minimal number of balls of radius ε whose union covers T, that is,

$$\mathcal{N}(T, \varepsilon, d) := \min\left\{l \in \mathbb{N} : T \subset \bigcup_{j=1}^{l} B(t_j, \varepsilon)\right\}$$

for some $\{t_j\}_{j=1}^l \subset \mathcal{M}$, where $B(t_j, \varepsilon) = \{t \in \mathcal{M} : d(t, t_j) \leqslant \varepsilon\}$.

The l^2-empirical covering number of a function set is defined by means of the normalized l^2-metric d_2 on the Euclidean space $\mathbb{R}^m$ given in [28] with $d_2(\boldsymbol{a},\boldsymbol{b}) = \left(\frac{1}{m}\sum_{i=1}^m |a_i - b_i|^2\right)^{\frac{1}{2}}$ for $\boldsymbol{a} = (a_i)_{i=1}^m, \boldsymbol{b} = (b_i)_{i=1}^m \in \mathbb{R}^m$.

Definition 2. Let $\mathcal{G}$ be a set of functions on X, $\boldsymbol{x} = (x_i)_{i=1}^m \subset X^m$, and let

$$\mathcal{G}|_{\boldsymbol{x}} := \{(f(x_i))_{i=1}^m : f \in \mathcal{G}\} \subset R^m.$$

Set $\mathcal{N}_{2,\boldsymbol{x}}(\mathcal{G},\varepsilon) = \mathcal{N}(\mathcal{G}|_{\boldsymbol{x}},\varepsilon,d_2)$. The l^2-empirical covering number of $\mathcal{G}$ is defined by

$$\mathcal{N}_2(\mathcal{G},\varepsilon) := \sup_{m\in\mathbb{N}}\sup_{\boldsymbol{x}\in X^m}\mathcal{N}_{2,\boldsymbol{x}}(\mathcal{G},\varepsilon), \quad \varepsilon > 0.$$

The following two lemmas can be found, respectively, in [28, Theorem 2] and [29].

Lemma 3. If ϕ satisfies (1), then for arbitrary $\varepsilon > 0$,

$$\log \mathcal{N}_2(B_1,\varepsilon) \leqslant C_1 \varepsilon^{-\frac{2d}{d+2s}},$$

where B_R is the ball in $\mathcal{L}_1(\mathcal{D})$ with radius R, and C_1 is a constant depending only on s, C_s and X.

Lemma 4. Let $\mathcal{F}$ be a class of measurable functions on Z. Assume that there are constants $B, c > 0$ and $\alpha \in [0,1]$ such that $\|f\|_\infty \leqslant B$ and $\mathrm{E}f^2 \leqslant c(\mathrm{E}f)^\alpha$ for every $f \in \mathcal{F}$, where E represents expectation. If for some $a > 0$ and $p \in (0,2)$,

$$\log \mathcal{N}_2(\mathcal{F},\varepsilon) \leqslant a\varepsilon^{-p}, \quad \forall \varepsilon > 0, \tag{10}$$

then there exists a constant c_p' depending only on p such that for any $t > 0$, with probability at least $1 - \mathrm{e}^{-t}$, there holds

$$\mathrm{E}f - \frac{1}{m}\sum_{i=1}^m f(z_i) \leqslant \frac{1}{2}\eta^{1-\alpha}(\mathrm{E}f)^\alpha + c_{p'}\eta + 2\left(\frac{ct}{m}\right)^{\frac{1}{2-\alpha}} + \frac{18Bt}{m}, \quad \forall f \in \mathcal{F}, \tag{11}$$

where

$$\eta := \max\left\{c^{\frac{2-p}{4-2\alpha+p\alpha}}\left(\frac{a}{m}\right)^{\frac{2}{4-2\alpha+p\alpha}}, B^{\frac{2-p}{2+p}}\left(\frac{a}{m}\right)^{\frac{2}{2+p}}\right\}.$$

We are now in a position to establish an upper bound estimate for $\mathcal{S}_2(\boldsymbol{z},k)$.

Lemma 5. Let $f_{\boldsymbol{z},k}^t$ be defined as in Algorithm 2 and $0 < \delta < 1$, then with confidence $1 - \frac{\delta}{2}$, there holds

$$\mathcal{S}_2(\boldsymbol{z},k) \leqslant \frac{1}{2}\{\mathcal{E}(\pi_M f_{\boldsymbol{z},k}^t) - \mathcal{E}(f_\rho)\} + C_3 \log\frac{4}{\delta} s_k^{\frac{d}{d+s}} m^{-\frac{2s+d}{2d+2s}},$$

where C_3 is a constant depending only on d, X, ϕ and M.

