深度学习-背景与简介

史春奇

2017 年

目录

第一部分

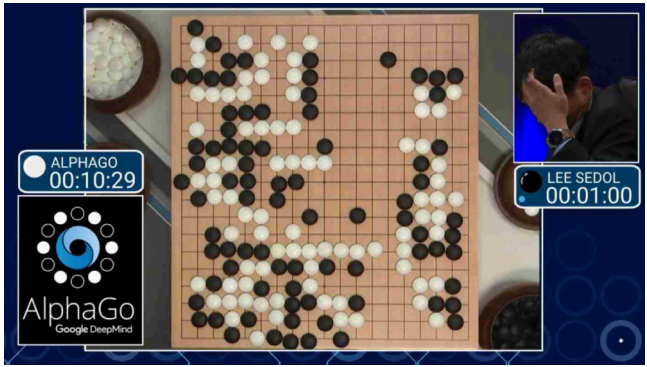
- 深度学习的由来
  - 谁发明了神经网络？
  - 谁创造了深度学习？
  - 深度学习什么时候成功的？为什么以前没有成功？
- 反向传播算法的理解
  - 哪些参数会影响反向 BP 算法？
- TensorFlow 应用
  - 如何快速入门 TensorFlow ？

1 深度学习引爆

1.1 谷歌 Alpha GO 2.0 – 围棋

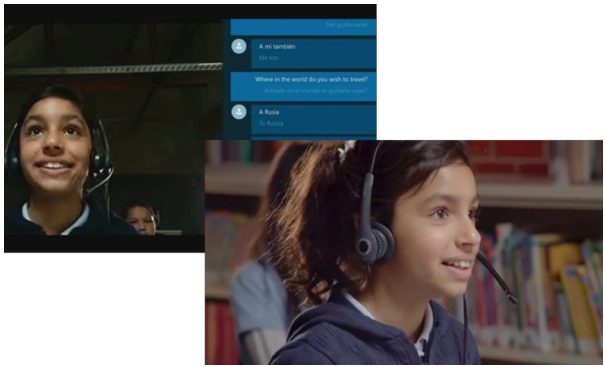
2017 年 5 月 23 日到 27 日，在中国乌镇围棋峰会上，阿尔法围棋以 3 比 0 的总比分战胜排名世界第一的世界围棋冠军柯洁。

1.2 谷歌 Alpha GO – 围棋



2016 年 3 月，阿尔法围棋与围棋世界冠军、职业九段棋手李世石进行围棋人机大战，以 4 比 1 的总比分获胜；

1.3 微软 Skype – 语音翻译



2014 年 12 月，Skype 上线语音到语音的机器翻译，可以实时不同语言对话。

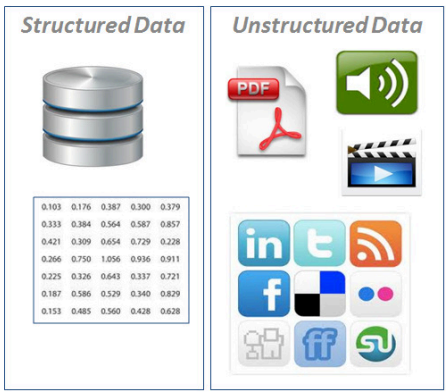
1.4 Facebook DeepFace – 人脸识别

2014 年 3 月, Facebook 人脸识别系统 DeepFace 达到 97.35% 准确率，超过人的识别准确率。

1.5 典型深度学习神经网络

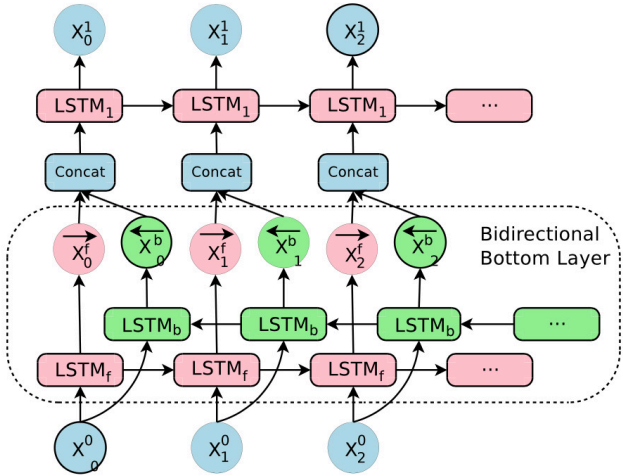
1.5.1 非结构数据

- 文本、图像视频、音频: 海量非结构化数据!



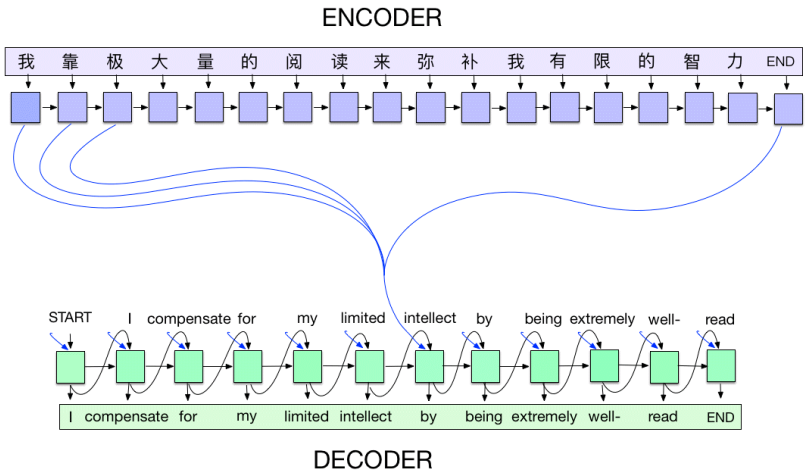
1.5.2 自然语言处理

- 递归神经网络 RNN



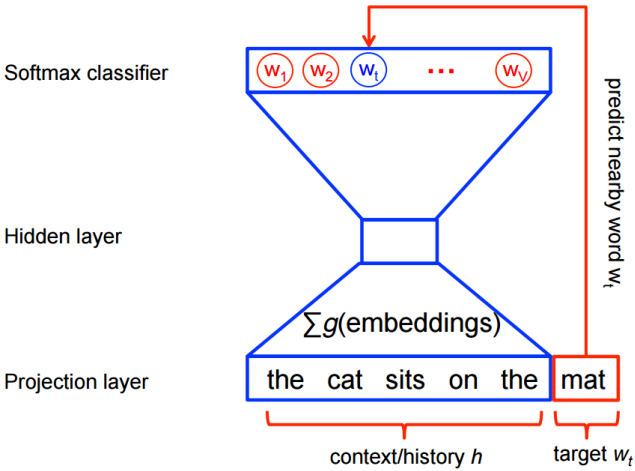
Recurrent Neural Network!

- 编码器解码器 Encoder-Decoder

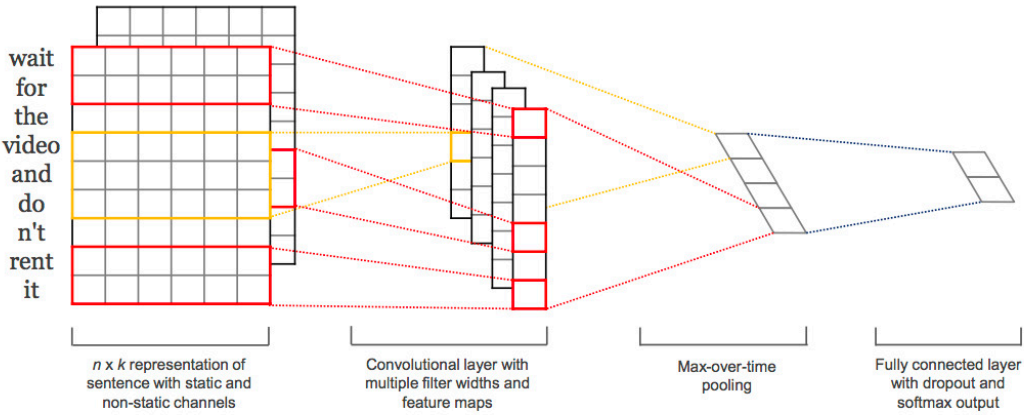


Encoder-Decoder！

- 词嵌入 Word Embedding



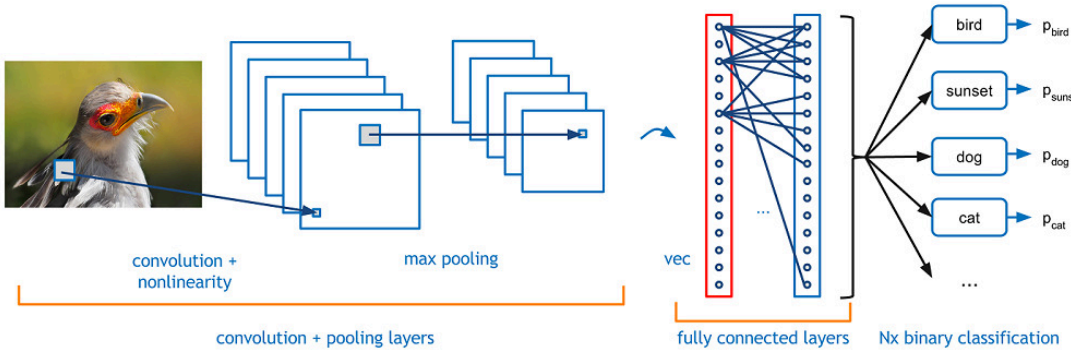
- 卷积神经网络 CNN



Convolutional Neural Network!

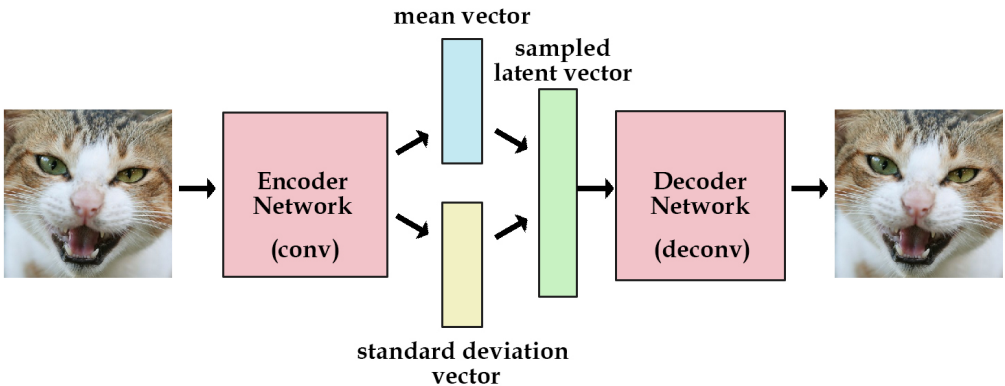
1.5.3 图像视频分类、识别、分割

- 卷积神经网络 CNN



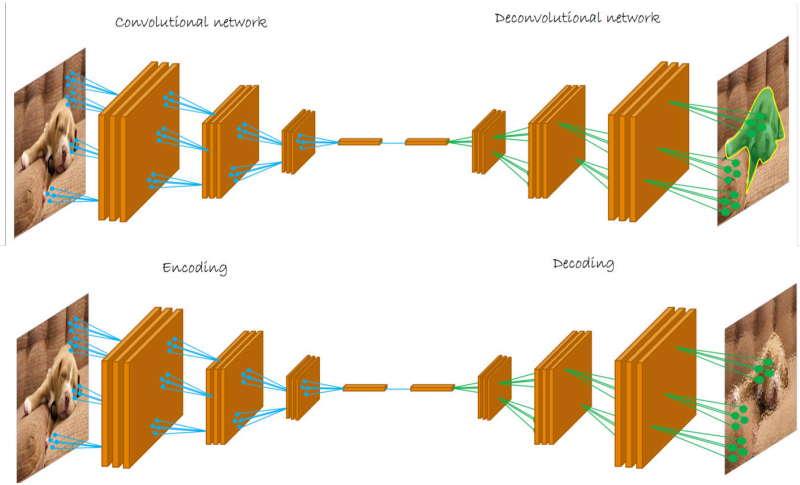
Convolutional Neural Network!

- 变分自动编码器 VAE

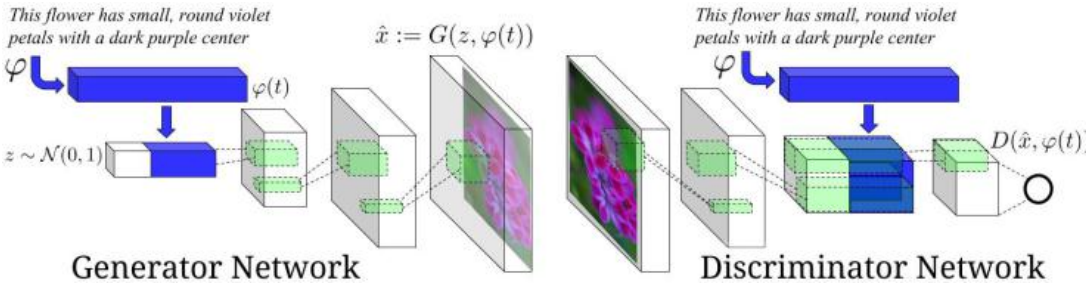


Variational AutoEncoder!

- AE 和 Deconv 网络对比

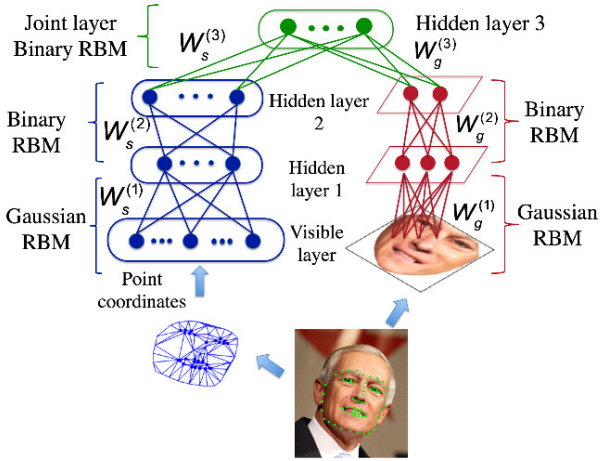


- 生成对抗网络 GAN



Generative Adversarial Network!

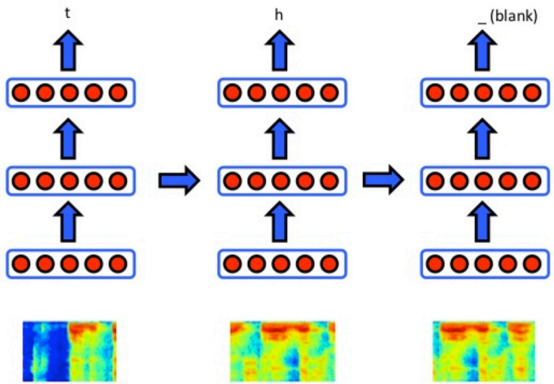
- 深度玻尔兹曼机 DBM



Deep Boltzmann Machine!

1.5.4 语音识别

- 递归神经网络 RNN (Recurrent Neural Network)



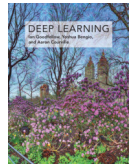
1.5.5 其他领域

- 机器人
- 自动驾驶
- 音乐
- 游戏
- 艺术

2 深度学习简史

- 《深度学习》中定义

  针对这些比较直观的问题，本书讨论一种解决方案。该方案可以让计算机从经验中学习，并根据层次化的概念体系来理解世界，而每个概念则通过与某些相对简单的概念之间的关系来定义。让计算机从经验获取知识，可以避免由人类来给计算机形式化地指定它需要的所有知识。层次化的概念让计算机构建较简单的概念来学习复杂概念。如果绘制出这些概念如何建立在彼此之上的图，我们将得到一张“深”（层次很多）的图。基于这个原因，我们称这种方法为 AI **深度学习**（deep learning）。

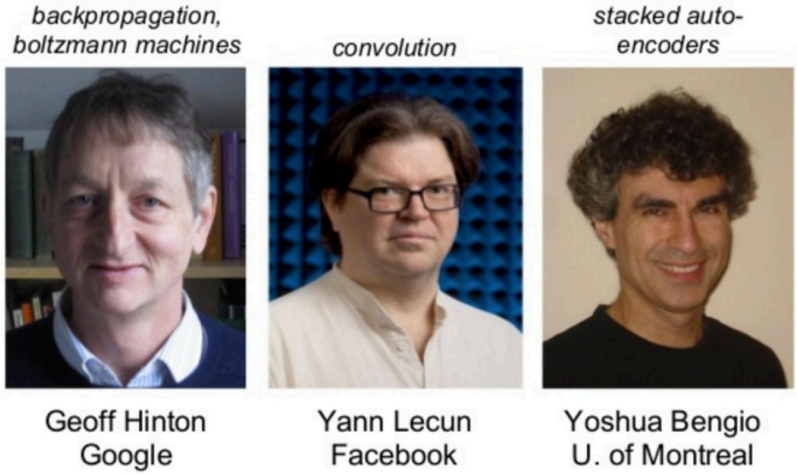


- Nature 杂志中定义

  Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification. Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the

2.1 代表人物：三巨头

- 辛顿 Hinton、杨乐昆 Lecun、本吉奥 Bengio



- 辛顿 Hinton

- 吴恩达 Ng

- 施米德胡贝 Schmidhuber

Geoffrey Hinton

Emeritus Professor of Computer Science, University of Toronto
Engineering Fellow, Google Inc.
machine learning, neural networks, artificial intelligence, cognitive science, computer science
Verified email at cs.toronto.edu - Homepage

Citation indices	All	Since 2012
Citations	182775	92246
h-index	132	97
i10-index	311	225

Yann LeCun

Director of AI Research at Facebook & Silver Professor at the Courant Institute, New York University
AI, machine learning, computer vision, robotics, image compression
Verified email at cs.nyu.edu - Homepage

Citation indices	All	Since 2012
Citations	56086	39890
h-index	96	78
i10-index	218	179

Yoshua Bengio

Professor, U. Montreal (Computer Sc. & Op. Res.), MILA, CIFAR, CRM, REPARTI, GRSNC
Machine learning, deep learning, artificial intelligence
Verified email at umontreal.ca - Homepage

Citation indices	All	Since 2012
Citations	79304	68064
h-index	101	92
i10-index	338	284

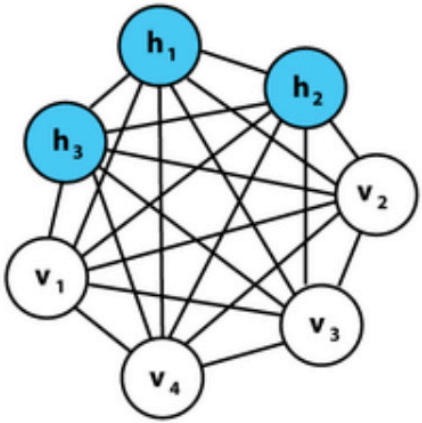
Juergen Schmidhuber

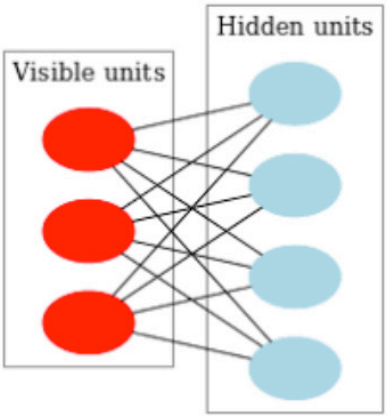
The Swiss AI Lab IDSIA / USI & SUPSI
Verified email at idsia.ch - Homepage

Citation indices	All	Since 2012
Citations	31037	23463
h-index	77	58
i10-index	294	207

这些巨擘之间什么关系？有哪些重要贡献？

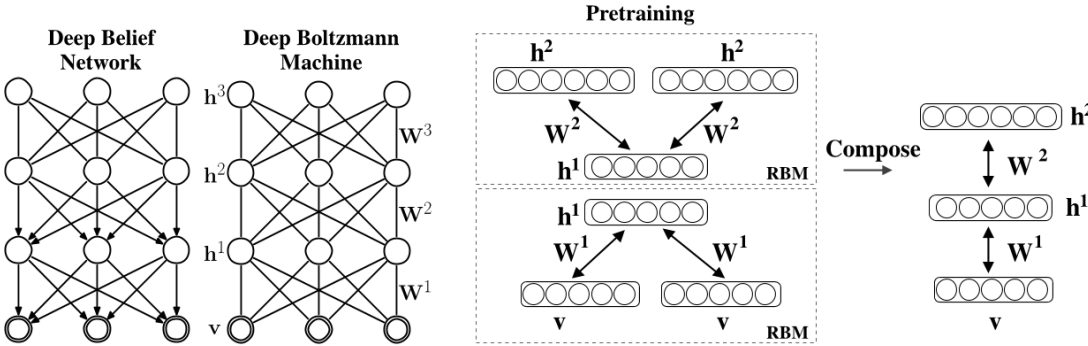
- Hinton 的玻尔兹曼机 BM





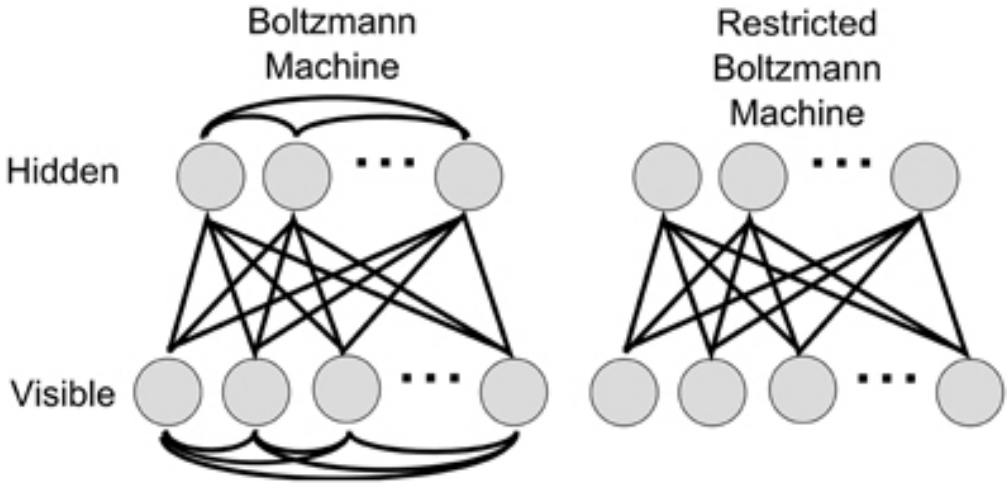
基于 BM 的 DBM 开启了深度学习！

- Hinton 的深度玻尔兹曼机 DBM



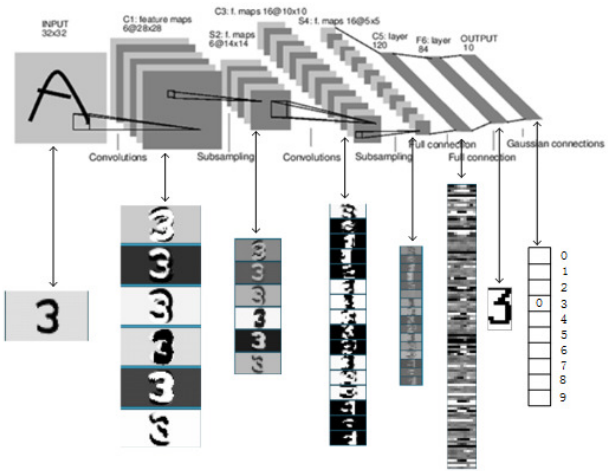
DBM 是受限玻尔兹曼机 RBM 的交互叠加！

- 受限玻尔兹曼机 RBM



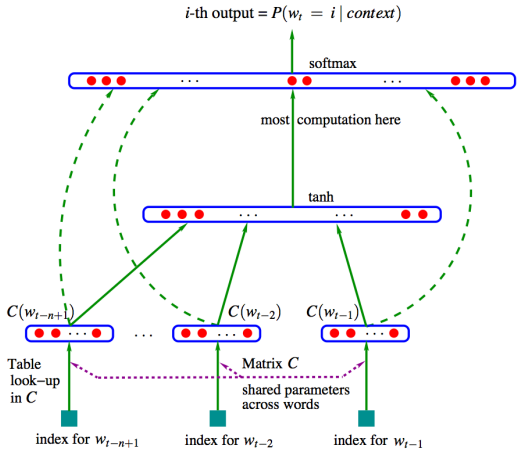
DBM 是受限玻尔兹曼机 RBM 的交互叠加！

- LeCun 的卷积神经网络 LeNet



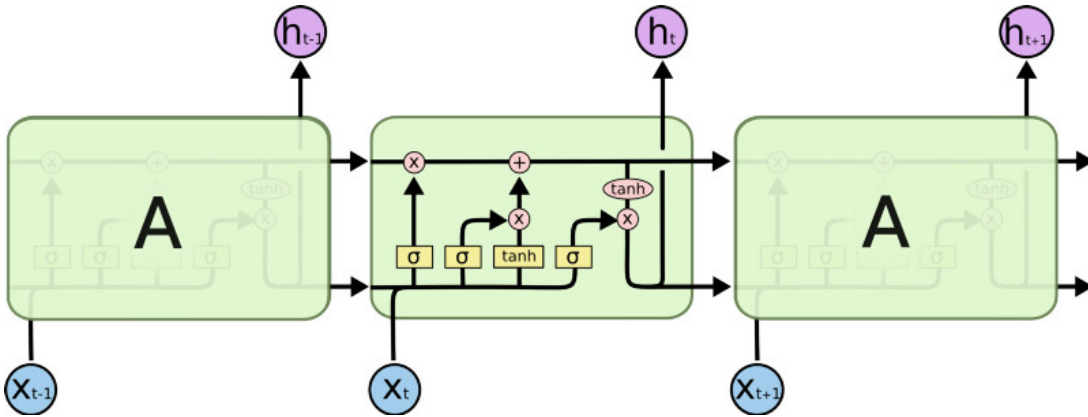
5 层的卷积神经网络：5-LeNet！

- Bengio 的神经网络语言模型 NNLM



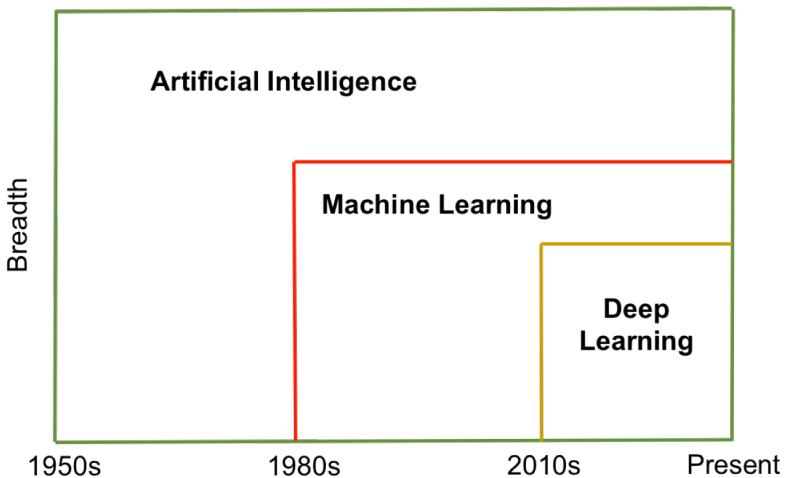
Neural Network Language Model!

