Representation and regression problems in neural networks

Mean-field relaxation, generalization, and numerics

Kang Liu

joint work with Enrique Zuazua

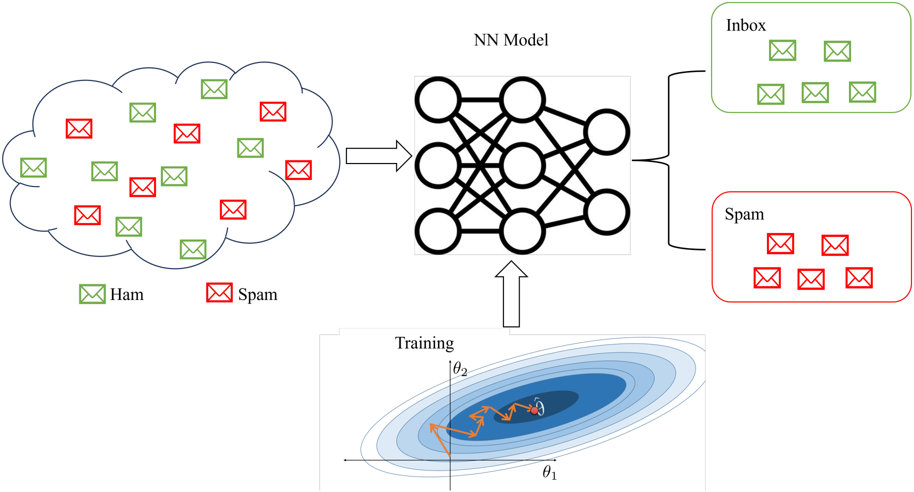
August 2024

Friedrich-Alexander-Universität
DYNAMICS, CONTROL, MACHINE LEARNING AND NUMERICS

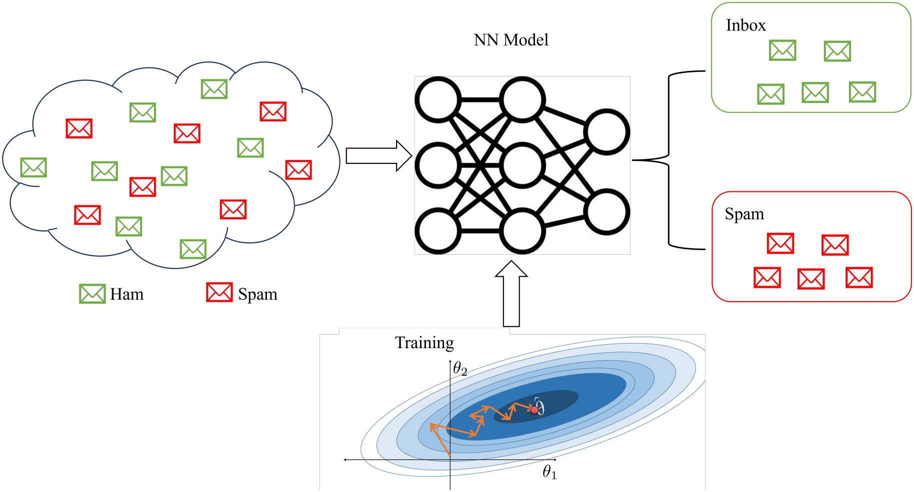
Table of Contents

Table of Contents

A diagram of classification task by NNs



A diagram of classification task by NNs



Key Points: **Data**, **Neural Network Model**, **Training**.

General framework of training problems

- **Data**: $\{(x_i, y_i) \in \mathbb{R}^{d+1}\}_{i=1}^{N}$.

General framework of training problems

- **Data**: $\{(x_i, y_i) \in \mathbb{R}^{d+1}\}_{i=1}^{N}$.
- **NN architecture**:

$$f : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}, \; (x, \Theta) \mapsto f(x, \Theta), \quad \text{where}$$

$$x : \text{ feature (input)}, \quad \Theta : \text{ parameter (control)}, \quad f(x, \Theta) : \text{ prediction (output)}.$$

General framework of training problems

- **Data**: $\{(x_i, y_i) \in \mathbb{R}^{d+1}\}_{i=1}^{N}$.
- **NN architecture**:

$$f : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}, \; (x, \Theta) \mapsto f(x, \Theta), \quad \text{where}$$

x : feature (input), Θ : parameter (control), $f(x, \Theta)$: prediction (output).

- **Three training scenarios**:

General framework of training problems

- **Data**: $\{(x_i, y_i) \in \mathbb{R}^{d+1}\}_{i=1}^{N}$.
- **NN architecture**:

$$f : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}, \; (x, \Theta) \mapsto f(x, \Theta), \quad \text{where}$$

x : feature (input), Θ : parameter (control), $f(x, \Theta)$: prediction (output).

- **Three training scenarios**:
  1. **Exact representation**:

$$f(x_i, \Theta) = y_i, \quad \text{for } i = 1, \ldots, N.$$

General framework of training problems

- **Data**: $\{(x_i, y_i) \in \mathbb{R}^{d+1}\}_{i=1}^{N}$.
- **NN architecture**:

$$f : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R},\ (x, \Theta) \mapsto f(x, \Theta), \quad \text{where}$$

x : feature (input), Θ : parameter (control), $f(x, \Theta)$: prediction (output).

- **Three training scenarios**:
  1. **Exact representation**:

$$f(x_i, \Theta) = y_i, \quad \text{for } i = 1, \ldots, N.$$

  2. **Approximate representation**:

$$\|f(x_i, \Theta) - y_i\| \le \epsilon, \quad \text{for } i = 1, \ldots, N.$$

General framework of training problems

- **Data**: $\{(x_i, y_i) \in \mathbb{R}^{d+1}\}_{i=1}^{N}$.
- **NN architecture**:

$$f : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R},\ (x, \Theta) \mapsto f(x, \Theta), \quad \text{where}$$

$$x: \text{ feature (input)}, \quad \Theta: \text{ parameter (control)}, \quad f(x, \Theta): \text{ prediction (output)}.$$

- **Three training scenarios**:
  1. **Exact representation**:
  $$f(x_i, \Theta) = y_i, \quad \text{for } i = 1, \ldots, N.$$
  2. **Approximate representation**:
  $$\|f(x_i, \Theta) - y_i\| \leq \epsilon, \quad \text{for } i = 1, \ldots, N.$$
  3. **Regression**:
  $$\inf_\Theta \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i, \Theta) - y_i).$$

General framework of training problems

- **Data**: $\{(x_i, y_i) \in \mathbb{R}^{d+1}\}_{i=1}^{N}$.
- **NN architecture**:

$$f : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R},\ (x, \Theta) \mapsto f(x, \Theta), \quad \text{where}$$

x : feature (input), Θ : parameter (control), $f(x, \Theta)$: prediction (output).

