Sparse Deep Neural Networks Through $L_{1,\infty}$-Weight Normalization

Zhouwang Yang

University of Science and Technology of China

June 3, 2021

Outline

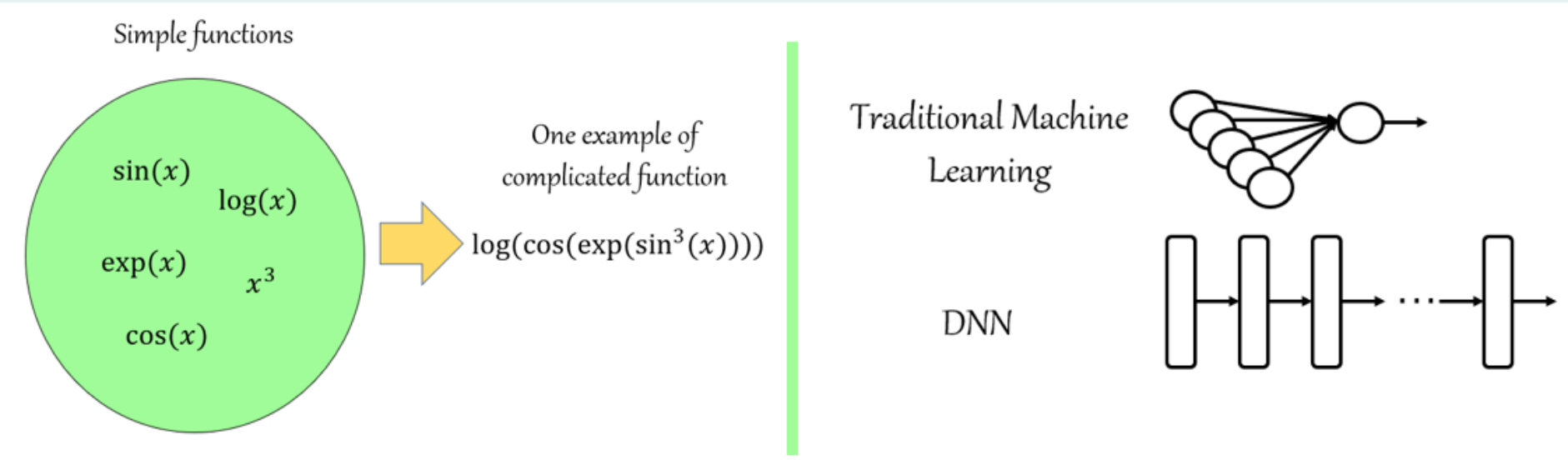
Outline

Outline

DNNs

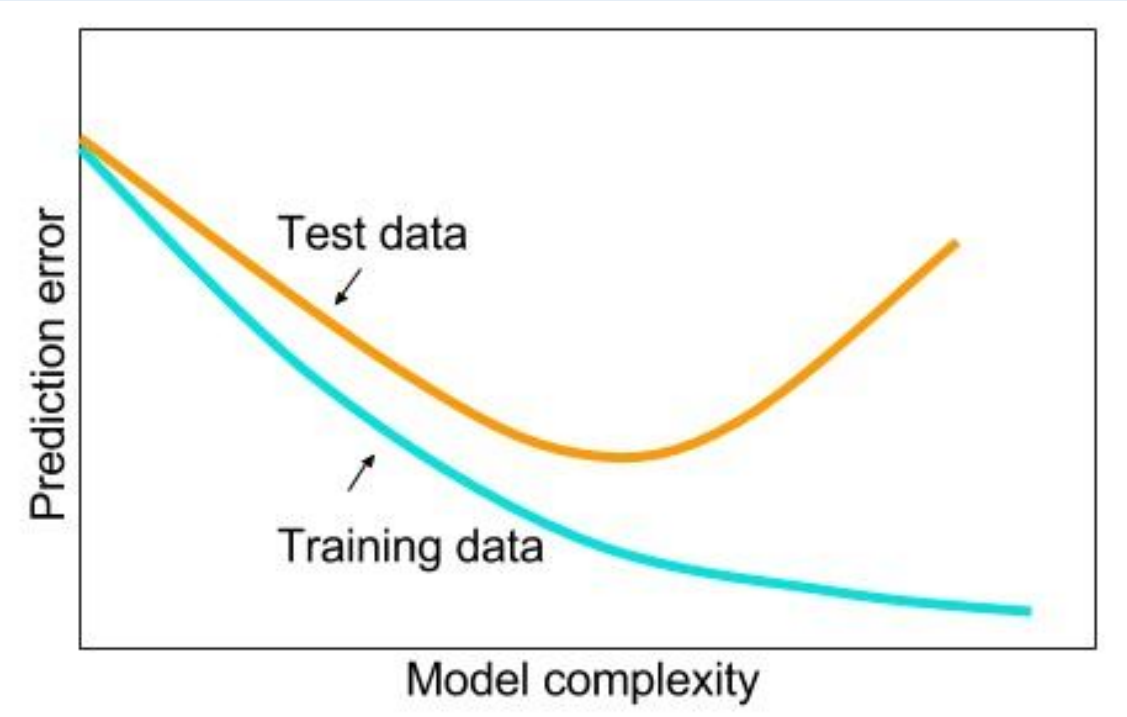
Deep neural networks (DNNs) have recently attracted a lot of attentions due to their successful applications.

Traditional methods of machine learning, like logistic regression and SVM, are hard to express the complicated functions with limited computing units. But DNNs, with deep non-linear construction, have great learning ability.



Dilemma of DNNs

However, when the model gets small training loss, overfitting becomes a notorious feature of DNNs.



Strategies for Dealing with the Dilemma

Lots of methods have also been developed to address the issue of overfitting.

- L_0, L_1, and L_2 regularization
- Weight sharing
- Dropout
- ...

Related Work of Sparsity in DNNs

Recently, empirical evidence suggests that inducing sparsity can relieve overfitting and save computation resources. Some related work is done for inducing sparsity in DNNs.

- Louizos et al. (2018) [4]: learned the L_0 regularization, and studied the sparsity.
- Srinivas et al. (2017) [8]: introduced trainable gate variables to restricts the total number of parameters.

Cons: theoretical investigations or justifications on sparse DNNs are less explored in the literature.

Related Work of Theoretical Investigation

The dependence of the generalization bounds on the depth k of the neural networks was investigated in prior studies.

- Bartlett (1998) [1]; Neyshabur et al. (2015) [7]; Sun et al. (2016) [9]: 2^k.
- Bartlett et al. (2017) [2]; Neyshabur et al. (2018) [6]: $k^{3/2}$.
- Golowich et al. (2018) [3]: $k^{1/2}$, without bias neurons.

Note that results in [3] are without bias neurons, and hard to extend to cases with bias neurons.

Related Work of Theoretical Investigation

Group norm is often used in theoretical investigations of weight normalization.

The (p, q)-norm of a matrix $A_{s_1 \times s_2}$ is defined as

$$\|A\|_{p,q} = \left(\sum_{j=1}^{s_2} \left(\sum_{i=1}^{s_1} |a_{ij}|^p \right)^{\frac{q}{p}} \right)^{\frac{1}{q}}.$$

Note that the general (p, q)-norm regularization does not result in sparsity.

$L_{1,\infty}$-Weight Normalization

We borrow the strength of weight normalization and induce the sparsity by bounding the $L_{1,\infty}$ norm of the weight matrix for each layer.

The $L_{1,\infty}$ norm of a matrix $A_{s_1 \times s_2}$ is defined as

$$\|A\|_{1,\infty} = \max_{1 \le j \le s_2} \left(\sum_{i=1}^{s_1} |a_{ij}| \right).$$

The norm has good sparsity, so $L_{1,\infty}$-weight normalization has good generalization capacity.