- Schmidhuber 的递归神经网络 LSTM



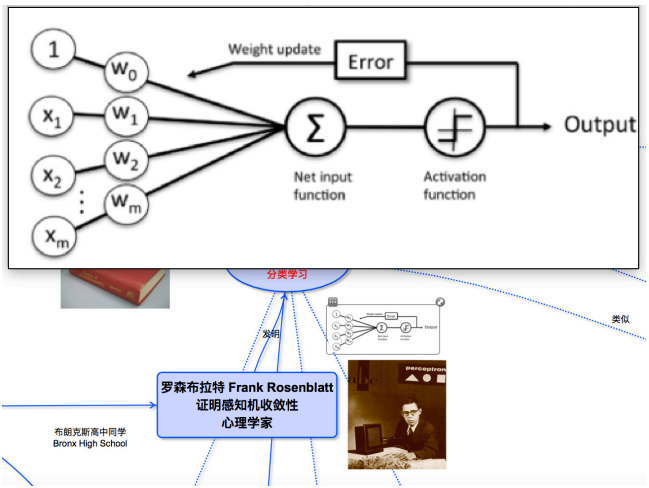
长短期记忆网络 Long Short-Term Memory！

2.2 人工智能到深度学习



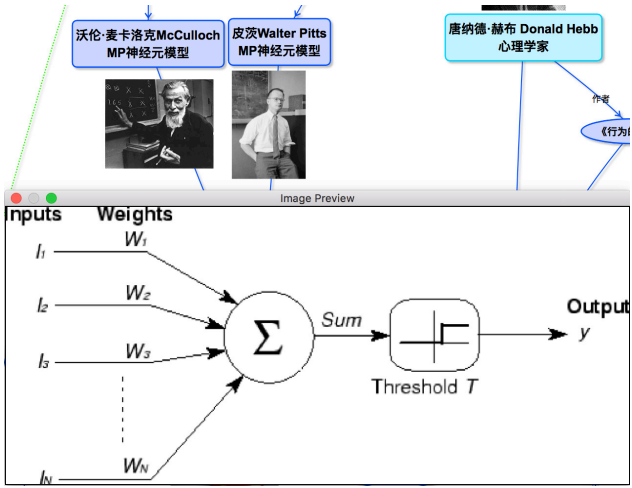
到底是什么点燃了深度学习的火种？首先回顾一下神经网络的发展史

2.2.1 感知机诞生



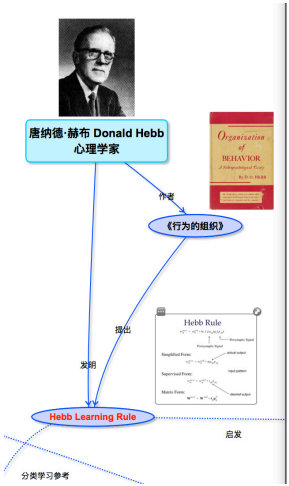
最简单的神经网络模型，单个神经元

- MP 模型



MP 模型是一个静态架构

- Hebb 学习



Hebb Rule

$$w_{ij}^{new} = w_{ij}^{old} + \alpha\, f_i(a_{iq}) g_j(p_{jq})$$

Postsynaptic Signal

Presynaptic Signal

Simplified Form:

actual output

$$w_{ij}^{new} = w_{ij}^{old} + \alpha a_{iq} p_{jq}$$

Supervised Form:

input pattern

$$w_{ij}^{new} = w_{ij}^{old} + t_{iq} p_{jq}$$

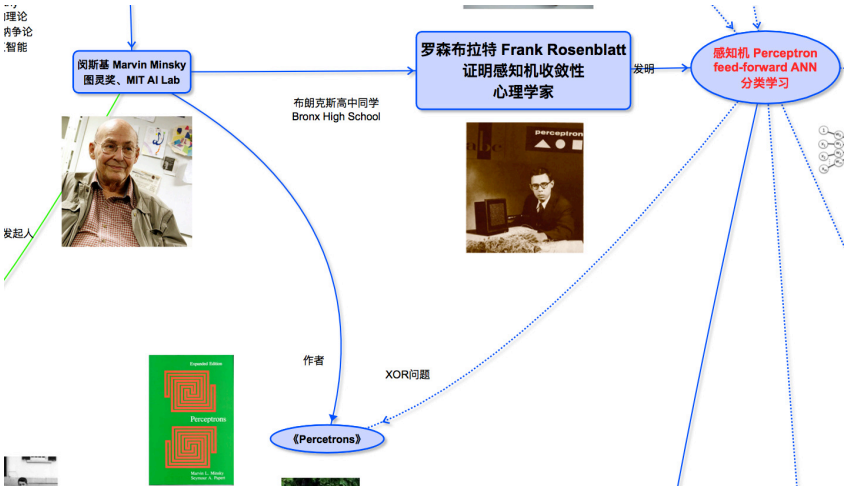
Matrix Form:

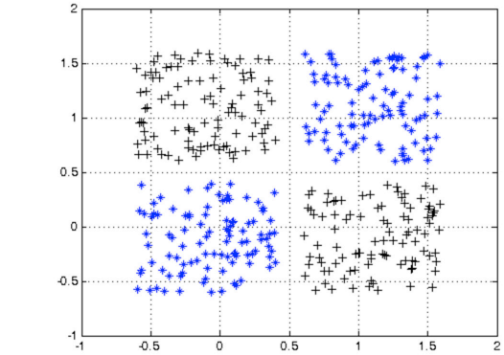
desired output

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q \mathbf{p}_q^T$$

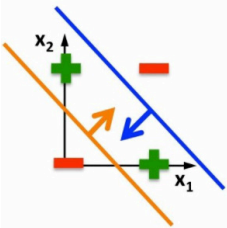
带来了动态学习能力

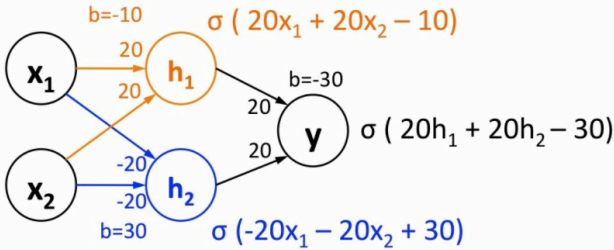
- XOR 问题


理论
争论
智能
发起人
闵斯基 Marvin Minsky
图灵奖、MIT AI Lab
布朗克斯高中同学
Bronx High School
罗森布拉特 Frank Rosenblatt
证明感知机收敛性
心理学家
发明
感知机 Perceptron
feed-forward ANN
分类学习
perceptron
作者
XOR问题
《Percetrons》
Perceptrons


A
B
0,1
1,1
0,0
1,0
Input A
Input B
A and B
A or B
A xor B

Linear classifiers
cannot solve this



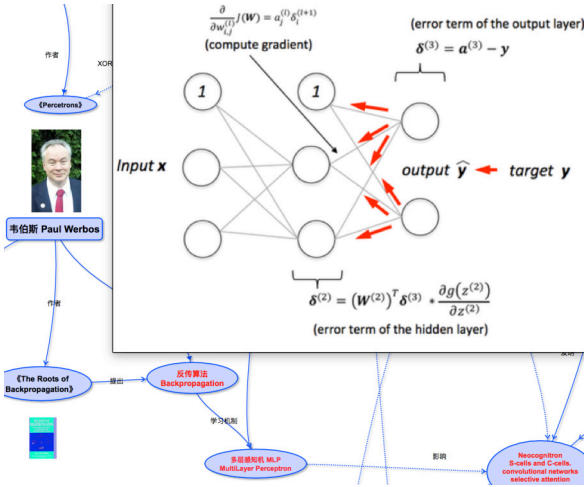


σ(20*0 + 20*0 – 10) ≈ 0
σ(20*1 + 20*1 – 10) ≈ 1
σ(20*0 + 20*1 – 10) ≈ 1
σ(20*1 + 20*0 – 10) ≈ 1

σ (-20*0 – 20*0 + 30) ≈ 1
σ (-20*1 – 20*1 + 30) ≈ 0
σ (-20*0 – 20*1 + 30) ≈ 1
σ (-20*1 – 20*0 + 30) ≈ 1

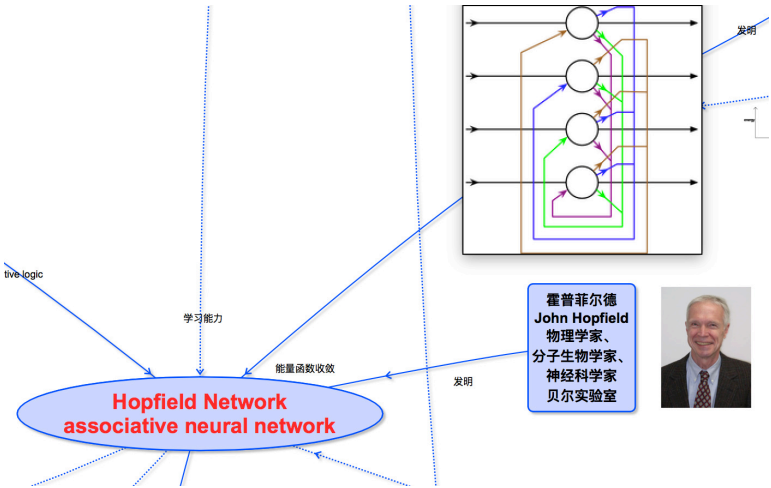
σ (20*0 + 20*1 – 30) ≈ 0
σ (20*1 + 20*0 – 30) ≈ 0
σ (20*1 + 20*1 – 30) ≈ 1
σ (20*1 + 20*1 – 30) ≈ 1

- 反向传播 BP 算法



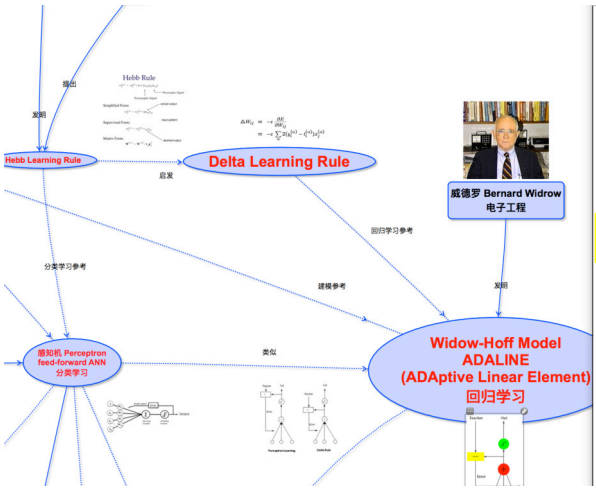
但是影响力不大！

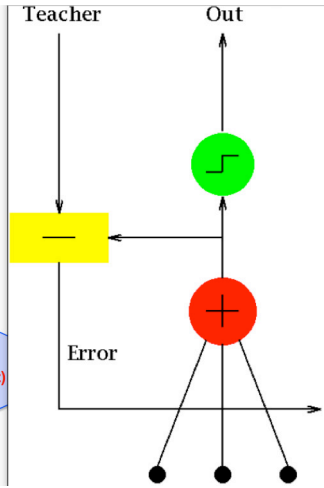
- 霍普菲尔德 Hopfield 网络



神经网络回归！

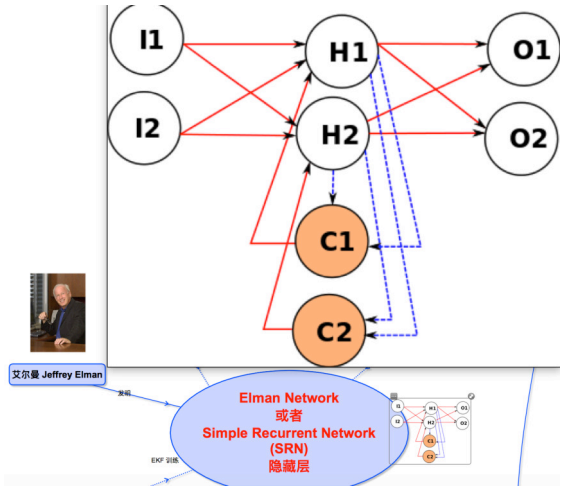
- 为什么 Hopfield 网络能存活？ADALINE 的成功





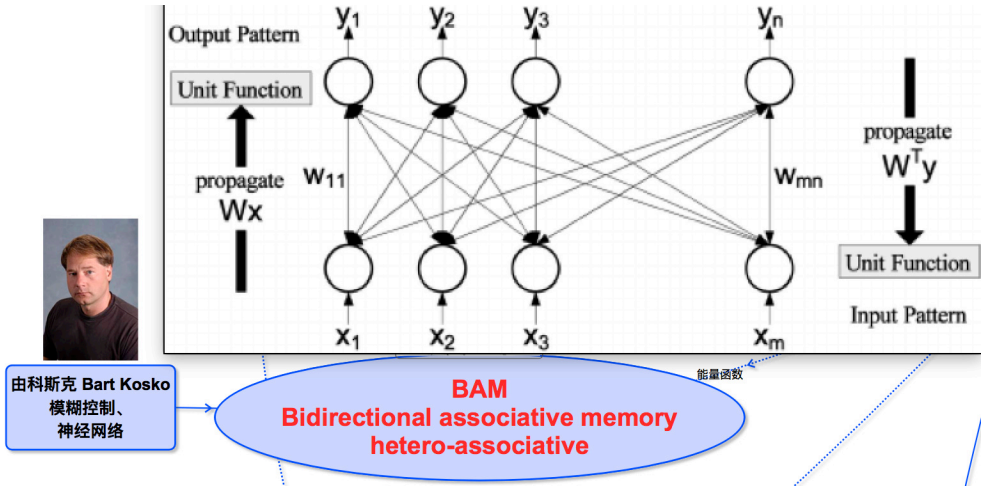
基于 Delta 学习的 ADALINE 在物理领域相当成功！

- Hopfield 网络影响力之一：SRN 网络



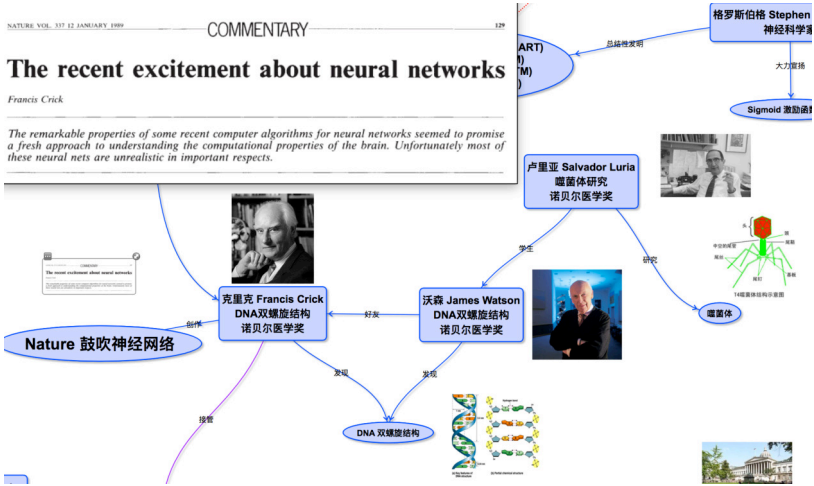
递归神经网络 RNN 的前生！

- Hopfield 网络影响力之二：BAM 网络



玻尔兹曼机 BM 的前生！

- Hopfield 网络影响力之三：克里克 Crick 研究小组

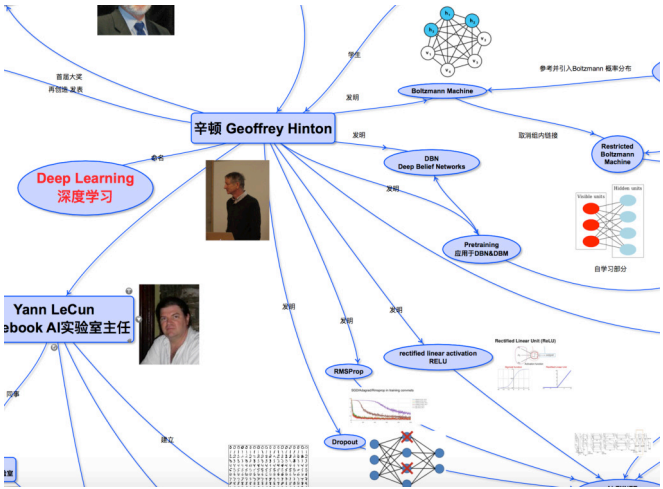


DNA 双螺旋结构的发明人 Crick！在自然杂志鼓吹神经网络！

- Crick 研究小组成员：鲁梅尔哈特 Rumelhart

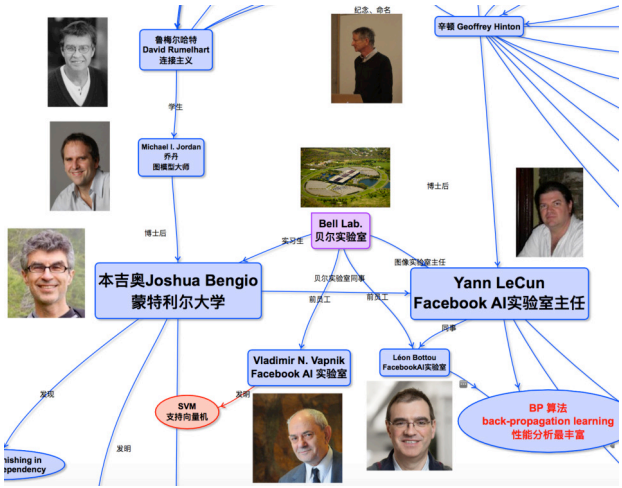
BP 算法再造和推广；Bengio 和 Ng 的师祖；连接主义的核心！

- Crick 研究小组成员：辛顿 Hinton



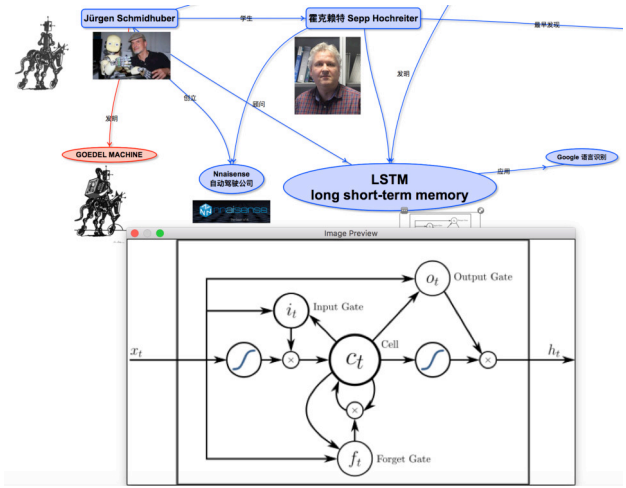
BP 算法再造和推广；LeCun 的导师；深度学习的核心！

- 深度学习主力：LeCun 和 Bengio



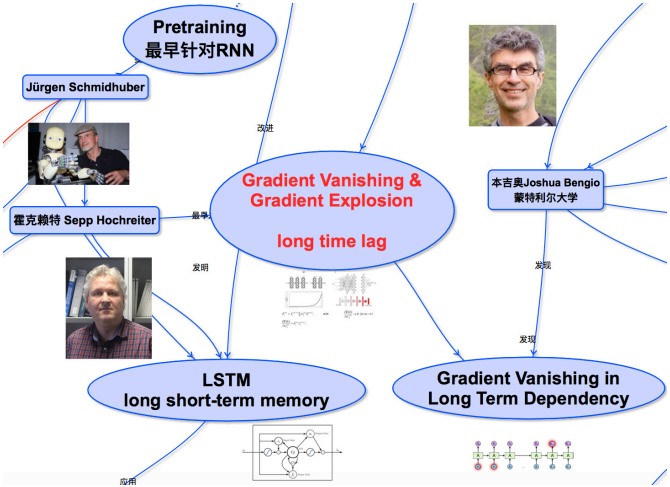
连接主义的追随、坚持者；贝尔实验室同事；

- 施米德胡贝 Schmidhuber 改造 SRN 网络发明 LSTM



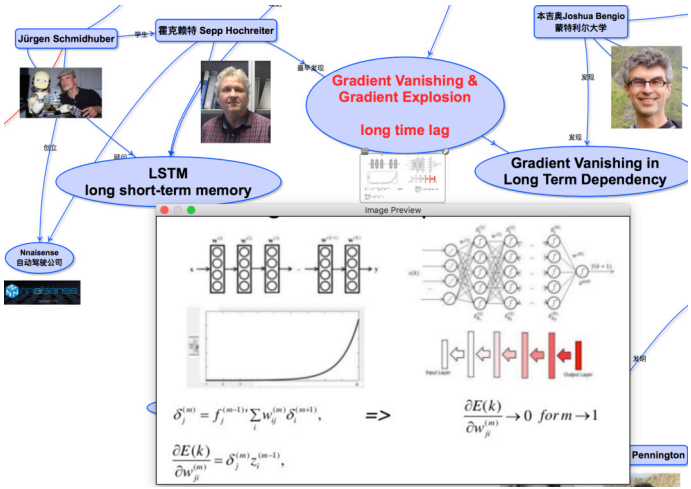
梯度消失爆炸问题 Vanishing/Exploding Gradient！发现 Pretraining！

