Introduction to AI

Lecture 4:

Neural Networks Basics

by Hong-Han Shuai
National Yang Ming Chiao Tung University

Syllabus

Week	Date	Contents
1	9/2	Lecture 1: Class Overview and Unsupervised Learning (HW#1)
2	9/9	Lecture 2: Traditional Classification-Part 1
3	9/16	Lecture 3: Traditional Classification-Part 2
4	9/23	Lecture 4: Neural Networks Basics (HW#2)
5	9/30	Hands-on Tutorials on PyTorch
6	10/7	Lecture 5: Deep Learning in Practice
7	10/14	Lecture 6: Introduction to Natural Language Processing
8	10/21	Midterm

Introduction to AI Community

AI社群入門指南

為新進大學部與研究生量身打造的AI學術社群導航手冊。學會在哪裡尋找資源、追蹤哪些動態，以及如何開始貢獻你的力量。

AI領域全景圖

核心領域

學習、表徵、推理

熱門領域

視覺、語言、語音、機器人、強化學習、多模態

研究產出

會議、期刊、程式碼、基準測試、資料集

跨領域橋樑

數據、系統、安全、人機交互、倫理

頂級綜合AI會議

NeurIPS

理論與實踐並重，涵蓋面極廣。擁有豐富的工作坊生態系統和數據集資源，是AI領域最具影響力的會議之一。

ICML

專注機器學習核心，理論與演算法並重。以嚴謹的學術標準和創新的方法論著稱。

ICLR

表徵學習與深度學習專家聚集地。獨特的開放式同儕評議文化，促進透明的學術討論。

各領域頂級會議

電腦視覺

- CVPR - 電腦視覺與模式識別
- ICCV - 國際電腦視覺會議
- ECCV - 歐洲電腦視覺會議

自然語言處理

- ACL - 計算語言學協會
- EMNLP - 自然語言處理實證方法
- NAACL - 北美計算語言學分會會

規劃與廣義AI

- AAAI - 人工智慧促進協會
- IJCAI - 國際人工智慧聯合會議

選擇會議時考慮：目標受眾、評議文化、時程安排、工作坊與主會議的適配性

重要學術期刊

通用機器學習

JMLR - 機器學習研究期刊

TMLR - 機器學習研究彙刊

電腦視覺

TPAMI - 模式分析與機器智慧彙刊

IJCV - 國際電腦視覺期刊

自然語言處理

TACL - 計算語言學協會彙刊

Computational Linguistics - 計算語算語言學

期刊優勢：成熟研究成果、深度分析空間、典藏價值

論文搜尋核心工具

Google Scholar

追蹤作者頁面、引用關係，設定關鍵字提醒。最全面的學術搜尋引擎。

arXiv

每日更新的預印本論文庫。訂閱cs.AI、cs.CV、cs.CL、stat.ML等分類。

OpenReview

ICLR和眾多工作坊的開放評議平台。可查看修訂版本和同儕討論過程。

Semantic Scholar

主題頁面整理、引用圖譜分析、「高影響力」篩選功能。AI驅動的學術搜尋。

進階發現工具

01

Papers with Code

程式碼連結、排行榜、資料集頁面的一站式平台

02

DBLP

清晰的書目搜尋和會議篩選功能

03

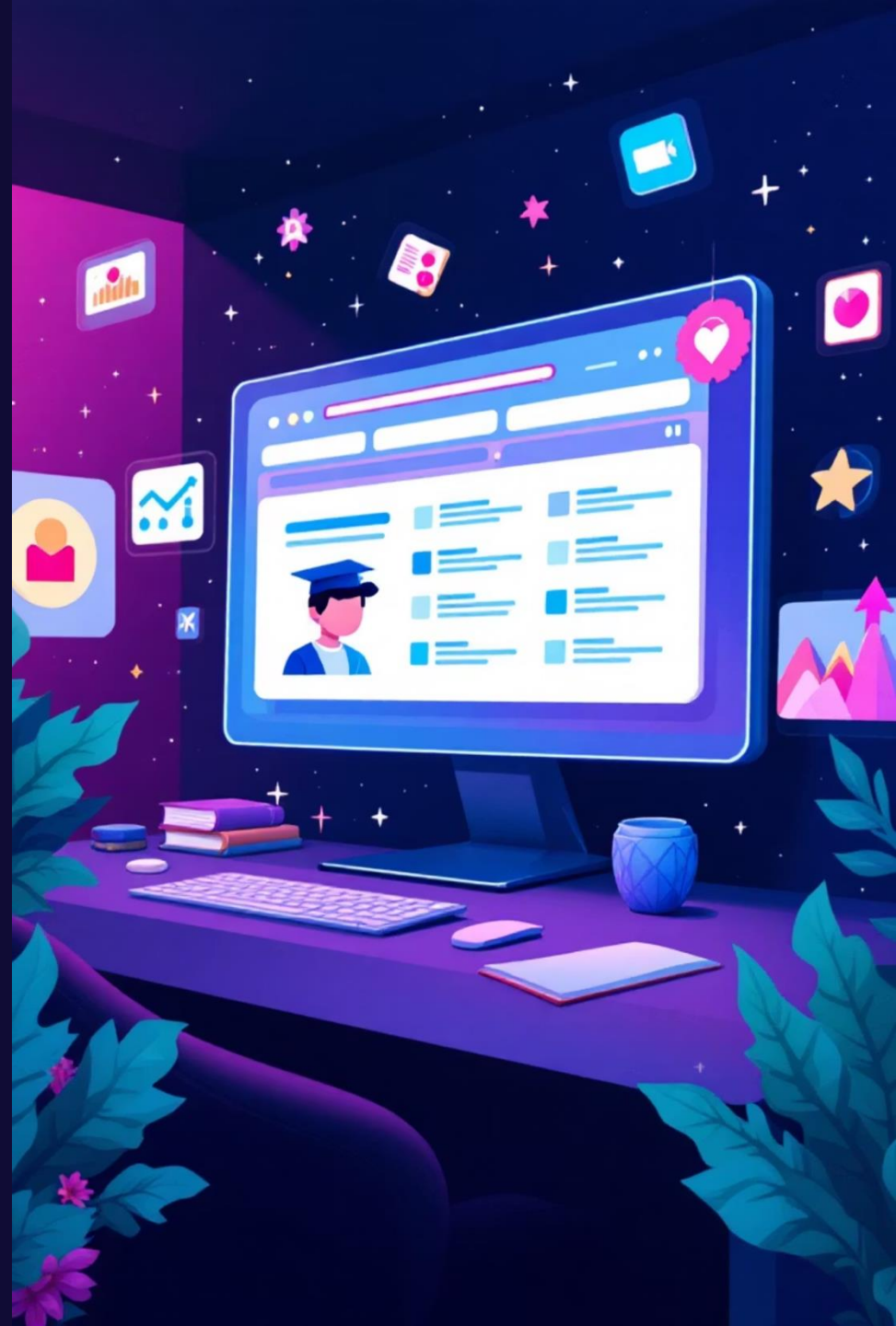
Connected Papers

論文關聯圖譜探索，發現相關研究

04

大學研討會

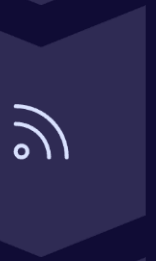
實驗室部落格和研討會日程，搶先預覽最新研究



保持資訊更新

設定Scholar提醒

針對關鍵字和作者設定自動通知



訂閱arXiv摘要

每日接收目標分類的最新論文

追蹤社群媒體

關注會議帳號、領域主席和實驗室負責人

加入社群

參與Discord或Slack的子領域和開源專案群組

高效閱讀策略

1. **第一遍：5分鐘快讀**
   掌握問題、方法概念、結果和要點
2. **第二遍：深入理解**
   研讀圖表、方程式、假設、數據和計算
3. **第三遍：實踐驗證**
   重現小部分實驗或與已知基準比較

建立閱讀紀錄：每篇論文5個要點、1張圖像、1個批評、1個問題。在讀書會中分享摘要以建立集體記憶。

調查資源與模式

以下是一些有效收集與分析AI領域資源的策略與方法：

資源彙整

尋找「Awesome + 主題」清單，快速掌握領域概況和關鍵資源。

效能基準

從Papers with Code收集基準測試和排行榜，了解最新技術進展。

趨勢追蹤

關注NeurIPS、ICML、CVPR、ACL等會議的教學和專題演講，掌握研究趨勢。

研究歸納

使用試算表矩陣歸納論文：方法、數據、計算、指標、程式碼、授權、優缺點。

客觀評估

先撰寫中立摘要，再加入個人觀點，確保研究評估的客觀性。

如何參與社群

參與研討會

輕量級、快速回饋、主題聚焦，是學習新知與交流的絕佳途徑。

參與競賽

迅速掌握特定資料集和評估方法，實踐與學習並重。

重現性挑戰

強化實證研究能力，培養嚴謹的科學態度。

開源貢獻

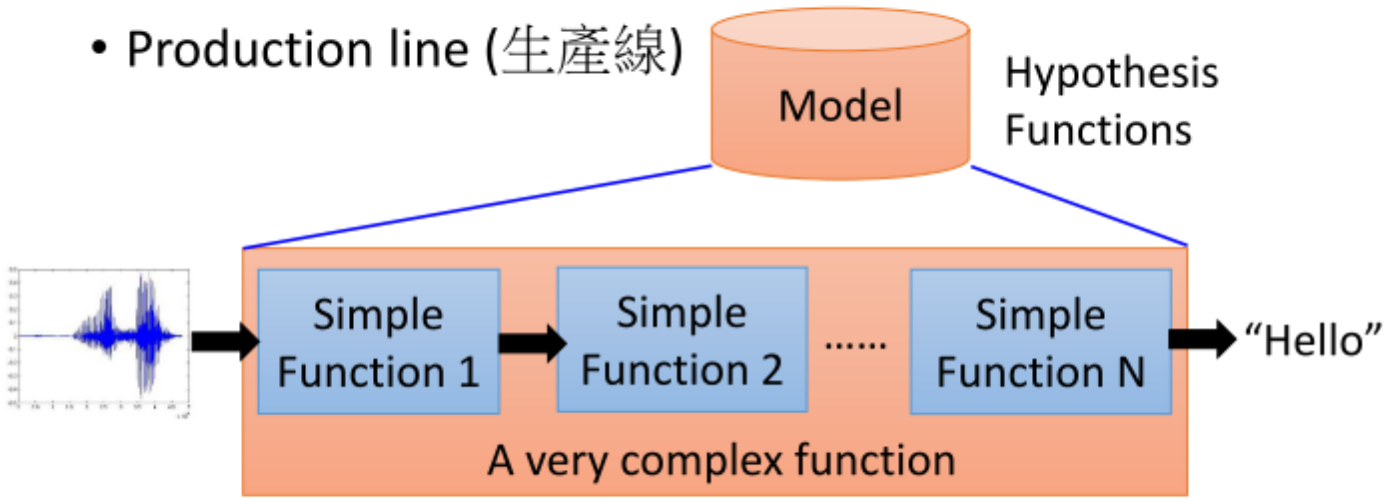
從文件撰寫、測試到小型修復，點滴累積貢獻，學習團隊協作。

學生志工

深入會議評審生態，結識會議組織者與領域專家。

What is Deep Learning?

- Production line (生產線)

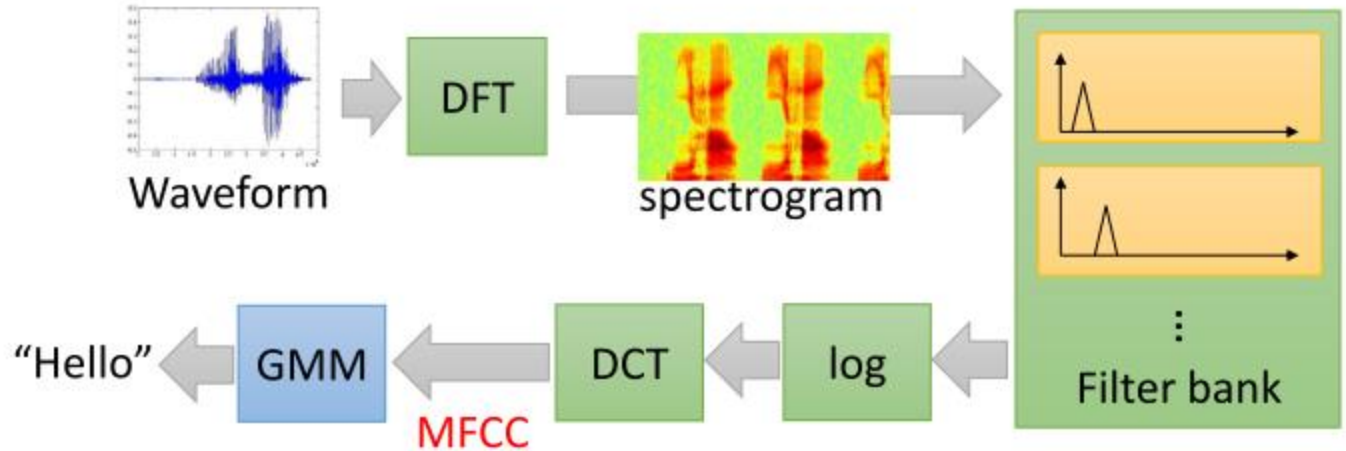


End-to-end training:

What each function should do is learned automatically

Deep v.s. Shallow
- Speech Recognition

- Shallow Approach



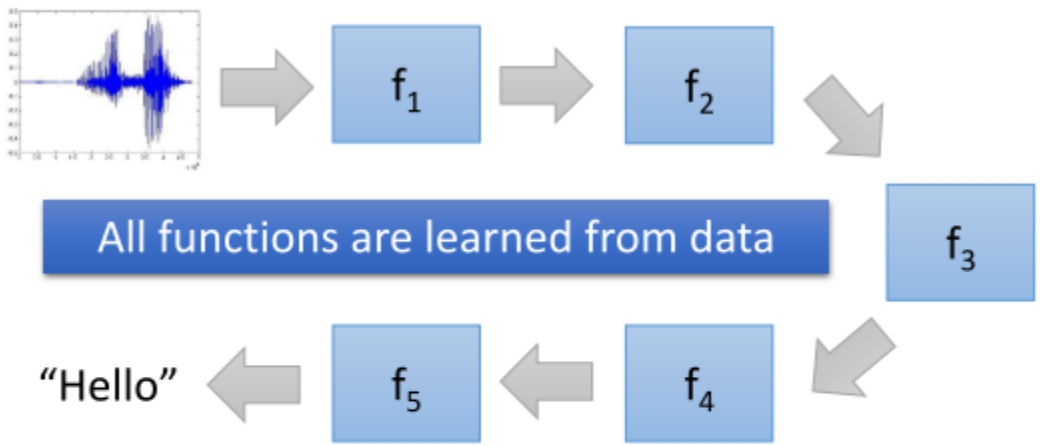
Each box is a simple function in the production line:

:hand-crafted :learned from data

Deep v.s. Shallow - Speech Recognition

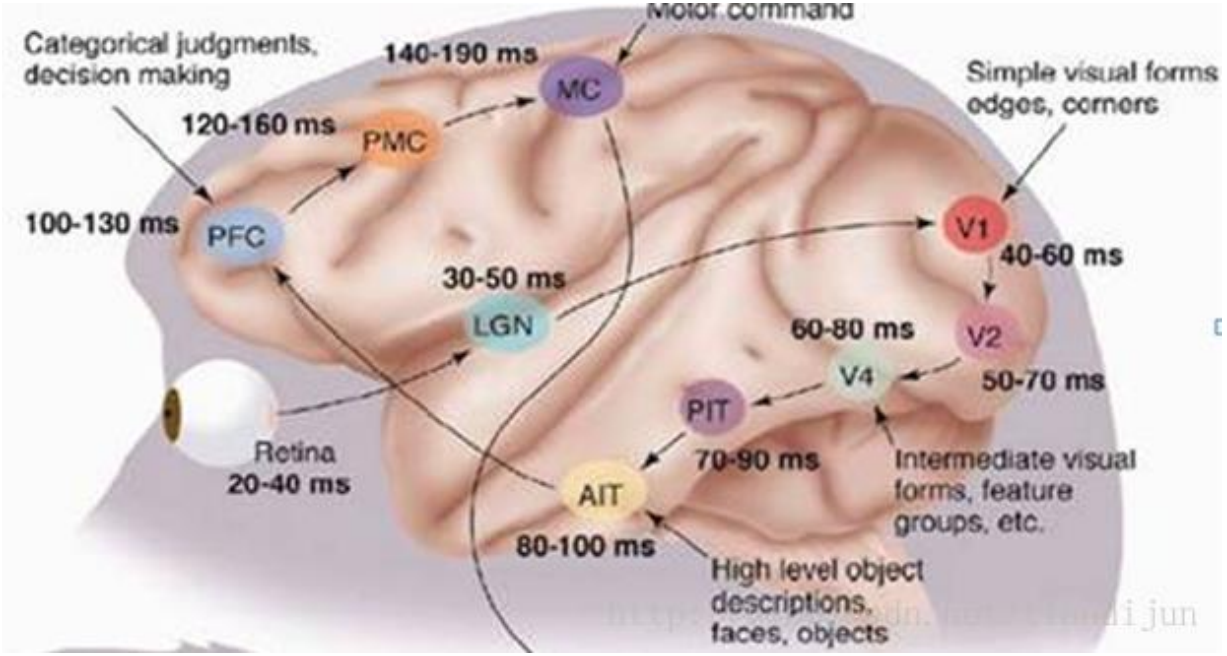
- Deep Learning

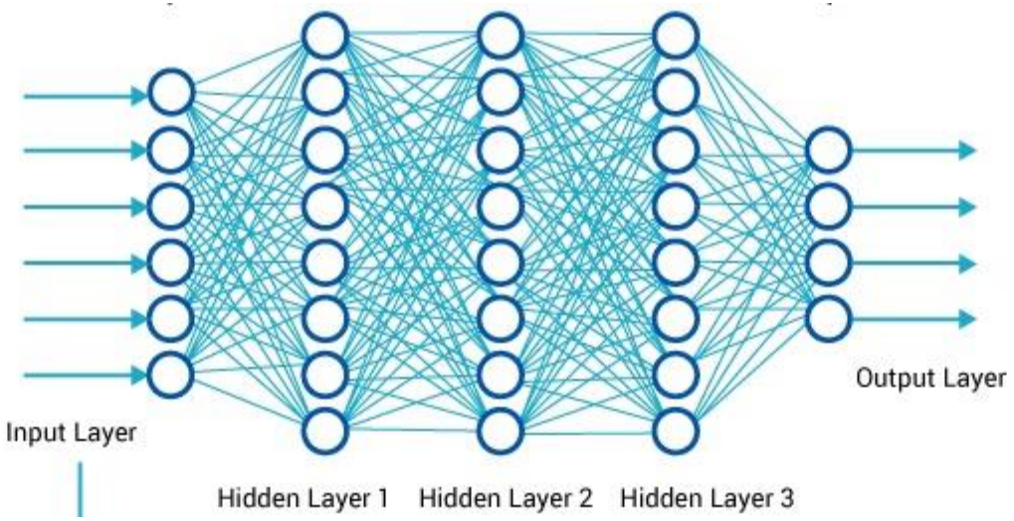
"Bye bye, MFCC"
- Deng Li in Interspeech 2014