$L_{1,\infty}$-Weight Normalization

The overall goal of our study is listed as follows.

- Theoretically investigate the generalization error bounds for both regression and classification DNNs with bias neurons under the $L_{1,\infty}$-weight normalization.
- Develop an easily implemented algorithm to practically obtain a sparse neural network.
- Perform experiments to validate the theory and demonstrate the effectiveness of the resulting approach.

Outline

The General Prediction Problem

Assume that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are n independent random variables on $\mathcal{X} \subseteq \mathbb{R}^{m_1}$, $\mathrm{u}_1, \ldots, \mathrm{u}_n$ are on $\mathcal{U} \subseteq \mathbb{R}^{m_2}$, $y_1, \ldots, y_n$ are on $\mathcal{Y} \subseteq \mathbb{R}$, and the noise $\varepsilon_1, \ldots, \varepsilon_n$ are independent while satisfying that $\mathbb{E}(\varepsilon_i) = 0$.

The general prediction problem is defined as

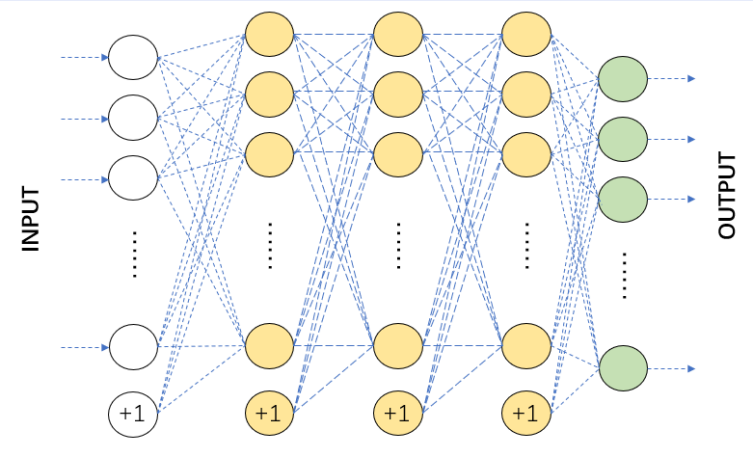
$$\begin{aligned} \mathrm{u}_i &= f(\boldsymbol{x}_i) + \varepsilon_i \\ y_i &= \phi(\mathrm{u}_i), \end{aligned} \tag{1}$$

where $f : \mathcal{X} \to \mathcal{U}$ is an unknown function, and $\phi : \mathcal{U} \to \mathcal{Y}$ is a fixed function related to the prediction problem.

- Regression: consider ϕ is an identity transformation and $m_2 = 1$.
- Classification: consider $\phi = \text{argmax}$ and $\mathcal{Y} = \{1, 2, \cdots, m_2\}$.

Notations of DNNs

We begin with some notations for fully-connected neural networks.



A neural network on $\mathbb{R}^{d_0} \to \mathbb{R}^{d_{k+1}}$ with k hidden layers is defined by a set of $k+1$ affine transformations

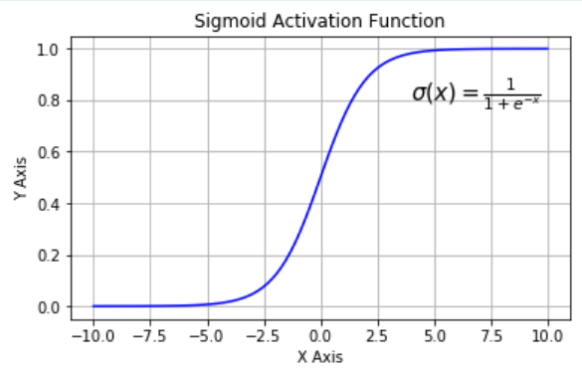
$$T_1 : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1}, T_2 : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}, \cdots, T_{k+1} : \mathbb{R}^{d_k} \to \mathbb{R}^{d_{k+1}},$$

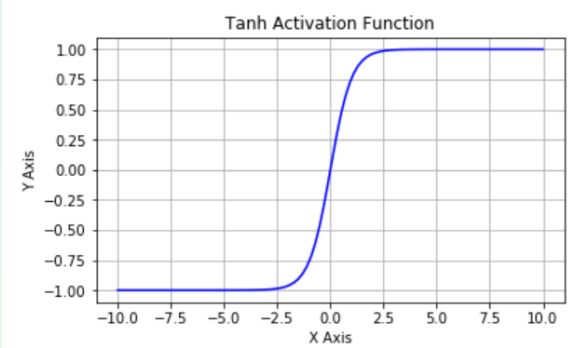
and an activation function σ.

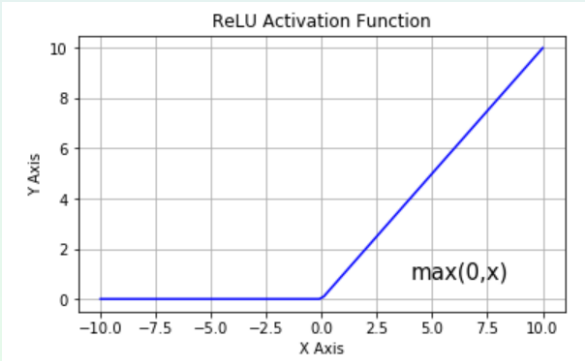
In this talk we consider activation functions satisfying that $\sigma(0) = 0$. Note that this condition holds for widely used activation functions including Sigmoid, Tanh, and ReLU.

Notations of DNNs: activation function

- Sigmoid: $\sigma_{\text{Sigmoid}}(x) = \dfrac{1}{1+e^{-x}}$. It ranges in $(0, 1)$.
- Tanh: $\sigma_{\text{Tanh}}(x) = \dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$. It ranges in $(-1, 1)$.
- ReLU: $\sigma_{\text{ReLU}}(x) = \max(0, x)$. It ranges in $(0, \infty)$.







Notations of DNNs

The affine transformations are parameterized by $T_\ell(\boldsymbol{x}^{(\ell-1)}) = \boldsymbol{W}_\ell^T \boldsymbol{x}^{(\ell-1)} + \boldsymbol{b}_\ell$, where $\boldsymbol{W}_\ell \in \mathbb{R}^{d_{\ell-1} \times d_\ell}, \boldsymbol{b}_\ell \in \mathbb{R}^{d_\ell}$ for $\ell = 1, \cdots, k+1$. The function represented by this neural network is

$$f(\boldsymbol{x}) = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ \sigma \circ T_1 \circ \boldsymbol{x}.$$

So we got

$$\begin{aligned} f_1(\boldsymbol{x}) &= T_1 \circ \boldsymbol{x} \triangleq \langle \tilde{\mathrm{V}}_1, (1, \boldsymbol{x}^T)^T \rangle \\ f_\ell(\boldsymbol{x}) &= T_\ell \circ \sigma \circ f_{\ell-1}(\boldsymbol{x}) \triangleq \langle \tilde{\mathrm{V}}_\ell, (1, \sigma \circ f_{\ell-1}^T(\boldsymbol{x}))^T \rangle, \end{aligned}$$

for $\ell = 2, \ldots, k+1$, where $\tilde{\mathrm{V}}_\ell = (\boldsymbol{b}_\ell, \boldsymbol{W}_\ell^T)^T \in \mathbb{R}^{(d_{\ell-1}+1) \times d_\ell}$.

Sparse DNNs

Define $\mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}$ as the collection of all sparse DNNs

$$f(\boldsymbol{x}) = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ \sigma \circ T_1 \circ \boldsymbol{x}$$

satisfying

(a) It has k hidden layers. The number of neurons in the ℓth hidden layer is d_ℓ for $\ell = 1, 2, \cdots, k$. The dimension of input is d_0, and output d_{k+1}.

