Analysis of Neural Networks (2)

Jinchao Xu 许进超

Penn State University

xu@math.psu.edu http://www.math.psu.edu/xu/

CUHK, July 2, 2021

Acknowledgement:
NSF: DMS-1819157

Analysis of Neural Network

Jinchao Xu

This lecture series will provide some mathematical understanding of neural networks and machine learning by studying their close relationship with classic numerical methods such as finite element and multigrid methods. Applications will be given to image classification and numerical partial differential equations

1. Finite Element Connection and Approximation Theory
   - ReLU neural networks $=$ linear finite elements
   - Largest function class that a stable neural network can approximate
   - Optimal approximation rates for popular neural networks
2. Multigrid and Image Classification
   - Linear separable sets and logistic regression
   - A model for feature extractions
   - Image classification by multigrid method
3. Neural Network and Numerical PDEs
   - Error analysis of neural network for numerical PDEs
   - Numerical quadrature and Rademacher complexity analysis
   - Training algorithm that achieves the best asymptotic convergence rate

A basic AI problem: classification

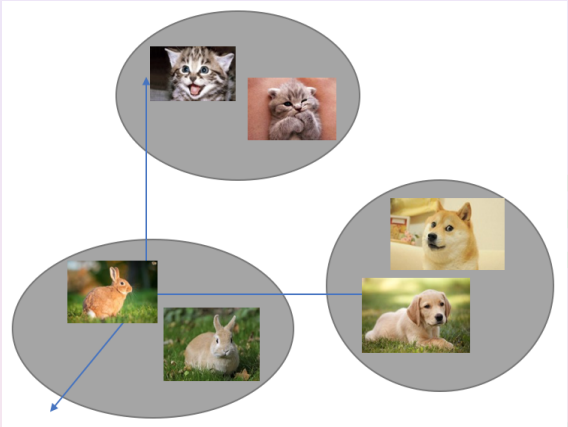
- Can a machine (function) tell the difference ?

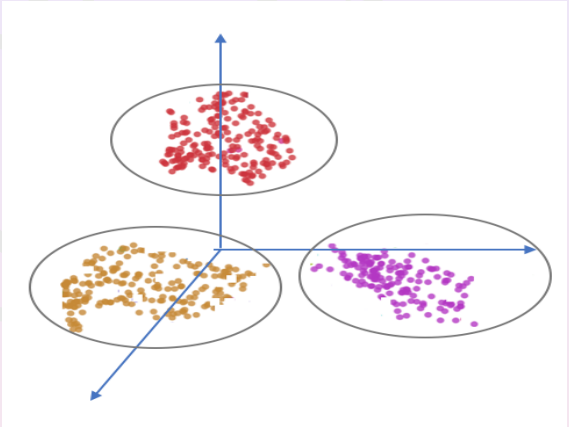
Supervised learning $\leftrightarrow$ Function interpolation

- Each image = a big vector of pixel values
  - $n = 1280 \times 720 \times 3$ (width$\times$ height $\times$ RGB channel) $\approx$ 3M.

Supervised learning $\leftrightarrow$ Function interpolation

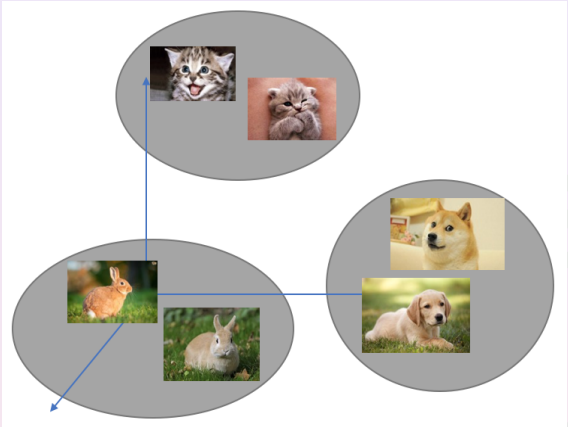
- Each image = a big vector of pixel values
  - $n = 1280 \times 720 \times 3$ (width$\times$ height $\times$ RGB channel) $\approx$ 3M.
- Want to find a function $f(\cdot, \Theta) : \mathbb{R}^n \to \mathbb{R}^3$ which separates the classes

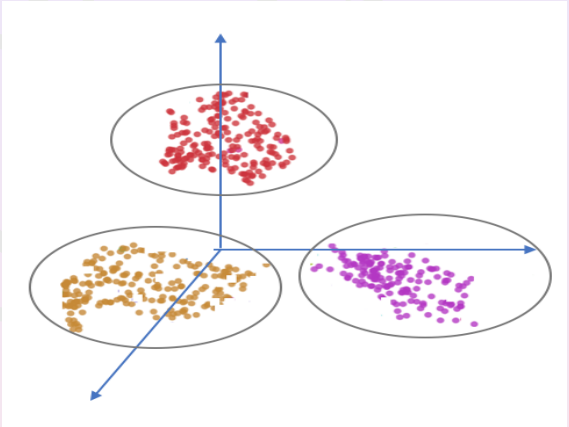




Supervised learning $\leftrightarrow$ Function interpolation

- Each image = a big vector of pixel values
  - $n = 1280 \times 720 \times 3$ (width$\times$ height $\times$ RGB channel) $\approx$ 3M.
- Want to find a function $f(\cdot, \Theta) : \mathbb{R}^n \to \mathbb{R}^3$ which separates the classes





- Mathematical formulation: Find $f(\cdot; \Theta) : \mathbb{R}^n \to \mathbb{R}^3$ such that:

$$f(\text{  }; \Theta) \approx \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad f(\text{  }; \Theta) \approx \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad f(\text{  }; \Theta) \approx \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

“Learning” $\leftrightarrow$ Optimization

“Learning” $\leftrightarrow$ Optimization

- Given data: $\{x_i, y_i\}_{i=1}^N$
  - Image Classification: x_i image, y_i object class
  - Speech Recognition: x_i audio, y_i text
  - Face Recognition: x_i graph with face, y_i identity
  - ...

“Learning” $\leftrightarrow$ Optimization

- Given data: $\{x_i, y_i\}_{i=1}^N$
  - Image Classification: x_i image, y_i object class
  - Speech Recognition: x_i audio, y_i text
  - Face Recognition: x_i graph with face, y_i identity
  - ...
- How can we find $f(\cdot, \Theta)$?.

"Learning" $\leftrightarrow$ Optimization

- Given data: $\{x_i, y_i\}_{i=1}^N$
  - Image Classification: x_i image, y_i object class
  - Speech Recognition: x_i audio, y_i text
  - Face Recognition: x_i graph with face, y_i identity
  - ...
- How can we find $f(\cdot, \Theta)$?.
- Mathematically: solve the **optimization problem**

$$\min_{\Theta} \sum_{i=1}^{N} \ell(f(x_i, \Theta), y_i) \tag{1}$$

where ℓ is an appropriate *loss function*.

"Learning" $\leftrightarrow$ Optimization

- Given data: $\{x_i, y_i\}_{i=1}^N$
  - Image Classification: x_i image, y_i object class
  - Speech Recognition: x_i audio, y_i text
  - Face Recognition: x_i graph with face, y_i identity
  - ...
- How can we find $f(\cdot, \Theta)$?.
- Mathematically: solve the **optimization problem**

$$\min_{\Theta} \sum_{i=1}^{N} \ell(f(x_i, \Theta), y_i) \tag{1}$$

  where ℓ is an appropriate *loss function*.
- Application (generalization): image classification:

“Learning” $\leftrightarrow$ Optimization

- Given data: $\{x_i, y_i\}_{i=1}^N$
  - Image Classification: x_i image, y_i object class
  - Speech Recognition: x_i audio, y_i text
  - Face Recognition: x_i graph with face, y_i identity
  - ...
- How can we find $f(\cdot, \Theta)$?.
- Mathematically: solve the **optimization problem**

$$\min_{\Theta} \sum_{i=1}^{N} \ell(f(x_i, \Theta), y_i) \tag{1}$$

  where ℓ is an appropriate *loss function*.
- Application (generalization): image classification:

$$f(\boxed{\ ?\ }; \Theta) = \begin{pmatrix} 0.7 \\ 0.2 \\ 0.1 \end{pmatrix}$$

"Learning" $\leftrightarrow$ Optimization

- Given data: $\{x_i, y_i\}_{i=1}^N$
  - Image Classification: x_i image, y_i object class
  - Speech Recognition: x_i audio, y_i text
  - Face Recognition: x_i graph with face, y_i identity
  - ...
- How can we find $f(\cdot, \Theta)$?.
- Mathematically: solve the **optimization problem**

$$\min_{\Theta} \sum_{i=1}^{N} \ell(f(x_i, \Theta), y_i) \tag{1}$$

  where ℓ is an appropriate *loss function*.
- Application (generalization): image classification:

$$f(\boxed{?}; \Theta) = \begin{pmatrix} 0.7 \\ 0.2 \\ 0.1 \end{pmatrix} \quad \Longrightarrow \quad \boxed{?} =$$

"Learning" $\leftrightarrow$ Optimization

- Given data: $\{x_i, y_i\}_{i=1}^N$
  - Image Classification: x_i image, y_i object class
  - Speech Recognition: x_i audio, y_i text
  - Face Recognition: x_i graph with face, y_i identity
  - ...
- How can we find $f(\cdot, \Theta)$?.
- Mathematically: solve the **optimization problem**

$$\min_{\Theta} \sum_{i=1}^{N} \ell(f(x_i, \Theta), y_i) \tag{1}$$

  where ℓ is an appropriate *loss function*.
- Application (generalization): image classification:

$$f(\boxed{?}; \Theta) = \begin{pmatrix} 0.7 \\ 0.2 \\ 0.1 \end{pmatrix} \quad \Longrightarrow \quad \boxed{?} = \text{cat}$$

Training and Generalization Error

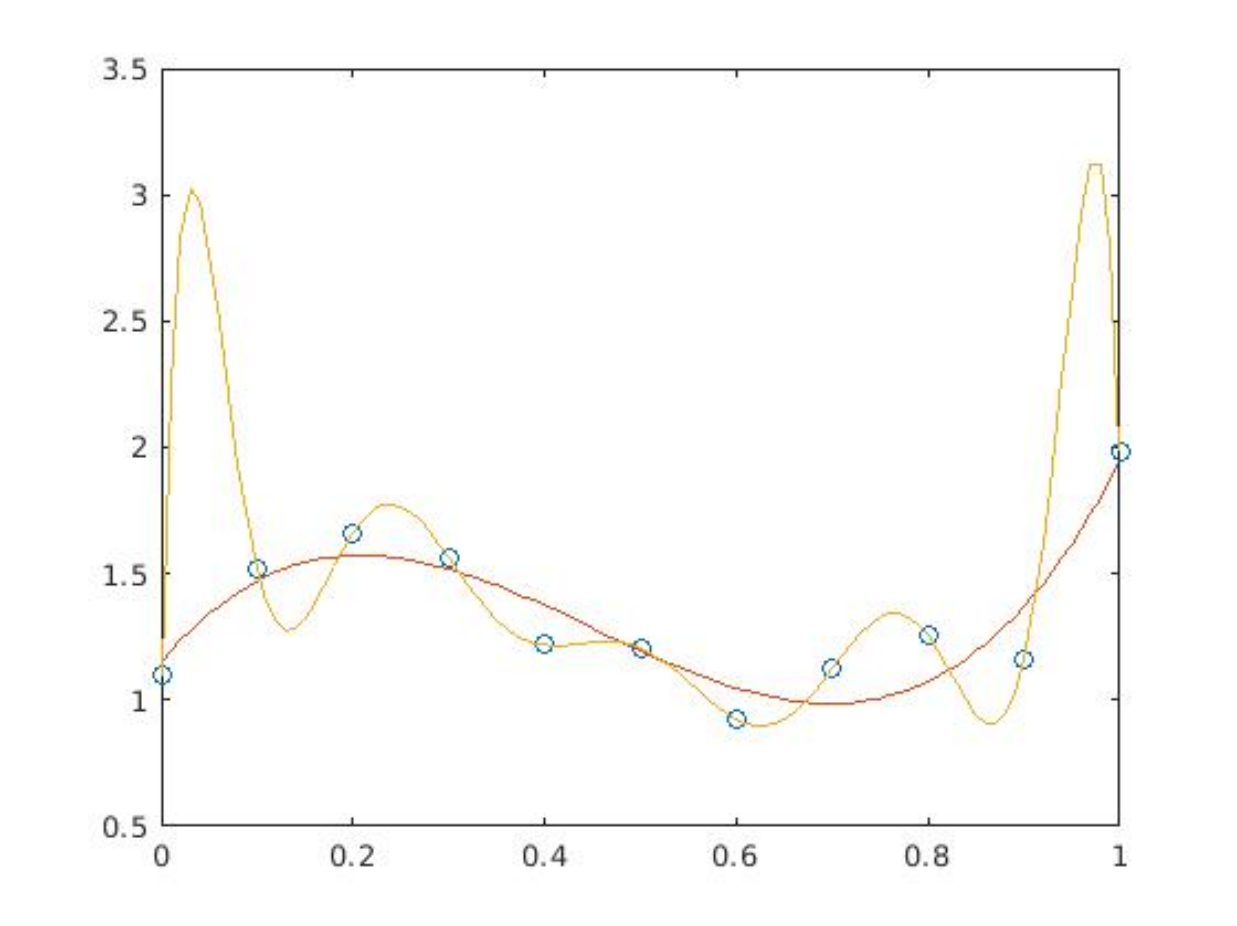
- Training Error:
  - Fit the training dataset
  - Evaluate error or loss

Training and Generalization Error

- Training Error:
  - Fit the training dataset
  - Evaluate error or loss
- Generalization Error:
  - Performance in the *real world*
  - Theoretical, cannot be measured exactly

Overfitting

- Fit noisy function values



- Training error is 0
- Generalization Error is large

Regularization

- How can we prevent overfitting?
  - Overfitting occurs because the model class is to large relative to the data
  - Need to restrict the model class $\mathcal{F}$ in some way
- Typical way of restricting the model class:
  - Add a *regularization* term $R(\theta)$ to the loss function
- Typical regularizers:
  - ℓ^2-loss: $R(\theta) = \|\theta\|_2^2$
  - ℓ^1-loss: $R(\theta) = \|\theta\|_1$
- In deep learning SGD acts as a regularizer!
  - Deep learning model and training algorithm are tied together.

Bounding the Generalization Error

- Main Tool:
  - Uniform law of large numbers[1]:

$$\mathbb{P}\left(\max_{f\in\mathcal{F}}\left|\frac{1}{N}\sum_{i=1}^{N}L(f(x_i),y_i)-\mathbb{E}L(f(x),y)\right|>\epsilon(N)\right)<1-\delta(N). \tag{2}$$

- If the model class $\mathcal{F}$ is small, $\epsilon(N),\delta(N)\to 0$ very quickly
- Main tools for bounding this
  - Rademacher Complexity
  - VC dimension
  - Metric Entropy (more advanced)

[1] **vapnik2013nature**.

Generalization Error (cont.)

- The Rademacher complexity of a class of function $\mathcal{F}$ on Ω

$$R_N(F) = \mathbb{E}_{x_1,\ldots,x_N}\mathbb{E}_{\xi_1,\ldots,\xi_N}\left(\sup_{f\in\mathcal{F}}\frac{1}{N}\sum_{i=1}^{N}\xi_i f(x_i)\right), \tag{3}$$

  where x_i are drawn uniformly at random from Ω and ξ_i are uniformly random signs.
- Can use the Rademacher complexity to obtain a uniform law of large numbers[2]
- Rademacher complexity of neural networks $O(N^{-\frac{1}{2}})$[3][4]

[2] **vapnik2013nature**.

[3] **ma2019barron**.

[4] **hong2021rademacher**.

Linearly separable sets: 2-class

Definition

We say $A_1, A_2 \subset \mathbb{R}^n$ are linearly separable, if there exists a hyperplane

$$H_0 = \{x : wx + b = 0\}, \tag{4}$$

such that $wx + b > 0$ if $x \in A_1$ and $wx + b < 0$ if $x \in A_2$.

Linearly separable sets: 2-class

Definition

We say $A_1, A_2 \subset \mathbb{R}^n$ are linearly separable, if there exists a hyperplane

$$H_0 = \{x : wx + b = 0\}, \tag{4}$$

such that $wx + b > 0$ if $x \in A_1$ and $wx + b < 0$ if $x \in A_2$.

