Deep Learning and Numerical PDEs

Introduction, Finite Element and ReLU DNN

Jinchao Xu

KAUST and Penn State

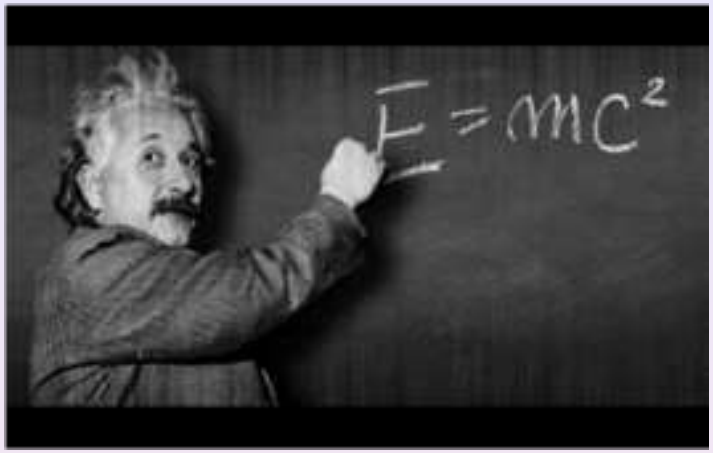
xu@multigrid.org

Morgan State University, June 19th, 2023

CBMS Lecture Series

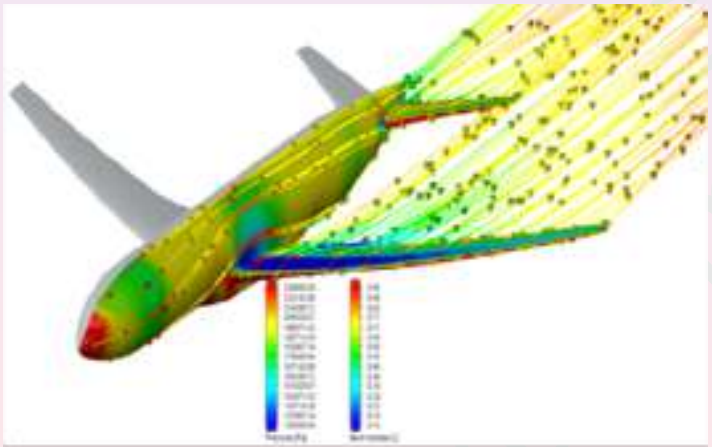
NSF DMS-2111387, KAUST Baseline Research Fund

Four major methods of scientific research



Theoretical Science

Experimental Science



Computational Science

Data Science

Data Science

One big success:

Deep Learning

One big question:

Is “deep learning” really a science?

Deep Concern

Deep learning is "alchemy".

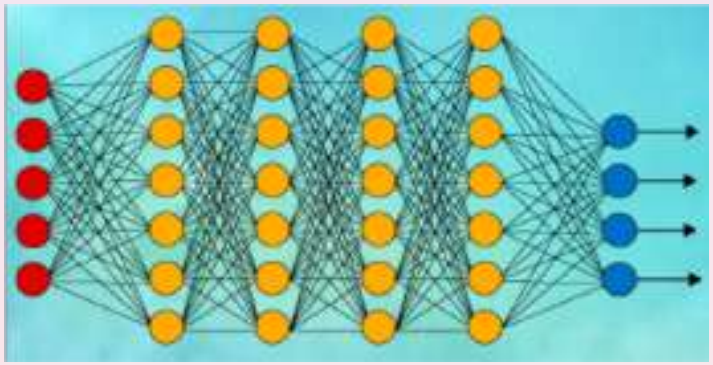
This series of lectures:

- Some mathematical understanding of deep learning ...
- Application to numerical solution of partial differential equations (PDEs)

What is intelligence?

- **Intelligence**: ability to learn, understand, and judge
- **Natural**: existing in nature and not designed and made by human
- **Artificial**: designed and made by human
- **Artificial intelligence**: also called machine intelligence, refers to the intelligence expressed by machines and computer programs developed by humans

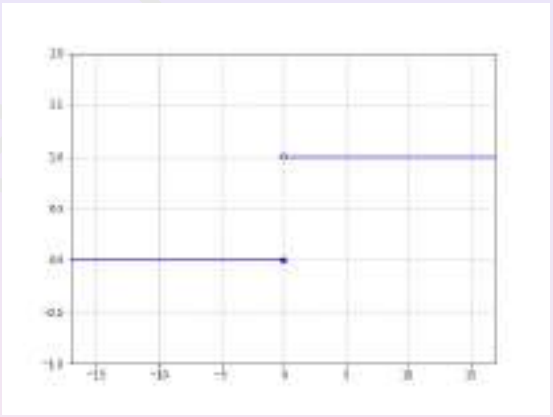
Natural Brain



Machine Brain

Decision making

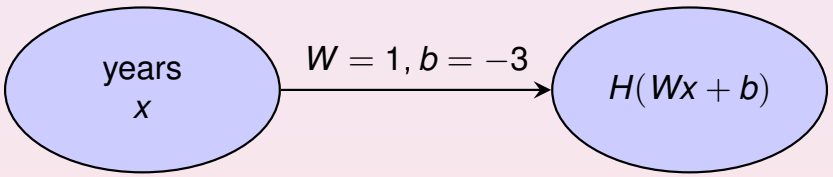
$$H(x) = \begin{cases} 1, & x > 0 \quad (Yes) \\ 0, & x \leq 0 \quad (No) \end{cases}$$



Example: hiring based on experience

- x = experience (years)

$$H(x-3) = \begin{cases} 1, & x > 3 \quad (Yes) \\ 0, & x \leq 3 \quad (No) \end{cases}$$



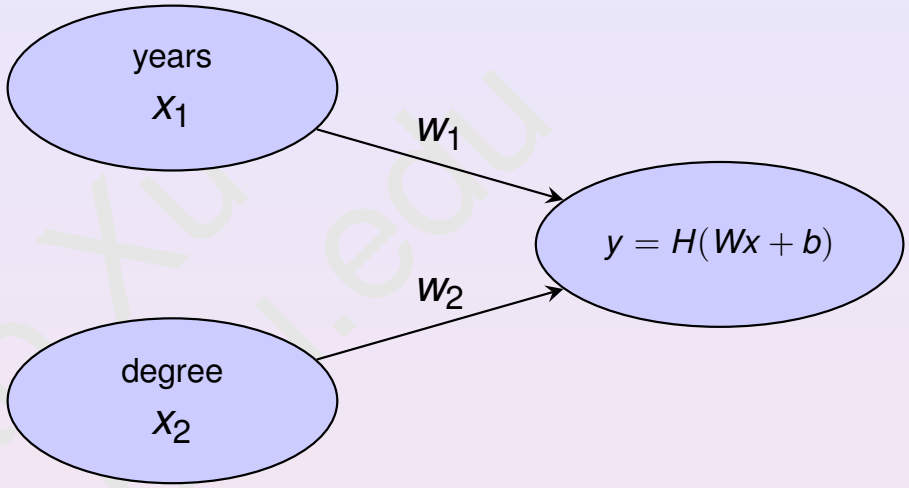
Example: hiring based on two factors

factors:

- x_1 = experience (years)
- x_2 = education (degree)

score: $w_1 x_1 + w_2 x_2 + b = Wx + b$

$$W = \begin{pmatrix} w_1 & w_2 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$



decision: $y = H(Wx + b)$

- $y = 1$, Yes
- $y = 0$, No

$$w_1 = 1, w_2 = 2, b = -5$$

candidate	years	degree	score	decision
1	3	Ph.D(4)	6	Yes
2	6	Master(2)	5	Yes
3	4	Bachelor(1)	1	Yes
4	1	Master(2)	0	No

- H: only gives two values 0 or 1, not continuous

Example: hiring with salary

Continuous activation:

$$\mathrm{ReLU}(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

Refined decision:

salary: $y = W^1\,\mathrm{ReLU}(W^0 x + b^0)$

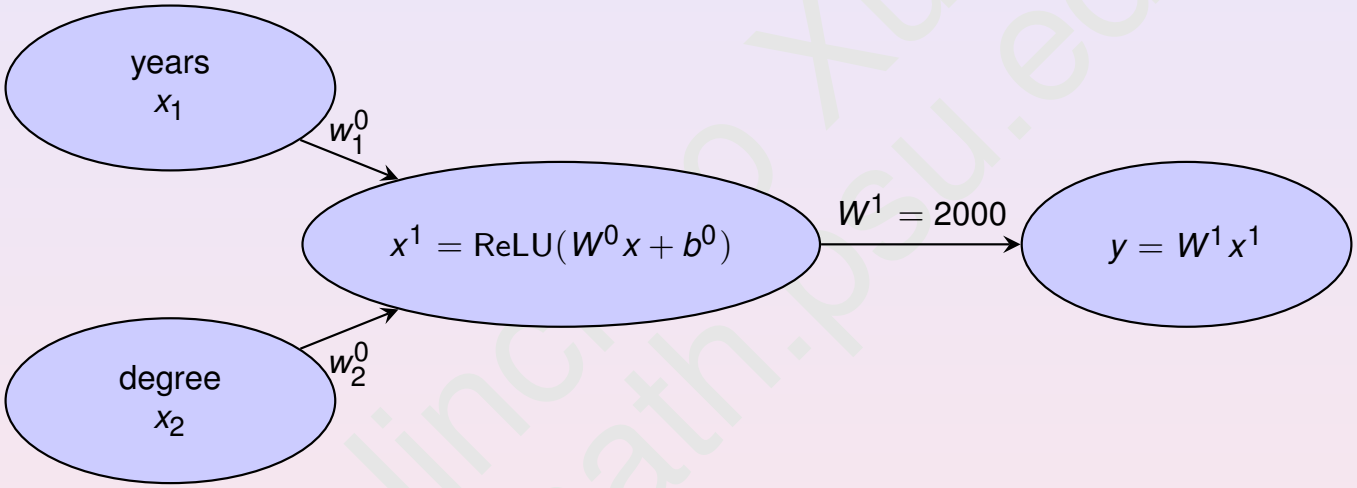
- $y > 0$, Yes with salary
- $y \leq 0$, No

For example: $W^1 = 2000$

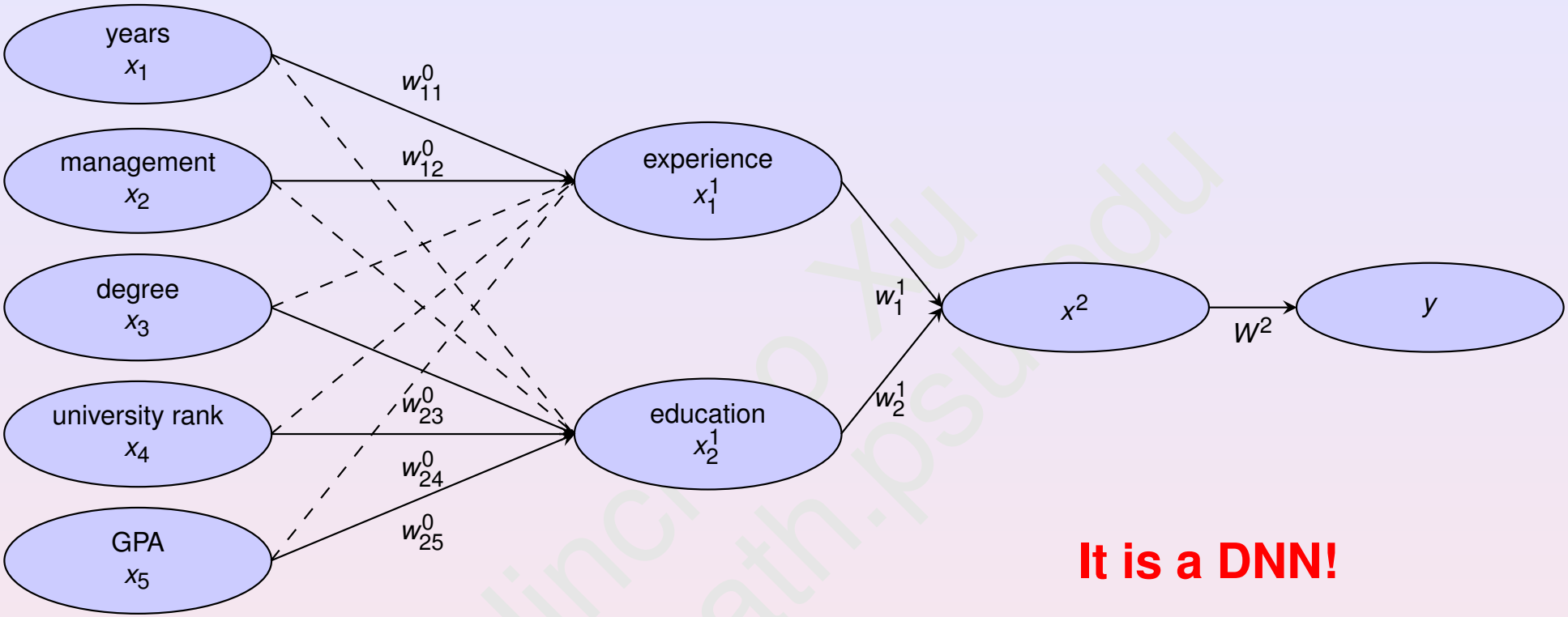
$y = 2000\,\mathrm{ReLU}(W^0 x + b^0)$

candidate	years	degree	salary
1	3	Ph.D(4)	12000
2	6	Master(2)	10000
3	4	Bachelor(1)	2000
4	1	Master(2)	0

Example: hiring with salary



Example: a more refined model



It is a DNN!