(b) $\|T_\ell\|_{1,\infty} \triangleq \left\|\tilde{\mathrm{V}}_\ell\right\|_{1,\infty} \leq c$ for $\ell = 1, \cdots, k$.

(c) The L_1 norm of the jth column of $\tilde{\mathrm{V}}_{k+1}$ is bounded by the jth element of $\boldsymbol{o}$: $\left\|\tilde{\mathrm{V}}_{k+1}[\cdot, j]\right\|_1 \leq o_j$ for $j = 1, \cdots, d_{k+1}$.

Furthermore, define the collection of the sparse DNNs without any constraint on the output layer as

$$\mathcal{S}_c^{k,\boldsymbol{d},\sigma} = \cup_{\boldsymbol{o} \geq 0} \mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}.$$

Outline

Outline

Definition of Rademacher Complexities

Rademacher complexity is commonly used to measure the complexity of a hypothesis class with respect to a sample or a probability distribution, and to analyze generalization bounds.

The *empirical Rademacher complexity* of the hypothesis class $\mathcal{F}$ with respect to a data set $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ is defined as:

$$\widehat{\mathfrak{R}}_S(\mathcal{F}) = \mathbb{E}_\epsilon \left[\sup_{f \in \mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(\boldsymbol{x}_i) \right) \right]$$

where $\epsilon = \{\epsilon_1, \ldots, \epsilon_n\}$ are n independent Rademacher random variables.

The *Rademacher complexity* of the hypothesis class $\mathcal{F}$ with respect to n samples is defined as:

$$\mathfrak{R}_n(\mathcal{F}) = \mathbb{E}_{S \sim \mathcal{D}^n} \left[\widehat{\mathfrak{R}}_S(\mathcal{F}) \right].$$

Rademacher Complexities of $\mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}$

We bound the Rademacher complexity of $\mathcal{SN}_{c,o}^{k,\boldsymbol{d},\sigma}$ when the output dimension is 1 using following theorem.

Theorem 1

Fix $k \geq 0, c, o > 0, d_\ell \in \mathbb{N}_+$ for $\ell = 1, \cdots, k$, and $d_{k+1} = 1$, then for any set $S = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\} \subseteq \mathcal{X} = [-1, 1]^{m_1}$, we have

$$\widehat{\mathfrak{R}}_S(\mathcal{SN}_{c,o}^{k,\boldsymbol{d},\sigma}) \leq o\sqrt{\frac{(k+1)\log 16}{n}}\left(\sum_{\ell=0}^{k}(\gamma_\sigma c)^\ell + (\gamma_\sigma c)^k\right) + o(\gamma_\sigma c)^k\sqrt{\frac{2\log(2m_1)}{n}}.$$

Where γ_σ is the Lipschitz constant of the activation σ. Note that ReLU and tanh are both 1-Lipschitz continuous, while sigmoid is 0.25-Lipschitz continuous.

Rademacher Complexities of $\mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}$

Remark 1

When $\log(m_1)$ *is small, we briefly summarize the dependence of the above bound on* k *under different choice of* c*:*

- $\gamma_\sigma c < 1$: $O(\sqrt{k}\frac{1-(\gamma_\sigma c)^{k+1}}{1-\gamma_\sigma c})$
- $\gamma_\sigma c = 1$: $O(k^{\frac{3}{2}})$
- $\gamma_\sigma c > 1$: $O(\sqrt{k}\frac{(\gamma_\sigma c)^{k+1}-1}{\gamma_\sigma c-1})$

The complexity bound does not depend on the network width. In addition to the product of the $L_{1,\infty}$ *norms of each layer, the complexity bound depends on the depth* k *by* $O(\sqrt{k})$ *when* $\gamma_\sigma c < 1$*, and* $k^{3/2}$ *when* $\gamma_\sigma c = 1$*.*

Rademacher Complexities of $\mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}$

We also provide a lemma to bound the output of a sparse DNN, which is useful to derive the generalization error bounds.

Lemma 2

For any $f \in \mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}$ *and* $\boldsymbol{x} \in \mathcal{X}$,

$$\|f(\boldsymbol{x})\|_\infty \leq \|\boldsymbol{o}\|_\infty \max\left(1, (\gamma_\sigma c)^k\right).$$

A Useful Lemma for Generalization Bound

Let $L(f(\cdot),\cdot):\mathcal{X}\times\mathcal{Y}\to\mathbb{R}$ be the loss function. Define the expected and empirical risks, respectively, as

$$\mathbb{E}_L(f)=\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[L(f(\boldsymbol{x}),y)],\quad \widehat{\mathbb{E}}_L(f)=\frac{1}{n}\sum_{i=1}^{n}L(f(\boldsymbol{x}_i),y_i),$$

for any hypothesis $f:\mathcal{X}\to\mathcal{Y}$.
Generally, a learning algorithm is said to *overfit* if it is more accurate in fitting known data but less accurate in predicting new data.
Mathematically, the difference between the expected risk and the empirical risk, called *generalization error*. Let

$$\mathcal{E}_L(f)=\left|\ \mathbb{E}_L(f)-\widehat{\mathbb{E}}_L(f)\ \right|.$$

A Useful Lemma for Generalization Bound

The lemma below is a more general version of Mohri et al. (2012) [5, Theorem 3.1] and the proof is very similar to the original one.

Lemma 3

Let z be a random variable of support $\mathcal{Z}$ and distribution $\mathcal{D}$. Let $S = \{z_1 \dots z_n\}$ be a data set of n i.i.d. samples drawn from $\mathcal{D}$. Let $\mathcal{F}$ be a hypothesis class satisfying $\mathcal{F} \subseteq \{L(f(\cdot), \cdot) \mid L(f(\cdot), \cdot) : \mathcal{Z} \to [0, A_0]\}$. Fix $\delta \in (0, 1)$. With probability at least $1 - \delta$ over the choice of S, the following holds for all $L(f(\cdot), \cdot) \in \mathcal{F}$:

$$\mathcal{E}_L(f) \leq 2\mathfrak{R}_n(\mathcal{F}) + A_0 \sqrt{\frac{\log(1/\delta)}{2n}}.$$

Outline

Notations

Regression: consider ϕ is an identity transformation and $m_2 = 1$.

Assume the following conditions:

- $(\boldsymbol{x}, y)$ is a random variable of support $\mathcal{X} \times \mathcal{Y}$, $\mathcal{D}$ is a distribution, and $S = \{(\mathrm{x}_i, y_i)\}_{i=1}^n$ is a dataset of n i.i.d. samples drawn from $\mathcal{D}$.
- The loss function $L(f(\boldsymbol{x}), y) : \mathcal{Y} \times \mathcal{Y} \to [0, A_0]$, is ρ Lipschitz continuous on its first argument.
- For any $y \in \mathcal{Y}$, $|y| \leq B_0$.

Lemma for Generalization Bound of Regression

We use following lemma to bound the generalization error of regression.

Lemma 4

Fix $\delta \in (0,1)$, $c, o > 0$, the number of hidden layers $k \in [0, \infty)$, and widths $\boldsymbol{d} \in \mathbb{N}_+^{k+2}$ with $d_0 = m_1$ and $d_{k+1} = 1$. With probability at least $1-\delta$ over the choice of S, every $f \in \mathcal{SN}_{c,o}^{k,\boldsymbol{d},\sigma}$ satisfies that

$$\mathcal{E}_L(f) \leq A_0 \sqrt{\frac{\log(1/\delta)}{2n}} + \frac{2o\rho}{\sqrt{n}} * \left[\sqrt{(k+1)\log 16} \left(\sum_{\ell=0}^{k} (c\gamma_\sigma)^\ell + (c\gamma_\sigma)^k \right) + (c\gamma_\sigma)^k \sqrt{2\log(2m_1)} \right].$$

Mean Square Error

Mean square error: $L_{se}(f(\boldsymbol{x}), y) = \frac{1}{2}(y - f(\boldsymbol{x}))^2$.