- 梯度消失爆炸问题和长期依赖



Bengio 把梯度消失问题宣传到连接主义圈子！

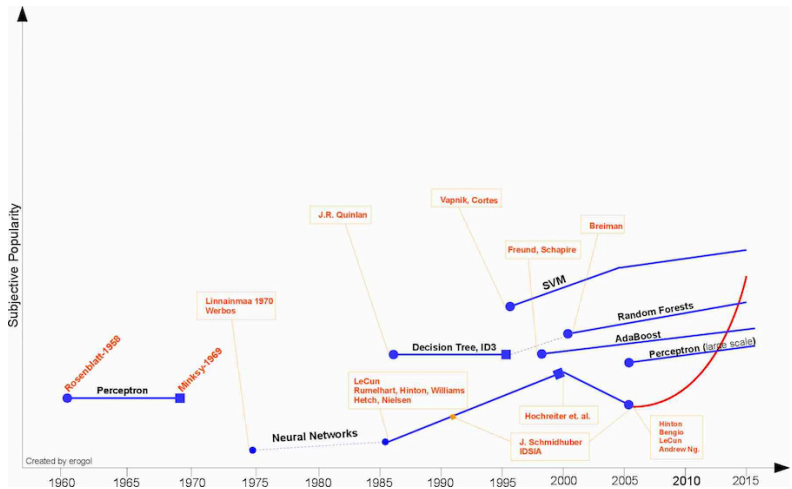
- 梯度消失爆炸问题和反向传播



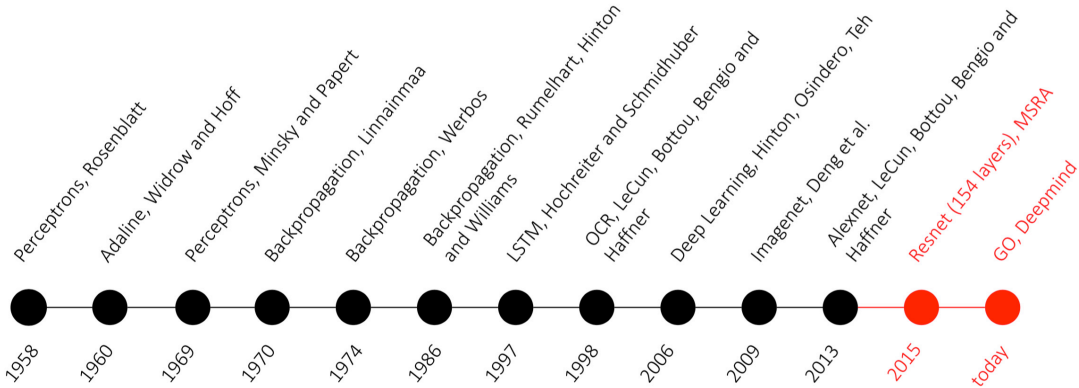
反向传播随着层数积累导致梯度消失！

2.3 为什么深度学习会成功

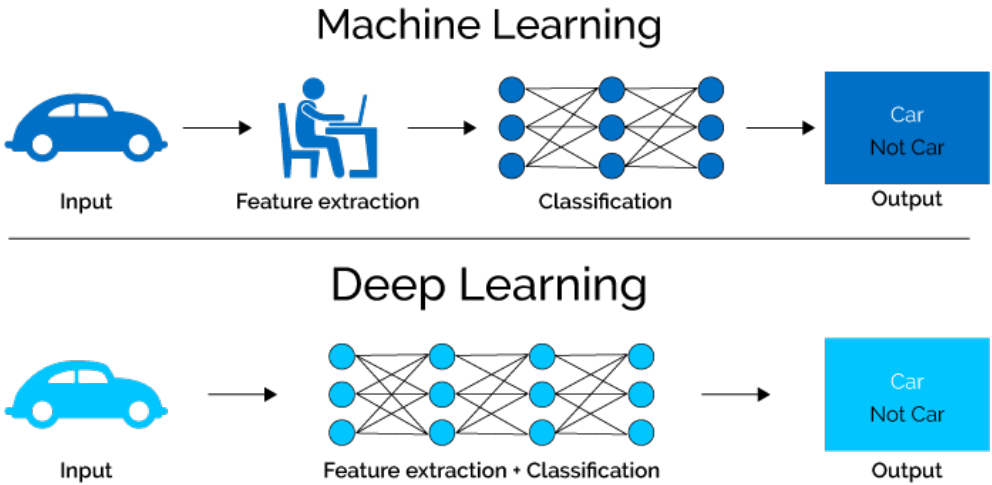
- 从机器学习到深度学习：SVM 、集成学习、贝叶斯网络



- 深度学习发展简化时间表

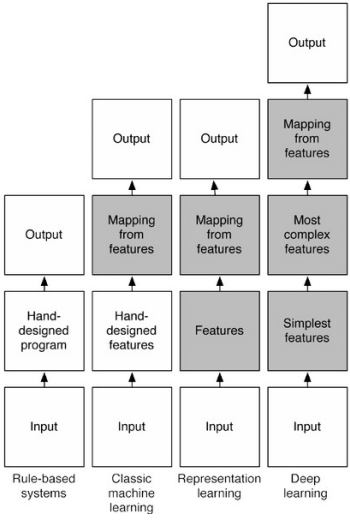


- 人工特征 VS 自动特征

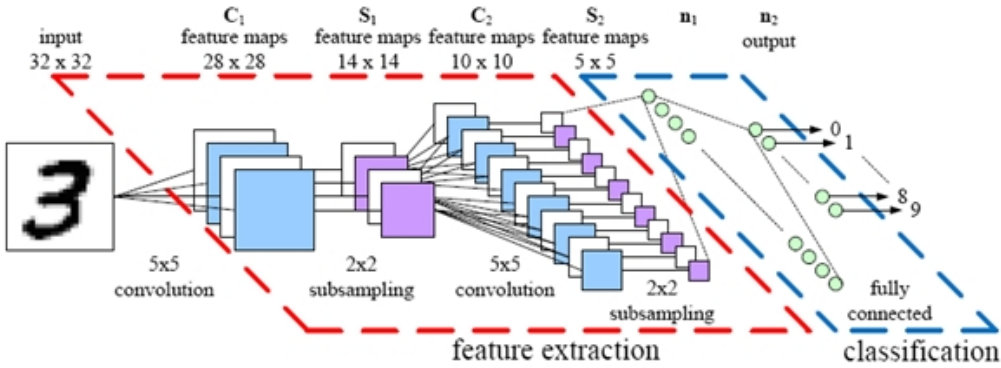


自适应特征提取！

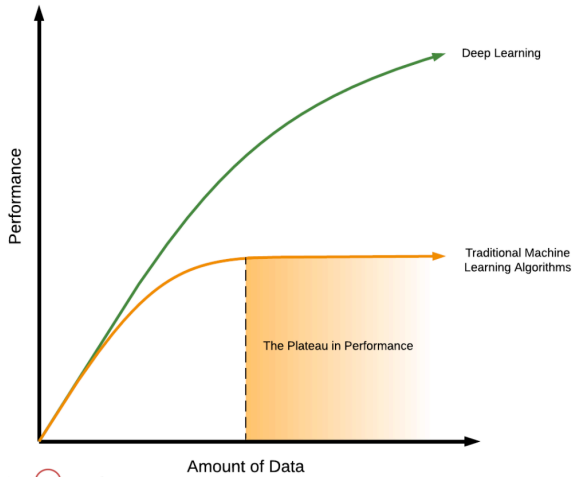
- 表征学习：自动特征提取是不是就够了？



- 表征学习：手写体识别

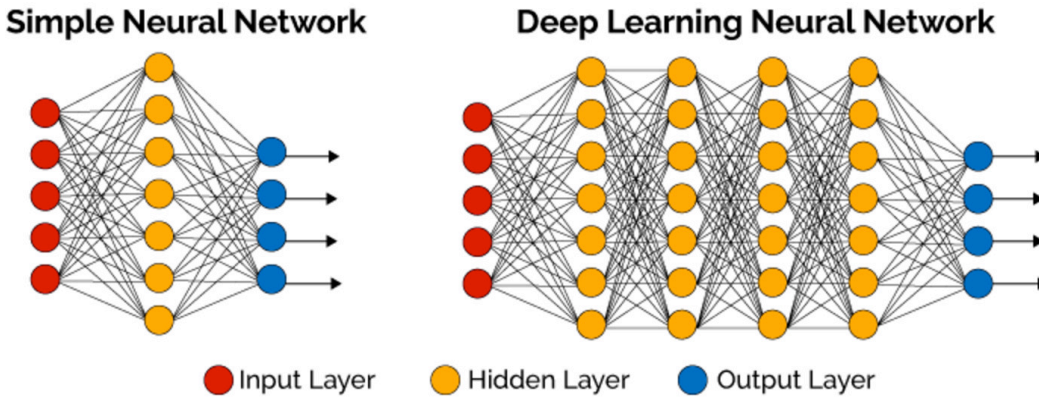

input
32 x 32
C1
feature maps
28 x 28
S1
feature maps
14 x 14
C2
feature maps
10 x 10
S2
feature maps
5 x 5
n1
n2
output
0
1
8
9
5x5
convolution
2x2
subsampling
5x5
convolution
2x2
subsampling
fully
connected
feature extraction
classification

- 数据量 VS 学习能力容量



强大的学习和拟合能力！

- 简单网络 VS 复杂网络



5 层以上，7 层以上，10 层以上！

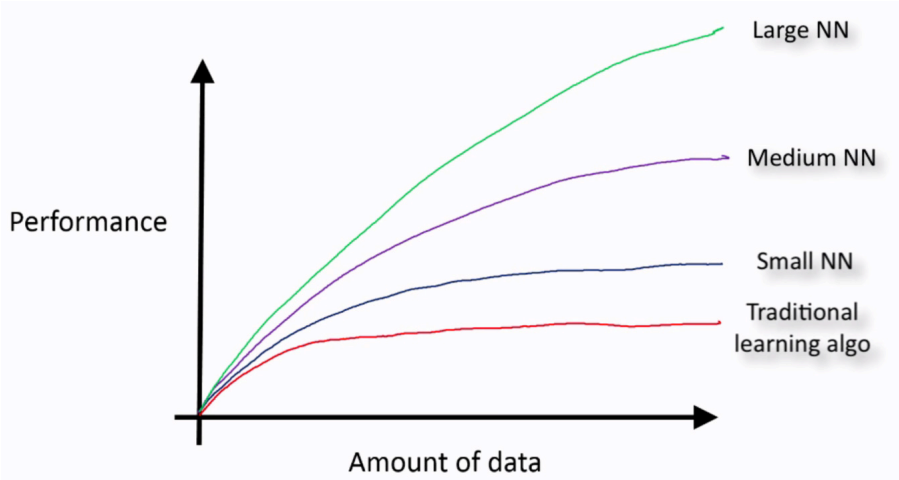
- 超复杂网络

[1]Inception 5 (GoogLeNet)

Inception 7a

[1]Going Deeper with Convolutions, [C. Szegedy et al, CVPR 2015]

- 简单网络和复杂网络学习能力



复杂网络、庞大的超参数更适合超大数据集！

2.3.1 深度学习的破局：语言识别突破、图像识别突破

- 2006 年，Hinton 和学生 Salakhutdinov 发表 Pre-Training 文章

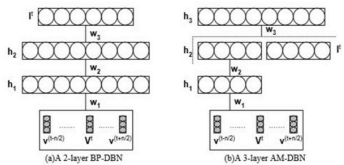
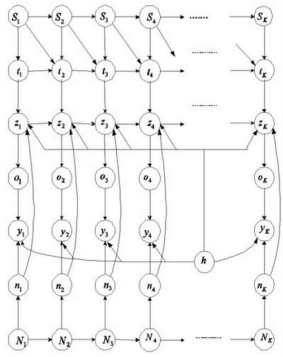


Table 4: *Reported results on TIMIT core test set*

Method	PER
Stochastic Segmental Models [28]	36%
Conditional Random Field [29]	34.8%
Large-Margin GMM [30]	33%
CD-HMM [4]	27.3%
Augmented conditional Random Fields [4]	26.6%
Recurrent Neural Nets [31]	26.1%
Bayesian Triphone HMM [32]	25.6%
Monophone HTMs [33]	24.8%
Heterogeneous Classifiers [34]	24.40%
Deep Belief Networks(DBNs)	23.0%

(Mohamed, Dahl, Hinton, 2009, 2012)

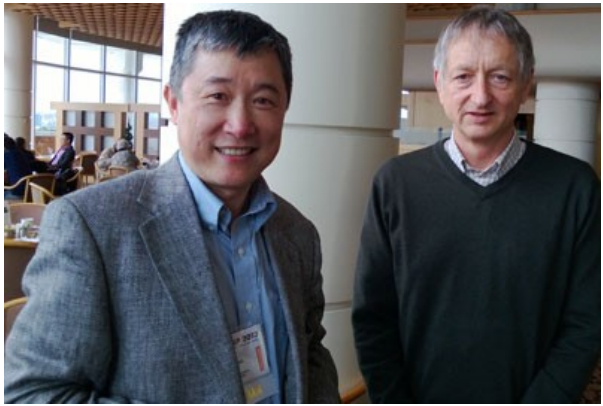
(after adding Backprop to the generative DBN)



(Deng et al., 2006; Deng & Yu, 2007)

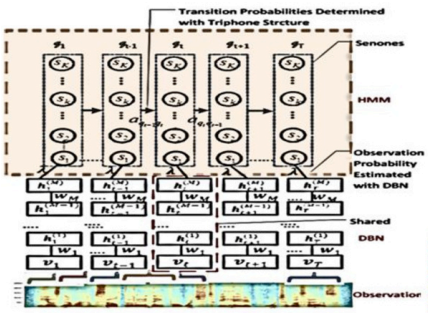
DBN 效果比 HMM 集成效果好！

- 2009 年，邓力邀请辛顿访问微软语言实验室！



他们一起召开了 2009 年 NIPS “Deep Learing for Speech Recognition” 的 Workshop！合作的第一个原型系统：超过 30% 的准确率的提升！

- 2011 年，微软将深度学习放到语言识别产品！

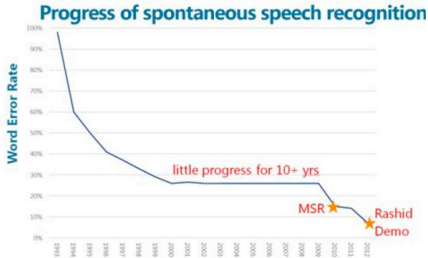


CD-DNN-HMM

Dahl, Yu, Deng, and Acero, "Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition," *IEEE Trans. ASLP*, Jan. 2012 (also ICASSP 2011)

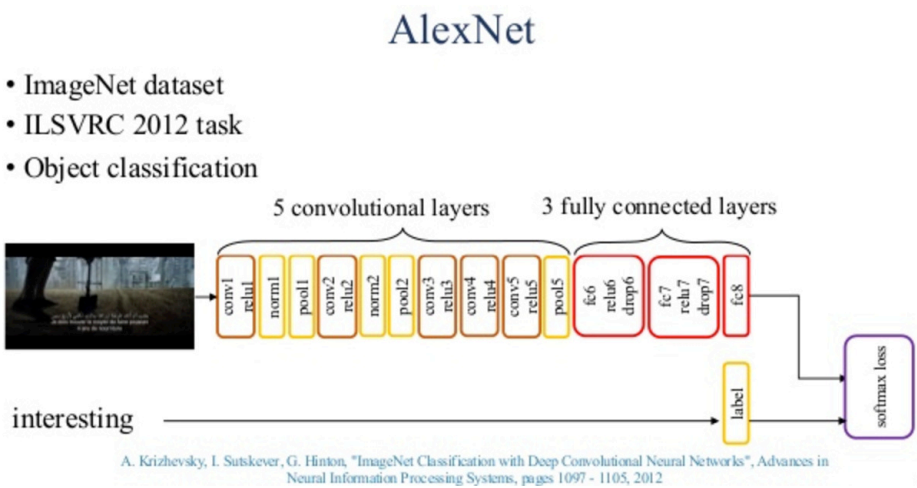
Seide et al, Interspeech, 2011.

After no improvement for 10+ years by the research community... ...MSR reduced error from ~23% to <13% (and under 7% for Rick Rashid's S2S demo in 2012)!



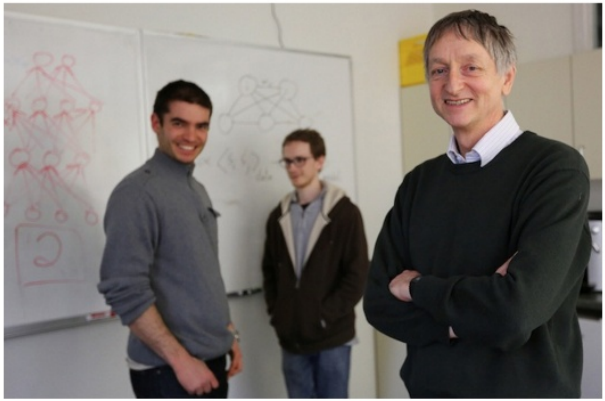
随后，Google 开始跟进！

- 2012 深度学习在图像识别突破



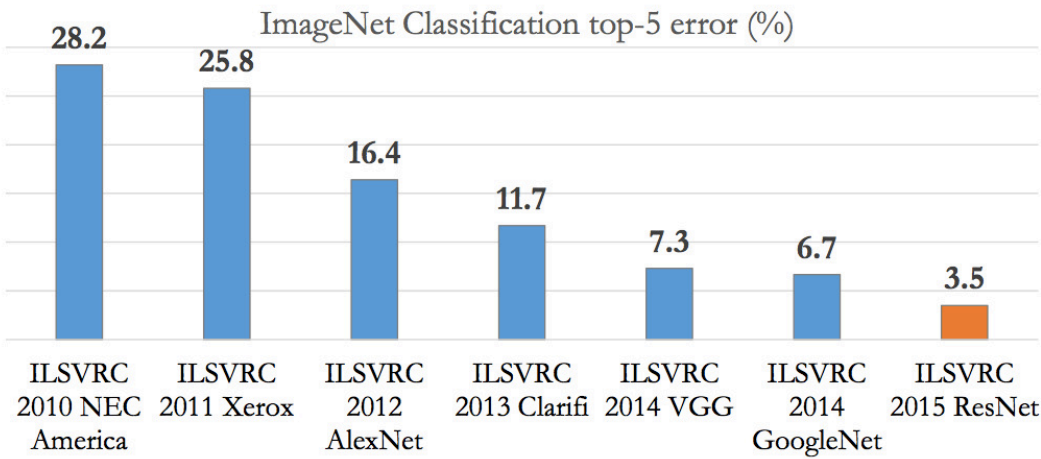
AlexNet 在 ILSVRC2012 竞赛上大幅度超过第二名！

- ILSVRC2012 冠军团队



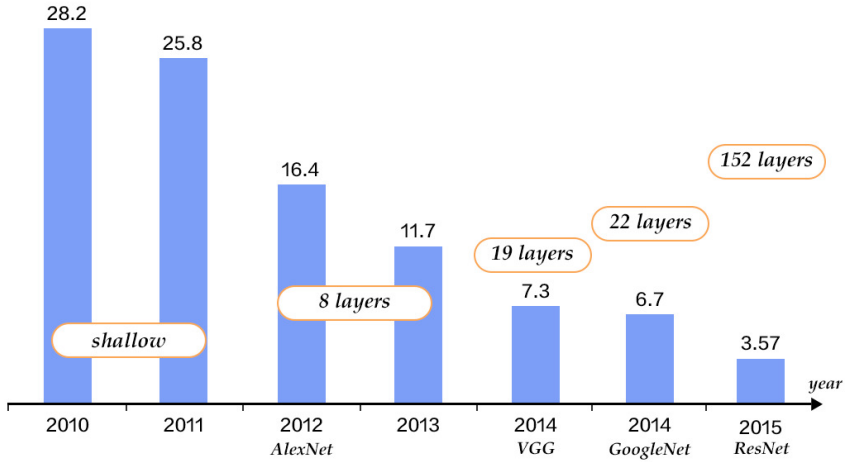
Ilya Sutskever, Alex Krizhevsky, Geoffrey Hinton

- ILSVRC 持续被深度学习突破



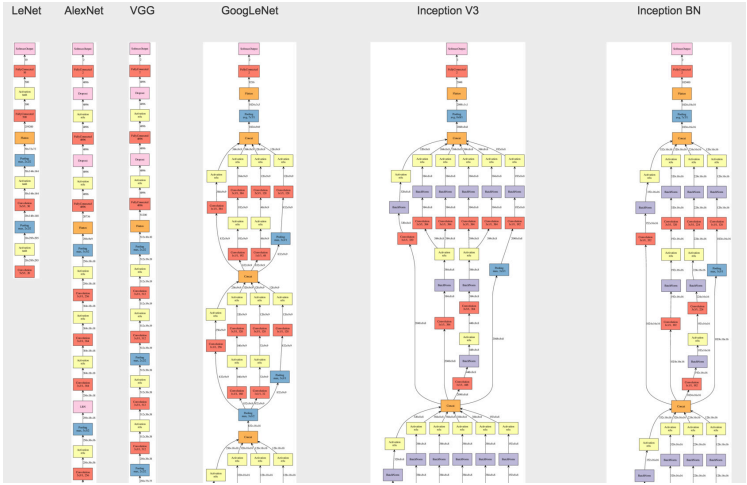
ILSVRC 随后几年一直被深度学习突破！

- ILSVRC 深度网络层数

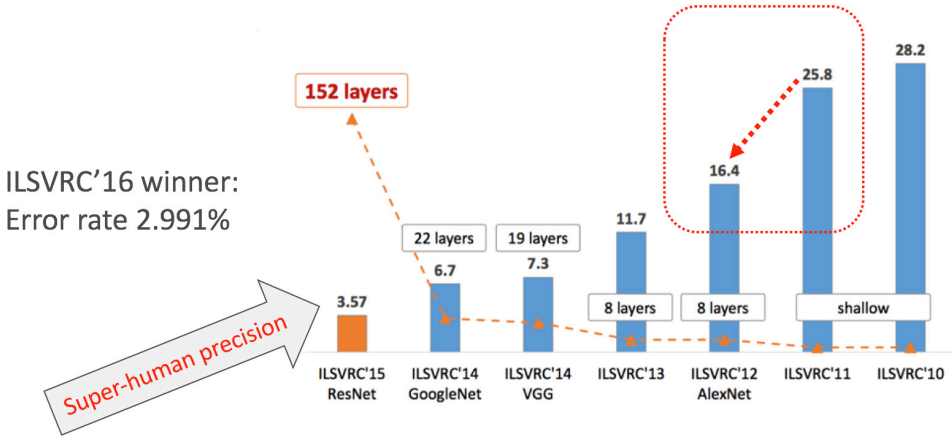


深度网络的层数一直被突破！

- 网络结构越来越复杂



- ILSVRC 2016 年超越人类



- 各大深度学习平台 (到 2017 年)

Software	Platform	Interface	GPU Support	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution
Caffe / Caffe2	Linux, Mac OS X, AWS, Windows support by Microsoft Research	C++, command line, Python, MATLAB	Yes	Yes	Yes	No	Yes
Deeplearning4j	Linux, Mac OS X, Windows, Android (Cross-platform)	Java, Scala, Clojure	Yes	Yes	Yes	Yes	Yes
Keras	Linux, Mac OS X, Windows	Python	Yes	Yes	Yes	Yes	Yes
Microsoft Cognitive Toolkit - CNTK	Windows, Linux (OSX via Docker on roadmap)	Python, C++, Command line, BrainScript (.NET on roadmap)	Yes	Yes	Yes	No	Yes
MXNet	Linux, Mac OS X, Windows, AWS, Android, iOS, JavaScript	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala	Yes	Yes	Yes	Yes	Yes
PaddlePaddle	Linux, Mac OS X	Python, C++	Yes	Yes	Yes	?	Yes
TensorFlow	Linux, Mac OS X, Windows	Python, (C/C++ public API only for executing graphs)	Yes	Yes	Yes	Yes	Yes
Theano	Cross-platform	Python	Yes	Yes	Yes	Yes	Yes
Torch	Linux, Mac OS X, Windows, Android, iOS	Lua, LuaJIT, C, utility library for C++/OpenCL	Yes	Yes	Yes	Yes	Yes

