Functional Gradient Boosting for Learning Residual-like Networks with Statistical Guarantees

Atsushi Nitanda\textsuperscript{1,2,3}  
nitanda@mist.i.u-tokyo.ac.jp  
Taiji Suzuki\textsuperscript{1,2}  
taiji@mist.i.u-tokyo.ac.jp  
\textsuperscript{1}Graduate School of Information Science and Technology, The University of Tokyo  
\textsuperscript{2}Center for Advanced Intelligence Project, RIKEN  
\textsuperscript{3}PRESTO, Japan Science and Technology Agency

Abstract

Recently, several studies have proposed progressive or sequential layer-wise training methods based on the boosting theory for deep neural networks. However, most studies lack the global convergence guarantees or require weak learning conditions that can be verified \textit{a posteriori} after running methods. Moreover, generalization bounds usually have a worse dependence on the network depth. In this paper, to resolve these problems, we propose a new functional gradient boosting for learning deep residual-like networks in a layer-wise fashion with its statistical guarantees on multi-class classification tasks. In the proposed method, each residual block is recognized as a functional gradient (i.e., weak learner), and the functional gradient step is performed by stacking it on the network, resulting in a strong optimization ability. In the theoretical analysis, we show the global convergence of the method under a standard margin assumption on a data distribution instead of a weak learning condition, and we eliminate a worse dependence on the network depth in a generalization bound via a fine-grained convergence analysis. Moreover, we show that the existence of a learnable function with a large margin on a training dataset significantly improves a generalization bound. Finally, we experimentally demonstrate that our proposed method is certainly useful for learning deep residual networks.

1 Introduction

Residual networks (ResNets) \cite{He2016} are state-of-the-art models equipped with a notable structure called \textit{skip connections} or \textit{resblocks}. Thus, several studies have been devoted to understanding the reason for their success. Recently, interesting aspects of ResNets have been reported; one is the ensemble view \cite{Veit2016, Littwin2016}, where resblocks are assembled as weak learners to make a stronger model, and the other is the optimization or ordinary differential equations view \cite{Jastrzebski2017}, where resblocks provide a discretization of gradient flows or ordinary differential equations to minimize an objective function. Motivated by these perspectives and gradient boosting machines \cite{Mason1999, Friedman2001, Chen2016, Ke2017}, functional gradient boosting (ResFGB) \cite{Nitanda2018} for learning deep ResNets has been proposed. In their method, each resblock can be recognized as an approximation to a functional gradient, and functional gradient descent is performed by adding a resblock on the top of the feature extraction layer to optimize a network in a function space. Simultaneously, another boosting method \cite{Huang2018} for ResNets has been proposed for the same purpose.

Usually, a generalization bound is composed of an approximation term and a variance term. In \cite{Huang2018, Nitanda2018}, the approximation term for a network obtained in their method was evaluated through an algorithmic convergence rate analysis of the method, and the variance term was estimated by utilizing a conventional analysis of Rademacher complexity \cite{Koltchinskii2002} for neural networks. In their analysis, a type of weak learning condition which is verified only after running algorithms was assumed to guarantee global convergence as well as many other boosting methods. In the generalization bounds, linear and exponential dependence on the network depth appeared in \cite{Huang2018}.
We first propose a new functional gradient boosting method, called ResFGB-FW, which is closely related to our method in terms of optimization ability and a fine-grained convergence analysis. A key step in the proof for global convergence is to show that an obtained weak learner in the method exhibits comparable performance with a pure functional gradient in terms of optimization ability in a function space. We show this property by making a margin assumption where a given training dataset is separable with a sufficient margin by a good classifier in a set of learnable functions assembled by a shallow neural network, while this type of property is usually guaranteed by a weak learning condition in other methods. As a result, we can show not only that a worse dependence on the network depth can be eliminated but also that the existence of a good classifier providing a large margin can significantly improve a generalization bound. Finally, we experimentally compare our proposed method with state-of-the-art methods and demonstrate certain superior performances.

Contributions

- We propose a new functional gradient boosting for learning deep ResNets by combining ResFGB with the Frank-Wolfe method and show superior empirical performance over existing methods on multiclass classification tasks.
- We provide a generalization bound for a trained ResNet via a global convergence analysis of the method under a margin condition instead of a weak learning condition. Theoretical contributions in this study are that (i) we guarantee the global convergence without a type of weak learning condition, (ii) we show that our network architecture is helpful in ensuring the convergence, (iii) we eliminate a worse dependence on the network depth in a generalization bound via a convergence analysis.

Related Work

Recently, it is known that a progressive learning scheme or sequential layer-wise training has a powerful generalization ability as well as a conventional learning scheme on the image classification task (Belilovsky et al., 2019; Nøkland and Eidnes, 2019) and it is useful in network architecture search (Liu et al., 2018; Kim et al., 2018; Well et al., 2019). We provide a generalization bound without a type of weak learning condition. Theoretical contributions in this work are that (i) we guarantee the global convergence without such a weak learning condition, (ii) we show that our network architecture is helpful in ensuring the convergence, (iii) we eliminate a worse dependence on the network depth in a generalization bound via a convergence analysis.