Generalization bound holds uniformly in $\mathcal{S}_c^{k,\boldsymbol{d},\sigma}$:

Theorem 5

Fix $\delta \in (0,1)$, $c > 0$, the number of hidden layers $k \in [0, \infty)$, and widths $\boldsymbol{d} \in \mathbb{N}_+^{k+2}$ with $d_0 = m_1$ and $d_{k+1} = 1$. With probability at least $1 - \delta$ over the choice of S, for every sparse DNN $f_T = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ \sigma \circ T_1 \in \mathcal{S}_c^{k,\boldsymbol{d},\sigma}$, we have

$$
\begin{aligned}
\mathcal{E}_{Lse}(f_T) \leq & \sqrt{\frac{\log(\frac{1}{\delta}) + 2\log(\|T_{k+1}\|_1 + 2)}{2n}} * \\
& \left(B_0^2 + (\|T_{k+1}\|_1 + 1)^2 \max\left(1, (\gamma_\sigma c)^{2k}\right)\right) + \\
& \frac{2}{\sqrt{n}} \left(B_0 + (\|T_{k+1}\|_1 + 1) \max\left(1, (\gamma_\sigma c)^k\right)\right) (\|T_{k+1}\|_1 + 1) * \\
& \left[\sqrt{(k+1)\log 16} \left(\sum_{\ell=0}^{k} (\gamma_\sigma c)^\ell + (\gamma_\sigma c)^k\right) + (\gamma_\sigma c)^k \sqrt{2\log(2m_1)}\right].
\end{aligned}
$$

Mean Square Error

Remark 2

When $\log(m_1)$ *is small, the above generalization bound is:*

- $\gamma_\sigma c < 1$: $O\left(\frac{\sqrt{k}(B_0+\|T_{k+1}\|_1)^2}{\sqrt{n}}\frac{1-(\gamma_\sigma c)^{k+1}}{(1-\gamma_\sigma c)}\right)$
- $\gamma_\sigma c = 1$: $O\left(\frac{k^{3/2}(B_0+\|T_{k+1}\|_1)^2}{\sqrt{n}}\right)$
- $\gamma_\sigma c > 1$: $O\left(\frac{B_0^2}{\sqrt{n}}+\frac{\sqrt{k}(B_0+(\gamma_\sigma c)^k\|T_{k+1}\|_1)\|T_{k+1}\|_1}{\sqrt{n}}\frac{(\gamma_\sigma c)^{k+1}-1}{\gamma_\sigma c-1}\right)$

Mean Absolute Error

Mean absolute error: $L_{ae}(f(\boldsymbol{x}), y) = |y - f(\boldsymbol{x})|$.

Also, we can drive the result that the upper bound will depend on the product of the $L_{1,\infty}$ norm of each layer by

$$O(\|T_{k+1}\|_1 (\gamma_\sigma c)^k),$$

as mean absolute error is 1-Lipschitz continuous on its first argument.

Remark 3

Condition $|y| \leq B_0$ *is not always met, especially when* $y = f(\boldsymbol{x}) + \epsilon$, *where* $\epsilon \sim N(0,1)$. *However, it is still reasonable to assume that* $\mathbb{P}(|y| \leq B_0) \geq 1 - \delta_0$ *for some fixed* $\delta_0 > 0$ *and* $B_0 > 0$. *Under this alternative assumption, The inequation is replaced by*

$$\begin{aligned}\mathcal{E}_{L_{ae}}(f_T) \leq & (1-\delta_0)\frac{\left(B_0 + (\|T_{k+1}\|_1 + 1)(c\gamma_\sigma)^k\right)^2}{\sqrt{n}} * \\ & \left(\sqrt{\log\sqrt{\frac{2}{\delta}} + \log(\|T_{k+1}\|_1 + 2)}\right. \\ & \left. + 2\sqrt{(k+3)\log 4} + 2\sqrt{2\log(2m_1)}\right).\end{aligned}$$

Outline

Notations

Classification: consider $\phi = \text{argmax}$ and $\mathcal{Y} = \{1, 2, \cdots, m_2\}$.

Assume the following conditions:

- $(\boldsymbol{x}, y)$ is a random variable of support $\mathcal{X} \times \mathcal{Y}$, $\mathcal{D}$ is a distribution, and $S = \{(x_i, y_i)\}_{i=1}^n$ is a dataset of n i.i.d. samples drawn from $\mathcal{D}$.
- For any $\boldsymbol{z} \in \mathcal{Z} \subseteq \mathbb{R}^{m_2}$ and $j = 1, \cdots, m_2$, $|\boldsymbol{z}[j]| \leq B_j$.
- The loss function $L(f(\boldsymbol{x}), y) : \mathcal{Z} \times \mathcal{Y} \to [0, A_0]$, satisfies that

$$|L(f_1(\boldsymbol{x}), y) - L(f_2(\boldsymbol{x}), y)| \leq \rho \|f_1(\boldsymbol{x}) - f_2(\boldsymbol{x})\|_2$$

for any $\boldsymbol{x} \in \mathcal{X}, y \in \mathcal{Y}$.

Lemma for Generalization Bound of Classification

We use following lemma to bound the generalization error of classification.

Lemma 6

Fix $\delta \in (0,1)$, $c > 0$, $\boldsymbol{o} \geq 0$, the number of hidden layers $k \in [0, \infty)$, and widths $\boldsymbol{d} \in \mathbb{N}_+^{k+2}$ with $d_0 = m_1$ and $d_{k+1} = m_2$. With probability at least $1 - \delta$ over the choice of S, every $f \in \mathcal{SN}_{c,\boldsymbol{o}}^{k,\boldsymbol{d},\sigma}$ satisfies that

$$\mathcal{E}_L(f) \leq A_0 \sqrt{\frac{\log(1/\delta)}{2n}} + \frac{2\sqrt{2}\rho}{\sqrt{n}} \left(\sum_{j=1}^{m_2} o_j \right) * \left[\sqrt{(k+1)\log 16} \left(\sum_{\ell=0}^{k} (c\gamma_\sigma)^\ell + (c\gamma_\sigma)^k \right) + (c\gamma_\sigma)^k \sqrt{2\log(2m_1)} \right].$$

Cross Entropy

The cross-entropy loss function is defined as

$$L_{ce}(f(\boldsymbol{x}), y) = -\log \frac{\exp(f(\boldsymbol{x})[y])}{\sum_{j=1}^{m_2} \exp(f(\boldsymbol{x})[j])}.$$

Cross Entropy

Theorem 7

Fix $\delta \in (0,1)$, $c > 0$, the number of hidden layers $k \in [0, \infty)$, and widths $\boldsymbol{d} \in \mathbb{N}_+^{k+2}$ with $d_0 = m_1$ and $d_{k+1} = m_2$. With probability at least $1 - \delta$ over the choice of S, every $f_T = T_{k+1} \circ \sigma \circ T_k \circ \cdots \circ \sigma \circ T_1 \in \mathcal{S}_c^{k,\boldsymbol{d},\sigma}$ satisfies that

$$\mathcal{E}_{L_{ce}}(f_T) \leq \left(2(\|T_{k+1}\|_{1,\infty} + \frac{1}{m_2}) \max\left(1, (\gamma_\sigma c)^k\right) + \log m_2\right) *$$

$$\sqrt{\frac{\log(1/\delta) + 2\sum_{j=1}^{m_2} \log(m_2 \|T_{k+1}[j]\|_1 + 2)}{2n}} + \frac{2\sqrt{2}}{\sqrt{n}} \left(\|T_{k+1}\|_{1,1} + 1\right) *$$

$$\left(1 + \frac{\sqrt{m_2 - 1}}{1 + (m_2 - 1)\exp(-2(\|T_{k+1}\|_{1,\infty} + \frac{1}{m_2}) \max\left(1, (\gamma_\sigma c)^k\right))}\right) *$$

$$\left[\sqrt{(k+1)\log 16}\left(\sum_{\ell=0}^{k} (\gamma_\sigma c)^\ell + (\gamma_\sigma c)^k\right) + (\gamma_\sigma c)^k \sqrt{2\log(2m_1)}\right].$$