Proof. We apply Lemma 4 to the set of functions $\mathcal{F}_R$, where

$$\mathcal{F}_R := \left\{(y - \pi_M f(x))^2 - (y - f_\rho(x))^2 : f \in B_R\right\}. \tag{12}$$

Each function $g \in \mathcal{F}_R$ has the form

$$g(z) = (y - \pi_M f(x))^2 - (y - f_\rho(x))^2, \quad f \in B_R,$$

and is automatically a function on Z. Hence

$$\mathrm{E}g = \mathcal{E}(f) - \mathcal{E}(f_\rho) = \|\pi_M f - f_\rho\|_\rho^2$$

and

$$\frac{1}{m}\sum_{i=1}^m g(z_i) = \mathcal{E}_{\boldsymbol{z}}(\pi_M f) - \mathcal{E}_{\boldsymbol{z}}(f_\rho).$$

Observe that

$$g(z) = (\pi_M f(x) - f_\rho(x))((\pi_M f(x) - y) + (f_\rho(x) - y)).$$

Using the obvious inequalities $\|f\|_\infty \leqslant M$ a.e. and $|f_\rho| \leqslant M$ a.e., we get the inequalities

$$|g(z)| \leqslant (M+M)(M+3M) \leqslant 8M^2$$

and

$$\mathrm{E}g^2 = \int_Z (2y - \pi_M f(x) - f_\rho(x))^2 (\pi_M f(x) - f_\rho(x))^2 \mathrm{d}\rho \leqslant (4M)^2 \mathrm{E}g.$$

For $g_1, g_2 \in \mathcal{F}_R$, we have

$$|g_1(z) - g_2(z)| = |(y - \pi_M f_1(x))^2 - (y - \pi_M f_2(x))^2| \leqslant 4M|f_1(x) - f_2(x)|.$$

It follows that

$$\mathcal{N}_{2,\boldsymbol{z}}(\mathcal{F}_R, \varepsilon) \leqslant \mathcal{N}_{2,\boldsymbol{x}}\left(B_R, \frac{\varepsilon}{4M}\right) \leqslant \mathcal{N}_{2,\boldsymbol{x}}\left(B_1, \frac{\varepsilon}{4MR}\right).$$

Using the above inequality and Lemma 3, we have

$$\log \mathcal{N}_{2,\boldsymbol{z}}(\mathcal{F}_R, \varepsilon) \leqslant C_1 (4MR)^{\frac{2d}{d+2s}} \varepsilon^{-\frac{2d}{d+2s}}.$$

By Lemma 4 with $B = c = 16M^2$, $\alpha = 1$ and $a = C_1(4MR)^{\frac{2d}{d+2s}}$, we know that for any $\delta \in (0,1)$, with confidence $1 - \frac{\delta}{2}$, there exists a constant C depending only on d, X, and ϕ such that for all $g \in \mathcal{F}_R$,

$$\mathrm{E}g - \frac{1}{m}\sum_{i=1}^m g(z_i) \leqslant \frac{1}{2}\mathrm{E}g + C\eta + C(M+1)^2 \frac{\log(4/\delta)}{m}.$$

Here

$$\eta = \{16M^2\}^{\frac{s}{s+d}} \left(\frac{(4MR)^{\frac{2d}{d+2s}}}{m}\right)^{\frac{d+2s}{2d+2s}}.$$

Therefore, there exists a constant C_2 depending only on d, X, ϕ and M such that

$$\eta \leqslant C_2 R^{\frac{d}{d+s}} m^{-\frac{d+2s}{2d+2s}},$$

which implies

$$\mathrm{E}g - \frac{1}{m}\sum_{i=1}^m g(z_i) \leqslant \frac{1}{2}\mathrm{E}g + CC_2 R^{\frac{d}{d+s}} \log\frac{4}{\delta} m^{-\frac{d+2s}{2d+2s}}.$$

Due to the definition of $f^t_{\boldsymbol{z},k}$, it is easy to see that $\|f^t_{\boldsymbol{z},k}\|_{\mathcal{L}_1} \leqslant s_k := \sum_{i=0}^k h_k$. It follows that there exists a constant C_3 depending only on d, X, ϕ and M such that

$$\mathcal{S}_2(\boldsymbol{z}, k) \leqslant \frac{1}{2}\{\mathcal{E}(\pi_M f^t_{\boldsymbol{z},k}) - \mathcal{E}(f_\rho)\} + C_3 \log\frac{4}{\delta} s_k^{\frac{d}{d+s}} m^{-\frac{2s+d}{2d+2s}}.$$

Then, we turn to bound the hypothesis error $\mathcal{P}(\boldsymbol{z}, k)$.