- **Three training scenarios**:
  1. **Exact representation**:

$$f(x_i, \Theta) = y_i, \quad \text{for } i = 1, \dots, N.$$

  2. **Approximate representation**:

$$\|f(x_i, \Theta) - y_i\| \leq \epsilon, \quad \text{for } i = 1, \dots, N.$$

  3. **Regression**:

$$\inf_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i, \Theta) - y_i).$$

General framework of training problems

- **Data**: $\{(x_i, y_i) \in \mathbb{R}^{d+1}\}_{i=1}^{N}$.
- **NN architecture**:

$$f : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R},\ (x, \Theta) \mapsto f(x, \Theta), \quad \text{where}$$

$$x: \text{ feature (input)}, \quad \Theta: \text{ parameter (control)}, \quad f(x, \Theta): \text{ prediction (output)}.$$

- **Three training scenarios**:
  1. **Exact representation**:

$$f(x_i, \Theta) = y_i, \quad \text{for } i = 1, \dots, N.$$

  2. **Approximate representation**:

$$\|f(x_i, \Theta) - y_i\| \leq \epsilon, \quad \text{for } i = 1, \dots, N.$$

  3. **Regression**:

$$\inf_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i, \Theta) - y_i).$$

Problems

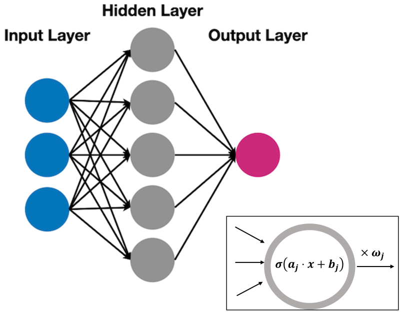
Existence, design of loss function, generalization property, numerical algorithms...

Shallow Neural Network

Shallow NNs with P neurons

$$f_{\text{shallow}}(x, \Theta) = \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x \rangle + b_j),$$

where $\Theta = (\omega_j, a_j, b_j)_{j=1}^{P}$, with $\omega_j \in \mathbb{R}$ and $(a_j, b_j) \in \mathbb{R}^{d+1}$.

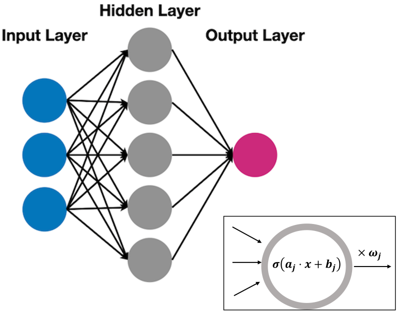


Shallow Neural Network

Shallow NNs with P neurons

$$f_{\text{shallow}}(x, \Theta) = \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x \rangle + b_j),$$

where $\Theta = (\omega_j, a_j, b_j)_{j=1}^{P}$, with $\omega_j \in \mathbb{R}$ and $(a_j, b_j) \in \mathbb{R}^{d+1}$.



Why shallow NNs

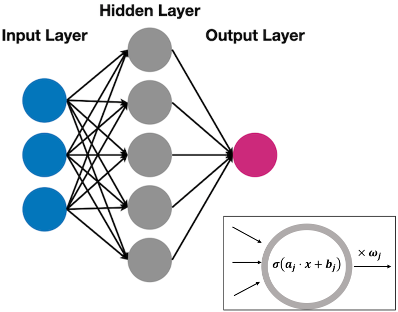
- Simple structure;

Shallow Neural Network

Shallow NNs with P neurons

$$f_{\text{shallow}}(x, \Theta) = \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x\rangle + b_j),$$

where $\Theta = (\omega_j, a_j, b_j)_{j=1}^{P}$, with $\omega_j \in \mathbb{R}$ and $(a_j, b_j) \in \mathbb{R}^{d+1}$.



Why shallow NNs

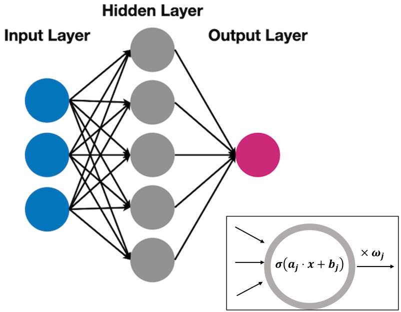
- Simple structure;
- Universal approximation property [Cybenko, 1989];

Shallow Neural Network

Shallow NNs with P neurons

$$f_{\text{shallow}}(x, \Theta) = \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x \rangle + b_j),$$

where $\Theta = (\omega_j, a_j, b_j)_{j=1}^{P}$, with $\omega_j \in \mathbb{R}$ and $(a_j, b_j) \in \mathbb{R}^{d+1}$.



Why shallow NNs

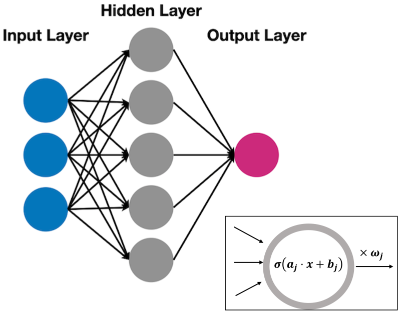
- Simple structure;
- Universal approximation property [Cybenko, 1989];
- Finite-sample representation property [Pinkus, 1999];

Shallow Neural Network

Shallow NNs with P neurons

$$f_{\text{shallow}}(x, \Theta) = \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x \rangle + b_j),$$

where $\Theta = (\omega_j, a_j, b_j)_{j=1}^{P}$, with $\omega_j \in \mathbb{R}$ and $(a_j, b_j) \in \mathbb{R}^{d+1}$.



Why shallow NNs

- Simple structure;
- Universal approximation property [Cybenko, 1989];
- Finite-sample representation property [Pinkus, 1999];
- "Convergence" of the SGD algorithm [Chizat-Bach, 2018].

Finite-sample representation property

Recall that

$$f_{\text{shallow}}(x, \Theta) = \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x\rangle + b_j).$$

Finite-sample representation property [Pinkus 1999]

Assume that $P \geq N$ and $m = 1$. If σ is non-polynomial, then for any distinct dataset $\{x_i, y_i\}_{i=1}^{N}$, there exists Θ such that

$$f_{\text{shallow}}(x_i, \Theta) = y_i, \quad \text{for } i = 1, \dots, N.$$

Finite-sample representation property

Recall that

$$f_{\text{shallow}}(x, \Theta) = \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x \rangle + b_j).$$

Finite-sample representation property [Pinkus 1999]

Assume that $P \geq N$ and $m = 1$. If σ is non-polynomial, then for any distinct dataset $\{x_i, y_i\}_{i=1}^{N}$, there exists Θ such that

$$f_{\text{shallow}}(x_i, \Theta) = y_i, \quad \text{for } i = 1, \ldots, N.$$

We extend in [L.-Zuazua, 2024] the previous result to the case where y_i is in high dimension and (a_j, b_j) are within a compact set. The proof is by induction and the application of the Hahn-Banach Theorem.