Cross Entropy

Remark 4

Assume $\log(m_1)$ *and* $\log(m_2)$ *are small and* $\exp(\|T_{k+1}\|_{1,\infty}(\gamma_\sigma c)^k) \leq a_0$, *then the above generalization bound is:*

- $\gamma_\sigma c < 1$: $O\left(\frac{\sqrt{k}a_0\sqrt{m_2}\|T_{k+1}\|_{1,\infty}}{\sqrt{n}}\frac{1-(\gamma_\sigma c)^{k+1}}{1-\gamma_\sigma c}\right)$
- $\gamma_\sigma c = 1$: $O\left(\frac{k^{3/2}a_0\sqrt{m_2}\|T_{k+1}\|_{1,\infty}}{\sqrt{n}}\right)$
- $\gamma_\sigma c > 1$: $O\left(\frac{\sqrt{k}a_0\sqrt{m_2}\|T_{k+1}\|_{1,\infty}}{\sqrt{n}}\frac{(\gamma_\sigma c)^{k+1}-1}{\gamma_\sigma c-1}\right)$

There is always a $\sqrt{m_2}$ *dependence on the number of classes* m_2.

Outline

Gradient Projection Descent Algorithm

We propose a gradient projection descent algorithm to solve the optimization problem

$$\min_f \left\{ \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) \,\middle|\, f \in \mathcal{S}_c^{k,\boldsymbol{d},\sigma} \right\}. \tag{2}$$

$$\begin{aligned} \min_f \quad & \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) \\ \text{s.t.} \quad & \|T_\ell\|_{1,\infty} \triangleq \left\|\tilde{\mathrm{V}}_\ell\right\|_{1,\infty} \le c, \quad \ell = 1, \cdots, k. \end{aligned} \tag{3}$$

- Gradient Projection Descent Algorithm is designed, which could be easily implemented as a variation of any gradient descent method.
- In each iteration of the original gradient descent method, we project its output to the sparse DNN function class.

Gradient Projection Descent Algorithm

The algorithm is shown as follows:

Algorithm 1 Gradient Projection Descent Algorithm

In each iteration:

Input: $\tilde{\mathbf{V}}^{(t)} = (\tilde{\mathbf{V}}_1^{(t)}, \cdots, \tilde{\mathbf{V}}_k^{(t)})$

for all $\ell = 1, \ldots, k$ **do**

$\tilde{\mathbf{V}}_\ell^{(t+1)} := \tilde{\mathbf{V}}_\ell^{(t)} - \gamma_t \nabla L(\tilde{\mathbf{V}}_\ell^{(t)})$,

where γ_t is the stepsize at iteration t

for all columns v in $\mathbf{V}_\ell^{(t+1)}$ **do**

if $\|v\|_1 > c$ **then**

$v = \text{proj}_{\|\cdot\|_1 \le c} v$ by **Algorithm 2**

end if

end for

end for

Output: $\tilde{\mathbf{V}}^{(t+1)} = (\tilde{\mathbf{V}}_1^{(t+1)}, \cdots, \tilde{\mathbf{V}}_k^{(t+1)})$

Gradient Projection Descent Algorithm

Projection algorithm used in **Algorithm 1** is shown in **Algorithm 2**. This algorithm aims to project any points to L_1 norm ball with radius c.

Algorithm 2 Projection to L_1 norm ball (Duchi et al. 2008)

Input: $v \in \mathbb{R}^s$, c

Sort $\text{abs}(v)$ into $\mu : \mu_1 \geqslant \mu_2 \geqslant \cdots \geqslant \mu_s$

Find $p^* = \max\{p \in [s] : \mu_p - \frac{1}{p}(\sum_{q=1}^{p} \mu_q - c) > 0\}$

Define $\theta = \frac{1}{p^*}(\sum_{q=1}^{p^*} \mu_q - c)$

Output: w s.t. $w_p = \text{sign}(v_p) \cdot \max\{\text{abs}(v_p) - \theta, 0\}$

Outline

Numerical Experiments for Validating Theorem

We validate our theorem using both simulated and real data experiments.

For each setting, we measure the training error, generalization error, test accuracy, and model sparsity in order to demonstrate c's influence on the generalization ability as well as the sparsity of the model.

- Generalization error: the difference between training error and test error.
- Test accuracy: the classification accuracy for testing data.
- Sparsity rate: the ratio of the number of zero parameter estimates to the size of weight matrices.

Outline

A Synthesis Regression Experiment

We evaluate our algorithm on a high-dimensional linear regression problem $y = \boldsymbol{x}^T\boldsymbol{\beta} + \varepsilon$, where the coefficient $\boldsymbol{\beta}$ is a sparse vector. We sample 500 random samples $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^{1000} \times \mathbb{R}, i = 1, \cdots, 500$ for training, and 1500 samples for testing from the distribution below.

1. Generate the coefficient $\boldsymbol{\beta}$ by $\beta_i \sim \mathrm{Unif}(0.15, 150)$ for $i = 1, \cdots, 100$, while setting the rest of $\boldsymbol{\beta}$ to be zero.
2. For $\forall i$, first independently sample an auxiliary variable $z_i \in \mathbb{R}^{1000}$ from $N(0, \boldsymbol{I})$. Then generate $\boldsymbol{x}_i$ by $x_{i1} = z_{i1}$, and $x_{ij} = z_{ij} + 0.2(z_{i,j+1} + z_{i,j-1})$ for $j = 2, \cdots, 1000$. Finally sample y_i from $N(\boldsymbol{x}_i^T\boldsymbol{\beta}, 1)$

We train the model with one fully connected layer with 300 output units using ReLU, and the loss function is mean square error.

A Synthesis Regression Experiment

The result is shown as follows.