Lemma 6. If $f^t_{\boldsymbol{z},k}$, f^*_k and Λ_k are defined in Algorithm 2 and (6), then we have

$$\mathcal{P}(\boldsymbol{z}, k) \leqslant \frac{M^2\mathcal{B}}{\mathcal{B} + s_k} + \sum_{l=0}^k h_l^2.$$

Proof. Write $f^*_k = \sum_{j=1}^n w_j g_j$ and $f^t_{\boldsymbol{z},k} = \sum_{j=1}^n a_j g_j$. Then, we have from $\|f^*_k\|_{\mathcal{L}_1(\mathcal{D})} \leqslant \mathcal{B}$ that

$$\Delta W_k := \sum_{j=1}^n |w_j - a_j| \leqslant \sum_{j=1}^n |w_j| + \sum_{j=1}^n |a_j| \leqslant \sum_{j=1}^n |w_j| + s_k \leqslant \mathcal{B} + s_k,$$

where $s_k := \sum_{j=1}^n h_j$. As $\Lambda_k = [-h_k, h_k]$ is symmetric, then it follows from the definition of $f^t_{\boldsymbol{z},k}$ that

$$\mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k+1}) \leqslant \mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k} + h_k b_j g_j)$$

with $b_j = \mathrm{sign}(w_j - a_j)$. Thus,

$$\begin{aligned}\Delta W_k \mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k+1}) &\leqslant \sum_{j=1}^{n} |w_j - a_j| \frac{1}{m} \sum_{i=1}^{m} \left(f^t_{\boldsymbol{z},k}(x_i) + h_k b_j g_j(x_i) - y_i\right)^2 \\ &= \sum_{j=1}^{n} |w_j - a_j| \frac{1}{m} \sum_{i=1}^{m} \left((f^t_{\boldsymbol{z},k}(x_i) - y_i)^2 + 2(f^t_{\boldsymbol{z},k}(x_i) - y_i) h_k b_j g_j(x_i) + h_k^2 g_j^2(x_i)\right) \\ &\leqslant \Delta W_k \mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k}) + 2h_k \frac{1}{m} \sum_{i=1}^{m} (f^t_{\boldsymbol{z},k}(x_i) - y_i)(f^*_k(x_i) - f^t_{\boldsymbol{z},k}(x_i)) + \Delta W_k h_k^2 \|g_j\|^2_\infty \\ &\leqslant \Delta W_k \mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k}) + h_k(\mathcal{E}_{\boldsymbol{z}}(f^*_k) - \mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k})) + C\Delta W_k h_k^2,\end{aligned}$$

where we use the fact that

$$\begin{aligned}\frac{1}{m} \sum_{i=1}^{m} (f^t_{\boldsymbol{z},k}(x_i) - y_i)(f^*_k(x_i) - f^t_{\boldsymbol{z},k}(x_i)) &= \frac{1}{m} \sum_{i=1}^{m} (f^t_{\boldsymbol{z},k}(x_i) - y_i)(f^*_k(x_i) - y_i + y_i - f^t_{\boldsymbol{z},k}(x_i)) \\ &= \frac{1}{m} \sum_{i=1}^{m} (f^t_{\boldsymbol{z},k}(x_i) - y_i)(f^*_k(x_i) - y_i) - \mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k}),\end{aligned}$$

and

$$2\frac{1}{m} \sum_{i=1}^{m} (f^t_{\boldsymbol{z},k}(x_i) - y_i)(f^*_k(x_i) - y_i) \leqslant \frac{1}{m} \sum_{i=1}^{m} (f^t_{\boldsymbol{z},k}(x_i) - y_i)^2 + \frac{1}{m} \sum_{i=1}^{m} (f^*_k(x_i) - y_i)^2.$$

Hence, we have

$$\mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k+1}) \leqslant \mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k}) + \frac{h_k}{\Delta W_k}(\mathcal{E}_{\boldsymbol{z}}(f^*_k) - \mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k})) + Ch_k^2.$$