Design of loss function/regularization

- A well-known principle [1] in machine learning is the following:

"sparsity" mitigates "overfitting".

[1] Srivastava et al. "Dropout: A simple way to prevent Neural Networks from overfitting". In JMLR, 2014.

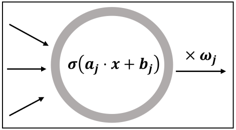
[2] Candes and Romberg. "Quantitative robust uncertainty principles and optimally sparse decompositions". In FOCM, 2006.

Design of loss function/regularization

- A well-known principle [1] in machine learning is the following:

"sparsity" mitigates "overfitting".

- In shallow NNs, the number of activated neurons is $\|\omega\|_{\ell^0}$.



[1] Srivastava et al. "Dropout: A simple way to prevent Neural Networks from overfitting". In JMLR, 2014.

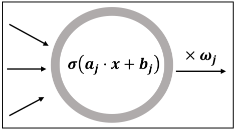
[2] Candes and Romberg. "Quantitative robust uncertainty principles and optimally sparse decompositions". In FOCM, 2006.

Design of loss function/regularization

- A well-known principle [1] in machine learning is the following:

"sparsity" mitigates "overfitting".

- In shallow NNs, the number of activated neurons is $\|\omega\|_{\ell^0}$.



- The function $\|\omega\|_{\ell^0}$ is non-convex. A practical replacement from compressed sensing [2]:

$$\|\omega\|_{\ell^0} \mapsto \|\omega\|_{\ell^1}.$$

[1] Srivastava et al. "Dropout: A simple way to prevent Neural Networks from overfitting". In JMLR, 2014.

[2] Candes and Romberg. "Quantitative robust uncertainty principles and optimally sparse decompositions". In FOCM, 2006.

Primal problems

Let Ω be a compact subset of $\mathbb{R}^{d+1}$. Note $\Theta = (\omega_j, a_j, b_j)_{j=1}^P$.

- The sparse **exact representation** problem:

$$\inf_{\Theta \in (\mathbb{R}\times\Omega)^P} \|\omega\|_{\ell^1}, \quad \text{s.t. } \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x_i\rangle + b_j) = y_i, \quad \text{for } i = 1, \dots, N. \qquad (\mathrm{P}_0)$$

Primal problems

Let Ω be a compact subset of $\mathbb{R}^{d+1}$. Note $\Theta = (\omega_j, a_j, b_j)_{j=1}^P$.

- The sparse **exact representation** problem:

$$\inf_{\Theta \in (\mathbb{R}\times\Omega)^P} \|\omega\|_{\ell^1}, \quad \text{s.t.} \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x_i\rangle + b_j) = y_i, \quad \text{for } i = 1, \dots, N. \qquad (\mathrm{P}_0)$$

- The sparse **approximate representation** problem:

$$\inf_{\Theta \in (\mathbb{R}\times\Omega)^P} \|\omega\|_{\ell^1}, \quad \text{s.t.} \left|\sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x_i\rangle + b_j) - y_i\right| \leq \epsilon, \quad \text{for } i = 1, \dots, N, \qquad (\mathrm{P}_\epsilon)$$

where $\epsilon > 0$ is a hyperparameter.

Primal problems

Let Ω be a compact subset of $\mathbb{R}^{d+1}$. Note $\Theta = (\omega_j, a_j, b_j)_{j=1}^P$.

- The sparse **exact representation** problem:

$$\inf_{\Theta \in (\mathbb{R}\times\Omega)^P} \|\omega\|_{\ell^1}, \quad \text{s.t.} \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x_i \rangle + b_j) = y_i, \quad \text{for } i = 1, \dots, N. \qquad (\mathrm{P}_0)$$

- The sparse **approximate representation** problem:

$$\inf_{\Theta \in (\mathbb{R}\times\Omega)^P} \|\omega\|_{\ell^1}, \quad \text{s.t.} \left| \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x_i \rangle + b_j) - y_i \right| \leq \epsilon, \quad \text{for } i = 1, \dots, N, \qquad (\mathrm{P}_\epsilon)$$

  where $\epsilon > 0$ is a hyperparameter.

- The sparse **regression** problem:

$$\inf_{\Theta \in (\mathbb{R}\times\Omega)^P} \|\omega\|_{\ell^1} + \frac{\lambda}{N} \sum_{i=1}^{N} \ell\left(\sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x_i \rangle + b_j) - y_i \right), \qquad (\mathrm{P}_\lambda^{\mathrm{reg}})$$

  where $\lambda > 0$ is a hyperparameter.

Problem

- How can we address these high-dimensional and non-convex optimization problems?

Table of Contents

Mean-field relaxation

Primal problems (P_0), (P_ϵ), and $(\mathrm{P}_\lambda^{\mathrm{reg}})$ are non-convex optimization problems, where the non-convexity is from the non-linearity of shallow NNs, e.g.,

$$\left\{ \Theta \,\middle|\, \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x_i \rangle + b_j) = y_i, \ \forall i = 1, \ldots, N \right\} \text{ is a non-convex set.}$$

Mean-field relaxation

Primal problems (P_0), (P_ϵ), and $(\mathrm{P}_\lambda^{\mathrm{reg}})$ are non-convex optimization problems, where the non-convexity is from the non-linearity of shallow NNs, e.g.,

$$\left\{ \Theta \,\middle|\, \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x_i \rangle + b_j) = y_i,\ \forall i = 1, \ldots, N \right\} \text{ is a non-convex set.}$$

The mean-field relaxation technique is commonly employed in shallow NNs, see [Mei-Montanari-Nguyen, 2018] and [Chizat-Bach, 2018].

Mean-field relaxation

Primal problems (P_0), (P_ϵ), and $(\mathrm{P}_\lambda^{\mathrm{reg}})$ are non-convex optimization problems, where the non-convexity is from the non-linearity of shallow NNs, e.g.,

$$\left\{\Theta \,\middle|\, \sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x_i\rangle + b_j) = y_i,\ \forall i = 1, \ldots, N\right\} \text{ is a non-convex set.}$$

The mean-field relaxation technique is commonly employed in shallow NNs, see [Mei-Montanari-Nguyen, 2018] and [Chizat-Bach, 2018].

Shallow NN

The original shallow NN writes:

$$\sum_{j=1}^{P} \omega_j \sigma(\langle a_j, x\rangle + b_j),$$

where $(\omega_j, a_j, b_j) \in \mathbb{R} \times \Omega$ for all j.