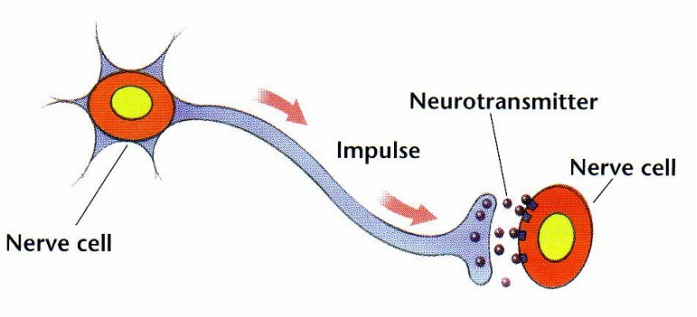
从神经网络到深度学习的变化 (加深以外)：

- 以前问题
  - 处理能力弱 (没有 GPU)
  - 数据量不够
  - 过拟合处理不行
  - 梯度消失
- 现在突破
  - 硬件加速发展
  - 数据量更大
  - 正则化方法发展，如 Dropout
  - 优化算法发展，如 Adam, Batch Normalization

3 神经网络基础和数学基础

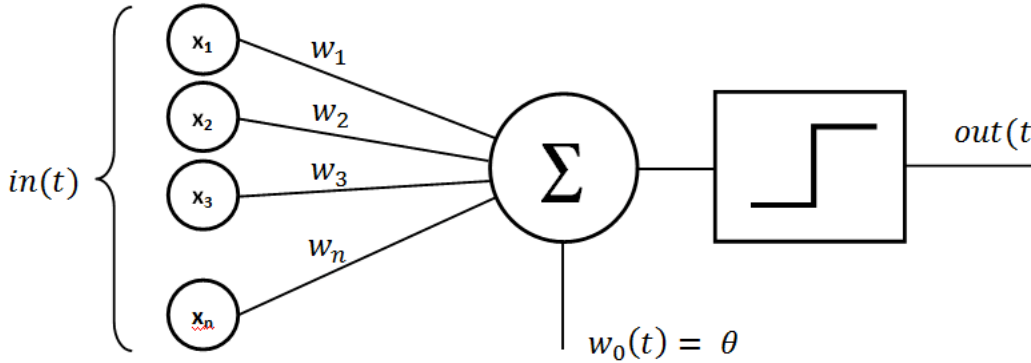
3.1 神经网络模型

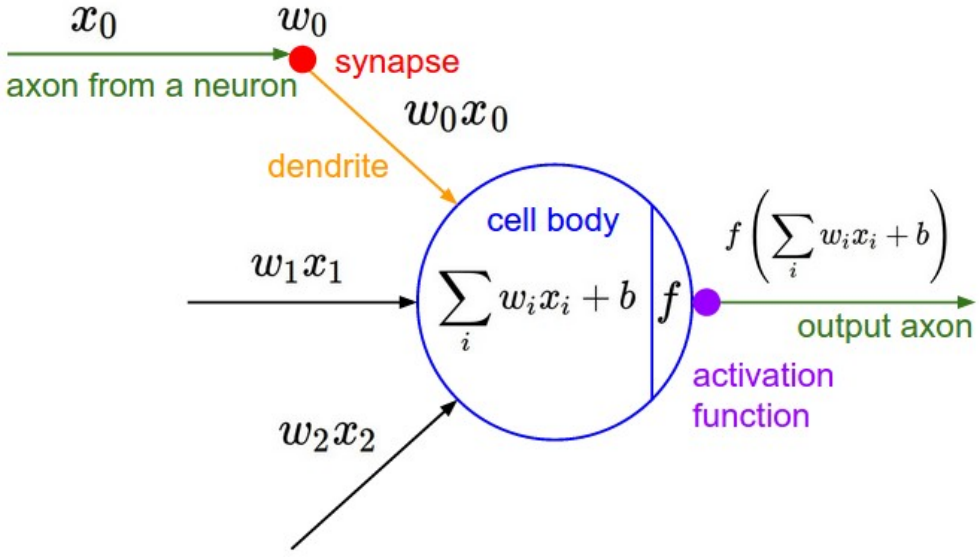
- 神经细胞



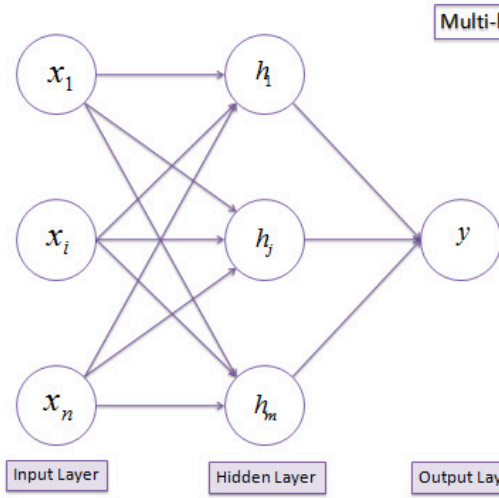
信号加权求和、阶跃激活函数

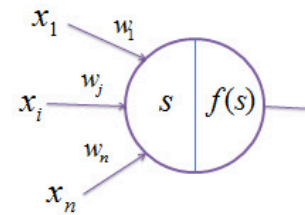
- MP 神经元模型




x_0
w_0
synapse
axon from a neuron
w_0x_0
dendrite
cell body
$f\left(\sum_i w_ix_i + b\right)$
w_1x_1
$\sum_i w_ix_i + b$
f
output axon
activation function
w_2x_2

- 神经网络模型





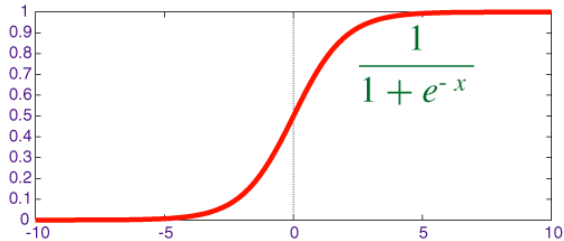
Summation

$$s = \sum w \cdot x$$

Transformation

$$f(s) = \frac{1}{1+e^{-s}}$$

- Sigmoid 激活函数和导数

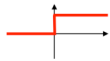
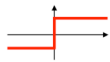
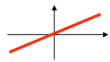
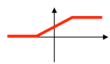
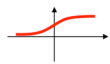
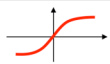
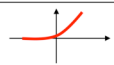


$$f(x) = \frac{1}{1+e^{-x}}$$

$$f'(x) = f(x)(1-f(x))$$

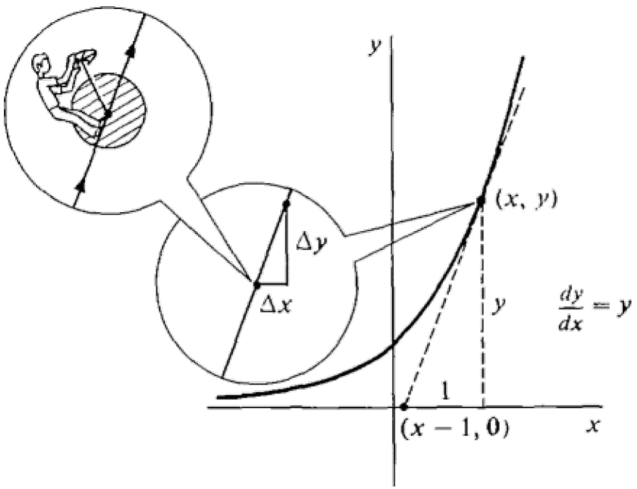
良好的导数性质！

$$
\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] \\
&= \frac{d}{dx}\left(1+\mathrm{e}^{-x}\right)^{-1} \\
&= -(1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{e^{-x}}{\left(1+e^{-x}\right)^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) \\
&= \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) \\
&= \sigma(x) \cdot (1-\sigma(x))
\end{aligned}
$$

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

3.2 导数工具

- 什么是导数



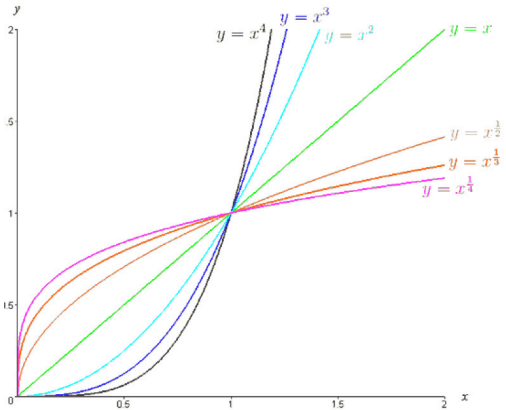
- 导数统一对函数的认识！

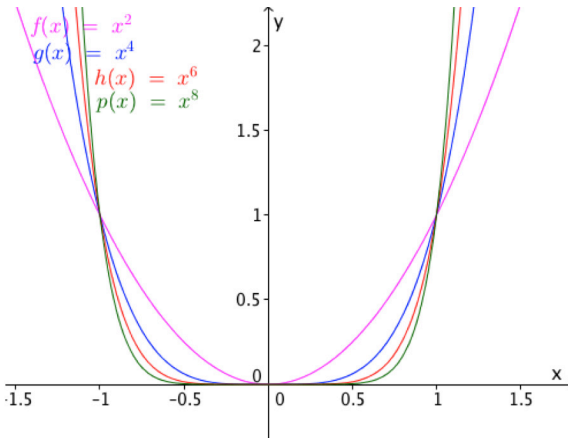
1) 幂函数：长度、面积、体积
2) 指数函数：复利计算，利滚利
3) 对数函数：天文大数字计算化简

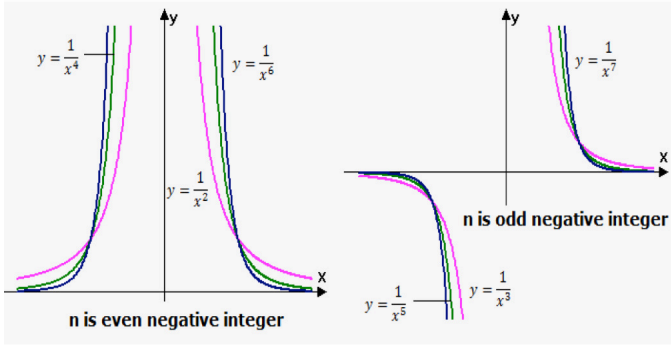
- 如何统一？

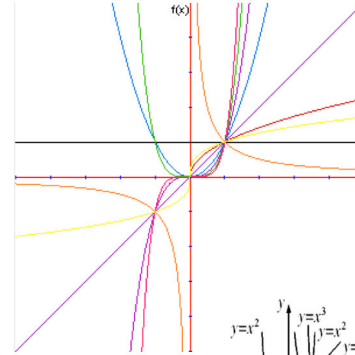
1) 任意函数泰勒展开到幂函数！
2) 幂函数求导成直线！

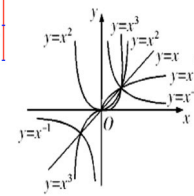
- 幂函数：长度、面积、体积


y
x
$y = x^4$
$y = x^3$
$y = x^2$
$y = x$
$y = x^{\frac{1}{2}}$
$y = x^{\frac{1}{3}}$
$y = x^{\frac{1}{4}}$
反函数

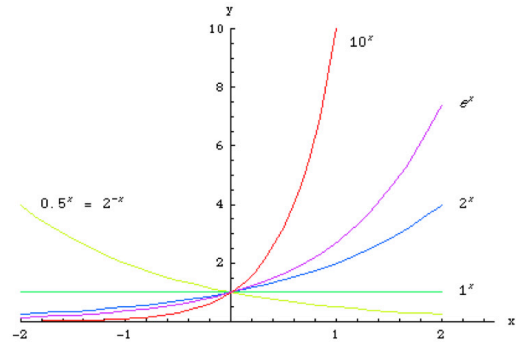

$f(x) = x^2$
$g(x) = x^4$
$h(x) = x^6$
$p(x) = x^8$
y
x

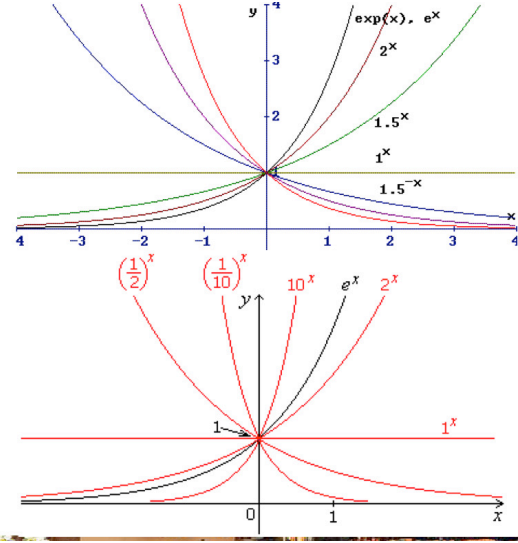

y
x
$y=\frac{1}{x^4}$
$y=\frac{1}{x^6}$
$y=\frac{1}{x^2}$
n is even negative integer
y
x
$y=\frac{1}{x^7}$
n is odd negative integer
$y=\frac{1}{x^5}$
$y=\frac{1}{x^3}$


f(x)
x

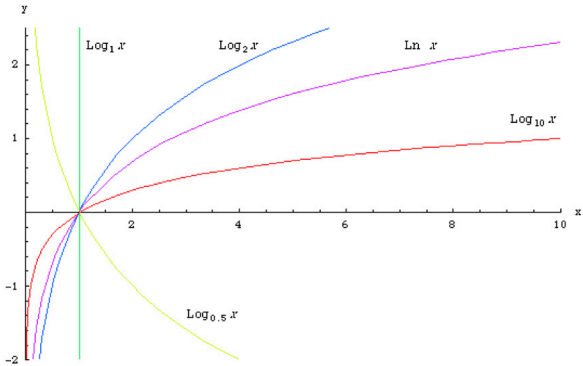

y
x
0
$y=x^3$
$y=x^2$
$y=x$
$y=x^{\frac{1}{2}}$
$y=x^{-1}$
$y=x^{-1}$
$y=x^3$

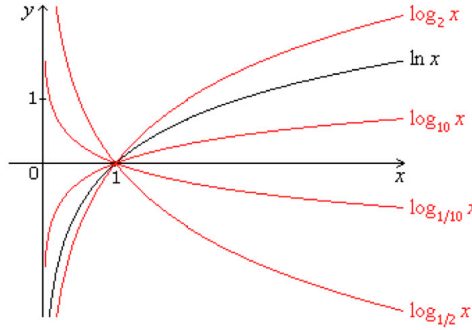
- 指数函数：复利计算，利滚利



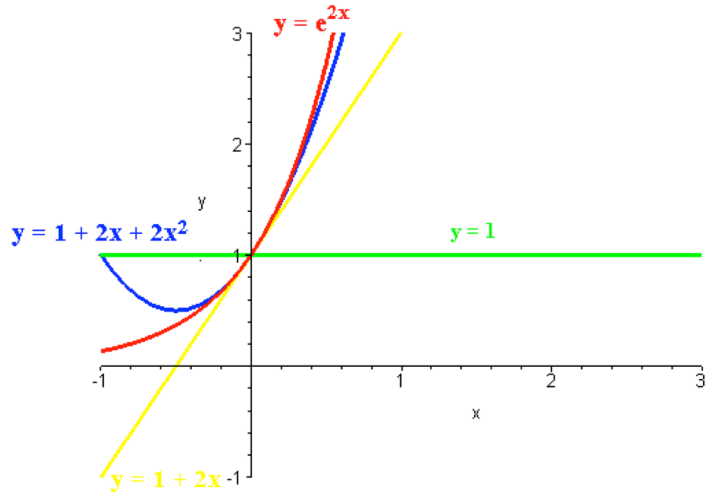


- 对数函数：天文大数字计算化简





- 幂函数之和的近似
  - y = 1
  - y = 1 + 2x
  - y = 1 + 2x + 2x²

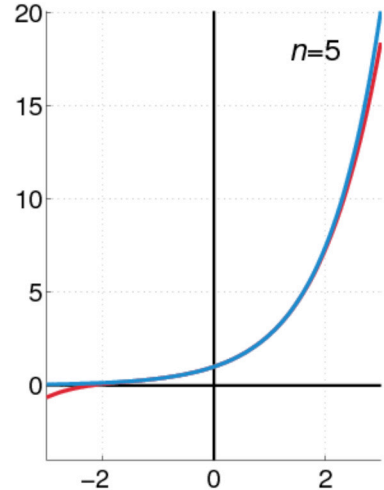


- 泰勒展开

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$



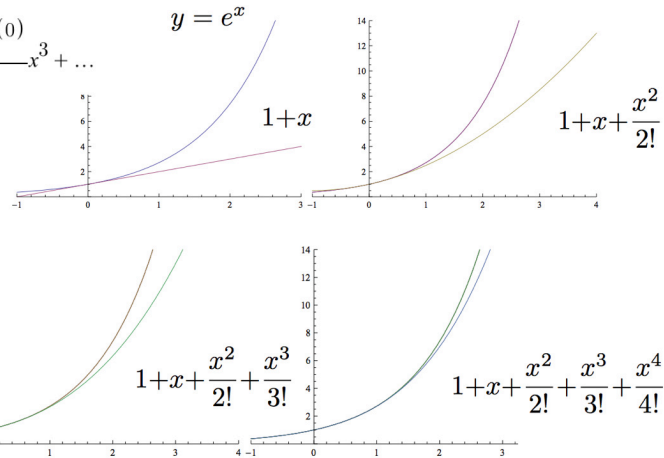
$$= e^0 + \frac{\frac{d}{dx}(e^x)(0)}{1!}x + \frac{\frac{d^2}{dx^2}(e^x)(0)}{2!}x^2 + \frac{\frac{d^3}{dx^3}(e^x)(0)}{3!}x^3 + \dots$$

$\frac{d}{dx}(e^x)(0)$: 1

$\frac{d^2}{dx^2}(e^x)(0)$: 1

$\frac{d^3}{dx^3}(e^x)(0)$: 1

$\frac{d^4}{dx^4}(e^x)(0)$: 1



$$f(x) = \sum_{n=0}^{\infty} a_n (x-c)^n = a_0 + a_1(x-c) + a_2(x-c)^2 + a_3(x-c)^3 + a_4(x-c)^4 + \cdots$$

$$\begin{aligned}
f(x) &= a_0 + a_1(x-c) + a_2(x-c)^2 + a_3(x-c)^3 + a_4(x-c)^4 + \cdots \\
f'(x) &= a_1 + 2a_2(x-c) + 3a_3(x-c)^2 + 4a_4(x-c)^3 + \cdots \\
f''(x) &= 2a_2 + 3\cdot 2\cdot a_3(x-c) + 4\cdot 3\cdot a_4(x-c)^2 + 5\cdot 4\cdot a_5(x-c)^3 + \cdots \\
f'''(x) &= 3\cdot 2\cdot a_3 + 4\cdot 3\cdot 2\cdot a_4(x-c) + 5\cdot 4\cdot 3\cdot a_5(x-c)^2 + \cdots \\
f''''(x) &= 4\cdot 3\cdot 2\cdot a_4 + 5\cdot 4\cdot 3\cdot 2\cdot a_5(x-c) + \cdots \\
&\vdots \quad \vdots \quad \vdots
\end{aligned}$$

$$\frac{d}{dx} x^n = nx^{n-1}$$

$$f(x) = x^n = nx^{n-1}$$

decrease by 1

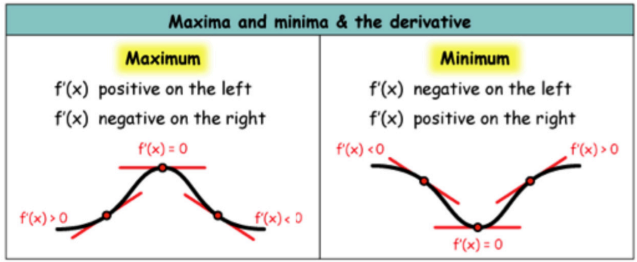
$$
\begin{aligned}
f(c) &= a_0 \\
f'(c) &= a_1 \\
f''(c) &= 2a_2 \\
f'''(c) &= 3 \cdot 2 \cdot a_3 \\
f''''(c) &= 4 \cdot 3 \cdot 2 \cdot a_4 \\
\vdots &\quad \vdots \quad \vdots
\end{aligned}
$$

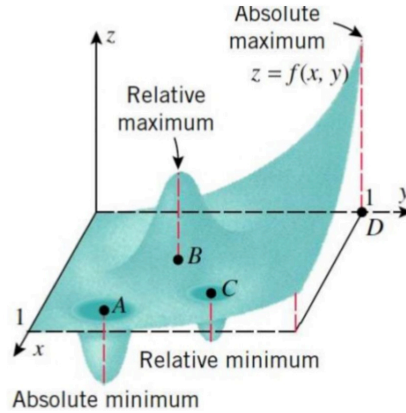
$$
f(x) = \sum_{n=0}^{\infty} a_n (x-c)^n
$$

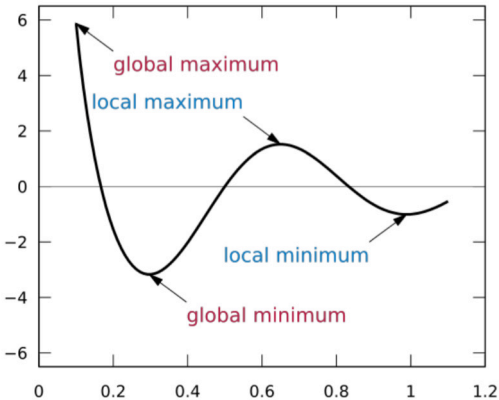
$$
\sum_{n=0}^{\infty} \frac{f^{(n)}(c)}{n!} (x-c)^n
$$

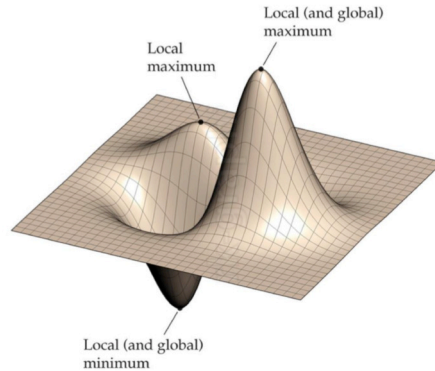
$$
\frac{f^{(n)}(c)}{n!} \qquad a_n
$$

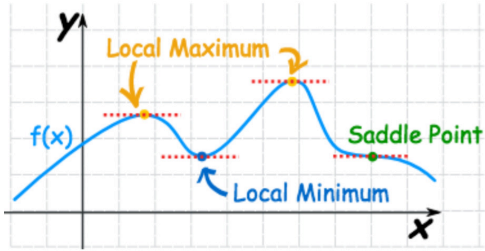
- 导数求极值：必要条件

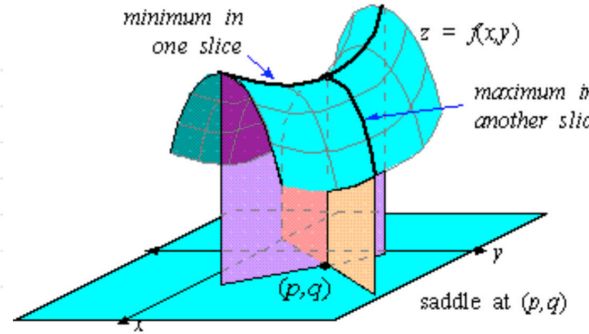

Maxima and minima & the derivative
Maximum
f'(x) positive on the left
f'(x) negative on the right
f'(x) = 0
f'(x) > 0
f'(x) < 0
Minimum
f'(x) negative on the left
f'(x) positive on the right
f'(x) < 0
f'(x) > 0
f'(x) = 0