2 Preliminary

In this section, we provide several notations to describe a problem setting of the classification and a notion of functional gradients used in the proposed method and theoretical analyses. Moreover, we provide a basic skeleton of generalization bound to derive an explicit bound later.

2.1 Problem setting

Let $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y}$ be a feature space and a finite label set of cardinality $c$. We denote by $\nu$ a true Borel probability measure on $\mathcal{X} \times \mathcal{Y}$ and
by \( \nu_n \) an empirical probability measure by observations \( (x_i, y_i)_{i=1}^{n} \) independently drawn from \( \nu_0 \), i.e.,
\[
d\nu_0(x, y) = \sum_{i=1}^{n} \delta(x_i, y_i)(x, y)dx dy/n,\]
where \( \delta \) denotes the Dirac delta function. For \( \nu_n \), we denote by \( \nu_n^X \) the marginal distribution on \( X \). In general, for a probability measure \( \mu \), we denote by \( \mathbb{E}_\mu \) the expectation with respect to a random variable according to \( \mu \). We denote by \( L_2(\nu_n^X) \) the real-valued function space on \( X \) equipped with inner product \( \langle \cdot, \cdot \rangle_{L_2(\nu_n^X)} \) and by \( L_2^2(\nu_n^X) \) the product space of \( L_2(\nu_n^X) \): for \( \forall \xi, \eta \in L_2^2(\nu_n^X) \),
\[
\langle \xi, \eta \rangle_{L_2^2(\nu_n^X)} \overset{def}{=} \mathbb{E}_{\nu_n^X} \left[ \sum_{j=1}^{m} \xi_j(X) \eta_j(X) \right].
\]

\( L_p \)-norm is defined by:
\[
\|\xi\|_{L_p(\nu_n^X)} \overset{def}{=} \mathbb{E}_{\nu_n^X} \|\xi(X)\|_p^p = \mathbb{E}_{\nu_n^X} \left[ \sum_{j=1}^{m} |\xi_j(X)|^p \right]^{\frac{1}{p}}
\]
for \( p \in \{1, 2\} \) and \( \xi \in L_2^2(\nu_n^X) \).

Let \( \mathcal{F} \) be a given function class of predictors. The problem considered in this paper is formalized as the empirical risk minimization problem:
\[
\min_{f \in \mathcal{F}} \mathcal{L}_n(f) \overset{def}{=} \mathbb{E}_{\nu_n} [l(f(X), Y)] ,
\]
where \( l \) be the logistic loss function for multiclass classification problems:
\[
l(\xi, y) = -\log \left( \frac{\exp(\xi_y)}{\sum_{y \in Y} \exp(\xi_y)} \right).
\]

Generally, regularization or constraint is needed for a function class to guarantee generalization. Thus, we next describe a network architecture belonging to \( \mathcal{F} \).

### 2.2 Network architecture

We split the predictor \( f \) into the feature extractor and linear predictor: \( f(x) = w^T \phi(x) \), where \( w \in \mathbb{R}^{d \times c} \) is a weight for the last layer and \( \phi \) is a feature extractor from \( X \) to \( \mathcal{X} \). For simplicity, we pre-train \( w \) on a distribution \( \nu_0 \) as a linear classifier and fix it before optimizing \( \phi \). In addition, we assume \( \|w\|_{\infty} \leq A_w \) for \( \forall y \in \mathcal{Y} \). Thus, we focus on the problem of learning a feature extractor \( \phi \) and denote \( \mathcal{R}_n(\phi) = \mathcal{L}_n(\tau \phi^\top) \).

We here describe the entire network architecture that is a type of residual networks with slight modifications (see Figure 1).

**Definition 1.** Let \( \mathcal{B}_1 \) and \( \mathcal{B}_2 \) be sets of neural networks: \( X \rightarrow \mathcal{X} \), including the zero function. We denote by \( \phi_t \) the output of \( \phi \) at the \( t \)-th layer. A set of residual blocks (resblocks) at the \( t \)-th layer is
\[
\mathbb{E}_{\phi_t} \overset{def}{=} \{ \iota_1 + \iota_2 \circ \phi | \iota_1 \in \mathcal{B}_1, \ i_2 \in \mathcal{B}_2 \}.
\]

A connection between \( t \)- and \( (t+1) \)-th layers of \( \phi \) is described as follows:
\[
\phi_{t+1}(x) \overset{def}{=} (1 - \eta_t)\phi_t(x) + \eta_t\iota_t(x)
\]

![Figure 1: High level diagram of \( t \)-th layer of the network obtained in the proposed method. A linear classifier and an auxiliary loss are used to train resblocks at \( t \)-layer and discarded after training \( t \)-th layer.](image)

**Remark** Our model has two major differences from the conventional residual networks and the model in [Nitanda and Suzuki (2018a)]: one is that the output \( \phi_t(x) \) is reduced at the rate of \( 1 - \eta_t \) stemming from the Frank-Wolfe method and the other is that resblocks receive input data directly like DenseNet (Huang et al., 2017). These modifications lead to provable statistical guarantees without weak learning assumptions.