Cost function: $\|\omega\|_{\ell^1}$.

Mean-field shallow NN

The mean-field shallow NN writes:

$$\int_\Omega \sigma(\langle a, x\rangle + b)d\mu(a, b),$$

where $\mu \in \mathcal{M}(\Omega)$. The outcome is linear with respect to μ.

Cost function: $\|\mu\|_{\mathrm{TV}}$.

Relaxed problems

Let $Y = (y_1, \ldots, y_N)$. Define the following linear mapping:

$$\phi\,\mu := (\phi_i\,\mu)_{i=1}^N = \left(\int_\Omega \sigma(\langle a, x_i\rangle + b)d\mu(a,b)\right)_{i=1}^N$$

Relaxed problems

Let $Y = (y_1, \ldots, y_N)$. Define the following linear mapping:

$$\phi\,\mu := (\phi_i\,\mu)_{i=1}^N = \left(\int_\Omega \sigma(\langle a, x_i\rangle + b)d\mu(a,b)\right)_{i=1}^N$$

Convex relaxations:

- The relaxation of (P_0):

$$\inf_{\mu\in\mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}}, \quad \text{s.t. } \phi\,\mu = Y. \qquad (\mathrm{PR}_0)$$

Relaxed problems

Let $Y = (y_1, \dots, y_N)$. Define the following linear mapping:

$$\phi\,\mu := (\phi_i\,\mu)_{i=1}^N = \left(\int_\Omega \sigma(\langle a, x_i\rangle + b)d\mu(a,b)\right)_{i=1}^N$$

Convex relaxations:

- The relaxation of (P_0):

$$\inf_{\mu\in\mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}}, \quad \text{s.t. } \phi\,\mu = Y. \qquad (\mathrm{PR}_0)$$

- The relaxation of (P_ϵ):

$$\inf_{\mu\in\mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}}, \quad \text{s.t. } \|\phi\,\mu - Y\|_{\ell^\infty} \le \epsilon. \qquad (\mathrm{PR}_\epsilon)$$

Relaxed problems

Let $Y = (y_1, \ldots, y_N)$. Define the following linear mapping:

$$\phi\,\mu := (\phi_i\,\mu)_{i=1}^N = \left(\int_\Omega \sigma(\langle a, x_i\rangle + b)d\mu(a,b)\right)_{i=1}^N$$

Convex relaxations:

- The relaxation of (P_0):

$$\inf_{\mu\in\mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}}, \quad \text{s.t. } \phi\,\mu = Y. \qquad (\mathrm{PR}_0)$$

- The relaxation of (P_ϵ):

$$\inf_{\mu\in\mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}}, \quad \text{s.t. } \|\phi\,\mu - Y\|_{\ell^\infty} \leq \epsilon. \qquad (\mathrm{PR}_\epsilon)$$

- The relaxation of $(\mathrm{P}_\lambda^{\mathrm{reg}})$:

$$\inf_{\mu\in\mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}} + \frac{\lambda}{N}\sum_{i=1}^N \ell\left(\phi_i\,\mu - y_i\right). \qquad (\mathrm{PR}_\lambda^{\mathrm{reg}})$$

Free of relaxation gap

Theorem (L.-Zuazua,2024)

Under mild assumptions [1] *on* σ *and* Ω*, if* $P \geq N$*, then*

$$\mathrm{val}(\mathrm{P}_0) = \mathrm{val}(\mathrm{PR}_0); \quad \mathrm{val}(\mathrm{P}_\epsilon) = \mathrm{val}(\mathrm{PR}_\epsilon); \quad \mathrm{val}(\mathrm{P}_\lambda^{\mathrm{reg}}) = \mathrm{val}(\mathrm{PR}_\lambda^{\mathrm{reg}}).$$

Moreover, the extreme points of the solution sets of relaxed problems have the following form:

$$\mu^* = \sum_{j=1}^{N} \omega_j^* \delta_{(a_j^*, b_j^*)}.$$

[1] An example of (σ, Ω): σ is the ReLU function and Ω is the unit ball.

[2] Similar results for particular scenarios of exact representation and regression in ML obtained by representer theorems are studied in [Unser, 2019] and [Dios-Bruna, 2020].

Free of relaxation gap

Theorem (L.-Zuazua,2024)

Under mild assumptions [1] *on σ and Ω, if $P \geq N$, then*

$$\mathrm{val}(\mathrm{P}_0) = \mathrm{val}(\mathrm{PR}_0); \quad \mathrm{val}(\mathrm{P}_\epsilon) = \mathrm{val}(\mathrm{PR}_\epsilon); \quad \mathrm{val}(\mathrm{P}_\lambda^{\mathrm{reg}}) = \mathrm{val}(\mathrm{PR}_\lambda^{\mathrm{reg}}).$$

Moreover, the extreme points of the solution sets of relaxed problems have the following form:

$$\mu^* = \sum_{j=1}^{N} \omega_j^* \delta_{(a_j^*, b_j^*)}.$$

Main techniques in the proof:

- Existence of solutions: finite-sample representation property.
- "Representer Theorem" [2] from [Fisher-Jerome, 1975].

[1] An example of (σ, Ω): σ is the ReLU function and Ω is the unit ball.

[2] Similar results for particular scenarios of exact representation and regression in ML obtained by representer theorems are studied in [Unser, 2019] and [Dios-Bruna, 2020].

Problems

- How should the hyperparameters ϵ and λ be chosen in these problems? (Generalization)

- How can the relaxed problems be solved, and how can solutions of the primal problems be found? (Numerical algorithms)

Table of Contents

A generalization bound

- **Training**/**Testing** dataset: $\{(x_i, y_i)\}_{i=1}^{N}$ / $\{(x_i', y_i')\}_{i=1}^{N'}$.