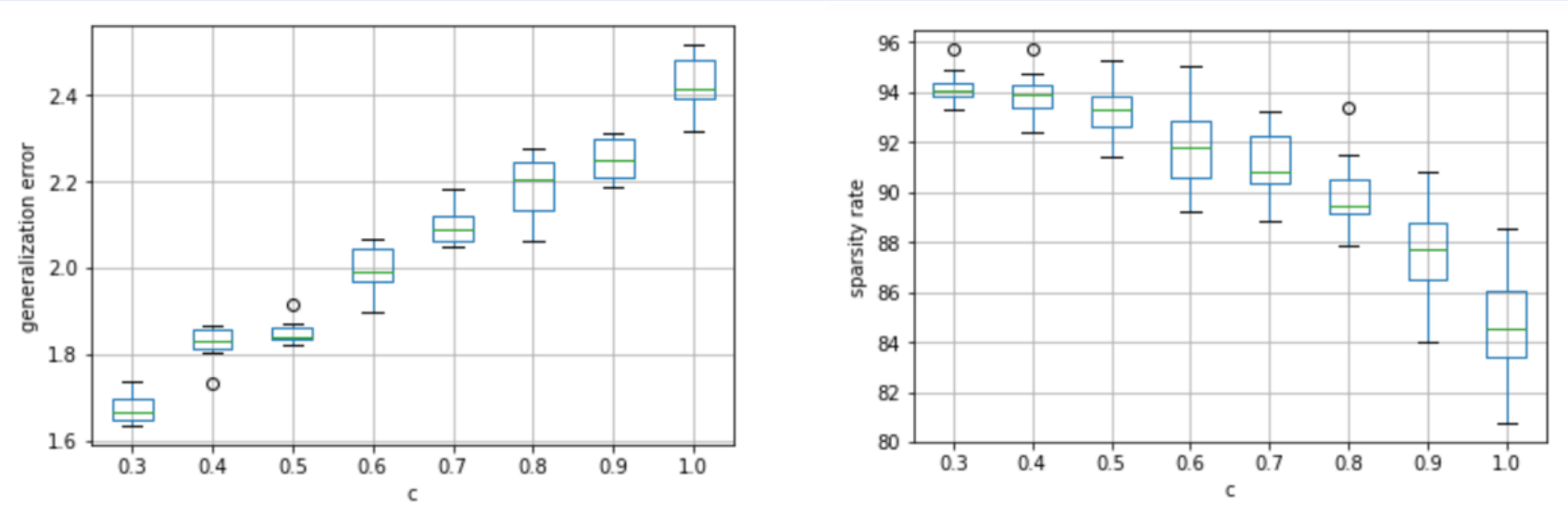


Figure: Boxplots of generalization error and sparsity rate with different c for the classification experiment.

As illustrated in the figure, the generalization error decreases as c decreases. Also, the network becomes sparser with a smaller c.

Outline

A Synthesis Classification Experiment

We sample 500 random samples $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^{500} \times \{0, 1\}, i = 1, \cdots, 500$ for training, and 1000 samples for testing from the distribution below.

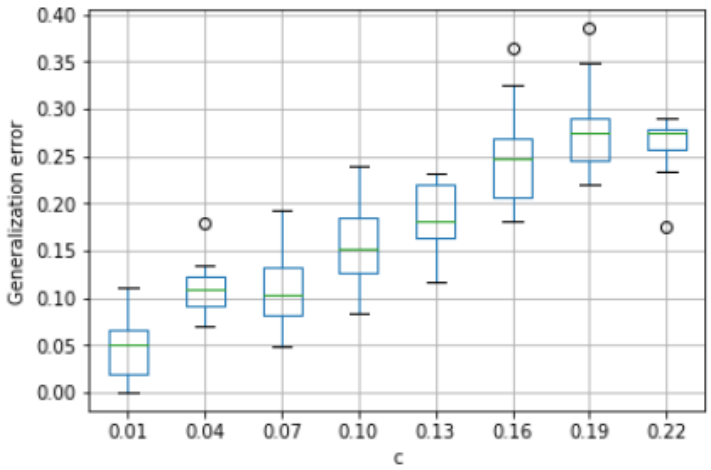
1. Generate $\zeta \sim N(0, 1)$.
2. For $\forall i$, independently sample x_{ij}: the jth element of $\boldsymbol{x}_i$, from $N(\frac{\zeta}{2}, \frac{1}{4})$ for $j = 1, \cdots, 500$, and

$$y_i = \begin{cases} 1, & e^{x_{i1}} + x_{i2}^2 + 5\sin(x_{i3}x_{i4}) - 3 > 0 \\ 0, & \text{otherwise} \end{cases}$$

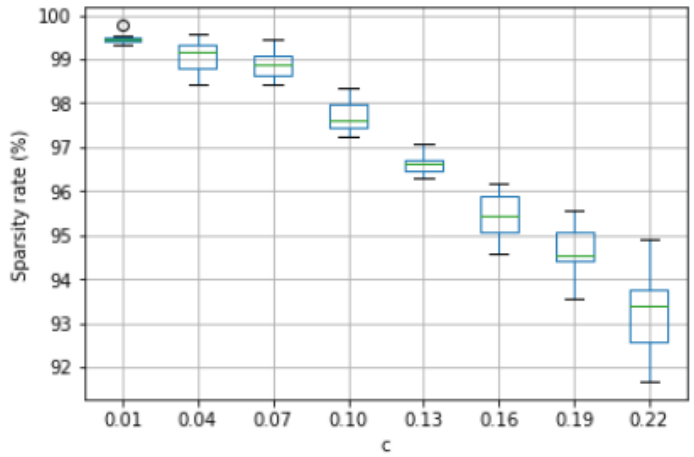
We use a 500-100-50-20-2 fully connected neural network with ReLU activation, and the loss function is cross entropy.

A Synthesis Classification Experiment

The result is shown as follows.



(a) Generalization error vs. c.

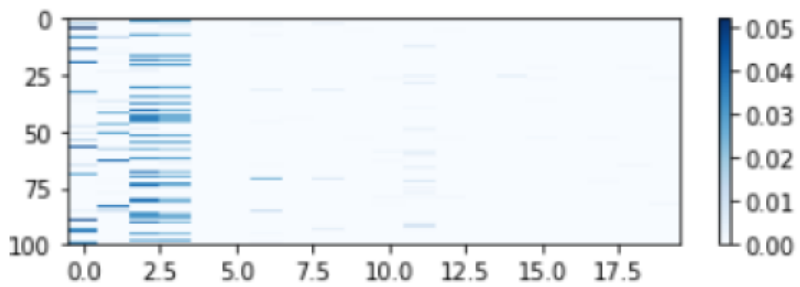


(b) Sparsity rate vs. c.

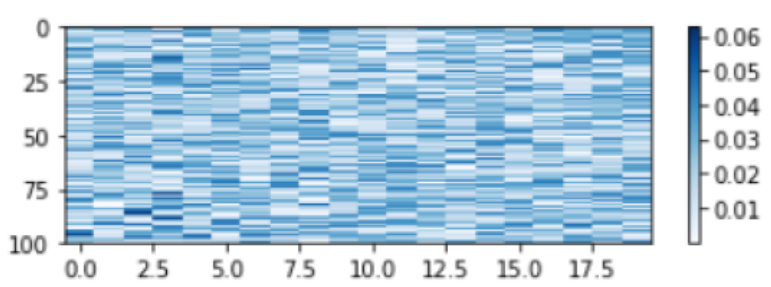
Figure: Boxplots of generalization error and sparsity rate with different c for the classification experiment.

As illustrated in the figure, the generalization error decreases as c decreases. Also, the network becomes sparser with a smaller c.

A Synthesis Classification Experiment



(a) $L_{1,\infty}$-weight normalization with $c = 0.07$.



(b) With no regularization.

Figure: Visualization of the first twenty columns of the resulting weight matrix representing the first hidden layer with/without $L_{1,\infty}$-weight normalization.

We also observe that the first 4 columns of the resulting weight matrix are dense, while the others are sparse. It is because that only the first four elements of the input are included in the true model.

A Synthesis Classification Experiment

We make experiments on the relationship between the depth of the network and accuracy and generazation error. Results show that deeper structure leads to larger generazation error. Also, with deeper structure, the model gets harder to converge with our algorithm, which may be drawbacks of our algorithm.

	100-20-2	100-50-20-2	100-100-50-20-2
∞	1.535/70.30	1.674/69.90	1.510/69.10
$c = 0.50$	0.461/83.14	0.478/84.78	0.542/82.42
$c = 0.16$	0.351/84.67	0.441/87.03	0.456/84.41
$c = 0.13$	0.322/85.35	0.376/87.23	0.431/84.89
$c = 0.10$	0.312/86.10	0.324/87.78	0.383/86.03
$c = 0.07$	0.245/88.42	0.260/88.34	0.274/87.98
$c = 0.04$	0.103/89.12	0.134/89.57	0.131/88.33
$c = 0.01$	0.072/87.74	0.068/88.48	-

Table: Generation error and accuracy of different network structure and sample size in classification problem, where '-' represents that the network cannot converge with projection stochastic gradient descent.