To provide theoretical analyses, some restrictions on resblocks are required.

**Assumption 1.** Let \( \Lambda_i^C, \Lambda_i^D \ (i \in \{1, 2\}) \) be positive values and \( \sigma \) be a 1-Lipschitz activation function. Assume that residual blocks are shallow neural networks defined as follows:
\[
\mathcal{B}_i = \{ z \mapsto C_i \sigma(D_i z) | \max_j \| (C_i)_{j,*} \|_1 \leq \Lambda_i^C, \max_k \| (D_i)_{k,*} \|_1 \leq \Lambda_i^D, C_i \in \mathbb{R}^{d \times d'}, D_i \in \mathbb{R}^{d' \times d} \}.
\]

Assumption 1 is used for guaranteeing the generalization bound. Although, we adopt shallow networks as resblocks in this condition for simplicity, but we can extend them to deeper resblocks. A typical example of \( \sigma \) is the sigmoid function.
2.3 Functional gradient

The key notion used in this study is the functional gradient in a function space. We define
\[ \nabla_{\phi} R_n(\phi)(x) \overset{\text{def}}{=} \begin{cases} \partial_x l(\phi^T z, y_i) |_{x = \phi(x_i)} & (x = x_i), \\ 0 & \text{(otherwise)} \end{cases} \]

This is nothing but a functional gradient (Fréchet differential) in \( L_2^d(\nu_n^X) \), i.e., for \( \forall \phi, \forall \xi \in L_2^d(\nu_n^X) \),
\[ R_n(\phi + \xi) = R_n(\phi) + \langle \nabla_{\phi} R_n(\phi), \xi \rangle_{L_2^d(\nu_n^X)} + o(\|\xi\|_{L_2^d(\nu_n^X)}). \]

Thus, the functional gradient descent using \( \nabla_{\phi} R_n \) optimizes \( R_n \) in \( L_2^d(\nu_n^X) \), but we need a smoothing technique introduced later to guarantee generalization because \( \nabla R_n \) has no information on unseen data; in other words, this method is meaningless for expected minimization problems. We similarly define a functional gradient \( \nabla f \mathcal{L}_n(f) \) and see \( \nabla_{\phi} R_n(\phi) = \nabla f \mathcal{L}_n(f) \).

The following proposition shows the Fréchet differentiability and Lipschitz smoothness for the multiclass logistic loss.

**Proposition 1** (Nitanda and Suzuki 2018a). \( R_n(\phi) \) is convex and Fréchet differentiable in \( L_2^d(\nu_n^X) \). Moreover, there exists \( \exists L > 0 \) such that \( \forall \phi, \forall \xi \in L_2^d(\nu_n^X) \),
\[ R_n(\phi + \xi) \leq R_n(\phi) + \langle \nabla_{\phi} R_n(\phi), \xi \rangle_{L_2^d(\nu_n^X)} + \frac{L}{2} \|\xi\|_{L_2^d(\nu_n^X)}^2. \]

**Remark** Although, the Lipschitz parameter of \( R_n(\phi) \) depends on \( \pi, \) we consider it as a uniform constant because \( \pi \) is fixed in our theoretical analysis.

2.4 Generalization Bound

We provide a skeleton of generalization bound on an expected classification error which is derived by a conventional analysis of Rademacher complexity for neural networks (Koltchinski and Panchenko 2002). This bound is composed of the \( L_1 \)-functional gradient norm and the conventional margin defined below:
\[ \| \nabla f \mathcal{L}_n(f) \|_{L_1^2(\nu_X^Y)} \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} | \partial_{x_i} l(f(x_i), y_i) |, \]
\[ m_f(x, y) \overset{\text{def}}{=} f_y(x) - \max_{y' \neq y} f_{y'}(x). \]

**Theorem 1.** Suppose Assumption \( \square \) holds, \( \Lambda_2^C \Lambda_2^D \leq 1/2, \eta_t \leq 1 \), and \( |x|^2 \leq \Lambda_\infty \) on \( X \). Fix \( \delta > 0. \) Then, for \( \forall \rho > 0 \), with probability at least \( 1 - \rho \) over the random choice from \( \nu_n^X \), we have that for \( \forall f \in \mathcal{F} \) with the network depth \( T \in \mathbb{Z}_+ \),
\[ \mathbb{P}_\nu [ m_f(X, Y) \leq 0 ] \leq \frac{2e^{3\Lambda_\infty \Lambda_\infty}}{\delta \sqrt{n}} \left( 1 + \sum_{t=0}^{T-1} \eta_t \Lambda_2^C \Lambda_2^D \prod_{s=t+1}^{T-1} \left( 1 - \frac{\eta_s}{2} \right) \right) + \frac{1}{2n} \log \rho^{-1} + \frac{1}{2} \left( 1 + \exp(\delta) \right) \| \nabla f \mathcal{L}_n(f) \|_{L_1^2(\nu_X^Y)}. \]

A remaining problem for deriving an explicit generalization bound is to confirm how small functional gradient norms can be achievable by given rates of \( \eta_t \) and upper bounds: \( \Lambda_2^C, \Lambda_2^D, \Lambda_2^C, \Lambda_2^D \). We estimate these quantities by analyzing a convergence behavior of a proposed functional gradient boosting for learning deep residual networks.