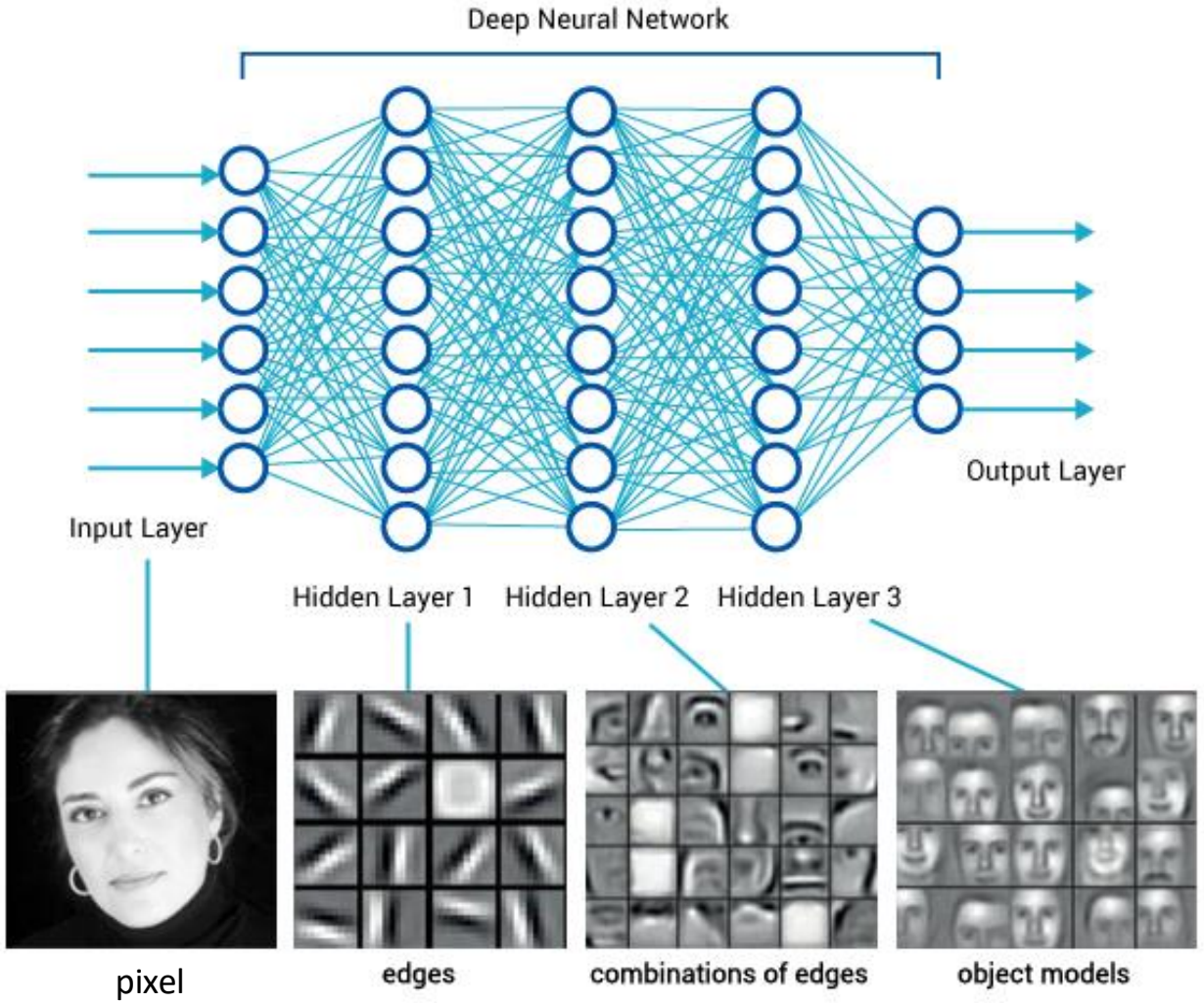
Less engineering labor, but machine learns more

Idea from human brain



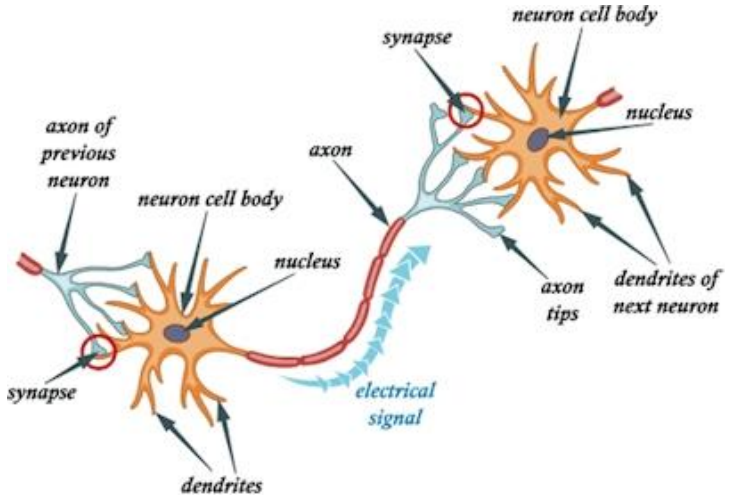


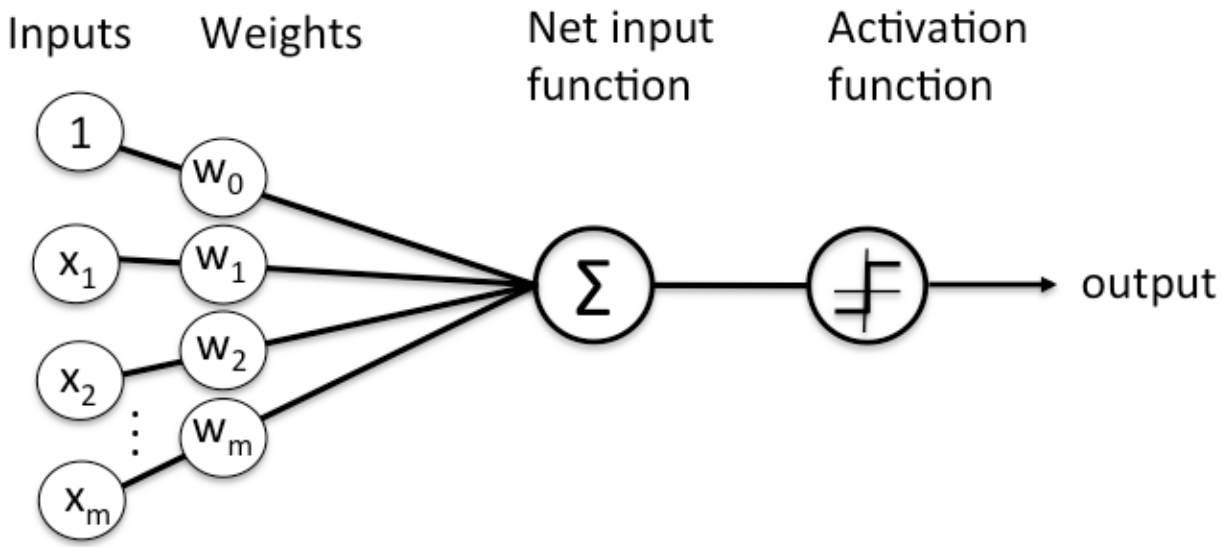
Artificial Neuron Network(ANN)



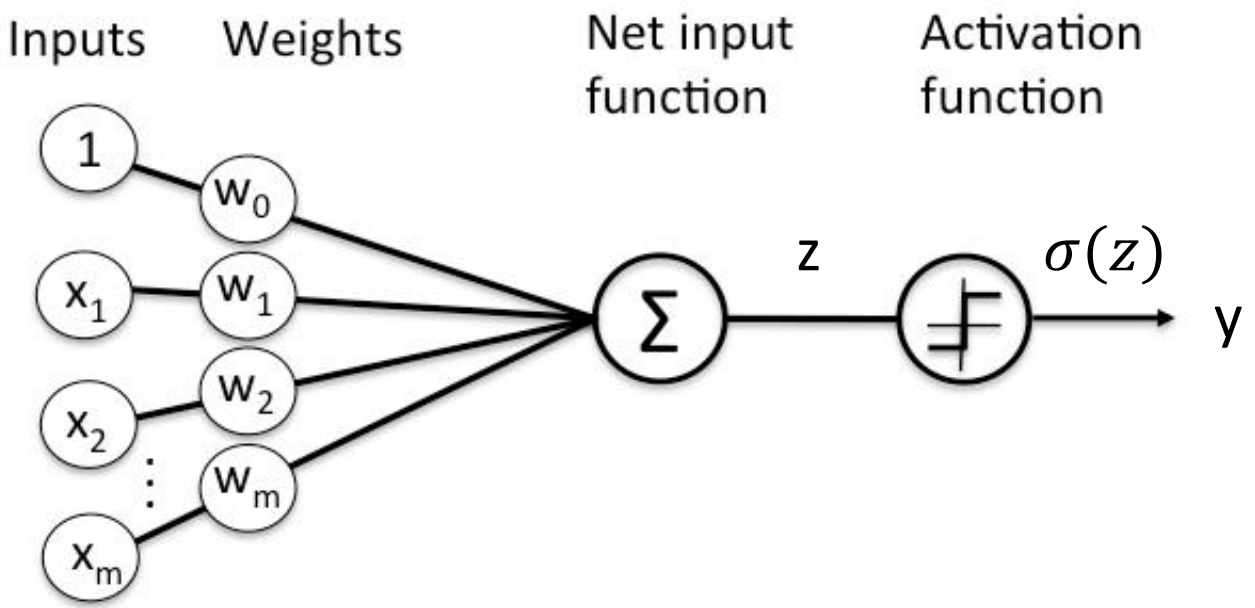
Source:http://www.amax.com/blog/?p=804

Neuron


synapse
neuron cell body
axon of previous neuron
axon
neuron cell body
nucleus
nucleus
axon tips
dendrites of next neuron
synapse
electrical signal
dendrites


Inputs
Weights
Net input function
Activation function
1
w_0
x_1
w_1
x_2
w_2
w_m
x_m
Σ
output

Neuron

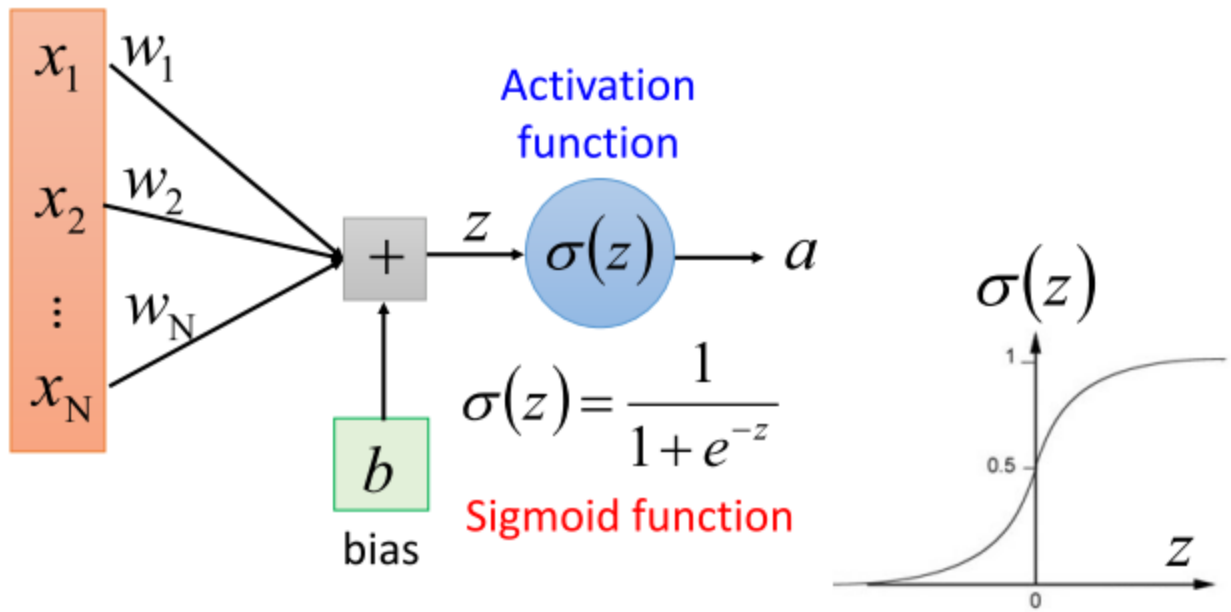


$z = 1*w_0 + x_1*w_1 + x_2*w_2 + \ldots + x_m*w_m$

$y = \sigma(z)$

A Neuron for Machine

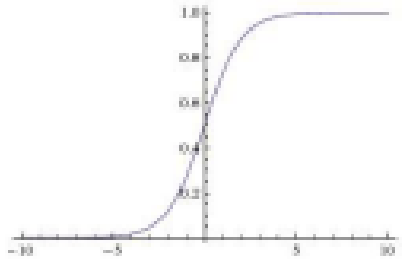
Each neuron is a very simple function



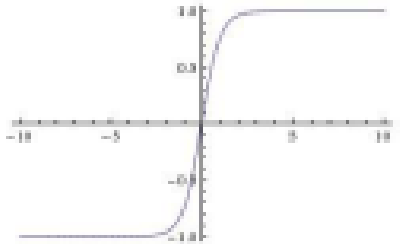
Activation Functions

Sigmoid

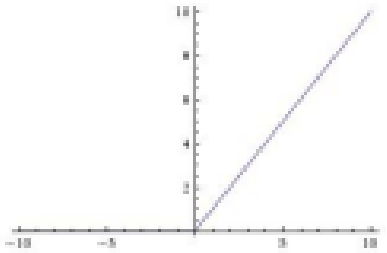
$$\sigma(x) = 1/(1 + e^{-x})$$



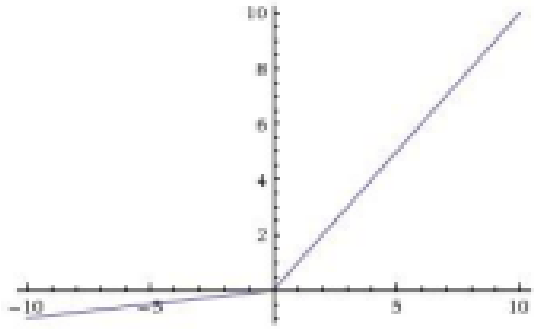
tanh tanh(x)



ReLU max(0,x)



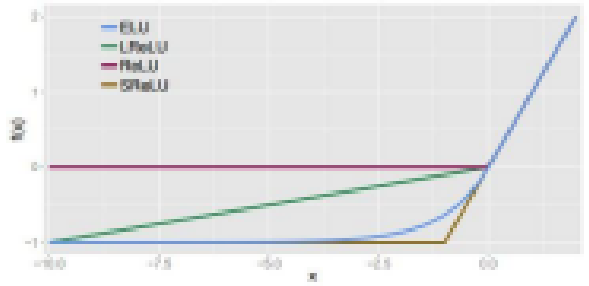
Leaky ReLU
max(0.1x, x)



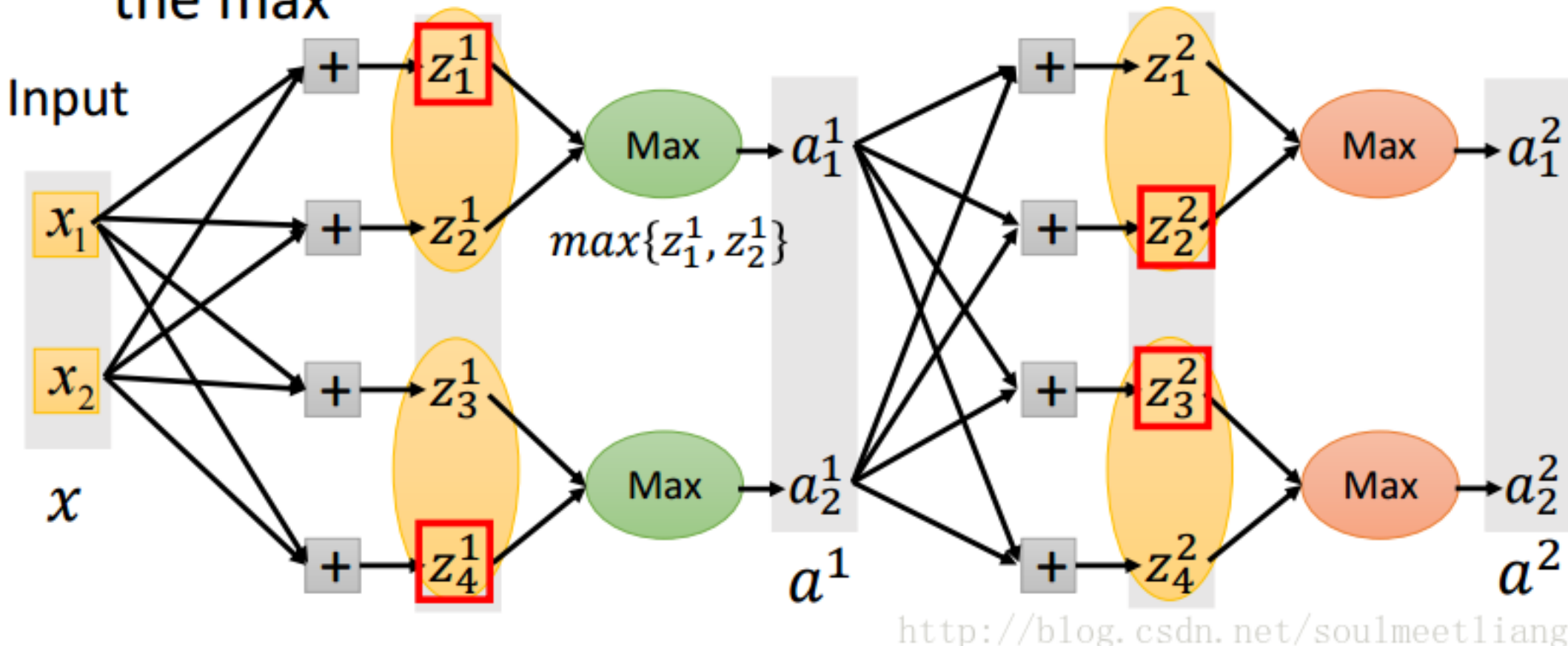
Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$