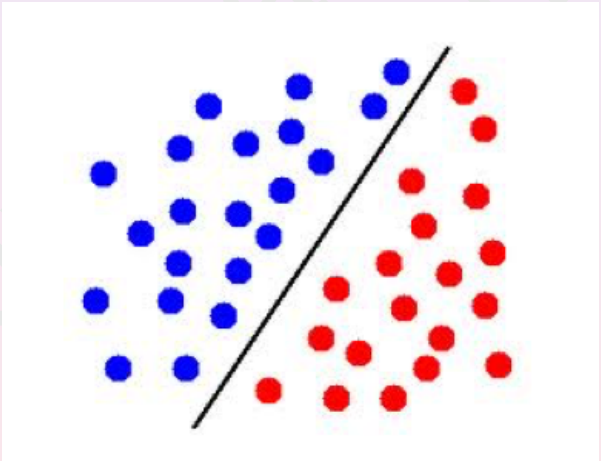


Figure: Two linearly separable sets

Linearly separable sets: k-class

Definition (Linearly Separable)

A collection of subsets $A_1, ..., A_k \subset \mathbb{R}^n$ are linearly separable if there exist

$$W = \begin{pmatrix} w_1 \\ \vdots \\ w_k \end{pmatrix} \in \mathbb{R}^{k \times n}, b = \begin{pmatrix} b_1 \\ \vdots \\ b_k \end{pmatrix} \in \mathbb{R}^{k \times n}, \tag{5}$$

such that, for each $1 \leq i \leq k$ and $j \neq i$, for any $x \in A_i$

$$w_i x + b_i > w_j x + b_j, \text{ or } (w_i - w_j)x + b_i - b_j > 0. \tag{6}$$

Linearly separable sets: k-class

Definition (Linearly Separable)

A collection of subsets $A_1, ..., A_k \subset \mathbb{R}^n$ are linearly separable if there exist

$$W = \begin{pmatrix} w_1 \\ \vdots \\ w_k \end{pmatrix} \in \mathbb{R}^{k \times n}, b = \begin{pmatrix} b_1 \\ \vdots \\ b_k \end{pmatrix} \in \mathbb{R}^{k \times n}, \tag{5}$$

such that, for each $1 \leq i \leq k$ and $j \neq i$, for any $x \in A_i$

$$w_i x + b_i > w_j x + b_j, \text{ or } (w_i - w_j)x + b_i - b_j > 0. \tag{6}$$

Special case: $k = 2$

$$(w, b) = (w_1 - w_2, b_1 - b_2).$$

Linearly separable sets: k-class

Definition (Linearly Separable)

A collection of subsets $A_1, ..., A_k \subset \mathbb{R}^n$ are linearly separable if there exist

$$W = \begin{pmatrix} w_1 \\ \vdots \\ w_k \end{pmatrix} \in \mathbb{R}^{k \times n}, b = \begin{pmatrix} b_1 \\ \vdots \\ b_k \end{pmatrix} \in \mathbb{R}^{k \times n}, \tag{5}$$

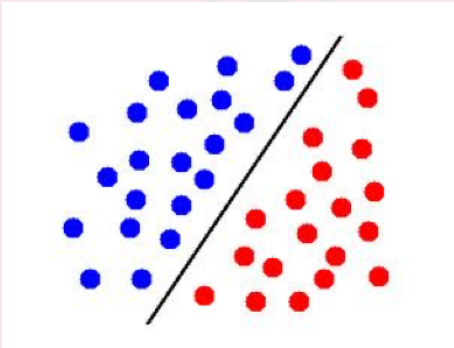
such that, for each $1 \leq i \leq k$ and $j \neq i$, for any $x \in A_i$

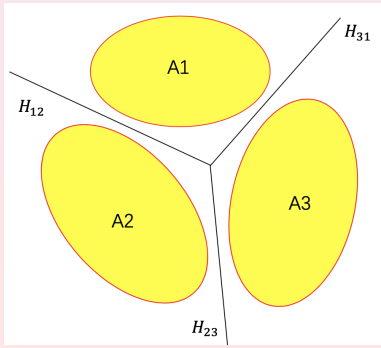
$$w_i x + b_i > w_j x + b_j, \text{ or } (w_i - w_j)x + b_i - b_j > 0. \tag{6}$$

Special case: $k = 2$

$$(w, b) = (w_1 - w_2, b_1 - b_2).$$

Examples of linear separable sets





Logistic regression

Soft-max:

$$softmax(x)$$

Logistic regression

Soft-max:

$$softmax(x) = \frac{1}{\sum_{i=1}^{k} e^{x_i}} \begin{pmatrix} e^{x_1} \\ e^{x_2} \\ \cdots \\ e^{x_k} \end{pmatrix},$$

Logistic regression

Soft-max:

$$softmax(x) = \frac{1}{\sum_{i=1}^{k} e^{x_i}} \begin{pmatrix} e^{x_1} \\ e^{x_2} \\ \cdots \\ e^{x_k} \end{pmatrix}, \tag{7}$$

Likelyhood function

$$P(\theta) = \prod_{i=1}^{k} \prod_{x \in A_i} p_i(\theta, x), \quad 0 < p_i(\boldsymbol{\theta}, x) < 1, \quad \sum_{i=1}^{k} p_i(\boldsymbol{\theta}, x) = 1. \tag{8}$$

where $\boldsymbol{p}(\boldsymbol{\theta}, x) = softmax(Wx + b)$ and $\boldsymbol{p}(\boldsymbol{\theta}, x) = (p_1(\boldsymbol{\theta}, x), \cdots, p_k(\boldsymbol{\theta}, x))$.

Logistic regression

Logistic regression

Loss function

$$\theta_\lambda = (W_\lambda, b_\lambda)$$

Logistic regression

Loss function

$$\theta_\lambda = (W_\lambda, b_\lambda) = \arg\max_\theta [P(\theta)]$$

Logistic regression

Loss function

$$\theta_\lambda = (W_\lambda, b_\lambda) = \arg\max_\theta [P(\theta)] = \arg\min_\theta \left(-\log P(\theta)\right), \tag{9}$$

here

$$-\log P(\theta) = -\sum_{j=1}^{N} y_j \cdot \log(\boldsymbol{p}(\theta, x_j)) \Leftrightarrow \text{cross-entropy}. \tag{10}$$

Classification

If

$$p_i(\theta_\lambda, x_j) > p_m(\theta_\lambda, x_j), \quad \forall m \neq i, \tag{11}$$

then

$$x_j \in A_i. \tag{12}$$

Existence

Consider:

$$\theta_\lambda = (W_\lambda, b_\lambda)$$

Existence

Consider:

$$\theta_\lambda = (W_\lambda, b_\lambda) = \arg\min_\theta \left(\sum_{j=1}^{N} \ell(y_j, \boldsymbol{p}(\theta, x_j)) + \lambda R(\theta) \right) \tag{13}$$

with $\ell(y, p(\theta, x)) = -y \cdot \log(\boldsymbol{p}(\theta, x))$

Existence

Consider:

$$\theta_\lambda = (W_\lambda, b_\lambda) = \arg\min_\theta \left(\sum_{j=1}^{N} \ell(y_j, \boldsymbol{p}(\theta, x_j)) + \lambda R(\theta) \right) \tag{13}$$

with $\ell(y, p(\theta, x)) = -y \cdot \log(\boldsymbol{p}(\theta, x))$

Assume that $\{x_j\}_{j=1}^N = \cup_{i=1}^k A_i$ is linearly separable,

Existence

Consider:

$$\theta_\lambda = (W_\lambda, b_\lambda) = \arg\min_\theta \left(\sum_{j=1}^{N} \ell(y_j, \boldsymbol{p}(\theta, x_j)) + \lambda R(\theta) \right) \tag{13}$$

with $\ell(y, p(\theta, x)) = -y \cdot \log(\boldsymbol{p}(\theta, x))$

Assume that $\{x_j\}_{j=1}^N = \cup_{i=1}^k A_i$ is linearly separable,

1. If $\lambda > 0$ is sufficiently small, $W_\lambda x + b_\lambda$ classifies the sets correctly.

Existence

Consider:

$$\theta_\lambda = (W_\lambda, b_\lambda) = \arg\min_\theta \left(\sum_{j=1}^{N} \ell(y_j, \boldsymbol{p}(\theta, x_j)) + \lambda R(\theta) \right) \tag{13}$$

with $\ell(y, p(\theta, x)) = -y \cdot \log(\boldsymbol{p}(\theta, x))$

Assume that $\{x_j\}_{j=1}^N = \cup_{i=1}^k A_i$ is linearly separable,

1. If $\lambda > 0$ is sufficiently small, $W_\lambda x + b_\lambda$ classifies the sets correctly.
2. If $\lambda = 0$, no global miminum

Existence

Consider:

$$\theta_\lambda = (W_\lambda, b_\lambda) = \arg\min_\theta \left(\sum_{j=1}^{N} \ell(y_j, \boldsymbol{p}(\theta, x_j)) + \lambda R(\theta) \right) \tag{13}$$

with $\ell(y, p(\theta, x)) = -y \cdot \log(\boldsymbol{p}(\theta, x))$

Assume that $\{x_j\}_{j=1}^N = \cup_{i=1}^k A_i$ is linearly separable,

1. If $\lambda > 0$ is sufficiently small, $W_\lambda x + b_\lambda$ classifies the sets correctly.
2. If $\lambda = 0$, no global miminum

Comments:

Existence

Consider:

$$\theta_\lambda = (W_\lambda, b_\lambda) = \arg\min_\theta \left(\sum_{j=1}^{N} \ell(y_j, \boldsymbol{p}(\theta, x_j)) + \lambda R(\theta) \right) \tag{13}$$

with $\ell(y, p(\theta, x)) = -y \cdot \log(\boldsymbol{p}(\theta, x))$

Assume that $\{x_j\}_{j=1}^N = \cup_{i=1}^k A_i$ is linearly separable,

1. If $\lambda > 0$ is sufficiently small, $W_\lambda x + b_\lambda$ classifies the sets correctly.
2. If $\lambda = 0$, no global miminum

Comments:

- $-\log P(\theta) \Leftrightarrow$ cross-entropy.

Existence

Consider:

$$\theta_\lambda = (W_\lambda, b_\lambda) = \arg\min_\theta \left(\sum_{j=1}^{N} \ell(y_j, \boldsymbol{p}(\theta, x_j)) + \lambda R(\theta) \right) \tag{13}$$

with $\ell(y, p(\theta, x)) = -y \cdot \log(\boldsymbol{p}(\theta, x))$

Assume that $\{x_j\}_{j=1}^N = \cup_{i=1}^k A_i$ is linearly separable,

1. If $\lambda > 0$ is sufficiently small, $W_\lambda x + b_\lambda$ classifies the sets correctly.
2. If $\lambda = 0$, no global miminum

Comments:

- $-\log P(\theta) \Leftrightarrow$ cross-entropy.
- Regularization is necessary even for convex model for linearly separable sets!

Nonlinear classification models

Definition

Given data $\{x_j\}_{j=1}^N = \cup_{i=1}^k A_i$, if there exist a continuous nonlinear (feature) map $\phi(\cdot, \theta) : \mathbb{R}^d \mapsto \mathbb{R}^n$ so that

$$\{\phi(x_i, \theta)\}_{i=1}^N = \bigcup_{i=1}^k \phi(A_i, \theta)$$

is linearly separable, then $\{x_j\}_{j=1}^N$ is called nonlinearly separable.

Nonlinear classification models

Definition

Given data $\{x_j\}_{j=1}^N = \cup_{i=1}^k A_i$, if there exist a continuous nonlinear (feature) map $\phi(\cdot, \theta) : \mathbb{R}^d \mapsto \mathbb{R}^n$ so that

$$\{\phi(x_i, \theta)\}_{i=1}^N = \bigcup_{i=1}^k \phi(A_i, \theta)$$

is linearly separable, then $\{x_j\}_{j=1}^N$ is called nonlinearly separable.

Example: $A_1 \subset \{(x, y)|x^2 + y^2 < 1\}, \quad A_2 \subset \{(x, y)|x^2 + y^2 > 1\}$

Nonlinear classification models

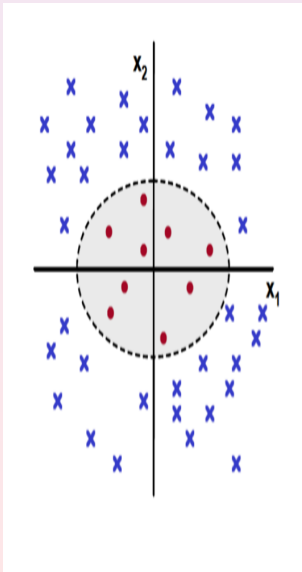
Definition

Given data $\{x_j\}_{j=1}^N = \cup_{i=1}^k A_i$, if there exist a continuous nonlinear (feature) map $\phi(\cdot, \theta) : \mathbb{R}^d \mapsto \mathbb{R}^n$ so that

$$\{\phi(x_i, \theta)\}_{i=1}^N = \bigcup_{i=1}^k \phi(A_i, \theta)$$

is linearly separable, then $\{x_j\}_{j=1}^N$ is called nonlinearly separable.

Example: $A_1 \subset \{(x, y)|x^2 + y^2 < 1\}, \quad A_2 \subset \{(x, y)|x^2 + y^2 > 1\}$



Nonlinear classification models

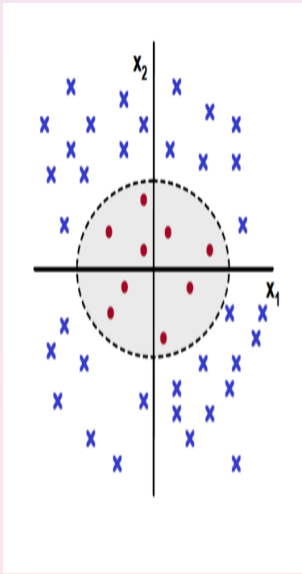
Definition

Given data $\{x_j\}_{j=1}^N = \cup_{i=1}^k A_i$, if there exist a continuous nonlinear (feature) map $\phi(\cdot, \theta) : \mathbb{R}^d \mapsto \mathbb{R}^n$ so that

$$\{\phi(x_i, \theta)\}_{i=1}^N = \bigcup_{i=1}^k \phi(A_i, \theta)$$

is linearly separable, then $\{x_j\}_{j=1}^N$ is called nonlinearly separable.

Example: $A_1 \subset \{(x, y)|x^2 + y^2 < 1\}, \quad A_2 \subset \{(x, y)|x^2 + y^2 > 1\}$



$$\xrightarrow[\text{feature map}]{\phi(x,y)=x^2+y^2}$$

Nonlinear classification models

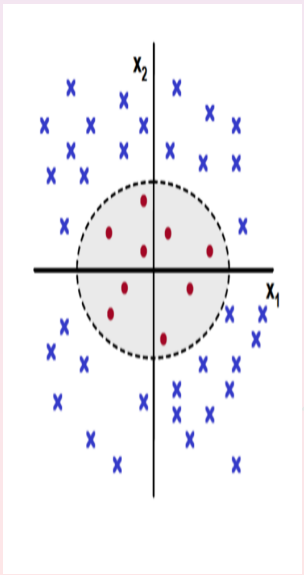
Definition

Given data $\{x_j\}_{j=1}^N = \cup_{i=1}^k A_i$, if there exist a continuous nonlinear (feature) map $\phi(\cdot, \theta) : \mathbb{R}^d \mapsto \mathbb{R}^n$ so that

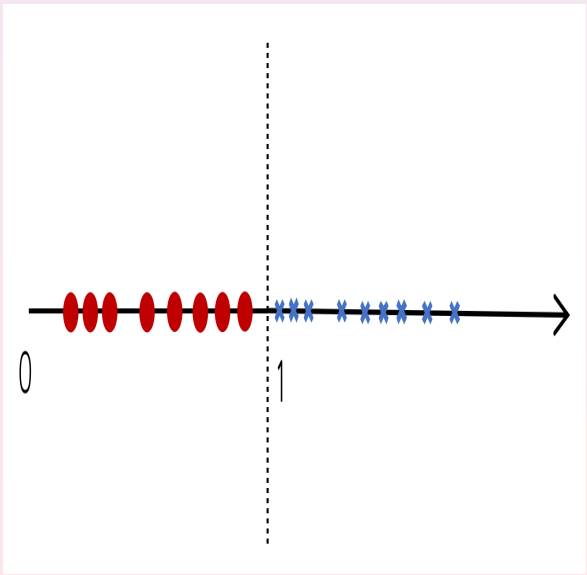
$$\{\phi(x_i, \theta)\}_{i=1}^N = \bigcup_{i=1}^k \phi(A_i, \theta)$$

is linearly separable, then $\{x_j\}_{j=1}^N$ is called nonlinearly separable.

Example: $A_1 \subset \{(x, y)|x^2 + y^2 < 1\}, \quad A_2 \subset \{(x, y)|x^2 + y^2 > 1\}$



$\xrightarrow[\text{feature map}]{\phi(x,y)=x^2+y^2}$



Neural network classification models

Theorem

If data $\{x_j\}_{j=1}^N$ is nonlinearly separable, then there exist a neural network function ϕ such that $\{\phi(x_j,\theta)\}_{j=1}^N$ is linearly separable.

- J. Xu, Deep Neural Networks and Multigrid Methods, (Lecture Notes at Penn State), 2021.

Nonlinear classification models

Popular nonlinear models for $\phi(\cdot, \theta)$:

Nonlinear classification models

Popular nonlinear models for $\phi(\cdot, \theta)$:

1. Polynomials

Nonlinear classification models

Popular nonlinear models for $\phi(\cdot, \theta)$:

1. Polynomials
2. Kernel functions in SVM ($k(x, y) = (\phi(x), \phi(y))$)

Nonlinear classification models

Popular nonlinear models for $\phi(\cdot, \theta)$:

1. Polynomials
2. Kernel functions in SVM ($k(x, y) = (\phi(x), \phi(y))$)
3. Neural Network functions

Nonlinear classification models

Popular nonlinear models for $\phi(\cdot, \theta)$:

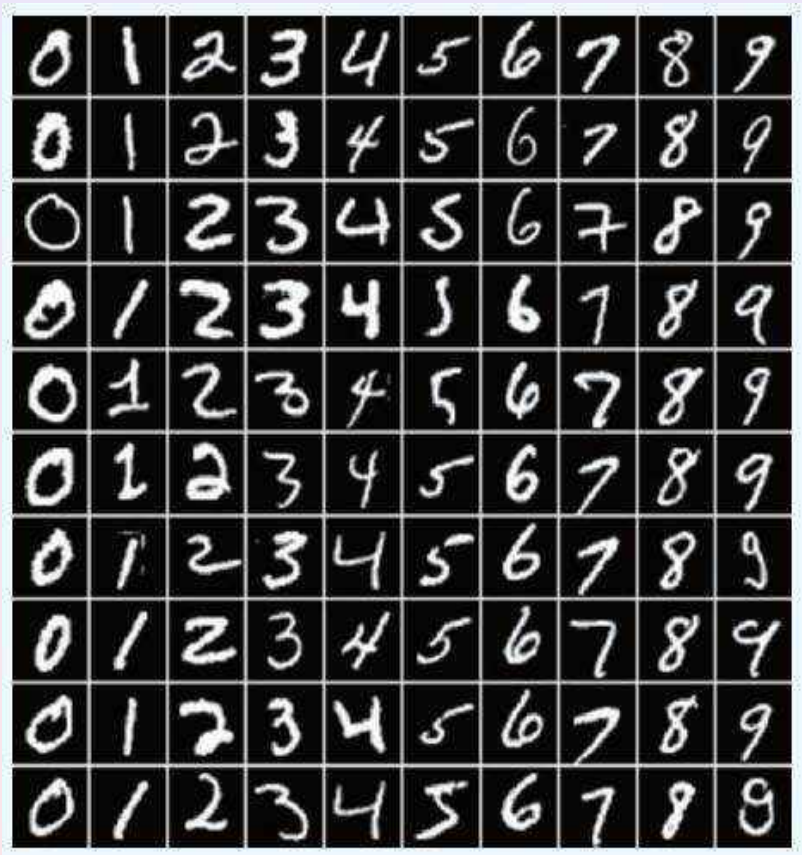
1. Polynomials
2. Kernel functions in SVM ($k(x, y) = (\phi(x), \phi(y))$)
3. Neural Network functions

Loss function using regularized logistic regression:

$$L = \sum_{j=1}^{N} \ell(y_j, p(\theta_L, \phi(x_j, \theta))) + \lambda \left(R(\theta_L) + R_N(\theta)\right)$$

Example: MNIST data set

(Modified National Institute of Standards and Technology database)



Handwritten digits:

- Training set : 60,000
- Test set : 10,000
- Image size : 28*28*1
- Classes: 10

MNIST is not linearly separable

Logistic regression

$$P = LR(x) = softmax(Wx + b) \tag{14}$$

where, $W \in \mathbb{R}^{10\times 784}$.

MNIST is not linearly separable

Logistic regression

$$P = LR(x) = softmax(Wx + b) \tag{14}$$

where, $W \in \mathbb{R}^{10 \times 784}$.

Training accuracy	Test accuracy
93.44	92.50

Table: Logistic regression on MNIST.

MNIST is not linearly separable

Logistic regression

$$P = LR(x) = softmax(Wx + b) \tag{14}$$

where, $W \in \mathbb{R}^{10\times 784}$.

Training accuracy	Test accuracy
93.44	92.50

Table: Logistic regression on MNIST.

Logistic regression with DNN_1

$$P = softmax(W_2\sigma(W_1 x + b_1) + b_2) \tag{15}$$

where, $W_1 \in \mathbb{R}^{1000\times 784}$, $W_2 \in \mathbb{R}^{10\times 1000}$ and $\sigma = \mathrm{ReLU}$.

