賴明治　　教授執行　108　年度國家講座成果報告表

該學年度國家講座執行情形：

一、教學課程綱要

1 微分方程(大學部一學期必修(108 學年度第一學期，60 人選此門課))

本課程為針對本系大二生的必修課程，其中有一位大一相當優秀的學生提前來修課，並在學期末時向我提出下學期後隨我做專題研究的請求。

Some basic mathematical models from biology and mechanical systems; First-order differential equations using method of the integrating factors and separable differential equations; Exact differential equations; Second-order linear differential equations with constant coefficients; Solutions of linear homogeneous equations and Wronskian; Methods of undetermined coefficients and variation of parameters for non-homogeneous equations; Higher-order linear differential equations; Series solutions of second-order linear differential equations; Systems of first-order linear differential equations, the fundamental matrix of solutions; diagonalization of the matrix and Jordan form with applications for solving linear system of differential equations;　The Laplace transform.

2 流體力學專題(107 學年度第二學期，研究生選修)

Basic fluid mechanics and the derivation of Navier-Stokes equations. Introduction to the projection method for solving the incompressible Navier-Stokes equations. Immersed boundary method and applications to the interfacial flows, Immersed interface method and its implementations, Interfacial flows with insoluble and soluble surfactant: mathematical models, numerical schemes and applications

3 微分方程(大學部一學期必修(107 學年度第一學期，25 人選此門課))

本課程為針對本系大二生的必修課程，也有幾位他系的學生來修課，除此之外尚有一位大一的學生來修課，今年(108 年)初他也提出跟我做專題研究的請求。

Some basic mathematical models from biology and mechanical systems; First-order differential equations using method of the integrating factors and separable differential equations; Exact differential equations; Second-order linear differential equations with constant coefficients; Solutions of linear homogeneous equations and Wronskian; Methods of undetermined coefficients and variation of parameters for non-homogeneous equations; Higher-order linear differential equations; Series solutions of second-order linear differential equations; Systems of first-order linear differential equations, the fundamental matrix of solutions; diagonalization of the matrix and Jordan form with applications for solving linear system of differential equations;　The Laplace transform.

二、研究重點及方向

今年的的研究重點及方向，除了延續先前流體界面的問題外，本人也嚐試開始在**數學與機器學習**這個前沿的研究議題上組成研究團隊並開展新的研究方向，希望帶給國內數學界在這方面研究一個領頭的作用。由於近年來人工智慧(AI)的應用已經愈來越普及至我們的日常生活中，在可以預見的未來，這個趨勢將是進步快速且是不可逆轉的，而機器學習正是人工智慧成功的最主要關鍵。機器學

習的觀念存在已久，然而真正實現卻是在大數據的獲得及電腦計算能力的強化所致。我想大概沒有人會否認數學在機器學習上所扮演的重要角色，因此對於這個趨勢，我們針對利用機器學習來解決傳統數學問題提出一些可以研究的議題 (詳細內容請參考附件)。

我們將研究機器學習應用在下列三種重要且有趣的傳統應用數學問題上面。(1) 高維度的偏微分方程數值解 (如多個粒子的薛丁格方程式，或多資產的 Black-Scholes 方程)，(2) 改善傳統流體力學的數學模型與數值方法 (如 Turbulence 模型及天氣預測的模型及參數設定)，(3) 解有關在物理學上由資料驅動的非線性偏微分方程。本人認為如果只用傳統方法不可行的話，機器學習將會是解決上述問題的一個可以利用的工具。另外，我們也將研究機器學習演算法在數學理論上收斂性的研究，以及機器學習在安全計算上的實務應用，當然我們的研究團隊也會做一些實務上的應用，目前已經跟榮總麻醉科醫生有一個問題在合作。

此外，我也對目前機器學習在數學學門可以規畫發展的次領域提出一些個人的看法，譬如:數值優化方法的發展及不確定性的量化。機器學習中的一個數學支柱便是數值優化，計算系統中的優化參數使產生的模型或結果預測更為精準。另外由於系統的不確定性在模型，數值方法或實驗上的參數取得都有可能發生，因此量化並控制這些不確定性產生的結果，將對精準的預測更有助益。然而這些次領域的發展，則有藉於科學計算，機率與統計的跨領域合作。

三、學校資源配合狀況

（一）學校對於講座主持人教學研究各項資源配合內容

本校除提供相關的各項資源配合主持人的開課要求外，亦提供彈性薪資獎勵與補助與國際高引用學者合作的相關費用，在此感謝本校的各項配合措施。

（二）國家講座開設跨校性選修課程、辦理全國巡迴講座並宣揚研究教學成果情形

如在二(研究重點及方向)所述，主持人今年在執行國家講座期間，已經開始嚐試發展在**數學與機器學習**這方面的研究，並在本系建立**數學與機器學習實驗室**(如附件所示)，組成交通大學數學與機器學習的研究團隊，其成員包括本校智慧科學暨綠能學院魏澤人副教授，本系林得勝助理教授，徐仕軒與呂揚凱兩位博士後，楊俊彥博士生，及一位碩士生和兩位大四生參與專題研究，以及**跨校合作的團隊**成員如清大數學的朱家杰助理教授，及中原大學的謝天長助理教授等，另外還有一位**跨國際**的合作者(Dr. Kian Chuan Ong, Fields Institute, Canada)。下一步當然是辦理巡迴講座並宣揚此一研究議題及方向，然而由於這對主持人而言也是一個新的嘗試，非短期可見成果，必須是經過一段時間的醞釀，方能整理出一系列相關的演講講義才能對全國辦理大規模的巡迴演講，目前預計在二月中前往高雄大學應用數學系所辦的研討會，對年輕的研究學者及大學部學生，做一個**深度學習的數學架構**簡介(A brief introduction to deep learning network from applied mathematics aspect，其講義如附件)。當然，主持人也積極推動國內在這方面的發展及協助舉辦研討會(如 2019/12/28 NCTS workshop on Scientific Computing and Machine Learning)。另外，主持人擔任理事長的台灣工業與應用數學會，也將今年年會主題定調與 AI 及機器學習相關的數學方法及應用，將於五月份舉辦的年會當中將邀請 Hans De Sterck 教授擔任大會主講有關 AI 與科學計算的研究議題，另外還有 AI 與 Data Science, Math and Machine Learning, Uncertainty Quantification 等相關的迷你論壇，這些皆是呼應主持人在執行此

計畫的實際推行步驟。當然，對於推動數學研究往下扎根方面，我們也持續協助成功大學舉辦高中數學建模黑克松活動。

四、執行效益自我評估

主持人在執行國家講座期間，在教學方面，開設大學部的基礎課程"微分方程"，為本系的大二必修課程，兩學期共吸引了 85 位學生(108 第一學期礙於上課教室的選課人數上限)來選課，除應用數學系的學生外，尚有其他系所如機械，運管等系的學生來選課，顯現主持人在開設基礎課程方面有一定的吸引力，教學內容以數學模型為前導，再配合如何求解方程式，最後再以解釋其物理意義，深入淺出的方式成為上課的主軸，並配合大一所學的課程讓學生能融會貫通，值得一提的是有兩位在大一就提前來修本課程的本系頂尖學生，隨後都將與我做專題研究。另外，主持人也帶領兩位即將畢業的大四學生做專題研究，進行實際寫程式的訓練，並於期末分組做專題報告。就研究成果方面而言，在執行講座期間，2019 年共有 6 篇 SCI 文章發表(參考電子檔附件)，包含 JCP，SISC 與 JFM 等在科學計算與流體力學的頂尖期刊，整體研究成果豐碩。此外，再加上上述所言開展的新穎研究議題數學與機器學習，主持人除原本執行的科技部三年期計畫之外，再獲得科技部兩年共 4 百多萬的研究計畫(數學與機器學習:方法與應用，108/11/01-110/07/31)，顯現主持人所欲推廣發展的研究議題也獲科技部的認同，雖然目前計畫剛執行尚未有具體的研究成果，但效果是可預期的，就執行國家講座效益而言，個人認為還算符合預期。

五、檢討與建議

目前國內數學界對機器學習與數學的相關研究幾乎毫無著墨，未來希望能夠盡快整理出數學與機器學習相關的基本知識的講義，並配合研究實際的問題，帶領團隊在學術研究及實務應用(如產學合作上)有所進展，在平常舉行定期的小型討論班，暑期利用國家理論科學研究中心的資源幫忙籌畫大專生暑期研究計畫，或於學期中開設專題課程進行遠距教學，讓更多的學生能夠受益。在研究方面，除了在目前所研究的計算流體力學的議題中加入探討 machine learning 的元素，也將再利用機器學習來解資料不完整的偏微分方程進行研究，目前已有初步的想法，希望在未來的一年能有實際的研究成果產出。

附件	□成果報告 1 冊（含實施計畫、課程講義、回饋分析表或相關佐證資料）及成果電子檔 1 份 □收支結算表 1 份(填列附件)

填表人： 賴明治 （簽章）

填表日期： 2020 年 1 月 31 日

A brief introduction to deep learning network from applied math aspect

Ming-Chih Lai
mclai@math.nctu.edu.tw

Department of Applied Mathematics
National Chiao Tung University
Taiwan

The Machine Learning (ML) and Artificial Intelligence (AI) Eruption

- In 2012, AlexNet architecture achieved groundbreaking image classification results
  - Computer trained "learns" to identify cats
  - Dramatic improvement in object identification
- In 2015, AlphaGO developed by the Google DeepMind team to play game GO by beating the human European champion by 5:0
- Continued dramatic progress in many applications
  - Text translation
  - Image recognition
  - Speech recognition
  - Targeted advertising
  - Drug discovery
  - Credit rating for loans
  - Autonomous vehicles
- No end in sight!