Absolute maximum
$z = f(x, y)$
Relative maximum
z
y
x
1
1
D
A
B
C
Relative minimum
Absolute minimum


global maximum
local maximum
local minimum
global minimum
6
4
2
0
−2
−4
−6
0
0.2
0.4
0.6
0.8
1
1.2


Local (and global) maximum
Local maximum
Local (and global) minimum


y
Local Maximum
f(x)
Saddle Point
Local Minimum
x


minimum in one slice
z = f(x,y)
maximum in another slice
y
(p,q)
saddle at (p,q)
x

3.3 经验风险最小和梯度下降

- 经验风险最小

1. Given training data:

$$\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$$

2. Choose each of these:
   - Decision function

$$\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

   - Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

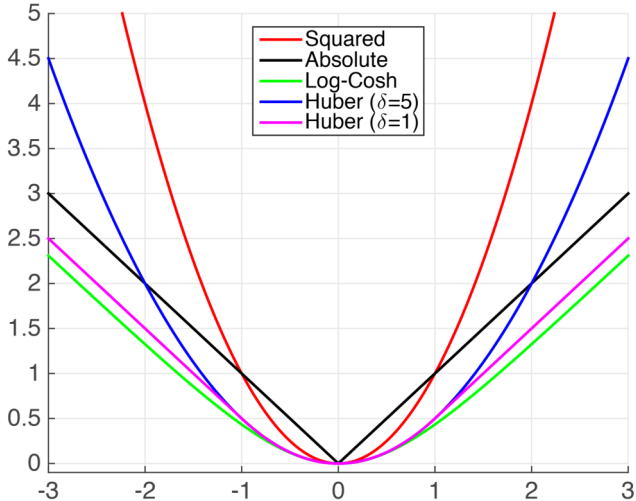
3. Define goal:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

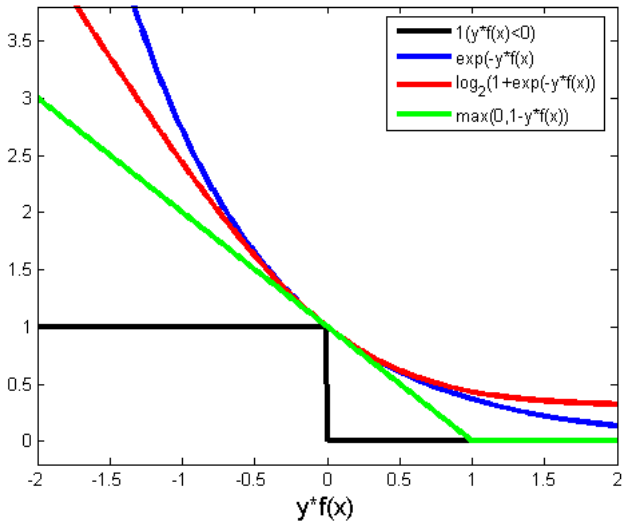
4. Train with SGD:
(take small steps opposite the gradient)

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

- 常见回归损失函数 Loss Function



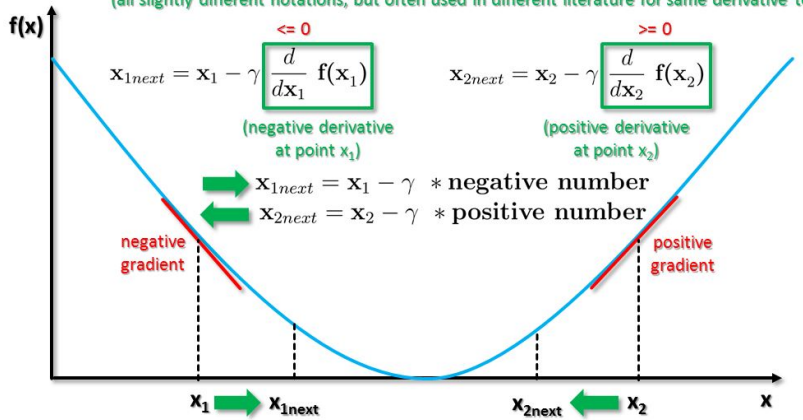
- 常见分类损失函数 Loss Function

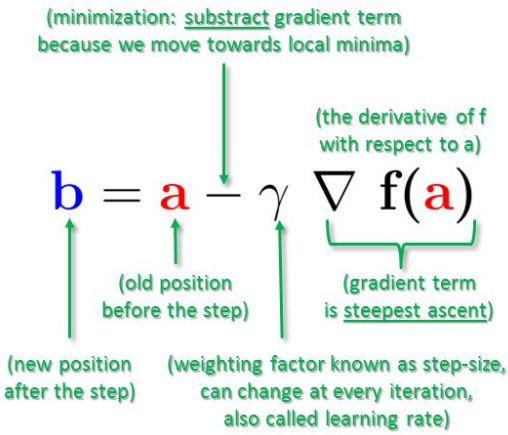


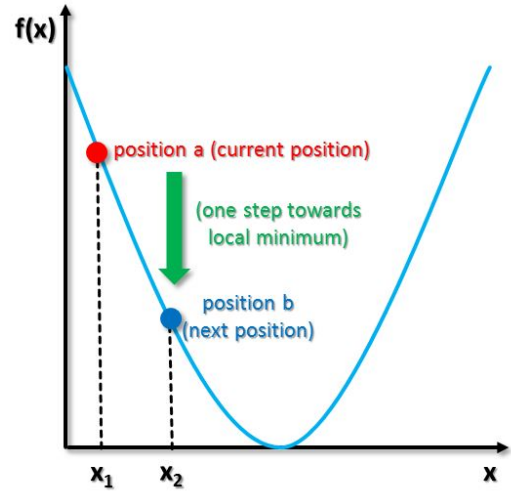
- 梯度下降 GD

$$\mathbf{b} = \mathbf{a} - \gamma \nabla \mathbf{f(a)} \quad \mathbf{b} = \mathbf{a} - \gamma \frac{\partial}{\partial \mathbf{a}} \mathbf{f(a)} \quad \mathbf{b} = \mathbf{a} - \gamma \frac{d}{d\mathbf{a}} \mathbf{f(a)}$$

(all slightly different notations, but often used in different literature for same derivative term)



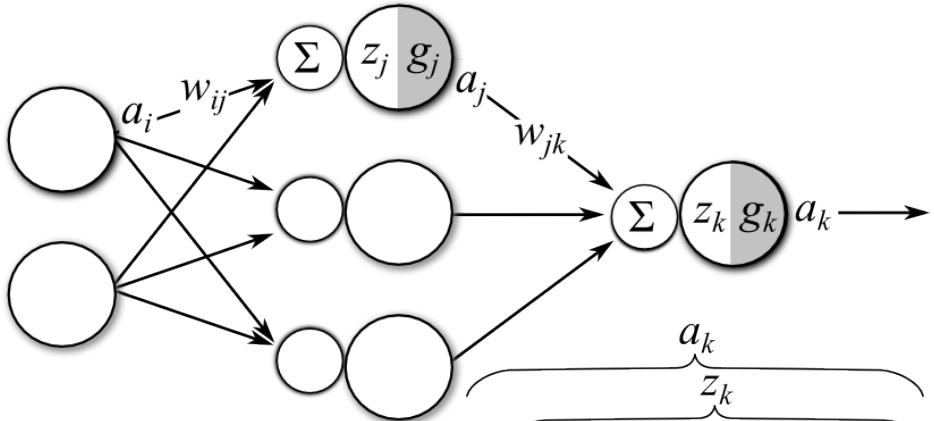




当风险函数不可以直接求导时候，迭代求解就显得很必要！

3.4 反向传播 BP 算法

- 3 层网络


Σ
z_j g_j
a_j
w_{ij}
a_i
w_{jk}
Σ
z_k g_k
a_k
a_k
z_k
$a_k = g_k(b_k + \Sigma_j g_j(b_j + \Sigma_i a_i w_{ij})w_{jk})$
z_j
a_j

$$z_j = b_j + \sum_i a_i w_{ij}$$

- 误差 (风险) 函数：

$$E = \frac{1}{2}\sum_{k\in K}(a_k - t_k)^2$$

- 输出层权重导数：jk 层

$$\begin{aligned}\frac{\partial E}{\partial w_{jk}} &= \frac{1}{2}\sum_{k\in K}(a_k - t_k)^2 \\ &= (a_k - t_k)\frac{\partial}{\partial w_{jk}}(a_k - t_k)\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial w_{jk}} &= (a_k - t_k)\frac{\partial}{\partial w_{jk}}a_k \\ &= (a_k - t_k)\frac{\partial}{\partial w_{jk}}g_k(z_k) \\ &= (a_k - t_k)g_k^{'}(z_k)\frac{\partial}{\partial w_{jk}}z_k,\end{aligned}$$

已知：

$$z_k = b_j + \sum_j g_j(z_j) w_{jk}$$

那么：

$$\frac{\partial z_k}{\partial w_{jk}} = g_j(z_j) = a_j$$

$$\frac{\partial E}{\partial w_{jk}} = (a_k - t_k) g_k^{'}(z_k) a_j$$

设定：

$$\delta_k = (a_k - t_k) g_k^{'}(z_k)$$

那么：

$$\frac{\partial E}{\partial w_{jk}} = \delta_k a_j$$

根据 GD 算法：

$$w_{jk} \leftarrow w_{jk} - \eta \frac{\partial E}{\partial w_{jk}}$$

类似，对于截距项：

$$\frac{\partial}{\partial b_k} z_k = \frac{\partial}{\partial b_k} \left[b_k + \sum_j g_j(z_j) \right] = 1$$

$$\begin{aligned}\frac{\partial E}{\partial b_k} &= (a_k - t_k) g_k^{'}(z_k)(1) \\ &= \delta_k\end{aligned}$$

- 隐藏层：ij 层

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}} &= \frac{1}{2}\sum_{k\in K}(a_k - t_k)^2 \\ &= \sum_{k\in K}(a_k - t_k)\frac{\partial}{\partial w_{ij}} a_k\end{aligned}$$

$$a_k = g_k(z_k)$$

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}} &= \sum_{k\in K}(a_k - t_k)\frac{\partial}{\partial w_{ij}}g_k(z_k) \\ &= \sum_{k\in K}(a_k - t_k)g_k^{'}(z_k)\frac{\partial}{\partial w_{ij}}z_k\end{aligned}$$

$$\begin{aligned}z_k &= b_k + \sum_j a_j w_{jk} \\ &= b_k + \sum_j g_j(z_j)w_{jk} \\ &= b_k + \sum_j g_j(b_i + \sum_i z_i w_{ij})w_{jk}\end{aligned}$$

$$
\begin{aligned}
\frac{\partial z_k}{\partial w_{ij}} &= \frac{\partial z_k}{\partial a_j}\frac{\partial a_j}{\partial w_{ij}} \\
&= \frac{\partial}{\partial a_j} a_j w_{jk} \frac{\partial a_j}{\partial w_{ij}} \\
&= w_{jk}\frac{\partial a_j}{\partial w_{ij}} \\
&= w_{jk}\frac{\partial g_j(z_j)}{\partial w_{ij}} \\
&= w_{jk} g_j^{'}(z_j)\frac{\partial z_j}{\partial w_{ij}} \\
&= w_{jk} g_j^{'}(z_j)\frac{\partial}{\partial w_{ij}}(b_i + \textstyle\sum_i a_i w_{ij}) \\
&= w_{jk} g_j^{'}(z_j) a_i
\end{aligned}
$$

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}} &= \sum_{k\in K}(a_k - t_k)g_k^{'}(z_k)w_{jk}g_j^{'}(z_j)a_i \\ &= g_j^{'}(z_j)a_i\sum_{k\in K}(a_k - t_k)g_k^{'}(z_k)w_{jk} \\ &= a_i g_j^{'}(z_j)\sum_{k\in K}\delta_k w_{jk}\end{aligned}$$

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}} &= a_i g_j^{'}(z_j)\sum_{k\in K}\delta_k w_{jk} \\ &= \delta_j a_i\end{aligned}$$

where

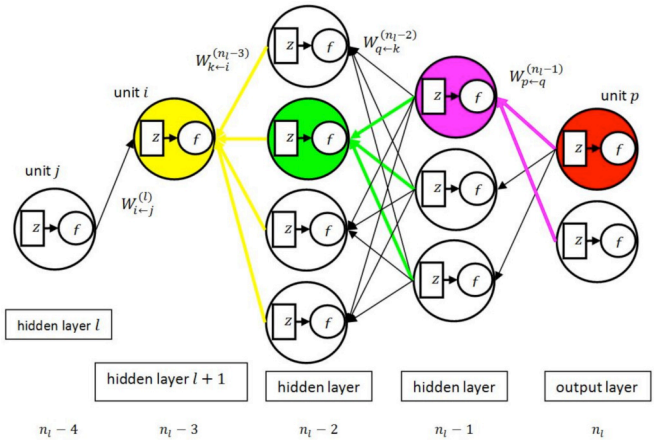
$$\delta_j = g_j^{'}(z_j)\sum_{k\in K}\delta_k w_{jk}$$

$$
\begin{aligned}
\frac{\partial z_k}{\partial b_i} &= w_{jk} g_j^{'}(z_j) \frac{\partial z_j}{\partial b_i} \\
&= w_{jk} g_j^{'}(z_j) \frac{\partial}{\partial b_i}(b_i + \sum_i a_i w_{ij}) \\
&= w_{jk} g_j^{'}(z_j)(1),
\end{aligned}
$$

giving

$$
\begin{aligned}
\frac{\partial E}{\partial b_i} &= g_j^{'}(z_j) \sum_{k \in K} \delta_k w_{jk} \\
&= \delta_j
\end{aligned}
$$

- 5 层网络连续推导


unit i
unit j
unit p
$W_{k\leftarrow i}^{(n_l-3)}$
$W_{q\leftarrow k}^{(n_l-2)}$
$W_{p\leftarrow q}^{(n_l-1)}$
$W_{i\leftarrow j}^{(l)}$
z
f
hidden layer l
hidden layer $l+1$
hidden layer
hidden layer
output layer
n_l-4
n_l-3
n_l-2
n_l-1
n_l

- 对第一层的参数求导数

根据：

$$\theta \leftarrow \theta - \eta \frac{\partial E}{\partial \theta}$$

将平方误差按第一层参数 $n_l - 4$ 层进行求导：

$$\frac{\partial J(W,b;x,y)}{\partial W_{i\leftarrow j}^{(n_l-4)}}=\frac{\partial\left(\frac{1}{2}\|y-h(W,b;x)\|^2\right)}{\partial W_{i\leftarrow j}^{(n_l-4)}}=\frac{\partial\left(\frac{1}{2}\sum_{p=1}^{s_{n_l}}\left(y_p-h_p(W,b;x)\right)^2\right)}{\partial W_{i\leftarrow j}^{(n_l-4)}}$$

$$=-\sum_{p=1}^{s_{n_l}}\left(y_p-h_p(W,b;x)\right)\cdot\frac{\partial h_p(W,b;x)}{\partial W_{i\leftarrow j}^{(n_l-4)}}$$

$$\frac{\partial h_p(W,b;x)}{\partial W_{i\leftarrow j}^{(n_l-4)}}=\frac{\partial a_p^{(n_l)}}{\partial W_{i\leftarrow j}^{(n_l-4)}}=\frac{da_p^{(n_l)}}{dz_p^{(n_l)}}\cdot\frac{\partial z_p^{(n_l)}}{\partial W_{i\leftarrow j}^{(n_l-4)}}=\frac{da_p^{(n_l)}}{dz_p^{(n_l)}}\cdot\frac{\partial\left(W_p^{(n_l-1)}\cdot a^{(n_l-1)}\right)}{\partial W_{i\leftarrow j}^{(n_l-4)}}$$

$$=\frac{da_p^{(n_l)}}{dz_p^{(n_l)}}\cdot\left(\sum_{q=1}^{s_{n_l-1}}W_{p\leftarrow q}^{(n_l-1)}\cdot\frac{\partial a_q^{(n_l-1)}}{\partial W_{i\leftarrow j}^{(n_l-4)}}\right)=\frac{da_p^{(n_l)}}{dz_p^{(n_l)}}\cdot\left(\sum_{q=1}^{s_{n_l-1}}W_{p\leftarrow q}^{(n_l-1)}\cdot\left(\frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}}\cdot\frac{\partial z_q^{(n_l-1)}}{\partial W_{i\leftarrow j}^{(n_l-4)}}\right)\right)$$

$$=\frac{da_p^{(n_l)}}{dz_p^{(n_l)}}\cdot\left(\sum_{q=1}^{s_{n_l-1}}W_{p\leftarrow q}^{(n_l-1)}\cdot\left(\frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}}\cdot\left(\sum_{k=1}^{s_{n_l-2}}W_{q\leftarrow k}^{(n_l-2)}\cdot\frac{\partial a_k^{(n_l-2)}}{\partial W_{i\leftarrow j}^{(n_l-4)}}\right)\right)\right)$$

$$=\frac{da_p^{(n_l)}}{dz_p^{(n_l)}}\cdot\left(\sum_{q=1}^{s_{n_l-1}}W_{p\leftarrow q}^{(n_l-1)}\cdot\left(\frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}}\cdot\left(\sum_{k=1}^{s_{n_l-2}}W_{q\leftarrow k}^{(n_l-2)}\cdot\left(\frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}}\cdot\frac{\partial z_k^{(n_l-2)}}{\partial W_{i\leftarrow j}^{(n_l-4)}}\right)\right)\right)\right)$$

$$=\frac{da_p^{(n_l)}}{dz_p^{(n_l)}}\cdot\left(\sum_{q=1}^{s_{n_l-1}}W_{p\leftarrow q}^{(n_l-1)}\cdot\left(\frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}}\cdot\left(\sum_{k=1}^{s_{n_l-2}}W_{q\leftarrow k}^{(n_l-2)}\cdot\left(\frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}}\cdot\left(\sum_{i=1}^{s_{n_l-3}}W_{k\leftarrow i}^{(n_l-3)}\cdot\frac{\partial a_i^{(n_l-3)}}{\partial W_{i\leftarrow j}^{(n_l-4)}}\right)\right)\right)\right)\right)$$

$$= \frac{da_p^{(n_l)}}{dz_p^{(n_l)}} \cdot \left(\sum_{q=1}^{s_{n_l-1}} W_{p\leftarrow q}^{(n_l-1)} \cdot \left(\frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}} \cdot \left(\sum_{k=1}^{s_{n_l-2}} W_{q\leftarrow k}^{(n_l-2)} \cdot \left(\frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}} \cdot W_{k\leftarrow i}^{(n_l-3)} \cdot \frac{\partial a_i^{(n_l-3)}}{\partial W_{i\leftarrow j}^{(n_l-4)}} \right) \right) \right) \right)$$

$$= \frac{da_p^{(n_l)}}{dz_p^{(n_l)}} \cdot \left(\sum_{q=1}^{s_{n_l-1}} W_{p\leftarrow q}^{(n_l-1)} \cdot \left(\frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}} \cdot \left(\sum_{k=1}^{s_{n_l-2}} W_{q\leftarrow k}^{(n_l-2)} \cdot \left(\frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}} \cdot W_{k\leftarrow i}^{(n_l-3)} \cdot \frac{da_i^{(n_l-3)}}{dz_i^{(n_l-3)}} \cdot \frac{\partial z_i^{(n_l-3)}}{\partial W_{i\leftarrow j}^{(n_l-4)}} \right) \right) \right) \right)$$

$$= \frac{da_p^{(n_l)}}{dz_p^{(n_l)}} \cdot \left(\sum_{q=1}^{s_{n_l-1}} W_{p\leftarrow q}^{(n_l-1)} \cdot \left(\frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}} \cdot \left(\sum_{k=1}^{s_{n_l-2}} W_{q\leftarrow k}^{(n_l-2)} \cdot \left(\frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}} \cdot W_{k\leftarrow i}^{(n_l-3)} \cdot \frac{da_i^{(n_l-3)}}{dz_i^{(n_l-3)}} \cdot \frac{\partial \left(W_i^{(n_l-4)} \cdot a^{(n_l-4)} \right)}{\partial W_{i\leftarrow j}^{(n_l-4)}} \right) \right) \right) \right)$$

$$= \frac{da_p^{(n_l)}}{dz_p^{(n_l)}} \cdot \left(\sum_{q=1}^{s_{n_l-1}} W_{p\leftarrow q}^{(n_l-1)} \cdot \left(\frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}} \cdot \left(\sum_{k=1}^{s_{n_l-2}} W_{q\leftarrow k}^{(n_l-2)} \cdot \left(\frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}} \cdot W_{k\leftarrow i}^{(n_l-3)} \cdot \frac{\partial a_i^{(n_l-3)}}{\partial W_{i\leftarrow j}^{(n_l-4)}} \right) \right) \right) \right)$$

$$= \frac{da_p^{(n_l)}}{dz_p^{(n_l)}} \cdot \left(\sum_{q=1}^{s_{n_l-1}} W_{p\leftarrow q}^{(n_l-1)} \cdot \left(\frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}} \cdot \left(\sum_{k=1}^{s_{n_l-2}} W_{q\leftarrow k}^{(n_l-2)} \cdot \left(\frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}} \cdot W_{k\leftarrow i}^{(n_l-3)} \cdot \frac{da_i^{(n_l-3)}}{dz_i^{(n_l-3)}} \cdot \frac{\partial z_i^{(n_l-3)}}{\partial W_{i\leftarrow j}^{(n_l-4)}} \right) \right) \right) \right)$$

$$= \frac{da_p^{(n_l)}}{dz_p^{(n_l)}} \cdot \left(\sum_{q=1}^{s_{n_l-1}} W_{p\leftarrow q}^{(n_l-1)} \cdot \left(\frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}} \cdot \left(\sum_{k=1}^{s_{n_l-2}} W_{q\leftarrow k}^{(n_l-2)} \cdot \left(\frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}} \cdot W_{k\leftarrow i}^{(n_l-3)} \cdot \frac{da_i^{(n_l-3)}}{dz_i^{(n_l-3)}} \cdot \frac{\partial \left(W_i^{(n_l-4)} \cdot a^{(n_l-4)} \right)}{\partial W_{i\leftarrow j}^{(n_l-4)}} \right) \right) \right) \right)$$

$$= \frac{da_p^{(n_l)}}{dz_p^{(n_l)}} \cdot \left(\sum_{q=1}^{s_{n_l-1}} W_{p\leftarrow q}^{(n_l-1)} \cdot \left(\frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}} \cdot \left(\sum_{k=1}^{s_{n_l-2}} W_{q\leftarrow k}^{(n_l-2)} \cdot \left(\frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}} \cdot W_{k\leftarrow i}^{(n_l-3)} \cdot \frac{da_i^{(n_l-3)}}{dz_i^{(n_l-3)}} \cdot a_j^{(n_l-4)} \right) \right) \right) \right)$$

$$= \sum_{q=1}^{s_{n_l-1}} \sum_{k=1}^{s_{n_l-2}} \frac{da_p^{(n_l)}}{dz_p^{(n_l)}} \cdot W_{p\leftarrow q}^{(n_l-1)} \cdot \frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}} \cdot W_{q\leftarrow k}^{(n_l-2)} \cdot \frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}} \cdot W_{k\leftarrow i}^{(n_l-3)} \cdot \frac{da_i^{(n_l-3)}}{dz_i^{(n_l-3)}} \cdot a_j^{(n_l-4)}$$

$$= \left(\sum_{q=1}^{s_{n_l-1}} \sum_{k=1}^{s_{n_l-2}} \frac{da_p^{(n_l)}}{dz_p^{(n_l)}} \cdot W_{p\leftarrow q}^{(n_l-1)} \cdot \frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}} \cdot W_{q\leftarrow k}^{(n_l-2)} \cdot \frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}} \cdot W_{k\leftarrow i}^{(n_l-3)} \right) \cdot \frac{da_i^{(n_l-3)}}{dz_i^{(n_l-3)}} \cdot a_j^{(n_l-4)}$$