- Given a training data x, we know which z would be the max



Computer Vision

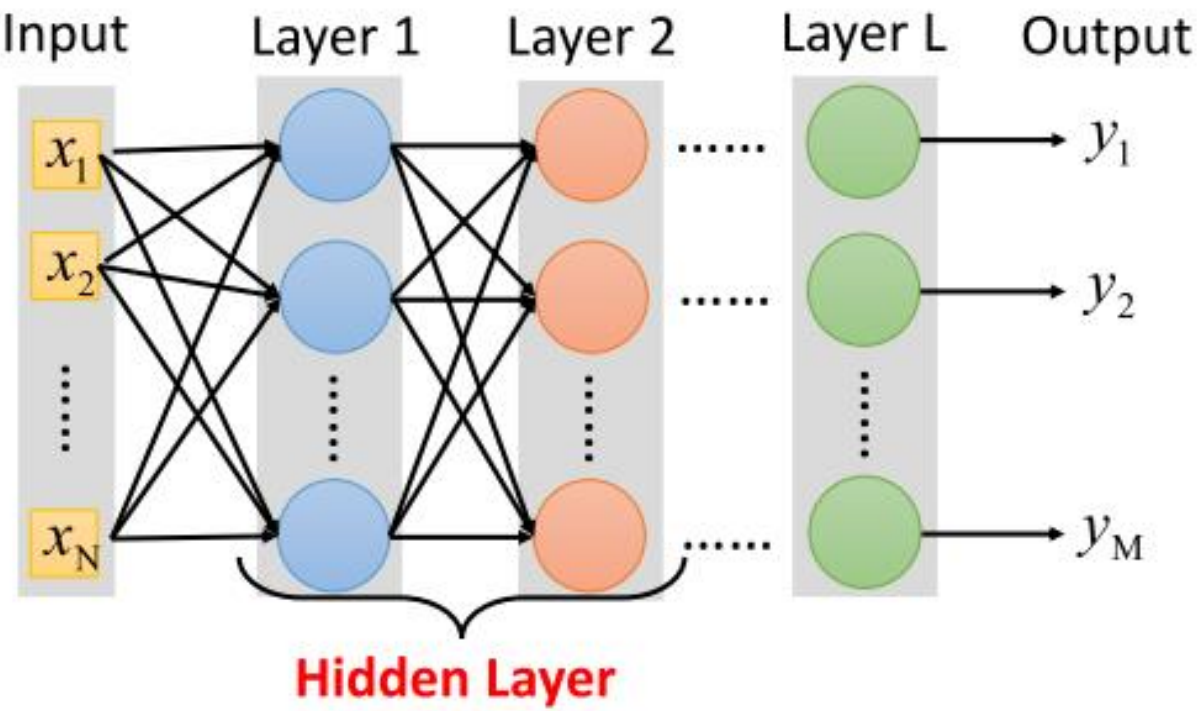
Deep Learning

A neural network is a complex function:

$$f : R^N \rightarrow R^M$$

- Cascading the neurons to form a neural network.

  Each layer is a simple function in the production line.



Ups and downs of Deep Learning

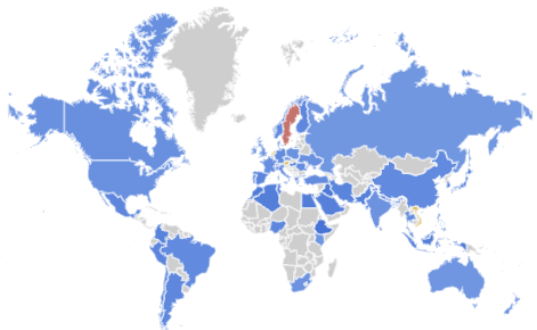
- 1960s: Perceptron (single layer neural network)
- 1969: Perceptron has limitation
- 1980s: Multi-layer perceptron
  - Do not have significant difference from DNN today
- 1986: Backpropagation
  - Usually more than 3 hidden layers is not helpful
- 1989: 1 hidden layer is “good enough”, why deep?
- 2006: RBM initialization (breakthrough)
- 2009: GPU
- 2011: Start to be popular in speech recognition
- 2012: win ILSVRC competition (image)

Become very popular

按區域比較細分資料

區域

deep learning decision tree SVM XGboost



顏色深淺代表搜尋百分比 瞭解詳情

排序依據： 「decision tree」的搜尋熱度

1	瑞典	
2	肯亞	
3	菲律賓	
4	紐西蘭	
5	南非	

顯示第 1-5 個區域，共 62 個

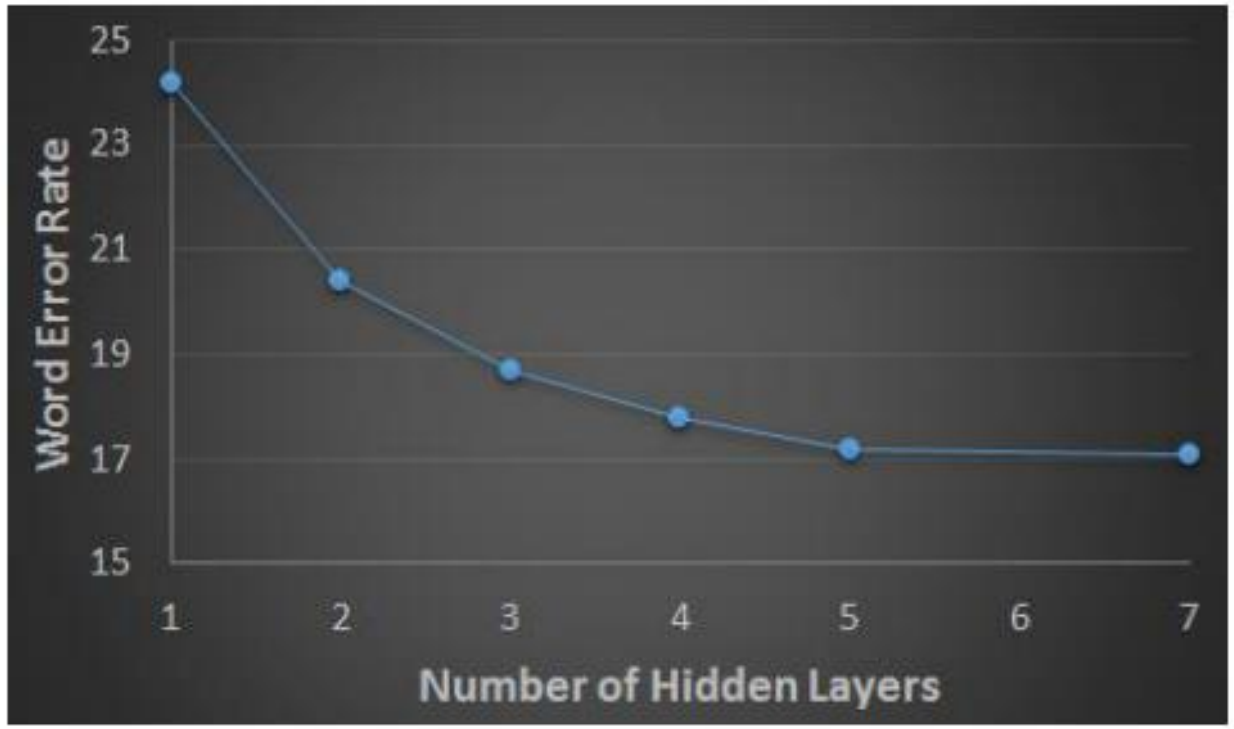
成交	116.26	昨收	116
開盤	116.46	漲跌幅	▲ 0.22%
最高	116.99	漲跌	▲ 0.26
最低	114.86	總量	199,117,313
一年內最高	140.76	一年內最低	39.23
買進	116.18	賣出	122.38
EPS (TTM)	2.13	總市值	2.852T
PE Ratio (TTM)	54.58		

註：TTM (Trailing Twelve Months) 數據是滾動的概念，根據時間的推進而變化。表示連續 12 個月內的統計數據或最近 4 個季度數據。

Why Deep Learning?

Deeper is Better.

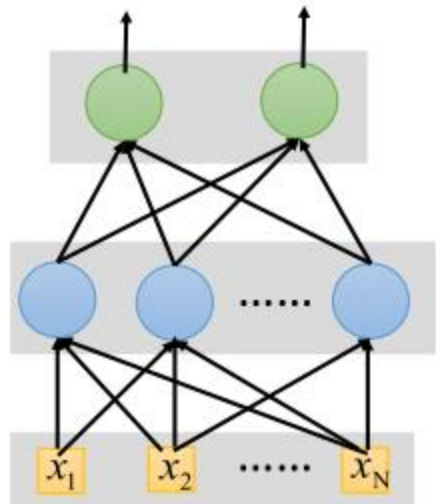
- Speech recognition



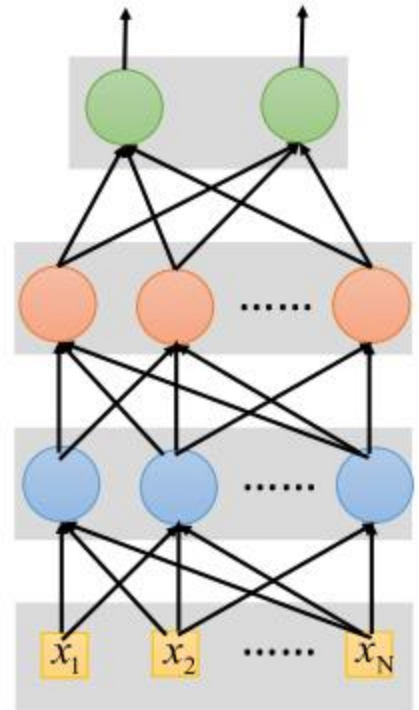
Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Why Deeper is Better?

Deep works better simply because it uses more parameters.



Shallow



Deep

Universality Theorem

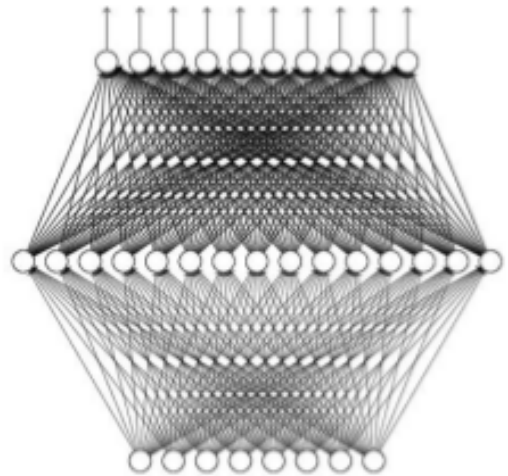
Any continuous function f

$$f : R^N \rightarrow R^M$$

Can be realized by a network with one hidden layer

(given **enough** hidden neurons)

What is the reason to be deep?



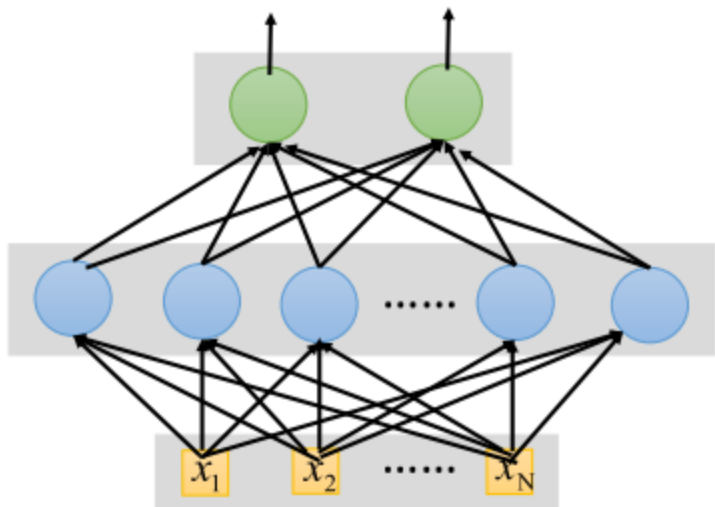
Reference:
http://neuralnetworksanddeeplearning.com/chap4.html

Why “Deep” neural network not “Fat” neural network?

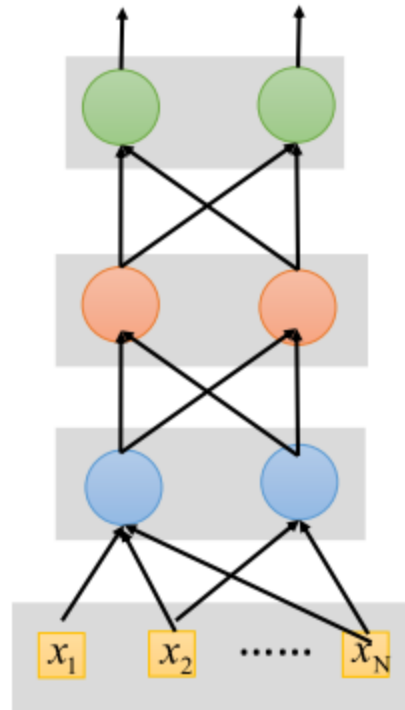
Fat + Short v.s. Thin + Tall

If they have the same parameters,

Which one is better?

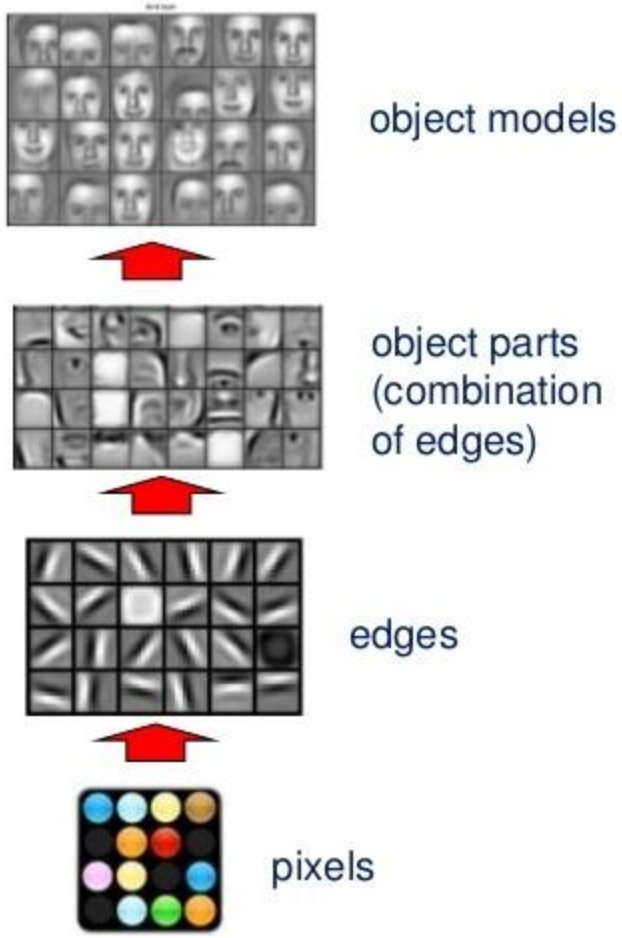


Shallow

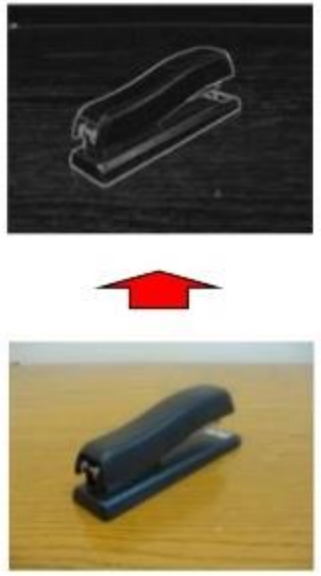


Deep

Why Deep? -> Modularization



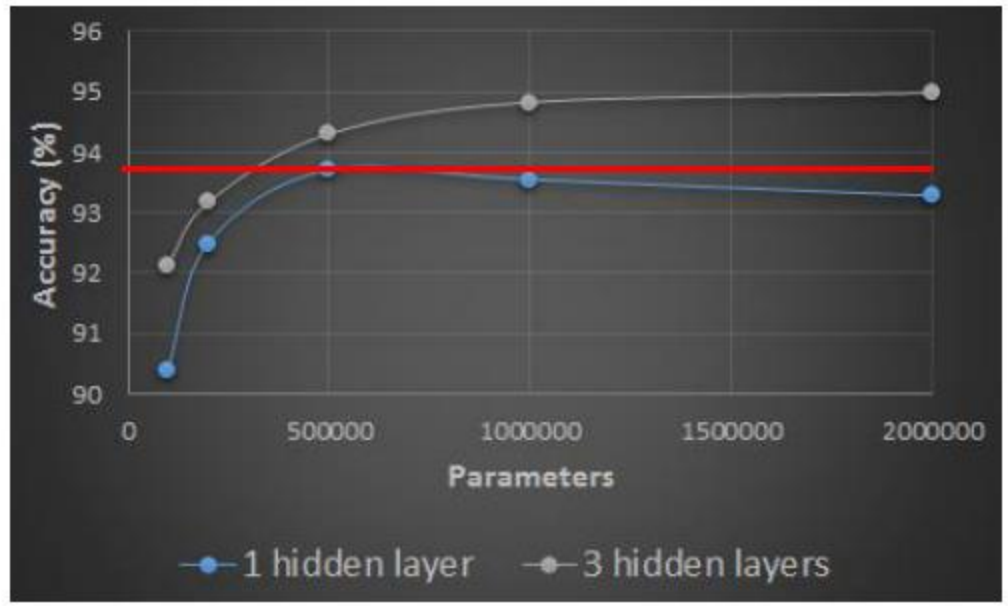
Multiple hidden layer



one hidden layer

Fat + Short v.s. Thin + Tall
Hand-writing digit classification

- Same parameters



Deeper: Using less parameters to achieve the same performance

Fat + Short v.s. Thin + Tall Speech Recognition

- Word error rate (WER)