$$x^0 = x = \begin{pmatrix} x_1 \\ \vdots \\ x_5 \end{pmatrix}$$

$$x^1 = \mathrm{ReLU}\left(W^0 x + b^0\right)$$

$$W^0 = \begin{pmatrix} w_{11}^0 & w_{12}^0 & 0 & 0 & 0 \\ 0 & 0 & w_{23}^0 & w_{24}^0 & w_{25}^0 \end{pmatrix}$$

$$b^0 = \begin{pmatrix} b_1^0 \\ b_2^0 \end{pmatrix}$$

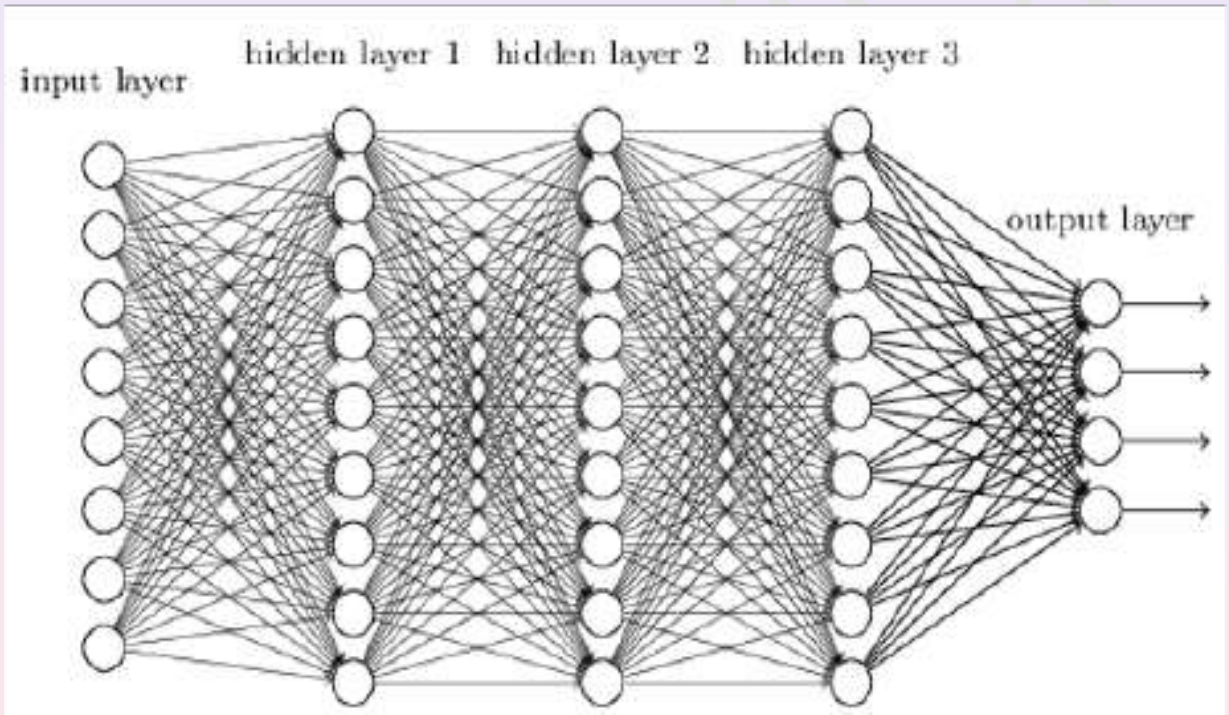
$$x^2 = \mathrm{ReLU}\left(W^1 x^1 + b^1\right)$$

$$W^1 = \begin{pmatrix} w_1^1 & w_2^1 \end{pmatrix}$$

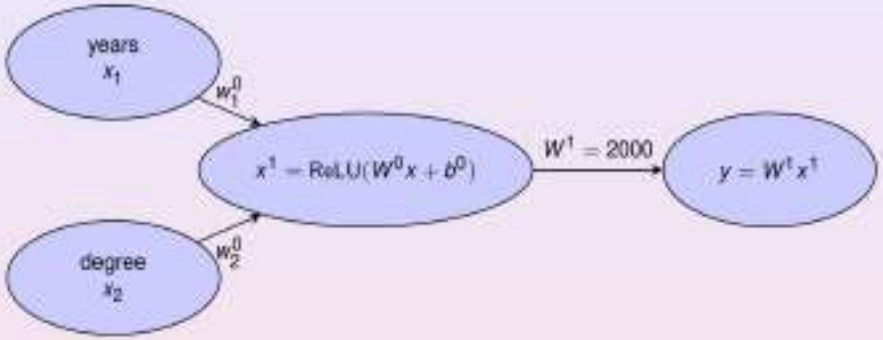
$$y = W^2 x^2$$

Deep neural network

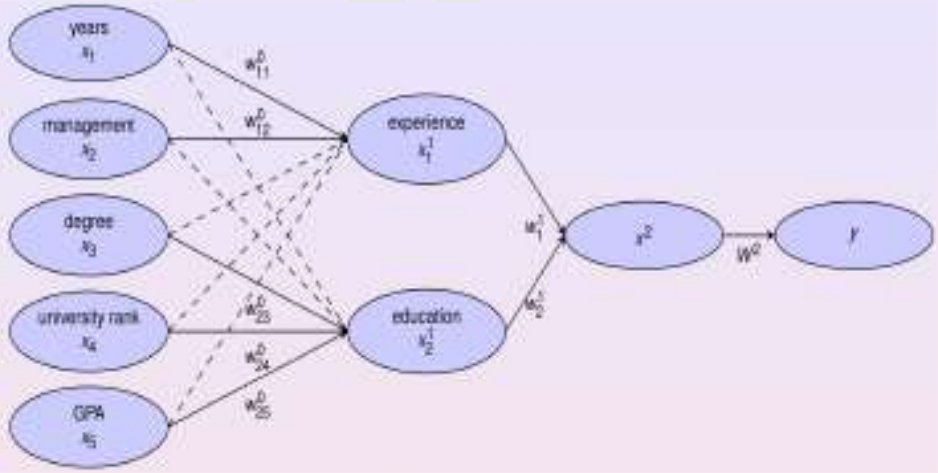
A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers



Deep neural network



3 layers DNN (4 neurons)



4 layers DNN (9 neurons)

How to determine the parameters w and b?

- **Natural brains**
  - HRs use their hiring experience
  - $w_1 = 1$, $w_2 = 2$, “education is more important than experience"
- **Machine brains (learning from data)**
  - Hiring data, performance of employees
  - w and b can be learned from data

How to learn?

What is learning?

- **Employees** p^i: $i = 1, 2, 3, \ldots, N$
- **Input** $x^i = \begin{pmatrix} x_1^i \\ x_2^i \end{pmatrix}$: experience, education of p^i
- y^i: evaluation of p^i, $y^i \in \{0, 1, 2, 3, 4\}$ and

$$\begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \leftrightarrow \begin{pmatrix} \text{bad} \\ \text{fair} \\ \text{good} \\ \text{very good} \\ \text{excellent} \end{pmatrix}$$

- **Model**: $f(x, \theta) = w^2\sigma(w^1\sigma(w^0 x + b^0) + b^1)$
- **Learning from data**: Find $\theta = (w, b)$ s.t.

$$f\left(x^i, \theta\right) \approx y^i \ : \ \min L(\theta) = \min \frac{1}{N} \sum_{i=1}^{N} \left| f\left(x^i, \theta\right) - y^i \right|^2$$

and

$$\theta^* = \arg\min L(\theta)$$

- **New hiring**: calculate the score of the new candidate

$$f(x^{New}, \theta^*) = ?$$

How to find θ^*?

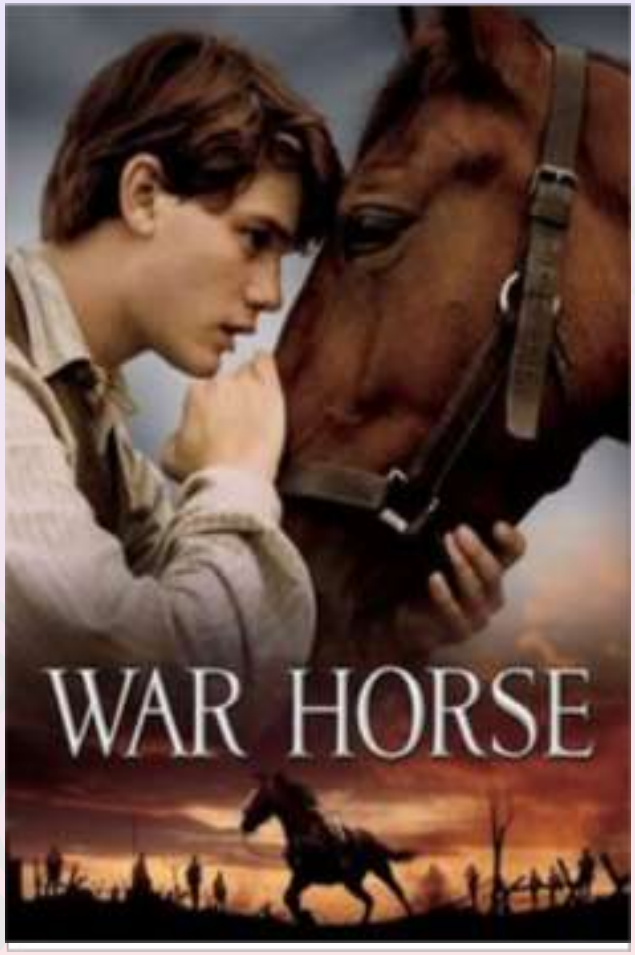
- **Gradient descent method**:

$$\theta_{t+1} = \theta_t - \eta_t \nabla L(\theta_t)$$

Figure: The fastest local decreasing direction: $-\nabla L(\theta_t)$

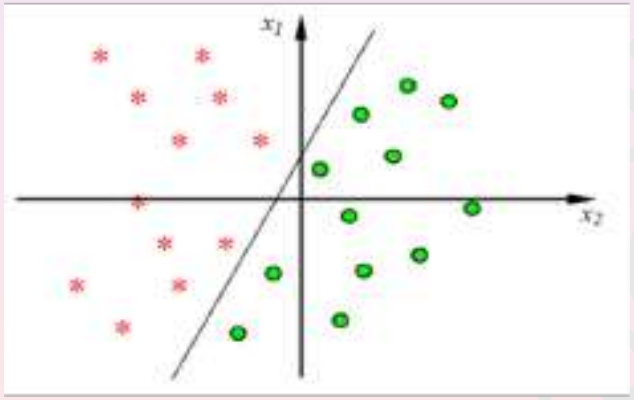
Examples of natural brains and machine brains