Introduction to Artificial Neural Network

- What is a deep neural network?
- How is a network trained? weights, biases, and activation
- What is the stochastic gradient (SG) method? Some literatures called stochastic gradient descent (SGD) method.
- A complete back propagation algorithm
- Matlab classification example

Reference: C. F. Higham & D. J. Higham, Deep Learning: An Introduction for Applied Mathematicians, SIAM Review, Vol. 61, No. 4. pp. 860-891, 2019

Why should I care about ML?

- What can the ML do for our current research?
- What can we do for ML itself?

My very humble opinions from applied and computational mathematics point of view:

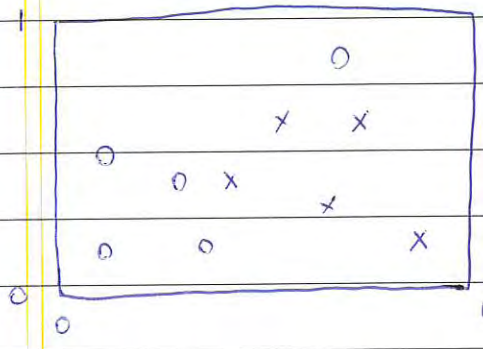
- Help solving high-dimensional PDEs to circumvent the "curse of dimensionality"
  - Many-Electron Schrödinger equation; N-particles means 3N spatial dimensions for the wave function so the kinetic energy operator involves the Laplacian operator of 3N dimensions
  - The nonlinear Black-scholes equation for pricing financial derivatives, in which many underlying financial assets (so many independent variables) are considered
- Help discovering the data-driven PDEs
  - Given the observed data, can we extract the underlying physical laws from the data?
  - Given the form of PDE with unknown parameters, can we find the parameters that best describe the observed data?

What can we do for ML itself?

- Why the artificial neural networks are so successful in may applications? The reasons behind the success are not fully understood.
- How do we decide on the structure of our network? How many hidden layers and how many neurons in each layer?
- Deep neural networks usually results in an optimization problem which is highly nonlinear and non-convex, what are the most successful optimization methods for large-scale problem, and why?
- Improve existing deep learning strategies: Which loss function? Which regularization?
- Lack of mathematical foundation for the remarkable efficacy of ML. Optimization algorithm with convergence guarantee.

Deep Learning : An introduction from applied math aspect

A simple example of ANN (Artificial Neural Network)



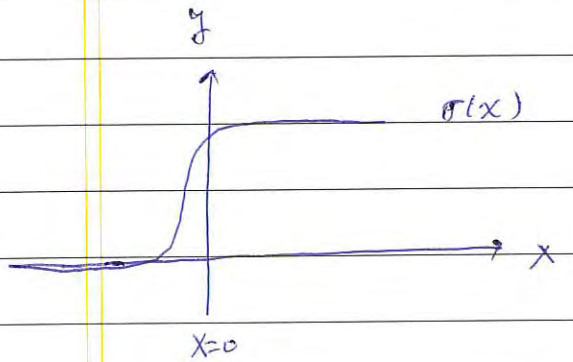
"o" Category A (successful oil drilling site)

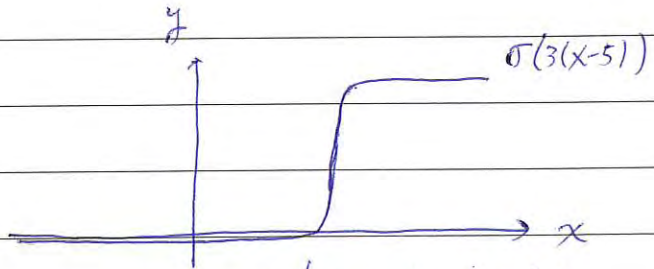
"x" Category B (otherwise)

Question : Can we use the data to categorize a newly proposed drilling site ?

The sigmoid function (as an activation function)

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





weighting & biasing (scaling & shifting)

And $\sigma'(x) = \sigma(x)(1-\sigma(x))$ Easy to compute the derivative

Notation : $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ for $z \in \mathbb{R}^m$

$(\sigma(z))_i = \sigma(z_i)$ vectorized version

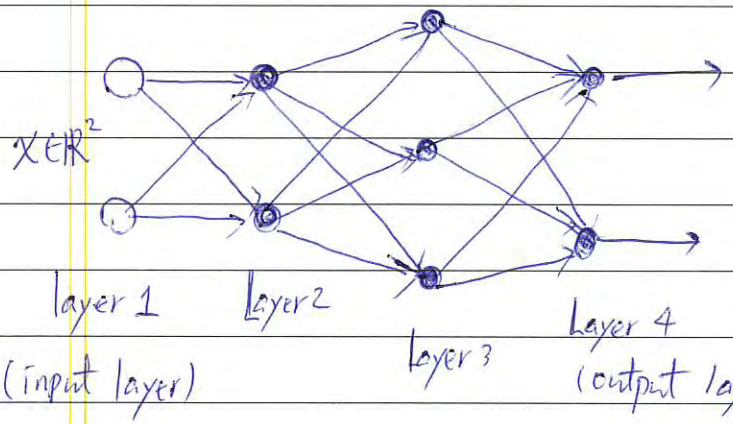
NCTS
National Center for Theoretical Sciences

With this notation, we set up layers of neurons.

In each layer, every neuron outputs a single real number, which is passed to every neuron in the next layer. At the next layer, each neuron forms its own weighted combination of these values, adds its own bias, and applies the activation function.

ie $\sigma(Wa+b)$, W is a matrix, b is a vector and a is the input vector produced at the previous layer.

i-th neuron can be written as $\sigma\left(\sum_j w_{ij} a_j + b_i\right)$



Input data $x \in \mathbb{R}^2$, $W^{[2]} \in \mathbb{R}^{2\times 2}$ $b^{[2]} \in \mathbb{R}^2$

Layer 2 has 2 neurons

The output from layer 2 is

$$\sigma\left(W^{[2]}x + b^{[2]}\right) \in \mathbb{R}^2$$

Similarly, the output from layer 3 is $\sigma\left(W^{[3]}\sigma(W^{[2]}x+b^{[2]})+b^{[3]}\right)$

$W^{[3]} \in \mathbb{R}^{3\times 2}$, $b^{[3]} \in \mathbb{R}^3$ $\in \mathbb{R}^3$

NCTS
National Center for Theoretical Sciences

The fourth layer (output layer) from the overall network has the form

$$F(x) = \sigma\left(W^{[4]}\sigma\left(W^{[3]}\sigma\left(W^{[2]}x + b^{[2]}\right) + b^{[3]}\right) + b^{[4]}\right) \in \mathbb{R}^2 \quad (1)$$

$$W^{[4]} \in \mathbb{R}^{2\times 3} \qquad b^{[4]} \in \mathbb{R}^2$$

The expression (1) defines a function $F: \mathbb{R}^2 \to \mathbb{R}^2$ in terms of 23 parameters (all the weights and biases)

$W^{[2]} \in \mathbb{R}^{2\times 2}$, $b^{[2]} \in \mathbb{R}^2$ $\Rightarrow$ 6 parameters of Layer 2

$W^{[3]} \in \mathbb{R}^{3\times 2}$, $b^{[3]} \in \mathbb{R}^3$ $\Rightarrow$ 9 parameters of Layer 3

$W^{[4]} \in \mathbb{R}^{2\times 3}$, $b^{[4]} \in \mathbb{R}^2$ $\Rightarrow$ 8 parameters of Layer 4

Recall our goal is to produce a classifier based on the data Fig 1.