A generalization bound

- **Training/Testing** dataset: $\{(x_i, y_i)\}_{i=1}^{N}$ / $\{(x_i', y_i')\}_{i=1}^{N'}$.
- **Predictions** on testing set by the shallow NN with parameter Θ:

$$\{(x_i', f_{\text{shallow}}(x_i', \Theta))\}_{i=1}^{N'}$$

A generalization bound

- **Training/Testing** dataset: $\{(x_i, y_i)\}_{i=1}^{N}$ / $\{(x_i', y_i')\}_{i=1}^{N'}$.
- **Predictions** on testing set by the shallow NN with parameter Θ:

$$\{(x_i', f_{\text{shallow}}(x_i', \Theta))\}_{i=1}^{N'}$$

- **Empirical measures**:

$$m_{\text{train}} = \frac{1}{N}\sum_{i=1}^{N}\delta_{(x_i, y_i)}, \quad m_{\text{test}} = \frac{1}{N'}\sum_{i=1}^{N'}\delta_{(x_i', y_i')}, \quad m_{\text{pred}}(\Theta) = \frac{1}{N'}\sum_{i=1}^{N'}\delta_{(x_i', f_{\text{shallow}}(x_i', \Theta))}.$$

A generalization bound

- **Training/Testing** dataset: $\{(x_i, y_i)\}_{i=1}^{N}$ / $\{(x_i', y_i')\}_{i=1}^{N'}$.
- **Predictions** on testing set by the shallow NN with parameter Θ:

$$\{(x_i', f_{\text{shallow}}(x_i', \Theta))\}_{i=1}^{N'}$$

- **Empirical measures**:

$$m_{\text{train}} = \frac{1}{N} \sum_{i=1}^{N} \delta_{(x_i, y_i)}, \quad m_{\text{test}} = \frac{1}{N'} \sum_{i=1}^{N'} \delta_{(x_i', y_i')}, \quad m_{\text{pred}}(\Theta) = \frac{1}{N'} \sum_{i=1}^{N'} \delta_{(x_i', f_{\text{shallow}}(x_i', \Theta))}.$$

A generalization bound

- **Training/Testing** dataset: $\{(x_i, y_i)\}_{i=1}^{N}$ / $\{(x_i', y_i')\}_{i=1}^{N'}$.
- **Predictions** on testing set by the shallow NN with parameter Θ:

$$\{(x_i', f_{\text{shallow}}(x_i', \Theta))\}_{i=1}^{N'}$$

- **Empirical measures**:

$$m_{\text{train}} = \frac{1}{N}\sum_{i=1}^{N}\delta_{(x_i, y_i)}, \quad m_{\text{test}} = \frac{1}{N'}\sum_{i=1}^{N'}\delta_{(x_i', y_i')}, \quad m_{\text{pred}}(\Theta) = \frac{1}{N'}\sum_{i=1}^{N'}\delta_{(x_i', f_{\text{shallow}}(x_i', \Theta))}.$$

Theorem (L.-Zuazua,2024)

Let $W_1(\cdot, \cdot)$ denote the Wassernstein-1 distance. If σ is 1-Lipschitz, then for any Θ,

$$W_1(m_{\text{test}}, m_{\text{pred}}(\Theta)) \leq \underbrace{2W_1(m_{\text{train}}, m_{\text{test}})}_{\text{Bias from datasets}} + r(\Theta), \quad \text{where}$$

$$r(\Theta) = \underbrace{\frac{1}{N}\sum_{i=1}^{N}|f_{\text{shallow}}(x_i, \Theta) - y_i|}_{\text{Bias from training}} + \underbrace{W_1(m_{\text{train}}, m_{\text{test}})\sum_{j=1}^{P}|\omega_j|\|a_j\|}_{\text{"Variance"}}.$$

Generalization bounds by optimal solutions

Fix the following:

- σ: ReLU;
- Ω: $B^{d+1}(0,1)$;
- $\ell(\cdot) = |\cdot|$.

Recall that

$$W_1(m_{\text{test}}, m_{\text{pred}}(\Theta)) \leq \underbrace{2W_1(m_{\text{train}}, m_{\text{test}})}_{\text{Bias from datasets}} + r(\Theta).$$

Generalization bounds by optimal solutions

Fix the following:

- σ: ReLU;
- Ω: $B^{d+1}(0,1)$;
- $\ell(\cdot) = |\cdot|$.

Recall that

$$W_1(m_{\text{test}}, m_{\text{pred}}(\Theta)) \le \underbrace{2W_1(m_{\text{train}}, m_{\text{test}})}_{\text{Bias from datasets}} + r(\Theta).$$

Proposition

Let $P \ge N$. For any $\epsilon \ge 0$ and $\lambda > 0$, let Θ_ϵ and $\Theta_\lambda^{\text{reg}}$ be the solutions of (P_ϵ) and $(\mathrm{P}_\lambda^{\text{reg}})$, respectively. Then,

$$r(\Theta_\epsilon) \le \mathcal{U}(\epsilon) := \epsilon + C\,\mathrm{val}(\mathrm{P}_\epsilon);$$

$$r(\Theta_\lambda^{\text{reg}}) \le \mathcal{L}(\lambda) := \max\{\lambda^{-1}, C\}\,\mathrm{val}(\mathrm{PR}_\lambda^{\text{reg}}),$$

where $C = W_1(m_{\text{train}}, m_{\text{test}})$.

Optimal hyperparameters

Recall that $C = W_1(m_{\text{train}}, m_{\text{test}})$.

- Optimal value of λ: $\lambda^* = C^{-1}$.

Optimal hyperparameters

Recall that $C = W_1(m_{\text{train}}, m_{\text{test}})$.

- Optimal value of λ: $\lambda^* = C^{-1}$.
- Optimal value of ϵ:

Optimal hyperparameters

Recall that $C = W_1(m_{\text{train}}, m_{\text{test}})$.

- Optimal value of λ: $\lambda^* = C^{-1}$.
- Optimal value of ϵ:
    1. if $C < c_0^{-1}$, then $\epsilon^* = 0$;

Optimal hyperparameters

Recall that $C = W_1(m_{\text{train}}, m_{\text{test}})$.

- Optimal value of λ: $\lambda^* = C^{-1}$.
- Optimal value of ϵ:
  1. if $C < c_0^{-1}$, then $\epsilon^* = 0$;
  2. if $C \geq c_0^{-1}$, then ϵ^* satisfies the first-order optimality condition $C^{-1} \in [c_{\epsilon^*}, C_{\epsilon^*}]$.

Optimal hyperparameters

Recall that $C = W_1(m_{\text{train}}, m_{\text{test}})$.

- Optimal value of λ: $\lambda^* = C^{-1}$.
- Optimal value of ϵ:
  1. if $C < c_0^{-1}$, then $\epsilon^* = 0$;
  2. if $C \geq c_0^{-1}$, then ϵ^* satisfies the first-order optimality condition $C^{-1} \in [c_{\epsilon^*}, C_{\epsilon^*}]$.

Optimal hyperparameters

Recall that $C = W_1(m_{\text{train}}, m_{\text{test}})$.

- Optimal value of λ: $\lambda^* = C^{-1}$.
- Optimal value of ϵ:
  1. if $C < c_0^{-1}$, then $\epsilon^* = 0$;
  2. if $C \geq c_0^{-1}$, then ϵ^* satisfies the first-order optimality condition $C^{-1} \in [c_{\epsilon^*}, C_{\epsilon^*}]$.