Bird

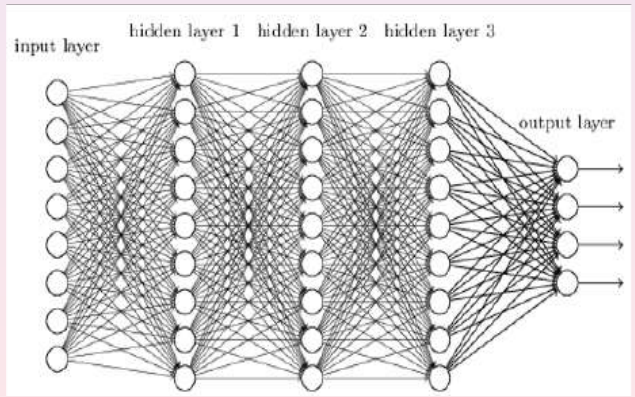


War Horse

Human

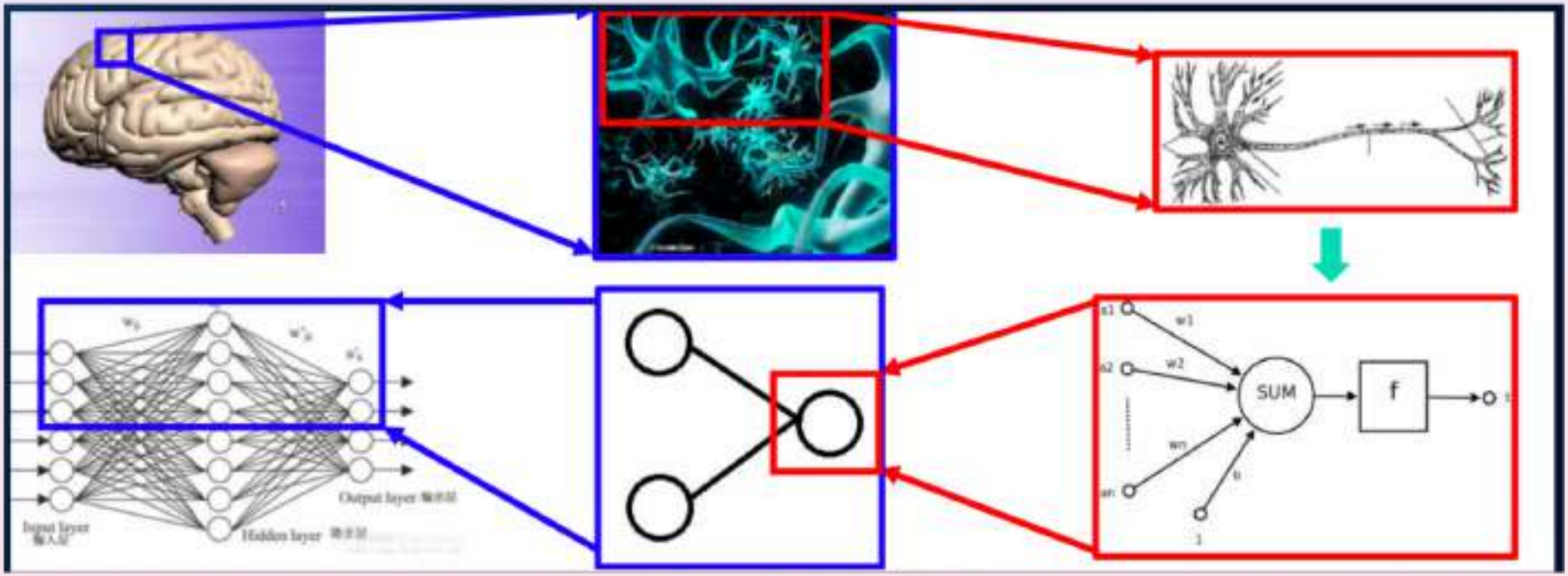


Linear Regression

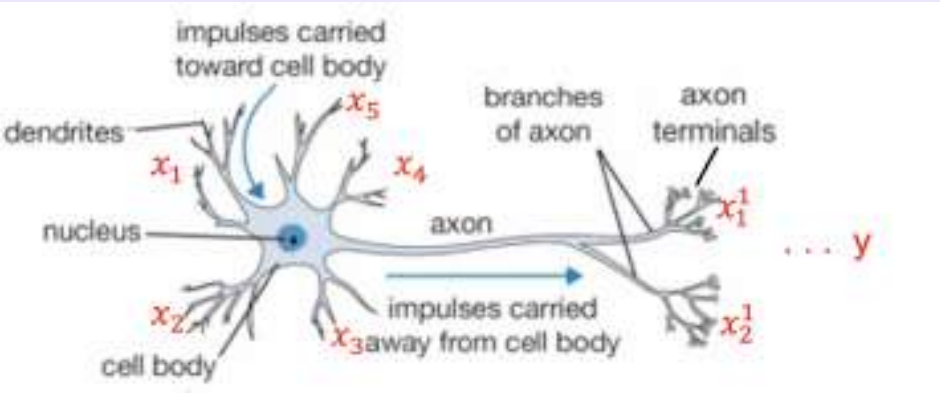


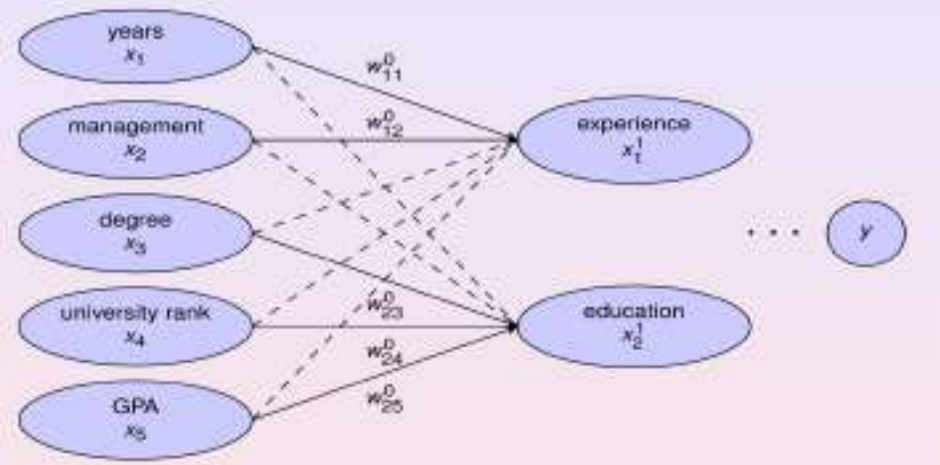
Deep Neural Network

Natural brains v.s. machine brains



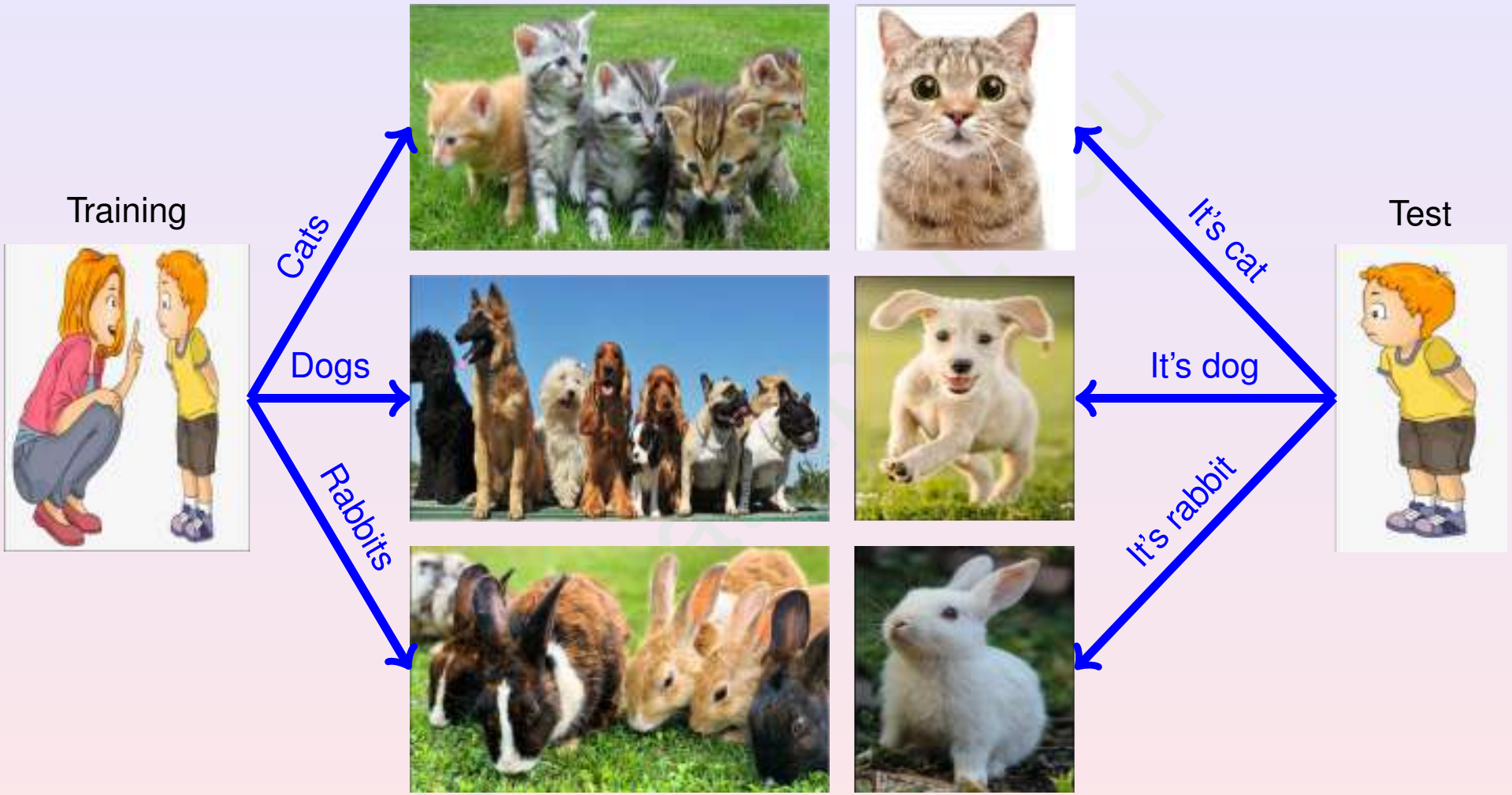
Natural brains v.s. machine brains



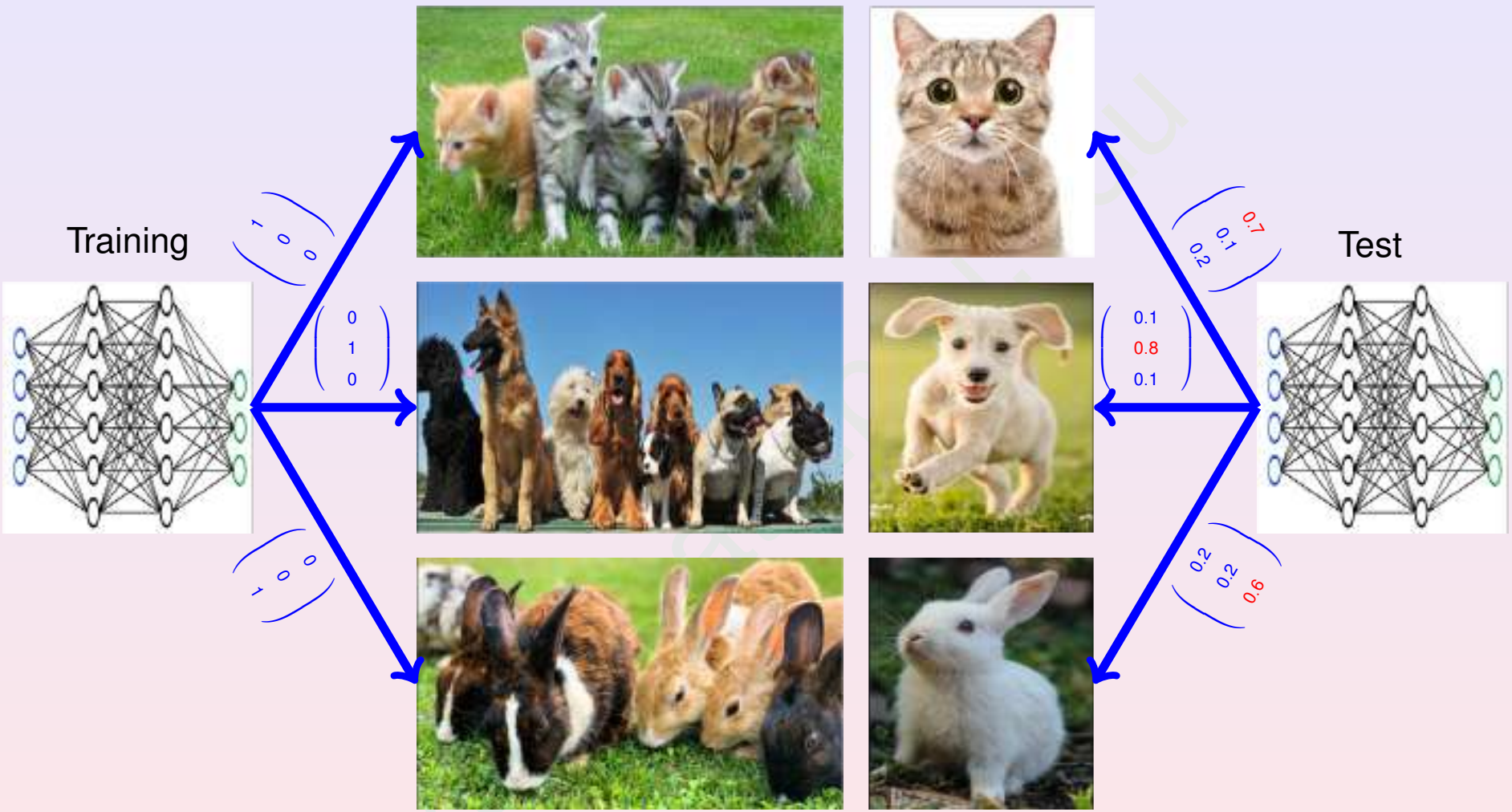


1. Dendrite receives input signal x
2. x is processed in the cell body
   $x^1 = ReLU(W^0 x + b^0)$
3. The output signal x^1 is carried from the cell body to next neuron through axon
4. Repeat steps 1-3 in the next neuron to obtain the score y

Natural brain: recognize between cats, dogs, rabbits



Machine brain : recognize between cats, dogs, rabbits



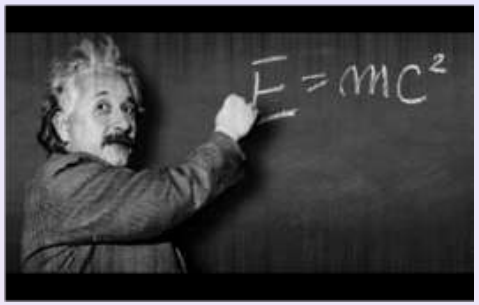
Different ways of learning

Example of language learning

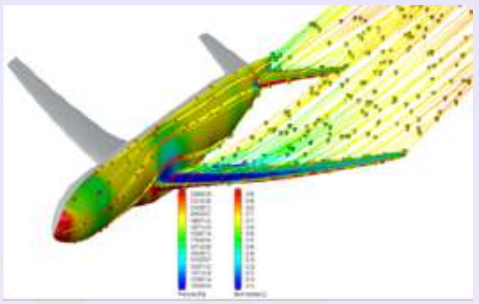
- Native language(data)
  - Imitation
  - Need of a lot of data
- Foreign language(logics)
  - Rules of pronunciation, syntax
  - No need of a lot of data
- More effective: data + logics
  - Native language: go to school
  - Foreign language: practice more