Require: $F(x)$ to be close to $\begin{pmatrix}1\\0\end{pmatrix}$ in A

$F(x) \approx \begin{pmatrix}0\\1\end{pmatrix}$ in B

Then given a new point $x \in \mathbb{R}^2$, it would be reasonable to classify it according to the largest component of $F(x)$ i.e $F_1(x) > F_2(x)$ in A

$F_1(x) < F_2(x)$ in B

NCTS
National Center for Theoretical Sciences

Denote 10 data points in Fig 1, $\{x^{(i)}\}_{i=1}^{10}$, we use

$$y(x^{(i)}) = \begin{cases} \begin{pmatrix} 1 \\ 0 \end{pmatrix} & \text{if } x^{(i)} \in A \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \text{if } x^{(i)} \in B \end{cases}$$

We need our cost function (objective function in optimization)

(2) $$\text{cost} = \frac{1}{10} \sum_{i=1}^{10} \frac{1}{2} \| y(x^{(i)}) - F(x^{(i)}) \|_2^2$$

Note: $\text{Cost}(W^{[2]}, W^{[3]}, W^{[4]}, b^{[2]}, b^{[3]}, b^{[4]})$

Choosing the weights and biases in a way to minimize the cost function is called "training" the network

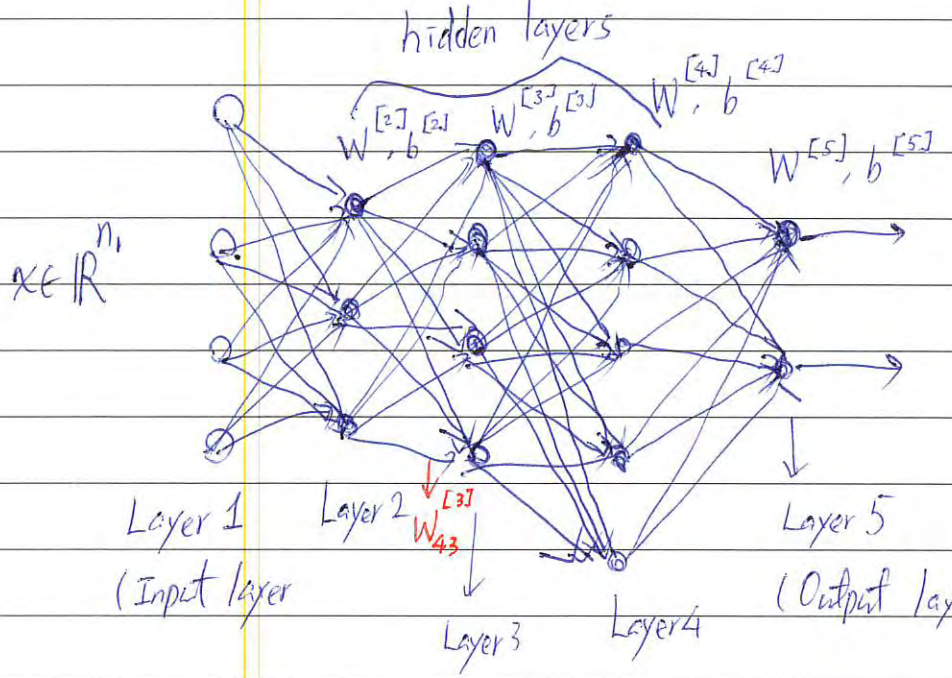
Matlab tool box to minimize the cost function (2) over the 23 parameters "lsqnonlin"

However, the underlying optimization problem, minimizing a "nonconvex" objective function over 23 variables is already fundamentally difficult. We cannot guarantee to find the global minimum of a nonconvex function.

(eg. Some initial guess for the weights and biases cannot find an acceptable solution).

NCTS
National Center for Theoretical Sciences

Neural Network general setup



(Fig 2)

Deep Learning : many hidden layers

Suppose that we have L layers with layers 1 and L being the input and output layers, respectively.

At layer ℓ, contains n_ℓ neurons.

" $1, 2, \ldots L$

n_1 : dimension of the input data

n_L : dimension of the output data.

Need to find $F : \mathbb{R}^{n_1} \to \mathbb{R}^{n_L}$

$W^{[\ell]} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$: the matrix of weights at layer ℓ

$W^{[\ell]}_{jk}$: the weight that neuron j at layer ℓ applies to the output from neuron k at layer $\ell-1$

$b^{[\ell]} \in \mathbb{R}^{n_\ell}$: the vector of biases at layer ℓ

$b^{[\ell]}_j$: the bias of neuron j at layer ℓ

NCTS
National Center for Theoretical Sciences

In Fig 2, $L=5$, $n_1=4$, $n_2=3$, $n_3=4$, $n_4=5$, $n_5=2$,

so $W^{[2]} \in \mathbb{R}^{3\times 4}$ $W^{[3]} \in \mathbb{R}^{4\times 3}$, $W^{[4]} \in \mathbb{R}^{5\times 4}$, $W^{[5]} \in \mathbb{R}^{2\times 5}$

$b^{[2]} \in \mathbb{R}^3$ $b^{[3]} \in \mathbb{R}^4$ $b^{[4]} \in \mathbb{R}^5$, $b^{[5]} \in \mathbb{R}^2$

Given an input $x \in \mathbb{R}^{n_1}$,

Denote $a_j^{[\ell]}$: the output or "activation" from neuron j at layer ℓ

$$a^{[1]} = x \in \mathbb{R}^{n_1}$$

$$a^{[\ell]} = \sigma\left(W^{[\ell]} a^{[\ell-1]} + b^{[\ell]}\right) \in \mathbb{R}^{n_\ell}, \quad \ell = 2, 3, \ldots L$$

Once all the $W^{[\ell]}$ and $b^{[\ell]}$ are known, we can feed the input forward through the network to produce an output $a^{[L]} \in \mathbb{R}^{n_L}$

in $\mathbb{R}^{n_1}$

Now suppose we have N pieces of data, denoted by $\{x^{(i)}\}_{i=1}^N$

for which there are given target outputs $\{y(x^{(i)})\}_{i=1}^N$ $\in \mathbb{R}^{n_L}$

Our goal is to minimize the cost function

$$\text{Cost} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \left\| y(x^{(i)}) - a^{[L]}(x^{(i)}) \right\|_2^2 .$$

Note: Cost is a function of $W^{[\ell]}$ and $b^{[\ell]}$

$\ell = 2, \ldots L$

NCTS
National Center for Theoretical Sciences

Stochastic Gradient : $\text{Cost} : \mathbb{R}^s \to \mathbb{R}$, $p \in \mathbb{R}^s$

Steepest descent or gradient descent :

Suppose that our current vector is $p \in \mathbb{R}^s$.

Q: How to choose a perturbation Δp so that $p + \Delta p$ ~~minimize~~ the cost better ?

$$\text{Cost}(p+\Delta p) \approx \text{Cost}(p) + \sum_{r=1}^{s} \frac{\partial \text{Cost}(p)}{\partial p_r} \Delta p_r + O(\Delta p^2)$$

or $$\text{Cost}(p+\Delta p) \approx \text{Cost}(p) + \nabla \text{Cost}(p)^T \Delta p$$

$\nabla \text{Cost}(p)$ ↳ gradient

Choose $\Delta p \propto // - \nabla \text{Cost}(p)$

$p \to p - \eta \nabla \text{Cost}(p)$, $\eta > 0$: a small stepsize called "learning rate"

Let $C_{x^{(i)}} = \frac{1}{2} \| y(x^{(i)}) - a^{[L]}(x^{(i)}) \|_2^2$

then $\nabla \text{Cost}(p) = \frac{1}{N} \sum_{i=1}^{N} \nabla C_{x^{(i)}}(p)$

However, when N and s are large, computing the gradient vector at every iteration of the steepest descent method is prohibitively expensive.

NCTS
National Center for Theoretical Sciences

Version ① Stochastic gradient with replacement

(1). Choose an integer i uniformly at random from $\{1, 2 \ldots N\}$

(2) Update

$$p \to p - \eta \nabla C_{X^{(i)}}(p)$$

(3) repeat (1) up to some iterations

In words, at each step, the stochastic gradient method uses one training points to represent the full training set.

