Tools Seminar

Week 8 - Deep Learning

Hongzheng Chen

Mar 16, 2020

1. Introduction

2. Deep Learning

3. Frameworks
   - Installation
   - Tutorials

4. Summary

1

Introduction

AI Milestones

- 1950, Alan Turing proposed the famous Turing test

AI Milestones

- 1950, Alan Turing proposed the famous Turing test
- 1955, John McCarthy created the term “artificial intelligence” (1971 Turing award)

AI Milestones

- 1950, Alan Turing proposed the famous Turing test
- 1955, John McCarthy created the term “artificial intelligence” (1971 Turing award)
- 1997, IBM’s Deep Blue beat world chess champion Garry Kasparov

AI Milestones

- 1950, Alan Turing proposed the famous Turing test
- 1955, John McCarthy created the term “artificial intelligence” (1971 Turing award)
- 1997, IBM’s Deep Blue beat world chess champion Garry Kasparov
- 2011, IBM Watson defeated two champions at quiz show Jeopardy

AI Milestones

- 1950, Alan Turing proposed the famous Turing test
- 1955, John McCarthy created the term “artificial intelligence” (1971 Turing award)
- 1997, IBM’s Deep Blue beat world chess champion Garry Kasparov
- 2011, IBM Watson defeated two champions at quiz show Jeopardy
- 2012, Jeff Dean and Andrew Ng used unsupervised learning to train neural network which learned to recognize cats

AI Milestones (Cont.)

- 2012, AlexNet achieved an error rate of only 16% in ImageNet Large Scale Visual Recognition Challenge

AI Milestones (Cont.)

- 2012, AlexNet achieved an error rate of only 16% in ImageNet Large Scale Visual Recognition Challenge
    - Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton (University of Toronto)

AI Milestones (Cont.)

- 2012, AlexNet achieved an error rate of only 16% in ImageNet Large Scale Visual Recognition Challenge
  - Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton (University of Toronto)
  - 2 GPU, a week, halved the error rate

AI Milestones (Cont.)

- 2012, AlexNet achieved an error rate of only 16% in ImageNet Large Scale Visual Recognition Challenge
  - Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton (University of Toronto)
  - 2 GPU, a week, halved the error rate
  - **Open the era of deep learning**

AI Milestones (Cont.)

- 2012, AlexNet achieved an error rate of only 16% in ImageNet Large Scale Visual Recognition Challenge
    - Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton (University of Toronto)
    - 2 GPU, a week, halved the error rate
    - **Open the era of deep learning**
- 2015, Kaiming He (MSRA) proposed ResNet which made machine see better than human (3.6% error rate in ImageNet)

AI Milestones (Cont.)

- 2012, AlexNet achieved an error rate of only 16% in ImageNet Large Scale Visual Recognition Challenge
    - Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton (University of Toronto)
    - 2 GPU, a week, halved the error rate
    - **Open the era of deep learning**
- 2015, Kaiming He (MSRA) proposed ResNet which made machine see better than human (3.6% error rate in ImageNet)
- 2016, Google DeepMind's AlphaGo defeated 9-dan Go master Lee sedol by 4:1

AI Milestones (Cont.)

- 2012, AlexNet achieved an error rate of only 16% in ImageNet Large Scale Visual Recognition Challenge
  - Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton (University of Toronto)
  - 2 GPU, a week, halved the error rate
  - **Open the era of deep learning**
- 2015, Kaiming He (MSRA) proposed ResNet which made machine see better than human (3.6% error rate in ImageNet)
- 2016, Google DeepMind's AlphaGo defeated 9-dan Go master Lee sedol by 4:1
  - Firstly proposed deep reinforcement learning

AI Milestones (Cont.)

- 2012, AlexNet achieved an error rate of only 16% in ImageNet Large Scale Visual Recognition Challenge
  - Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton (University of Toronto)
  - 2 GPU, a week, halved the error rate
  - **Open the era of deep learning**
- 2015, Kaiming He (MSRA) proposed ResNet which made machine see better than human (3.6% error rate in ImageNet)
- 2016, Google DeepMind's AlphaGo defeated 9-dan Go master Lee sedol by 4:1
  - Firstly proposed deep reinforcement learning
- 2018, Google BERT model achieved the state-of-the-art performance in 11 NLP tasks

ImageNet & Deep Neural Network

ImageNet [Feifei Li, Stanford] Large Scale Visual Recognition Challenge

	LeNet	AlexNet	VGG	GoogleLeNet	ResNet
# of Layers	5	8	19	22	**152**
Top 5 Error	N/A	16.4%	7.3%	6.7%	3.57%
Year	1994	2012	2014	2014	2015