Intelligence and learning

Logics



Theoretical Science



Computational Science

Data

Experimental Science

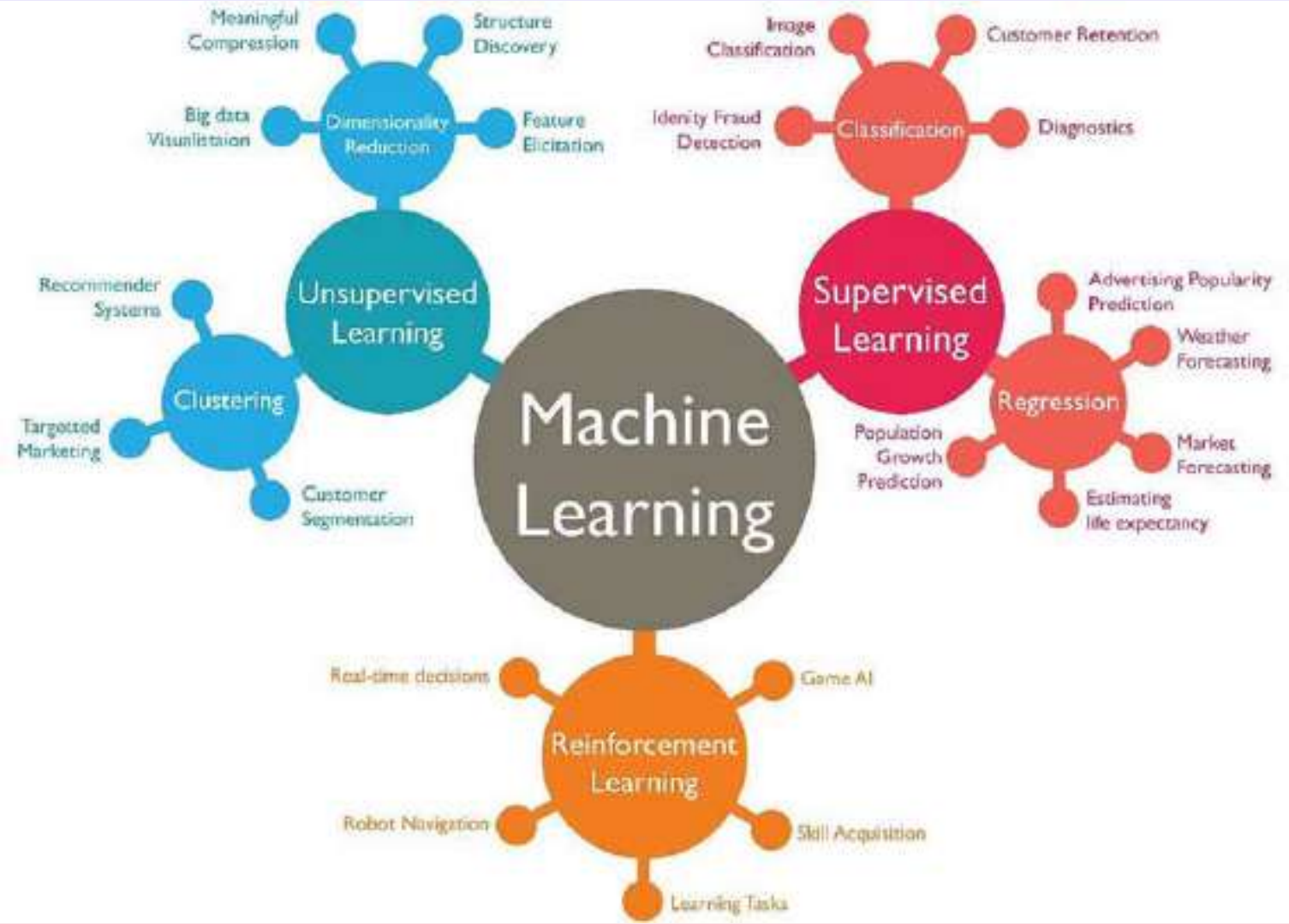
Data Science

Data + Logics: Logics Based Machine Learning (LBML)

Example: Physics-Informed Neural Networks (PINN)

- M. Raissi, P. Perdikaris, G. Karniadakis, 2019.

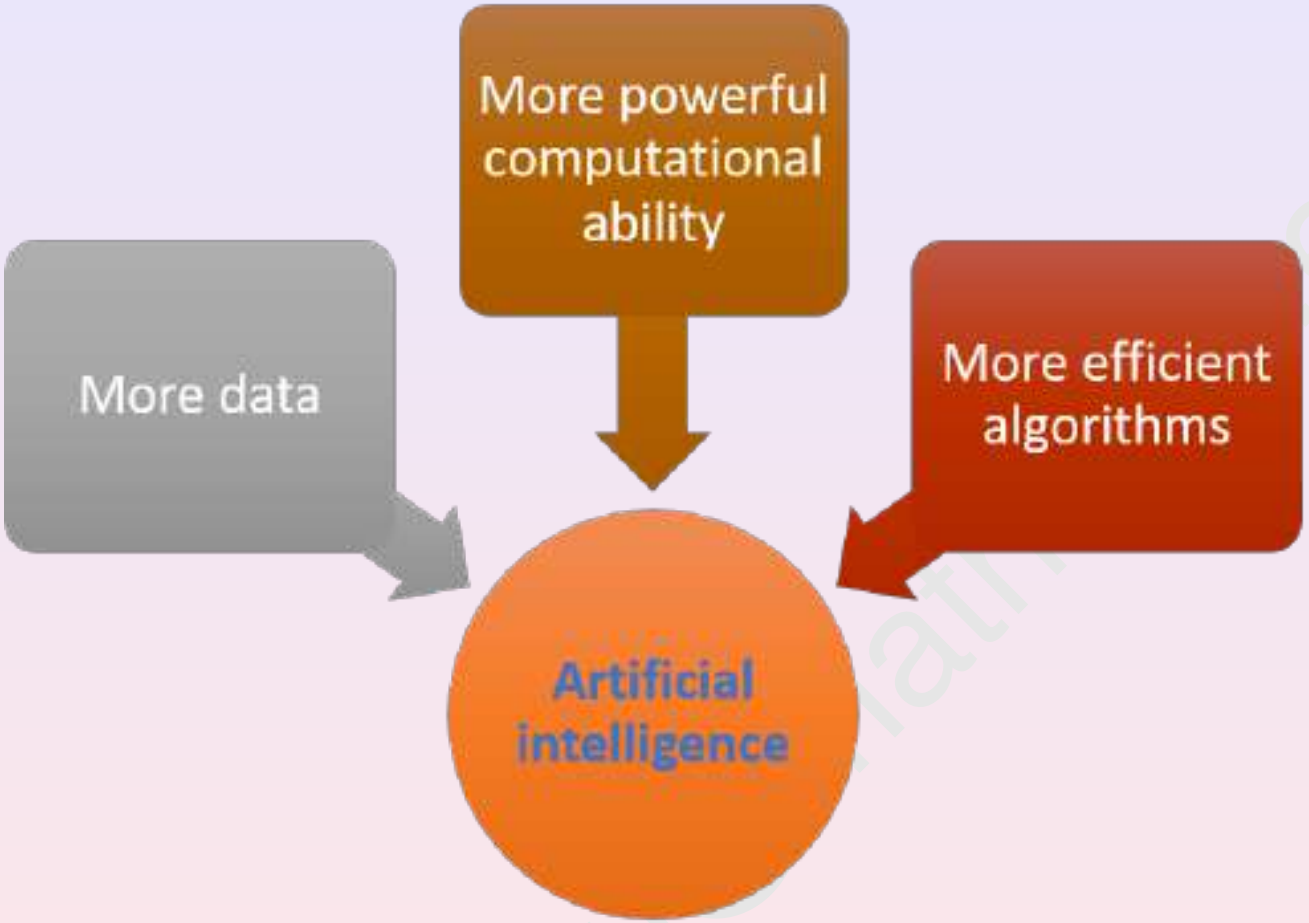
Machine learning



[1]

[1] https://towardsdatascience.com/introduction-to-machine-learning-for-beginners-eed6024fdb08

The era of the artificial intelligence



Question: what are the state of the art deep learning technologies?

Success Stories - I: Image Classification

- In 1998, LeCun etc. proposed the neural network LeNet-5 based on the convolution and applied to the hand-written digits recognition successfully. Hence LeNet-5 is also called the first convolutional neural network(CNN) successfully applied.
- In 2012, Hinton and his student Alex joined the graph recognition competition by ImageNet and improved the accuracy significantly with the CNN AlexNet.
- In 2013 Google purchased a Canadian startup company on neural network, DNNResearch. It's set up by Geoffrey Hinton and his graduate students Alex Krizhevsky and Ilya Sutskever in 2012.
- In 2015, Kaiming He etc. proposed the ResNet CNN structure, which has become a state-of-art CNN structure and widely used in industry and discussed in academia.
- Afterwards CNN is widely used in the field of computer vision and breaks the records ceaselessly.
- Transformers ...

Success Stories - II: AlphaGo & AlphaZero

- From 2016 to 2017, the AI program AlphaGo developed by Google DeepMind beat down all the Go champions worldwide.
- 2018 AlphaZero gave a unified principle for many other board games.
- The CEO of Google DeepMind Demis Hassabis announced to integrate AlphaGo with medical, robots and so on. They can learn by themselves since they are artificial intelligence, and transfer learning can be done with enough data.

Success Stories - III: Auto-driving

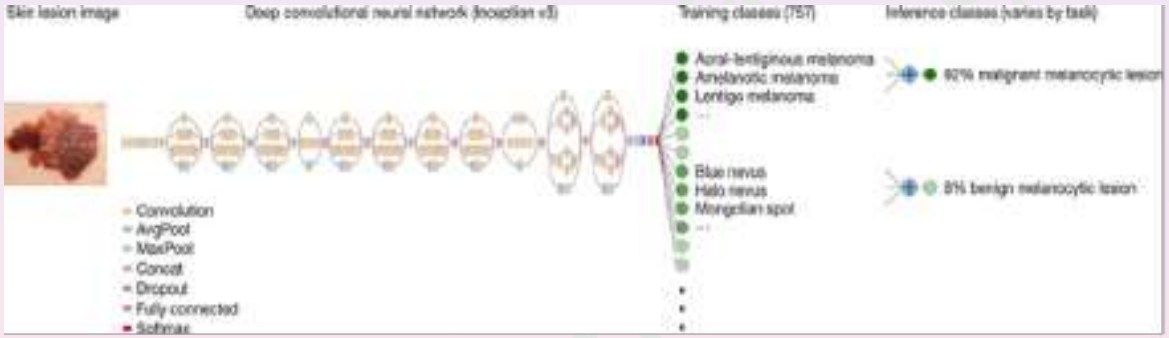
Auto-driving (the next competition in AI)

- In 2009, Google proposed the plan to replace human driving with softwares. Afterwards, many large technology companies such as Tesla, Google, Uber and Benz devoted a lot on investigating the technology of automated driving.
- There are about dozens of companies focus on automatic drive techniques from L2 to L4.
- In many countries, road examination autonomous vehicles are allowed with applications.
- Tesla – Autopilot; Google – Waymo; Baidu – Apollo; GM – Cruise; Volkswagen – DAS Autonomy...

Success Stories - IV: Diagnosis and Classification of Cancer

AI can automatically diagnose the cancer

- In 2017, a team in Stanford University achieved AI automatic diagnosis of the skin cancer with the CNN, the accuracy of which is as high as a human expert.
- This model is trained based on a public model by Google, while the original model is only used to classify the cats and dogs in photos.



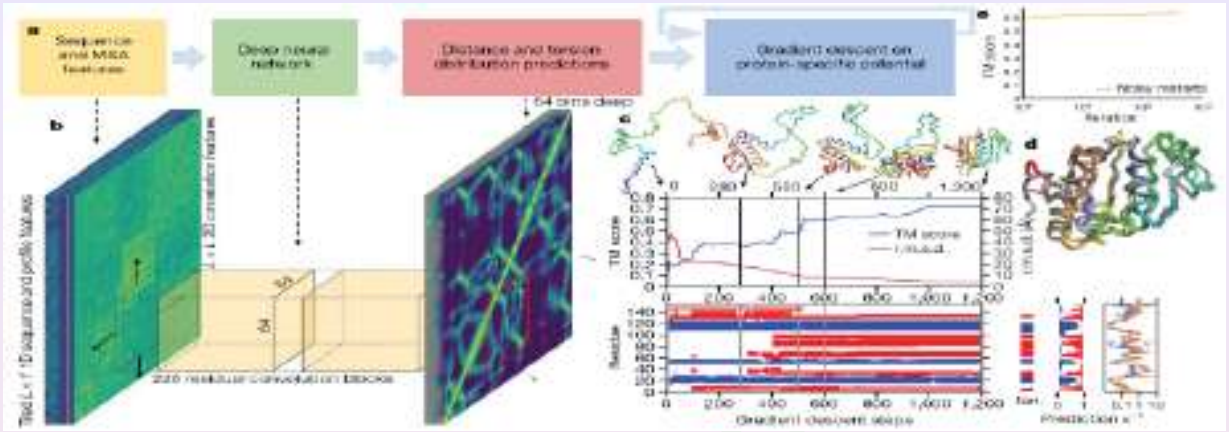
Ref: *Dermatologist-level classification of skin cancer with deep neural networks*, Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau & Sebastian Thrun, *Nature* 542.7639 (2017): 115-118.

Success Stories - V: ChatGPT

- ChatGPT (Chat Generative Pre-trained Transformer) is a chatbot launched by OpenAI in November 2022.
- It is built on top of OpenAI's GPT-3 family of large language models, and is fine-tuned with both supervised and reinforcement learning techniques.
- ChatGPT is very versatile:
  - write and debug computer programs;
  - compose music, teleplays, fairy tales, and student essays, poetry and song lyrics;
  - emulate a Linux system;
  - simulate an entire chat room;
  - . . .

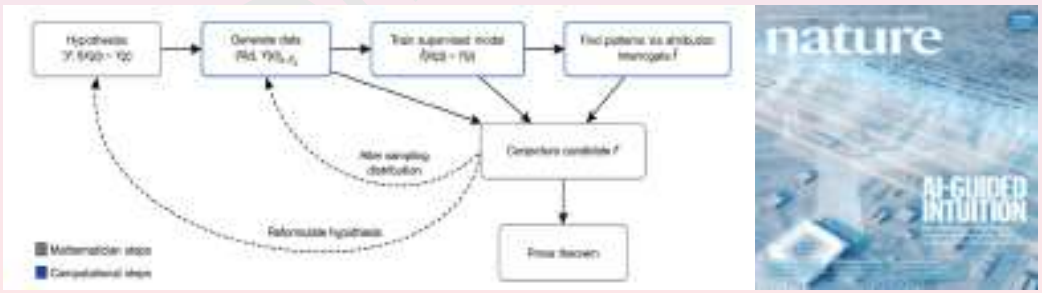
Success Stories - VI: Science and Math

AlphaFold (Nature, Jul. 2021): Protein structure prediction with score 92.4% using CNNs.