3 Brief Review of Functional Gradient

Functional gradient methods including gradient boosting (Mason et al., 1999; Friedman, 2001) and gradient descent for kernel methods (Kivinen et al., 2004; Smale and Yao, 2006; Ying and Zhou, 2006; Raskutti et al., 2014) and over-parameterized neural networks (Jacot et al., 2018; Du et al., 2018; Arora et al., 2019) have powerful optimization ability because these methods naturally perform in function spaces.

In gradient boosting, \( \nabla_{\phi} R_n(\phi) \) is approximated in a given set of weak learners \( \mathcal{G} \) as follows:
\[ \xi_\phi \in \argmax_{\xi \in \mathcal{G}} (\nabla_{\phi} R_n(\phi), \xi)_{L_2^d(\nu_X^Y)} \]
and gradient method in a function space is performed using a descent direction \( -\xi_\phi \). This approximation procedure is recognized as a type of smoothing of functional gradients.

In particular, this smoothing procedure is realized by using the following kernel smoothing technique in the kernel method and over-parameterized neural networks:
\[ T_k \nabla_{\phi} R_n(\phi) \overset{\text{def}}{=} \mathbb{E}_{\nu_n^X} [ \nabla_{\phi} R_n(\phi)(X) k(X, \cdot) ] \]
\[ = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\phi} R_n(\phi)(x_i) k(x_i, \cdot), \]
where, \( k \) is a kernel function that is usually prefixed in kernel methods; on the contrary, it depends on a map \( \phi \) for over-parameterized neural networks and is called neural kernel (Jacot et al. 2018).

We note that a kernel smoothing \( \square \) is characterized as a special case of \( \square \) through the following relationship. Let \( (\mathcal{H}_k, \langle ., . \rangle_{\mathcal{H}_k}) \) be the reproducing Hilbert space associated with a kernel \( k \). Then,
\[ \frac{T_k \nabla_{\phi} R_n(\phi)}{\| T_k \nabla_{\phi} R_n(\phi) \|_{\mathcal{H}_k^d}} \in \argmax_{\xi \in G} (\nabla_{\phi} R_n(\phi), \xi)_{L_2^d(\nu_X^Y)}, \]
where \( \|\xi\|^2_{H^d_{\phi}} = \sum_{\alpha=1}^{d} \|\xi_{\alpha}\|^2_{H^d_{\phi}} \) for \( \xi = (\xi_{\alpha})_{\alpha=1}^{d} \in H^d_{\phi} \), which is confirmed by \( \langle \psi, \xi \rangle_{L^2_{\phi}(\nu_X)} = \langle T_k \psi, \xi \rangle_{H^d_{\phi}} \) for \( \forall \xi \in H^d_{\phi} \). Hence, these methods using kernel smoothing are considered as variants of functional gradient boosting.

To further improve the optimization ability compared to (4), one interesting idea is to maximize the following quantity, which corresponds to the right hand side of (4), with respect to \( k \):

\[
\langle \nabla \phi \mathcal{R}_n(\phi), T_k \nabla \phi \mathcal{R}_n(\phi) \rangle_{L^2_{\phi}(\nu_X)} = \| T_k \nabla \phi \mathcal{R}_n(\phi) \|^2_{H^d_{\phi}}. \tag{5}
\]

This approach was proposed in Nitanda and Suzuki (2018a) for learning residual networks by successively stacking layers, although there is a less explicit explanation about this viewpoint. In this paper, we also adopt this idea with several modifications for guaranteeing the global convergence property.

### 4 New Functional Gradient Boosting

We here propose a new functional gradient boosting for learning our models. The proposed method is essentially a combination of kernel learning for improving the approximation ability of kernel smoothing to functional gradients and the Frank-Wolfe method which is a basic scheme of boosting methods where weak learners are successively added into a prediction function. Usually, the Frank-Wolfe method is used for learning shallow neural networks (Barron 1993, Bach 2014, Ping et al. 2016), but we adapt it for learning deep residual networks by regarding residual blocks as weak learners. Since, in our method, residual blocks are composed of kernel smoothing of functional gradients and kernel functions are composition of a feature extractor and small networks, a modified Frank-Wolfe method makes a feature extractor one level deeper by adding this resblock. As a result, a network gradually grows as the algorithm proceeds, and finally, a deep residual network is obtained.

#### 4.1 Functional gradient boosting for learning deep residual networks

We herein describe the detail of the proposed method. We note that ResFGB [Nitanda and Suzuki (2018a)] is a special case of our method if a learning rate strategy is ignored and \( \epsilon_1 = 0 \). In our method, kernel functions used for kernel smoothing for resblocks are obtained by inner products of embedding maps. Let \( \mathcal{K}_1 \) and \( \mathcal{K}_2 \) be sets of such embedding maps \( \mathbb{R}^d \to \mathbb{R}^{D_i} \) (\( D_i \) may be set to \( \infty \)), and suppose \( 0 \in \mathcal{K}_i \) (\( i \in \{1, 2\} \)). In our theory, the sets \( \mathcal{K}_1 \) and \( \mathcal{K}_2 \) are sufficiently restrictive to guarantee global convergence of the proposed method and the generalization ability of an obtained residual network, which is described later.