Version ② Stochastic gradient without replacement

(1) Shuffle the integers $\{1, 2 \ldots N\}$ into a new order $\{K_1, K_2 \ldots K_N\}$

(2) For $i = 1$ upto N, update

$$p \to p - \eta \nabla C_{X^{(K_i)}}(p)$$

This is called "completing an epoch"

Version ③ with or Minibatch without replacement ($N = km$, complete k distinct minibatches)

For some $m << N$

(1) Choose m integers $K_1, K_2 \ldots K_m$ uniformly random from $\{1, 2, \ldots N\}$

(2) Update

$$p \to p - \eta \frac{1}{m} \sum_{i=1}^{m} \nabla C_{X^{(K_i)}}(p)$$

In this iteration, the set $\{x^{(K_i)}\}_{i=1}^{m}$ is known as a minibatch.

NCTS
National Center for Theoretical Sciences

Back propagation:

How to compute $\nabla C_{x^{(i)}}(p)$? p represents $w_{jk}^{[\ell]}$, $b_j^{[\ell]}$

Fix $x^{(i)}$, or

(*) $C = \frac{1}{2}\|y - a^{[L]}\|_2^2 = \frac{1}{2}\sum_{k=1}^{n_L}(y_k - a_k^{[L]})^2$.

Recall that $a^{[L]}$ is the output from the artificial neural network

The dependence of C on the weights and biases arises only through $a^{[L]}$.

Let $z^{[\ell]} = W^{[\ell]} a^{[\ell-1]} + b^{[\ell]} \in \mathbb{R}^{n_\ell}$, $\ell = 2, \dots L$

$z_j^{[\ell]}$: weighted input for neuron j at layer ℓ.

$a^{[\ell]} = \sigma(z^{[\ell]})$, $\ell = 2, 3, \dots L$

Let $\delta^{(\ell)} \in \mathbb{R}^{n_\ell}$ be defined by

$\delta_j^{[\ell]} = \frac{\partial C}{\partial z_j^{[\ell]}}$ for $1 \le j \le n_\ell$ and $2 \le \ell \le L$

Define Hadamard product of two vectors ($x, y \in \mathbb{R}^n$) $x \circ y \in \mathbb{R}^n$ (componentwise product) s.t. $(x \circ y)_i = x_i y_i$

Lemma:

① $\delta^{[L]} = \sigma'(z^{[L]}) \circ (a^{[L]} - y)$

② $\delta^{[\ell]} = \sigma'(z^{[\ell]}) \circ (W^{[\ell+1]})^T \delta^{[\ell+1]}$ $\quad 2 \le \ell \le L-1$

③ $\frac{\partial C}{\partial b_j^{[\ell]}} = \delta_j^{[\ell]}$ $\quad 2 \le \ell \le L$

④ $\frac{\partial C}{\partial w_{jk}^{[\ell]}} = \delta_j^{[\ell]} a_k^{[\ell-1]}$ $\quad 2 \le \ell \le L$

NCTS
National Center for Theoretical Sciences

pf:

① $a^{[L]} = \sigma(z^{[L]})$

$$\frac{\partial a_j^{[L]}}{\partial z_j^{[L]}} = \sigma'(z_j^{[L]})$$

$$\delta_j^{[L]} = \frac{\partial C}{\partial z_j^{[L]}} = \frac{\partial C}{\partial a_j^{[L]}} \cdot \frac{\partial a_j^{[L]}}{\partial z_j^{[L]}}$$

$$\frac{\partial C}{\partial a_j^{[L]}} = -(y_j - a_j^{[L]}) \quad (\text{from Eq. (*)})$$

$$= (a_j^{[L]} - y_j)$$

$$\delta_j^{[L]} = \sigma'(z_j^{[L]}) \cdot (a_j^{[L]} - y_j)$$

$$\Rightarrow \quad \delta^{[L]} = \sigma'(z^{[L]}) \circ (a^{[L]} - y)$$

② $$\delta_j^{[\ell]} = \frac{\partial C}{\partial z_j^{[\ell]}} = \sum_{k=1}^{n_{\ell+1}} \frac{\partial C}{\partial z_k^{[\ell+1]}} \cdot \frac{\partial z_k^{[\ell+1]}}{\partial z_j^{[\ell]}}$$

$$= \sum_{k=1}^{n_{\ell+1}} \delta_k^{[\ell+1]} \cdot \frac{\partial z_k^{[\ell+1]}}{\partial z_j^{[\ell]}}$$

$$z_k^{[\ell+1]} = \sum_{s=1}^{n_\ell} w_{ks}^{[\ell+1]} \sigma(z_s^{[\ell]}) + b_k^{[\ell+1]}$$

$$\frac{\partial z_k^{[\ell+1]}}{\partial z_j^{[\ell]}} = w_{kj}^{[\ell+1]} \sigma'(z_j^{[\ell]})$$

$$\delta_j^{[\ell]} = \sum_{k=1}^{n_{\ell+1}} w_{kj}^{[\ell+1]} \delta_k^{[\ell+1]} \sigma'(z_j^{[\ell]})$$

$$\Rightarrow \quad \delta^{[\ell]} = \left(w^{[\ell+1]}\right)^T \delta^{[\ell+1]} \circ \sigma'(z^{[\ell]})$$

NCTS
National Center for Theoretical Sciences

(3) $z_j^{[\ell]} = \left(W^{[\ell]}\,\sigma(z^{(\ell-1)})\right)_j + b_j^{[\ell]}$

$$\frac{\partial C}{\partial b_j^{[\ell]}} = \frac{\partial C}{\partial z_j^{[\ell]}} \frac{\partial z_j^{[\ell]}}{\partial b_j^{[\ell]}} = \delta_j^{[\ell]} \cdot 1 = \delta_j^{[\ell]}$$

(4) $$z_j^{[\ell]} = \sum_{k=1}^{n_{\ell-1}} w_{jk}^{[\ell]}\, a_k^{[\ell-1]} + b_j^{[\ell]}$$

$$\frac{\partial z_j^{[\ell]}}{\partial w_{jk}^{[\ell]}} = a_k^{[\ell-1]} \quad \text{and} \quad \frac{\partial z_s^{[\ell]}}{\partial w_{jk}^{[\ell]}} = 0 \quad \text{for } s \neq j$$

$$\frac{\partial C}{\partial w_{jk}^{[\ell]}} = \sum_{s=1}^{n_\ell} \frac{\partial C}{\partial z_s^{[\ell]}} \frac{\partial z_s^{[\ell]}}{\partial w_{jk}^{[\ell]}} = \frac{\partial C}{\partial z_j^{[\ell]}} \frac{\partial z_j^{[\ell]}}{\partial w_{jk}^{[\ell]}} = \delta_j^{[\ell]}\, a_k^{[\ell-1]}$$

$2 \le \ell \le L$

Key idea: forward pass then backward pass

$a^{[L]}$ can be evaluated from a forward pass through the network by computing $a^{[1]}, z^{[2]}, a^{[2]}, z^{[3]}, \ldots, z^{[L]}, a^{[L]}$ in order

Having done this, $\delta^{[L]} \to \delta^{[L-1]}$ (from (2)) $\to \delta^{[L-2]} \to \cdots \to \delta^{[2]}$

(backward pass)

Then we have all access for the derivatives $\frac{\partial C}{\partial b_j^{[\ell]}}$, $\frac{\partial C}{\partial w_{jk}^{[\ell]}}$

NCTS
National Center for Theoretical Sciences

So at the 4th neuron of layer 3,

$$\frac{\partial C}{\partial w^{[3]}_{4,3}} = \delta^{[3]}_4 \, a^{[2]}_3$$

$$\frac{\partial C}{\partial b^{[3]}_4} = \delta^{[3]}_4$$

Replace the Hadamard product by introducing diag. matrices.