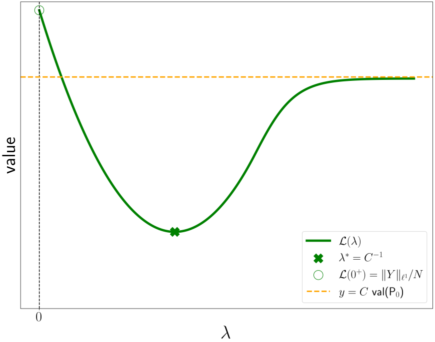
  Here, (c_ϵ, C_ϵ) is related to the solutions of the dual problem of (PR_ϵ).

Optimal hyperparameters

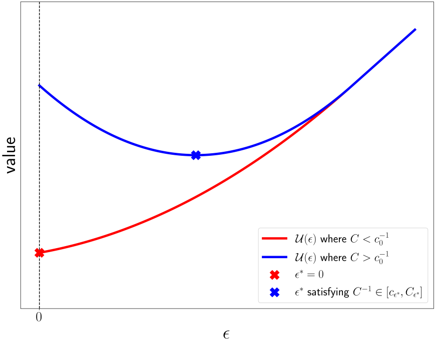
Recall that $C = W_1(m_{\text{train}}, m_{\text{test}})$.

- Optimal value of λ: $\lambda^* = C^{-1}$.
- Optimal value of ϵ:
  1. if $C < c_0^{-1}$, then $\epsilon^* = 0$;
  2. if $C \geq c_0^{-1}$, then ϵ^* satisfies the first-order optimality condition $C^{-1} \in [c_{\epsilon^*}, C_{\epsilon^*}]$.

  Here, (c_ϵ, C_ϵ) is related to the solutions of the dual problem of (PR_ϵ).



(a) Qualitative curve of $\mathcal{L}(\lambda)$.



(b) Two scenarios of $\mathcal{U}(\epsilon)$.

Table of Contents

Guideline for numerical algorithms

Relaxed problems are convex, but in an infinite-dimensional space.

$$\inf_{\mu \in \mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}}, \quad \text{s.t. } \|\phi\,\mu - Y\|_{\ell^\infty} \leq \epsilon. \tag{PR$_\epsilon$}$$

$$\inf_{\mu \in \mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}} + \frac{\lambda}{N} \sum_{i=1}^{N} |\phi_i\,\mu - y_i|. \tag{PR$_\lambda^{\mathrm{reg}}$}$$

Guideline for numerical algorithms

Relaxed problems are convex, but in an infinite-dimensional space.

$$\inf_{\mu\in\mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}}, \quad \text{s.t. } \|\phi\,\mu - Y\|_{\ell^\infty} \leq \epsilon. \tag{PR$_\epsilon$}$$

$$\inf_{\mu\in\mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}} + \frac{\lambda}{N}\sum_{i=1}^{N} |\phi_i\,\mu - y_i|. \tag{PR$_\lambda^{\mathrm{reg}}$}$$

A general approach: **Discretization**, then **Optimization**.

Guideline for numerical algorithms

Relaxed problems are convex, but in an infinite-dimensional space.

$$\inf_{\mu\in\mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}}, \quad \text{s.t. } \|\phi\,\mu - Y\|_{\ell^\infty} \leq \epsilon. \tag{PR_ϵ}$$

$$\inf_{\mu\in\mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}} + \frac{\lambda}{N}\sum_{i=1}^{N} |\phi_i\,\mu - y_i|. \tag{$\mathrm{PR}_\lambda^{\mathrm{reg}}$}$$

A general approach: **Discretization**, then **Optimization**.

Two numerical scenarios

1. When $\dim(\Omega) = d+1$ is small, discretize Ω by a mesh, then optimize by the simplex method.

Guideline for numerical algorithms

Relaxed problems are convex, but in an infinite-dimensional space.

$$\inf_{\mu\in\mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}}, \quad \text{s.t. } \|\phi\,\mu - Y\|_{\ell^\infty} \le \epsilon. \tag{PR$_\epsilon$}$$

$$\inf_{\mu\in\mathcal{M}(\Omega)} \|\mu\|_{\mathrm{TV}} + \frac{\lambda}{N}\sum_{i=1}^{N} |\phi_i\,\mu - y_i|. \tag{PR$_\lambda^{\mathrm{reg}}$}$$

A general approach: **Discretization**, then **Optimization**.

Two numerical scenarios

1. When $\dim(\Omega) = d+1$ is small, discretize Ω by a mesh, then optimize by the simplex method.
2. When $\dim(\Omega) = d+1$ is great, discretize (PR$_\lambda^{\mathrm{reg}}$) by an overparameterized version (problem (P$_\lambda^{\mathrm{reg}}$) with a large P), then optimize by the SGD algorithm.

Low-dimensional scenario

- Discretization of the domain:

$$\Omega \to \Omega_h = \{(a_j, b_j)\}_{j=1}^{M} .$$

Low-dimensional scenario

- Discretization of the domain:

$$\Omega \to \Omega_h = \{(a_j, b_j)\}_{j=1}^{M}.$$

- Discretized problems:

$$\inf_{\omega \in \mathbb{R}^M} \|\omega\|_{\ell^1}, \quad \text{s.t.} \quad \|A\omega - Y\|_{\ell^\infty} \leq \epsilon, \tag{PD$_\epsilon$}$$

$$\inf_{\omega \in \mathbb{R}^M} \|\omega\|_{\ell^1} + \frac{\lambda}{N}\|A\omega - Y\|_{\ell^1}, \tag{PD$_\lambda^{\text{reg}}$}$$

where $A \in \mathbb{R}^{N \times M}$ with $A_{ij} = \sigma(\langle a_j, x_i \rangle + b_j)$.

Low-dimensional scenario

- Discretization of the domain:

$$\Omega \to \Omega_h = \{(a_j, b_j)\}_{j=1}^{M}.$$

- Discretized problems:

$$\inf_{\omega \in \mathbb{R}^M} \|\omega\|_{\ell^1}, \quad \text{s.t.} \quad \|A\omega - Y\|_{\ell^\infty} \leq \epsilon, \qquad (\mathrm{PD}_\epsilon)$$

$$\inf_{\omega \in \mathbb{R}^M} \|\omega\|_{\ell^1} + \frac{\lambda}{N}\|A\omega - Y\|_{\ell^1}, \qquad (\mathrm{PD}_\lambda^{\mathrm{reg}})$$

where $A \in \mathbb{R}^{N \times M}$ with $A_{ij} = \sigma(\langle a_j, x_i \rangle + b_j)$.