For a current feature extractor \( \phi_t \), a kernel function is optimized to well approximate the functional gradient:

\[
\max_{\epsilon_1 \in \mathcal{K}_1} \| T_{k_{e_1}} \nabla \phi \mathcal{R}_n(\phi_t) \|^2_{H^d_{\phi}} \quad \text{and} \quad \max_{\epsilon_2 \in \mathcal{K}_2} \| T_{k_{e_2}} \nabla \phi \mathcal{R}_n(\phi_t) \|^2_{H^d_{\phi}}. \tag{6}
\]

Instead of (6), solving the following problem is practically better: for \( D' = d \),

\[
\min_{(\epsilon_1, \epsilon_2) \in \mathcal{K}_1 \times \mathcal{K}_2} \| \epsilon_1 + \epsilon_2 \circ \phi_1 - \nabla \phi \mathcal{R}_n(\phi_t) \|_{L^2_{\phi}(\nu_X)} \tag{7}
\]

because this problem (7) is much easier to solve and \( \nabla \phi \mathcal{R}_n(\phi_t) / \| \nabla \phi \mathcal{R}_n(\phi_t) \|_2 \) is a good solution of (6) if it is contained in a set: \( \mathcal{K}_1 + \mathcal{K}_2 \circ \phi_t \) as shown in Nitanda and Suzuki (2018a). However, we note that this approximation is just a heuristics and there is a room for improvement by exploring another approximation. We denote by \( \tau_1, \tau_2 \) approximate solutions of these problems.

Using kernels \( k_{e_1}, k_{e_2} \), a residual block that is a weak learner to be added into a feature extractor \( \phi_t \) in the proposed method is constructed as follows:

\[
\tau_1 = -\frac{T_{k_{e_1}} \nabla \phi \mathcal{R}_n(\phi_t)}{Z_1} - \frac{T_{k_{e_2}} \nabla \phi \mathcal{R}_n(\phi_t)}{Z_1}, \tag{8}
\]

where \( Z_1 \) is the normalization term: \( Z_1 = \| T_{k_{e_1}} \nabla \phi \mathcal{R}_n(\phi_t) \|_{\nu_X} \). We note that the components of a resblock, \( \tau_1 \) and \( \tau_2 \), are also defined in the above equation and are composed of neural networks:

\[
\epsilon_1 = A_1 \tau_1, \quad \epsilon_2 \circ \phi_t = A_2 \tau_2 \circ \phi_t, \quad \text{where} \quad A_1, A_2 \text{ are } (d, d)\text{-matrices}:
\]

\[
\begin{align*}
A_1 &= -\frac{1}{\nu_X} \sum_{i=1}^{n} \nabla \phi \mathcal{R}_n(\phi_t)(x_i) \tau_1 \Large(\nabla \phi \mathcal{R}_n(\phi_t)(x_i)\Large), \\
A_2 &= -\frac{1}{\nu_X} \sum_{i=1}^{n} \nabla \phi \mathcal{R}_n(\phi_t)(x_i) (\tau_2 \circ \phi_t) \Large(\nabla \phi \mathcal{R}_n(\phi_t)(x_i)\Large).
\end{align*}
\tag{9}
\]

Finally, a Frank-Wolfe-like update is performed as follows: for a given learning rate \( \eta_t \),

\[
\phi_{t+1} \leftarrow (1 - \eta_t) \phi_t + \eta_t \tau_t.
\]
As the right hand side of the above iteration is equal to 
\((1 - \eta_i)x + \eta_{i+1} \circ \phi_i(x) + \eta_{i+1} \circ \phi_i(x)\), we notice that 
\(\phi_{i+1}\) is one level deeper than \(\phi_i\) and a deep residual 
network is constructed.

We give a full description of the proposed method 
called ResFGB-FW in Algorithm 1. Although, we made 
the assumption where a weight \(\bar{\omega}\) for the last layer is 
pre-trained due to theoretical tractability, it may be 
practically better to optimize \(\bar{\omega}\) at each iteration, so 
that Algorithm 1 contains it as an option. We here note 
that this procedure is certainly similar to a progressive 
learning scheme in Belilovsky et al. (2019), except for 
details of the architecture of each layer, which exhibits 
comparable high performance with end-to-end training 
schemes on the image classification tasks.