Let $D^{[\ell]} \in \mathbb{R}^{n_\ell \times n_\ell} = \begin{bmatrix} \sigma'(z^{[\ell]}_1) & & 0 \\ & \ddots & \\ 0 & & \sigma'(z^{[\ell]}_{n_\ell}) \end{bmatrix}$

Then $\delta^{[L]} = D^{[L]}(a^{[L]} - y)$

$$\delta^{[\ell]} = D^{[\ell]} \left(W^{[\ell+1]}\right)^T \delta^{[\ell+1]}$$

$$\delta^{[L-1]} = D^{[L-1]} \left(W^{[L]}\right)^T \delta^{[L]}$$

$$\delta^{[L-2]} = D^{[L-2]} \left(W^{[L-1]}\right)^T \delta^{[L-1]}$$

$$\delta^{[\ell]} = D^{[\ell]} \left(W^{[\ell+1]}\right)^T D^{[\ell+1]} \left(W^{[\ell+2]}\right)^T \cdots D^{[L-1]} \left(W^{[L]}\right)^T D^{[L]} (a^{[L]} - y)$$

From ③, $\delta^{[\ell]} = \nabla_{b^{[\ell]}} C$

④ $\dfrac{\partial C}{\partial w^{[\ell]}_{jK}} = \delta^{[\ell]}_j \, a^{[\ell-1]}_K \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$

$\in \quad \downarrow$

$1 \le j \le n_\ell \quad 1 \le K \le n_{\ell-1}$

$\dfrac{\partial C}{\partial W^{[\ell]}} = \delta^{[\ell]} \left(a^{[\ell-1]}\right)^T$ a matrix

NCTS
National Center for Theoretical Sciences

SG method with Back Propagation (Training process)

Algorithm: Train a network using a fixed iteration number of SG method (basic version)

For counter = 1 upto Niter

Choose an integer k uniformly at random from $\{1, 2, \ldots, N\}$

$x^{(k)}$ is current training data point

$a^{[1]} = x^{(k)}$

For $\ell = 2$ upto L

$$z^{[\ell]} = W^{[\ell]} a^{[\ell-1]} + b^{[\ell]}$$

$$a^{[\ell]} = \sigma(z^{[\ell]})$$

$$D^{[\ell]} = \mathrm{diag}(\sigma'(z^{[\ell]}))$$

end

$$\delta^{[L]} = D^{[L]} (a^{[L]} - y(x^{(k)}))$$

For $\ell = L-1$ downto 2

$$\delta^{[\ell]} = D^{[\ell]} (W^{[\ell+1]})^T \delta^{[\ell+1]}$$

end

For $\ell = L$ downto 2

$$W^{[\ell]} \to W^{[\ell]} - \eta \delta^{[\ell]} (a^{[\ell-1]})^T$$

$$b^{[\ell]} \to b^{[\ell]} - \eta \delta^{[\ell]}$$

end

end

Note: Initial $W^{[\ell]}$ and $b^{[\ell]}$ are chosen randomly

NCTS
National Center for Theoretical Sciences

Mathematics and Machine Learning

Ming-Chih Lai*

1 Introduction

Artificial Intelligence (AI) has gained a more than ever attention in almost every living fields of our daily life. A key component of the success of AI applications falls into the rapidly expanding field of Machine Learning (ML). The machine learning is not a brand new idea but only becomes realistic as the available data and computing resources grow significantly in past decade. As no one will doubt the important role of mathematics can play in the rapid evolution of machine learning, the proposal is aimed to what and how our local mathematical society should react to this global trend. In the following, we shall list some research topics that ML can help to resolve or to tackle the problems that the traditional mathematics seems to be challenging. 1) Solving the high-dimensional PDEs arising from the quantum mechanics (eg. Schrodinger equations with many particles), finance (Black-Scholes equation with many considering assets) or others coming from different fields. 2) Predicting and solving data-driven nonlinear PDE models arising from the physical or biological problems. 3) Improving the mathematical models or algorithms in traditional computational fluid dynamics by deep learning (eg. Turbulence modeling or weather forecast model). In addition, we shall also list some research issues on the numerical algorithms of machine learning and their practical applications.

*Department of Applied Mathematics, National Chiao Tung University, 1001, Ta Hsueh Road, Hsinchu 300, TAIWAN. E-mail: mclai@math.nctu.edu.tw

2 Solving high-dimensional PDEs

As known, partial differential equations (PDEs) are used to model the natural phenomena arising from different fields in science and engineering. In most cases, this is a well-developed field for different kinds (elliptic, hyperbolic and parabolic) of PDEs both mathematically and numerically. However, when the considered dimension of the equation becomes high (say tens or even hundreds), the notoriously difficult problem known as "curse of dimensionality" occurs. For instance, these high-dimensional PDEs can arise from the quantum mechanics (eg. Schrodinger equations with many particles), or finance (Black-Scholes equation with many considering assets); just to name a few. Even with today's growing computing power and resources, those high-dimensional problems are still intractable if the traditional numerical approximations (finite difference, finite element or finite volume methods) are used. Therefore, an alternative must be sought to circumvent the difficulty of "curse of dimensionality". In [1, 2], Han, Jentzen and E have developed a deep learning algorithm to solve a general class of semi-linear parabolic PDEs by first reformulating the underlying PDEs as backward stochastic differential equations (BSDEs) and then designing a deep learning framework to solve the problem instead. We plan to study the relation between the semi-linear parabolic PDEs and backward stochastic differential equations first [3].

Designing a deep learning framework that fits to solve these BSDEs is still highly non-trivial. Our main goal is to find a better design of deep neural network architecture to solve such problems and explore more applications of different PDEs.

3 Uncertainty quantification and data-driven discovery of PDEs

One should note that, a stochastic differential equation (SDE) is different in nature from the random differential equation which the randomness of the latter one appears in parameters, initial or boundary conditions, or even the external forcing terms [4]. For the random differential equations (RDEs), uncertainly quantification (UQ) leads to another interesting research issue in recent days. A systematic quantification of uncertainties and errors appearing in models, numerical algorithms, and experiments and the analysis

how they propagate to affect the predicted outcomes can help calibrate the models in a more precise way. Again, this is also an active area of research and a new SIAM/ASA Journal on Uncertainty Quantification was borne out of it. This is also an interdisciplinary research topic across the mathematics and statistics.

Rather than quantifying the uncertainties in the random parameters, another interesting issue is to recover the parameters in PDEs from the observed data which is called the data-driven discovery of PDEs. This problem is to use the machine learning technique to identify the parameters of given form of PDEs that are best suitable for the observed data. Recently, Raissi and Karniadakis proposed a new paradigm of learning from small data called hidden physics models [9] which is claimed to be data-efficient learning machines being able to leverage the underlying physics laws expressed by time-dependent nonlinear PDEs using the framework of Gaussian processes. Raissi, Perdikaris, and Karniadakis took a statistical inference viewpoint by using numerical Gaussian processes to construct the solutions for time dependent nonlinear PDEs based on black box initial conditions with noisy data [10]. Recently, the authors applied a physics-informed neural network that is trained to solve supervised learning tasks to recover nonlinear PDEs using continuous and discrete time models. Another technique using a sparse regression method capable of discovering the governing PDEs of a given system by time series measurements in the spatial domain was proposed by Rudy, Alla, Brunton and Kutz [6]. The main advantage of the above method is that the measurements can be made in either Eulerian manner, where the sensors are fixed in space or Lagrangian manner, where the sensors are moving with the dynamics (in fluid mechanics, moving with the fluid velocity). They also proposed a data-driven method for the discovery of parametric PDEs based on group sparsity and introduced a new algorithm called sequential grouped threshold ridge regression in [7]. Very recently, a unified sparse optimization framework to learn parsimonious physics-informed models from data has been proposed by the same group of authors in [8].

From our point of view, these works are quite interesting and worth further investigating and indeed open a door of solving or recovering the underlying PDEs using machine learning techniques without knowing any physical laws. Our main goal is first to investigate these underlying techniques and make comparisons, then we seek more practical improvements or theoretical justifications such as numerical algorithms with convergence guarantee.

4 Deep Learning in Computational Fluid Dynamics

The purpose of applying machine learning for fluid mechanics is of major interests among applied mathematics community today. Practical fluid mechanics is rapidly advancing due to unprecedented larger amounts of data from experiments, field measurements, and simulation results from computational fluid dynamics (CFD) approaches. Deep learning consists of various algorithms to retrieve information from the data that can be translated into knowledge of the underlying physics of fluids. Deep learning also provides a powerful framework that can enhance and transform current CFD research and industrial applications. In the following subsections, we shall describe our research topics on deep learning for CFD that we will pursue: (I) Multiphase turbulence modeling using immersed boundary method and deep neural network; (II) flow control using immersed boundary method and reinforcement learning.

4.1 Multiphase turbulence modeling using immersed boundary method and deep neural network

Measured data from experiments and direct numerical simulations (DNS) of turbulence modeling have been used to calibrate models such as the Reynolds-averaged Navier–Stokes (RANS) equations and Large-Eddy Simulation (LES) model. However, these models may require careful tuning to match fully resolved simulations or experiments. It is well-known that these models often failed to capture the correct physics in a specific region, including separation and reattachment, transitional flow, and wall-bound flows.