- An AI system developed by DeepMind that predicts a protein's 3D structure from its amino acid sequence
- It regularly achieves accuracy competitive with experiment

Advancing mathematics (Nature, Dec. 2021): Guiding human intuition with AI (DNNs).



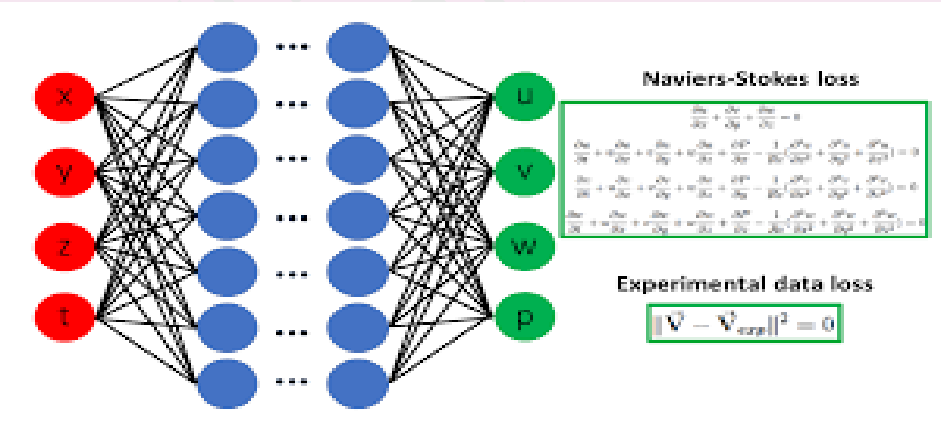
Success Stories - VII: Numerical PDEs

Approximation of PDE solutions:

- PINNs;
- Finite neuron methods;
- DeepRitz;
- . . .

Approximation of operators:

- DeepONet;
- FNO;
- Transformers;
- . . .



Goal of the lecture series

Develop a mathematical understanding of deep learning ...

In particular: deep learning for numerical PDEs ...

Topics in the lecture series

- Finite element and deep neural network functions
  - FE spaces and ReLU^k NNs
  - ReLU DNN $\equiv$ linear FE
- Iterative methods and frequency principle
  - Basic iterative methods
  - Frequency principles
  - Multigrid method
- Image Classification and MgNet
  - Logistic regression
  - Image classification
  - MgNet: A "trained" multigrid method
- Application to PDEs: Finite neuron method
  - Error analysis
  - Novel training algorithms
- Approximation theory of neural network functions
  - Variation space
  - Metric entropy
  - Error estimate

Finite element: Piecewise linear functions

- Uniform grid $\mathcal{T}_h$

$$0 = x_0 < x_1 < \cdots < x_{N+1} = 1, \quad x_j = \frac{j}{N+1} \ (j = 0 : N+1).$$

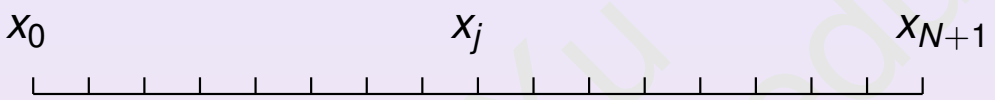


Figure: 1D uniform grid

- Linear finite element space

$$V_h = \{v_h : \ v \text{ is continuous and piecewise linear w.r.t. } \mathcal{T}_h \text{ , } v_h(0) = 0\}.$$

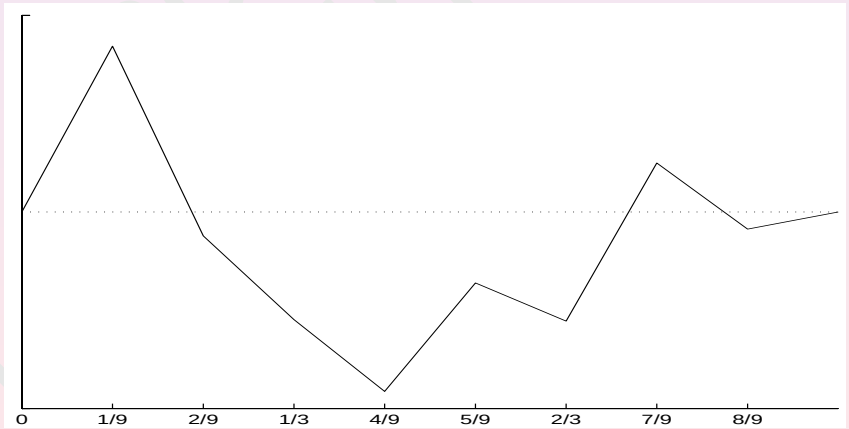
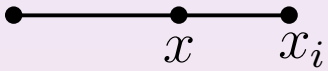


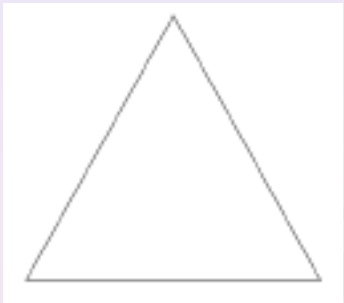
Figure: Linear finite element functions.

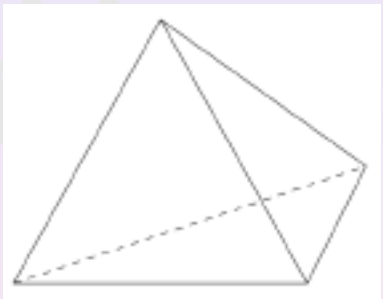
Finite element in multi-dimensions

($k = 1$)

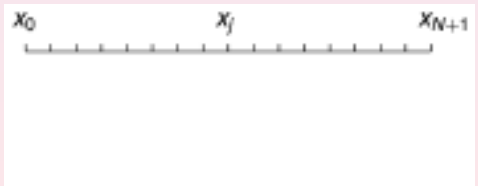
$w_1 x + b$ $\quad$ $w_1 x_1 + w_2 x_2 + b$ $\quad$ $w_1 x_1 + w_2 x_2 + w_3 x_3 + b$ $\quad \cdots$

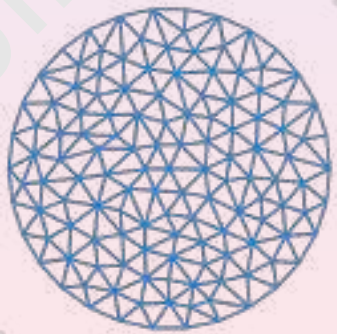


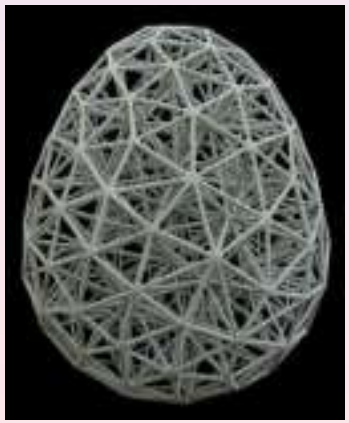




$\cdots$





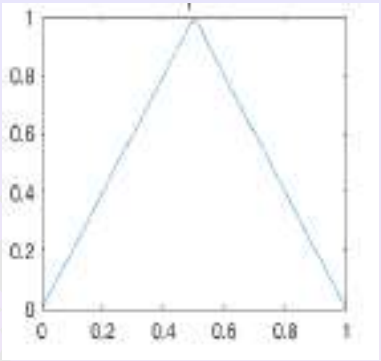


$\cdots$

Two basis functions for the finite element space V_h

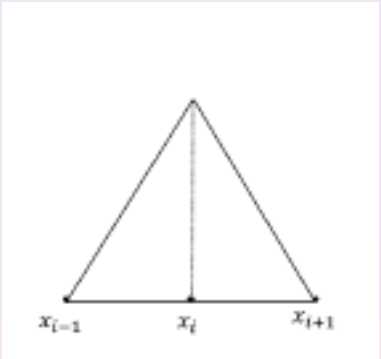
- Hat basis:

$$\varphi(x) = \begin{cases} 2x & x \in [0, \frac{1}{2}], \\ 2(1-x) & x \in [\frac{1}{2}, 1], \\ 0, & \text{otherwise.} \end{cases}$$



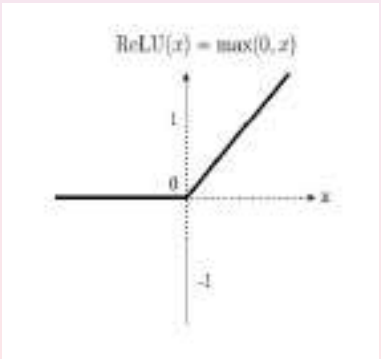
$$\varphi_i(x) = \frac{1}{\sqrt{h}}\varphi(\frac{x - x_{i-1}}{2h}) = \frac{1}{\sqrt{h}}\varphi(w_h x + b_i).$$

with $w_h = \frac{1}{2h}, b_i = \frac{-x_{i-1}}{2h}$.



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

$$r_i(x) = \frac{1}{\sqrt{h}}\mathrm{ReLU}(\frac{x - x_{i-1}}{2h}) = \frac{1}{\sqrt{h}}\mathrm{ReLU}(w_h x + b_i)$$



- $V_h = \mathrm{span}\{\varphi(w_h x + b_i)\} = \mathrm{span}\{\mathrm{ReLU}(w_h x + b_i)\}$

Finite element space and neural network functions

- Finite element function space on a uniform grid

$$V_h = \left\{ \sum_{i=1}^{n} a_i \varphi(w_h x + b_i), i = 1:n \right\} = \left\{ \sum_{i=1}^{n} a_i \mathrm{ReLU}(w_h x + b_i), i = 1:n \right\} \tag{1}$$

  with $w_h = \frac{1}{2h}, \quad b_i = \frac{-x_{i-1}}{2h}$.

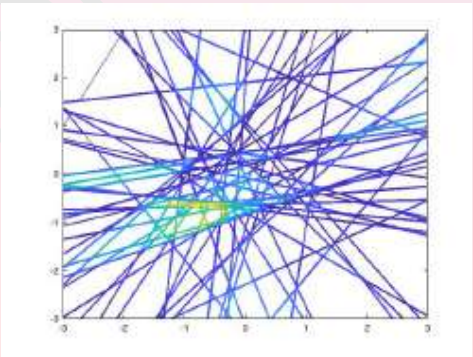
- Finite element function space on an arbitrary grid

$$\Sigma_n^{\mathrm{ReLU}} = \left\{ \sum_{i=1}^{n} a_i \mathrm{ReLU}(w_i x + b_i), w_i, b_i \in \mathbb{R}, i = 1:n \right\}, \quad x_i = -\frac{b_i}{w_i}, \tag{2}$$

  is contained in the ReLU neural network function space.

- ReLU neural network (shallow) functions in any dimension d

$$\Sigma_n^{\mathrm{ReLU}} = \left\{ \sum_{i=1}^{n} a_i \mathrm{ReLU}(w_i \cdot x + b_i), w_i \in \mathbb{R}^d, b_i \in \mathbb{R}, i = 1:n \right\} \tag{3}$$



Shallow NNs with general activation functions

Change ReLU to general activation function $\sigma : \mathbb{R} \mapsto \mathbb{R}$:

$$\Sigma_n^\sigma = \left\{ \sum_{i=1}^{n} a_i \sigma(w_i \cdot x + b_i), w_i \in \mathbb{R}^d, b_i \in \mathbb{R}, i = 1 : n \right\} \tag{4}$$

Popular activation functions:

- Heaviside $\sigma = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$
- Sigmoidal $\sigma = (1 + e^{-x})^{-1}$
- Rectified Linear $\sigma = \max(0, x)$
- Power of a ReLU $\sigma = [\max(0, x)]^k$
- Cosine $\sigma = \cos(x)$
- $\ldots$

Basic approximation properties

Can shallow networks approximate arbitrary functions?

- Recall $\Sigma_n^\sigma := \left\{\sum_{i=1}^n a_i \sigma(\omega_i \cdot x + b_i),\ a_i \in \mathbb{R},\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}\right\}$
- Is

$$\Sigma^\sigma = \bigcup_{n=1}^{\infty} \Sigma_n^\sigma \tag{5}$$

  dense in $C^0(\Omega)$ for bounded $\Omega \subset \mathbb{R}^d$?
- Yes! As long as σ is not a polynomial.

- Ref: M. Leshno, V. Lin, A. Pinkus and S. Schocken, 1993.

Activation function: non-polynomial

Lemma

Σ^σ is dense in $C^0(\Omega) \iff \sigma$ is not a polynomial (and almost everywhere continuous).

Proof.

1. Assume $\sigma \in C^\infty$
   - $[\sigma((\omega + he_j)\cdot x + b) - \sigma(\omega \cdot x + b)]/h \in \Sigma^\sigma$,
   - $\frac{\partial}{\partial \omega_j}\sigma(\omega \cdot x + b)|_{\omega=0} = x_j\sigma'(b) \in \overline{\Sigma^\sigma}$
     $\Rightarrow x_j \in \overline{\Sigma^\sigma}$ if $\sigma'(b) \neq 0$ for some b.
   - $D^\alpha_\omega \sigma(\omega \cdot x + b) = x^\alpha \sigma^{(|\alpha|)}(\omega \cdot x + b) \in \overline{\Sigma^\sigma}$
     $\Rightarrow x^\alpha = x_1^{\alpha_1} \cdots x_d^{\alpha_d} \in \overline{\Sigma^\sigma}$ if $\sigma^{(|\alpha|)}(b) \neq 0$ for some b
   - $\overline{\Sigma^\sigma}$ contains all polynomials.
2. If σ is not smooth ...