Algorithm 1 ResFGB-FW

| Input: | \(S = (x_i, y_i)_{i=1}^n\), the number of iterations \(T\), 
<table>
<thead>
<tr>
<th></th>
<th>and learning rates (\eta_t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\phi_0 \leftarrow \text{id}, \bar{\omega} \leftarrow \arg\min_{\omega \in \mathbb{R}^d} \sum_{i=1}^{n} \mathcal{L}_{n}(w^T \phi_0))</td>
<td></td>
</tr>
<tr>
<td>for (t = 0 \to T-1) do</td>
<td></td>
</tr>
<tr>
<td>(Option) (\bar{\omega} \leftarrow \arg\min_{\omega \in \mathbb{R}^d} \sum_{i=1}^{n} \mathcal{L}_{n}(w^T \phi_t))</td>
<td></td>
</tr>
<tr>
<td>Get (\mathcal{G}) by solving (6) or (7) on (S)</td>
<td></td>
</tr>
<tr>
<td>(A_1 = -\frac{1}{\eta_i} \sum_{i=1}^{n} \nabla_{\phi} \mathcal{R}_{n}(\phi_i) x_i )</td>
<td></td>
</tr>
<tr>
<td>(A_2 = -\frac{1}{\eta_i} \sum_{i=1}^{n} \nabla_{\phi} \mathcal{R}_{n}(\phi_i) x_i \phi_i^T (x_i))</td>
<td></td>
</tr>
<tr>
<td>(\mathcal{G}<em>{t} = A_1 \mathcal{G}</em>{t-1} + A_2 \phi_i)</td>
<td></td>
</tr>
<tr>
<td>(\phi_{t+1} \leftarrow (1 - \eta_i) \phi_t + \eta_i \mathcal{G}_{t})</td>
<td></td>
</tr>
<tr>
<td>end for</td>
<td></td>
</tr>
<tr>
<td>Return (\bar{\omega}^T \phi_T)</td>
<td></td>
</tr>
</tbody>
</table>

Remark There are two major differences from the method in 
[Nitanda and Suzuki (2018a)]; one is that the 
output \(\phi_i(x)\) is reduced at the rate of \(1 - \eta_i\) stemming 
from the Frank-Wolfe method and the other is that res-
blocks receive input data directly like DenseNet [Huang 
et al., 2017]. These modifications lead to provable sta-
tistical guarantees without weak learning assumptions 

5 Convergence Analyses

We here provide the convergence analysis of Algorithm 1. 
To do so, we have to specify the architecture of \(K_i\) 
and provide the required condition on resblocks 
obtained in Algorithm 1 to guarantee the global conver-
gence property. We assume \(K_i\) are sets of shallow neural 
networks: \(F_i(x)\) with norm constraints. Note that the 
parameters of resblocks introduced in Section 2 can be 
obtained by

\[ C_i = A_i F_i, \quad D_i = G_i. \] (10)

The set of learnable functions by our method is 
described as follows. We denote by \(D_{K_i}\) the union of unit 
balls in RKHSs defined by \(e_1 \in K_1\):

\[ D_{K_i} \equiv \bigcup_{e_1 \in K_1} \{ \xi \in \mathcal{H}_{K_1}^d \mid \| \xi \|_{\mathcal{H}_{K_1}^d} \leq 1 \}. \]

Then, we define \(\text{conv}(D_{K_i})\) as the convex combination 
of \(D_{K_i}\). We here make the following assumptions for 
theoretical guarantees.

Assumption 2.

(A1) Let \(\Lambda_i^F, \Lambda_i^G (i \in \{1, 2\})\) be positive values and \(\sigma\) 
be a 1-Lipschitz activation function. Let \(K_1, K_2\) be sets 
of shallow neural networks:

\[ K_i = \{ z \to F_i \sigma(G_i z) \mid \sum_j \| (F_i)_{*j} \|_2 \leq \Lambda_i^F, \max_k \| (G_i)_{k,*} \|_1 \leq \Lambda_i^G \}. \]

In addition, \(\| e_i(z) \|_2 \leq \sqrt{K} \) \((i \in \{1, 2\}, K > 0)\) over 
\(z \in \mathbb{R}^d\) and the maximum singular value of \(\bar{\omega}\) is upper 
bounded by \(\sum_{\bar{\omega}}\).

(A2) Resblocks: \(\mathcal{G}_1 = A_1 \mathcal{G}_1 + A_2 \phi_i\) satisfy the following:

\[ -\langle \nabla_{\phi} \mathcal{R}_{n}(\phi_i), \mathcal{G}_1 \rangle_{L_2^2(u^X)} \geq \frac{1}{2} \max_{\xi \in D_{K_1}} \langle \nabla_{\phi} \mathcal{R}_{n}(\phi), \xi \rangle_{L_2^2(u^X)}, \]

\[ -\langle \nabla_{\phi} \mathcal{R}_{n}(\phi_i), \mathcal{G}_2 \rangle_{L_2^2(u^X)} \geq \max_{\xi \in D_{K_1}} \langle \nabla_{\phi} \mathcal{R}_{n}(\phi), \xi \rangle_{L_2^2(u^X)}. \]

(A3) For a training set \(S = (x_i, y_i)_{i=1}^n\) sampled from \(\nu^X\), there exist \(M_n, \gamma_n > 0\) and \(\psi_n \in \text{conv}(D_{K_1})\) such that for \(v_{i,y} \equiv \bar{\omega}^T \psi_n(x_i), \forall i \in \{1, \ldots, n\}\),

\[ v_{i,y} \geq M_n + \gamma_n, v_{i,y} \leq M_n \quad (\forall y \neq y_i). \]