Recently, researchers have begun to explore machine learning methods to systematically inform the turbulence models with data, to quantify and reduce its uncertainties. Machine learning has been utilized to identify the discrepancies between model and high-fidelity simulations [12, 13, 14, 15, 17, 16]. The first deep neural network architecture is employed by [14]. The architecture consists of multiple hidden layers to model the anisotropic stress tensor. A multiplicative layer is used to embed Galilean invariance into the tensor calculations which provides a framework to consider known physical symmetries and invariances into the learning. This approach is promising in the future development for combining deep learning with physics.

There are limited studies relevant to modeling turbulent interfacial flows using RANS or LES. The main difficulty is that some new terms appear in the models without general closure, including the subgrid-scale surface tension term and the pressure fluctuations term. By exploiting knowledge in turbulence modeling and physical constraints, we can obtain effective models based on data-driven approaches. The objective of this work is to develop an IB method to simulate turbulence multiphase flow by augmenting data-driven models. This research directions are as follows.

1. Develop efficient and high accuracy IB methods for turbulent interfacial flow;

2. Develop deep neural networks with embedded physical constraint;

3. Use IB methods, the accurate numerical results obtained from DNS and the numerical results from RANS/LES to train the deep neural networks;

4. Validate the trained models against the baseline RANS/LES models and DNS numerical results.

4.2 Flow control using immersed boundary method and reinforcement learning

Reinforcement learning (RL) is a goal-oriented algorithm that learns to attain a complex objective based on Markov Decision Process. Beyond the Google DeepMind Challenge Match (a five-game Go match between world champion Lee Sedol and AlphaGo), RL has received increasing interest for problem-solving in the different domains including flow optimization and control. For instance, drag control of bluff body fluid dynamics [18], collective hydrodynamics in self-propelled swimmers that adapted their motion to gain substantial energetic benefits [19, 21, 20], identify glider's strategies of navigation that can cope with flow fluctuations [22], optimization of the motion of microswimmers [23, 24]. The computational efficiency of the IB method holds the potential to couple effectively with reinforcement learning algorithms, since remeshing is not required and fast solvers can be applied readily.

The objective of this work is to develop an IB method to simulate the fluid-structure interaction problem for flow control via reinforcement learning. This goal is planned to accomplish by:

1. develop efficient IB methods based on block LU decomposition to simulate fluid-structure interaction problem for rigid/flexible bodies;

2. coupling the IB method with reinforcement learning algorithm for solving the optimization problem of flow control;

3. applications to be studied include drag reduction of rigid bluff bodies, smart microswimmer or moving particles, and active flow control of flexible bodies such as aerofoils.

5 Participating members

The aims of this proposal are twofold. Firstly, we target some interesting research topics that mathematics could play roles in the growing machine learning field and its applications. Secondly, we want to organize a MATH-ML team and cultivate some young researchers to gain necessary knowledge to do this kind of interdisciplinary research. So the following are the participating faculty members, postdocs and Ph.D students. We hope that there are more master students joining us.

Faculty members

- Ming-Chih Lai (PI, Department of Applied Math, NCTU, focused on Solving high-dimensional PDEs, Uncertainty quantification and data-driven discovery of PDEs, and Deep Learning in Computational Fluid Dynamics)

- Tzer-jen Wei (AI college, NCTU, focused on industrial applications)

- Te-Sheng Lin (Department of Applied Math, NCTU, focused on Solving high-dimensional PDEs, Uncertainty quantification and data-driven discovery of PDEs)

- Chia-Chieh Jay Chu (Department of Math, NTHU, focused on Solving high-dimensional PDEs, Uncertainty quantification and data-driven discovery of PDEs)

- Tien-Tsan Shieh (Department of Applied Math, CYCU, focused on Solving high-dimensional PDEs, Uncertainty quantification and data-driven discovery of PDEs)

Postdocs

- Kian Chuan Ong (Fields Institute, Canada, focused on Deep Learning in Computational Fluid Dynamics)
- Shih-Hsuan Hsu (NCTU, focused on Solving high-dimensional PDEs, Uncertainty quantification and data-driven discovery of PDEs)
- Yang-Kai Lue (NTNU, focused on data-driven discovery of PDEs)

Ph.D students

- Jun-Yen Yang (Department of Applied Math, NCTU)
- Rajendra Kumar (ICST, NCTU, starting on Feb. 2020)

6 Working items

1. Organize a regular working seminar on Math and Machine Learning at NCTU to study the papers and learn the necessary math and statistics background
2. Update the research progress in the form of workshop in NCTU every half year
3. Make contacts with the industrial companies or hospitals to conduct realistic collaborations

7 Anticipated results

1. Build up a research team for Math and Machine Learning
2. Build up a practical working team on AI and Data Analysis for conducting industrial projects

References

[1] W. E., J. Han, A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, *arXiv:1706.04702v1.*

[2] J. Han, A. Jentzen, W. E., Solving high-dimensional partial differential equations using deep learning, *PNAS*, Vol 115, No 34, (2018) 8505-8510.

[3] E. Pardoux, S. Tang, Forward-backward stochastic differential equations and quasilinear parabolic PDEs, *Probab. Theor. Relat. Fields*, 114 (1999) 123-150.

[4] R. C. Smith, Uncertainty Quantification, *Theory, Implementation, and Applications*, SIAM 2014.

[5] M. Sugiyama, Introduction to Statistical Machine Learning, Morgan Kaufmann, 2016.

[6] S. H. Rudy, S. L. Brunton, J. L. Proctor, J. N. Kutz, Data-driven discovery of partial differential equations, *Sci. Adv.* 2017;3:e1602614.

[7] S. H. Rudy, A. Alla, S. L. Brunton, J. N. Kutz, Data-driven identification of parametric partial differential equations, *SIAM J. Applied Dynamical Systems*, Vol 18, No 2, (2019) 643-660.

[8] K. Champion, P. Zheng, A. Y. Aravkin, S. L. Brunton, J. N. Kutz, A unified sparse optimization framework to learn parimonious physics-informed models from data, *arXiv:1906.10612v1.*

[9] M. Raissi, G. E. Karniadakis, Hidden physics models: machine learning of nonlinear partial differential equations, *arXiv:1708.00588v2.*

[10] M. Raissi, P. Perdikaris, G. E. Karniadakis, Numerical gaussian processes for time-dependent and nonlinear partial differential equations, *SIAM J. Sci. Comput.*, Vol 40, No 1, (2018) A172-A198.

[11] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.* 378 (2019), 686-707.

[12] J. Ling, J. Templeton, *Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty*, Phys. Fluids, 27 (2015), 085103.

[13] J. Ling, R. Jones, J. Templeton, *Machine learning strategies for systems with invariance properties*, J. Comp. Phys., 318 (2016), 22-35.

[14] J. Ling, A. Kurzawski, J. Templeton, *Reynolds averaged turbulence modelling using deep neural networks with embedded invariances*, J. Fluid Mech., 807 (2016), 155-166

[15] H. Xiao, J. Wu, J. Wang, R. Sun, C. Roy, *Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier-Stokes simulations: A data-driven, physics-informed Bayesian approach*, J. Comp. Phys., 324 (2016), 115-136.

[16] J. Wang JX, J. Wu, H. Xiao, *Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data*, Phys. Rev. Fluids, 2 (2017) 034603.

[17] A. Singh, S. Medida, K. Duraisamy. *Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils.*, AIAA, 55 (2017), 2215-2227.

[18] F. Gueniat, L. Mathelin, M. Y. Hussaini, *A statistical learning strategy for closed-loop control of fluid flows*, Theoretical and Computational Fluid Dynamics, 30 (2016), 497-510.

[19] M. Gazzola, B. Hejazialhosseini, P. Koumoutsakos, *Reinforcement learning and wavelet adapted vortex Methods for simulations of self-propelled swimmers*, SIAM J. Sci. Comput., 36 (2014), B622-B639.

[20] S. Verma, G. Novati, P. Koumoutsakos, *Efficient collective swimming by harnessing vortices through deep reinforcement learning*, PNAS, 115 (2018), 5849-5854.

[21] M. Gazzola, A. A. Tchieu, D. Alexeev, A. de Brauer, P. Koumoutsakos, *Learning to school in the presence of hydrodynamic interactions*, Journal of Fluid Mechanics, 789 (2018), 726-749.