此致，已经逐层全部展开，再用进行替换，进行缩进。注意！替换的都是误差对 z 的导数！z 是神经元激励函数的输入！

$$= \left(\frac{da_i^{(n_l-3)}}{dz_i^{(n_l-3)}} \cdot \sum_{k=1}^{s_{n_l-2}} W_{k\leftarrow i}^{(n_l-3)} \cdot \left(\frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}} \cdot \sum_{q=1}^{s_{n_l-1}} W_{q\leftarrow k}^{(n_l-2)} \cdot \left(\frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}} \cdot W_{p\leftarrow q}^{(n_l-1)} \cdot \left(\frac{da_p^{(n_l)}}{dz_p^{(n_l)}} \right) \right) \right) \right) \cdot a_j^{(n_l-4)}$$

$$= \left(\frac{da_i^{(n_l-3)}}{dz_i^{(n_l-3)}} \cdot \sum_{k=1}^{s_{n_l-2}} W_{k\leftarrow i}^{(n_l-3)} \cdot \frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}} \cdot \left(\sum_{q=1}^{s_{n_l-1}} W_{q\leftarrow k}^{(n_l-2)} \cdot \frac{da_q^{(n_l-1)}}{dz_q^{(n_l-1)}} \cdot \left(W_{p\leftarrow q}^{(n_l-1)} \cdot \delta_p^{(n_l)} \right) \right) \right) \cdot a_j^{(n_l-4)}$$

$$= \left(\frac{da_i^{(n_l-3)}}{dz_i^{(n_l-3)}} \cdot \sum_{k=1}^{s_{n_l-2}} W_{k\leftarrow i}^{(n_l-3)} \cdot \frac{da_k^{(n_l-2)}}{dz_k^{(n_l-2)}} \cdot \left(\sum_{q=1}^{s_{n_l-1}} W_{q\leftarrow k}^{(n_l-2)} \cdot \delta_q^{(n_l-1)} \right) \right) \cdot a_j^{(n_l-4)}$$

$$= \left(\frac{da_i^{(n_l-3)}}{dz_i^{(n_l-3)}} \cdot \sum_{k=1}^{s_{n_l-2}} W_{k\leftarrow i}^{(n_l-3)} \delta_k^{(n_l-2)} \right) \cdot a_j^{(n_l-4)}$$

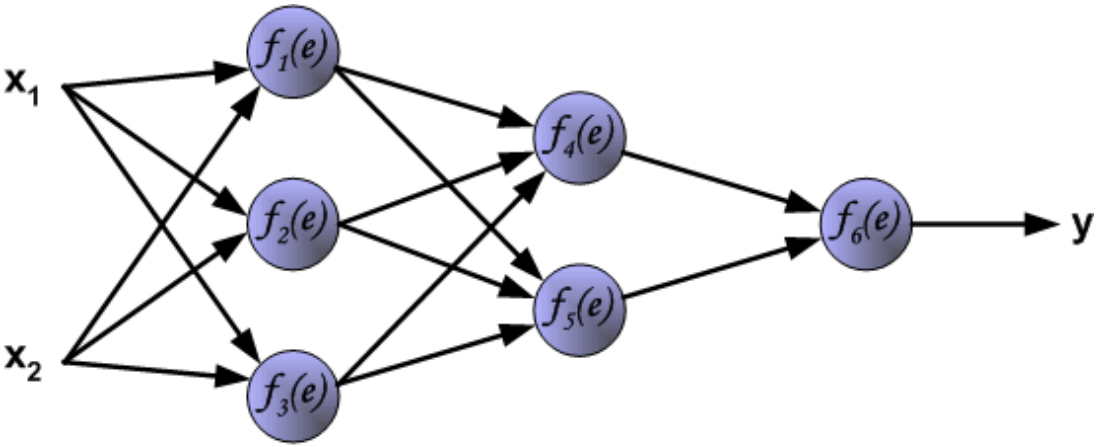
$$= \delta_i^{(n_l-3)} \cdot a_j^{(n_l-4)}$$

$$\delta_p^{(n_l)} = -\left(y - h_p(W,b;x,y)\right)\left(\frac{dh_p(W,b;x,y)}{dz_p^{(n_l)}}\right) == -\left(y - h_p(W,b;x,y)\right)\left(\frac{da_p^{(n_l)}}{dz_p^{(n_l)}}\right)$$

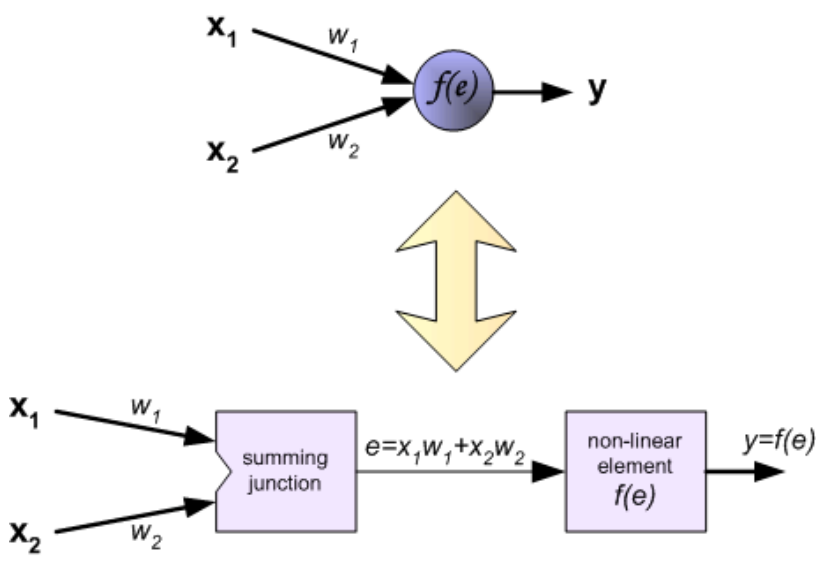
$$\delta_k^{(l)} = \left(\sum_{q=1}^{s_{l+1}} W_{q\leftarrow k}^{(l)} \cdot \delta_q^{(l+1)}\right) \cdot \frac{da_k^{(l)}}{dz_k^{(l)}}$$

$$\frac{\partial h_p(W,b;x,y)}{\partial W_{i\leftarrow j}^{(l)}} = \delta_i^{(l+1)} \cdot a_j^{(l)}$$

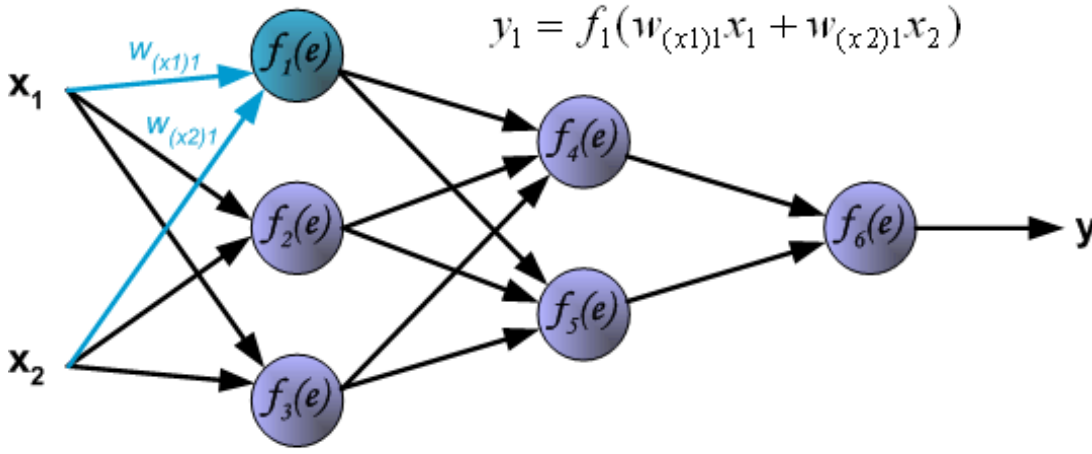
- 基于图形直观理解

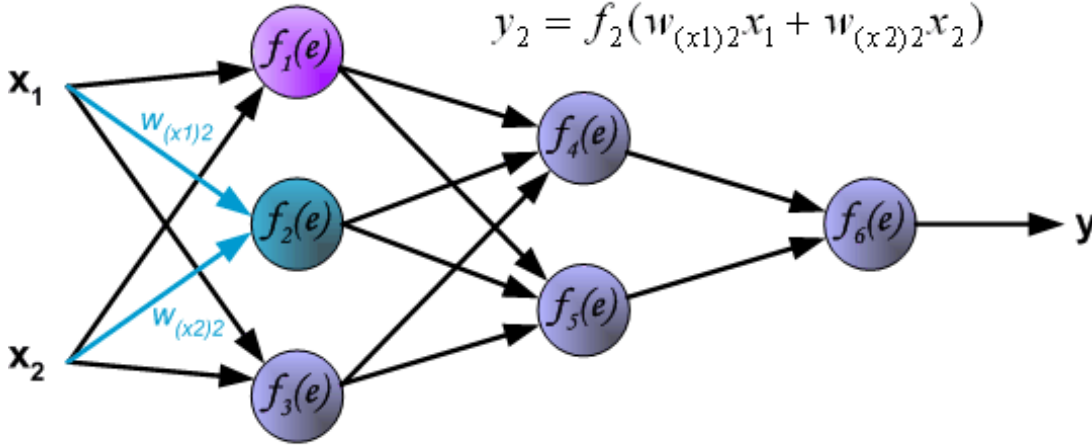


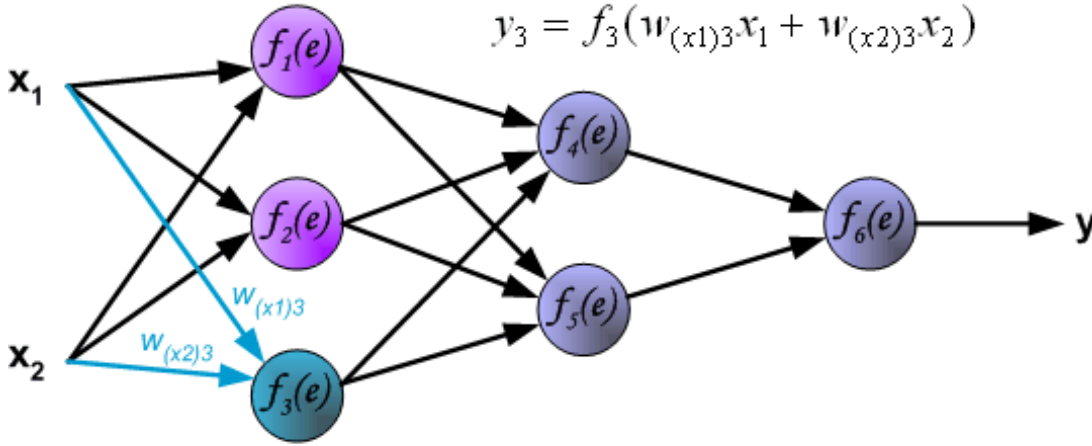
- 标记理解：

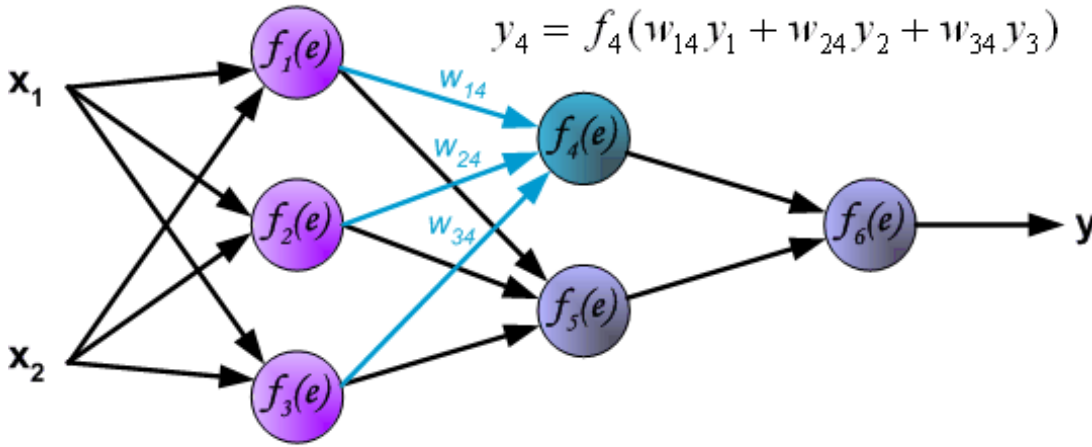

x1
w1
f(e)
y
x2
w2
x1
w1
summing
junction
e=x1w1+x2w2
non-linear
element
f(e)
y=f(e)
x2
w2

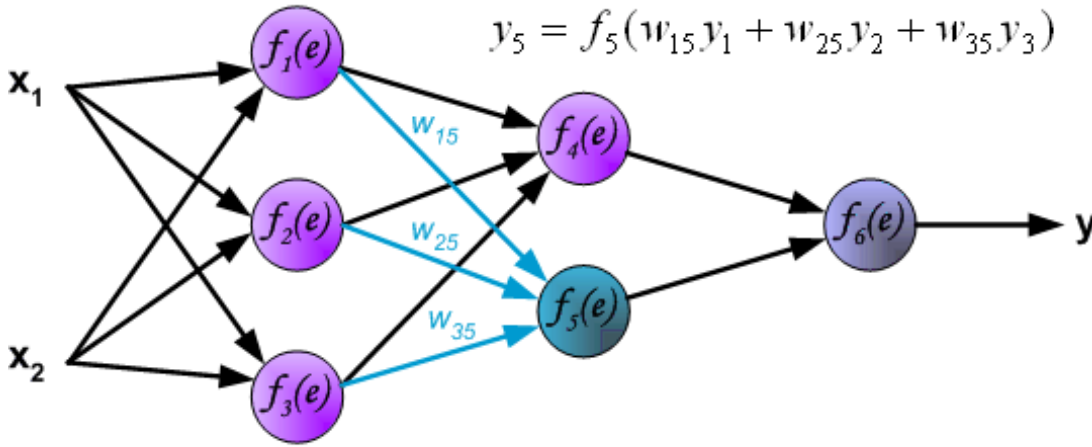
- 信号正向传播：

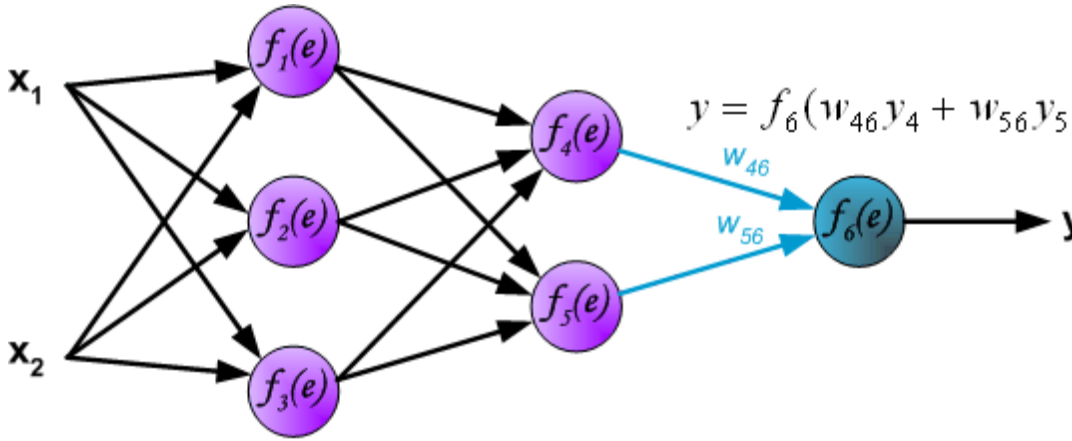



$y_2 = f_2(w_{(x1)2}x_1 + w_{(x2)2}x_2)$
x1
x2
W(x1)2
W(x2)2
f1(e)
f2(e)
f3(e)
f4(e)
f5(e)
f6(e)
y

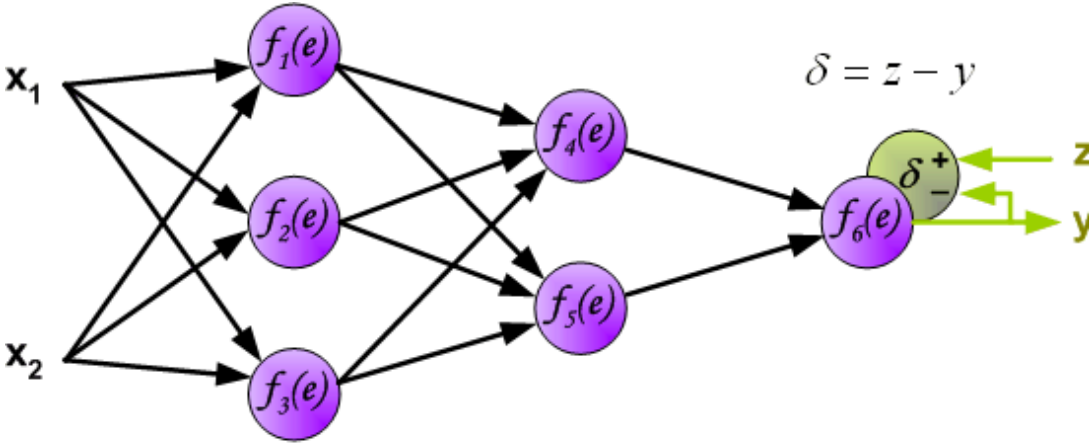

$y_3 = f_3(w_{(x1)3}x_1 + w_{(x2)3}x_2)$
x1
x2
f1(e)
f2(e)
f3(e)
f4(e)
f5(e)
f6(e)
y
W(x1)3
W(x2)3


$y_4 = f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3)$
x1
x2
f1(e)
f2(e)
f3(e)
w14
w24
w34
f4(e)
f5(e)
f6(e)
y

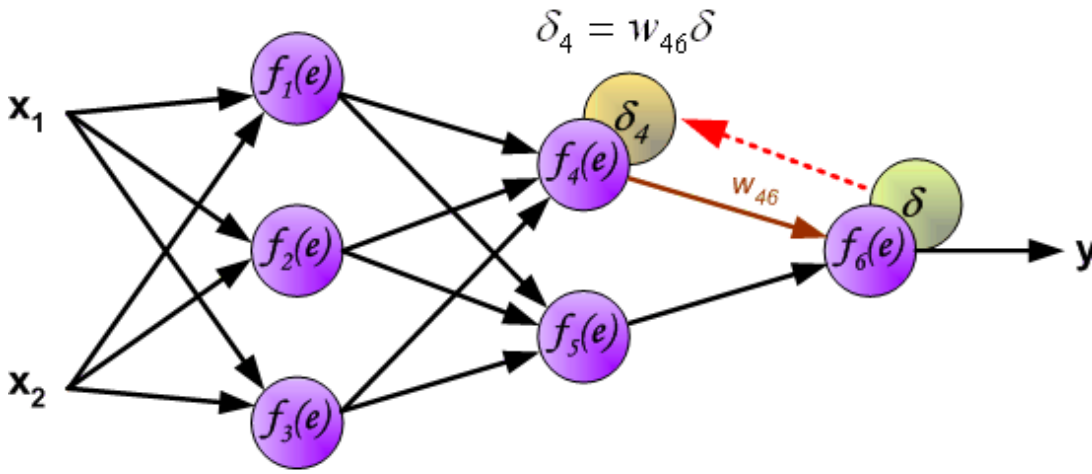

$y_5 = f_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3)$
x1
x2
f1(e)
f2(e)
f3(e)
f4(e)
f5(e)
f6(e)
w15
w25
w35
y

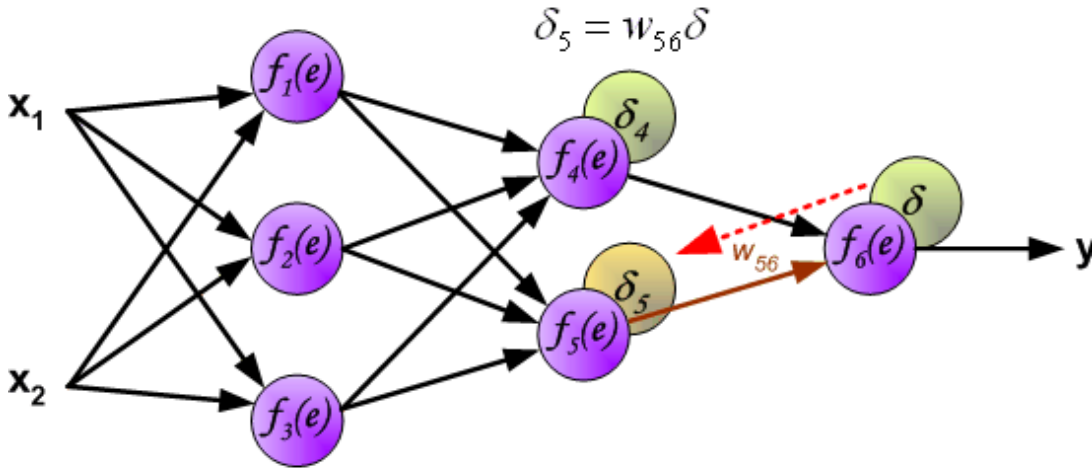

x1
x2
f1(e)
f2(e)
f3(e)
f4(e)
f5(e)
f6(e)
y = f6(w46y4 + w56y5)
w46
w56
y

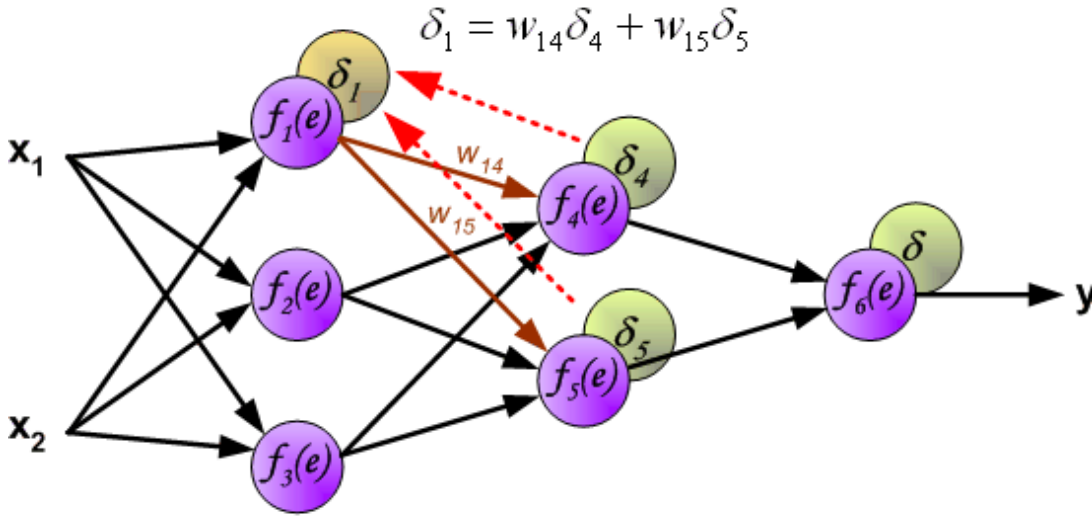
- 误差对激励函数输入求导：

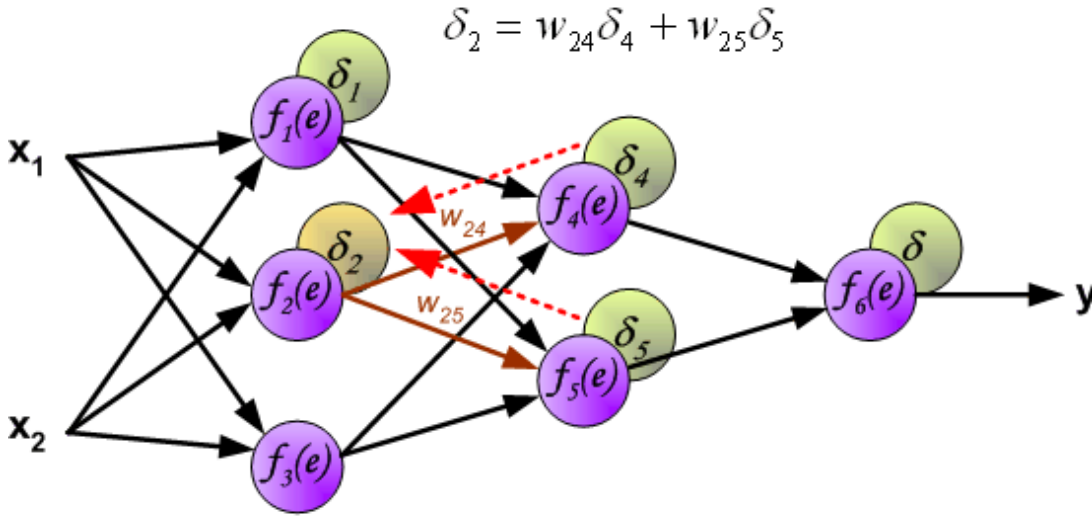


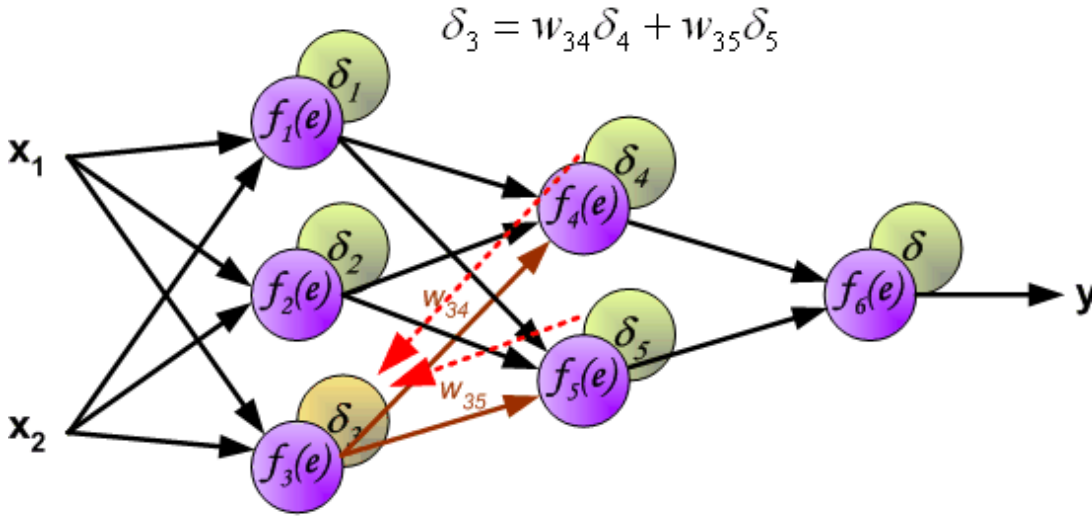
- 误差 (对激励函数输入导数) 反向传播：




$\delta_5 = w_{56}\delta$
x1
x2
f1(e)
f2(e)
f3(e)
f4(e)
f5(e)
f6(e)
δ4
δ5
δ
w56
y