Remark Note that since the first component of res-
block: \(-\nu_i = -A_1 \nu_i\) maximizes \(\langle \nabla_{\phi} \mathcal{R}_{n}(\phi_i), \xi \rangle_{L_2^2(u^X)}\) 
through equations (1) and (5) over \(\xi \in D_{K_1}\), a kernel 
\(\mathcal{G}_1\) always satisfies the condition (A2) if problems (6) 
are solved exactly, but (A2) also allows approximate 
solution thanks to a support by \(\nu_i\). In other words, 
a deep architecture alleviates the requirement of the 
optimality of weak learners for subproblems and is 
certainly useful in enhancing the global convergence. 
The condition (A3) means that a given training set is se-
parable by \(\text{conv}(D_{K_1})\) with a margin \(\gamma_n\). We note that 
this assumption is reasonable because \(\text{conv}(D_{K_1})\) is 
essentially a set of infinite-width shallow neural networks 
and has a universal approximation property. Hence, a 
margin \(\gamma_n\) actually exists as long as a data distribution 
separable by a continuous function.

To ensure the convergence and generalization prop-
rerties, we should derive upper bounds on norms of
parameters of resblock $\tau$ and norm $\|\tau(z)\|_2$, but it is not obvious because resblocks contain a normalization term $Z_1$. An interesting technical contribution is to derive the lower bound on $Z_1$ under Assumption 2.

**Proposition 2.** Suppose Assumption [2] holds. Then, iterates of Algorithm 1 satisfy

$$\frac{\gamma_n}{4\Sigma_\phi} \| \nabla_\phi R_n(\phi) \|_{L^4_t(\nu_n^\infty)} \leq \| T_{k^n} \nabla R_n(\phi) \|_{\psi_n^{k^n}}.$$

(11)

Proposition 2 says that the kernel smoothing in Algorithm 1 produces a sufficient reduction of objective in a function space as the margin $\gamma_n$ is large. As a result, norms of parameters and outputs of resblocks can be made small.

**Proposition 3.** Suppose Assumption [2] holds. Then, it follows that for resblocks $i_2 = A_i F_i \sigma(G_i)$ obtained in Algorithm 1,

$$\|(A_i F_i)_{\alpha^*}\|_1 \leq \frac{4}{\gamma_n} \Sigma_\phi \Lambda_F \sqrt{K},$$

(12)

$$\| \iota_1 \iota_2 \circ \phi \|_{L^2_t(\nu_n^\infty)} \leq \frac{4}{\gamma_n} \Sigma_\phi \psi K.$$  

(13)

This proposition is useful for both convergence and generalization guarantees, namely, the equation (12) is used for deriving a generalization bound by substituting it into the bound in Theorem 1, and the equation (13) is used for deriving a convergence rate.

**Global Convergence.** The next theorem shows the convergence property of Algorithm 1 in a better model than $\text{conv}(D_{K_1})$, which can be shown by adapting a proof for the vanilla Frank-Wolfe method in [Jaggi, 2013] to our method. For simplicity, we set $R^*_n \overset{\text{def}}{=} \inf_{\phi \in \text{conv}(D_{K_1})} R_n(\phi)$ and set for $t_0, \Lambda_\infty > 0$,\n
$$C \overset{\text{def}}{=} \max \left\{ \right.$$

$$t_0 (R_n(\phi_0) - R^*_n), 2L \left( \Lambda_\infty + \frac{16 \Sigma_\psi \Lambda F \sqrt{K}}{\gamma_n} \right)^2 \left.$$

$$\left. \right\}.$$

**Theorem 2 (Convergence rate).** Suppose Assumption [2] holds and $\|x\|_2 \leq \Lambda_\infty$ on $X$. Then, iterates of Algorithm 1 with learning rates $\eta_i = \frac{2}{t_0} \iota_i$ ($t_0 \geq 0$) satisfy

$$R_n(\phi_T) - R^*_n \leq \frac{C}{T + t_0}.$$

Note that this theorem provides a global convergence guarantee. An important point of our analysis is that a type of weak learning condition is not required unlike existing studies [Huang et al., 2018; Nitanda and Suzuki, 2018a]. In addition, we remark that the convergence becomes faster when the input distribution has a high degree of linear separability and when using a sufficient large $t_0$.

**Generalization Bound.** By combining Proposition 2, 3, and Theorem 2 with generalization bound in Theorem 1, we obtain the main result in this study. To describe the result, we define $e_n \overset{\text{def}}{=} (c - 1) \exp(-\gamma_n)$.

We can easily see that $e_n$ is an upper bound on $R_n^*$ as follows. We set $f = \psi \phi_n$ where $\psi_n \in \text{conv}(D_{K_1})$ is defined in Assumption 2 (A3). Under Assumption 2 (A3), for $i \in \{1, \ldots, n\}$,

$$l(f(x), y_i) = \log(1 + \sum_{y' \neq y_i} \exp(f(x)) - f(y_i)(x_i)) \leq \log(1 + (c - 1) \exp(-\gamma_n)) \leq e_n.$$

Thus, taking an expectation with respect to $\nu_n^\infty$, we see that $R^*_n \leq R_n(\psi_n) \leq e_n$. Moreover, we note that $e_n \rightarrow 0$ as $\gamma_n \rightarrow \infty$.