A Synthesis Classification Experiment

Experiments on the relationship between sample size and accuracy and generazation loss are done. For same c, larger sample size leads to better accuracy and smaller generazation loss.

	size=500	size=1000	size=1500	size=2000	size=2500
$c = \infty$	1.674/69.90	1.576/71.00	1.528/72.20	1.508/75.70	1.489/76.60
$c = 0.16$	0.441/87.03	0.343/88.10	0.258/89.30	0.208/91.50	0.199/92.74
$c = 0.13$	0.376/87.23	0.334/87.80	0.252/89.47	0.171/91.76	0.171/92.80
$c = 0.10$	0.324/87.78	0.280/87.60	0.223/90.30	0.176/90.70	0.169/90.80
$c = 0.07$	0.260/88.34	0.241/87.80	0.189/90.80	0.162/91.21	0.133/91.86
$c = 0.04$	0.134/89.57	0.112/89.94	0.102/91.31	0.084/91.72	0.073/92.24
$c = 0.01$	0.068/88.48	0.079/89.15	0.034/90.30	0.036/91.00	0.024/91.47

Table: Generation error and accuracy of different c and sample size in classification problem.

Outline

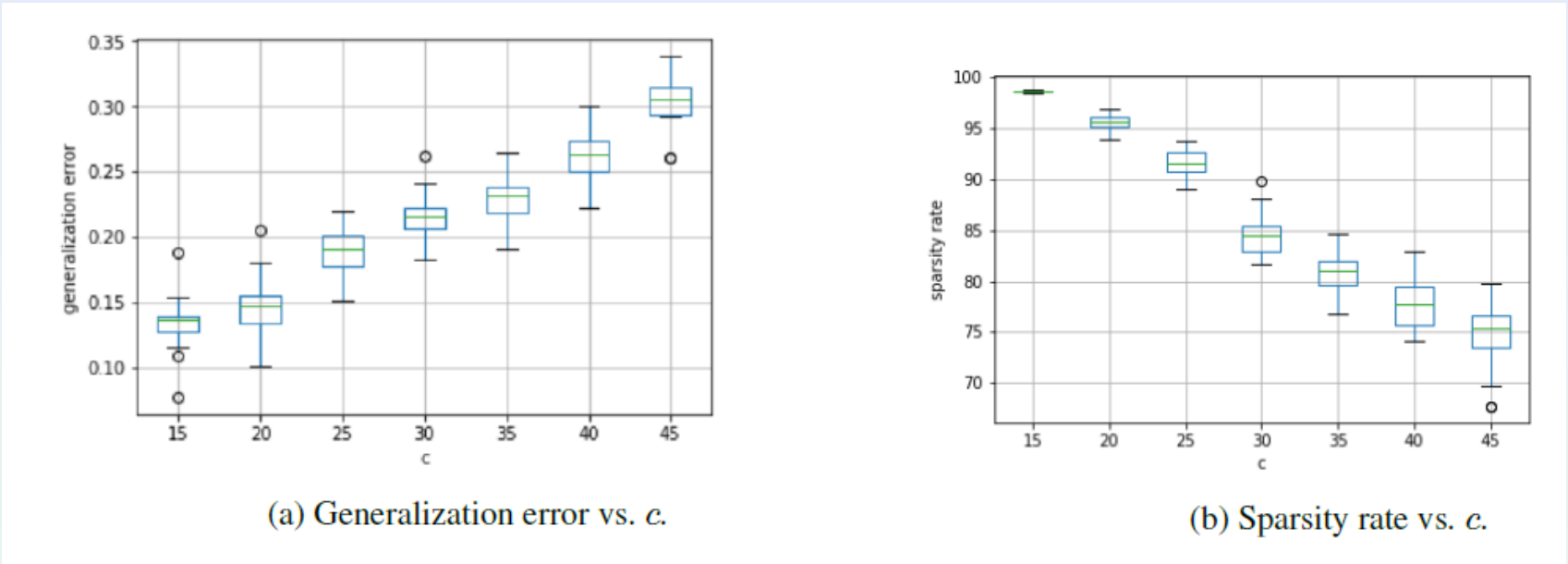
CIFAR-10

We extend our method to convolutional neural networks in the CIFAR-10 experiment, and give the setting as follows.

- The VGG-16 is used as a base network to train the model.
- The first two 3×3 convolutional layers are replaced by two 21×21 convolutional layers.
- A 20-fold cross-validation is done to test the model ability to predict new data.

CIFAR-10

The result is shown as follows.



(a) Generalization error vs. c.

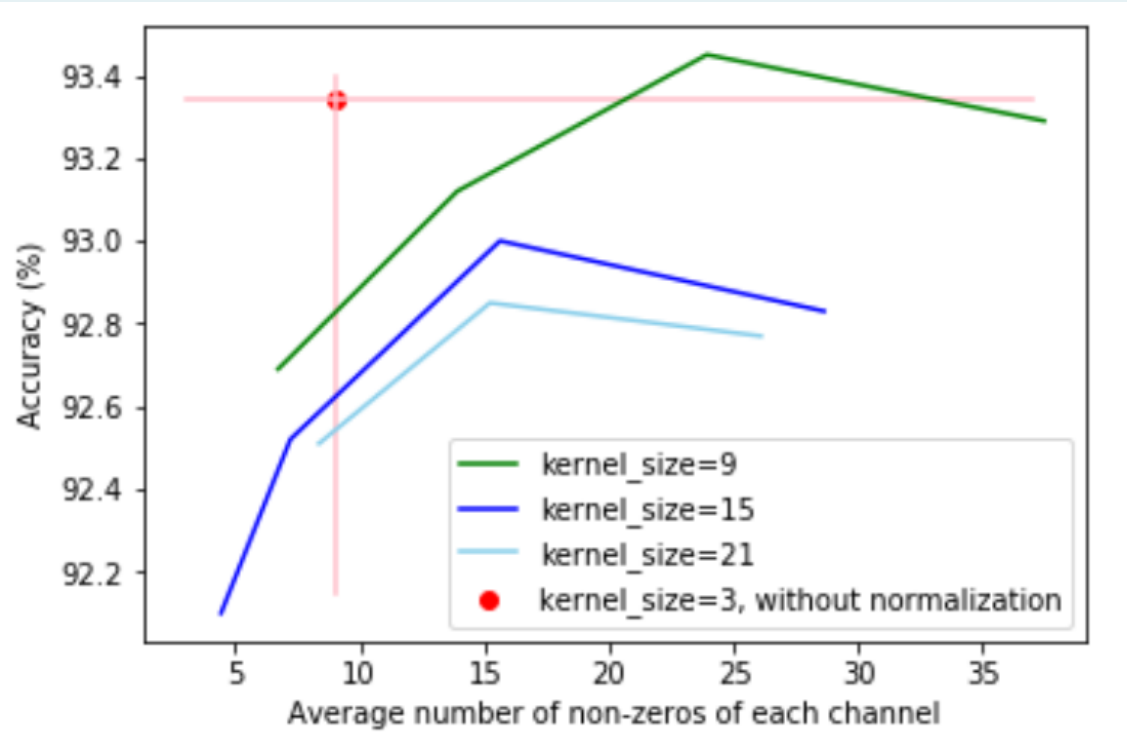
(b) Sparsity rate vs. c.

Figure: Boxplots of generalization error and sparsity rate with different c for CIFAR-10 experiment.