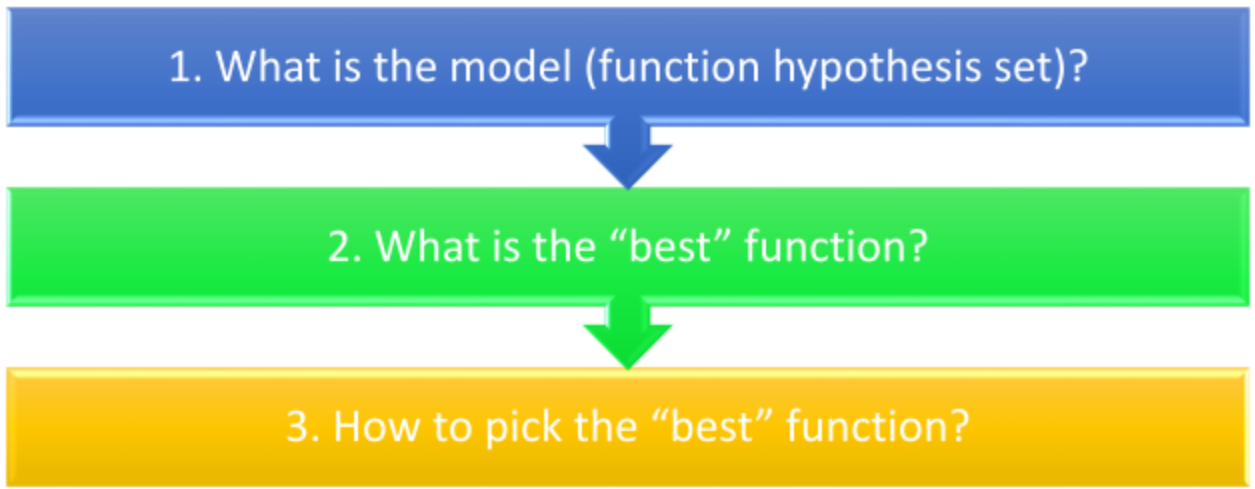
Multiple layers

1 hidden layer

LxN	DBN-PT (%)	1xN	DBN-PT (%)
1×2k	24.2		
2×2k	20.4		
3×2k	18.4		
4×2k	17.8		
5×2k	17.2	1×3,772	22.5
7×2k	17.1	1×4,634	22.6
		1×16K	22.1

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

Outline


1. What is the model (function hypothesis set)?
2. What is the “best” function?
3. How to pick the “best” function?

1. What is the model?

What is the function we are looking for?

- ***classification***

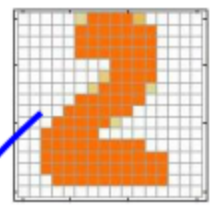
$$y = f(x) \quad \Rightarrow \quad f: R^N \rightarrow R^M$$

  - x: input object to be classified
  - y: class
  - ***Assume both x and y can be represented as fixed-size vector***
    - x is a vector with N dimensions, and y is a vector with M dimensions

What is the function we are looking for?

- ***Handwriting Digit Classification*** $f: R^N \rightarrow R^M$

x: image



16 x 16

Each pixel corresponds to an element in the vector

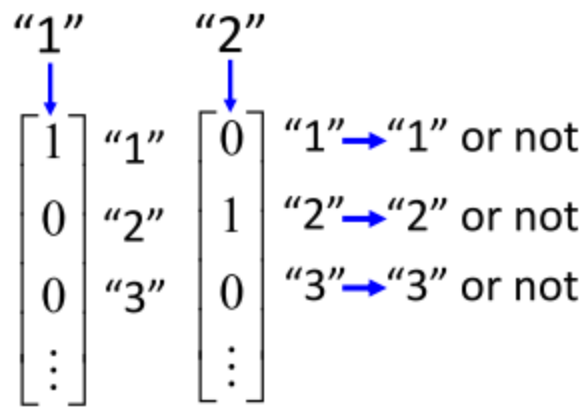
$$\begin{bmatrix} 0 \\ 1 \\ \vdots \end{bmatrix}$$

1: for ink,
0: otherwise

16 x 16 = 256 dimensions

y: class

10 dimensions for digit recognition

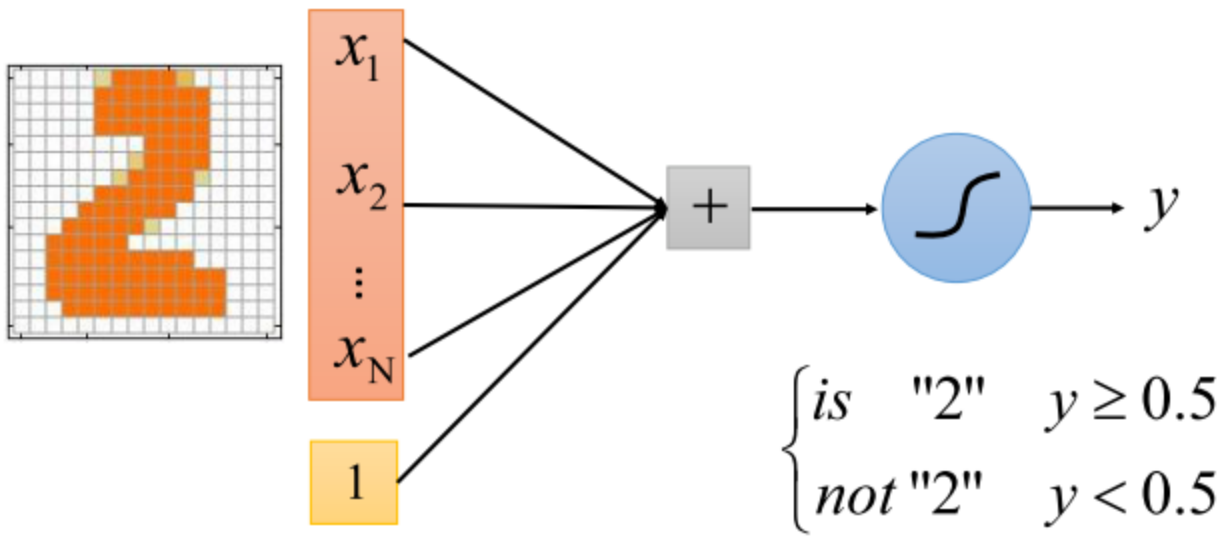


1. What is the model?

A Layer of Neuron

Single Neuron $f: R^N \rightarrow R$

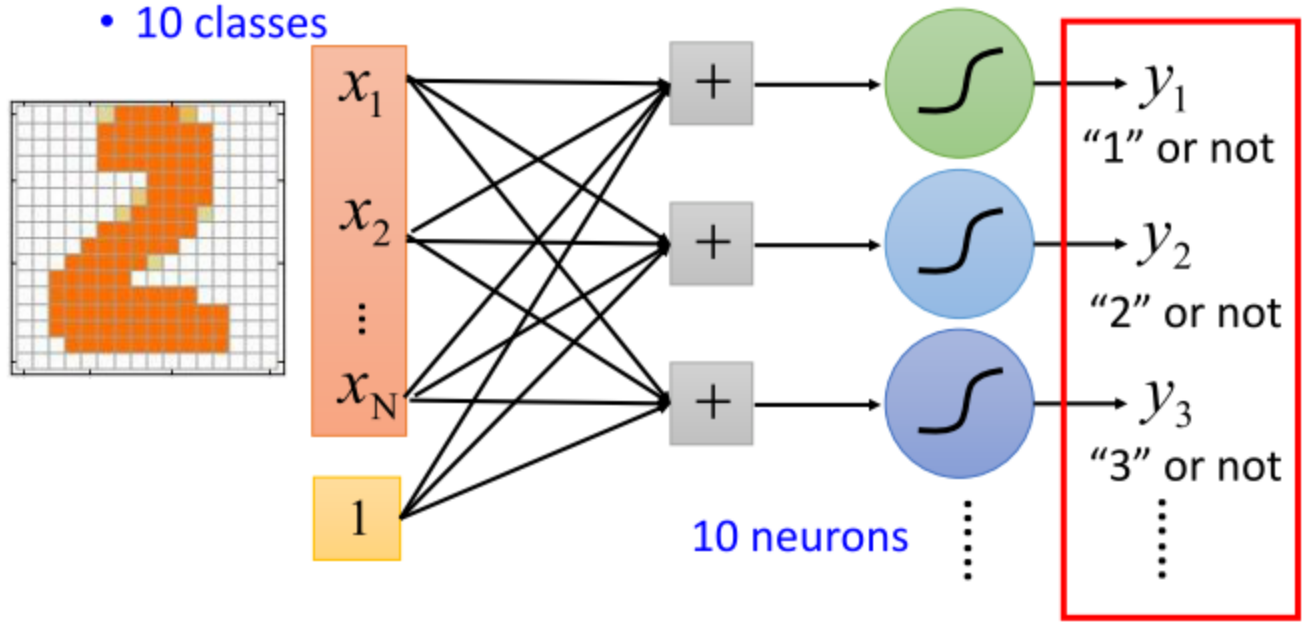
- Single neuron can only do binary classification, cannot handle multi-class classification



A Layer of Neuron $f: R^N \rightarrow R^M$

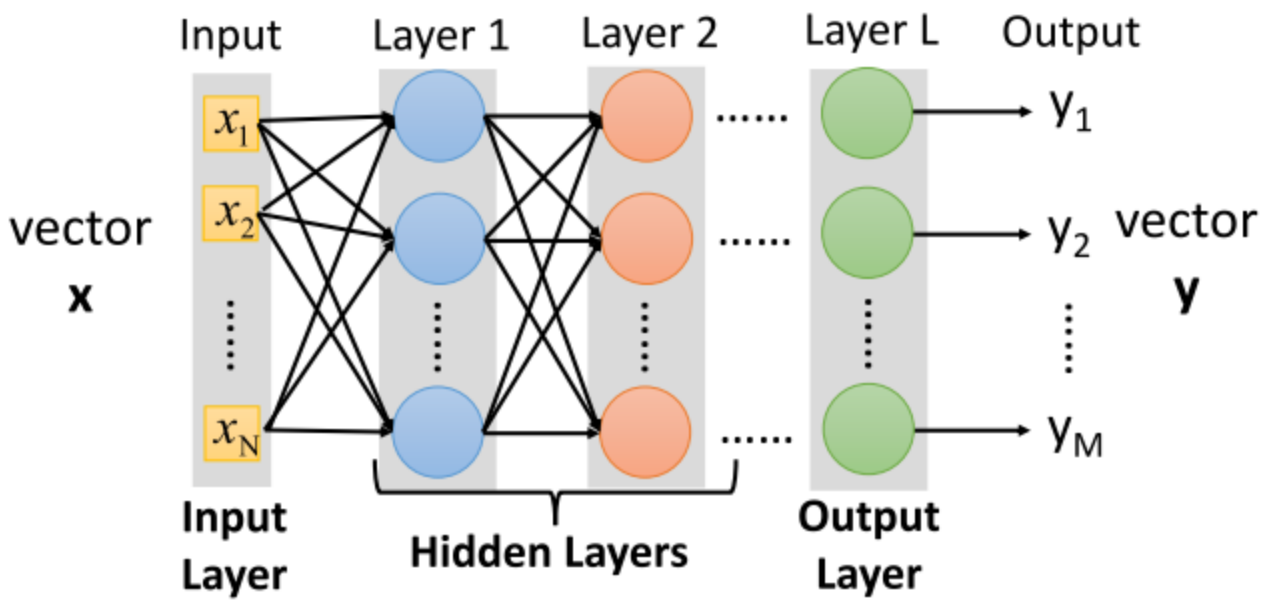
- Handwriting digit classification
  - Classes: "1", "2", …., "9", "0"
  - 10 classes

If y_2 is the max, then the image is "2".



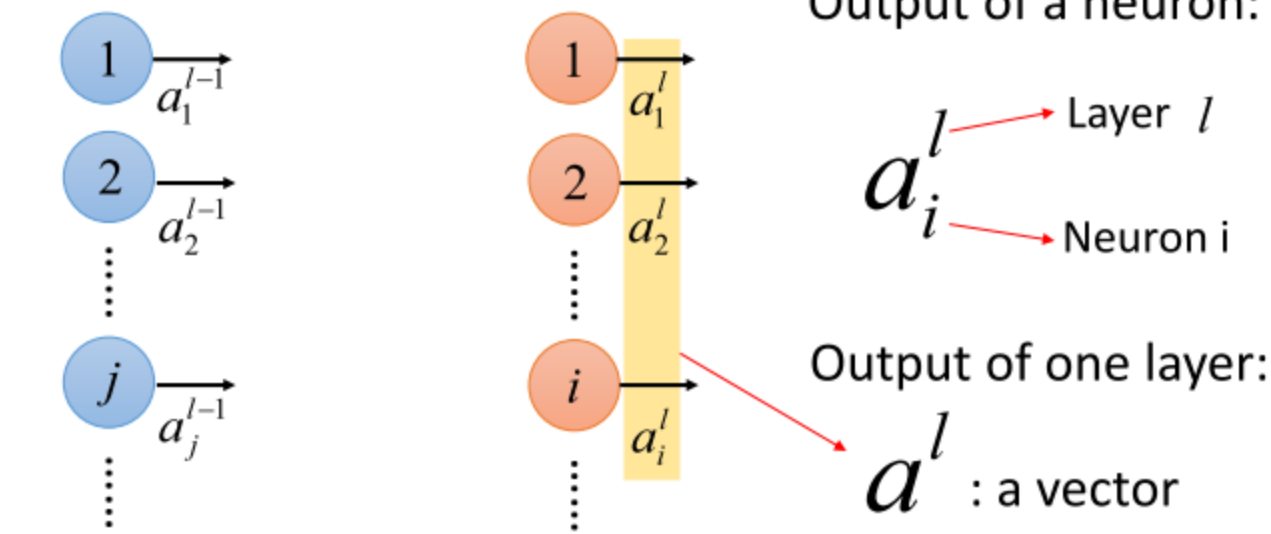
Neural Network as Model

$$f: R^N \rightarrow R^M$$



- Fully connected feedforward network
- Deep Neural Network: many hidden layers

Notation



Layer $l-1$

N_{l-1} nodes

Layer l

N_l nodes

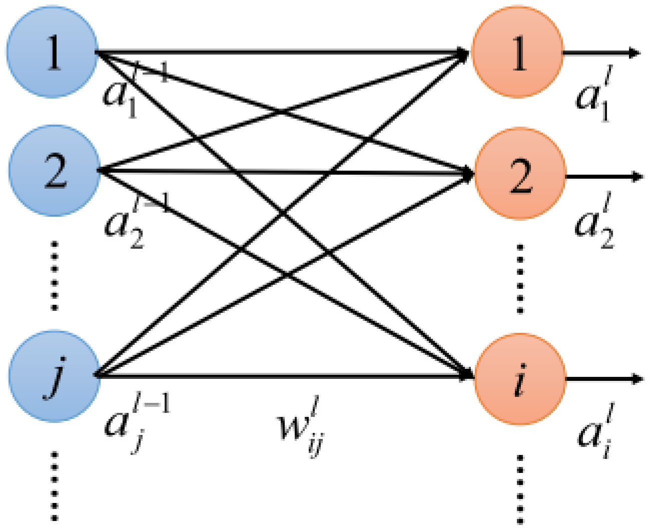
Output of a neuron:

a_i^l — Layer l; Neuron i

Output of one layer:

a^l : a vector

Notation



Layer $l-1$

N_{l-1} nodes

Layer l

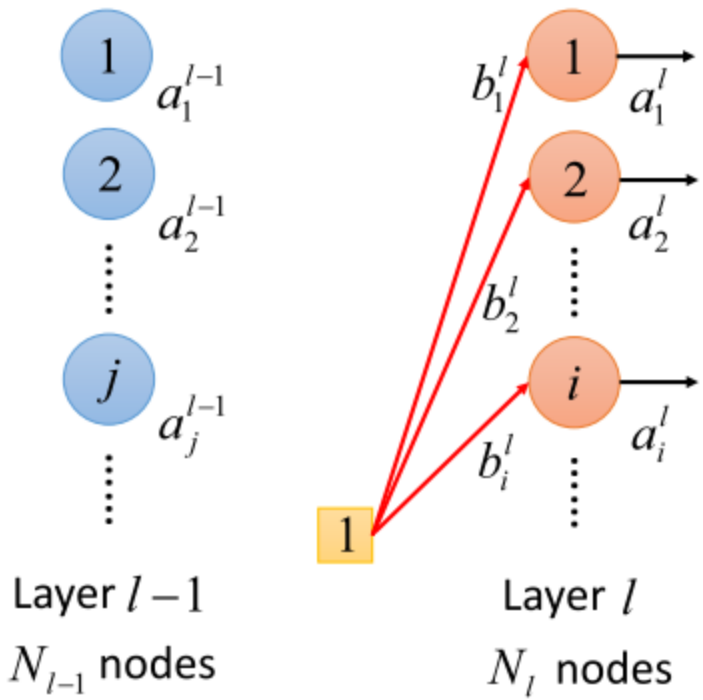
N_l nodes

w_{ij}^{l} → Layer $l-1$ to Layer l

↓ from neuron j (Layer $l-1$) to neuron i (Layer l)

$$W^{l} = \underbrace{\begin{bmatrix} w_{11}^{l} & w_{12}^{l} & \cdots \\ w_{21}^{l} & w_{22}^{l} & \\ \vdots & & \ddots \\ & & \end{bmatrix}}_{\mathrm{N}_{l-1}} \Bigg\} \mathrm{N}_{l}$$