$\delta_1 = w_{14}\delta_4 + w_{15}\delta_5$
δ_1
$f_1(e)$
w_{14}
w_{15}
δ_4
$f_4(e)$
x_1
$f_2(e)$
δ
$f_6(e)$
y
δ_5
$f_5(e)$
x_2
$f_3(e)$

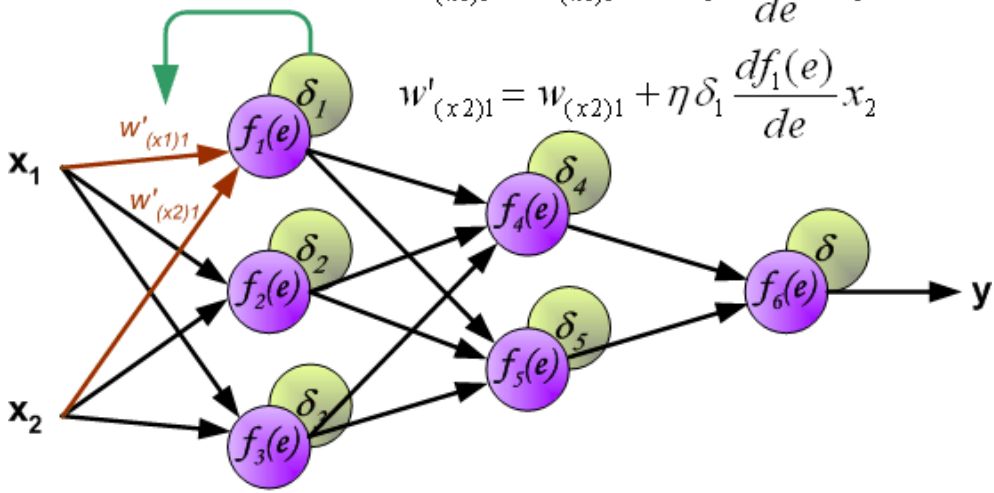

$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$
x1
x2
δ_1
$f_1(e)$
δ_2
$f_2(e)$
$f_3(e)$
δ_4
$f_4(e)$
δ_5
$f_5(e)$
δ
$f_6(e)$
y
w_{24}
w_{25}


$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5$
x1
x2
f1(e)
f2(e)
f3(e)
f4(e)
f5(e)
f6(e)
δ1
δ2
δ3
δ4
δ5
δ
w34
w35
y

- 基于 GD 的权重更新：

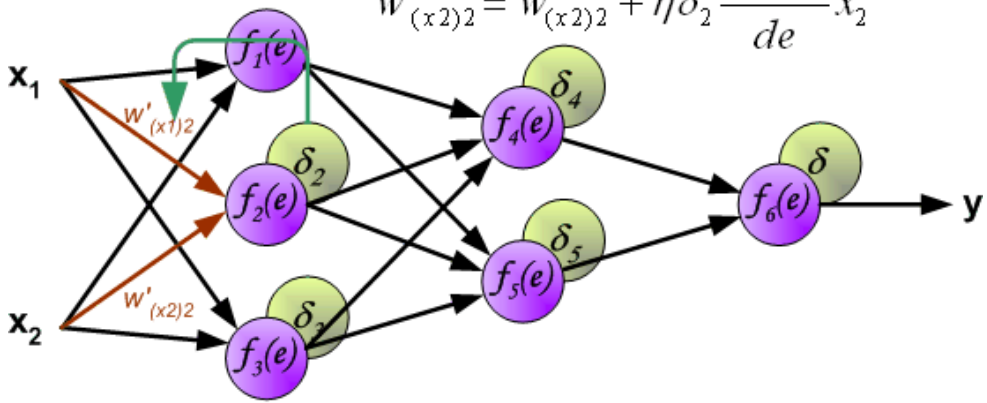
$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$

$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$



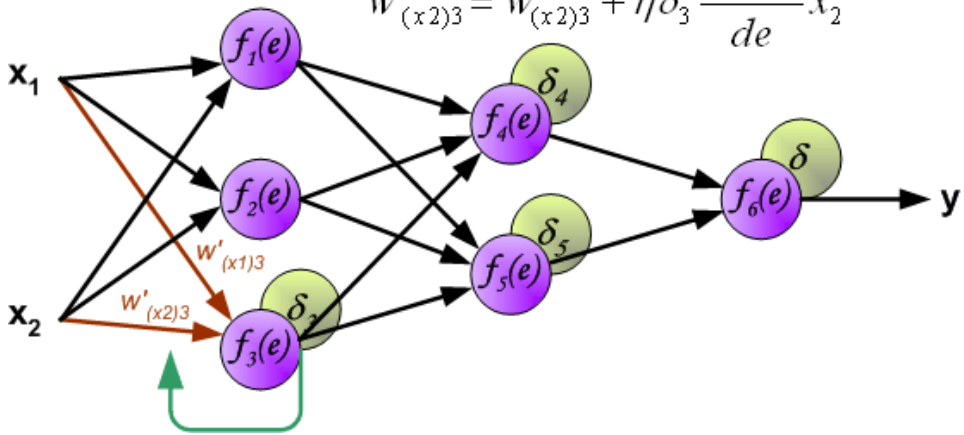
$$w'_{(x1)2} = w_{(x1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$



$$w'_{(x1)3} = w_{(x1)3} + \eta\delta_3 \frac{df_3(e)}{de} x_1$$

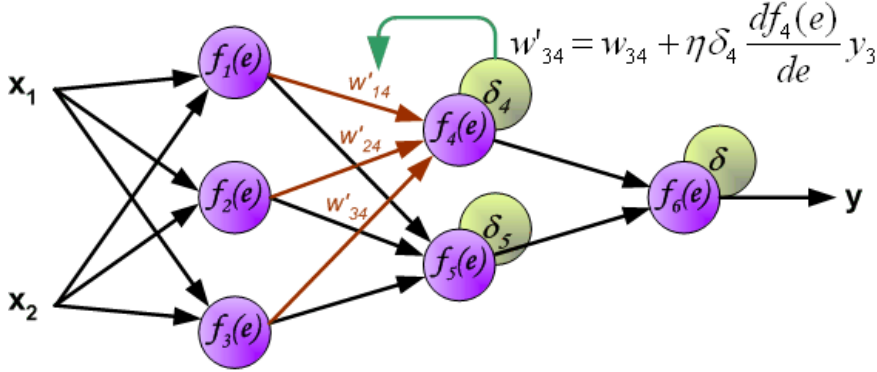
$$w'_{(x2)3} = w_{(x2)3} + \eta\delta_3 \frac{df_3(e)}{de} x_2$$



$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2$$

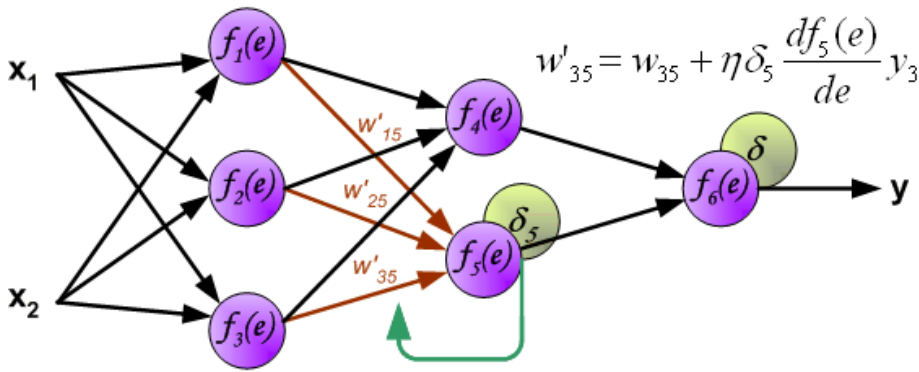
$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$



$$w'_{15} = w_{15} + \eta\delta_5 \frac{df_5(e)}{de} y_1$$

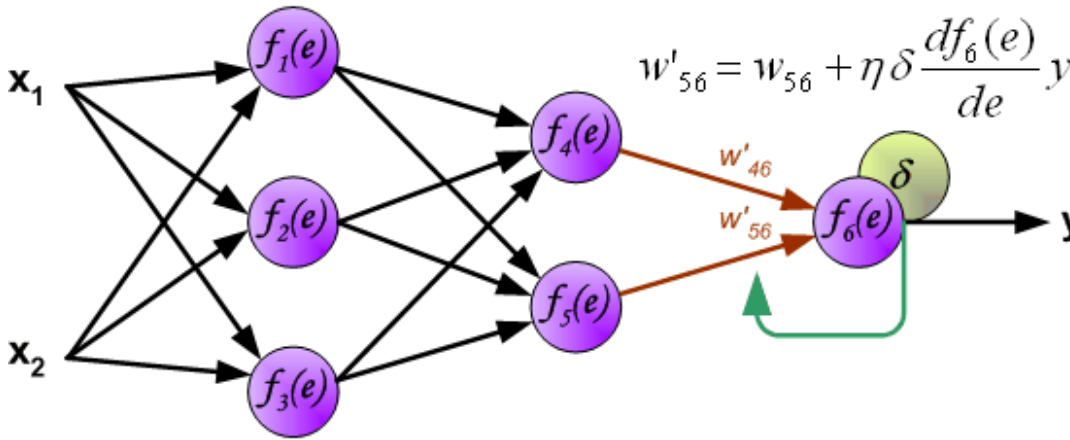
$$w'_{25} = w_{25} + \eta\delta_5 \frac{df_5(e)}{de} y_2$$

$$w'_{35} = w_{35} + \eta\delta_5 \frac{df_5(e)}{de} y_3$$

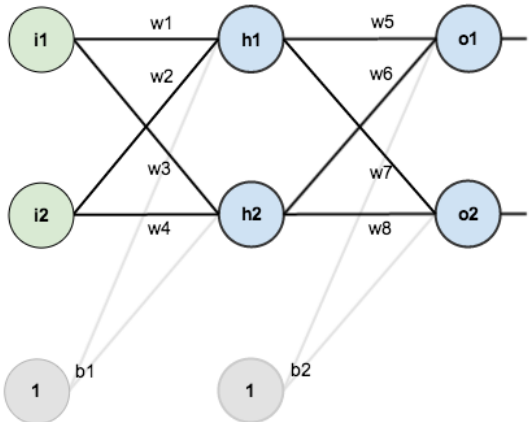


$$w'_{46} = w_{46} + \eta\,\delta\,\frac{df_6(e)}{de}\,y_4$$

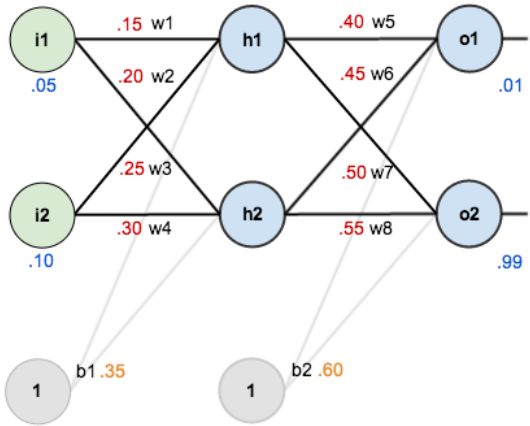
$$w'_{56} = w_{56} + \eta\,\delta\,\frac{df_6(e)}{de}\,y_5$$



- BP 手动计算



- 随机初始化后的网络


i1
.05
i2
.10
h1
h2
o1
.01
o2
.99
.15 w1
.20 w2
.25 w3
.30 w4
.40 w5
.45 w6
.50 w7
.55 w8
1
b1 .35
1
b2 .60

- 信号正向传播

  隐含层输入：

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

  隐含层输出：

$$out_{h1} = \frac{1}{1 + e^{-net_{h1}}} = \frac{1}{1 + e^{-0.3775}} = 0.593269992$$

$$out_{h2} = 0.596884378$$

输出层输入：

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

输出层输出：

$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}} = \frac{1}{1 + e^{-1.105905967}} = 0.75136507$$

$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}} = \frac{1}{1 + e^{-1.105905967}} = 0.75136507$$

$$out_{o2} = 0.772928465$$

误差计算：

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

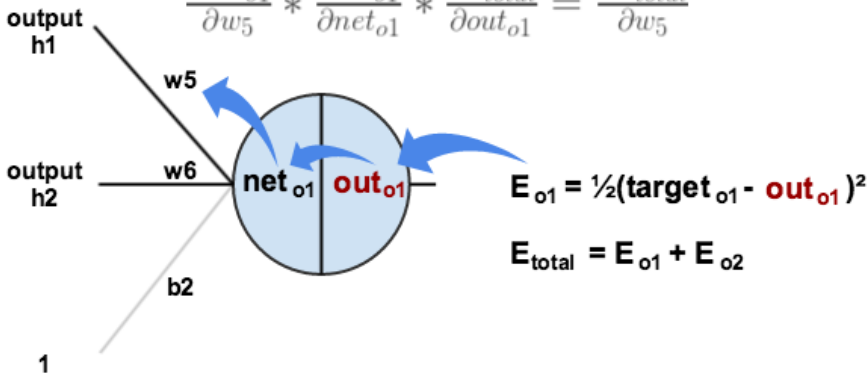
$$E_{o2} = 0.023560026$$

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

BP 反向传播公式：

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



总体误差计算：

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

误差对输出层输出导数：

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

误差对隐含层输出导数：

$$out_{o1} = \frac{1}{1 + e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

误差对输出层输入权重导数：

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

误差对输出层输入权重导数计算公式回顾：

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

$$\delta_{o1} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = \frac{\partial E_{total}}{\partial net_{o1}}$$

$$\delta_{o1} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1})$$

$$\frac{\partial E_{total}}{\partial w_5} = \delta_{o1} out_{h1}$$

输出层输入权重更新：

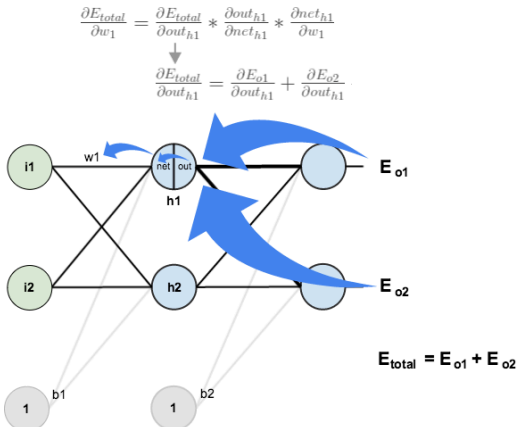
$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

同理，可计算其他输出层输入权重：

$$w_6^+ = 0.408666186\, w_7^+ = 0.511301270\, w_8^+ = 0.561370121$$

隐藏层输入权重公式，分成 3 块：

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$



第一块：误差对隐藏层输出导数：

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}}$$

$$\frac{\partial E_{o1}}{\partial net_{o1}} = \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} = 0.74136507 * 0.186815602 = 0.138498562$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial out_{h1}} = w_5 = 0.40$$

$$\frac{\partial E_{o1}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial out_{h1}} = 0.138498562 * 0.40 = 0.055399425$$

$$\frac{\partial E_{o2}}{\partial out_{h1}} = -0.019049119$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}} = 0.055399425 + -0.019049119 = 0.036350306$$

第二块：隐藏层输出对输入导数：

$$out_{h1} = \frac{1}{1 + e^{-net_{h1}}}$$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.241300709$$

第三块：隐藏层输入对输入权重导数：

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

三块合并，误差对隐藏层输入权重导数：

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

误差对隐藏层输入权重导数公式回顾：

$$\frac{\partial E_{total}}{\partial w_1} = (\sum_o \frac{\partial E_{total}}{\partial out_o} * \frac{\partial out_o}{\partial net_o} * \frac{\partial net_o}{\partial out_{h1}}) * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial w_1} = (\sum_o \delta_o * w_{ho}) * out_{h1}(1 - out_{h1}) * i_1$$

$$\frac{\partial E_{total}}{\partial w_1} = \delta_{h1} i_1$$

隐藏层输入权重更新：

$$w_1^+ = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

同理，可计算其他隐藏层输入权重：

$$w_2^+ = 0.19956143 \, w_3^+ = 0.24975114 \, w_4^+ = 0.29950229$$

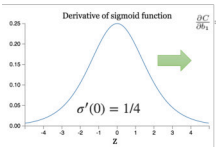
- BP 小结

1. BP 是经验风险最小理论的 GD 求解过程
2. BP 反传的是误差对激活函数的输入/输出求导
3. 误差对权重的更新，需要激活函数的导数
4. BP 参数包括：损失函数，激活函数，递归算法 (包括学习率)

- 梯度消失 Vanishing Gradient

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)\, w_2\sigma'(z_2)\, w_3\sigma'(z_3)\, w_4\sigma'(z_4)\, \frac{\partial C}{\partial a_4}$$



$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \overbrace{w_2\sigma'(z_2)}^{<\frac{1}{4}} \overbrace{w_3\sigma'(z_3)}^{<\frac{1}{4}} \underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}_{\text{common terms}}$$

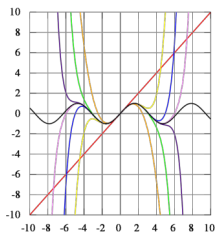
$$\frac{\partial C}{\partial b_3} = \sigma'(z_3) \underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}_{\text{common terms}}$$

4 数学上直观理解三大网络

4.1 卷积神经网络 CNN 的数学类比

- 近似任意函数的泰勒展开

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$



CNN 拥有泰勒展开的能力

- 历史上的泰勒网络 [Raul Rojas, 1996, “Neural Networks”]

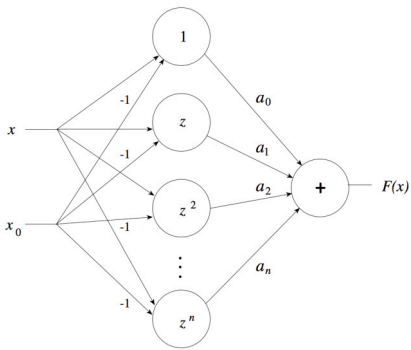
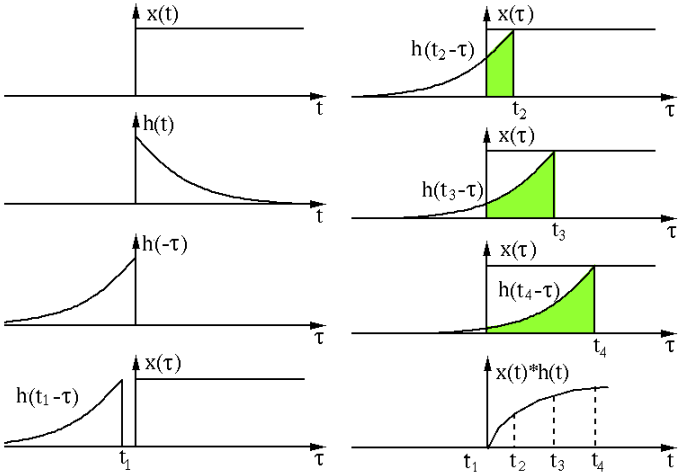