□

Our interest: What about approximation rates?

- Ref: M. Leshno, V. Lin, A. Pinkus and S. Schocken, 1993.

Convergence rate: approximation of Cosine networks

- Cosine networks

$$\Sigma_n^{\cos} = \left\{ u_n : u_n = \sum_{i=1}^{n} a_i \cos(\omega_i \cdot x + b_i), \quad \forall a_i, w_i, b_i \right\}$$

- Integral representation of u in terms of cosine functions

$$\begin{aligned}
u(x) &= Re \int_{\mathbb{R}^d} e^{2\pi i \omega \cdot x} \hat{u}(\omega) d\omega \\
&= \int_{\mathbb{R}^d} \cos(2\pi\omega \cdot x + b(\omega)) |\hat{u}(\omega)| d\omega \qquad (\text{let } \hat{u}(\omega) = |\hat{u}(\omega)| e^{ib(\omega)}) \\
&= \|\hat{u}\|_{L^1} \int_{\mathbb{R}^d} \cos(2\pi\omega \cdot x + b(\omega)) \lambda(\omega) d\omega \qquad (\text{let } \lambda(\omega) = \tfrac{|\hat{u}(\omega)|}{\|\hat{u}\|_{L^1}}) \\
&= \|\hat{u}\|_{L^1} \mathbb{E}\left(g(\omega, x)\right) \qquad (\text{let } g(\omega, x) = \cos(2\pi\omega \cdot x + b(\omega)))
\end{aligned}$$

- Sampling:

$$\mathbb{E}(g(\omega, x)) \approx \frac{1}{n} \sum_{j=1}^{n} \cos(2\pi\omega_j \cdot x + b(\omega_j)) \in \Sigma_n^{cos}$$

Sampling argument

1. Let $v = \mathbb{E}\left(g(\omega, x)\right) = \int_G g(\omega, x)\lambda(x)dx$ and the sampling $v_n = \frac{1}{n}\sum_{i=1}^{n} g(\omega_i, x)$

$$v(x) - v_n(x) = \mathbb{E}g(x) - \frac{1}{n}\sum_{i=1}^{n} g(\omega_i, x); \tag{6}$$

2. By a direct calculation

$$\begin{aligned}
&\mathbb{E}_n\left(\left\|\mathbb{E}g - \frac{1}{n}\sum_{i=1}^{n} g(\omega_i, \cdot)\right\|^2\right) \\
&= \int_{\mathbb{R}^{d\times n}} \left(\left\|\mathbb{E}g - \frac{1}{n}\sum_{i=1}^{n} g(\omega_i, \cdot)\right\|^2\right) \lambda(\omega_1)\cdots\lambda(\omega_n)d\omega_1\cdots d\omega_n \\
&= \frac{1}{n}(\mathbb{E}(\|g\|^2) - [E(g)]^2) \leq \frac{1}{n};
\end{aligned} \tag{7}$$

3. There exist $\{\omega_i^*\}_{i=1}^n$ such that

$$\|\mathbb{E}g - \frac{1}{n}\sum_{i=1}^{n} g(\omega_i^*, \cdot)\|_0^2 \leq n^{-1}. \tag{8}$$

Approximation rates for Cosine networks

The preceding *sampling argument* gives the *approximation rate*:

Theorem

There exists $u_n \in \Sigma_{n,M}^{\cos} = \left\{ \sum_{i=1}^{n} a_i \cos(\omega_i \cdot x + b_i), \sum_{i=1}^{n} |a_i| \leq M \right\}$ *such that*

$$\|u - u_n\| \leq n^{-\frac{1}{2}} \|\hat{u}\|_{L^1(\mathbb{R}^d)}. \tag{9}$$

where $M = \|\hat{u}\|_{L^1(\mathbb{R}^d)}$.

From Cosine networks to ReLU^k networks

$$u(x) = \frac{1}{4}\left(\frac{\pi}{2}\right)^k \int_{\mathbb{R}^d}\int_{\mathbb{R}} \mathrm{ReLU}^k\left(\pi^{-1}\omega\cdot x + b\right)|\hat{u}(\omega)|e^{-i(\pi b + \frac{\pi(k+1)}{2} + \theta(\omega))}dbd\omega \tag{10}$$

Sampling arguments $\Rightarrow$ approximation result

Theorem (X 2020)

$$\inf_{u_n\in\Sigma_n^k}\|u-u_n\|_{H^m(\Omega)} \lesssim \begin{cases} n^{-\frac{1}{2}-\frac{1}{d}}\|u\|_{B^{k+1}(\Omega)}, & k>m, \\ n^{-\frac{1}{2}}\|u\|_{B^{k+1}(\Omega)}, & k=m, \end{cases} \tag{11}$$

where

$$\|u\|_{B^s(\Omega)} = \inf_{u_e|\Omega=u}\int_{\mathbb{R}^d}(1+|\omega|)^s|\hat{u}_e(\omega)|d\omega. \tag{12}$$

More details and extensions for optimal rates and underlying function spaces, see other lectures.

Deep neural network (DNN)

1. Start from a linear function of $x \in \mathbb{R}^{n_0}$

$$W^0 x + b^0$$

2. Compose with the activation function:

$$x^{(1)} = \sigma(W^0 x + b^0) \in \mathbb{R}^{n_1}$$

3. Compose with another linear function:

$$W^1 x^{(1)} + b^1$$

4. Compose with the activation function:

$$x^{(2)} = \sigma(W^1 x^{(1)} + b^1) \in \mathbb{R}^{n_2}$$

5. ...

6.

$$x^{(L)} = \sigma(W^{L-1} x^{(L-1)} + b^{L-1}) \in \mathbb{R}^{n_L}$$

7. Compose with another linear function (final output layer)

$$f(x; \Theta) = W^L x^{(L)} + b^L$$

ReLU-DNN and FEM

Deep neural network functions with ℓ-hidden layers

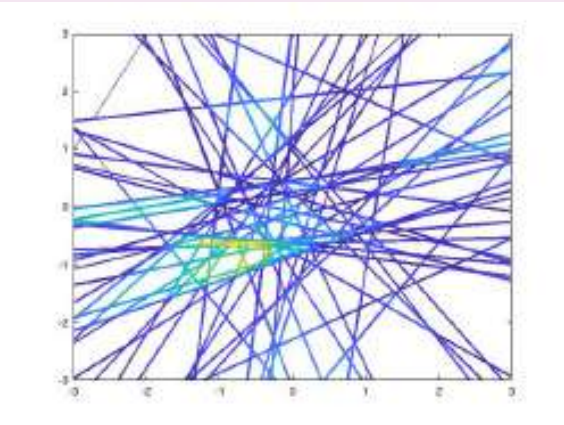
$$\Sigma^{\sigma}_{n_{1:\ell}} = \{ W^{\ell} x^{(\ell)} + b^{\ell},\ W^{\ell} \in \mathbb{R}^{n_{\ell+1} \times n_{\ell}},\ b^{i} \in \mathbb{R}^{n_{\ell+1}} \}$$

ReLU-DNN $\Leftrightarrow$ FEM

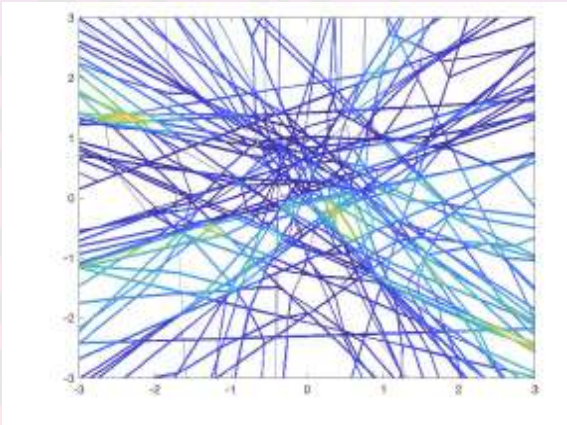
ReLU^k-DNN

$$\Sigma^{k}_{n_{1:\ell}} := \Sigma^{ReLU^k}_{n_{1:\ell}}. \tag{13}$$

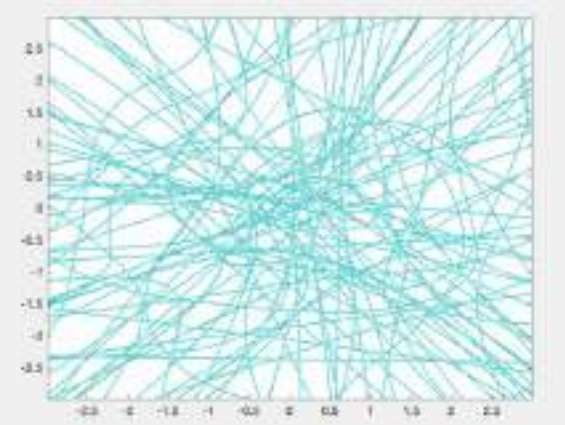
ReLU^k-DNN $= \Sigma^{k}_{n_{1:\ell}} =$ *piecewise polynomials* $\subset H^k(\Omega)$



(1) $\ell = 1, k \geq 1$



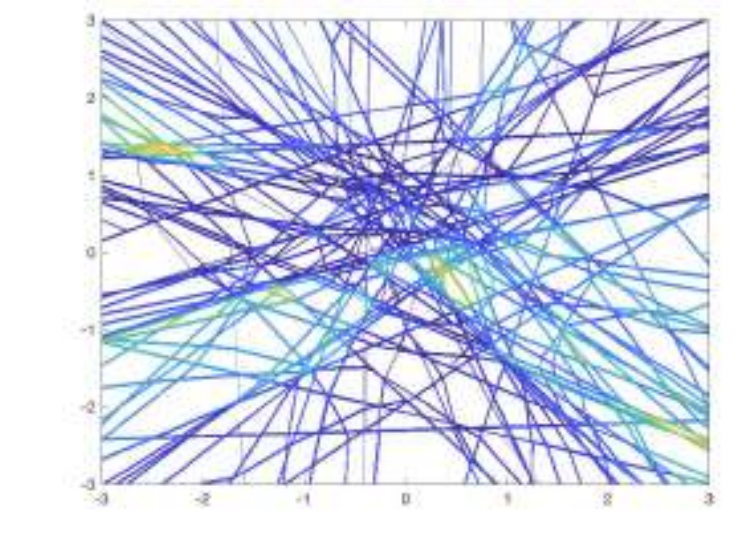
(2) $\ell = 2, k = 1$



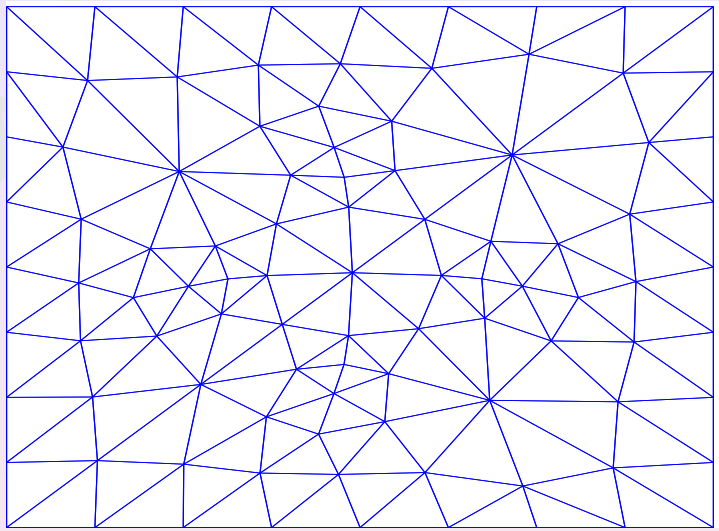
(3) $\ell = 2, k = 2$

ReLU-DNN = FEM

ReLU-DNN $= \Sigma^1_{n_{1:\ell}} \subset$ Linear FEM $\subset H^1(\Omega)$



$\Longleftrightarrow$



- Conforming piecewise polynomials of low order are trivial to construct using neural networks!

- Arora, R., Basu, A., Mianjy, P. & Mukherjee, A. (2016)., He, J., Li, L., Xu, J., & Zheng, C. (2020), He, J., Li, L., & Xu, J. (2022).

Connection of ReLU DNN and linear FEM

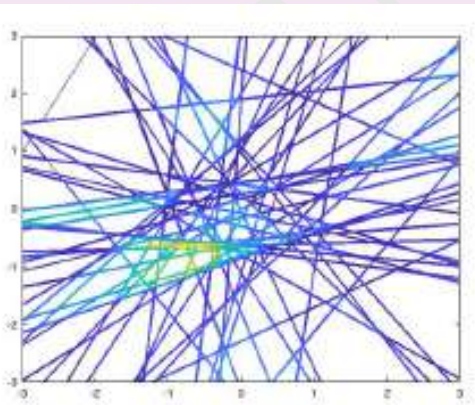
Theorem (He, Li, X and Zheng 2018)

Not all linear FE functions can be represented by a shallow neural network, namely

$$\mathrm{LFE} \not\subset \Sigma_{n_1}^1 .$$

Insights of this proof and result:

- $\Sigma_{n_1}^1$ can not represent locally supported functions;
  - Neither the basis function of LFE.



- Deep is necessary for high dimensional spaces;
- An open question: the optimal depth to represent any LFE functions?