Notation



b_i^l : bias for neuron i at layer l

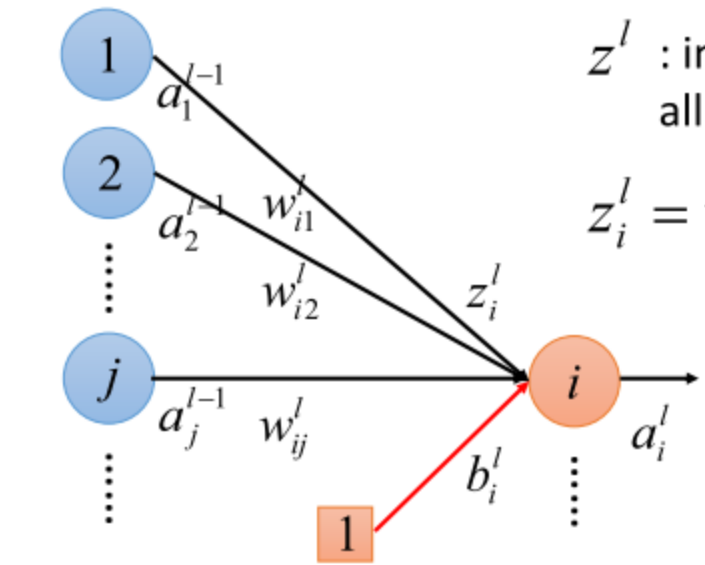
$$b^l = \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

bias for all neurons in layer l

Notation

z_i^l : input of the activation function for neuron i at layer l

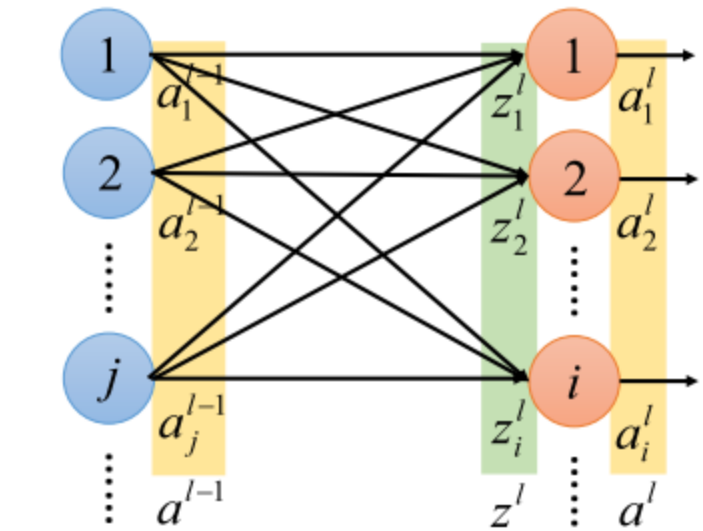
z^l : input of the activation function all the neurons in layer l



$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} \ldots + b_i^l$$

$$z_i^l = \sum_{j=1}^{N_{l-1}} w_{ij}^l a_j^{l-1} + b_i^l$$

Relations between Layer Outputs



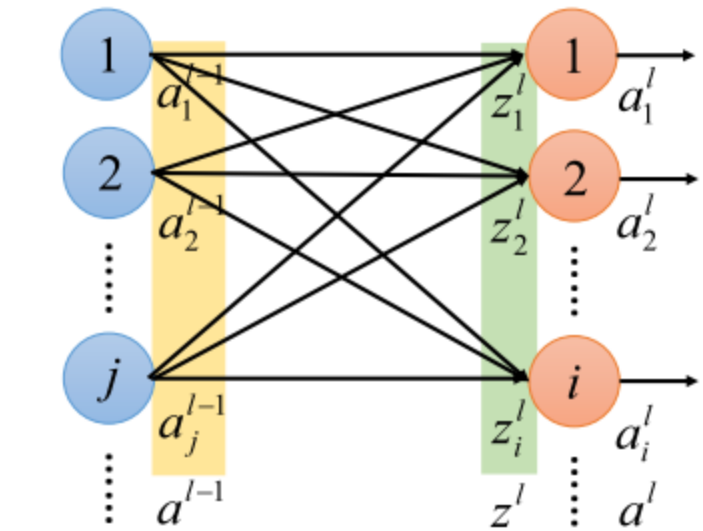
Layer $l-1$

N_{l-1} nodes

Layer l

N_l nodes

Relations between Layer Outputs



Layer $l-1$

N_{l-1} nodes

Layer l

N_l nodes

$$z_1^l = w_{11}^l a_1^{l-1} + w_{12}^l a_2^{l-1} + \cdots + b_1^l$$

$$z_2^l = w_{21}^l a_1^{l-1} + w_{22}^l a_2^{l-1} + \cdots + b_2^l$$

$$\vdots$$

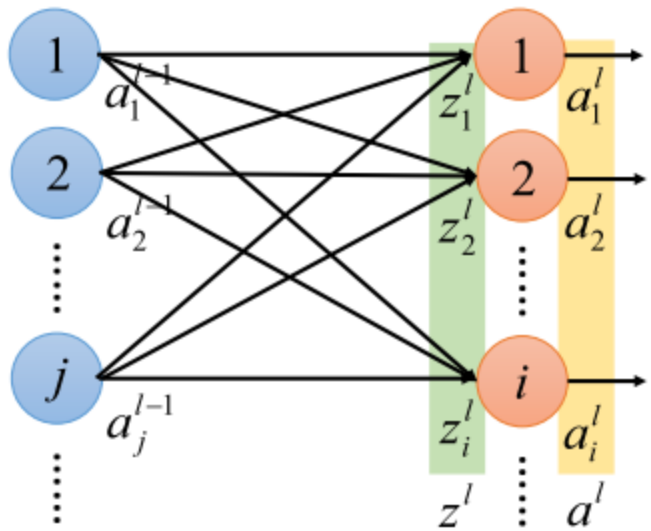
$$z_i^l = w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} + \cdots + b_i^l$$

$$\vdots$$

$$\begin{bmatrix} z_1^l \\ z_2^l \\ \vdots \\ z_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \end{bmatrix} \begin{bmatrix} a_1^{l-1} \\ a_2^{l-1} \\ \vdots \\ a_i^{l-1} \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

$$z^l = W^l a^{l-1} + b^l$$

Relations between Layer Outputs



Layer $l-1$

N_{l-1} nodes

Layer l

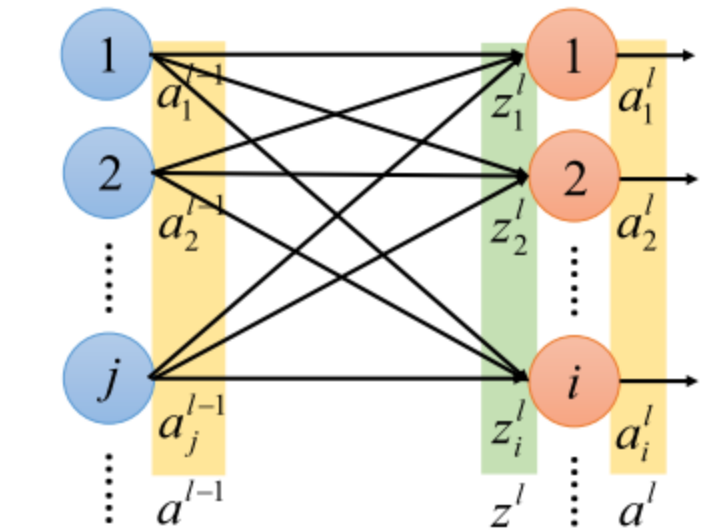
N_l nodes

$$a_i^l = \sigma\left(z_i^l\right)$$

$$\begin{bmatrix} a_1^l \\ a_2^l \\ \vdots \\ a_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} \sigma\left(z_1^l\right) \\ \sigma\left(z_2^l\right) \\ \vdots \\ \sigma\left(z_i^l\right) \\ \vdots \end{bmatrix}$$

$$a^l = \sigma\left(z^l\right)$$

Relations between Layer Outputs



Layer $l-1$

N_{l-1} nodes

Layer l

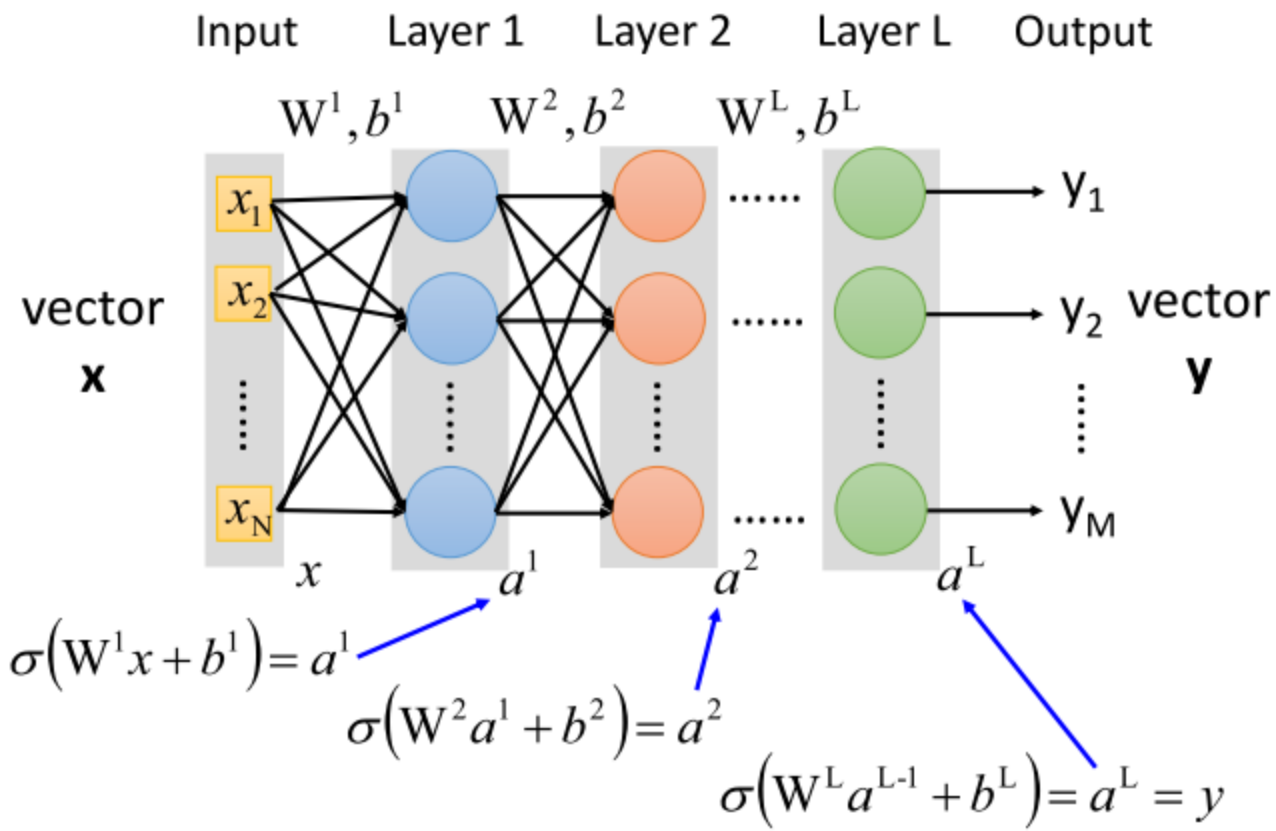
N_l nodes

$$z^l = W^l a^{l-1} + b^l$$

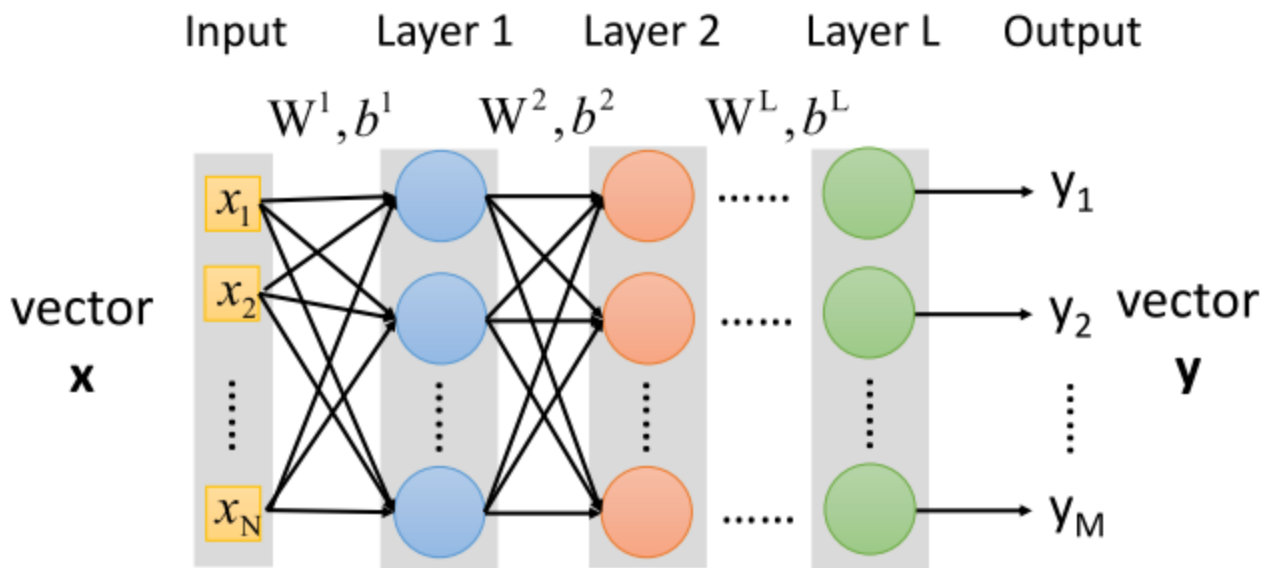
$$a^l = \sigma\left(z^l\right)$$

$$a^l = \sigma\left(W^l a^{l-1} + b^l\right)$$

Function of Neural Network



Function of Neural Network



$$y = f(x)$$
$$= \sigma\left(W^L \ldots \sigma\left(W^2 \sigma\left(W^1 x + b^1\right) + b^1\right) \ldots + b^L\right)$$

2. What is the “best” function?

Best Function = Best Parameters

$$y = f(x) = \sigma\left(W^L \ldots \sigma\left(W^2 \sigma\left(W^1 x + b^1\right) + b^1\right) \ldots + b^L\right)$$

function set

because different parameters W and b lead to different function

Formal way to define a function set:

$f(x;\theta)$ → parameter set

$$\theta = \left\{W^1, b^1, W^2, b^2 \cdots W^L, b^L\right\}$$

Pick the "best" function f*

Pick the "best" parameter set θ*

Cost Function

- Define a function for parameter set $C(\theta)$
  - $C(\theta)$ evaluate how bad a parameter set is
  - The best parameter set θ^* is the one that minimizes $C(\theta)$

$$\theta^* = arg \min_{\theta} C(\theta)$$

- $C(\theta)$ is called ***cost/loss/error function***
  - If you define the goodness of the parameter set by another function $O(\theta)$
  - $O(\theta)$ is called objective function

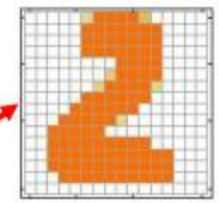
Cost Function

Given training data:

$$\left\{\left(x^1,\hat{y}^1\right)\ldots\left(x^r,\hat{y}^r\right)\ldots\left(x^R,\hat{y}^R\right)\right\}$$

- ***Handwriting Digit Classification***

sum over all training examples



$$C(\theta)=\frac{1}{R}\sum_{r}\left\|f\left(x^r;\theta\right)-\hat{y}^r\right\|$$

Minimize distance

$$\begin{bmatrix}0.1\\0.4\\0.2\\\vdots\end{bmatrix}\begin{matrix}\text{“1”}\\\text{“2”}\\\text{“3”}\\\end{matrix}$$

$$\begin{bmatrix}0\\1\\0\\\vdots\end{bmatrix}\begin{matrix}\text{“1”}\\\text{“2”}\\\text{“3”}\\\end{matrix}$$

3. How to pick the “best” function?

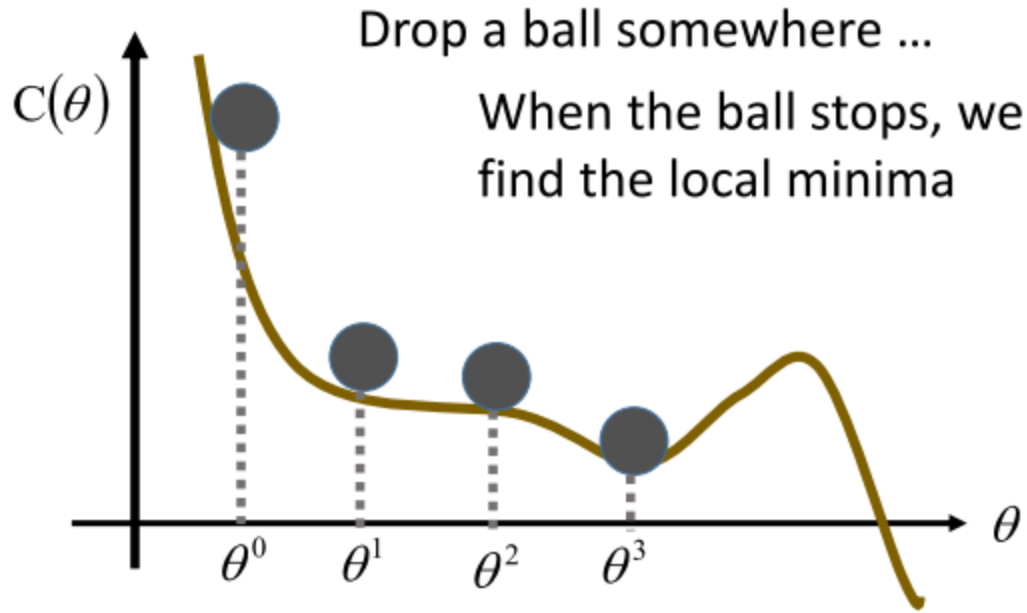
Gradient Descent

Statement of Problems

- Statement of problems:
  - There is a function C(θ)
    - θ represents parameter set
    - $\theta = \{\theta_1, \theta_2, \theta_3, \ldots\ldots\}$
  - Find θ^* that minimizes C(θ)
- Brute force?
  - Enumerate all possible θ
- Calculus?
  - Find θ^* such that $\left.\frac{\partial C(\theta)}{\partial \theta_1}\right|_{\theta=\theta^*} = 0, \left.\frac{\partial C(\theta)}{\partial \theta_2}\right|_{\theta=\theta^*} = 0, \ldots$