MNIST is not linearly separable

Logistic regression

$$P = LR(x) = softmax(Wx + b) \tag{14}$$

where, $W \in \mathbb{R}^{10\times 784}$.

Training accuracy	Test accuracy
93.44	92.50

Table: Logistic regression on MNIST.

Logistic regression with DNN_1

$$P = softmax(W_2\sigma(W_1 x + b_1) + b_2) \tag{15}$$

where, $W_1 \in \mathbb{R}^{1000\times 784}$, $W_2 \in \mathbb{R}^{10\times 1000}$ and $\sigma = \mathrm{ReLU}$.

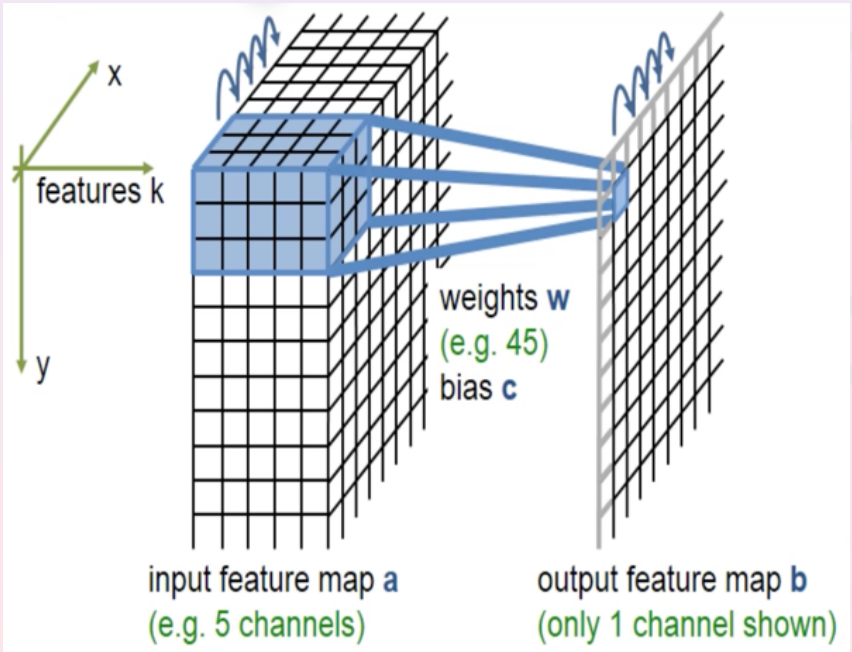
Training accuracy	Test accuracy
100.00	98.75

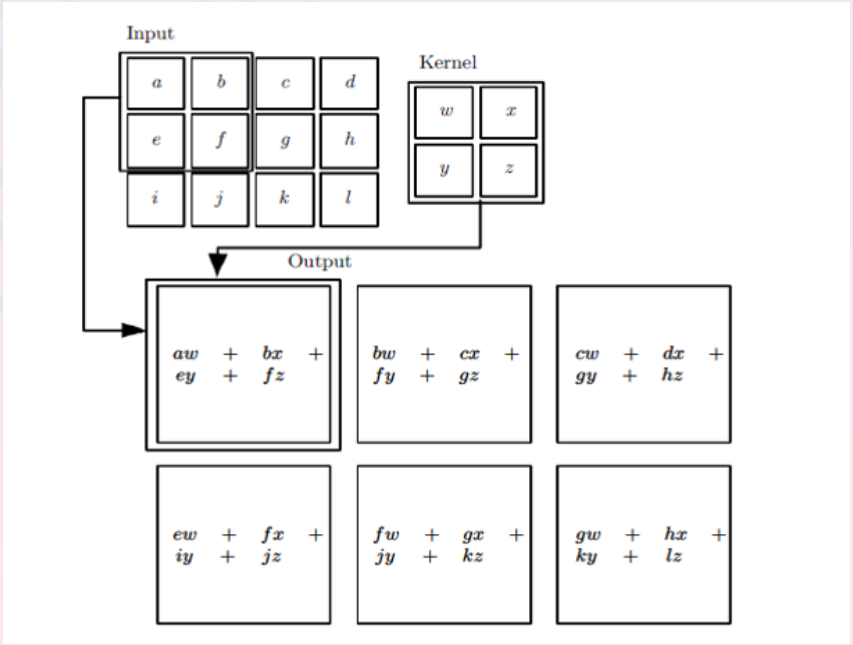
Table: LR with DNN_1 on MNIST.

Convolution: a special class of linear functions

$$(K * x)_{i,j} := \sum_{\ell=1}^{c} \sum_{s,t=-k}^{k} K_{s+k+1,t+k+1,\ell} x_{i+s,j+t,\ell}, \tag{16}$$

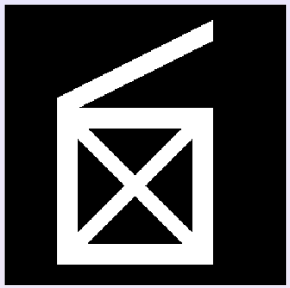
- (i, j): pixel points, ℓ: channel dimension





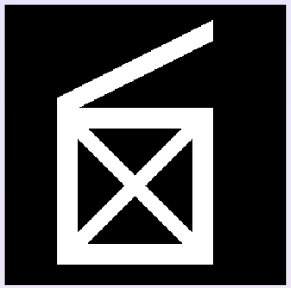
Examples: edge detection by convolution

Examples: edge detection by convolution

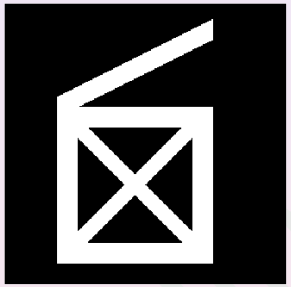


$$\begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix}$$

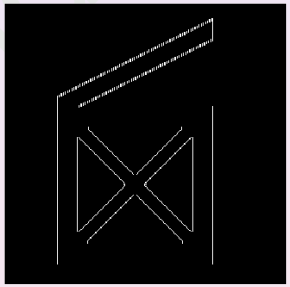
Examples: edge detection by convolution



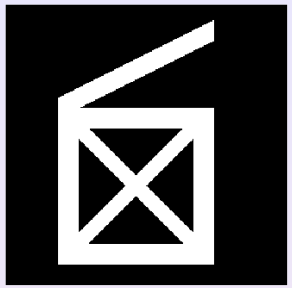
$$\begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix}$$



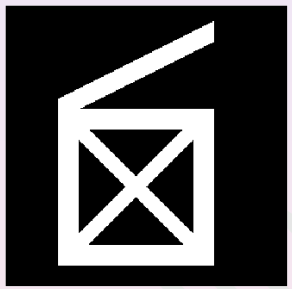
$$\begin{pmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{pmatrix}$$



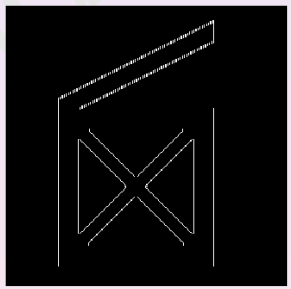
Examples: edge detection by convolution

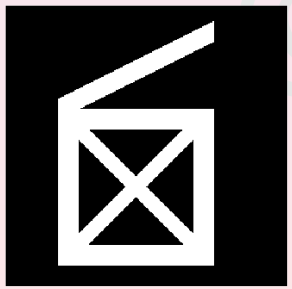


$$\begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix}$$

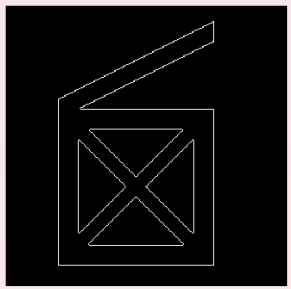


$$\begin{pmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{pmatrix}$$





$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$



Example of feature extraction by convolution: Laplacian of Gaussian

Example of feature extraction by convolution: Laplacian of Gaussian

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \xrightarrow{\text{filter}}$$

Example of feature extraction by convolution: Laplacian of Gaussian

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \xrightarrow{\text{filter}}$$

Example of feature extraction by convolution: Laplacian of Gaussian

$$\xrightarrow{\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}}$$

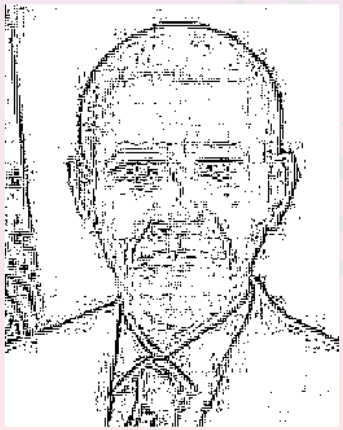
filter

$$\xrightarrow{\text{ReLU}}$$

Example of feature extraction by convolution: Laplacian of Gaussian

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \xrightarrow{\text{filter}}$$

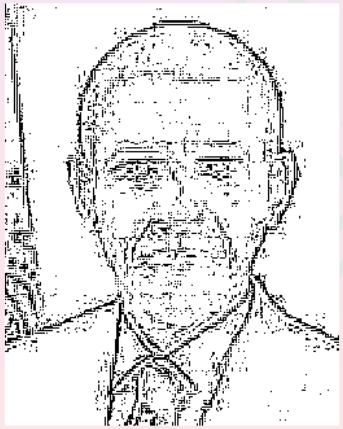
ReLU $\longrightarrow$



Example of feature extraction by convolution: Laplacian of Gaussian

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \xrightarrow[\text{filter}]{}$$

ReLU $\longrightarrow$

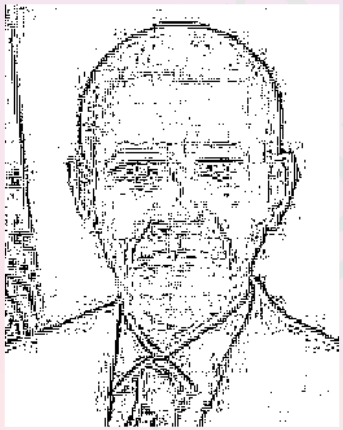


$$\begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix} \xrightarrow[\text{average}]{}$$

Example of feature extraction by convolution: Laplacian of Gaussian

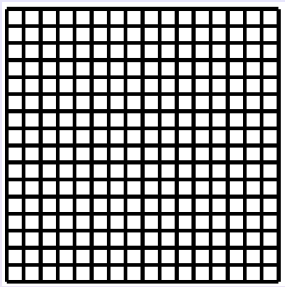
$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \xrightarrow[\text{filter}]{}$$

ReLU $\longrightarrow$

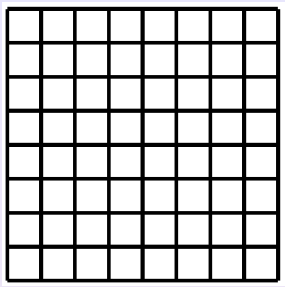


$$\begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix} \xrightarrow[\text{average}]{}$$

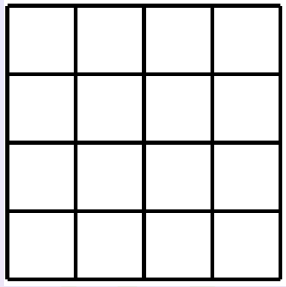
Sub-sampling (coarsening): stride and pooling



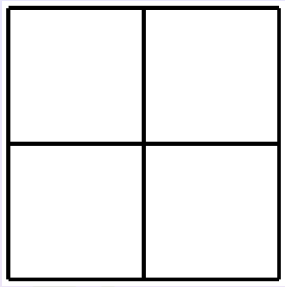
$\mathcal{T}_1$



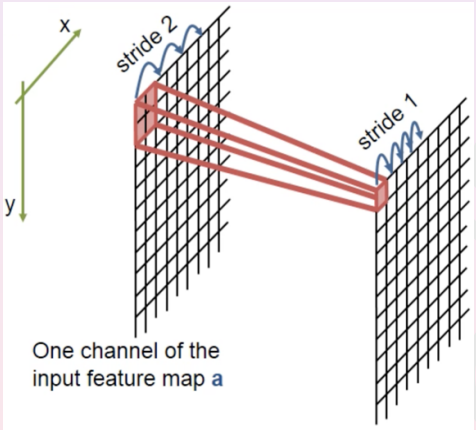
$\mathcal{T}_2$



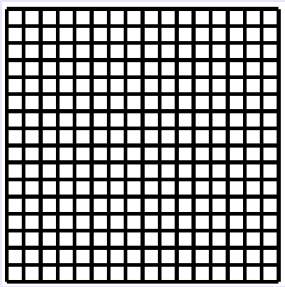
$\mathcal{T}_3$



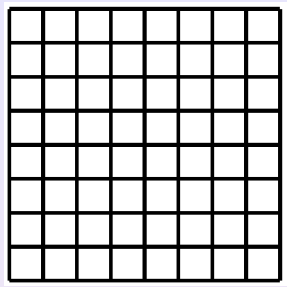
$\mathcal{T}_4$



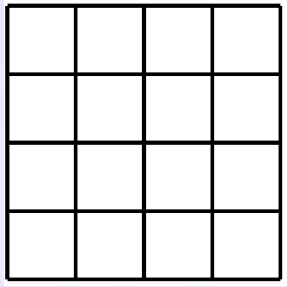
Sub-sampling (coarsening): stride and pooling



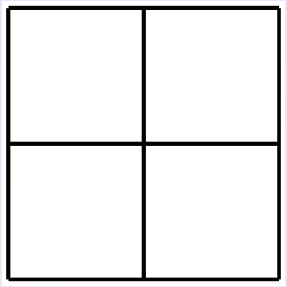
$\mathcal{T}_1$



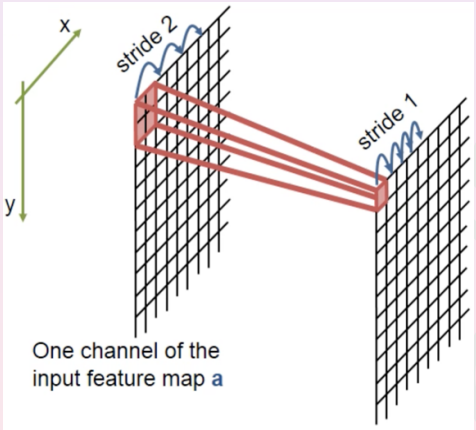
$\mathcal{T}_2$

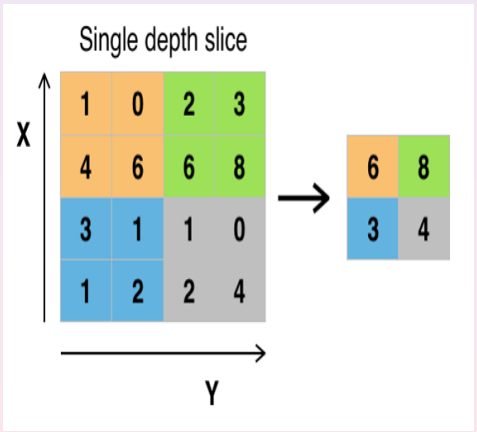


$\mathcal{T}_3$



$\mathcal{T}_4$





- Convolution with stride

$$(K *_s x)_{i,j} = \sum_{\ell=1}^{c} \sum_{p,q=-k}^{k} K_{p+k+1,q+k+1} x_{is+p,js+q}.$$

- Pooling (max-pooling)

$$r(x)_{i,j} = \max\{x_{(i-1)s+p,(j-1)s+q} \ : \ p,q = 1:s\}.$$

Classic CNN

Classic CNN

Classic CNN structure

1. Initialization of inputs: $g^{1,0}$

Classic CNN

Classic CNN structure

1. Initialization of inputs: $g^{1,0}$
2. For $\ell = 1 : J$

Classic CNN

Classic CNN structure

1. Initialization of inputs: $g^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

Classic CNN

Classic CNN structure

1. Initialization of inputs: $g^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

$$g^{\ell,i} = \sigma\left(\theta^{\ell,i} * g^{\ell,i-1}\right) \tag{17}$$

Classic CNN

Classic CNN structure

1. Initialization of inputs: $g^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

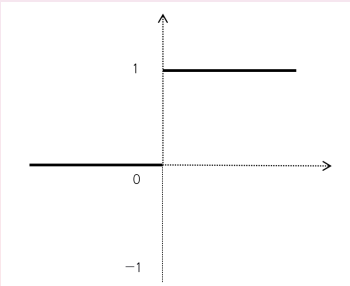
$$g^{\ell,i} = \sigma\left(\theta^{\ell,i} * g^{\ell,i-1}\right) \tag{17}$$

   - Restriction-Pooling

$$g^{\ell+1,0} = R_\ell^{\ell+1} *_2 g^{\ell,\nu_\ell} \tag{18}$$

3. Output g^{J,ν_J}.

Neuron Activation Function



$$\sigma = \mathrm{H}(x)$$

Classic CNN

Classic CNN structure

1. Initialization of inputs: $g^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

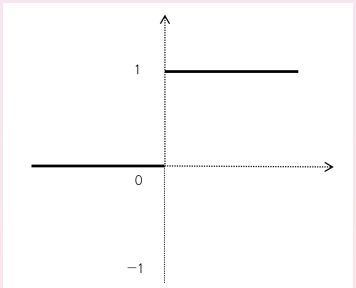
$$g^{\ell,i} = \sigma\left(\theta^{\ell,i} * g^{\ell,i-1}\right) \tag{17}$$

   - Restriction-Pooling

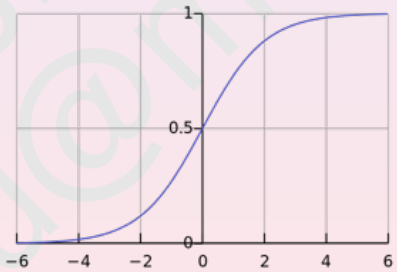
$$g^{\ell+1,0} = R_\ell^{\ell+1} *_2 g^{\ell,\nu_\ell} \tag{18}$$

3. Output g^{J,ν_J}.

Neuron Activation Function



$\sigma = \mathrm{H}(x)$



$$\sigma = \frac{1}{1 + e^{-x}}$$

Classic CNN

Classic CNN structure

1. Initialization of inputs: $g^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

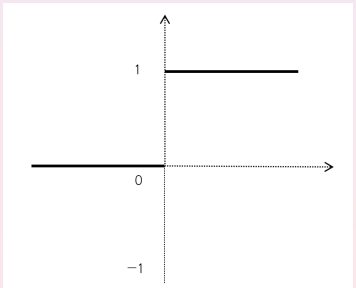
$$g^{\ell,i} = \sigma\left(\theta^{\ell,i} * g^{\ell,i-1}\right) \tag{17}$$

   - Restriction-Pooling

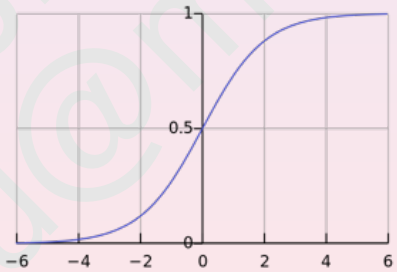
$$g^{\ell+1,0} = R_\ell^{\ell+1} *_2 g^{\ell,\nu_\ell} \tag{18}$$