**Corollary 1 (Main result).** Suppose Assumption 2 holds and $\|x\|_2 \leq \Lambda_\infty$ on $X$. Assume $\Lambda_\infty^2 \Lambda^2 \leq \gamma_n/8 \Sigma_\psi \sqrt{K}$. Fix $\delta > 0$. Then, for $\forall \rho > 0$ with probability at least $1 - \rho$ over random choice from $\nu_n^\infty$, a predictor $f_T$ obtained by Algorithm 1 with $\eta_i = \frac{2}{t_0 \iota_i}$ ($t_0 \geq 2$) and $T$-iterates satisfies the following bound.

$$P_{\nu}[m_f(X,Y) \leq 0] \leq (1 + \exp(\delta)) \left( \frac{e_n + \frac{C}{T + t_0}}{\delta \sqrt{n}} \right) + \left( \frac{1}{2n} \log \rho^{-1} \right).$$

From Corollary 1, we can see that a worse dependence on the network depth disappears; hence, the best generalization performance is achieved when $T \rightarrow \infty$. Specifically, if $\delta = \gamma_n/2$ and $\gamma_n$ are uniformly positive, then we get for a sufficiently large $T$,

$$P_{\nu}[m_f(X,Y) \leq 0] \leq O \left( \frac{e_n + \frac{1}{\gamma_n} \sqrt{n}}{\gamma_n \sqrt{n}} + \frac{1}{2n} \log \rho^{-1} \right).$$

From this bound, we can see that the standard margin $\gamma_n$ helps the generalization ability of the method, that is, the complexity term is considerably reduced when the margin $\gamma_n$ is sufficiently large.

6 Numerical Experiments

We present experimental results on the binary and multiclass classification tasks. Our implementation is done using Theano (Theano Development Team, 2016). The experiments are conducted on Intel Core i7-5700K and TITAN Xp 12GB with Ubuntu 16.04 (64-bit). We compare Algorithm 1 with ResFGB (Nitanda and Suzuki, 2018a), multilayer perceptron (MLP), support vector machine (SVM), random forest (RF), and gradient boosting methods (GBM).
We use the following datasets: letter, usps, ijcnn1, mnist, and covtype. We first divide each train dataset randomly into two sets: 80% for training and the rest for validation. We run each method on the train dataset with several hyperparameter settings and choose a setting providing the best classification accuracy on the validation dataset. We finally train each model on an entire train dataset and report a classification accuracy on the test dataset. We perform the above procedure 5-times.

We explain hyper-parameter settings. For Algorithm 1, three hidden-layers neural networks with 1000-nodes per layer are used as $e_i \in K_i$ ($i \in \{1, 2\}$) to define resblocks. Linear classifiers and $e_i$ are trained by Nesterov’s momentum method at each loop. The number of iterations $T$ and a parameter $t_0 \in \{10, 100, 1000\}$ for learning rates $\eta_t = 2/(t+t_0)$ are tuned based on the performance on the validation set. For ResFGB (Nitanda and Suzuki, 2018a), constant learning rates are chosen from $\{10^{-3}, 10^{-2}, 10^{-1}\}$ and similar settings as Algorithm 1 are adopted for the other hyper-parameters. For MLP, the ReLU is used as an activation and network depth is set to three, four, or five layers. The number of hidden units per layer is 100 or 1000. For SVM, a random Fourier feature (Rahimi and Recht, 2007) is adopted with an embedding dimension of $10^3$ or $10^4$. For RF, the maximum depth is set to 10, 20, or 30 and the number of trees is set to 100, 500, or 1000. For GBM, the learning rate is chosen from $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ with the maximum number of trees being set to 1000, and number of leaves in one tree is chosen from $\{2^4, 2^5, \ldots, 2^{10}\}$.

Table 1 shows the mean classification accuracies and standard deviations and Figure 2 depicts learning curves on training and test datasets. As seen in the table, the proposed method shows superior or comparable performance over other state-of-the-art methods. Therefore, we can conclude that our proposed method is certainly useful for training deep ResNets with a provable statistical guarantee.

7 Conclusion

We have proposed a new functional gradient boosting by combining ResFGB with the Frank-Wolfe method for learning deep ResNets. In theoretical analyses, we have provided a generalization bound via a global convergence analysis of the method under a margin condition instead of a weak learning condition. Compared to the previous study, our bound has two important points; one is the elimination of a worse dependence on the network depth, and the other is a significant improvement by the existence of a learnable classifier with a large margin on train datasets. Finally, we have demonstrated superior empirical performance of the proposed method over the state-of-the-art methods. An interesting future work is to adjust our method to the network architecture search by combining with a usual end-to-end training such as stochastic gradient descent method. To do so, it will be necessary to develop an effective criterion to switch the functional gradient boosting and the end-to-end training schemes.

Acknowledgement AN was partially supported by JSPS Kakenhi (19K20337) and JST-PRESTO. TS was partially supported by JSPS Kakenhi (26280009, 15H05707 and 18H03201), Japan Digital Design and JST-CREST.
References