Gradient Descent – Idea

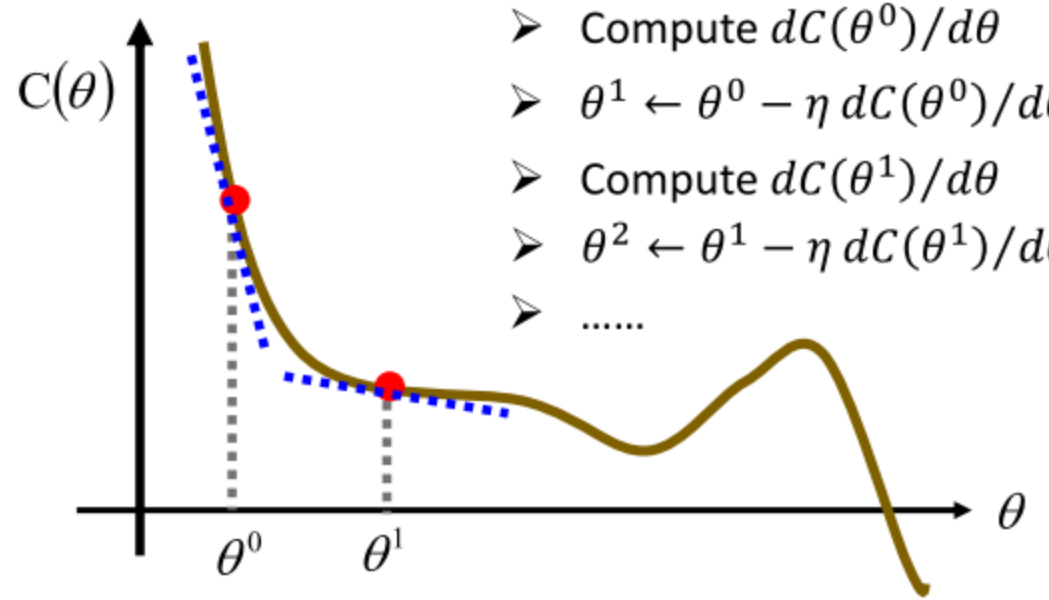
- For simplification, first consider that θ has only one variable



Gradient Descent – Idea

η is called *"learning rate"*

- For simplification, first consider that θ has only one variable



- Randomly start at θ^0
- Compute $dC(\theta^0)/d\theta$
- $\theta^1 \leftarrow \theta^0 - \eta\, dC(\theta^0)/d\theta$
- Compute $dC(\theta^1)/d\theta$
- $\theta^2 \leftarrow \theta^1 - \eta\, dC(\theta^1)/d\theta$
- ……

Gradient Descent

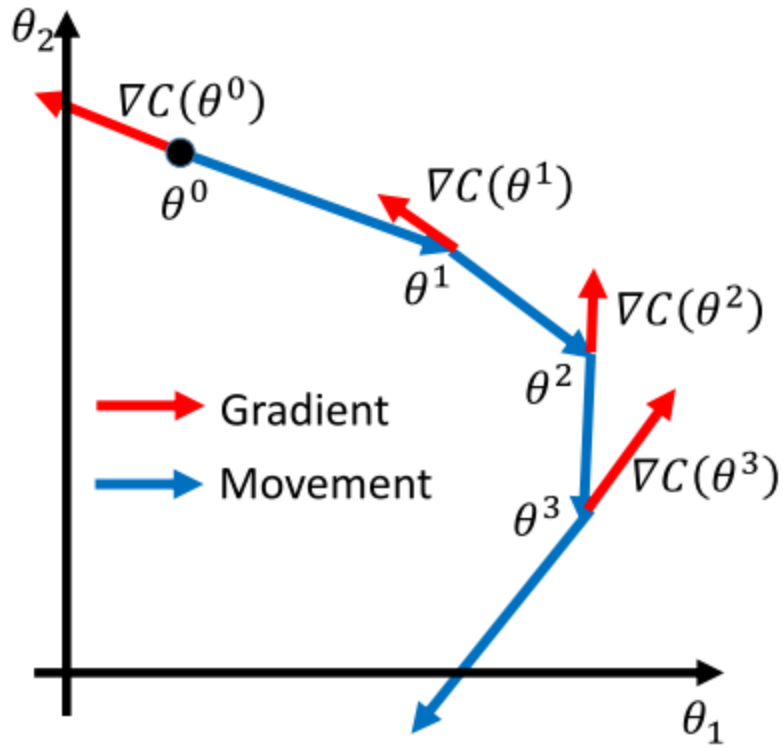
- Suppose that θ has two variables $\{\theta_1, \theta_2\}$

- Randomly start at $\theta^0 = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix}$
- Compute the gradients of $C(\theta)$ at θ^0: $\nabla C(\theta^0) = \begin{bmatrix} \partial C(\theta_1^0)/\partial\theta_1 \\ \partial C(\theta_2^0)/\partial\theta_2 \end{bmatrix}$
- Update parameters

$$\begin{bmatrix} \theta_1^1 \\ \theta_2^1 \end{bmatrix} = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix} - \eta \begin{bmatrix} \partial C(\theta_1^0)/\partial\theta_1 \\ \partial C(\theta_2^0)/\partial\theta_2 \end{bmatrix} \Rightarrow \theta^1 = \theta^0 - \eta\nabla C(\theta^0)$$

- Compute the gradients of $C(\theta)$ at θ^1: $\nabla C(\theta^1) = \begin{bmatrix} \partial C(\theta_1^1)/\partial\theta_1 \\ \partial C(\theta_2^1)/\partial\theta_2 \end{bmatrix}$
- ……

Gradient Descent



Start at position θ^0

Compute gradient at θ^0

Move to $\theta^1 = \theta^0 - \eta\nabla C(\theta^0)$

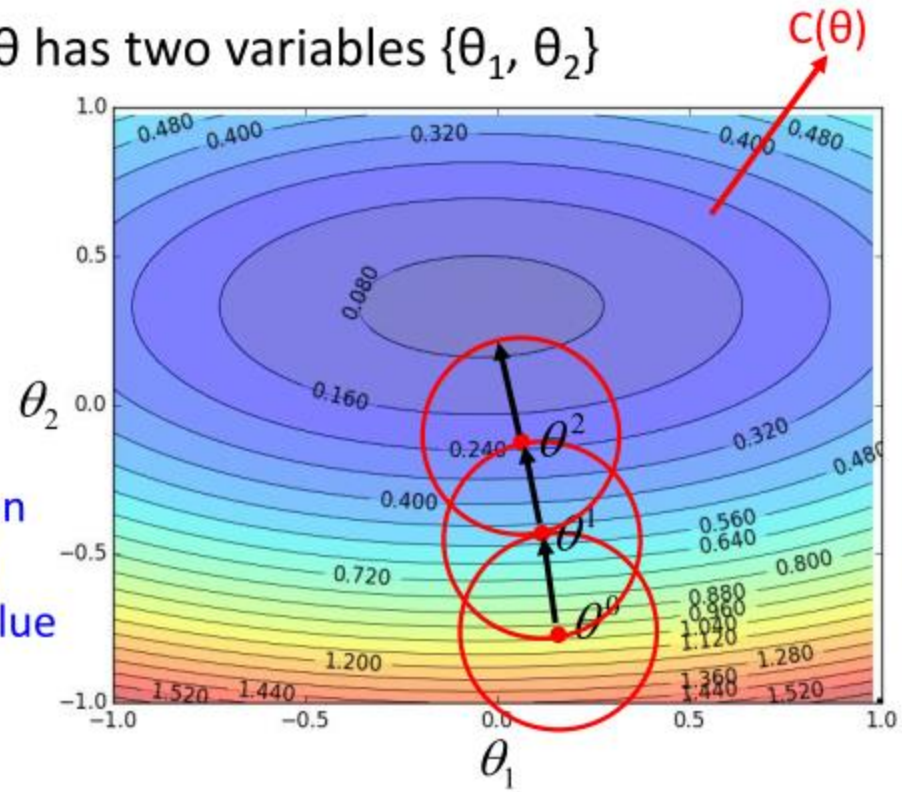
Compute gradient at θ^1

Move to $\theta^2 = \theta^1 - \eta\nabla C(\theta^1)$

⋮

Formal Derivation of Gradient Descent

- Suppose that θ has two variables $\{\theta_1, \theta_2\}$



Given a point, we can easily find the point with the smallest value nearby. How?

Gradient Descent

This is the “learning” of machines in deep learning ……

➡ Even alpha go using this approach.

I hope you are not too disappointed :p


380
280
0
150
10/12
0
工具時代
Gilfinz: 166/135
IMR D4RKN3Ss: 141/135
orcsbase: 123/135
BSV Baller: 113/135
127/134
110/134
市鎮中心
orcsbase (南韓人)
用來創建村民和推進到下一個時代。同時也是村民存放食物、木材、黃金及石料的地方。1座市鎮中心支持4個單位。
耗費：200
600/600

Formal Derivation of Gradient Descent

- **Taylor series**: Let h(x) be infinitely differentiable around x = x_0.

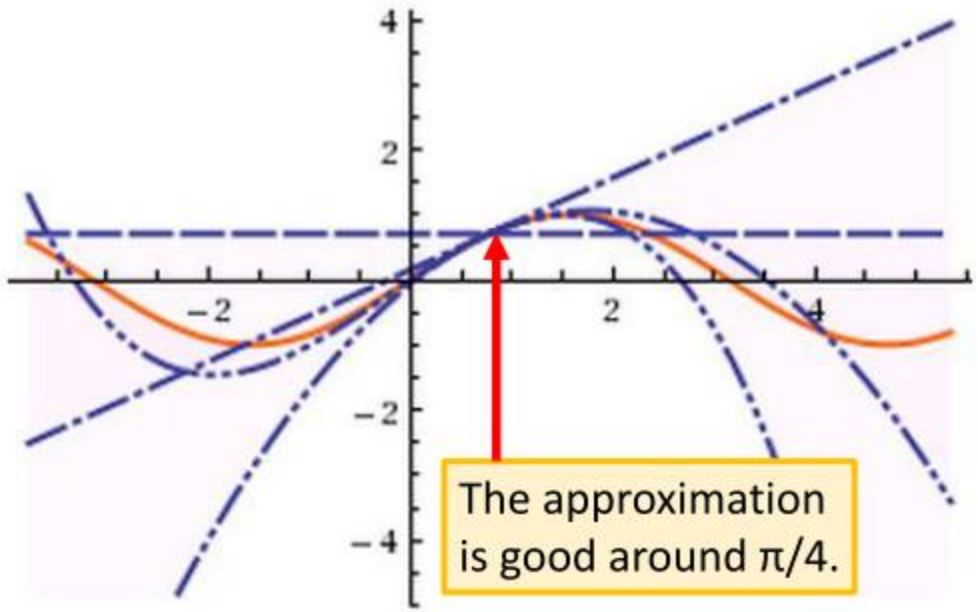
$$h(x) = \sum_{k=0}^{\infty} \frac{h^{(k)}(x_0)}{k!}(x - x_0)^k$$

$$= h(x_0) + h'(x_0)(x - x_0) + \frac{h''(x_0)}{2!}(x - x_0)^2 + \dots$$

When x is close to x_0 ⇒ $h(x) \approx h(x_0) + h'(x_0)(x - x_0)$

E.g. Taylor series for h(x)=sin(x) around $x_0=\pi/4$

$$\sin(x)= \frac{1}{\sqrt{2}} + \frac{x-\frac{\pi}{4}}{\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^2}{2\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^3}{6\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^4}{24\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^5}{120\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^6}{720\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^7}{5040\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^8}{40\,320\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^9}{362\,880\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^{10}}{3\,628\,800\sqrt{2}} + \ldots\ldots$$



Multivariable Taylor series

$$h(x,y) = h(x_0,y_0) + \frac{\partial h(x_0,y_0)}{\partial x}(x-x_0) + \frac{\partial h(x_0,y_0)}{\partial y}(y-y_0)$$

+ something related to $(x-x_0)^2$ and $(y-y_0)^2$ + ……

When x and y is close to x_0 and y_0

$$h(x,y) \approx h(x_0,y_0) + \frac{\partial h(x_0,y_0)}{\partial x}(x-x_0) + \frac{\partial h(x_0,y_0)}{\partial y}(y-y_0)$$

Formal Derivation of Gradient Descent

Based on Taylor Series:

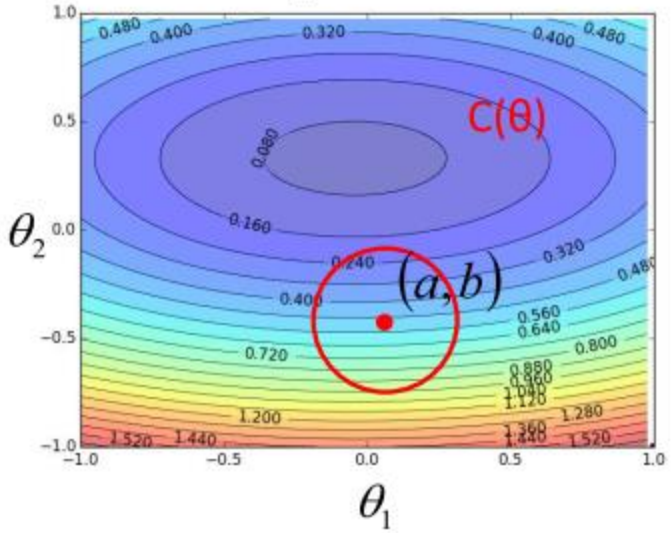
If the red circle is ***small enough***, in the red circle

$$C(\theta) \approx C(a,b) + \frac{\partial C(a,b)}{\partial \theta_1}(\theta_1 - a) + \frac{\partial C(a,b)}{\partial \theta_2}(\theta_2 - b)$$

$$s = C(a,b)$$

$$u = \frac{\partial C(a,b)}{\partial \theta_1}, v = \frac{\partial C(a,b)}{\partial \theta_2}$$

$$C(\theta) \approx s + u(\theta_1 - a) + v(\theta_2 - b)$$



Formal Derivation of Gradient Descent

$$u=\frac{\partial C(a,b)}{\partial \theta_1}, v=\frac{\partial C(a,b)}{\partial \theta_2}$$

Based on Taylor Series:

If the red circle is ***small enough***, in the red circle

$$C(\theta)\approx s+u(\theta_1-a)+v(\theta_2-b)$$

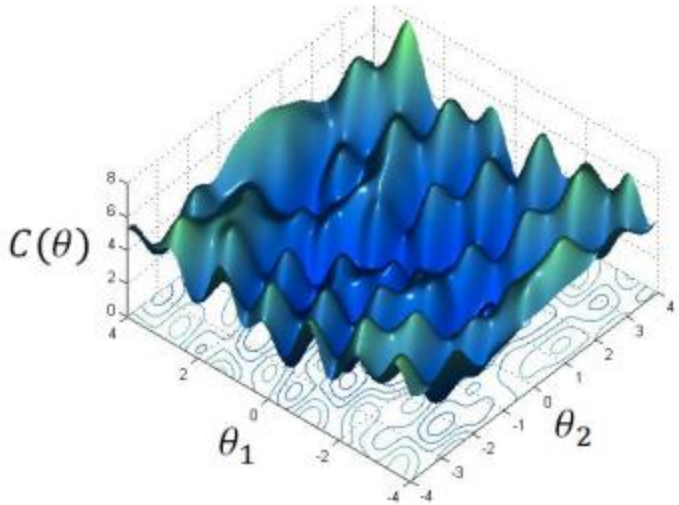
Find θ_1 and θ_2 yielding the smallest value of $C(\theta)$ in the circle

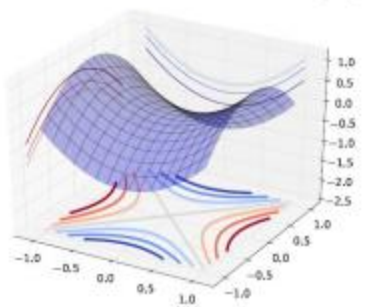
$$\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}=\begin{bmatrix}a\\ b\end{bmatrix}-\eta\begin{bmatrix}u\\ v\end{bmatrix}=\begin{bmatrix}a\\ b\end{bmatrix}-\eta\begin{bmatrix}\frac{\partial C(a,b)}{\partial \theta_1}\\ \frac{\partial C(a,b)}{\partial \theta_2}\end{bmatrix}=\begin{bmatrix}a\\ b\end{bmatrix}-\eta\nabla C(a,b)$$

Its value depending on the radius of the circle, u and v.

This is how gradient descent updates parameters.

Stuck at local minima?





Saddle point

- Who is Afraid of Non-Convex Loss Functions?
- http://videolectures.net/eml07_lecun_wia/
- Deep Learning: Theoretical Motivations
- http://videolectures.net/deeplearning2015_bengio_theoretical_motivations/

Gradient Descent for Neural Network

$$Compute\ \nabla C(\theta^0)$$
$$\theta^1 = \theta^0 - \eta \nabla C(\theta^0)$$
$$Compute\ \nabla C(\theta^1)$$
$$\theta^2 = \theta^1 - \eta \nabla C(\theta^1)$$

Starting Parameters $\theta^0 \longrightarrow \theta^1 \longrightarrow \theta^2 \longrightarrow \cdots\cdots$

$$\nabla C(\theta) \qquad \theta = \left\{ W^1, b^1, W^2, b^2, \cdots, W^l, b^l, \cdots, W^L, b^L \right\}$$

$$= \begin{bmatrix} \vdots \\ \dfrac{\partial C(\theta)}{\partial w_{ij}^l} \\ \vdots \\ \dfrac{\partial C(\theta)}{\partial b_i^l} \\ \vdots \end{bmatrix} \qquad \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \\ \vdots & & \ddots \end{bmatrix} \qquad \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

Millions of parameters ……

To compute the gradients efficiently, we use ***backpropagation***.