3. Output g^{J,ν_J}.

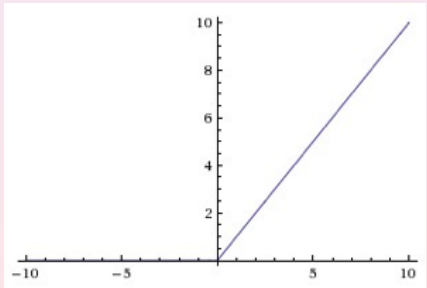
Neuron Activation Function



$\sigma = \mathrm{H}(x)$



$$\sigma = \frac{1}{1 + e^{-x}}$$



$\sigma = \mathrm{ReLU}(x) := max(0, x)$

Two classical CNN examples: LeNet-5 & AlexNet

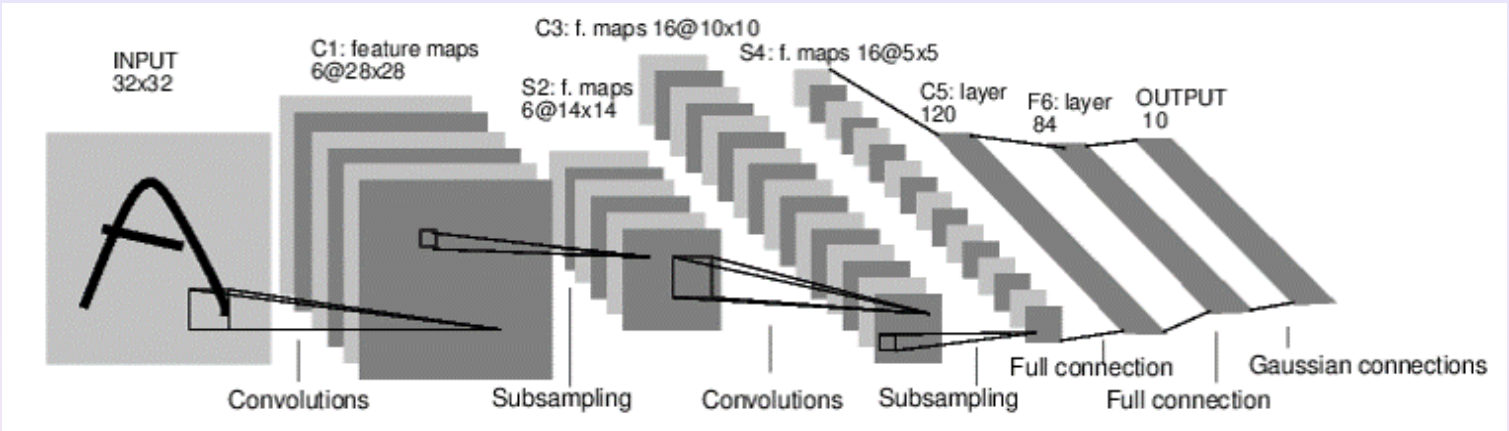


Figure: First successful convolutional neural network: LeNet-5 (Y.LeCun, L.Bottou, Y.Bengio, and P.Haffner, 1998)

Two classical CNN examples: LeNet-5 & AlexNet

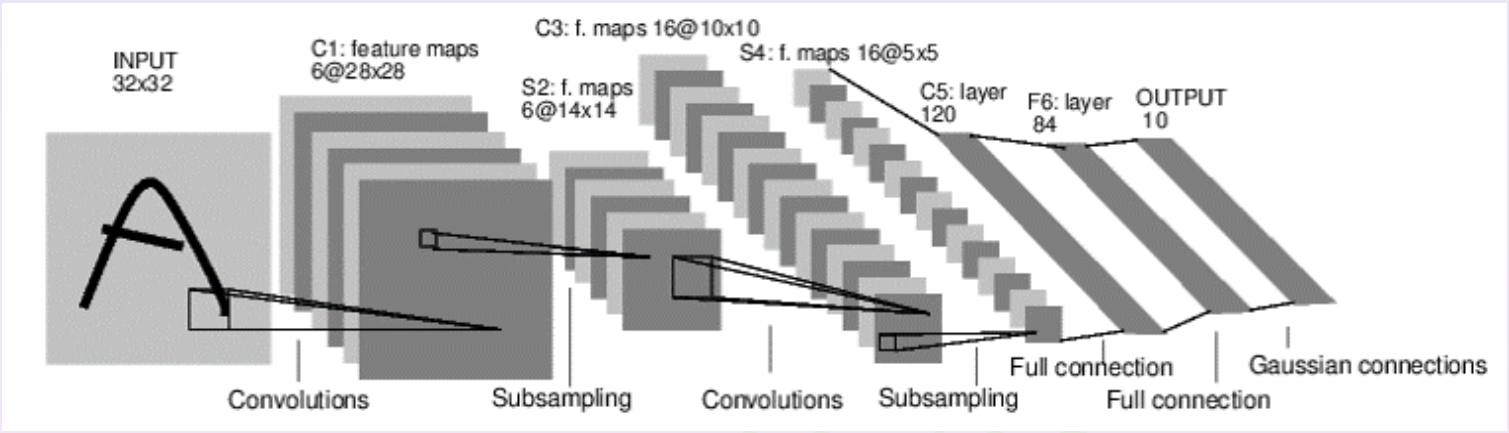


Figure: First successful convolutional neural network: LeNet-5 (Y.LeCun, L.Bottou, Y.Bengio, and P.Haffner, 1998)

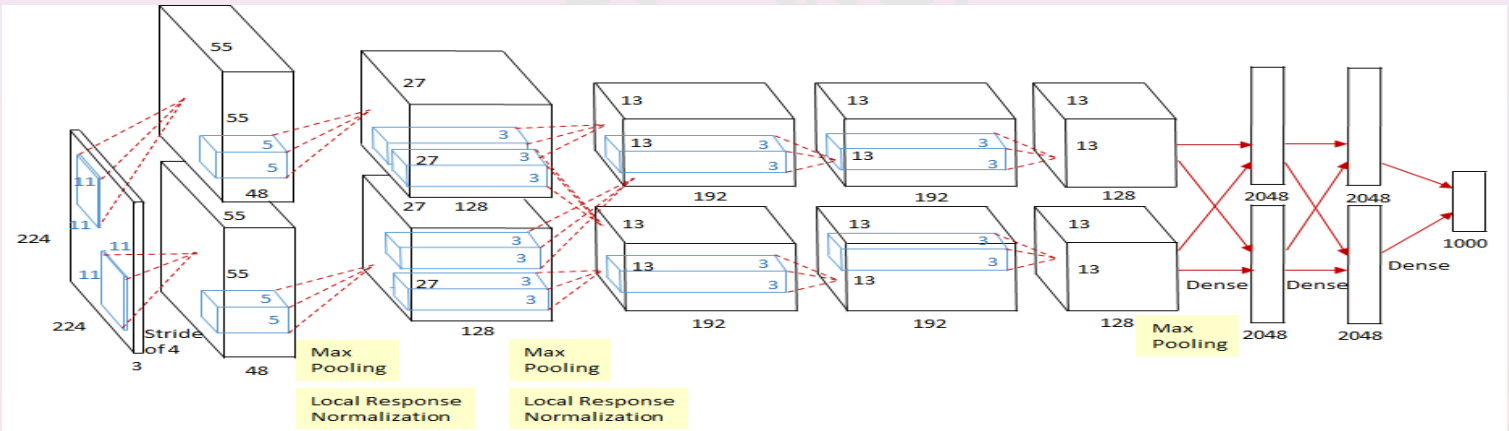


Figure: The beginning of new revolution: AlexNet (A. Krizhevsky, I. Sutskever and G. Hinton, 2012)

ResNet and Pre-act ResNet

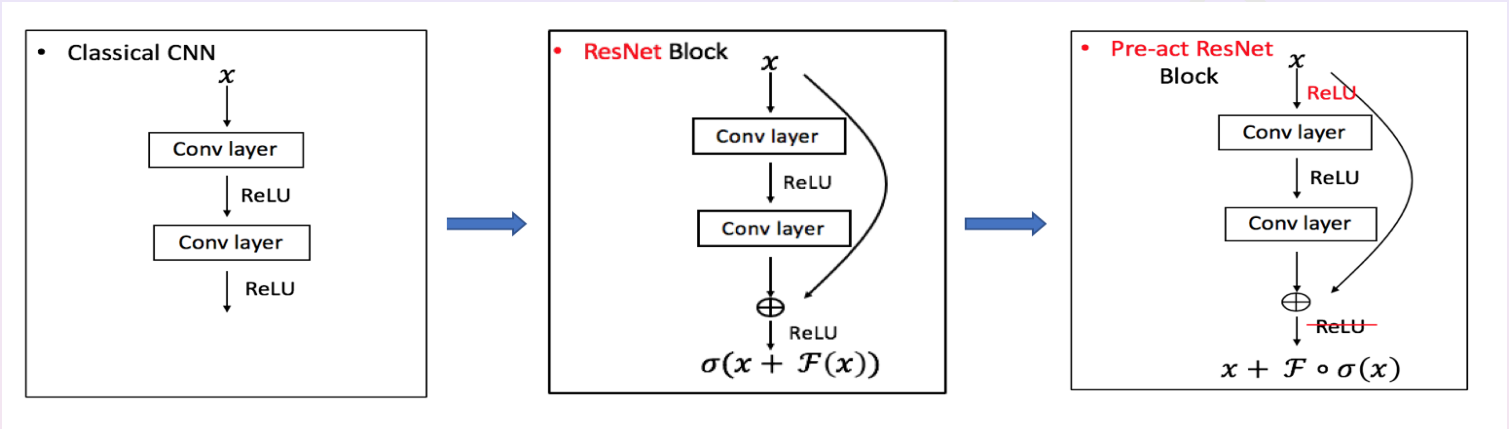


Figure: From classical CNNs to ResNet and Pre-act ResNet (K. He, X. Zhang, S. Ren and J. Sun, 2015 and 2016)

ResNet and Pre-act ResNet

ResNet and Pre-act ResNet

ResNet

1. Initialization of inputs: $r^{1,0}$

ResNet and Pre-act ResNet

ResNet

1. Initialization of inputs: $r^{1,0}$
2. For $\ell = 1 : J$

ResNet and Pre-act ResNet

ResNet

1. Initialization of inputs: $r^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

ResNet and Pre-act ResNet

ResNet

1. Initialization of inputs: $r^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

$$r^{\ell,i} = \sigma\left(r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * r^{\ell,i-1}\right). \tag{19}$$

ResNet and Pre-act ResNet

ResNet

1. Initialization of inputs: $r^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

$$r^{\ell,i} = \sigma\left(r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * r^{\ell,i-1}\right). \tag{19}$$

   - Restriction-Pooling

$$r^{\ell+1,0} = \sigma\left(R_\ell^{\ell+1} *_2 r^{\ell,\nu_\ell} + A^{\ell+1,0} \circ \sigma \circ B^{\ell+1,0} *_2 r^{\ell,\nu_\ell}\right). \tag{20}$$

3. Output g^{J,ν_J}.

Pre-act ResNet

1. Initialization of inputs: $r^{1,0}$

ResNet and Pre-act ResNet

ResNet

1. Initialization of inputs: $r^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

$$r^{\ell,i} = \sigma\left(r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * r^{\ell,i-1}\right). \tag{19}$$

   - Restriction-Pooling

$$r^{\ell+1,0} = \sigma\left(R_\ell^{\ell+1} *_2 r^{\ell,\nu_\ell} + A^{\ell+1,0} \circ \sigma \circ B^{\ell+1,0} *_2 r^{\ell,\nu_\ell}\right). \tag{20}$$

3. Output g^{J,ν_J}.

Pre-act ResNet

1. Initialization of inputs: $r^{1,0}$
2. For $\ell = 1 : J$

ResNet and Pre-act ResNet

ResNet

1. Initialization of inputs: $r^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

$$r^{\ell,i} = \sigma\left(r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * r^{\ell,i-1}\right). \tag{19}$$

   - Restriction-Pooling

$$r^{\ell+1,0} = \sigma\left(R_\ell^{\ell+1} *_2 r^{\ell,\nu_\ell} + A^{\ell+1,0} \circ \sigma \circ B^{\ell+1,0} *_2 r^{\ell,\nu_\ell}\right). \tag{20}$$

3. Output g^{J,ν_J}.

Pre-act ResNet

1. Initialization of inputs: $r^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

ResNet and Pre-act ResNet

ResNet

1. Initialization of inputs: $r^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

$$r^{\ell,i} = \sigma\left(r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * r^{\ell,i-1}\right). \tag{19}$$

   - Restriction-Pooling

$$r^{\ell+1,0} = \sigma\left(R_\ell^{\ell+1} *_2 r^{\ell,\nu_\ell} + A^{\ell+1,0} \circ \sigma \circ B^{\ell+1,0} *_2 r^{\ell,\nu_\ell}\right). \tag{20}$$

3. Output g^{J,ν_J}.

Pre-act ResNet

1. Initialization of inputs: $r^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

$$r^{\ell,i} = r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * \sigma(r^{\ell,i-1}). \tag{21}$$

ResNet and Pre-act ResNet

ResNet

1. Initialization of inputs: $r^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

$$r^{\ell,i} = \sigma\left(r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * r^{\ell,i-1}\right). \tag{19}$$

   - Restriction-Pooling

$$r^{\ell+1,0} = \sigma\left(R_\ell^{\ell+1} *_2 r^{\ell,\nu_\ell} + A^{\ell+1,0} \circ \sigma \circ B^{\ell+1,0} *_2 r^{\ell,\nu_\ell}\right). \tag{20}$$

3. Output g^{J,ν_J}.

Pre-act ResNet

1. Initialization of inputs: $r^{1,0}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

$$r^{\ell,i} = r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * \sigma(r^{\ell,i-1}). \tag{21}$$

   - Restriction-Pooling

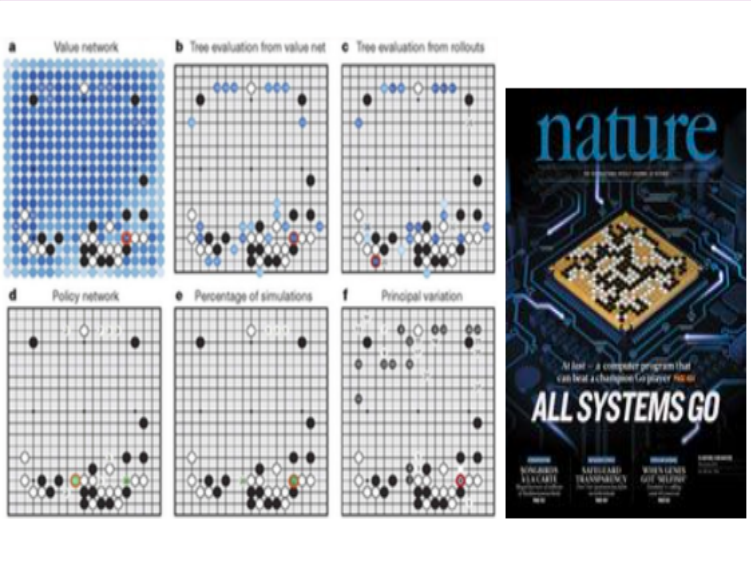
$$r^{\ell+1,0} = R_\ell^{\ell+1} *_2 r^{\ell,\nu_\ell} + A^{\ell+1,0} \circ \sigma \circ B^{\ell+1,0} *_2 \sigma(r^{\ell,\nu_\ell}). \tag{22}$$

3. Output g^{J,ν_J}.

Applications of CNN

CNN has been successfully used in:

- Computer vision
  - Classification, detection, segmentation...
  - Medical image processing,
  - Face recognition,
- Reinforcement learning
  - AlphaGo,
  - Automated driving,
- Natural language processing
  - Speech recognition,
  - Machine translation,
- ...



Images versus Features

Images versus Features

Data space: g

Feature space: u

Images versus Features

Data space: g

Feature extraction $\longrightarrow$

Feature space: u

Images versus Features

Data space: g

Feature extraction $\longrightarrow$

Feature space: u

Images versus Features

Data space: g

Feature extraction $\longrightarrow$

Feature space: u

Images versus Features

Data space: g

Feature extraction $\longrightarrow$

Feature extraction $\longrightarrow$

Feature space: u

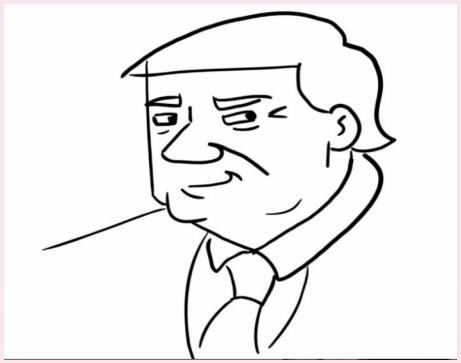
Images versus Features

Data space: g

Feature space: u

Feature extraction $\longrightarrow$

Feature extraction $\longrightarrow$



Mathematical understanding of feature extraction

A linear model: given an image g, find its feature u satisfying

$$A * u = g \tag{23}$$

with a constraint

$$u \geq 0. \tag{24}$$

Mathematical understanding of feature extraction

A linear model: given an image g, find its feature u satisfying

$$A * u = g \tag{23}$$

with a constraint

$$u \geq 0. \tag{24}$$

Questions:

1. What is A?

Mathematical understanding of feature extraction

A linear model: given an image g, find its feature u satisfying

$$A * u = g \tag{23}$$

with a constraint

$$u \geq 0. \tag{24}$$

Questions:

1. What is A? (to be trained ...)
2. How to solve (23)?

Mathematical understanding of feature extraction

A linear model: given an image g, find its feature u satisfying

$$A * u = g \tag{23}$$

with a constraint

$$u \geq 0. \tag{24}$$

Questions:

1. What is A? (to be trained ...)
2. How to solve (23)? (iterative methods)

Iterative methods for $Au = g$: residual correction

$$u^0, u^1, \ldots, u^{k-1} \longrightarrow u^k$$

Basic ideas:

Iterative methods for $Au = g$: residual correction

$$u^0, u^1, \ldots, u^{k-1} \longrightarrow u^k$$

Basic ideas:

1. Form the residual: $r = g - Au^{k-1}$

Iterative methods for $Au = g$: residual correction

$$u^0, u^1, \ldots, u^{k-1} \longrightarrow u^k$$

Basic ideas:

1. Form the residual: $r = g - Au^{k-1}$
2. Solve the residual eqn $Ae = r$ approximately $\hat{e} = Br$ with $B \approx A^{-1}$

Iterative methods for $Au = g$: residual correction

$$u^0, u^1, \ldots, u^{k-1} \longrightarrow u^k$$

Basic ideas:

1. Form the residual: $r = g - Au^{k-1}$
2. Solve the residual eqn $Ae = r$ approximately $\hat{e} = Br$ with $B \approx A^{-1}$
3. Update $u^i = u^{i-1} + \hat{e}$

Iterative methods for $Au = g$: residual correction

$$u^0, u^1, \ldots, u^{k-1} \longrightarrow u^k$$

Basic ideas:

1. Form the residual: $r = g - Au^{k-1}$
2. Solve the residual eqn $Ae = r$ approximately $\hat{e} = Br$ with $B \approx A^{-1}$
3. Update $u^i = u^{i-1} + \hat{e}$

A basic iterative method:

$$u^i = u^{i-1} + B^i(g - Au^{i-1}) \tag{25}$$

Iterative methods for $Au = g$: residual correction

$$u^0, u^1, \ldots, u^{k-1} \longrightarrow u^k$$

Basic ideas:

1. Form the residual: $r = g - Au^{k-1}$
2. Solve the residual eqn $Ae = r$ approximately $\hat{e} = Br$ with $B \approx A^{-1}$
3. Update $u^i = u^{i-1} + \hat{e}$

A basic iterative method:

$$u^i = u^{i-1} + B^i(g - Au^{i-1}) \tag{25}$$

An example: $A = D + L + U$ with $D = \mathrm{DIAG}(A)$

Iterative methods for $Au = g$: residual correction

$$u^0, u^1, \ldots, u^{k-1} \longrightarrow u^k$$

Basic ideas:

1. Form the residual: $r = g - Au^{k-1}$
2. Solve the residual eqn $Ae = r$ approximately $\hat{e} = Br$ with $B \approx A^{-1}$
3. Update $u^i = u^{i-1} + \hat{e}$

A basic iterative method:

$$u^i = u^{i-1} + B^i(g - Au^{i-1}) \tag{25}$$

An example: $A = D + L + U$ with $D = \mathrm{DIAG}(A)$

1. $B = \mathrm{DIAG}(A)^{-1}$

Iterative methods for $Au = g$: residual correction

$$u^0, u^1, \ldots, u^{k-1} \longrightarrow u^k$$

Basic ideas:

1. Form the residual: $r = g - Au^{k-1}$
2. Solve the residual eqn $Ae = r$ approximately $\hat{e} = Br$ with $B \approx A^{-1}$
3. Update $u^i = u^{i-1} + \hat{e}$

A basic iterative method:

$$u^i = u^{i-1} + B^i(g - Au^{i-1}) \tag{25}$$

An example: $A = D + L + U$ with $D = \mathrm{DIAG}(A)$

1. $B = \mathrm{DIAG}(A)^{-1}$ (Jacobi),

Iterative methods for $Au = g$: residual correction

$$u^0, u^1, \ldots, u^{k-1} \longrightarrow u^k$$

Basic ideas:

1. Form the residual: $r = g - Au^{k-1}$
2. Solve the residual eqn $Ae = r$ approximately $\hat{e} = Br$ with $B \approx A^{-1}$
3. Update $u^i = u^{i-1} + \hat{e}$

A basic iterative method:

$$u^i = u^{i-1} + B^i(g - Au^{i-1}) \tag{25}$$

An example: $A = D + L + U$ with $D = \mathrm{DIAG}(A)$

1. $B = \mathrm{DIAG}(A)^{-1}$ (Jacobi),
2. $B = \mathrm{TRIL}(A)^{-1}$

Iterative methods for $Au = g$: residual correction

$$u^0, u^1, \ldots, u^{k-1} \longrightarrow u^k$$

Basic ideas:

1. Form the residual: $r = g - Au^{k-1}$
2. Solve the residual eqn $Ae = r$ approximately $\hat{e} = Br$ with $B \approx A^{-1}$
3. Update $u^i = u^{i-1} + \hat{e}$

A basic iterative method:

$$u^i = u^{i-1} + B^i(g - Au^{i-1}) \tag{25}$$

An example: $A = D + L + U$ with $D = \mathrm{DIAG}(A)$

1. $B = \mathrm{DIAG}(A)^{-1}$ (Jacobi),
2. $B = \mathrm{TRIL}(A)^{-1}$ (Gauss-Seidel)

Iterative schemes for the constrained linear model

Iterative schemes for the constrained linear model

1. Recall iterative methods without constraint

$$u^i = u^{i-1} + B^i * (g - A * u^{i-1})$$

Iterative schemes for the constrained linear model

1. Recall iterative methods without constraint

$$u^i = u^{i-1} + B^i * (g - A * u^{i-1})$$

2. Image classification [$\sigma = \mathrm{ReLU}$: dropping the negative values]

$$u^i = u^{i-1} + \sigma * B^i * \sigma(g - A * u^{i-1})$$

or, in terms of residual

$$r^i = (I - A * \sigma * B^i * \sigma) r^{i-1}.$$

Numerical Examples for Constraint Linear Models

Model	CIFAR10	CIFAR100	# Parameters
ResNet18	93.45	74.45	11M
ResNet18-A^{ℓ}-$B^{\ell,i}$	**93.54**	**74.46**	8.1M
pre-act ResNet18	93.75	74.33	11M
pre-act ResNet18-A^{ℓ}-$B^{\ell,i}$	**93.83**	**74.51**	8.1M
ResNet34	94.71	77.20	21M
ResNet34-A^{ℓ}-$B^{\ell,i}$	**94.84**	**77.24**	13M
pre-act ResNet34	94.76	77.25	21M
pre-act ResNet34-A^{ℓ}-$B^{\ell,i}$	**94.84**	**77.40**	13M

Table: The accuracy and number of parameters for ResNet, pre-act ResNet and their variants of modified versions on CIFAR10 and CIFAR100.

Comments:

Comments:

1. The simple constrained linear model seems adequate for image classifications

Comments:

1. The simple constrained linear model seems adequate for image classifications
2. Feature extractions can be obtained using standard iterative methods

Comments:

1. The simple constrained linear model seems adequate for image classifications
2. Feature extractions can be obtained using standard iterative methods
   - providing a validation of constrained linear model

Comments:

1. The simple constrained linear model seems adequate for image classifications
2. Feature extractions can be obtained using standard iterative methods
   - providing a validation of constrained linear model
3. Potentially more efficient models can be developed.

Gradient Descent Method: $Au = g$

- Consider

$$Au = g \Longleftrightarrow \arg\min \frac{1}{2} u^T Au - g^T u$$

Gradient Descent Method: $Au = g$

- Consider

$$Au = g \Longleftrightarrow \arg\min \frac{1}{2} u^T Au - g^T u$$

- Gradient descent method:

$$u^{k+1} = u^k - \eta \nabla E(u^k)$$

Gradient Descent Method: $Au = g$

- Consider

$$Au = g \Longleftrightarrow \arg\min \frac{1}{2} u^T Au - g^T u$$

- Gradient descent method:

$$u^{k+1} = u^k - \eta \nabla E(u^k) = u^k - \eta(Au^k - g)$$

Gradient Descent Method: $Au = g$

- Consider

$$Au = g \Longleftrightarrow \arg\min \frac{1}{2} u^T Au - g^T u$$

- Gradient descent method:

$$u^{k+1} = u^k - \eta \nabla E(u^k) = u^k - \eta(Au^k - g)$$

- Scaled gradient descent method:

$$u^{k+1} = u^k - \eta[\mathrm{diag}(A)]^{-1}(Au^k - g)$$

Gradient Descent Method: $Au = g$

- Consider
$$Au = g \Longleftrightarrow \arg\min \frac{1}{2} u^T Au - g^T u$$
- Gradient descent method:
$$u^{k+1} = u^k - \eta \nabla E(u^k) = u^k - \eta(Au^k - g)$$
- Scaled gradient descent method:
$$u^{k+1} = u^k - \eta[\mathrm{diag}(A)]^{-1}(Au^k - g)$$
- It converges for any symmetric, positive and definite A if $\eta \ll 1$

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	37

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	37
10^{-1}	

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	37
10^{-1}	236

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	37
10^{-1}	236
10^{-2}	

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	37
10^{-1}	236
10^{-2}	1,918

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	37
10^{-1}	236
10^{-2}	1,918
10^{-3}	

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	37
10^{-1}	236
10^{-2}	1,918
10^{-3}	16,115

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	37
10^{-1}	236
10^{-2}	1,918
10^{-3}	16,115
10^{-4}	

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	37
10^{-1}	236
10^{-2}	1,918
10^{-3}	16,115
10^{-4}	130,168

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	37
10^{-1}	236
10^{-2}	1,918
10^{-3}	16,115
10^{-4}	130,168
0 [singular case]	

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	37
10^{-1}	236
10^{-2}	1,918
10^{-3}	16,115
10^{-4}	130,168
0 [singular case]	20

Gradient descent for a nearly singular system

Consider: $(A_0 + \epsilon I)u = g$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply Gradient descent method with $\|Au^k - b\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1	37
10^{-1}	236
10^{-2}	1,918
10^{-3}	16,115
10^{-4}	130,168
0 [singular case]	20

Iterative method usually is OK for singular system, but subtle for nearly singular system!

Ref for semi-definite case: Keller 1965

Remedy of GD: Multilevel methods

$$V_1 = \mathbb{R}^3, \quad V_2 = \text{span}\{p\} \subset V_1$$

1. Initialization of inputs

$$A_1 = A_\epsilon, \quad g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

Remedy of GD: Multilevel methods

$$V_1 = \mathbb{R}^3, \quad V_2 = \mathrm{span}\{p\} \subset V_1$$

1. Initialization of inputs
$$A_1 = A_\epsilon, \quad g_1 \leftarrow g, \quad u_1 \leftarrow \mathrm{random}.$$
2. Iterate:
   1. One step of GD method on V_1
   $$u_1 \leftarrow u_1 + \eta D_\epsilon^{-1}(g_1 - A_1 u_1).$$
   2. Consider $A_1 e_1 = r_1 \equiv g_1 - A_1 u_1$ and "pool" it to V_2 and solve it:
   $$A_2 u_2 = g_2,$$

Remedy of GD: Multilevel methods

$$V_1 = \mathbb{R}^3, \quad V_2 = \text{span}\{p\} \subset V_1$$

1. Initialization of inputs
$$A_1 = A_\epsilon, \quad g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$
2. Iterate:
   1. One step of GD method on V_1
   $$u_1 \leftarrow u_1 + \eta D_\epsilon^{-1}(g_1 - A_1 u_1).$$
   2. Consider $A_1 e_1 = r_1 \equiv g_1 - A_1 u_1$ and "pool" it to V_2 and solve it:
   $$A_2 u_2 = g_2, \quad u_2 = A_2^{-1} g_2$$
   with
   $$A_2 = p^T A_1 p, g_2 = p^T r_1$$
   3. update $u_1 \leftarrow u_1 + p u_2$.

Convergence behavior of the multilevel algorithm

Convergence behavior of the multilevel algorithm

# GD with $\eta = 0.7$		
ϵ	original	normalized expanded
1.	37	
10^{-1}	236	
10^{-2}	1,918	
10^{-3}	16,115	
10^{-4}	130,168	
10^{-5}	$> 1,000,000$	
10^{-9}	$> 1,000,000$	
10^{-10}	21	
0.	20	

Convergence behavior of the multilevel algorithm

# GD with $\eta = 0.7$		
ϵ	original	normalized expanded
1.	37	
10^{-1}	236	
10^{-2}	1,918	
10^{-3}	16,115	
10^{-4}	130,168	
10^{-5}	$> 1,000,000$	
10^{-9}	$> 1,000,000$	
10^{-10}	21	
0.	20	

Convergence behavior of the multilevel algorithm

# GD with $\eta = 0.7$		
ϵ	original	normalized expanded
1.	37	
10^{-1}	236	
10^{-2}	1,918	
10^{-3}	16,115	
10^{-4}	130,168	
10^{-5}	$> 1,000,000$	
10^{-9}	$> 1,000,000$	
10^{-10}	21	
0.	20	

Convergence behavior of the multilevel algorithm

# GD with $\eta = 0.7$		
ϵ	original	normalized expanded
1.	37	13
10^{-1}	236	
10^{-2}	1,918	
10^{-3}	16,115	
10^{-4}	130,168	
10^{-5}	$> 1,000,000$	
10^{-9}	$> 1,000,000$	
10^{-10}	21	
0.	20	

Convergence behavior of the multilevel algorithm

# GD with $\eta = 0.7$		
ϵ	original	normalized expanded
1.	37	13
10^{-1}	236	14
10^{-2}	1,918	
10^{-3}	16,115	
10^{-4}	130,168	
10^{-5}	$>1,000,000$	
10^{-9}	$>1,000,000$	
10^{-10}	21	
0.	20	

Convergence behavior of the multilevel algorithm

# GD with $\eta = 0.7$		
ϵ	original	normalized expanded
1.	37	13
10^{-1}	236	14
10^{-2}	1,918	14
10^{-3}	16,115	
10^{-4}	130,168	
10^{-5}	$> 1,000,000$	
10^{-9}	$> 1,000,000$	
10^{-10}	21	
0.	20	

Convergence behavior of the multilevel algorithm

# GD with $\eta = 0.7$		
ϵ	original	normalized expanded
1.	37	13
10^{-1}	236	14
10^{-2}	1,918	14
10^{-3}	16,115	16
10^{-4}	130,168	
10^{-5}	$> 1,000,000$	
10^{-9}	$> 1,000,000$	
10^{-10}	21	
0.	20	

Convergence behavior of the multilevel algorithm

# GD with $\eta = 0.7$		
ϵ	original	normalized expanded
1.	37	13
10^{-1}	236	14
10^{-2}	1,918	14
10^{-3}	16,115	16
10^{-4}	130,168	16
10^{-5}	$> 1,000,000$	16
10^{-9}	$> 1,000,000$	15
10^{-10}	21	15
0.	20	

Convergence behavior of the multilevel algorithm

# GD with $\eta = 0.7$		
ϵ	original	normalized expanded
1.	37	13
10^{-1}	236	14
10^{-2}	1,918	14
10^{-3}	16,115	16
10^{-4}	130,168	16
10^{-5}	$> 1,000,000$	16
10^{-9}	$> 1,000,000$	15
10^{-10}	21	15
0.	20	

Variational formulation for 1D Poisson equation

Variational formulation for 1D Poisson equation

- 1D Poisson equation on $\Omega = (0, 1)$

$$-u'' = f, \quad u(0) = u(1) = 0.$$

Variational formulation for 1D Poisson equation

- 1D Poisson equation on $\Omega = (0, 1)$

$$-u'' = f, \quad u(0) = u(1) = 0.$$

- Variational principle:

$$-u'' - f = 0 \quad \Longleftrightarrow \int_0^1 (-u'' - f)v \, dx = 0$$

for all "good" test functions v.

Variational formulation for 1D Poisson equation

- 1D Poisson equation on $\Omega = (0, 1)$

$$-u'' = f, \quad u(0) = u(1) = 0.$$

- Variational principle:

$$-u'' - f = 0 \quad \Longleftrightarrow \int_0^1 (-u'' - f)v \, dx = 0$$

for all "good" test functions v.

1. Function space: $V = \{v \text{ is piecewise smooth and continuous }, v(0) = v(1) = 0\}$.

Variational formulation for 1D Poisson equation

- 1D Poisson equation on $\Omega = (0, 1)$

$$-u'' = f, \quad u(0) = u(1) = 0.$$

- Variational principle:

$$-u'' - f = 0 \quad \Longleftrightarrow \int_0^1 (-u'' - f)v \, dx = 0$$

for all "good" test functions v.

1. Function space: $V = \{v \text{ is piecewise smooth and continuous }, v(0) = v(1) = 0\}$.
2. Integration by parts: for any $v \in V$

$$\int_0^1 -u''v \, dx = \int_0^1 u'v' \, dx - u'v|_0^1 = \int_0^1 u'v' \, dx.$$

Variational formulation for 1D Poisson equation

- 1D Poisson equation on $\Omega = (0,1)$

$$-u'' = f, \quad u(0) = u(1) = 0.$$

- Variational principle:

$$-u'' - f = 0 \quad \Longleftrightarrow \int_0^1 (-u'' - f)v \, dx = 0$$

for all "good" test functions v.

1. Function space: $V = \{v \text{ is piecewise smooth and continuous }, v(0) = v(1) = 0\}$.
2. Integration by parts: for any $v \in V$

$$\int_0^1 -u''v \, dx = \int_0^1 u'v' \, dx - u'v|_0^1 = \int_0^1 u'v' \, dx.$$

3. Variational formulation: Find $u \in V$, such that

$$\int_0^1 u'v' \, dx = \int_0^1 fv \, dx \quad \forall v \in V.$$

Galerkin method

Galerkin method

- Galerkin method: Find $u_h \in V_h$ such that

$$\int_0^1 u_h' v_h' \, dx = \int_0^1 f v_h \, dx \quad \forall v_h \in V_h.$$

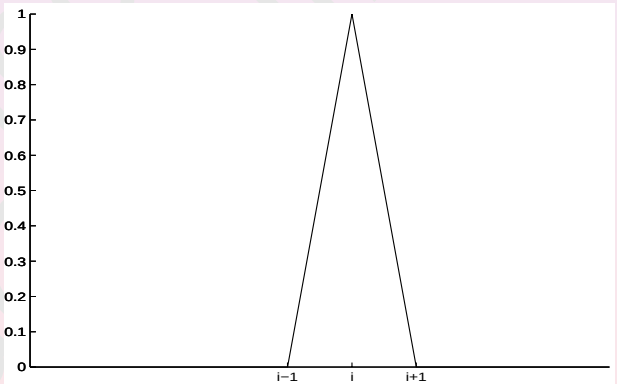
Galerkin method

- Galerkin method: Find $u_h \in V_h$ such that

$$\int_0^1 u_h' v_h' \, dx = \int_0^1 f v_h \, dx \quad \forall v_h \in V_h.$$

- Nodal basis: $\varphi_i(x_j) = \delta_{ij}, u_h = \sum_{i=1}^{N} u_i \varphi_i(x)$,

$$\varphi_i(x) = \begin{cases} \frac{x - x_{i-1}}{h}, & x \in [x_{i-1}, x_i]; \\ \frac{x_{i+1} - x}{h}, & x \in [x_i, x_{i+1}]; \\ 0 & \text{elsewhere.} \end{cases}$$