This is why it's called "**deep**" learning

2018 Turing Award

2018 Turing Award: Geoffrey Hinton, Yoshua Bengio, Yann LeCun

> *"for conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing"*

- Geoffrey Hinton: Backpropagation, Boltzmann Machines, Improvements to Convolutional Neural Network (CNN)
- Yoshua Bengio: Probabilistic models of sequences, High-dimensional word embeddings and attention, Generative adversarial networks (GAN)
- Yann LeCun: CNN, backprop, Broadening the vision of neural networks

Impetus of Deep Learning

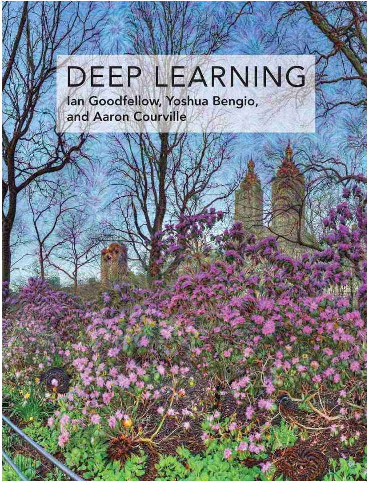
Looking back, we may know what leads to the boom of DL in 2010s

- Large amount of labeled **data**: ImageNet (2010)
- Improvement of **algorithms**: deep networks, dropout (2012)
- Invention of deep learning **systems**: TensorFlow (2015), PyTorch (2016)
- Improvement of **hardware**: GPU → GPGPU (general-purpose GPU)

All of them are indispensable and make up the whole DL stack

Introductory Books and Courses

- Feifei Li, Stanford cs231n: Convolutional Neural Networks for Visual Recognition (highly recommended!)
- Chris Manning, Stanford cs224n: Natural Language Processing with Deep Learning
- Ian Goodfellow, Deep Learning Chinese version



2

Deep Learning

Linear Regression

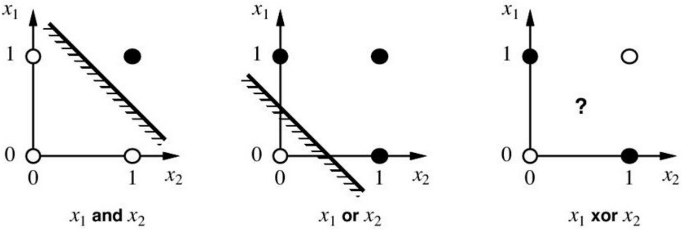
Recall the linear regression problem

$$y = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b = \begin{bmatrix}\mathbf{x}^{\mathrm{T}} & b\end{bmatrix}\begin{bmatrix}\mathbf{x} \\ 1\end{bmatrix} = \boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}$$

To minimize loss function (MSE)

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \mathbf{w}^{\mathrm{T}}\mathbf{x}_i - b)^2$$

But linear function can only deal with linearly separable problems



Powerful Models

Can we build a model that is powerful enough to represent all the functions?

- $f(\text{image}) = \text{location}$
- $f(\text{question}) = \text{answer}$
- $f(\text{speech}) = \text{text}$
- $\cdots$

Powerful Models

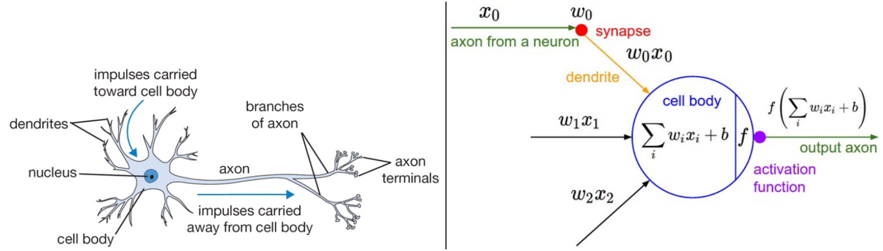
Can we build a model that is powerful enough to represent all the functions?