Connection of ReLU DNN and linear FEM

Theorem (He, Li, X and Zheng 2018)

Given a locally convex finite element grid $\mathcal{T}_h$, any linear finite element function with N degrees of freedom, it can be written as a function in $\Sigma^1_{n_{1:J}}$ with

$$J = 2 + \lceil \log_2 d_h \rceil \text{ and } N_{\text{neurons}} = \mathcal{O}(d_h N).$$

where d_h is the largest number of elements that share one vertex.

- Theoretically, $J \leq \lceil \log_2(d+1) \rceil$, however

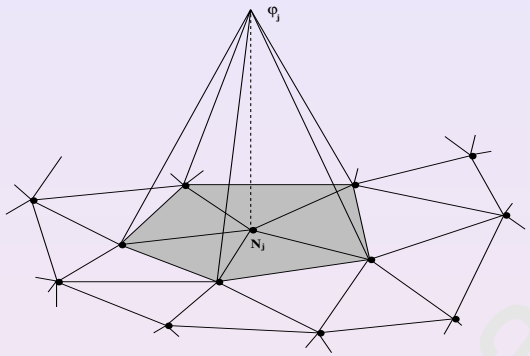
$$N_{\text{neurons}} \leq \mathcal{O}\left(d 2^{N!+dN}\right)$$

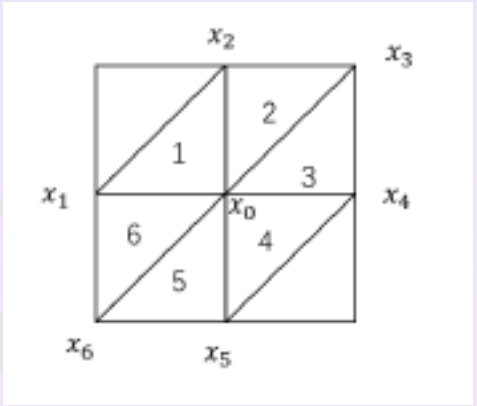
  - Ref: R. Arora, A. Basu, P. Mianjy and A. Mukherjee, 2016

- Key to the proof: How to represent basis functions of LFE more efficiently.
  - See the following example.

A 2D example: FE basis function

$\phi(x)$:





Here g_i is linear in Domain i, and $x_7 = x_1$, satisfying

$$g_i(x_0) = 1 \qquad g_i(x_i) = 0 \qquad g_i(x_{i+1}) = 0$$

$$\phi(x) = \begin{cases} g_i(x), & x \in \text{Domain } i \\ 0. & x \in \mathbb{R}^2 - \overline{x_1 x_2 x_3 x_4 x_5 x_6} \end{cases} \tag{14}$$

Then we have

$$\phi(x) = \mathrm{ReLU}(\min(g_1, g_2, g_3, g_4, g_5, g_6)), \tag{15}$$

if the support of $\phi(x)$ is convex.

Properties of ReLU

Important identities:

$$x = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x), |x| = \mathrm{ReLU}(x) + \mathrm{ReLU}(-x)$$

and

$$\min(a,b) = \frac{a+b}{2} - \frac{|a-b|}{2} = v \cdot \mathrm{ReLU}(W \cdot [a,b]^T) \tag{16}$$

where

$$v = \frac{1}{2}[1,-1,-1,-1] \qquad W = \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Properties of ReLU

Important identities:

$$x = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x), |x| = \mathrm{ReLU}(x) + \mathrm{ReLU}(-x)$$

and

$$\min(a,b) = \frac{a+b}{2} - \frac{|a-b|}{2} = v \cdot \mathrm{ReLU}(W \cdot [a,b]^T) \tag{17}$$

where

$$v = \frac{1}{2}[1,-1,-1,-1] \qquad W = \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Example of DNN and FEM: 2D-FEM basis function

$$
\begin{aligned}
\min(a,b,c) &= \min(\min(a,b),c) \\
&= v\cdot \mathrm{ReLU}\left(W\cdot\begin{pmatrix}\min(a,b)\\ c\end{pmatrix}\right) \\
&= v\cdot \mathrm{ReLU}\left(W\cdot\begin{pmatrix}v\cdot \mathrm{ReLU}(W\cdot[a,b]^T)\\ \mathrm{ReLU}(c)-\mathrm{ReLU}(-c)\end{pmatrix}\right) \\
&= v\cdot \mathrm{ReLU}\left(W\cdot\begin{pmatrix}v\cdot \mathrm{ReLU}(W\cdot[a,b]^T)\\ [1,-1]\cdot\mathrm{ReLU}([1,-1]^T c)\end{pmatrix}\right) \\
&= v\cdot \mathrm{ReLU}\left(W_2\cdot \mathrm{ReLU}(W_1\cdot[a,b,c]^T)\right)
\end{aligned}
$$

where

$$
W_2 = \begin{bmatrix}
\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 1 & -1\\
-\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -1 & 1\\
\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -1 & 1\\
-\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 1 & -1
\end{bmatrix}.
\qquad
W_1 = \begin{bmatrix}
1 & 1 & 0\\
-1 & -1 & 0\\
1 & -1 & 0\\
-1 & 1 & 0\\
0 & 0 & 1\\
0 & 0 & -1
\end{bmatrix}.
$$

Example of DNN and FEM: 2D-FEM basis function

It follows that

$$
\begin{aligned}
g \equiv & \min(g_1, g_2, g_3, g_4, g_5, g_6) = \min(\min(g_1, g_2, g_3), \min(g_4, g_5, g_6)) \\
= & v \cdot \mathrm{ReLU}(W \cdot \begin{bmatrix} \min(g_1, g_2, g_3) \\ \min(g_4, g_5, g_6) \end{bmatrix}) \\
= & v \cdot \mathrm{ReLU}(W \cdot \begin{bmatrix} v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_1, g_2, g_3]^T)) \\ v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_4, g_5, g_6]^T)) \end{bmatrix})
\end{aligned}
$$

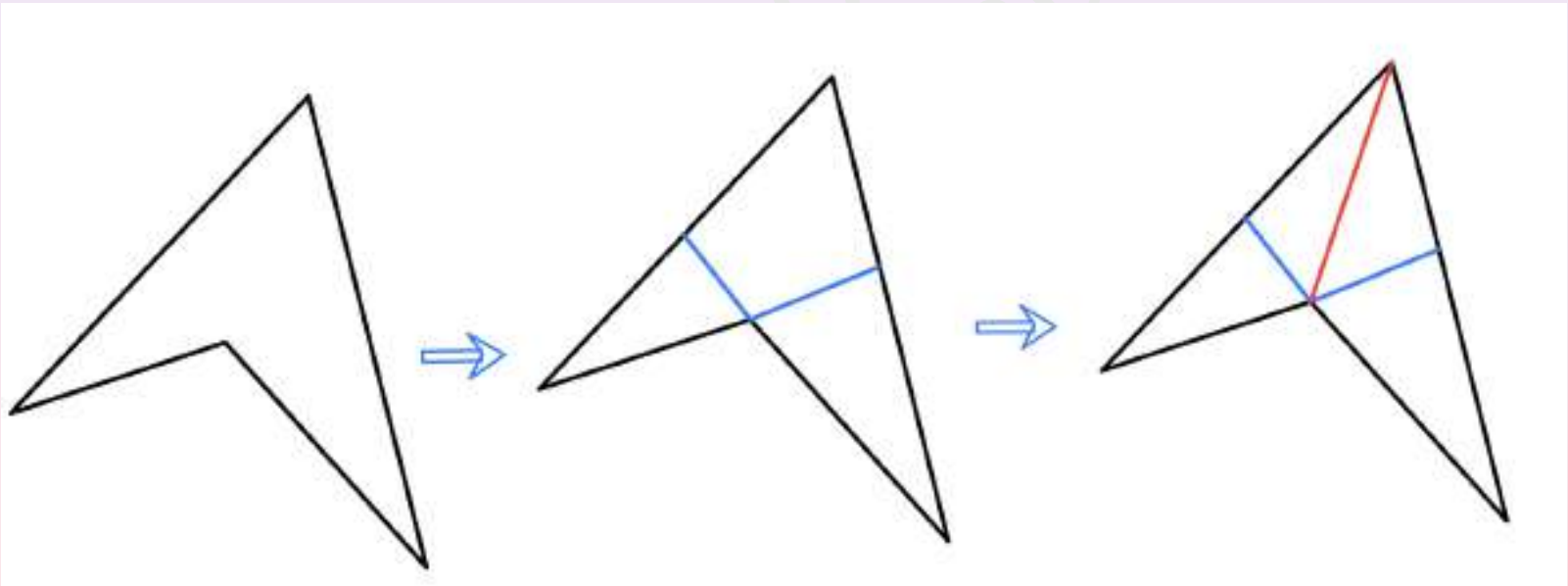
Thus

$$
\phi = \mathrm{ReLU}\left(v \cdot \mathrm{ReLU}\left(W \cdot \begin{bmatrix} v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_1, g_2, g_3]^T)) \\ v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_4, g_5, g_6]^T)) \end{bmatrix} \right) \right) \in \Sigma^1_{n_{1:4}}
$$

ReLU DNN = LFE

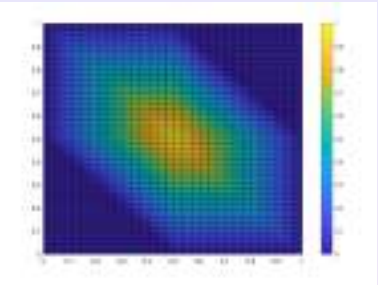
For non-convex support sets of linear finite elements:

- Any polyhedron can be divided by a union of convex polyhedrons;
  - Introduce extra vertexes and faces.
- Divide convex polyhedrons to simplex.



A more compact result for 2D LFE on the uniform grid

The following identity holds on $\mathbb{R}^2$,



$$= \frac{1}{2}\left(g_2\left(\frac{\mathrm{ReLU1}(x)}{2}\right) + g_2\left(\frac{\mathrm{ReLU1}(y)}{2}\right) - g_2\left(\frac{\mathrm{ReLU1}(x)+\mathrm{ReLU1}(y)}{2}\right)\right),$$

where

$$g_2(x) = \quad = \sum_{i=0}^{4} \alpha_i \mathrm{ReLU}\left(x - \frac{i}{4}\right) \in \Sigma_5^1, \tag{18}$$

and

$$\mathrm{ReLU1}(x) := \mathrm{ReLU}(x) - \mathrm{ReLU}(x-1) \in \Sigma_2^1. \tag{19}$$

Question: Other approaches for this representation?

Theorem (He, Li and X 2022)

Let V_h be the LFE space on a uniform mesh on 2D, then

$$V_h \subset \Sigma^1_{n_{1:2}}, \tag{20}$$

and it can be represented explicitly and concisely.

- J. He, L. Li, J. Xu and C. Zheng, 2018: $u(x, y)$ can not be reproduced by one hidden layer.
- R. Arora, A. Basu, P. Mianjy and A. Mukherjee, 2016: $u(x, y) \in \Sigma^1_{n_{1:2}}$, but it is almost impossible to represent $u(x, y)$ explicitly.

ReLU-DNN = LFE

Conclusion:

Any linear finite element function can be represented by a deep ReLU neural network with a relatively deeper and narrower structure.

Theorem

$$\textit{ReLU-DNN} = \mathrm{LFE}$$

Shallow ReLU^k neural networks and polynomials

Theorem (X 2020, Chen, He and Hao 2022)

Polynomials of order k in $\mathbb{R}^d$ $\subset \Sigma_{n_1}^k$.

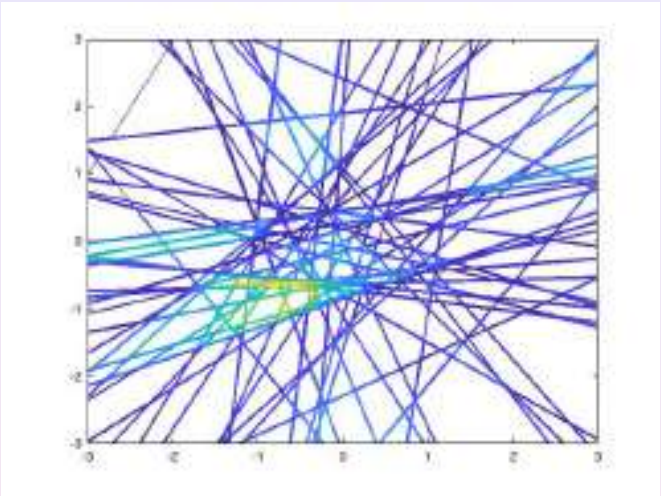
Sketch of the proof:

- $$x^k = \mathrm{ReLU}^k(x) + (-1)^k \mathrm{ReLU}^k(-x), \quad \forall x \in \mathbb{R}$$
- Generalized Vandermonde determinant identity $\rightarrow$ homogeneous polynomials on $\mathbb{R}^d$ with order k
- $$x^{k-1} = \sum_{i=0}^{k} a_i (x-i)^k$$
- Generalized Vandermonde determinant identity $\rightarrow$ homogeneous polynomials on $\mathbb{R}^d$ with order $k-1$
- Repeat for $k-2$
- ...