1D linear system on uniform grid

1D linear system on uniform grid

- Stiffness matrix $A_{ij} = \int_0^1 \varphi_j' \varphi_i' \, dx$, $g_i = \int_0^1 f \varphi_i \, dx$

$$A = \frac{1}{h} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \quad g = h \begin{pmatrix} g(x_1) \\ g(x_2) \\ \vdots \\ g(x_{N-1}) \\ g(x_N) \end{pmatrix} + \mathcal{O}(h^3).$$

or

$$\frac{1}{h}(-u_{i-1} + 2u_i - u_{i+1}) = g_i, \quad 1 \le i \le n$$

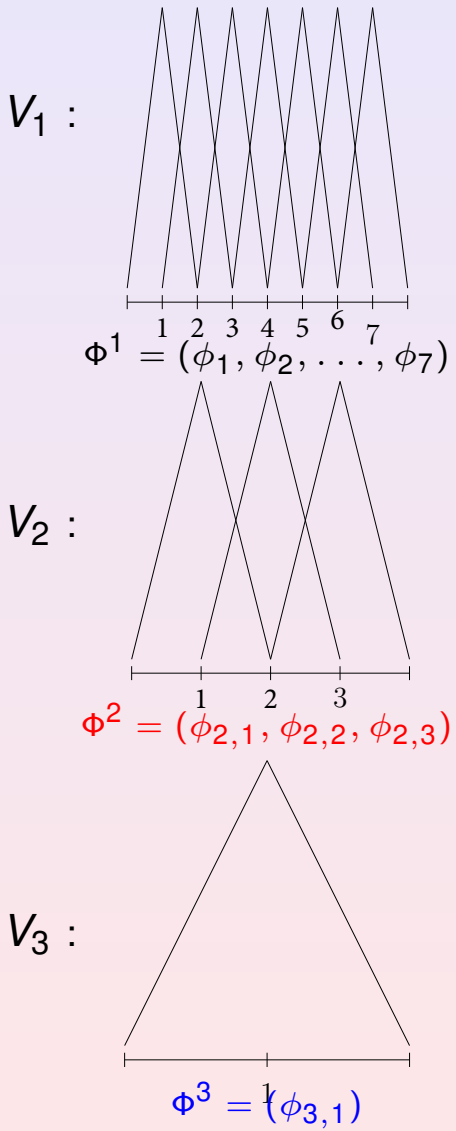
Namely

$$A * u = g$$

where

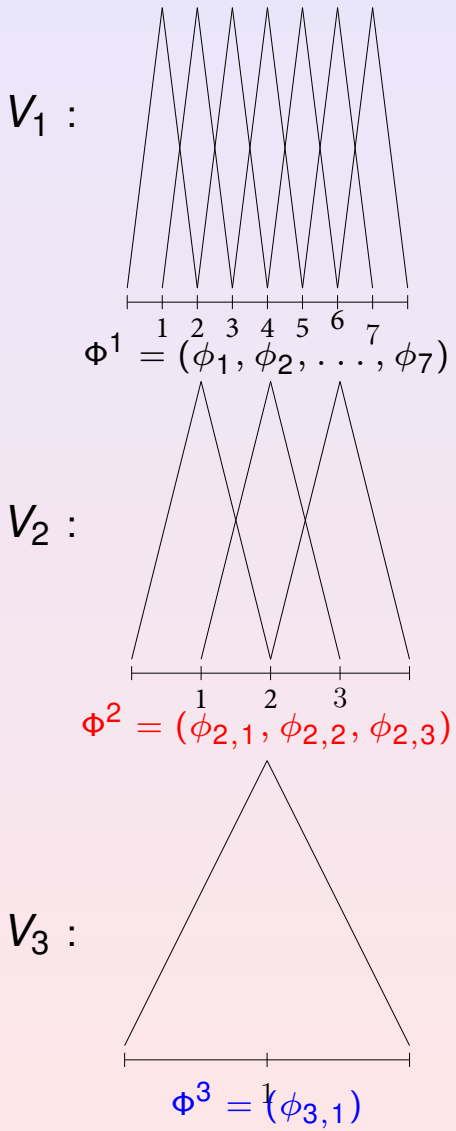
$$A = (-1, 2, -1)/h.$$

Multilevel spaces in 1D

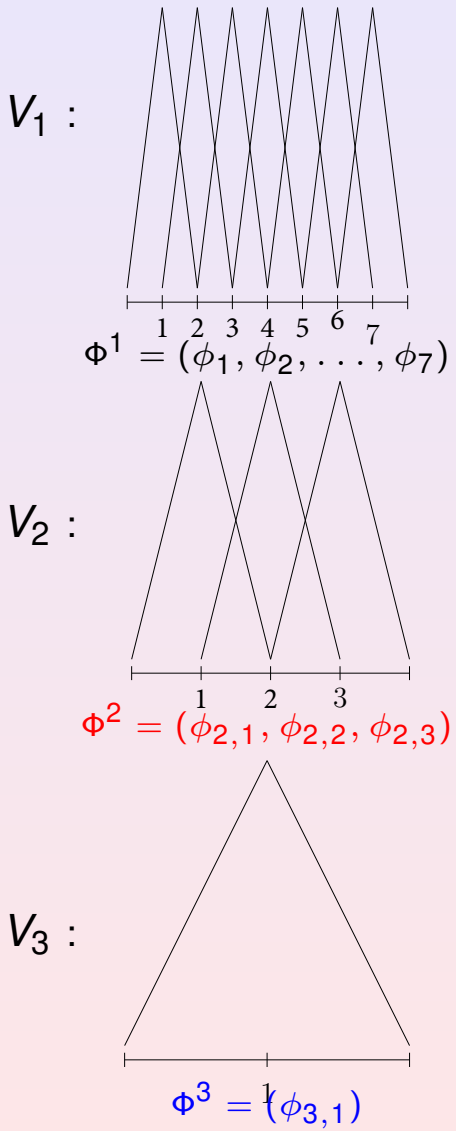


Multilevel spaces in 1D

$$A = 8 \times \text{tridiag}(-1, 2, -1) = \left((\phi_i', \phi_j')\right) \in \mathbb{R}^{7\times 7}$$



Multilevel spaces in 1D

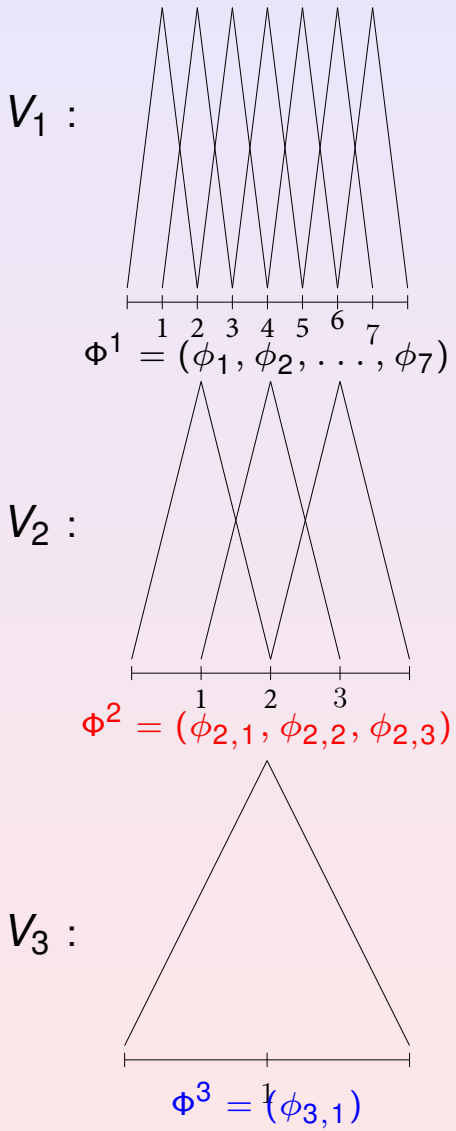


$$A = 8 \times \text{tridiag}(-1, 2, -1) = \left((\phi_i', \phi_j')\right) \in \mathbb{R}^{7\times 7}$$

Note

$$\phi_{3,1} = \frac{1}{2}\phi_{2,1} + \phi_{2,2} + \frac{1}{2}\phi_{2,3}$$

Multilevel spaces in 1D



$$A = 8 \times \text{tridiag}(-1, 2, -1) = \left((\phi_i', \phi_j')\right) \in \mathbb{R}^{7\times 7}$$

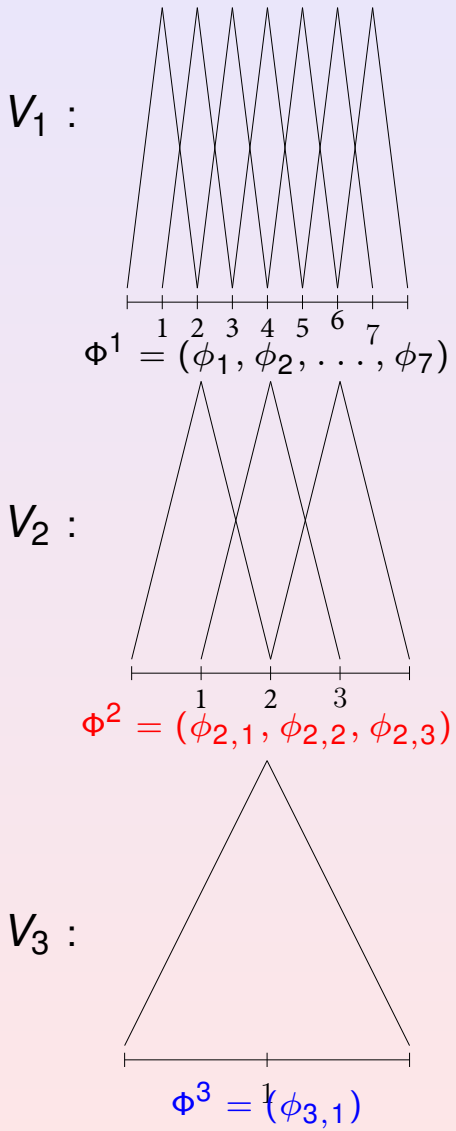
Note

$$\phi_{3,1} = \frac{1}{2}\phi_{2,1} + \phi_{2,2} + \frac{1}{2}\phi_{2,3}$$

This can be written as

$$\Phi^3 = \Phi^2 R$$

Multilevel spaces in 1D



$$A = 8 \times \text{tridiag}(-1, 2, -1) = \left((\phi_i', \phi_j')\right) \in \mathbb{R}^{7\times 7}$$

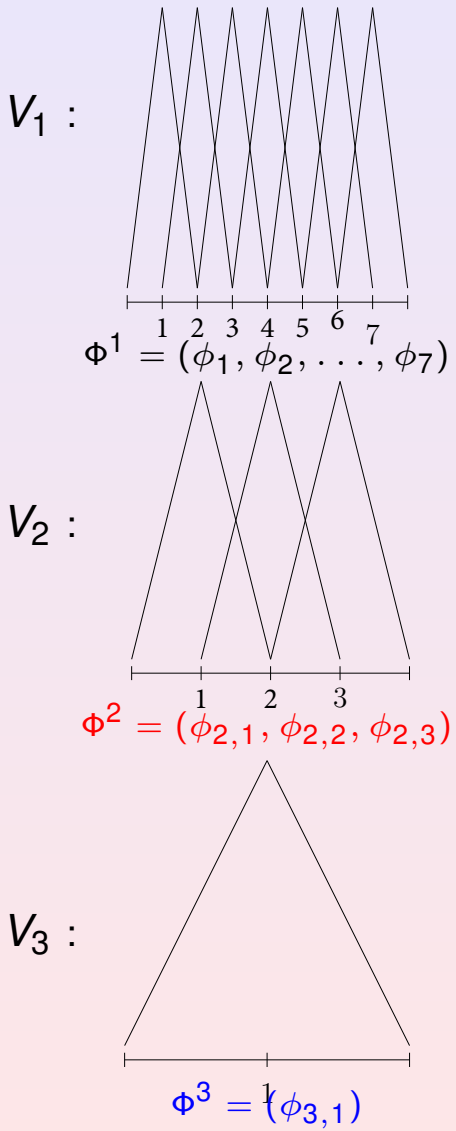
Note

$$\phi_{3,1} = \frac{1}{2}\phi_{2,1} + \phi_{2,2} + \frac{1}{2}\phi_{2,3}$$

This can be written as

$$\Phi^3 = \Phi^2 R = R *_2 \Phi^2$$

Multilevel spaces in 1D



$$A = 8 \times \text{tridiag}(-1, 2, -1) = \left((\phi_i', \phi_j')\right) \in \mathbb{R}^{7\times 7}$$

Note

$$\phi_{3,1} = \frac{1}{2}\phi_{2,1} + \phi_{2,2} + \frac{1}{2}\phi_{2,3}$$

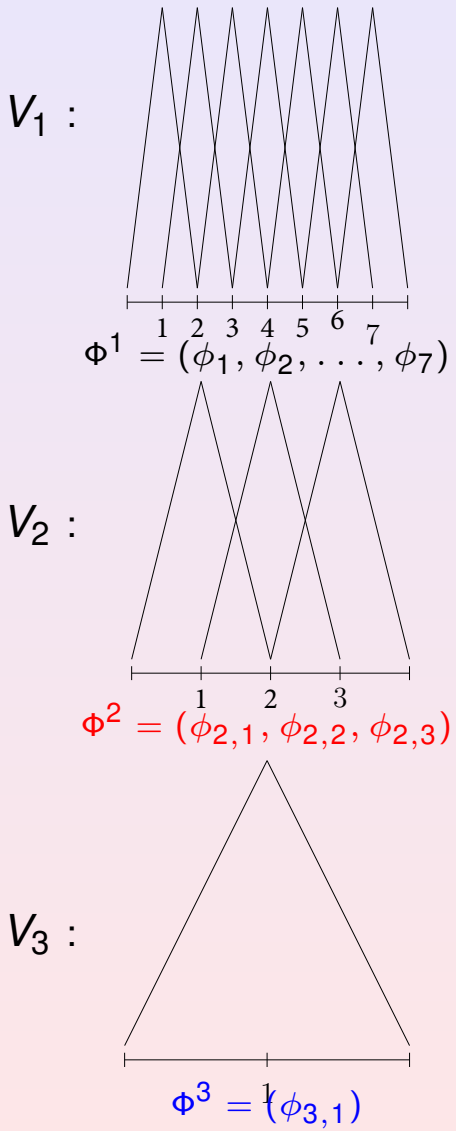
This can be written as

$$\Phi^3 = \Phi^2 R = R *_2 \Phi^2$$

Similarly

$$\Phi^2 = \Phi^1 P$$

Multilevel spaces in 1D



$$A = 8 \times \text{tridiag}(-1, 2, -1) = \left((\phi_i', \phi_j')\right) \in \mathbb{R}^{7\times 7}$$

Note

$$\phi_{3,1} = \frac{1}{2}\phi_{2,1} + \phi_{2,2} + \frac{1}{2}\phi_{2,3}$$

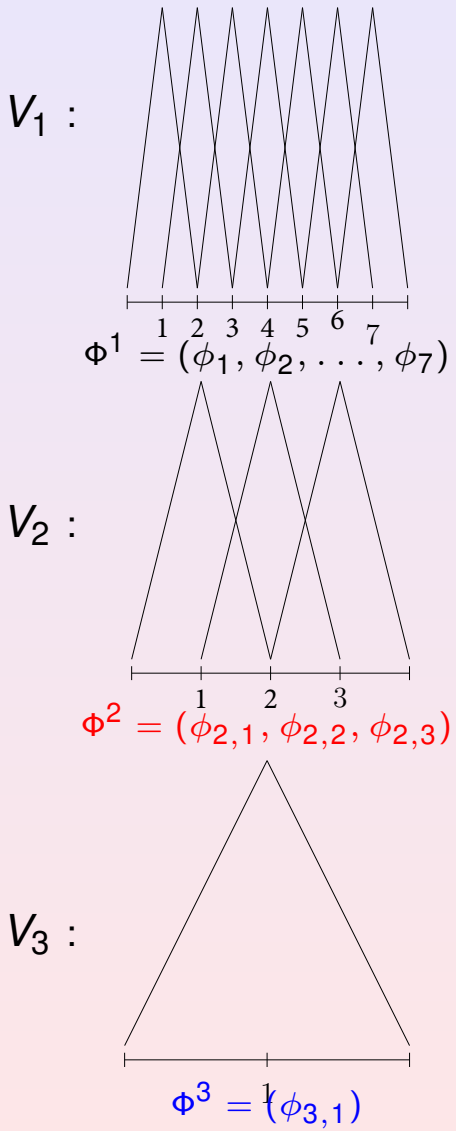
This can be written as

$$\Phi^3 = \Phi^2 R = R *_2 \Phi^2$$

Similarly

$$\Phi^2 = \Phi^1 P = R *_2 \Phi^1$$

Multilevel spaces in 1D



$$A = 8 \times \text{tridiag}(-1, 2, -1) = \left((\phi_i', \phi_j')\right) \in \mathbb{R}^{7\times 7}$$

Note

$$\phi_{3,1} = \frac{1}{2}\phi_{2,1} + \phi_{2,2} + \frac{1}{2}\phi_{2,3}$$

This can be written as

$$\Phi^3 = \Phi^2 R = R *_2 \Phi^2$$

Similarly

$$\Phi^2 = \Phi^1 P = R *_2 \Phi^1$$

where

$$R = \begin{pmatrix} \frac{1}{2} \\ 1 \\ \frac{1}{2} \end{pmatrix}, \quad P = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

Two grid method for 1D case

$$V_1 = \text{span}\{\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6, \phi_7\}, \quad V_2 = \text{span}\{\phi_{2,1}, \phi_{2,2}, \phi_{2,3}\} \subset V_1$$

1. Initialization of inputs

$$A_1 = A, \quad g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

Two grid method for 1D case

$$V_1 = \text{span}\{\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6, \phi_7\}, \quad V_2 = \text{span}\{\phi_{2,1}, \phi_{2,2}, \phi_{2,3}\} \subset V_1$$

1. Initialization of inputs
$$A_1 = A, \quad g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$
2. Iterate:
   1. A few steps of GD method on V_1
   $$u_1 \leftarrow u_1 + \eta(g_1 - A_1 u_1).$$
   2. Consider $A_1 e_1 = r_1 \equiv g_1 - A_1 u_1$ and "pool" it to V_2 and solve it:
   $$A_2 u_2 = g_2,$$

Two grid method for 1D case

$$V_1 = \text{span}\{\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6, \phi_7\}, \quad V_2 = \text{span}\{\phi_{2,1}, \phi_{2,2}, \phi_{2,3}\} \subset V_1$$

1. Initialization of inputs
$$A_1 = A, \quad g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$
2. Iterate:
   1. A few steps of GD method on V_1
   $$u_1 \leftarrow u_1 + \eta(g_1 - A_1 u_1).$$
   2. Consider $A_1 e_1 = r_1 \equiv g_1 - A_1 u_1$ and "pool" it to V_2 and solve it:
   $$A_2 u_2 = g_2, \quad u_2 = A_2^{-1} g_2$$
   with
   $$A_2 = P^T A_1 P, \quad g_2 = P^T r_1$$
   3. update $u_1 \leftarrow u_1 + P u_2$.