- Error estimates:

$$|\mathrm{val}(\mathrm{PD}_\epsilon) - \mathrm{val}(\mathrm{PR}_\epsilon)|, \ |\mathrm{val}(\mathrm{PD}_\lambda^{\mathrm{reg}}) - \mathrm{val}(\mathrm{PR}_\lambda^{\mathrm{reg}})| = \mathcal{O}(d_{\mathrm{Hausdorff}}(\Omega, \Omega_h)).$$

Low-dimensional scenario

- Discretization of the domain:

$$\Omega \to \Omega_h = \{(a_j, b_j)\}_{j=1}^{M}.$$

- Discretized problems:

$$\inf_{\omega \in \mathbb{R}^M} \|\omega\|_{\ell^1}, \quad \text{s.t.} \quad \|A\omega - Y\|_{\ell^\infty} \le \epsilon, \qquad (\text{PD}_\epsilon)$$

$$\inf_{\omega \in \mathbb{R}^M} \|\omega\|_{\ell^1} + \frac{\lambda}{N}\|A\omega - Y\|_{\ell^1}, \qquad (\text{PD}_\lambda^{\text{reg}})$$

where $A \in \mathbb{R}^{N\times M}$ with $A_{ij} = \sigma(\langle a_j, x_i\rangle + b_j)$.

- Error estimates:

$$|\text{val}(\text{PD}_\epsilon) - \text{val}(\text{PR}_\epsilon)|,\ |\text{val}(\text{PD}_\lambda^{\text{reg}}) - \text{val}(\text{PR}_\lambda^{\text{reg}})| = \mathcal{O}(d_{\text{Hausdorff}}(\Omega, \Omega_h)).$$

- Equivalent to linear programming problems, solvable using the simplex method.

Low-dimensional scenario

- Discretization of the domain:

$$\Omega \to \Omega_h = \{(a_j, b_j)\}_{j=1}^{M}.$$

- Discretized problems:

$$\inf_{\omega\in\mathbb{R}^M} \|\omega\|_{\ell^1}, \quad \text{s.t. } \|A\omega - Y\|_{\ell^\infty} \leq \epsilon, \tag{PD_ϵ}$$

$$\inf_{\omega\in\mathbb{R}^M} \|\omega\|_{\ell^1} + \frac{\lambda}{N}\|A\omega - Y\|_{\ell^1}, \tag{$\mathrm{PD}_\lambda^{\mathrm{reg}}$}$$

where $A \in \mathbb{R}^{N\times M}$ with $A_{ij} = \sigma(\langle a_j, x_i\rangle + b_j)$.

- Error estimates:

$$|\mathrm{val}(\mathrm{PD}_\epsilon) - \mathrm{val}(\mathrm{PR}_\epsilon)|,\ |\mathrm{val}(\mathrm{PD}_\lambda^{\mathrm{reg}}) - \mathrm{val}(\mathrm{PR}_\lambda^{\mathrm{reg}})| = \mathcal{O}(d_{\mathrm{Hausdorff}}(\Omega, \Omega_h)).$$

- Equivalent to linear programming problems, solvable using the simplex method.
  - **Advantage**: Terminates at an extreme point of the solution set, which corresponds to a solution of the primal problems.

Low-dimensional scenario

- Discretization of the domain:

$$\Omega \to \Omega_h = \{(a_j, b_j)\}_{j=1}^{M}.$$

- Discretized problems:

$$\inf_{\omega \in \mathbb{R}^M} \|\omega\|_{\ell^1}, \quad \text{s.t.} \quad \|A\omega - Y\|_{\ell^\infty} \le \epsilon, \qquad (\mathrm{PD}_\epsilon)$$

$$\inf_{\omega \in \mathbb{R}^M} \|\omega\|_{\ell^1} + \frac{\lambda}{N}\|A\omega - Y\|_{\ell^1}, \qquad (\mathrm{PD}_\lambda^{\mathrm{reg}})$$

where $A \in \mathbb{R}^{N \times M}$ with $A_{ij} = \sigma(\langle a_j, x_i \rangle + b_j)$.

- Error estimates:

$$|\mathrm{val}(\mathrm{PD}_\epsilon) - \mathrm{val}(\mathrm{PR}_\epsilon)|, \ |\mathrm{val}(\mathrm{PD}_\lambda^{\mathrm{reg}}) - \mathrm{val}(\mathrm{PR}_\lambda^{\mathrm{reg}})| = \mathcal{O}(d_{\mathrm{Hausdorff}}(\Omega, \Omega_h)).$$

- Equivalent to linear programming problems, solvable using the simplex method.
  - **Advantage**: Terminates at an extreme point of the solution set, which corresponds to a solution of the primal problems.
  - **Limitation**: Suffer from the curse of dimensionality.

High-dimensional scenario

- Apply the SGD algorithm to the following overparameterized problem:

$$\inf_{\Theta\in(\mathbb{R}\times\Omega)^{\bar{P}}} \|\omega\|_{\ell^1} + \frac{\lambda}{N}\sum_{i=1}^{N} \ell\left(\sum_{j=1}^{\bar{P}} \omega_j \sigma(\langle a_j, x_i\rangle + b_j) - y_i\right),$$

where $\bar{P}$ is large [1].

[1] The convergence properties of SGD for the training of overparameterized NNs have been extensively studied recently, including [Chitzat-Bach, 2018], [Zhu-Li-Song, 2019], [Bach, 2024, Chp.12], etc.

High-dimensional scenario

- Apply the SGD algorithm to the following overparameterized problem:

$$\inf_{\Theta\in(\mathbb{R}\times\Omega)^{\bar{P}}} \|\omega\|_{\ell^1} + \frac{\lambda}{N}\sum_{i=1}^{N} \ell\left(\sum_{j=1}^{\bar{P}} \omega_j \sigma(\langle a_j, x_i\rangle + b_j) - y_i\right),$$

  where $\bar{P}$ is large [1].

- Use the sparsification method developed in [L.-Zuazua, 2024] to filter the previous solution, obtaining one with fewer than N activated neurons.

[1] The convergence properties of SGD for the training of overparameterized NNs have been extensively studied recently, including [Chitzat-Bach, 2018], [Zhu-Li-Song, 2019], [Bach, 2024, Chp.12], etc.

High-dimensional scenario

- Apply the SGD algorithm to the following overparameterized problem:

$$\inf_{\Theta\in(\mathbb{R}\times\Omega)^{\bar{P}}} \|\omega\|_{\ell^1} + \frac{\lambda}{N}\sum_{i=1}^{N} \ell\left(\sum_{j=1}^{\bar{P}} \omega_j \sigma(\langle a_j, x_i\rangle + b_j) - y_i\right),$$

  where $\bar{P}$ is large [1].

- Use the sparsification method developed in [L.-Zuazua, 2024] to filter the previous solution, obtaining one with fewer than N activated neurons.

[1] The convergence properties of SGD for the training of overparameterized NNs have been extensively studied recently, including [Chitzat-Bach, 2018], [Zhu-Li-Song, 2019], [Bach, 2024, Chp.12], etc.