Background

- Cost Function $C(\theta)$
  - Given training examples: $\{(x^1, \hat{y}^1), \cdots, (x^r, \hat{y}^r), \cdots, (x^R, \hat{y}^R)\}$
  - Find a set of parameters θ^* minimizing C(θ)
    - $C(\theta) = \frac{1}{R}\sum_r C^r(\theta), C^r(\theta) = \|f(x^r;\theta) - \hat{y}^r\|$
- Gradient Descent
  - $\nabla C(\theta) = \frac{1}{R}\sum_r \nabla C^r(\theta)$
  - Given w_{ij}^l and b_i^l, we have to compute $\partial C^r/\partial w_{ij}^l$ and $\partial C^r/\partial b_i^l$
- There is an efficient way to compute the gradients of the network parameters – ***backpropagation***.

Best Function = Best Parameters

$$y = f(x) = \sigma\left(W^L \ldots \sigma\left(W^2 \sigma\left(W^1 x + b^1\right) + b^1\right) \ldots + b^L\right)$$

function set

because different parameters W and b lead to different function

Formal way to define a function set:

$f(x;\theta)$ → parameter set

$$\theta = \left\{W^1, b^1, W^2, b^2 \cdots W^L, b^L\right\}$$

Pick the "best" function f*

Pick the "best" parameter set θ*

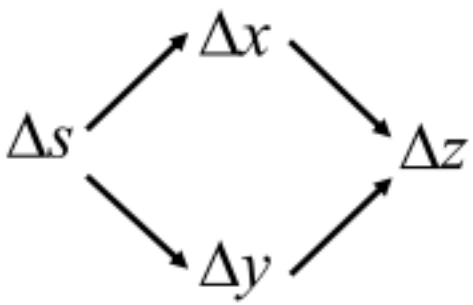
Chain Rule

Case 1 $y = g(x) \quad z = h(y)$

$$\Delta x \to \Delta y \to \Delta z \qquad \frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

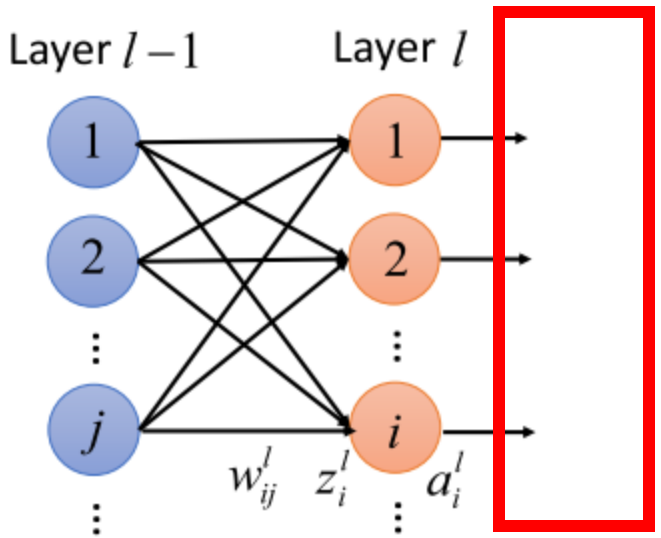
Case 2

$$x = g(s) \qquad y = h(s) \qquad z = k(x, y)$$



$$\frac{\partial z}{\partial s} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial s}$$

$\partial C^r / \partial w_{ij}^l$

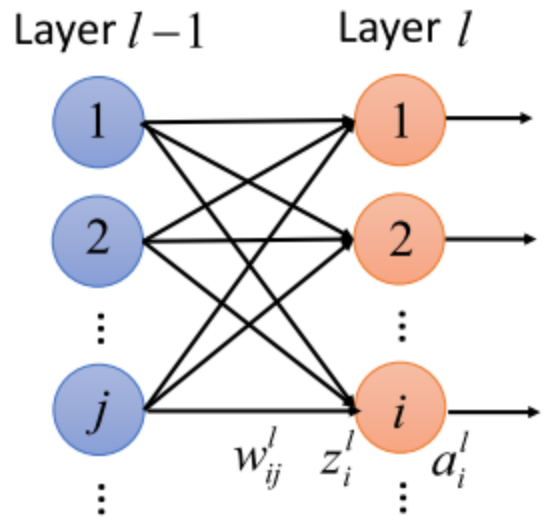


$$\Delta w_{ij}^l \rightarrow \Delta z_i^l \cdots\cdots \rightarrow \Delta C^r$$

$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \frac{\partial C^r}{\partial z_i^l}$$

- $\frac{\partial C^r}{\partial w_{ij}^l}$ is the multiplication of two terms

$\partial C^r / \partial w_{ij}^l$ - First Term



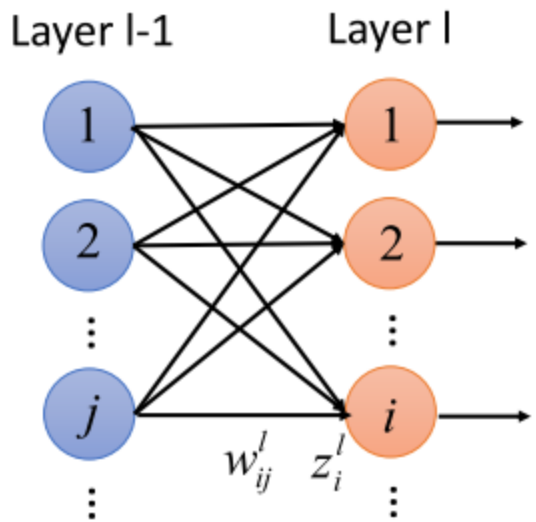
$$\Delta w_{ij}^l \rightarrow \Delta z_i^l \cdots\cdots \rightarrow \Delta C^r$$

$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \frac{\partial C^r}{\partial z_i^l}$$

- $\frac{\partial C^r}{\partial w_{ij}^l}$ is the multiplication of two terms

$\partial C^r / \partial w_{ij}^l$ - First Term

$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \frac{\partial C^r}{\partial z_i^l}$$

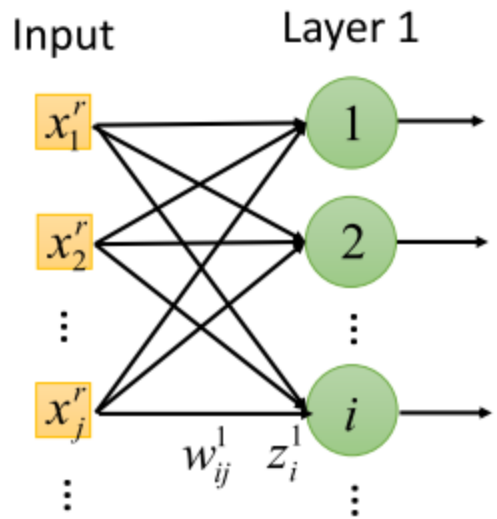


If l > 1

$$z_i^l = \sum_j w_{ij}^l a_j^{l-1} + b_i^l \qquad \frac{\partial z_i^l}{\partial w_{ij}^l} = a_j^{l-1}$$

$\partial C^r / \partial w_{ij}^l$ - First Term

$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \frac{\partial C^r}{\partial z_i^l}$$



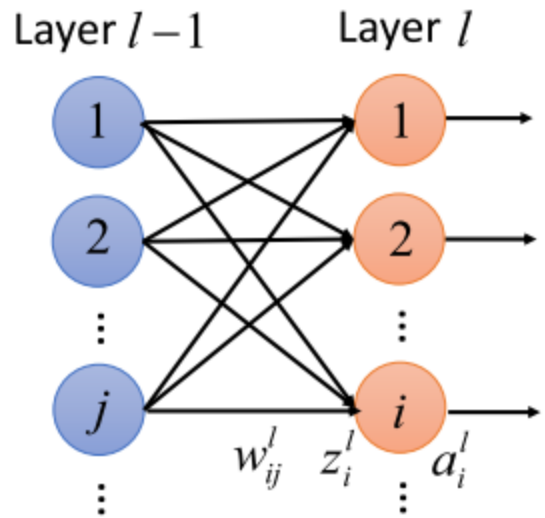
If l > 1

$$z_i^l = \sum_j w_{ij}^l a_j^{l-1} + b_i^l \qquad \frac{\partial z_i^l}{\partial w_{ij}^l} = a_j^{l-1}$$

If l = 1

$$z_i^1 = \sum_j w_{ij}^1 x_j^r + b_i^1 \qquad \frac{\partial z_i^1}{\partial w_{ij}^1} = x_j^r$$

$\partial C^r / \partial w_{ij}^l$ - Second Term



$$\Delta w_{ij}^l \rightarrow \Delta z_i^l \cdots\cdots \rightarrow \Delta C^r$$

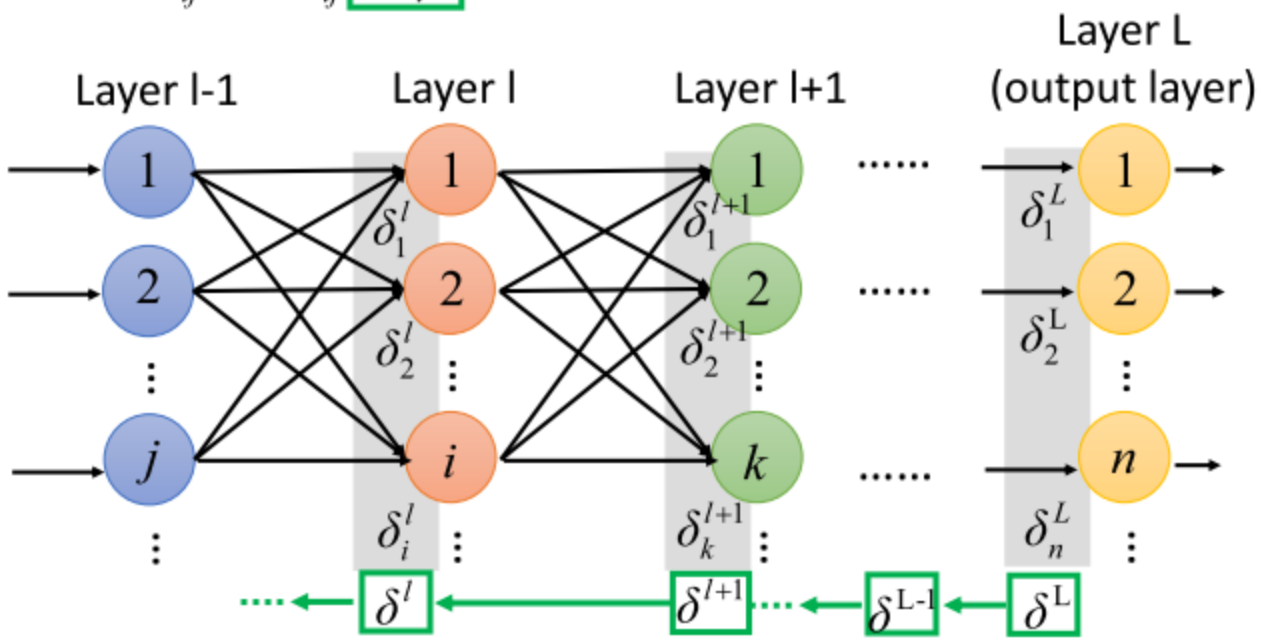
$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \boxed{\frac{\partial C^r}{\partial z_i^l}} \rightarrow \delta_i^l$$

- $\dfrac{\partial C^r}{\partial w_{ij}^l}$ is the multiplication of two terms

$\partial C^r / \partial w_{ij}^l$ - Second Term

$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \boxed{\frac{\partial C^r}{\partial z_i^l}} \rightarrow \delta_i^l$$

1. How to compute δ^L
2. The relation of δ^l and δ^{l+1}



$\partial C^r / \partial w_{ij}^l$ - Second Term

$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \frac{\partial C^r}{\partial z_i^l} \rightarrow \delta_i^l$$

1. How to compute δ^L

2. The relation of δ^l and δ^{l+1}

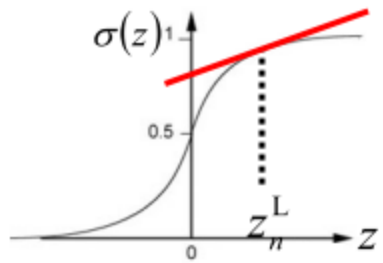
$$\delta_n^{\mathrm{L}} = \frac{\partial C^r}{\partial z_n^{\mathrm{L}}}$$

$$\Delta z_n^{\mathrm{L}} \rightarrow \Delta a_n^{\mathrm{L}} = \Delta y_n^{\mathrm{r}} \rightarrow \Delta C^{\mathrm{r}}$$

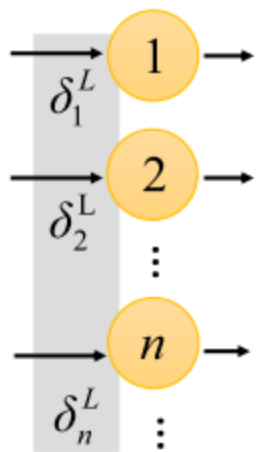
$$= \frac{\partial y_n^{\mathrm{r}}}{\partial z_n^{\mathrm{L}}} \frac{\partial C^r}{\partial y_n^{\mathrm{r}}}$$

Depending on the definition of cost function

$$\sigma'\left(z_n^{\mathrm{L}}\right)$$



Layer L
(output layer)



$\partial C^r / \partial w_{ij}^l$ - Second Term

$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \frac{\partial C^r}{\partial z_i^l} \rightarrow \delta_i^l$$

1. How to compute δ^L

2. The relation of δ^l and δ^{l+1}

δ^L?

$$\delta_n^L = \frac{\partial C^r}{\partial z_n^L}$$

$$= \frac{\partial y_n^r}{\partial z_n^L} \frac{\partial C^r}{\partial y_n^r}$$

$$= \sigma'\left(z_n^L\right) \frac{\partial C^r}{\partial y_n^r}$$

$$\sigma'\left(z^L\right) = \begin{bmatrix} \sigma'\left(z_1^L\right) \\ \sigma'\left(z_2^L\right) \\ \vdots \\ \sigma'\left(z_n^L\right) \\ \vdots \end{bmatrix} \quad \nabla C^r\left(y^r\right) = \begin{bmatrix} \partial C^r / \partial y_1^r \\ \partial C^r / \partial y_2^r \\ \vdots \\ \partial C^r / \partial y_n^r \\ \vdots \end{bmatrix}$$

$$\delta^L = \sigma'\left(z^l\right) \bullet \nabla C^r\left(y^r\right)$$

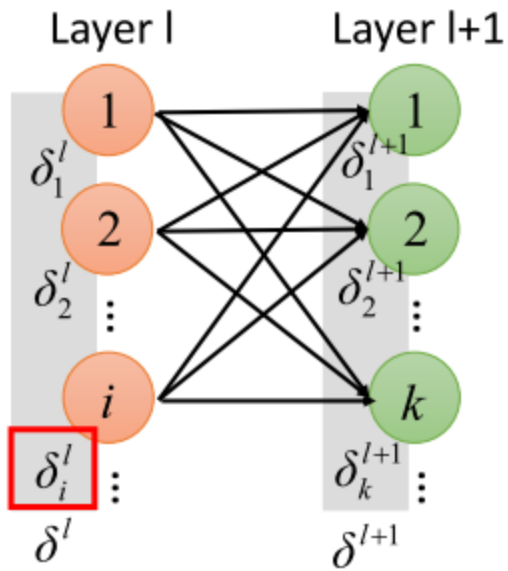
element-wise multiplication

$\partial C^r / \partial w_{ij}^l$ - Second Term

$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \boxed{\frac{\partial C^r}{\partial z_i^l}} \rightarrow \delta_i^l$$

1. How to compute δ^L

2. The relation of δ^l and δ^{l+1}



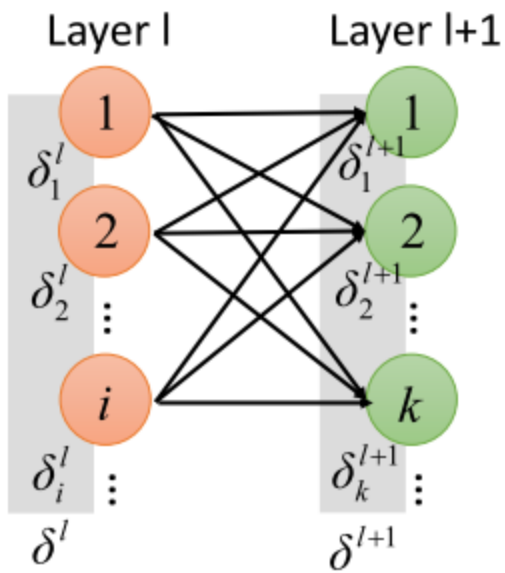
$$\delta_i^l = \frac{\partial C^r}{\partial z_i^l}$$

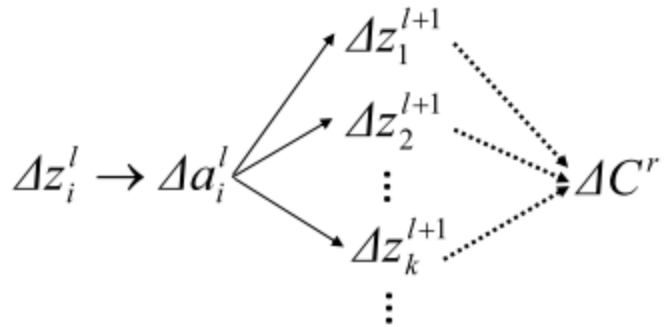
$$\Delta z_i^l \rightarrow \Delta a_i^l \rightarrow \Delta z_1^{l+1}, \Delta z_2^{l+1}, \ldots, \Delta z_k^{l+1}, \ldots \rightarrow \Delta C^r$$

$$\delta_i^l = \frac{\partial C^r}{\partial z_i^l} = \frac{\partial a_i^l}{\partial z_i^l} \sum_k \frac{\partial z_k^{l+1}}{\partial a_i^l} \boxed{\frac{\partial C^r}{\partial z_k^{l+1}}} \rightarrow \delta_k^{l+1}$$