As shown in the figure, when c decreases, the network becomes sparser. Also, we can conclude that picking a larger c might result in poorer generalization.

CIFAR-10

For different kernel sizes (≥ 3), we can control the number of non-zero parameters of each channel by changing c. The following figure shows the relationship between test accuracy and the number of non-zero parameters of each channel. The result of larger kernel size performs as well as cases that kernel size is equal to 3. With kernel size equal to 9, we even get better test accuracy.



Outline

The k-fold Cross Validation

We use k-fold cross validation to choose the parameter. We divided the training dataset into 10 folds. Each time we choose one of the 10 folds as test fold and the rest as training fold. Finally, we choose c that has best average score.

Figure: The k-fold cross validation.

Cross Validation Results on Synthesis Experiments

We make 20-fold cross validation to choose c in regression and classification experiment.

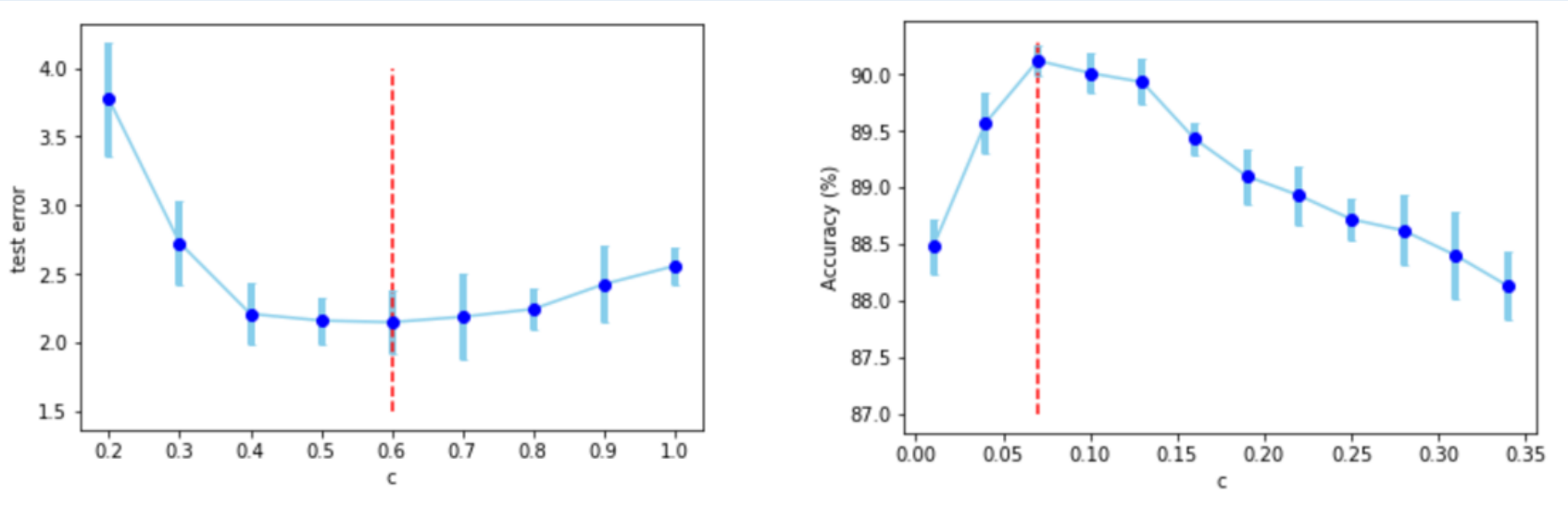


Figure: The 20-fold cross validation results on synthesis regression and classification experiments.

Outline

Comparison with Other Regularizers

We have also make comparison with other regularizers. As shown in the table, our approach is competitive with methods based on other common regularizations.

Regularizer	Cla. (Synthesized Data)	MNIST	CIFAR10
No	71.6	97.85	93.34
L_1, $\lambda = 0.1$	50.0	13.87	48.32
L_1, $\lambda = 0.01$	50.0	11.35	92.53
L_1, $\lambda = 0.001$	**81.6**	97.33	93.43
L_1, $\lambda = 0.0001$	73.2	97.99	93.64
L_2, $\lambda = 0.1$	70.8	80.30	80.41
L_2, $\lambda = 0.01$	73.4	92.30	90.38
L_2, $\lambda = 0.001$	73.2	94.43	91.41
L_2, $\lambda = 0.0001$	71.8	97.90	**93.72**
Dropout, $r_d = 0.5$	73.5	**98.11**	93.42
Our approach	**91.0**	**98.03**	**93.75**

Outline

Conclusion

The main contributions of our work are summarized as follows.

- Developed a systematic framework for sparse DNNs through $L_{1,\infty}$-weight normalization.
- Established the Rademacher complexity of the related sparse DNN space.
- Derived generalization error bounds for both regression and classification.
- Performed experiments to show that the proposed $L_{1,\infty}$ minimization process leads to neural network sparsification, which empirically validates our theoretical findings.

Future Work

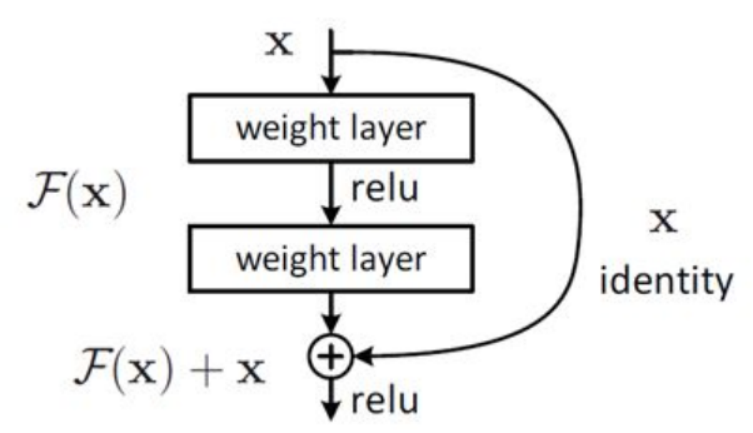


Figure: A structural unit of the widely used residual network.

- It is of theoretical significance and practical value to introduce $L_{1,\infty}$-weight normalization approach to residual networks.
- Our paper *"Theoretical Investigation of Generalization Bound for Residual Networks"* has been accepted for publication in the proceedings of IJCAI 2019.

References I

P. L. Bartlett.

For valid generalization the size of the weights is more important than the size of the network.

In *Advances in neural information processing systems*, pages 134–140, 1998.

P. L. Bartlett, D. J. Foster, and M. J. Telgarsky.

Spectrally-normalized margin bounds for neural networks.

In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

N. Golowich, A. Rakhlin, and O. Shamir.

Size-independent sample complexity of neural networks.

In *Conference On Learning Theory*, pages 297–299, 2018.

C. Louizos, M. Welling, and D. P. Kingma.

Learning sparse neural networks through L_0 regularization.

arXiv preprint arXiv:1712.01312, 2017.

References II

M. Mohri, A. Rostamizadeh, and A. Talwalkar.

Foundations of machine learning.

MIT press, 2012.

B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro.

Towards understanding the role of over-parametrization in generalization of neural networks.

arXiv preprint arXiv:1805.12076, 2018.

B. Neyshabur, R. Tomioka, and N. Srebro.

Norm-based capacity control in neural networks.

In *Conference on Learning Theory*, pages 1376–1401, 2015.

S. Srinivas, A. Subramanya, and R. V. Babu.

Training sparse neural networks.

arXiv preprint arXiv:1611.06694, 2016.

References III

S. Sun, W. Chen, L. Wang, X. Liu, and T.-Y. Liu.

On the depth of deep neural networks: A theoretical view.

In *AAAI*, pages 2066–2072, 2016.

Thanks for your attention!