Piecewise (bi-)linear functions on multilevel grids

Images:

Piecewise (bi-)linear functions on multilevel grids

Images:

- Piecewise (bi-)linear functions

Piecewise (bi-)linear functions on multilevel grids

Images:

- Piecewise (bi-)linear functions
- Initial grid $\mathcal{T}$ with size:

$$m = 2^s + 1, n = 2^t + 1$$

Piecewise (bi-)linear functions on multilevel grids

Images:

- Piecewise (bi-)linear functions
- Initial grid $\mathcal{T}$ with size:

$$m = 2^s + 1, n = 2^t + 1$$

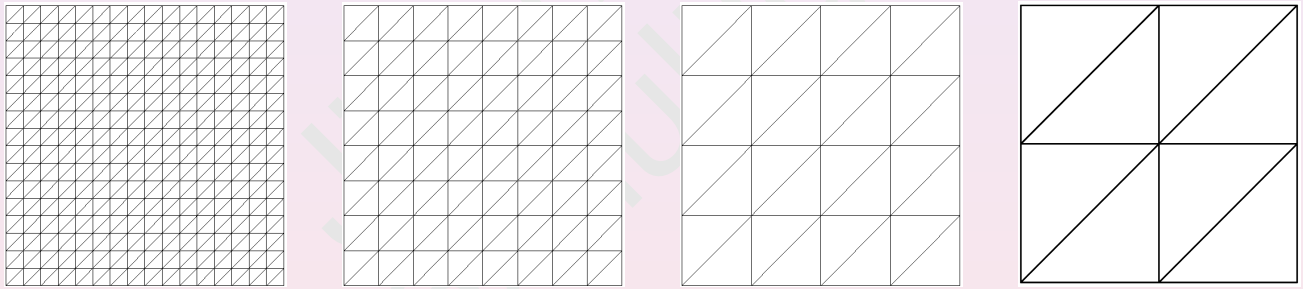
Piecewise (bi-)linear functions on multilevel grids

Images:

- Piecewise (bi-)linear functions
- Initial grid $\mathcal{T}$ with size:

$$m = 2^s + 1, n = 2^t + 1$$

piecewise linear functions on multilevel grids



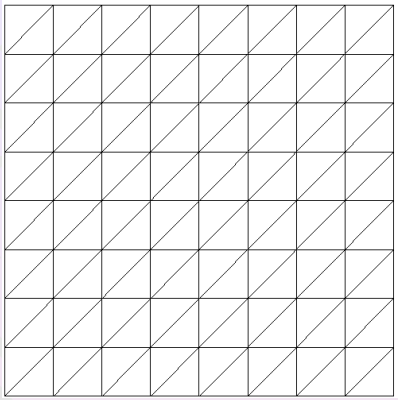
$\mathcal{T}_1$ $\mathcal{T}_2$ $\mathcal{T}_3$ $\mathcal{T}_4$

2D linear system on uniform grid

2D linear system on uniform grid

- Model Problem:

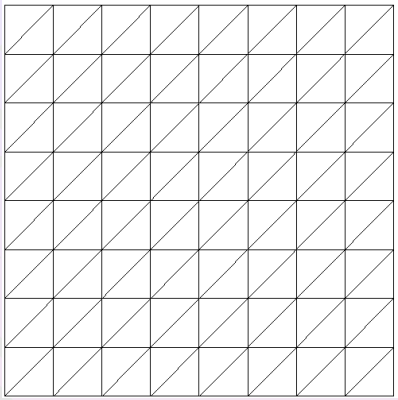
$$-\Delta u := -(u_{xx} + u_{yy}) = g, \text{ in } \Omega,$$
$$u = 0 \text{ on } \partial\Omega, \quad \Omega = (0,1)^d.$$



2D linear system on uniform grid

- Model Problem:

$$-\Delta u := -(u_{xx} + u_{yy}) = g, \text{ in } \Omega,$$
$$u = 0 \text{ on } \partial\Omega, \quad \Omega = (0,1)^d.$$

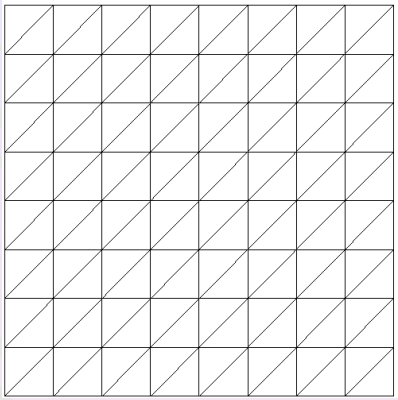


- Discrete case:

2D linear system on uniform grid

- Model Problem:

$$-\Delta u := -(u_{xx} + u_{yy}) = g, \text{ in } \Omega,$$
$$u = 0 \text{ on } \partial\Omega, \quad \Omega = (0,1)^d.$$

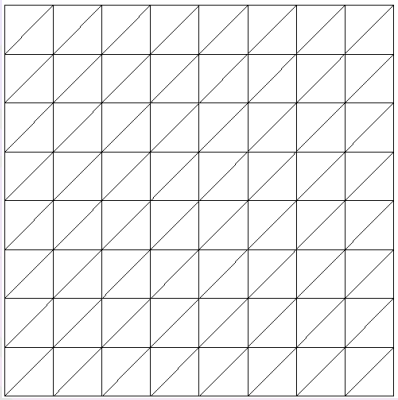


- Discrete case:

2D linear system on uniform grid

- Model Problem:

$$-\Delta u := -(u_{xx} + u_{yy}) = g, \text{ in } \Omega,$$
$$u = 0 \text{ on } \partial\Omega, \quad \Omega = (0,1)^d.$$



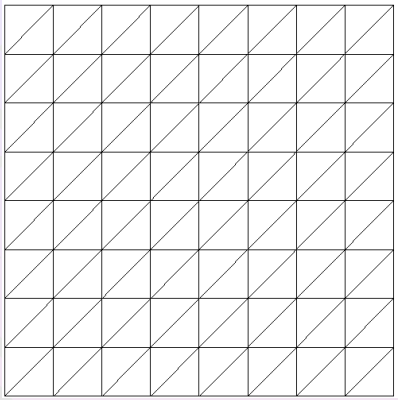
- Discrete case:
  - $d = 2$:

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = g_{i,j}, \tag{26}$$

2D linear system on uniform grid

- Model Problem:

$$-\Delta u := -(u_{xx} + u_{yy}) = g, \text{ in } \Omega,$$
$$u = 0 \text{ on } \partial\Omega, \quad \Omega = (0,1)^d.$$



- Discrete case:
  - $d = 2$:

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = g_{i,j}, \tag{26}$$

with

$$A * u = f, \quad \text{for} \quad A = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}, \tag{27}$$

Smoother and restriction in terms of convolution

Smoother and restriction in terms of convolution

Smoother:

- Gradient descent, say $\eta = 1/8$

$$u^{k+1} = u^k - \eta(A * u^k - g),$$

Smoother and restriction in terms of convolution

Smoother:

- Gradient descent, say $\eta = 1/8$

$$u^{k+1} = u^k - \eta(A * u^k - g),$$

- Apply GD method twice, we get (from 1×1 to 3×3 kernels)

$$u^{k+1} = u^k - S * (A * u^k - g),$$

where

$$S = \frac{1}{64}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 12 & 1 \\ 0 & 1 & 0 \end{pmatrix}. \tag{28}$$

Smoother and restriction in terms of convolution

Smoother:

- Gradient descent, say $\eta = 1/8$

$$u^{k+1} = u^k - \eta(A * u^k - g),$$

- Apply GD method twice, we get (from 1×1 to 3×3 kernels)

$$u^{k+1} = u^k - S * (A * u^k - g),$$

where

$$S = \frac{1}{64}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 12 & 1 \\ 0 & 1 & 0 \end{pmatrix}. \tag{28}$$

Restriction (pooling)

$$R_\ell^{\ell+1} : \mathbb{R}^{m_\ell \times n_\ell} \mapsto \mathbb{R}^{m_{\ell+1} \times n_{\ell+1}} \text{ is } R_\ell^{\ell+1} = R *_2$$

where

$$R = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}, \quad \Phi^{\ell+1} = R *_2 \Phi^\ell. \tag{29}$$

A multigrid method in terms of convolution

A multigrid method in terms of convolution

1. Initialization of inputs

$$g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

A multigrid method in terms of convolution

1. Initialization of inputs

$$g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

2. Smoothing and restriction

A multigrid method in terms of convolution

1. Initialization of inputs

$$g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

2. Smoothing and restriction
   - For $\ell = 1 : J$

A multigrid method in terms of convolution

1. Initialization of inputs

$$g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

2. Smoothing and restriction
   - For $\ell = 1 : J$
     - ★ For $i = 1 : \nu_\ell$

A multigrid method in terms of convolution

1. Initialization of inputs

$$g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

2. Smoothing and restriction
   - For $\ell = 1 : J$
     - ★ For $i = 1 : \nu_\ell$

A multigrid method in terms of convolution

1. Initialization of inputs

$$g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

2. Smoothing and restriction
   - For $\ell = 1 : J$
     - ★ For $i = 1 : \nu_\ell$

$$u_\ell \leftarrow u_\ell + S_\ell * (g_\ell - A_\ell * u_\ell). \tag{30}$$

A multigrid method in terms of convolution

1. Initialization of inputs

$$g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

2. Smoothing and restriction
   - For $\ell = 1 : J$
     - ★ For $i = 1 : \nu_\ell$

$$u_\ell \leftarrow u_\ell + S_\ell * (g_\ell - A_\ell * u_\ell). \tag{30}$$

     - ★ Form restricted residual and set initial guess:

$$u_{\ell+1,0} \leftarrow 0, \quad g_{\ell+1} \leftarrow R_\ell *_2 (g_\ell - A_\ell * u_\ell), \quad A_{\ell+1} = R_\ell *_2 A_\ell * R_\ell *_2^\top .$$

A multigrid method in terms of convolution

1. Initialization of inputs

$$g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

2. Smoothing and restriction
   - For $\ell = 1 : J$
     - ★ For $i = 1 : \nu_\ell$

$$u_\ell \leftarrow u_\ell + S_\ell * (g_\ell - A_\ell * u_\ell). \tag{30}$$

     - ★ Form restricted residual and set initial guess:

$$u_{\ell+1,0} \leftarrow 0, \quad g_{\ell+1} \leftarrow R_\ell *_2 (g_\ell - A_\ell * u_\ell), \quad A_{\ell+1} = R_\ell *_2 A_\ell * R_\ell *_2^\top .$$

A multigrid method in terms of convolution

1. Initialization of inputs

$$g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

2. Smoothing and restriction
   - For $\ell = 1 : J$
     - ★ For $i = 1 : \nu_\ell$

$$u_\ell \leftarrow u_\ell + S_\ell * (g_\ell - A_\ell * u_\ell). \tag{30}$$

     - ★ Form restricted residual and set initial guess:

$$u_{\ell+1,0} \leftarrow 0, \quad g_{\ell+1} \leftarrow R_\ell *_2 (g_\ell - A_\ell * u_\ell), \quad A_{\ell+1} = R_\ell *_2 A_\ell * R_\ell *_2^\top .$$

3. Update with prolongation:

A multigrid method in terms of convolution

1. Initialization of inputs

$$g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

2. Smoothing and restriction
   - For $\ell = 1 : J$
     - ★ For $i = 1 : \nu_\ell$

$$u_\ell \leftarrow u_\ell + S_\ell * (g_\ell - A_\ell * u_\ell). \tag{30}$$

     - ★ Form restricted residual and set initial guess:

$$u_{\ell+1,0} \leftarrow 0, \quad g_{\ell+1} \leftarrow R_\ell *_2 (g_\ell - A_\ell * u_\ell), \quad A_{\ell+1} = R_\ell *_2 A_\ell * R_\ell *_2^\top .$$

3. Update with prolongation: For $\ell = J - 1 : 1$

$$u_\ell \leftarrow u_\ell + R_\ell *_2^\top (u_{\ell+1} - u_{\ell+1,0}).$$

A multigrid method in terms of convolution

1. Initialization of inputs

$$g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

2. Smoothing and restriction
   - For $\ell = 1 : J$
     - ★ For $i = 1 : \nu_\ell$

$$u_\ell \leftarrow u_\ell + S_\ell * (g_\ell - A_\ell * u_\ell). \tag{30}$$

     - ★ Form restricted residual and set initial guess:

$$u_{\ell+1,0} \leftarrow 0, \quad g_{\ell+1} \leftarrow R_\ell *_2 (g_\ell - A_\ell * u_\ell), \quad A_{\ell+1} = R_\ell *_2 A_\ell * R_\ell *_2^\top .$$

3. Update with prolongation: For $\ell = J - 1 : 1$

$$u_\ell \leftarrow u_\ell + R_\ell *_2^\top (u_{\ell+1} - u_{\ell+1,0}).$$

Gauss Elimination versus Multigrid Method

G.E. on the supercomputer: Fugaku(~400 petaFLOPs)

Gauss Elimination versus Multigrid Method

G.E. on the supercomputer: Fugaku(~400 petaFLOPs)

G. E. on Fugaku

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$

General dense matrix

Gauss Elimination versus Multigrid Method

G.E. on the supercomputer: Fugaku(~400 petaFLOPs)

G. E. on Fugaku

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.93s			

General dense matrix

Gauss Elimination versus Multigrid Method

G.E. on the supercomputer: Fugaku(~400 petaFLOPs)

G. E. on Fugaku

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.93s	0.27 hr		

General dense matrix

Gauss Elimination versus Multigrid Method

G.E. on the supercomputer: Fugaku(~400 petaFLOPs)

G. E. on Fugaku

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.93s	0.27 hr	11 days	

General dense matrix

Gauss Elimination versus Multigrid Method

G.E. on the supercomputer: Fugaku(~400 petaFLOPs)

G. E. on Fugaku

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.93s	0.27 hr	11 days	30.1 Y

General dense matrix

Gauss Elimination versus Multigrid Method

G.E. on the supercomputer: Fugaku(~400 petaFLOPs)

G. E. on Fugaku

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.93s	0.27 hr	11 days	30.1 Y

General dense matrix

Multigrid on 1 core

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$

3D Poisson equation

Gauss Elimination versus Multigrid Method

G.E. on the supercomputer: Fugaku(~400 petaFLOPs)

G. E. on Fugaku

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.93s	0.27 hr	11 days	30.1 Y

General dense matrix

Multigrid on 1 core

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.25 s			

3D Poisson equation

Gauss Elimination versus Multigrid Method

G.E. on the supercomputer: Fugaku(~400 petaFLOPs)

G. E. on Fugaku

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.93s	0.27 hr	11 days	30.1 Y

General dense matrix

Multigrid on 1 core

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.25 s	2.5 s		

3D Poisson equation

Gauss Elimination versus Multigrid Method

G.E. on the supercomputer: Fugaku(~400 petaFLOPs)

G. E. on Fugaku

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.93s	0.27 hr	11 days	30.1 Y

General dense matrix

Multigrid on 1 core

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.25 s	2.5 s	25 s	

3D Poisson equation

Gauss Elimination versus Multigrid Method

G.E. on the supercomputer: Fugaku(~400 petaFLOPs)

G. E. on Fugaku

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.93s	0.27 hr	11 days	30.1 Y

General dense matrix

Multigrid on 1 core

$n = 10^6$	$n = 10^7$	$n = 10^8$	$n = 10^9$
0.25 s	2.5 s	25 s	250 s

3D Poisson equation

Comment and Questions

Comment and Questions

- Multigrid is powerful;

Comment and Questions

- Multigrid is powerful;
- Can the power of multigrid be transformed to CNN?

Comment and Questions

- Multigrid is powerful;
- Can the power of multigrid be transformed to CNN?
  1. better structure CNN with fewer weights?

Comment and Questions

- Multigrid is powerful;
- Can the power of multigrid be transformed to CNN?
  1. better structure CNN with fewer weights?
  2. faster training algorithms? [main original goal, making progress]

Data and feature spaces

Data and feature spaces

Partial differential equations

$$-\Delta u = g. \tag{31}$$

Data and feature spaces

Partial differential equations

$$-\Delta u = g. \tag{31}$$

Constrained linear model: Given an image g, find its feature u such that

$$A * u = g. \tag{32}$$

Data and feature spaces

Partial differential equations

$$-\Delta u = g. \tag{31}$$

Constrained linear model: Given an image g, find its feature u such that

$$A * u = g. \tag{32}$$

Main idea:

- Use a geometric multigrid method for PDE (31) to solve the data-feature equation (32)!

MgNet: a “trained” multigrid method

MgNet: a "trained" multigrid method

1. Initialization of inputs: $g_1 \leftarrow \theta * g, \quad u_1 \leftarrow \text{random}$

MgNet: a "trained" multigrid method

1. Initialization of inputs: $g_1 \leftarrow \theta * g, \quad u_1 \leftarrow \text{random}$
2. For $\ell = 1 : J$

MgNet: a "trained" multigrid method

1. Initialization of inputs: $g_1 \leftarrow \theta * g, \quad u_1 \leftarrow \text{random}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

MgNet: a "trained" multigrid method

1. Initialization of inputs: $g_1 \leftarrow \theta * g, \quad u_1 \leftarrow \text{random}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

$$u_\ell \leftarrow u_\ell + \sigma \circ S_\ell * \sigma(g_\ell - A_\ell * u_\ell). \tag{33}$$

MgNet: a "trained" multigrid method

1. Initialization of inputs: $g_1 \leftarrow \theta * g, \quad u_1 \leftarrow \text{random}$
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$

$$u_\ell \leftarrow u_\ell + \sigma \circ S_\ell * \sigma(g_\ell - A_\ell * u_\ell). \tag{33}$$

   - Restriction-Pooling

$$u_{\ell+1,0} \leftarrow \Pi_\ell^{\ell+1} u_\ell, \quad g_{\ell+1} \leftarrow R_\ell *_2 (g_\ell - A_\ell * u_\ell) + A_{\ell+1} * u_{\ell+1}^0,$$

3. Output = feature map :

$$\phi(g) = u_J.$$

MgNet: From multigrid to CNN

Multigrid:

MgNet: From multigrid to CNN

Multigrid:

- A_ℓ, S_ℓ, R_ℓ are all given a priori

CNN:

MgNet: From multigrid to CNN

Multigrid:

- A_ℓ, S_ℓ, R_ℓ are all given a priori

CNN:

- Almost identically same structure as multigrid!

MgNet: From multigrid to CNN

Multigrid:

- A_ℓ, S_ℓ, R_ℓ are all given a priori

CNN:

- Almost identically same structure as multigrid!
- $A_\ell^i, S_\ell^i, R_\ell^i$ are all trained!

MgNet: From multigrid to CNN

Multigrid:

- A_ℓ, S_ℓ, R_ℓ are all given a priori

CNN:

- Almost identically same structure as multigrid!
- $A_\ell^i, S_\ell^i, R_\ell^i$ are all trained!
- Activation, ReLU, is introduced (to drop-off negative pixel values).

MgNet: From multigrid to CNN

Multigrid:

- A_ℓ, S_ℓ, R_ℓ are all given a priori

CNN:

- Almost identically same structure as multigrid!
- $A_\ell^i, S_\ell^i, R_\ell^i$ are all trained!
- Activation, ReLU, is introduced (to drop-off negative pixel values).
- Extra channels are introduced.

MgNet: From multigrid to CNN

Multigrid:

- A_ℓ, S_ℓ, R_ℓ are all given a priori

CNN:

- Almost identically same structure as multigrid!
- $A_\ell^i, S_\ell^i, R_\ell^i$ are all trained!
- Activation, ReLU, is introduced (to drop-off negative pixel values).
- Extra channels are introduced.

MgNet: From multigrid to CNN

Multigrid:

- A_ℓ, S_ℓ, R_ℓ are all given a priori

CNN:

- Almost identically same structure as multigrid!
- $A_\ell^i, S_\ell^i, R_\ell^i$ are all trained!
- Activation, ReLU, is introduced (to drop-off negative pixel values).
- Extra channels are introduced.