[22] G. Reddy, A. Celani, T. J. Sejnowski, M. Vergassola, *Learning to soar in turbulent environments*, PNAS, 113 (2016) E4877-E4884

[23] S. Colabrese, K. Gustavsson, A. Celani, L. Biferale, *Flow Navigation by Smart Microswimmers via Reinforcement Learning*, Phys. Rev. Lett. 118 (2017) 158004.

[24] S. Colabrese, K. Gustavsson, A. Celani, L. Biferale, *Smart inertial particles*, Phys. Rev. Fluids 3 (2018), 084301

[25] Elad Hazan, Sparse approximate solutions to semidefinite programs. In Eduardo Sany Laber, Claudson Bornstein, Loana Tito Nogueira, and Luerbio Faria, editors, *LATIN 2008: Theoretical Informatics*, pages 306–316, 2008.

[26] Jeongwan Haah, Aram W. Harrow, Zhengfeng Ji, Xiaodi Wu, and Nengkun Yu, Sample-optimal tomography of quantum states, *IEEE Trans. Inf. Theory*, 63(9):5628–5641, 2017.

[27] H. Häffner, W. Hänsel, C. F. Roos, J. Benhelm, D. Check-al-kar, M. Chwalla, T. Körber, U. D. Rapol, M. Riebe, P. O. Schmidt, C. Becher, O. Gühne, W. Dür, and R. Blatt, Scalable multiparticle entanglement of trapped ions, *Nature*, 438:643–646, 2005.

[28] Yen-Huan Li and Volkan Cevher, Convergence of the exponentiated gradient method with Armijo line search, *J. Optim. Theory Appl.*, 181(2):588–607, 2019.

[29] Stéphane Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, Academic Press, Burlington, MA, third edition, 2009.

[30] G. Molina-Terriza, A. Vaziri, J Řeháček, Z. Hradil, and A. Zeilinger, Triggered qutrits for quantum communication protocols, *Phys. Rev. Lett.*, 92(16), 2004.

[31] Philippe Rigollet and Alexandre B. Tsybakov, Sparse estimation by exponential weighting, *Stat. Sci.*, 27(4):558–575, 2012.

[32] Shai Shalev-Shwartz, Ohad Shamir, and Eran Tromer, Using more data to speed-up training time, *Proc. 15th Int. Conf. Artificial Intelligence and Statistics*, 2012.

[33] Jaroslav Řeháček, Zdeněk Hradil, E. Knill, and A. I. Lvovsky, Diluted maximum-likelihood algorithm for quantum tomography, *Phys. Rev. A*, 75, 2007.

[34] Y. Vardi, L. A. Shepp, and L. Kaufman, A statistical model for positron emission tomography, *J. Am. Stat. Assoc.*, 80(389):8–20, 1985.

This slide is provided by Prof Richard Tsai of UT-Austin supported by the present MoE grant.

Opportunities around Deep Learning

Current thoughts from my own limited experience

Richard Tsai

The University of Texas at Austin

What can deep learning (DL) do for me?

- Approximate something which is hard to tabulate or compute by conventional methods

- Use DL in more tolerant setups: e.g. some contractive iterations

    - Help getting into radius of convergence: (quasi)-Newton methods

    - Improve convergence properties: pre-conditioners, error smoothers, post-processing

    - Need to figure out clever ways to tabulate and interpolate already-made computations (supervised learning)

    - Certain low-rank approximation of matrices: e.g force evaluation in boundary integral computations

    - A posteriori error control?

What is necessary for DL to work for me?

- Generate relevant data:
  - Need casually consistent and representative data

  - If DL is supposed to help compute some dynamical proesses, then generate data from simulations of such dynamical processes

What can I do for deep learning

- Obviously, one can work on approximation theory for deep learning.

- But for which architecture? New ones are coming out all the time.

- Improve existing deep learning strategies:

  - Which loss function? Which regularization?

  - Which geometry (gradient flows)

  - Stability of deep neural network

Opportunities for numerical analysts and mathematicians

- Mitigate adversarial attacks: e.g. robust optimization
- Principled designed of neural network models: e.g. random graphs and graph coarsening, stable deep neural networks
- CNN on manifolds
- Improve training efficiency
    - Choosing a loss function that makes the problem more convex or easier to perform gradient descent: e.g. cross-entropy function instead of L2
    - Decoupling of deep neural networks
    - Speed up gradient descents
    - Algorithms for efficient generation or sampling of data
      Efficiency = minimal data for maximal generalization
    - Variance reduction
    - Theory for “batch normalization” and “dropouts”
- Consistency?