- $f(\text{image}) = \text{location}$
- $f(\text{question}) = \text{answer}$
- $f(\text{speech}) = \text{text}$
- $\cdots$

Actually, in ML area, we have built decision tree, SVM, etc., but they usually need feature engineering and are not flexible

Neuron Model

We can refer to our brain and see how we learn

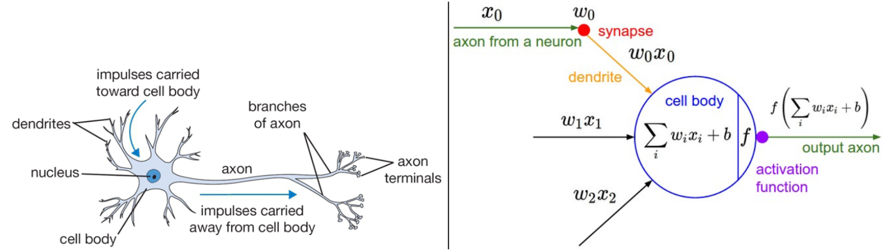


A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Fig source: http://cs231n.github.io/neural-networks-1/

Neuron Model

We can refer to our brain and see how we learn



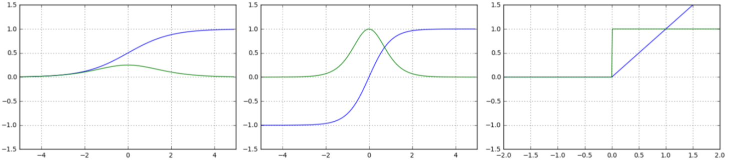
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Fig source: http://cs231n.github.io/neural-networks-1/

A neuron is just a linear model with activation function!

Activation Function

Used to add non-linear part to the model, enabling it to approximate much more complex functions



Sigmoid tanh ReLU

- Sigmoid: $g(z) = 1/(1 + \mathrm{e}^{-z})$ (S curve)
- Tanh: $g(z) = \tanh(z)$
- ReLU (Rectified Linear Unit): $g(z) = \max(0, z)$, can avoid gradient vanishing

From one to more

Only one neuron can do limited things, what about more?

- More neurons in width:

  > *A feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of $\mathbb{R}^n$.*
  >
  > — *The Universal Approximation Theorem*

  The question is that the theorem does not tell us how many neurons we need

From one to more

Only one neuron can do limited things, what about more?

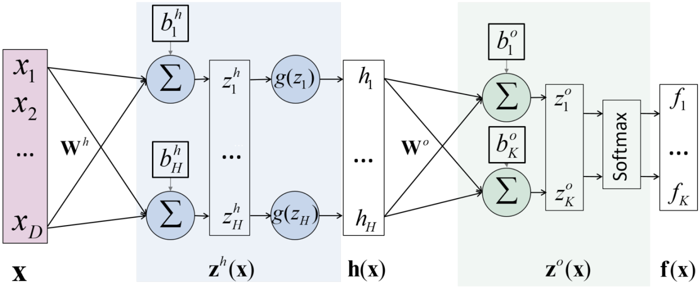
- More neurons in width:

  > *A feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of $\mathbb{R}^n$.*
  >
  > — *The Universal Approximation Theorem*

  The question is that the theorem does not tell us how many neurons we need

- More neurons in depth: We have multi-layer perceptron (MLP) — the basic model of nowadays deep learning!

Multi-Layer Perceptron (MLP) / Fully-connected NN



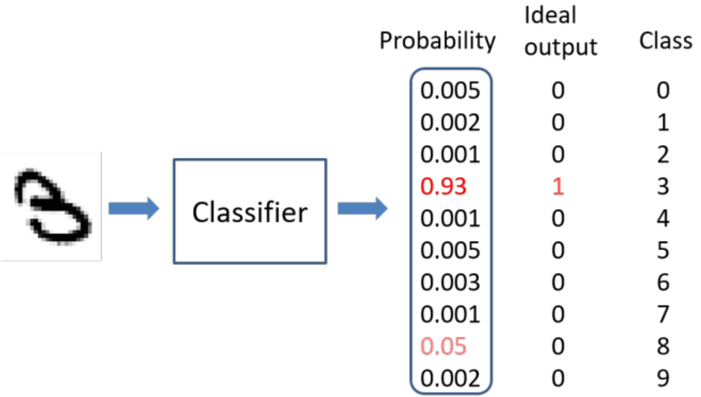
- Input layer: $\mathbf{x}$
- Hidden layer: $h(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(W^h\mathbf{x} + \mathbf{b}^h)$
- Output layer: $f(\mathbf{x}) = \sigma(\mathbf{z}^o(\mathbf{x})) = \sigma(W^o h(\mathbf{x}) + b^o)$

* Softmax function: change output to probability (K-dimensional)

$$\sigma(\mathbf{z})_j = \frac{\mathrm{e}^{z_j}}{\sum_{k=1}^{K} \mathrm{e}^{z_k}}$$