CNN versus multigrid: classic approaches versus MgNet

MgNet: From multigrid to CNN

Multigrid:

- A_ℓ, S_ℓ, R_ℓ are all given a priori

CNN:

- Almost identically same structure as multigrid!
- $A_\ell^i, S_\ell^i, R_\ell^i$ are all trained!
- Activation, ReLU, is introduced (to drop-off negative pixel values).
- Extra channels are introduced.

CNN versus multigrid: classic approaches versus MgNet

1. CNN:
   - Classic: Almost all the components are unrelated and need to be trained
   - MgNet: Most of the components are related and can be given a priori
2. Multigrid
   - Classic: Almost all the components are a priori given
   - MgNet: Some of components can be trained!

Approximation properties of MgNet and deep CNNs

Approximation properties of MgNet and deep CNNs

Theorem (He, Li and Xu 2021)

Assume $f : \Omega \subset \mathbb{R}^{d\times d} \mapsto \mathbb{R}$ *and* Ω *is bounded, then there exist one version of MgNet* $\tilde{f} : R^{d\times d} \mapsto \mathbb{R}$ *with multichannel and* 3×3 *kernels in smoothing, where*

$$\begin{aligned} \textit{depth (number of smoothing):} \quad & \leq d, \\ \textit{width (number of channels for } i\textit{-th smoothing):} \quad & \leq 2(2i-1)^2, \quad i = 1 : d-1, \end{aligned} \tag{34}$$

and number of channel for the last layer is N*, such that*

$$\|f - \tilde{f}\|_{L^2(\Omega)} \lesssim N^{-\frac{1}{2}-\frac{3}{2d^2}}. \tag{35}$$

Approximation properties of MgNet and deep CNNs

Theorem (He, Li and Xu 2021)

Assume $f : \Omega \subset \mathbb{R}^{d\times d} \mapsto \mathbb{R}$ *and* Ω *is bounded, then there exist one version of MgNet* $\tilde{f} : R^{d\times d} \mapsto \mathbb{R}$ *with multichannel and* 3×3 *kernels in smoothing, where*

$$\begin{aligned} \textit{depth (number of smoothing):} \quad & \leq d, \\ \textit{width (number of channels for } i\textit{-th smoothing):} \quad & \leq 2(2i-1)^2, \quad i = 1 : d-1, \end{aligned} \tag{34}$$

and number of channel for the last layer is N*, such that*

$$\|f - \tilde{f}\|_{L^2(\Omega)} \lesssim N^{-\frac{1}{2}-\frac{3}{2d^2}}. \tag{35}$$

Main properties:

- For 2D images with channels.
- Similar results hold for classical, ResNet and pre-act ResNet type networks.
- Convolution (the most commonly used type): 3 by 3 kernel with multichannel for both periodic and zero padding.
- Achieve the same asymptotic approximation rate to DNN with one-hidden layer.

Pre-act ResNet: a "residual" version of MgNet

Pre-act ResNet: a "residual" version of MgNet

Theorem (MgNet and pre-act ResNet, He and Xu 2019)

The MgNet model recovers the pre-act ResNet (K. He et al 2016) as follows

$$r^{\ell,i} = r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * \sigma(r^{\ell,i-1}), \quad i = 1 : \nu_\ell, \tag{36}$$

where

$$r^{\ell,i} = g^\ell - A^\ell * u^{\ell,i}.$$

Pre-act ResNet: a "residual" version of MgNet

Theorem (MgNet and pre-act ResNet, He and Xu 2019)

The MgNet model recovers the pre-act ResNet (K. He et al 2016) as follows

$$r^{\ell,i} = r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * \sigma(r^{\ell,i-1}), \quad i = 1 : \nu_\ell, \tag{36}$$

where

$$r^{\ell,i} = g^\ell - A^\ell * u^{\ell,i}.$$

Proof.

$$u^{\ell,i} = u^{\ell,i-1} + \sigma \circ B^{\ell,i} * \sigma(g^\ell - A^\ell * u^{\ell,i-1}),$$

Pre-act ResNet: a "residual" version of MgNet

Theorem (MgNet and pre-act ResNet, He and Xu 2019)

The MgNet model recovers the pre-act ResNet (K. He et al 2016) as follows

$$r^{\ell,i} = r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * \sigma(r^{\ell,i-1}), \quad i = 1 : \nu_\ell, \tag{36}$$

where

$$r^{\ell,i} = g^\ell - A^\ell * u^{\ell,i}.$$

Proof.

$$u^{\ell,i} = u^{\ell,i-1} + \sigma \circ B^{\ell,i} * \sigma(g^\ell - A^\ell * u^{\ell,i-1}),$$
$$\Rightarrow A^\ell * u^{\ell,i} = A^\ell * u^{\ell,i-1} + A^\ell * \sigma \circ B^{\ell,i} * \sigma(g^\ell - A^\ell * u^{\ell,i-1}),$$

Pre-act ResNet: a "residual" version of MgNet

Theorem (MgNet and pre-act ResNet, He and Xu 2019)

The MgNet model recovers the pre-act ResNet (K. He et al 2016) as follows

$$r^{\ell,i} = r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * \sigma(r^{\ell,i-1}), \quad i = 1 : \nu_\ell, \tag{36}$$

where

$$r^{\ell,i} = g^\ell - A^\ell * u^{\ell,i}.$$

Proof.

$$u^{\ell,i} = u^{\ell,i-1} + \sigma \circ B^{\ell,i} * \sigma(g^\ell - A^\ell * u^{\ell,i-1}),$$
$$\Rightarrow A^\ell * u^{\ell,i} = A^\ell * u^{\ell,i-1} + A^\ell * \sigma \circ B^{\ell,i} * \sigma(g^\ell - A^\ell * u^{\ell,i-1}),$$
$$\Rightarrow g^\ell - A^\ell * u^{\ell,i} = g^\ell - A^\ell * u^{\ell,i} - A^\ell * \sigma \circ B^{\ell,i} * \sigma(g^\ell - A^\ell * u^{\ell,i-1}),$$

Pre-act ResNet: a "residual" version of MgNet

Theorem (MgNet and pre-act ResNet, He and Xu 2019)

The MgNet model recovers the pre-act ResNet (K. He et al 2016) as follows

$$r^{\ell,i} = r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * \sigma(r^{\ell,i-1}), \quad i = 1 : \nu_\ell, \tag{36}$$

where

$$r^{\ell,i} = g^\ell - A^\ell * u^{\ell,i}.$$

Proof.

$$\begin{aligned}
u^{\ell,i} &= u^{\ell,i-1} + \sigma \circ B^{\ell,i} * \sigma(g^\ell - A^\ell * u^{\ell,i-1}),\\
\Rightarrow A^\ell * u^{\ell,i} &= A^\ell * u^{\ell,i-1} + A^\ell * \sigma \circ B^{\ell,i} * \sigma(g^\ell - A^\ell * u^{\ell,i-1}),\\
\Rightarrow g^\ell - A^\ell * u^{\ell,i} &= g^\ell - A^\ell * u^{\ell,i} - A^\ell * \sigma \circ B^{\ell,i} * \sigma(g^\ell - A^\ell * u^{\ell,i-1}),\\
\Rightarrow r_\ell^i &= r^{\ell,i} = r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * \sigma(r^{\ell,i-1}).
\end{aligned} \tag{37}$$

□

MgNet versus other CNNs

Model	Accuracy	# Parameters
ResNet18	95.28	11.2M
pre-act ResNet18	95.08	10.2M
MgNet[2,2,2,2],256	96.00	8.2M

Table: The comprison of MgNet and classical CNN on Cifar10

Model	Accuracy	# Parameters
ResNet18	77.54	11.2M
pre-act ResNet18	77.29	11.2M
MgNet[2,2,2,2],256	79.94	8.3M
MgNet[2,2,2,2],512	81.35	33.1M
MgNet[2,2,2,2],1024	82.46	132.2M

Table: The comprison of MgNet and classical CNN on Cifar100

Model	Accuracy	Parameters
ResNet18	72.12	11.2M
MgNet[2,2,2,2], [64,128,256,512]	73.36	13.0M
MgNet[3,4,6,3],[128,256,512,1024]	78.59	71.3M

Table: The comprison of MgNet and classical CNN on ImageNet.

A summary of MgNet

- J. Xu, Deep Neural Networks and Multigrid Methods, (Lecture Notes at Penn State), 2021.
- J. He, J. Xu. MgNet: A Unified Framework of Multigrid and Convolutional Neural Network. Sci China Math, 2019, 62: 1331-1354, https://doi.org/10.1007/s11425-019-9547-2
- J. He, Y. Chen, L. Zhang, J. Xu. Constrained Linear Data-feature Mapping for Image Classification. ArXiv:1911.10428, 2019.
- Y. Chen, B. Dong, J. Xu. Meta-MgNet: Meta Multigrid Networks for Solving Parameterized Partial Differential Equations Image Classification. ArXiv:2010.14088, 2020
- J. He, L. Li, J. Xu. Approximation Properties of Deep ReLU CNNs. ArXiv:2109.00190, 2021.

A summary of MgNet

- J. Xu, Deep Neural Networks and Multigrid Methods, (Lecture Notes at Penn State), 2021.
- J. He, J. Xu. MgNet: A Unified Framework of Multigrid and Convolutional Neural Network. Sci China Math, 2019, 62: 1331-1354, https://doi.org/10.1007/s11425-019-9547-2
- J. He, Y. Chen, L. Zhang, J. Xu. Constrained Linear Data-feature Mapping for Image Classification. ArXiv:1911.10428, 2019.
- Y. Chen, B. Dong, J. Xu. Meta-MgNet: Meta Multigrid Networks for Solving Parameterized Partial Differential Equations Image Classification. ArXiv:2010.14088, 2020
- J. He, L. Li, J. Xu. Approximation Properties of Deep ReLU CNNs. ArXiv:2109.00190, 2021.

Properties:

A summary of MgNet

- J. Xu, Deep Neural Networks and Multigrid Methods, (Lecture Notes at Penn State), 2021.
- J. He, J. Xu. MgNet: A Unified Framework of Multigrid and Convolutional Neural Network. Sci China Math, 2019, 62: 1331-1354, https://doi.org/10.1007/s11425-019-9547-2
- J. He, Y. Chen, L. Zhang, J. Xu. Constrained Linear Data-feature Mapping for Image Classification. ArXiv:1911.10428, 2019.
- Y. Chen, B. Dong, J. Xu. Meta-MgNet: Meta Multigrid Networks for Solving Parameterized Partial Differential Equations Image Classification. ArXiv:2010.14088, 2020
- J. He, L. Li, J. Xu. Approximation Properties of Deep ReLU CNNs. ArXiv:2109.00190, 2021.

Properties:

1. A uniform framework for understanding and designing CNNs.

A summary of MgNet

- J. Xu, Deep Neural Networks and Multigrid Methods, (Lecture Notes at Penn State), 2021.
- J. He, J. Xu. MgNet: A Unified Framework of Multigrid and Convolutional Neural Network. Sci China Math, 2019, 62: 1331-1354, https://doi.org/10.1007/s11425-019-9547-2
- J. He, Y. Chen, L. Zhang, J. Xu. Constrained Linear Data-feature Mapping for Image Classification. ArXiv:1911.10428, 2019.
- Y. Chen, B. Dong, J. Xu. Meta-MgNet: Meta Multigrid Networks for Solving Parameterized Partial Differential Equations Image Classification. ArXiv:2010.14088, 2020
- J. He, L. Li, J. Xu. Approximation Properties of Deep ReLU CNNs. ArXiv:2109.00190, 2021.

Properties:

1. A uniform framework for understanding and designing CNNs.
2. Unify and develop many classical CNN structures.

A summary of MgNet

- J. Xu, Deep Neural Networks and Multigrid Methods, (Lecture Notes at Penn State), 2021.
- J. He, J. Xu. MgNet: A Unified Framework of Multigrid and Convolutional Neural Network. Sci China Math, 2019, 62: 1331-1354, https://doi.org/10.1007/s11425-019-9547-2
- J. He, Y. Chen, L. Zhang, J. Xu. Constrained Linear Data-feature Mapping for Image Classification. ArXiv:1911.10428, 2019.
- Y. Chen, B. Dong, J. Xu. Meta-MgNet: Meta Multigrid Networks for Solving Parameterized Partial Differential Equations Image Classification. ArXiv:2010.14088, 2020
- J. He, L. Li, J. Xu. Approximation Properties of Deep ReLU CNNs. ArXiv:2109.00190, 2021.

Properties:

1. A uniform framework for understanding and designing CNNs.
2. Unify and develop many classical CNN structures.
   - AlexNet, ResNet, U-Net, Audo-encoder, DenseNet . . .

A summary of MgNet

- J. Xu, Deep Neural Networks and Multigrid Methods, (Lecture Notes at Penn State), 2021.
- J. He, J. Xu. MgNet: A Unified Framework of Multigrid and Convolutional Neural Network. Sci China Math, 2019, 62: 1331-1354, https://doi.org/10.1007/s11425-019-9547-2
- J. He, Y. Chen, L. Zhang, J. Xu. Constrained Linear Data-feature Mapping for Image Classification. ArXiv:1911.10428, 2019.
- Y. Chen, B. Dong, J. Xu. Meta-MgNet: Meta Multigrid Networks for Solving Parameterized Partial Differential Equations Image Classification. ArXiv:2010.14088, 2020
- J. He, L. Li, J. Xu. Approximation Properties of Deep ReLU CNNs. ArXiv:2109.00190, 2021.

Properties:

1. A uniform framework for understanding and designing CNNs.
2. Unify and develop many classical CNN structures.
   - AlexNet, ResNet, U-Net, Audo-encoder, DenseNet . . .
3. Construct the new

A summary of MgNet

- J. Xu, Deep Neural Networks and Multigrid Methods, (Lecture Notes at Penn State), 2021.
- J. He, J. Xu. MgNet: A Unified Framework of Multigrid and Convolutional Neural Network. Sci China Math, 2019, 62: 1331-1354, https://doi.org/10.1007/s11425-019-9547-2
- J. He, Y. Chen, L. Zhang, J. Xu. Constrained Linear Data-feature Mapping for Image Classification. ArXiv:1911.10428, 2019.
- Y. Chen, B. Dong, J. Xu. Meta-MgNet: Meta Multigrid Networks for Solving Parameterized Partial Differential Equations Image Classification. ArXiv:2010.14088, 2020
- J. He, L. Li, J. Xu. Approximation Properties of Deep ReLU CNNs. ArXiv:2109.00190, 2021.

Properties:

1. A uniform framework for understanding and designing CNNs.
2. Unify and develop many classical CNN structures.
   - AlexNet, ResNet, U-Net, Audo-encoder, DenseNet . . .
3. Construct the new

A summary of MgNet

- J. Xu, Deep Neural Networks and Multigrid Methods, (Lecture Notes at Penn State), 2021.
- J. He, J. Xu. MgNet: A Unified Framework of Multigrid and Convolutional Neural Network. Sci China Math, 2019, 62: 1331-1354, https://doi.org/10.1007/s11425-019-9547-2
- J. He, Y. Chen, L. Zhang, J. Xu. Constrained Linear Data-feature Mapping for Image Classification. ArXiv:1911.10428, 2019.
- Y. Chen, B. Dong, J. Xu. Meta-MgNet: Meta Multigrid Networks for Solving Parameterized Partial Differential Equations Image Classification. ArXiv:2010.14088, 2020
- J. He, L. Li, J. Xu. Approximation Properties of Deep ReLU CNNs. ArXiv:2109.00190, 2021.

Properties:

1. A uniform framework for understanding and designing CNNs.
2. Unify and develop many classical CNN structures.
   - AlexNet, ResNet, U-Net, Audo-encoder, DenseNet . . .
3. Construct the new
   - reduce the free parameters 1%, .1%, .01%...?
   - increase the generalization accuracy,
   - accelerate the training speed,
   - extend to general graph models,
   - ...

Concluding remarks

Concluding remarks

- General data classification

Concluding remarks

- General data classification
  - Linearly separable sets: logistic regressions

Concluding remarks

- General data classification
  - Linearly separable sets: logistic regressions
  - Nonlinear feature mapping: from nonlinear separable to linearly separable

Concluding remarks

- General data classification
  - Linearly separable sets: logistic regressions
  - Nonlinear feature mapping: from nonlinear separable to linearly separable
  - Deep learning: feature mapping given by deep neural networks

Concluding remarks

- General data classification
  - Linearly separable sets: logistic regressions
  - Nonlinear feature mapping: from nonlinear separable to linearly separable
  - Deep learning: feature mapping given by deep neural networks
- Image classification

Concluding remarks

- General data classification
  - Linearly separable sets: logistic regressions
  - Nonlinear feature mapping: from nonlinear separable to linearly separable
  - Deep learning: feature mapping given by deep neural networks
- Image classification
  - Constraint linear model for data-feature
  - Feature extraction by iterative methods

Concluding remarks

- General data classification
  - Linearly separable sets: logistic regressions
  - Nonlinear feature mapping: from nonlinear separable to linearly separable
  - Deep learning: feature mapping given by deep neural networks
- Image classification
  - Constraint linear model for data-feature
  - Feature extraction by iterative methods
- MgNet
  - A special CNN by making minor modification of multigrid
  - More mathematical structures lead to fewer weights

Concluding remarks

- General data classification
  - Linearly separable sets: logistic regressions
  - Nonlinear feature mapping: from nonlinear separable to linearly separable
  - Deep learning: feature mapping given by deep neural networks
- Image classification
  - Constraint linear model for data-feature
  - Feature extraction by iterative methods
- MgNet
  - A special CNN by making minor modification of multigrid
  - More mathematical structures lead to fewer weights
- Goals

Concluding remarks

- General data classification
  - Linearly separable sets: logistic regressions
  - Nonlinear feature mapping: from nonlinear separable to linearly separable
  - Deep learning: feature mapping given by deep neural networks
- Image classification
  - Constraint linear model for data-feature
  - Feature extraction by iterative methods
- MgNet
  - A special CNN by making minor modification of multigrid
  - More mathematical structures lead to fewer weights
- Goals
  - Slimmer CNN with higher generalization accuracy

Concluding remarks

- General data classification
  - Linearly separable sets: logistic regressions
  - Nonlinear feature mapping: from nonlinear separable to linearly separable
  - Deep learning: feature mapping given by deep neural networks
- Image classification
  - Constraint linear model for data-feature
  - Feature extraction by iterative methods
- MgNet
  - A special CNN by making minor modification of multigrid
  - More mathematical structures lead to fewer weights
- Goals
  - Slimmer CNN with higher generalization accuracy
  - Fast training algorithms (under development)

Concluding remarks

- General data classification
  - Linearly separable sets: logistic regressions
  - Nonlinear feature mapping: from nonlinear separable to linearly separable
  - Deep learning: feature mapping given by deep neural networks
- Image classification
  - Constraint linear model for data-feature
  - Feature extraction by iterative methods
- MgNet
  - A special CNN by making minor modification of multigrid
  - More mathematical structures lead to fewer weights
- Goals
  - Slimmer CNN with higher generalization accuracy
  - Fast training algorithms (under development)
  - More mathematical theory for deep learning