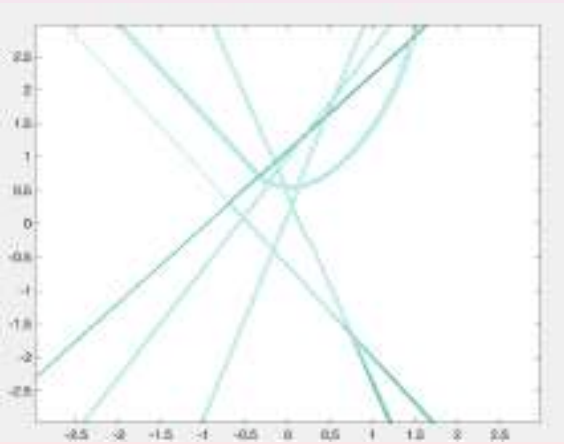
Deep ReLU^k neural networks

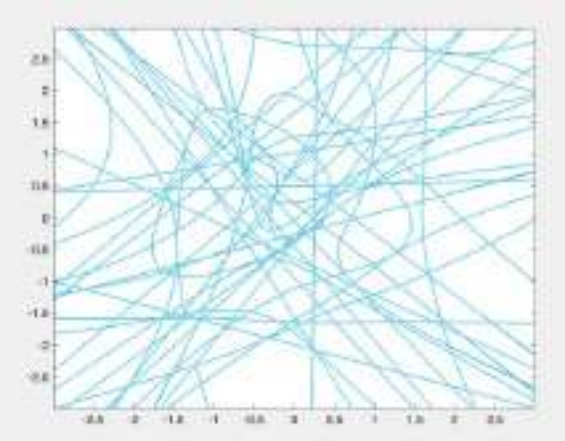
$\Sigma^k_{n_{1:\ell}}$ for $k \geq 2$: Piecewise polynomials :

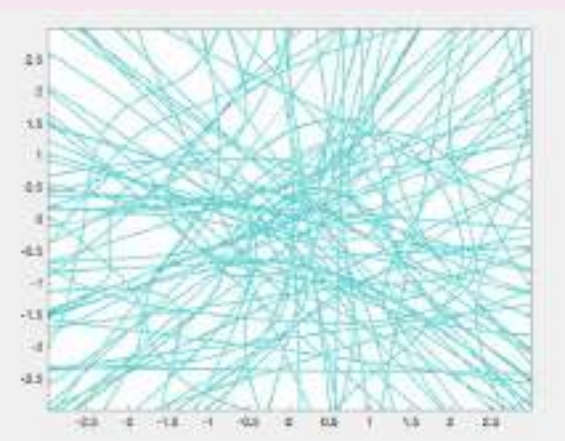
- "Simplex" (global) elements: any $k \geq 2$ but $\ell = 1$



- "Curved" elements: any $k \geq 2$ and $\ell \geq 2$:







Deep ReLU^k neural networks

1. Best possible error estimate $\mathcal{O}(n^{m-(k+1)} \log n)$
2. If $k \geq 2$, recover all polynomials as ℓ increase:
   - For $k = 2$, $x^2 \in \Sigma_2^2$;
   - $xy = \frac{1}{4}\left((x+y)^2 - (x-y)^2\right) \in \Sigma_{n_1}^2$;
   - Polynomial with order n on $R^d \subset \Sigma_{n_{1:\ell}}^2$ with $\ell \leq \log_2(n)$;
   - Spectral accuracy for smooth functions.
3. Possible multi-scale adaptivity features (?):
   - local singularity.
   - global smoothness

- Ref: J. Siegel and J. Xu (2019); B. Li, S. Tang, and H. Yu (2020).

Lower bound for conforming elements

Theorem

Assume that V_h^k is a finite element of degree k on quasi-uniform mesh $\{\mathcal{T}_h\}$ of $\mathcal{O}(N)$ elements. Assume u is sufficiently smooth and not piecewise polynomials, then we have

$$c(u)N^{-\frac{k}{d}} \leq \inf_{v_h \in V_h^k} \|u - v_h\|_{H^1(\Omega)} \leq C(u)N^{-\frac{k}{d}} = \mathcal{O}(h^k). \tag{21}$$

In general

$$\inf_{v_h \in V_h^k} \|u - v_h\|_{H_h^m(\Omega)} \eqsim N^{\frac{m-(k+1)}{d}} = \mathcal{O}(h^{k+1-m}). \tag{22}$$

Ref: Q. Lin, H. Xie and J. Xu , Lower Bounds of the Discretization Error for Piecewise Polynomials, Math. Comp., 83, 1-13 (2014)

Proof ($k = 1$)

Let Π_2 be quadratic interpolation. For any linear FE $v_h \in V_h$,

$$\begin{aligned} |u|_2 = |u - v_h|_{2,h} &\leq |u - \Pi_2 u|_{2,h} + |\Pi_2 u - v_h|_{2,h} \\ &\leq C_1 h |u|_3 + C_2 h^{-1} \|\Pi_2 u - v_h\|_1 \\ &\leq C_1 h |u|_3 + C_2 h^{-1} \|\Pi_2 u - u\|_1 + C_2 h^{-1} \|u - v_h\|_1 \\ &\leq C_1 h |u|_3 + C_3 h^{-1} h^2 |u|_3 + C_2 h^{-1} \|u - v_h\|_1 \end{aligned} \tag{23}$$

namely

$$|u|_2 - (C_1 + C_3) h |u|_3 \leq C_2 h^{-1} \|u - v_h\|_1 . \tag{24}$$

Noting that $v_h \in V_h$ is arbitrary, then for sufficiently small h, we have

$$\inf_{v_h \in V_h} \|u - v_h\|_1 \geq Ch = CN^{-\frac{1}{d}} \tag{25}$$

since $h = CN^{-\frac{1}{d}}$ for quasi-uniform mesh.

Approximation properties: DNN versus AFEM

Shallow NN:

$$\inf_{u_N \in \Sigma_N^k} |u - u_N|_{1,\Omega} \lesssim N^{-\frac{1}{2}} \int_{\mathbb{R}^n} |\omega|^2 |\hat{u}(\omega)| d\omega.$$

Adaptive FEM: [similar for N-term wavelets approximation]

$$\inf_{\dim V_N = N} \inf_{u_N \in V_N} |u - u_N|_{1,2,\Omega} \lesssim N^{-\frac{1}{d}} \|u\|_*$$

Observation:

- For $d \geq 3$, DNN seems to provide "better" asymptotic approximation than AFEM:

$$N^{-\frac{1}{2}} \ll N^{-\frac{1}{d}}$$

On the curse of dimensionality

Example: $d = 100$

$$\frac{1}{\sqrt{N}} \ll N^{-\frac{1}{100}}$$

Sharpness of AFEM estimate

Example: $u = |x|^2$

$$N^{-\frac{1}{d}} \lesssim \inf_{\dim V_N = N} \inf_{u_N \in V_N} |u - u_N|_{1,2,\Omega} \lesssim N^{-\frac{1}{d}}$$

$$N^{-\frac{2}{d}} \lesssim \inf_{\dim V_N = N} \inf_{u_N \in V_N} \|u - u_N\|_{0,\infty,\Omega} \lesssim N^{-\frac{2}{d}}$$

optimal grid: uniform grid.

d=1

- Grid $\mathcal{T} := \{0 = x_0 < x_1 < \cdots < x_{N+1} = 1\}$
- Local basis

$$\phi_i = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}}, & x \in (x_{i-1}, x_i], \\ \frac{x - x_{i+1}}{x_i - x_{i+1}}, & x \in (x_i, x_{i+1}], \\ 0, & \text{others.} \end{cases}.$$

- Interpolation: $I_{\mathcal{T}} u = \sum_i u_{x_i} \phi_i$.

Approximation of x^2 by multilevel FEM

Let $f(x) = x^2$ on $[0,1] \subset \mathbb{R}$. Recall

$$(I - I_J)f = \mathcal{O}(N^{-2})$$

with

$$I_J f = \sum_{i=0}^{2^J} f(x_i)\phi_i^J(x).$$

We write

$$I_J f = I_0 f + \sum_{k=1}^{J} (I_k - I_{k-1})f$$

Notice that $I_0 f = x$ and

$$(I_k - I_{k-1})f = -4^{-k} \sum_{j=1}^{2^{k-1}} \phi_{2j-1}^k(x).$$

Non-local basis: $g_i = \sum_{j=1}^{2^{i-1}} \phi_{2j-1}^i(x)$

$$\|x^2 - (x - \sum_{i=1}^{L} 4^{-i} g_i)\|_{0,\infty,\Omega} = \mathcal{O}(N^{-2}), \tag{26}$$

with $L = \log_2 N$!

Ref: D.Yarotsky 2017, W. E and Q. Wang 2018, J. He and J. Xu 2019.

Diagram for basis functions g_i

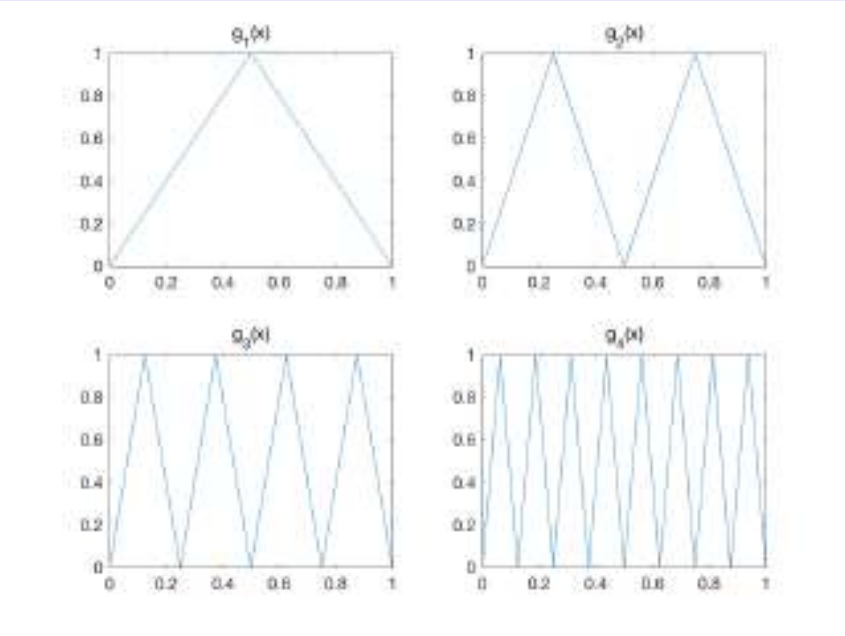


Figure: The figure of $g_i(x)$

Important observation: composition property:

$$g_i(x) = \underbrace{g_1 \circ g_1 \circ \cdots \circ g_1}_{i}(x) \in \Sigma^1_{n_{1:i}}. \tag{27}$$

Comparison between DNN and FEM on x^2

Recall: Approximating x^2 by AFEM

Theorem

$$\inf_{v(x)\in V_L} \max_{x\in[0,1]} |x^2 - v(x)| = \mathcal{O}(L^{-2}), \tag{28}$$

where V_L denotes the AFEM space with L elements on $[0,1]$.

Approximating x^2 by ReLU DNNs

Theorem

There exists $v(x) \in \Sigma^1_{n_{1:L}}$, such that

$$\max_{x\in[0,1]} |x^2 - v(x)| = \mathcal{O}(4^{-L}), \tag{29}$$

where $n_\ell \leq 4$ for $\ell = 1 : L$.

Ref: D.Yarotsky 2017, W. E and Q. Wang 2018, J. He, Lin Li and J. Xu 2022.

From x^2 to polynomials and smooth functions

We note that

$$g_i \in \Sigma^1_{n_{1:i}}, \quad \text{where} \quad n_j \leq 4, j = 1 : i.$$

Using the following identity

$$xy = 2\left(\left(\frac{x+y}{2}\right)^2 - \left(\frac{x}{2}\right)^2 - \left(\frac{y}{2}\right)^2\right)$$

Theorem

For any polynomial $p(x) = \sum_{|\boldsymbol{k}|\leq p} a_{\boldsymbol{k}} \boldsymbol{x}^{\boldsymbol{k}}$ with order p on $[0,1]^d$, there exists $v(x) \in \Sigma^1_{n_{1:CL}}$, such that

$$\max_{\boldsymbol{x}\in[0,1]^d} |p(\boldsymbol{x}) - v(\boldsymbol{x})| \leq 4(p-1)4^{-L} \sum_{\boldsymbol{k}} |a_{\boldsymbol{k}}|, \tag{30}$$

where $n_\ell \leq 2d + 4$ and $C = 3(p-1)\binom{p+d}{d}$ that is independent from L.

Corollary

ReLU DNN can approximate smooth functions as good as polynomials !

Ref: D.Yarotsky 2017, W. E and Q. Wang 2018, J. He, L. Li and J. Xu 2022, J. He and J. Xu 2023.

DNN versus FEM

FEM

1. Local basis functions
2. Multilevel basis functions give more "global" basis functions:
   - Sparse grid
   - Combination of hierarchical basis functions

DNN

1. It contains all AFEM
2. It generates "global" basis functions (from composition)
   - "Features" are mostly "global"?
3. Nonlinearity function generates redundant number of basis functions:

Lemma (He, Lin, X and Zheng 2018, Siegel and X 2019)

$\{\sigma(w_i \cdot x + b_i)\}_{i=1}^N$ are linearly independent if σ is not polynomials and $\begin{pmatrix} w_i \\ b_i \end{pmatrix}, \begin{pmatrix} w_j \\ b_j \end{pmatrix} \in \mathbb{R}^{d+1}$ are linear independent for any $i \neq j$.

Concluding remarks

Summary

- Finite elements $\Rightarrow$ neural networks
- Approximation properties of Cosine and ReLU^k shallow NNs
- ReLU DNN $\equiv$ linear finite elements
  - Local adaptivity
  - Super-approximation for polynomials
- ReLU^k-DNN can recover all polynomials

Open problems:

- Optimal depth to recover any LFE on $\mathbb{R}^d$
- Understand ReLU^k DNNs from adaptive finite elements (both mesh and degree of polynomials)
- Optimal representation of finite element functions using DNNs
- $\ldots$

Thank You !