Emergent mathematical areas

- Optimal transport, theory and computational methods

- Mean field games

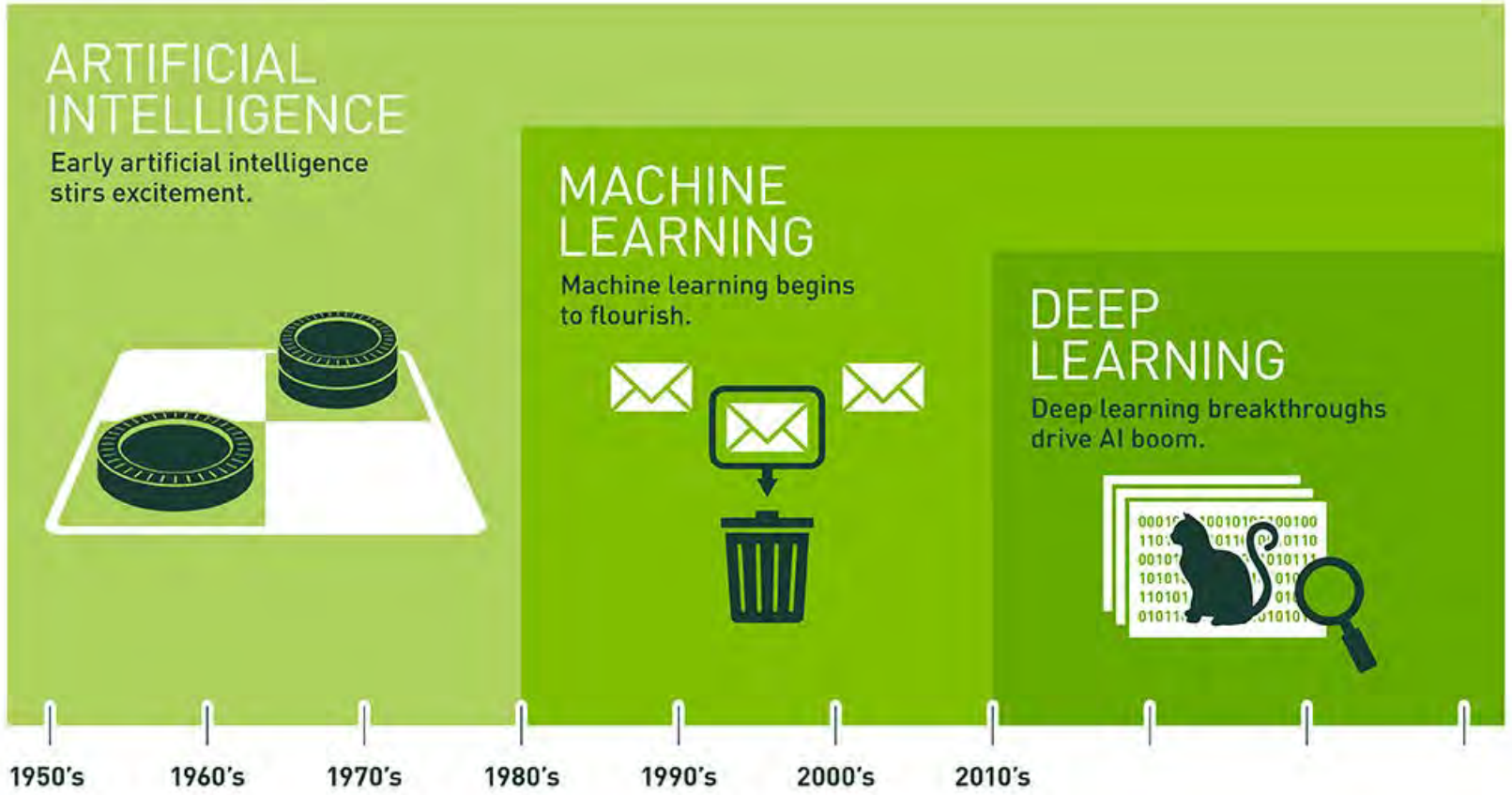
數學與機器學習實驗室

賴明治

應用數學系 / 講座教授

智慧科學暨綠能學院 /合聘教師

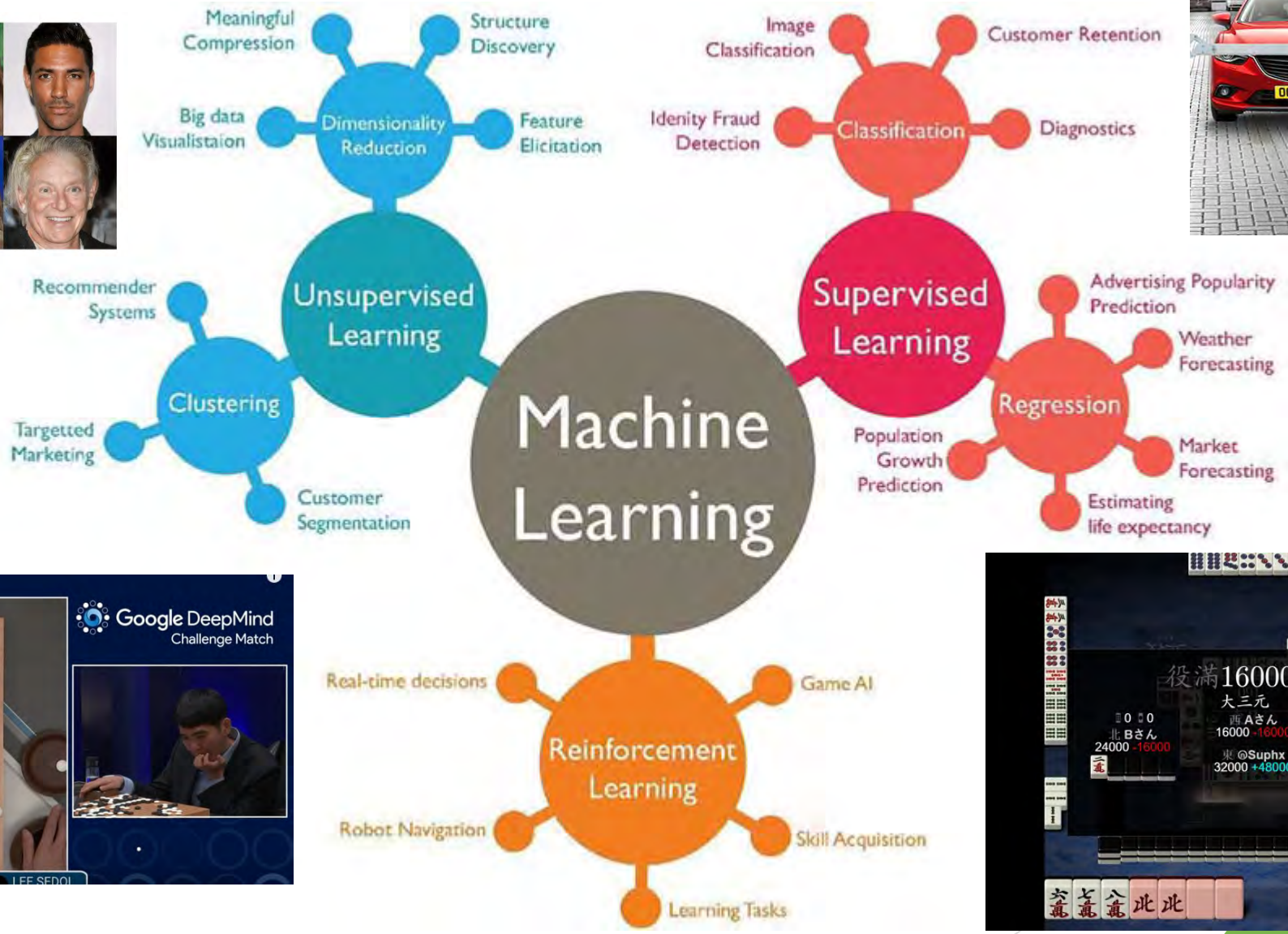
History and Background



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

DEEP LEARNING EXPLAINED, Chapter 1, Artificial intelligence , NVIDIA
https://blogs.nvidia.com.tw/2016/07/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

機器學習的問題種類與應用

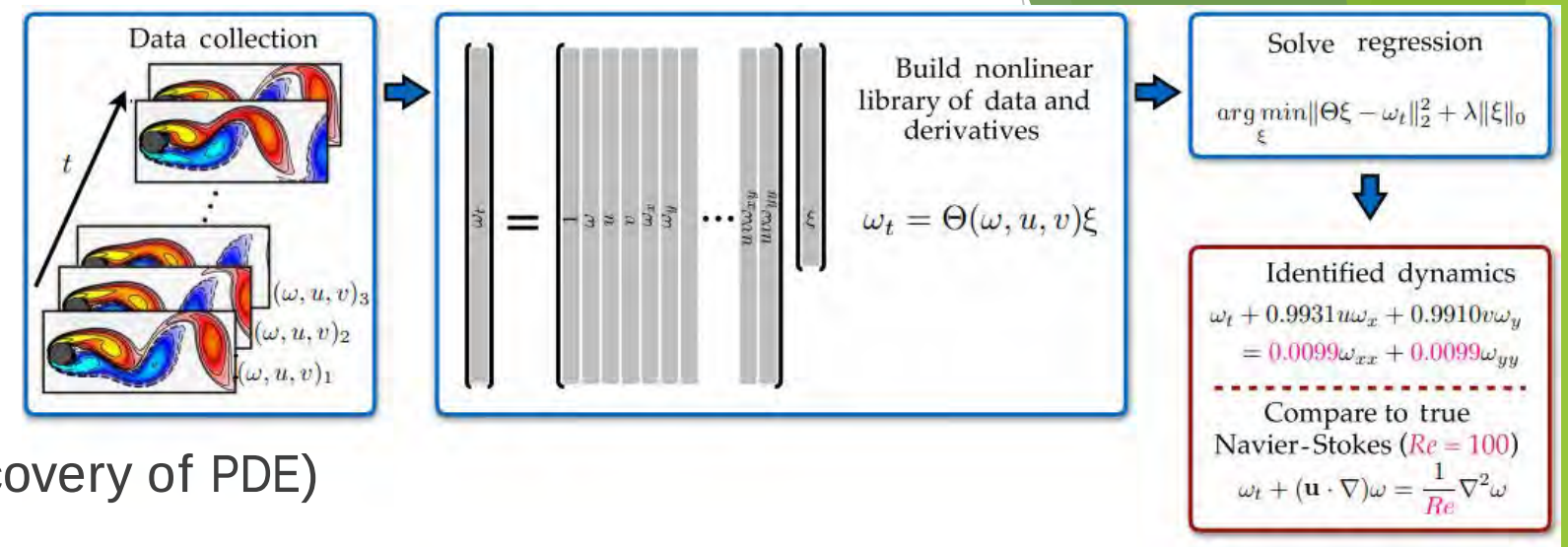


2017.01.04
60勝0敗！謎樣棋士橫空出世 台日韓中圍棋第一高手悉數稱臣

2019.08.29
微軟AI研究新突破：超級麻將AI 實力媲美頂級人類選手

研究方向

- 傳統流體力學的計算
- 資料驅動的偏微分方程解
  (data-driven solution and discovery of PDE)
- 高微度的偏微分方程數值解
- 機器學習方法收斂性分析
- 不確定性量化



High dimension of nonlinear parabolic partial differential equations (PDEs)

$$\begin{cases} \partial_t u + \mathcal{L}u + f(.,.,u,\sigma^\intercal D_x u) = 0, & \text{on } [0,T)\times\mathbb{R}^d \\ u(T,.) = g, & \text{on } \mathbb{R}^d \end{cases}$$

應用方面

- 影像生成
- 醫學影像辨識
- 數據分析

預備知識

- 微積分、線性代數、機率學、統計學、向量微積分、計算機概論
- 程式語言（Python、Matlab）

研究所修課方向

- 機器學習、深度學習
- 科學計算導論、數值偏微分方程、數值最佳化
- 影像處理、訊號處理
- 測度論、泛函分析、微分幾何

團隊成員

賴明治

應用數學系 / 講座教授

智慧科學暨綠能學院 / 合聘教師

研究領域專長

科學計算、流體力學

魏澤人

智慧科學暨綠能學院 / 副教授

機器學習 GDE (Google Developers Expert)

研究領域專長

集合論、密碼學、計算理論、深度學習、生成對抗網路

博士後研究人員： 徐士軒 博士、呂楊凱 博士

博士研究生： 楊俊彥 **碩士研究生及大學部**: 歡迎加入