$\partial C^r / \partial w_{ij}^l$ - Second Term

$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \boxed{\frac{\partial C^r}{\partial z_i^l}} \rightarrow \delta_i^l$$





$$\delta_i^l = \frac{\partial a_i^l}{\partial z_i^l} \sum_k \frac{\partial z_k^{l+1}}{\partial a_i^l} \delta_k^{l+1}$$

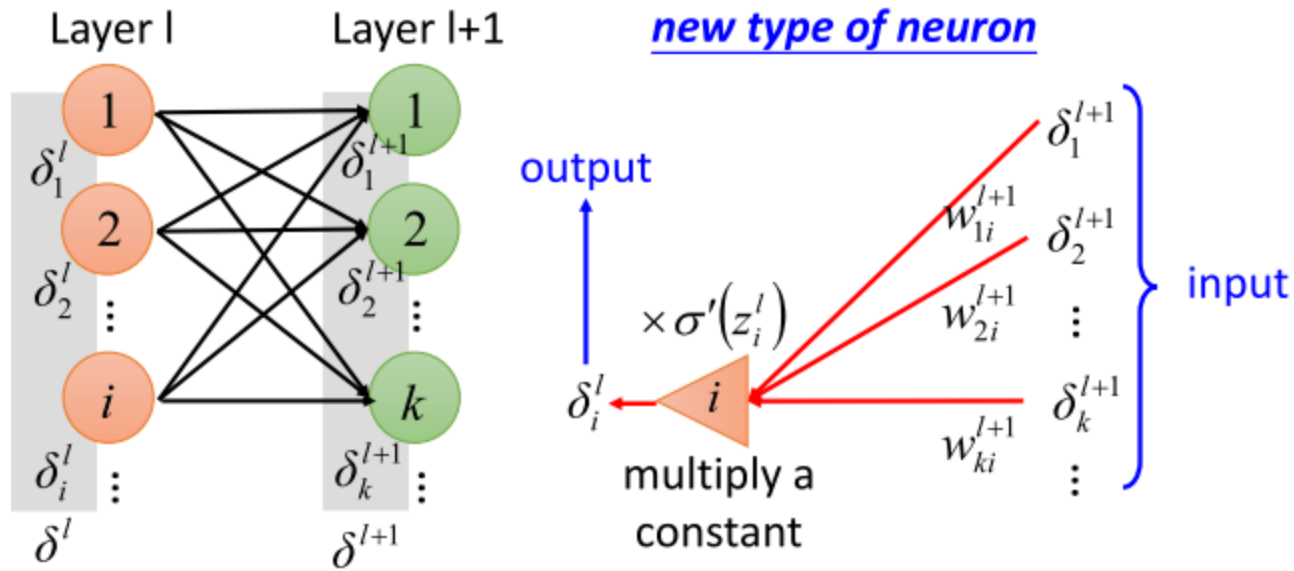
$$\sigma'\left(z_i^l\right) \qquad z_k^{l+1} = \sum_i w_{ki}^{l+1} a_i^l + b_k^{l+1}$$

$$\delta_i^l = \sigma'\left(z_i^l\right) \sum_k w_{ki}^{l+1} \delta_k^{l+1}$$

$\partial C^r / \partial w_{ij}^l$ - Second Term

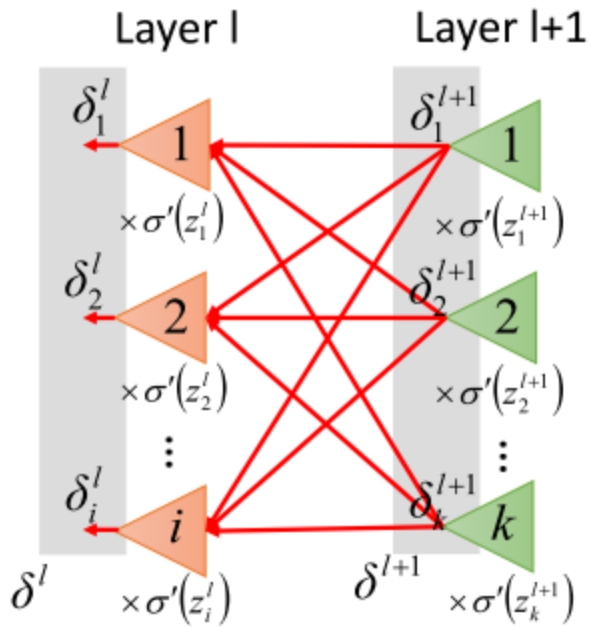
$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \boxed{\frac{\partial C^r}{\partial z_i^l}} \rightarrow \delta_i^l$$

$$\delta_i^l = \sigma'\left(z_i^l\right) \sum_k w_{ki}^{l+1} \delta_k^{l+1}$$



new type of neuron

$\partial C^r / \partial w_{ij}^l$ - Second Term

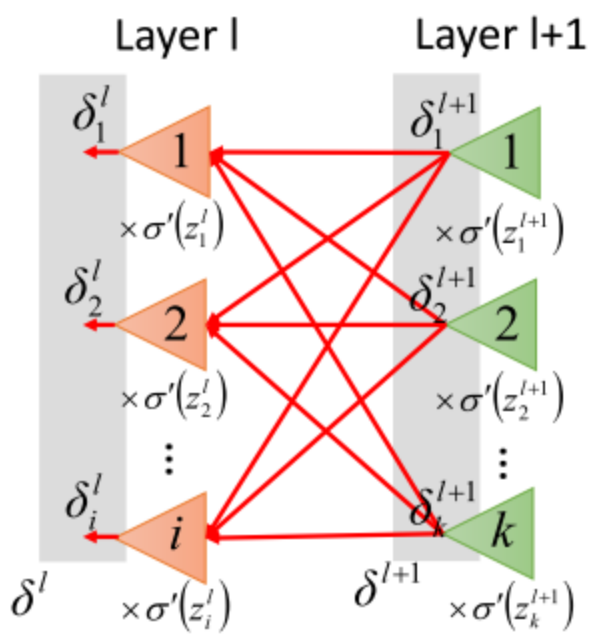


$$\delta_i^l = \sigma'\left(z_i^l\right)\sum_k w_{ki}^{l+1}\delta_k^{l+1}$$

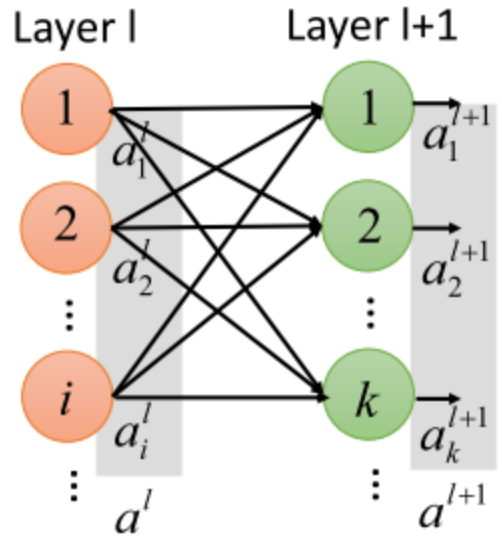
$$\sigma'\left(z^l\right) = \begin{bmatrix} \sigma'\left(z_1^l\right) \\ \sigma'\left(z_2^l\right) \\ \vdots \\ \sigma'\left(z_i^l\right) \\ \vdots \end{bmatrix}$$

$$\delta^l = \sigma'\left(z^l\right) \bullet \left(W^{l+1}\right)^T \delta^{l+1}$$

$\partial C^r / \partial w_{ij}^l$ - Second Term



Compare



$$\delta^l = \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1}$$

$$a^{l+1} = \sigma(W^{l+1} a^l + b^{l+1})$$

$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \boxed{\frac{\partial C^r}{\partial z_i^l}}$$

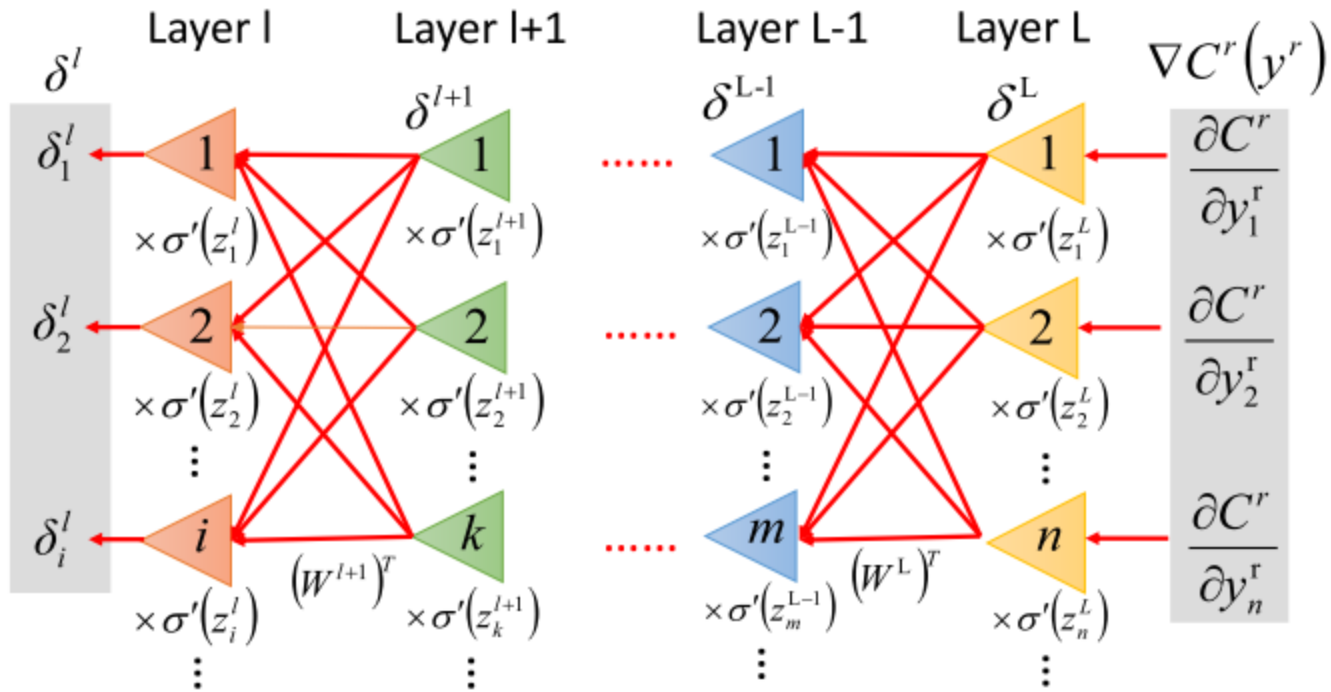
$$\boxed{\frac{\partial C^r}{\partial z_i^l}} \rightarrow \delta_i^l$$

1. How to compute δ^L

$$\Rightarrow \delta^{\mathrm{L}} = \sigma'\left(z^{\mathrm{L}}\right) \bullet \nabla C^r\left(y^r\right)$$

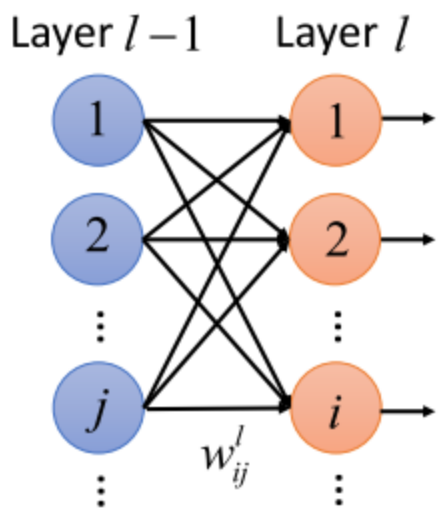
2. The relation of δ^l and δ^{l+1}

$$\Rightarrow \delta^l = \sigma'\left(z^l\right) \bullet \left(W^{l+1}\right)^T \delta^{l+1}$$



Concluding Remarks

$$\frac{\partial C^r}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \frac{\partial C^r}{\partial z_i^l}$$



$$\begin{cases} a_j^{l-1} & l>1 \\ x_j^r & l=1 \end{cases}$$

$$\delta_i^l$$

Forward Pass

$$z^1 = W^1 x^r + b^1$$

$$a^1 = \sigma(z^1)$$

$$\cdots\cdots$$

$$z^{l-1} = W^{l-1} a^{l-2} + b^{l-1}$$

$$a^{l-1} = \sigma(z^{l-1})$$

Backward Pass

$$\delta^L = \sigma'(z^L) \bullet \nabla C^r(y^r)$$

$$\delta^{L-1} = \sigma'(z^{L-1}) \bullet (W^L)^T \delta^L$$

$$\cdots\cdots$$

$$\delta^l = \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1}$$

$$\cdots\cdots$$

Stochastic Gradient Descent and Mini-batch

$$C(\theta)=\frac{1}{R}\sum_{r}\left\|f\left(x^{r};\theta\right)-\hat{y}^{r}\right\|$$

$$=\frac{1}{R}\sum_{r}\mathrm{C}^{r}\left(\theta\right)$$

◆***Gradient Descent***

$$\theta^{i}=\theta^{i-1}-\eta\nabla C\left(\theta^{i-1}\right)$$

$$\nabla C\left(\theta^{i-1}\right)=\frac{1}{R}\sum_{r}\nabla C^{r}\left(\theta^{i-1}\right)$$

◆***Stochastic Gradient Descent***

Faster!

Better!

Pick an example x^r

$$\theta^{i}=\theta^{i-1}-\eta\nabla C^{r}\left(\theta^{i-1}\right)$$

If all example x^r have equal probabilities to be picked

$$E\left[\nabla C^{r}\left(\theta^{i-1}\right)\right]=\frac{1}{R}\sum_{r}\nabla C^{r}\left(\theta^{i-1}\right)$$

Stochastic Gradient Descent and Mini-batch

What is epoch?

Training Data: $\left\{\left(x^1, \hat{y}^1\right), \left(x^2, \hat{y}^2\right), \cdots \left(x^r, \hat{y}^r\right), \cdots \left(x^R, \hat{y}^R\right)\right\}$

When using stochastic gradient descent

Starting at θ_0

pick x^1 $\quad \theta^1 = \theta^0 - \eta \nabla C^1\left(\theta^0\right)$

pick x^2 $\quad \theta^2 = \theta^1 - \eta \nabla C^2\left(\theta^1\right)$

$\vdots$ $\qquad\qquad \vdots$

pick x^r $\quad \theta^r = \theta^{r-1} - \eta \nabla C^r\left(\theta^{r-1}\right)$

$\vdots$ $\qquad\qquad \vdots$

pick x^R $\quad \theta^R = \theta^{R-1} - \eta \nabla C^R\left(\theta^{R-1}\right)$

Seen all the examples once

One epoch

---

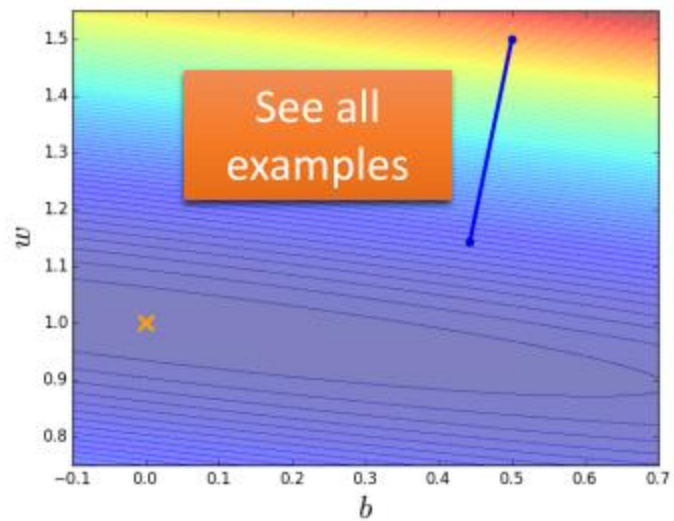
pick x^1 $\quad \theta^{R+1} = \theta^R - \eta \nabla C^1\left(\theta^R\right)$

Stochastic Gradient Descent and Mini-batch

- ***Toy Example***

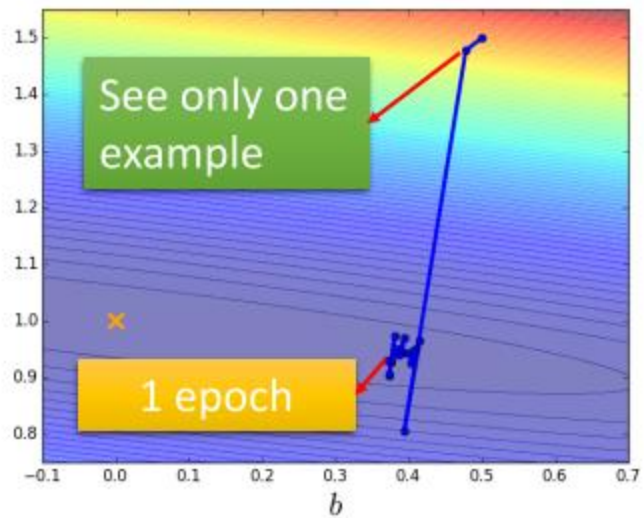
Gradient Descent

Update after seeing all examples



Stochastic Gradient Descent

If there are 20 examples, update 20 times in one epoch.



Stochastic Gradient Descent and Mini-batch

◆ ***Gradient Descent***

$$\theta^i = \theta^{i-1} - \eta \nabla C\left(\theta^{i-1}\right) \qquad \nabla C\left(\theta^{i-1}\right) = \frac{1}{R}\sum_r \nabla C^r\left(\theta^{i-1}\right)$$

◆ ***Stochastic Gradient Descent***

Pick an example x_r $\qquad \theta^i = \theta^{i-1} - \eta \nabla C^r\left(\theta^{i-1}\right)$

◆ ***Mini Batch Gradient Descent***

Pick B examples as a batch b

B is batch size

Shuffle your data

$$\theta^i = \theta^{i-1} - \eta \frac{1}{B}\sum_{x_r \in b} \nabla C^r\left(\theta^{i-1}\right)$$

Average the gradient of the examples in the batch b

Stochastic Gradient Descent and Mini-batch

- ***Handwriting Digit Classification***

Gradient Descent


Questions?