High-dimensional scenario

- Apply the SGD algorithm to the following overparameterized problem:

$$\inf_{\Theta\in(\mathbb{R}\times\Omega)^{\bar{P}}} \|\omega\|_{\ell^1} + \frac{\lambda}{N}\sum_{i=1}^{N}\ell\left(\sum_{j=1}^{\bar{P}}\omega_j\sigma(\langle a_j, x_i\rangle + b_j) - y_i\right),$$

  where $\bar{P}$ is large [1].

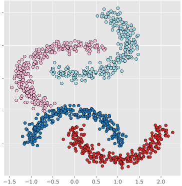
- Use the sparsification method developed in [L.-Zuazua, 2024] to filter the previous solution, obtaining one with fewer than N activated neurons.

This approach is free from the curse of dimensionality but lacks rigorous convergence analysis.

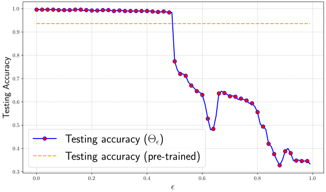
[1] The convergence properties of SGD for the training of overparameterized NNs have been extensively studied recently, including [Chitzat-Bach, 2018], [Zhu-Li-Song, 2019], [Bach, 2024, Chp.12], etc.

Table of Contents

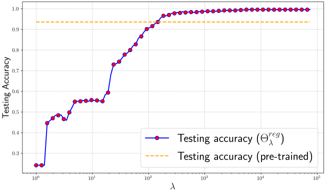
Classification in 2-D



(a) Datasets.

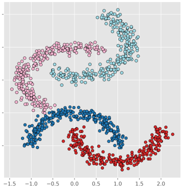


(b) Testing accuracy w.r.t. ϵ.

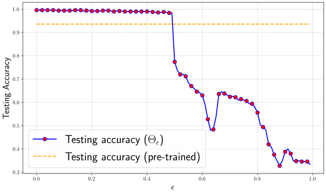


(c) Testing accuracy w.r.t. λ.

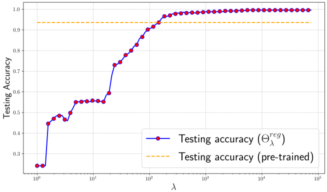
Classification in 2-D



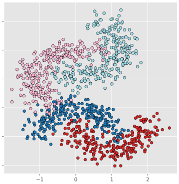
(a) Datasets.



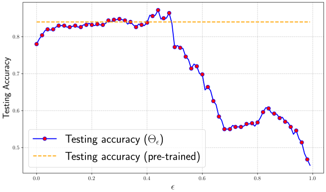
(b) Testing accuracy w.r.t. ϵ.



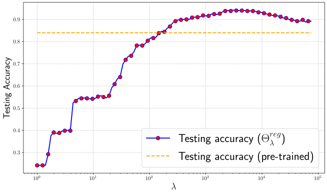
(c) Testing accuracy w.r.t. λ.



(a) Datasets.

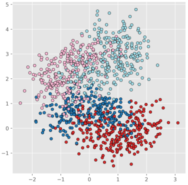


(b) Testing accuracy w.r.t. ϵ.

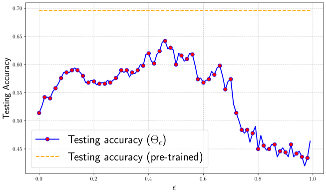


(c) Testing accuracy w.r.t. λ.

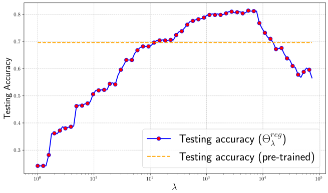
Classification in 2-D



(a) Datasets.

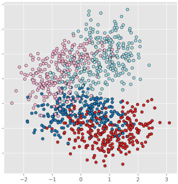


(b) Testing accuracy w.r.t. ϵ.

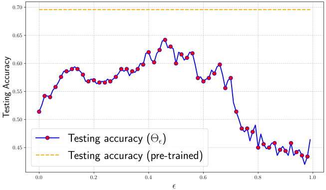


(c) Testing accuracy w.r.t. λ.

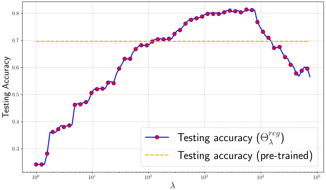
Classification in 2-D



(a) Datasets.



(b) Testing accuracy w.r.t. ϵ.

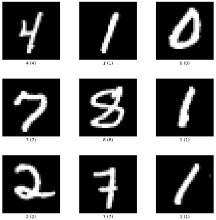


(c) Testing accuracy w.r.t. λ.

Conclusion:

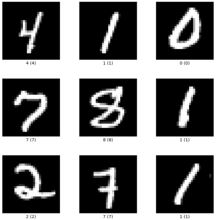
- If the datasets have clear separable boundaries, consider (P_0), (P_ϵ) with $\epsilon \to 0$, or $(\mathrm{P}_\lambda^{\mathrm{reg}})$ with $\lambda \to \infty$;
- If the datasets have heavily overlapping areas, consider the regression problem $(\mathrm{P}_\lambda^{\mathrm{reg}})$ with a particular range of $\lambda \sim W_1^{-1}(m_{\mathrm{train}}, m_{\mathrm{test}})$.

Classification in a high-dimensional space

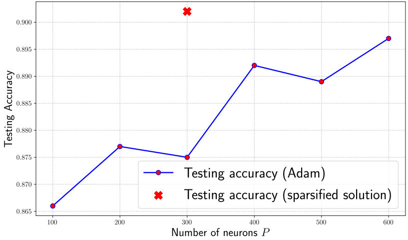


- The Mnist dataset, vectors in $\mathbb{R}^{28\times 28}$.
- Training data: 300 samples of numbers 0, 1, and 2.
- Testing data: 1000 samples of numbers 0, 1, and 2.

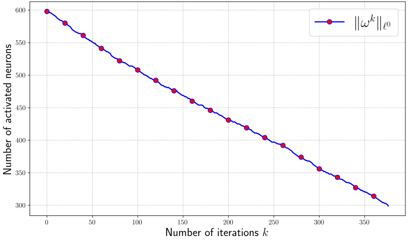
Classification in a high-dimensional space



- The Mnist dataset, vectors in $\mathbb{R}^{28\times 28}$.
- Training data: 300 samples of numbers 0, 1, and 2.
- Testing data: 1000 samples of numbers 0, 1, and 2.



(a) Testing accuracy w.r.t. P.



(b) $\|\omega\|_{\ell^0}$ w.r.t. the iteration number.

Thank you!