That is,

$$\mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k+1}) - \mathcal{E}_{\boldsymbol{z}}(f^*_k) \leqslant \left(1 - \frac{h_k}{\mathcal{B} + s_k}\right)\left(\mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k}) - \mathcal{E}_{\boldsymbol{z}}(f^*_k)\right) + Ch_k^2.$$

Note that for arbitrary $a \geqslant 0$,

$$\begin{aligned}\prod_{l=0}^{j} \left(1 - \frac{h_l}{s_l + a}\right) &= \exp\left(\sum_{l=0}^{k} \ln\left(1 - \frac{s_{l+1} - s_l}{s_l + a}\right)\right) \\ &\leqslant \exp\left(\sum_{l=0}^{k} -\frac{s_{l+1} - s_l}{s_l + a}\right) \leqslant \exp\left(-\int_{s_0}^{s_{k+1}} \frac{1}{v + a} \mathrm{d}v\right) \\ &= \frac{s_0 + a}{s_{k+1} + a}.\end{aligned}$$

Then we have

$$\mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k+1}) - \mathcal{E}_{\boldsymbol{z}}(f^*_k) \leqslant \prod_{l=0}^{k} \left(1 - \frac{h_l}{\mathcal{B} + s_l}\right) \mathcal{E}_{\boldsymbol{z}}(f_0) + C \sum_{l=0}^{k} h_l^2 \leqslant \frac{M^2 \mathcal{B}}{\mathcal{B} + s_k} + C \sum_{l=0}^{k} h_l^2.$$

Then the inequality $\mathcal{E}_{\boldsymbol{z}}(\pi_M f^t_{\boldsymbol{z},k}) \leqslant \mathcal{E}_{\boldsymbol{z}}(f^t_{\boldsymbol{z},k})$ yields the final estimate of the hypothesis error.

With the help of the above lemmas, we are in a position to prove Theorem 1.

Proof. (Proof of Theorem 1) Let $h_k \sim k^{-2/3}$. Then it is easy to deduce that $s_k \sim k^{1/3}$. We assemble the results in Lemma 1 through (6) and (7) to write

$$\begin{aligned}\mathcal{E}(\pi_M f^t_{\boldsymbol{z},k}) - \mathcal{E}(f_\rho) &\leqslant \mathcal{D}(k) + \mathcal{S}_1(\boldsymbol{z},k) + \mathcal{S}_2(\boldsymbol{z},k) + \mathcal{P}(\boldsymbol{z},k) \\ &\leqslant \mathcal{B}^2(k^{-1/2} + n^{-r})^2 + \frac{7(3M + \mathcal{B}\log\frac{4}{\delta})}{3m} \\ &\quad + \frac{1}{2}\{\mathcal{E}(\pi_M f^t_{\boldsymbol{z},k}) - \mathcal{E}(f_\rho)\} + C_3 \log\frac{4}{\delta} k^{\frac{d}{3d+3s}} m^{-\frac{d+2s}{2d+2s}} + C_4 \mathcal{B} k^{-1/3}\end{aligned}$$

holding with confidence at least $1-\delta$. Therefore

$$\mathcal{E}(\pi_M f^t_{\boldsymbol{z},k}) - \mathcal{E}(f_\rho) \leqslant C\mathcal{B}^2\left(\log\frac{4}{\delta}k^{\frac{d}{3d+3s}}m^{-\frac{2s+d}{2d+2s}} + k^{-\frac{1}{3}} + n^{-2r}\right)$$

holds with confidence at least $1-\delta$, where C is a constant depending only on ϕ, d, X and M. This completes the proof of Theorem 1.

6 Conclusion

In this paper, we applied another variant of greedy-type algorithms, called the truncated greedy algorithm (TGA), to improve the learning performance of the pure greedy algorithm (PGA). The contributions can be concluded as follows. Firstly, we applied the idea of truncation based on the bias and variance balance principle to modify PGA. To be detailed, different from other variants of PGA, such as the OGA and RGA, that devote to reduce the bias, TGA focuses on cutting down the variance. We rigorously prove that TGA is better than PGA in the sense that TGA possesses the faster learning rate than PGA. Secondly, we derived a relatively tight generalization error bound for the TGA learning. For some specified learning tasks, we theoretically proved that TGA can reach a learning rate as $\mathcal{O}(m^{-1/2})$, which was better than all the existing results concerning greedy learning. This theoretical guidance implies that under a certain special circumstance, TGA may perform better than all the existing greedy-type algorithms such as OGA, RGA and PGA. Finally, we verified the above assertion by both toy simulations and real data experiments.