Classifier Training

- Find optimal parameters $\boldsymbol{\theta}^*$ of a classifier $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$
- Rule: given input $\mathbf{x}$, classifier output $f(\mathbf{x}; \boldsymbol{\theta})$ should be as close to the ideal output as possible



Classifier

Probability	Ideal output	Class
0.005	0	0
0.002	0	1
0.001	0	2
0.93	1	3
0.001	0	4
0.005	0	5
0.003	0	6
0.001	0	7
0.05	0	8
0.002	0	9

Classifier Training

Use MSE or other loss function

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \| f(\mathbf{x}_i; \boldsymbol{\theta}) - y_i \|_2^2$$

Use gradient descent to optimize parameters

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \alpha \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

About how to optimize the above function on NN (backpropagation), please read http://cs231n.github.io/optimization-1/

Different Kinds of NN

- CNN (convolutional NN): CV
  - Pooling / subsampling
  - Dropout
  - Residual block
- RNN (recurrent NN): NLP
  - LSTM
  - GRU
- GAN (generative adversarial network): Image generation

3

Frameworks

Deep Learning Frameworks

Framework: A large package consisting of lots of deep learning primatives/operators, and users can easily call them by API

- Google: Tensorflow (commonly used in industry)
    - Static computation graph
    - Jeff Dean
- Facebook: PyTorch (commonly used in academics)
    - Dynamic computation graph
    - Yangqing Jia, Caffe
- Amazon: MXNet
    - Tianqi Chen (UW → CMU)

* We focus on **PyTorch** in this seminar

PyTorch

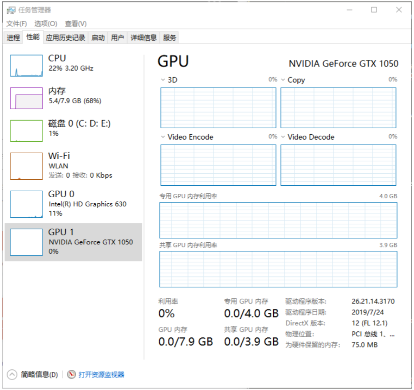
PyTorch: A Python-based scientific computing package

- A replacement for NumPy to use the power of GPUs
- A deep learning research platform that provides maximum flexibility and speed

Since it is highly embedded in Python, PyTorch is very Pythonic and easy-to-use

Pytorch Installation

Firstly check if your computer has discrete graphics card (GPU)



CPU
22% 3.20 GHz

内存
5.4/7.9 GB (68%)

磁盘 0 (C: D: E:)
1%

Wi-Fi
WLAN
发送: 0 接收: 0 Kbps

GPU 0
Intel(R) HD Graphics 630
11%

GPU 1
NVIDIA GeForce GTX 1050
0%

GPU NVIDIA GeForce GTX 1050

利用率
0%

专用 GPU 内存
0.0/4.0 GB

GPU 内存
0.0/7.9 GB

共享 GPU 内存
0.0/3.9 GB

驱动程序版本:	26.21.14.3170
驱动程序日期:	2019/7/24
DirectX 版本:	12 (FL 12.1)
物理位置:	PCI 总线 1、...
为硬件保留的内存:	75.0 MB

Install Nvidia driver: https://zhuanlan.zhihu.com/p/54350088

- CUDA 10.1
- cuDNN 7: Installation guide

Pytorch Installation

Select your configuration on this website and run the installation command

- Windows: Need to install Anaconda first
- WSL does not support GPU! Do NOT install Pytorch on WSL!
- Mac does not support GPU too (if you do not have external interface)!

e.g. For Windows with no GPUs

```
pip install torch==1.4.0+cpu torchvision==0.5.0+cpu -f https://
    download.pytorch.org/whl/torch_stable.html
```

Check if GPU works correctly by

```
import torch
print(torch.cuda.is_available())
```

Tutorials

PyTorch has very detailed documentations, make the best of them!

- Tutorials: `https://pytorch.org/tutorials/`
- Chinese tutorials: `https://pytorch.apachecn.org/`
- Documentation / API: `https://pytorch.org/docs/stable/index.html`
- Deep Learning with PyTorch: A 60 Minute Blitz
  - Chinese version
  - You can download the `.ipynb` file or directly run on Colab

4

Summary

Summary

- Introduction
- Deep Learning Framework: PyTorch
  - Once you get in some trouble concerning PyTorch, you can search the Docs of PyTorch for details. Alternatively you can try to find if there are similar problems on *PyTorch Discuss*.
- Get through cs231n!