The idea of truncation in greedy learning paves a new road to modify greedy-type algorithms. We guess that this idea can be successfully used in both OGA and RGA to improve their performances further. We will keep on with this study and report our new findings in future.

Acknowledgements This work was supported by National Basic Research Program of China (Grant No. 2013CB329404) and National Natural Science Foundation of China (Grant Nos. 61502342, 11401462).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Barron A R, Cohen A, Dahmen W, et al. Approximation and learning by greedy algorithms. Ann Stat, 2008, 36: 64–94
2 Chen H, Li L, Pan Z. Learning rates of multi-kernel regression by orthogonal greedy algorithm. J Statist Plan & Infer, 2013, 143: 276–282
3 Fang J, Lin S B, Xu Z B. Learning and approximation capabilities of orthogonal super greedy algorithm. Know Based Syst, 2016, 95: 86–98
4 Friedman J. Greedy function approximation: a gradient boosting machine. Ann Stat, 2001, 29: 1189–1232
5 Lin S B, Rong Y H, Sun X P, et al. Learning capability of relaxed greedy algorithms. IEEE Trans Neural Netw Learn Syst, 2013, 24: 1598–1608
6 Mannor S, Meir R, Zhang T. Greedy algorithms for classification-consistency, convergence rates, and adaptivity. J Mach Learn Res, 2003, 4: 713–742
7 Xu L, Lin S B, Zeng J S, et al. Greedy metrics in orthogonal greedy learning. ArXiv:1411.3553, 2014
8 Schmidt E. Zur Theorie der linearen und nichtlinearen integralgleichungen I. Math Annalen, 1906, 63: 433–476
9 Temlyakov V. Greedy approximation. Acta Numer, 2008, 17: 235–409
10 DeVore R A, Temlyakov V. Some remarks on greedy algorithms. Adv Comput Math, 1996, 5: 173–187
11 Livshitz E, Temlyakov V. Two lower estimates in greedy approximation. Constr Approx, 2003, 19: 509–523
12 Livshits E. Lower bounds for the rate of convergence of greedy algorithms. Izvestiya: Mathematics, 2009, 73: 1197–1215
13 Temlyakov V. Relaxation in greedy approximation. Constr Approx, 2008, 28: 1–25
14 Cucker F, Zhou D X. Learning Theory: an Approximation Theory Viewpoint. Cambridge: Cambridge University Press, 2007
15 Zhang T, Yu B. Boosting with early stopping: convergence and consistency. Ann Statis, 2005, 33: 1538–1579
16 Bagirov A, Clausen C, Kohler M. An L_2 boosting algorithm for estimation of a regression function. IEEE Trans Inf Theory, 2010, 56: 1417–1429
17 Tibshirani R. Regression shrinkage and selection via the lasso. J R Stat Soc, 1996, 1: 267–288

18 Golub G H, Heath M T, Wahba G. Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics, 1979, 21: 215–223
19 Efron B, Hastie T, Johnstone I, et al. Least angle regression. Ann Stat, 2004, 32: 407–499
20 Wendland H. Scattered Data Approximation. Cambridge: Cambridge University Press, 2005
21 Zhou D X, Jetter K. Approximation with polynomial kernels and SVM classifiers. Adv Comput Math, 2006, 25: 323–344
22 Breiman L, Friedman J, Stone C, et al. Classification and Regression Trees. Boca Raton: CRC Press, 1984
23 Blake C L, Merz C J. UCI Repository of machine learning databases. Irvine: University of California. http://www.ics.uci.edu/~mlearn/MLRepository.html. 1998, 55
24 Harrison D, Rubinfeld D L. Hedonic prices and the demand for clean air. J Environ Econ, 1978, 5: 81–102
25 Ye I C. Modeling of strength of high performance concrete using artificial neural networks. Cement Concrete Res, 1998, 28: 1797–1808
26 Nash W J, Sellers T L, Talbot S R, et al. The Population Biology of Abalone (Haliotis Species) in Tasmania: Blacklip Abalone (H. Rubra) From the North Coast and Islands of Bass Strait. Technical Report. 1994
27 Kreyszig E. Applied Mathematics. Hoboken: Wiley Press, 1979
28 Shi L, Feng Y, Zhou D X. Concentration estimates for learning with l^1-regularizer and data dependent hypothesis spaces. Appl Comput Harmon Anal, 2011, 31: 286–302
29 Wu Q, Ying Y, Zhou D X. Multi-kernel regularized classifiers. J Complex, 2007, 23: 108–134