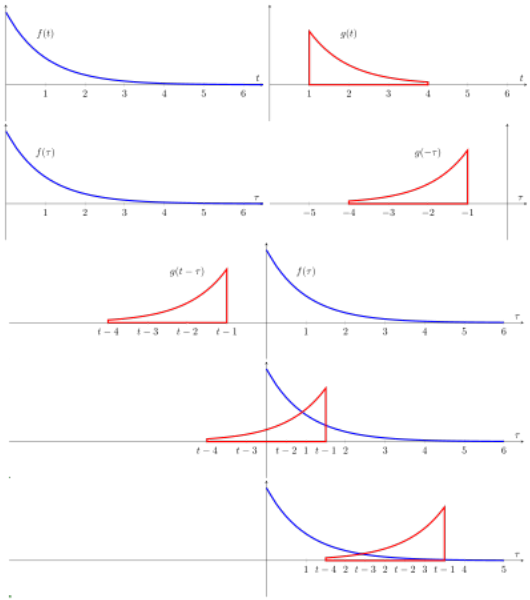
Fig. 1.16. A Taylor network

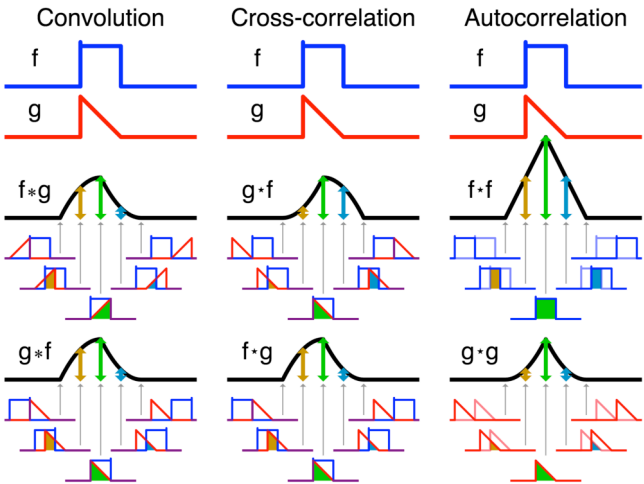
很明显这种浅层网络难以逐层学习，难以有表示学习的优势！

- 卷积的多种理解

1. 选定区域的积分
2. 强制幅度的积分
3. 反向的互相关

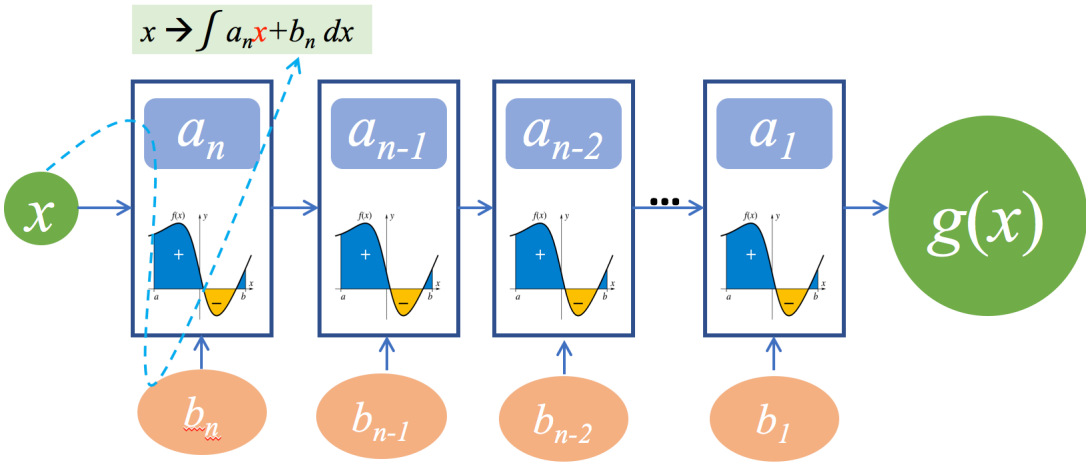


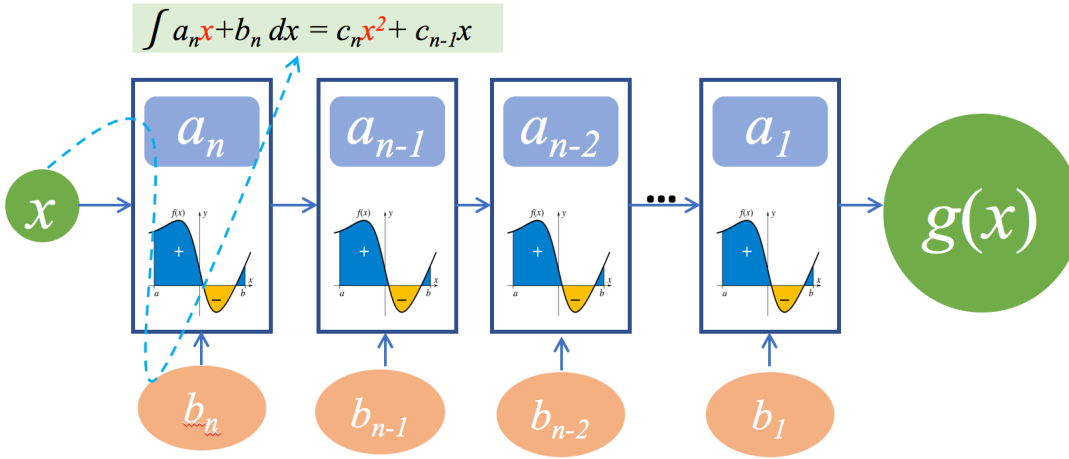

f(t)
t
1 2 3 4 5 6
g(t)
t
1 2 3 4 5 6
f(τ)
τ
1 2 3 4 5 6
g(−τ)
τ
−5 −4 −3 −2 −1
g(t − τ)
t − 4 t − 3 t − 2 t − 1
f(τ)
τ
1 2 3 4 5 6
τ
t − 4 t − 3 t − 2 1 t − 1 2 3 4 5 6
τ
1 t − 4 2 t − 3 2 t − 2 3 t − 1 4 5

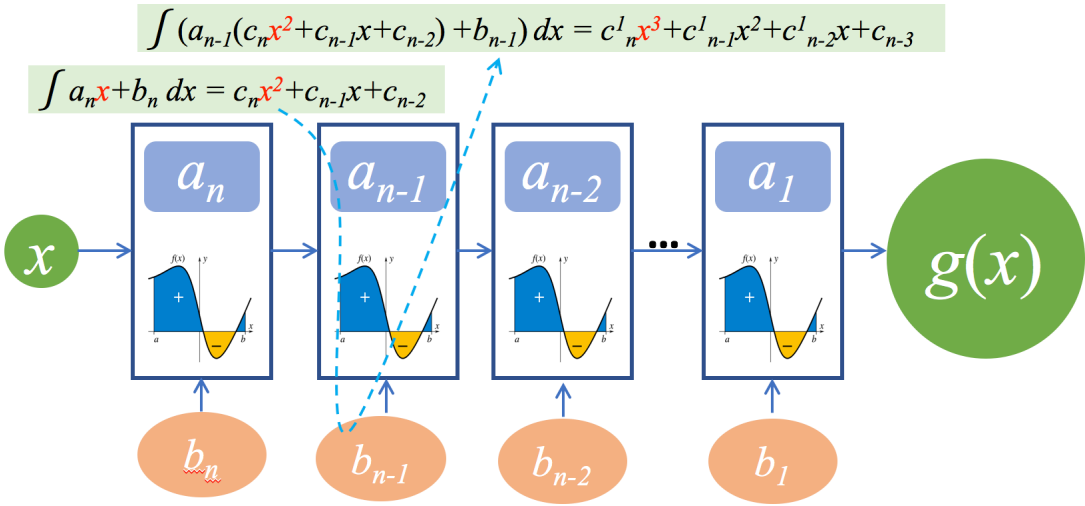

Convolution
Cross-correlation
Autocorrelation
f
g
f∗g
g⋆f
f⋆f
g∗f
f⋆g
g⋆g

卷积可以实现积分！

- 深度卷积网络看成深度积分网络

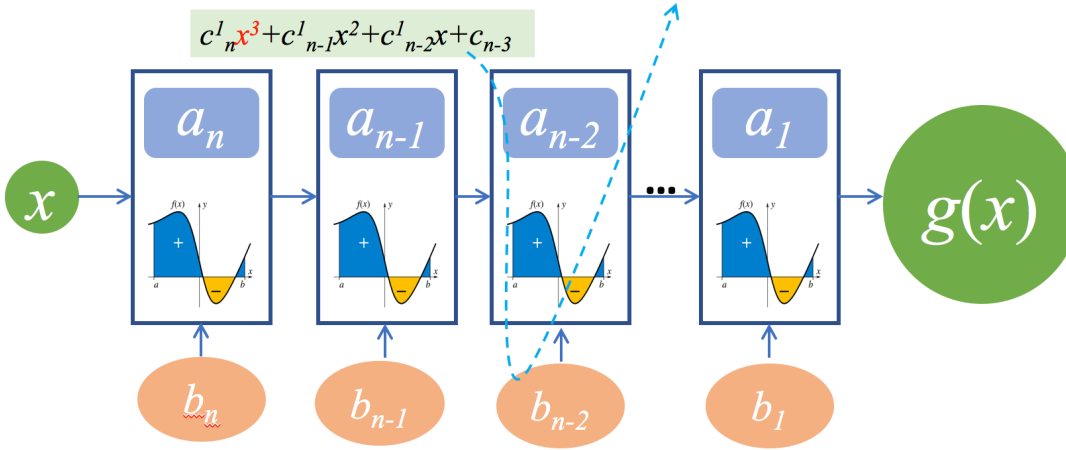



$\int a_n x + b_n \, dx = c_n x^2 + c_{n-1} x$
x
a_n
a_{n-1}
a_{n-2}
a_1
$g(x)$
b_n
b_{n-1}
b_{n-2}
b_1


$\int (a_{n-1}(c_n x^2+c_{n-1}x+c_{n-2})+b_{n-1})\,dx = c^1{}_n x^3+c^1{}_{n-1}x^2+c^1{}_{n-2}x+c_{n-3}$
$\int a_n x+b_n\,dx = c_n x^2+c_{n-1}x+c_{n-2}$
x
a_n
a_{n-1}
a_{n-2}
a_1
$g(x)$
b_n
b_{n-1}
b_{n-2}
b_1

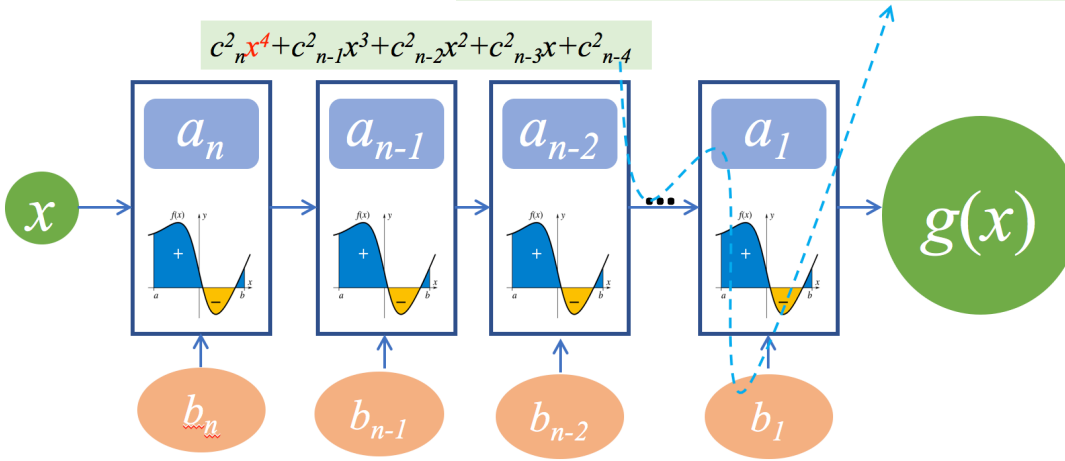
$$\int (a_{n-2}(c^1_n x^3+c^1_{n-1}x^2+c^1_{n-2}x+c_{n-2})+b_{n-2})\,dx = c^2_n x^4+c^2_{n-1}x^3+c^2_{n-2}x^2+c^2_{n-3}x+c^2_{n-4}$$

$$c^1_n x^3+c^1_{n-1}x^2+c^1_{n-2}x+c_{n-3}$$

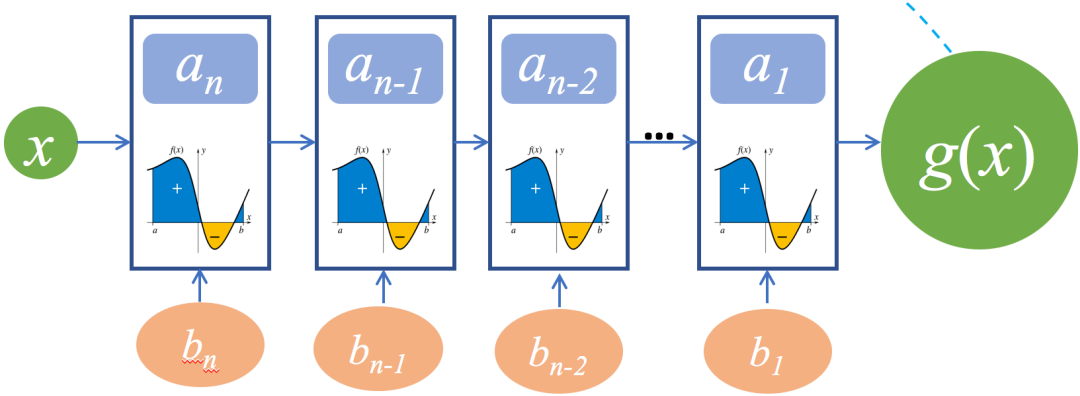


$$\int (a_1(c^{n-2}{}_n x^{n-1}+c^{n-2}{}_{n-1}x^{n-2}+\ldots+c^{n-2}{}_1)+b_1)\,dx$$
$$= c^{n-1}{}_n x^n+c^{n-1}{}_{n-1}x^{n-1}+c^{n-1}{}_{n-2}x^{n-2}+\ \ +c^{n-1}{}_1 x\ +c^{n-1}{}_0$$

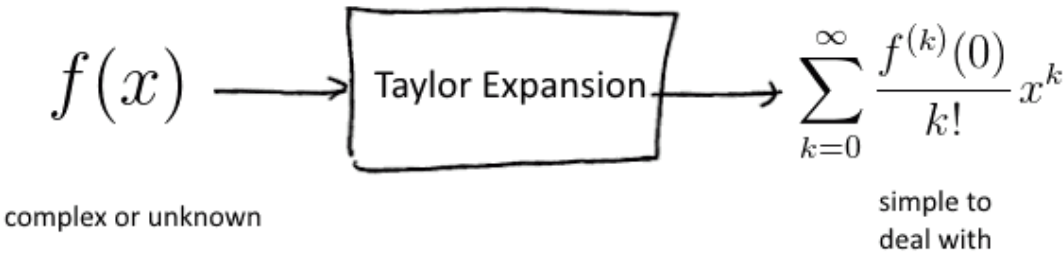
$$c^2{}_n x^4+c^2{}_{n-1}x^3+c^2{}_{n-2}x^2+c^2{}_{n-3}x+c^2{}_{n-4}$$

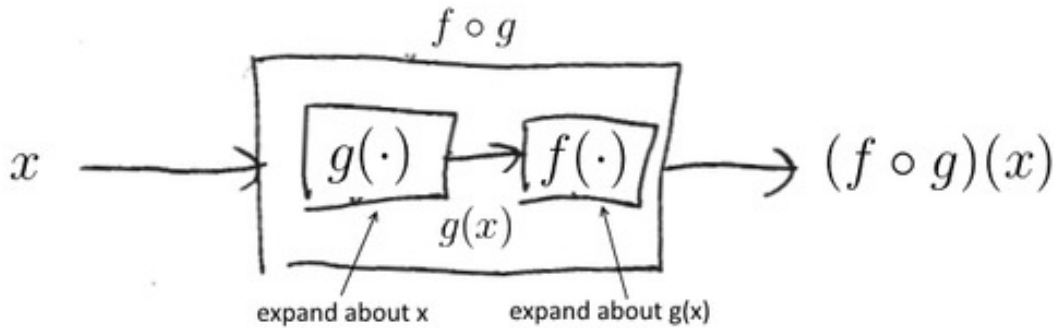


$$g(x)=c^{n-1}{}_n x^n+c^{n-1}{}_{n-1}x^{n-1}+c^{n-1}{}_{n-2}x^{n-2}+\ +c^{n-1}{}_1 x+c^{n-1}{}_0$$



- 未知函数的卷积网络


$f(x)$
Taylor Expansion
$\sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!} x^k$
complex or unknown
simple to deal with


$f \circ g$
x
$g(\cdot)$
$f(\cdot)$
$g(x)$
$(f \circ g)(x)$
expand about x
expand about g(x)

- 未知函数的泰勒展开

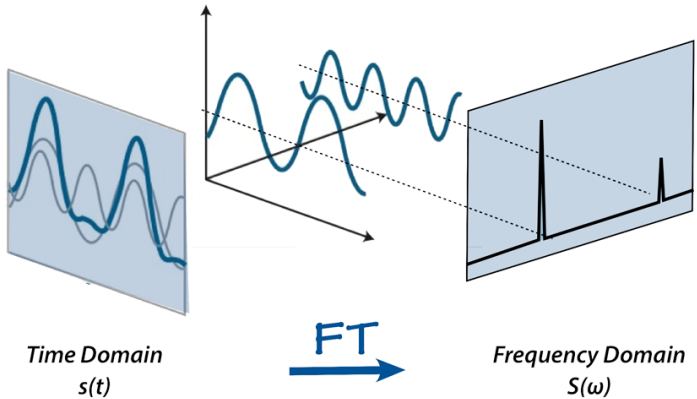
$$f(x)=\sum_{n=0}^{\infty}\frac{f^{(n)}(0)}{n!}x^n$$

$$=f(0)+f'(0)x+\frac{f''(0)}{2!}x^2+\frac{f'''(0)}{3!}x^3+\cdots$$

- 深度卷积网络具有泰勒展开的能力
  1. 模拟任意能力
  2. 高维波动信号在输入端，低维平稳信号在输出端
  3. 越深越好

4.2 递归神经网络 RNN 的数学类比

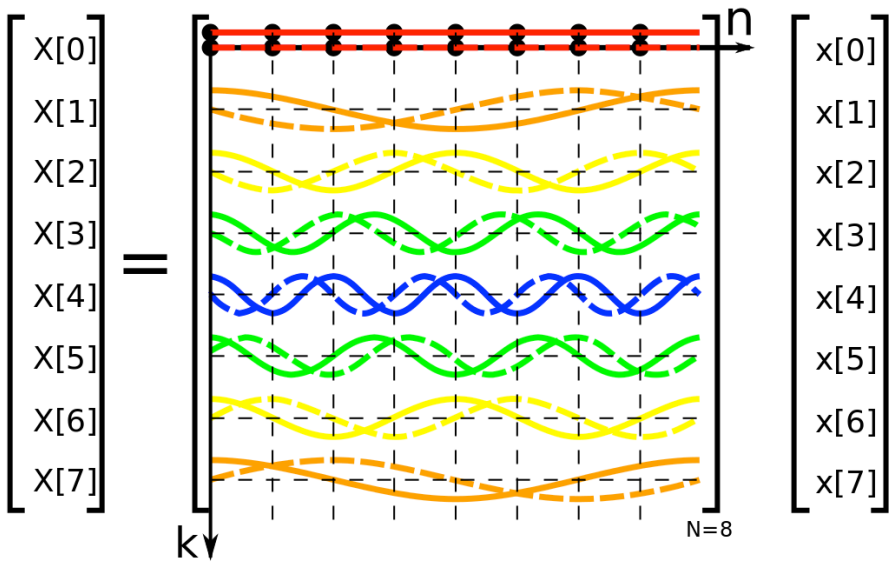
- 频谱化时间信号的傅里叶变化



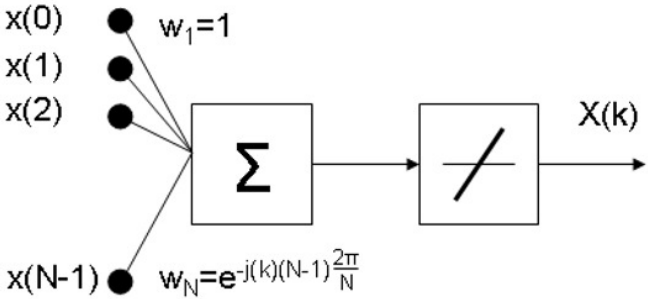
$$W = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix},$$

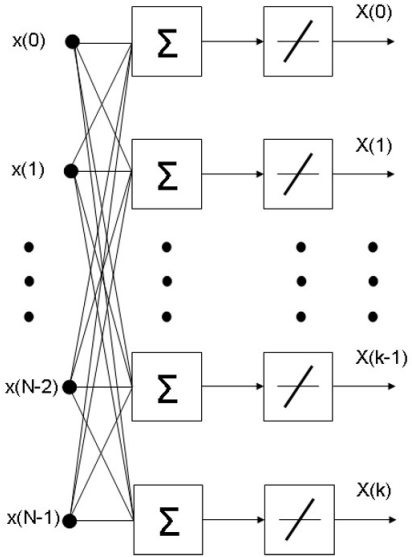
$$X = Wx\,\omega = e^{-2\pi i/N}$$

$$\begin{aligned} X_k &= \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N} \\ &= \sum_{n=0}^{N-1} x_n \cdot [\cos(2\pi kn/N) - i \cdot \sin(2\pi kn/N)], \end{aligned}$$

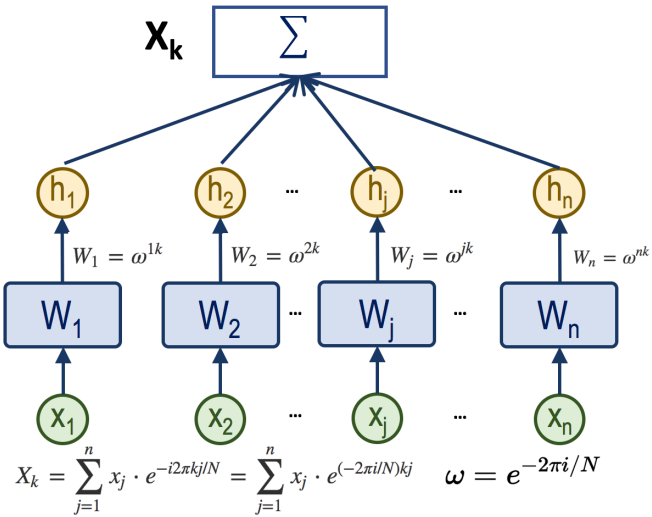


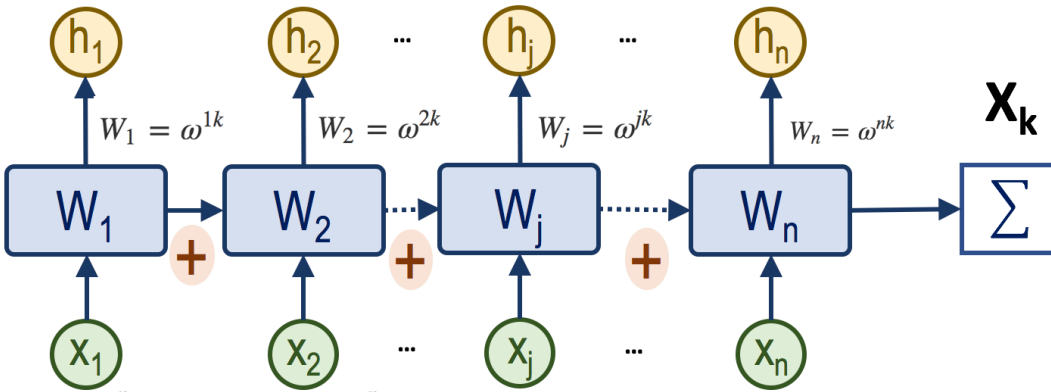
Rosemarie Velik, “Discrete Fourier Transform Computation Using Neural Networks”, 2008


x(0)
x(1)
x(2)
x(N-1)
w1=1
wN=e-j(k)(N-1)2π/N
Σ
X(k)

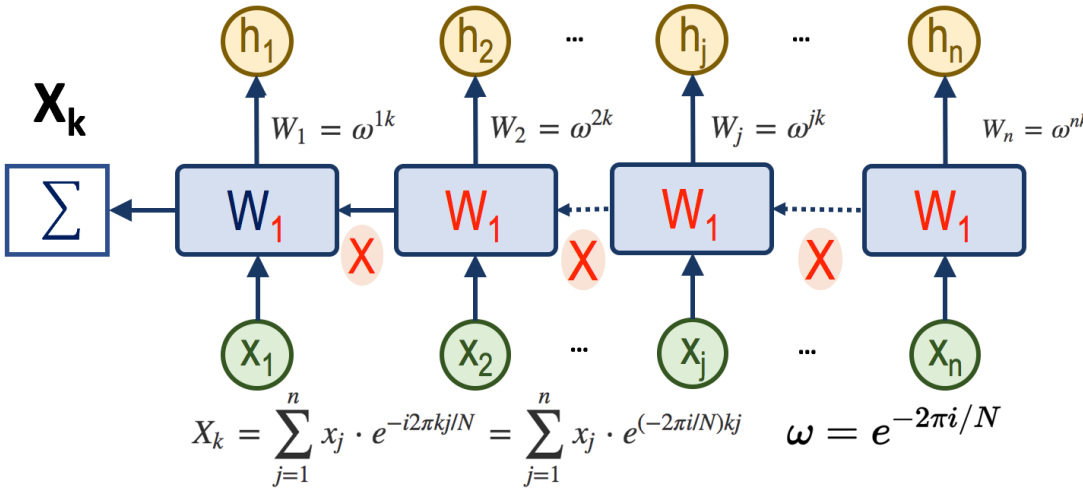

x(0)
x(1)
x(N-2)
x(N-1)
Σ
Σ
Σ
Σ
X(0)
X(1)
X(k-1)
X(k)

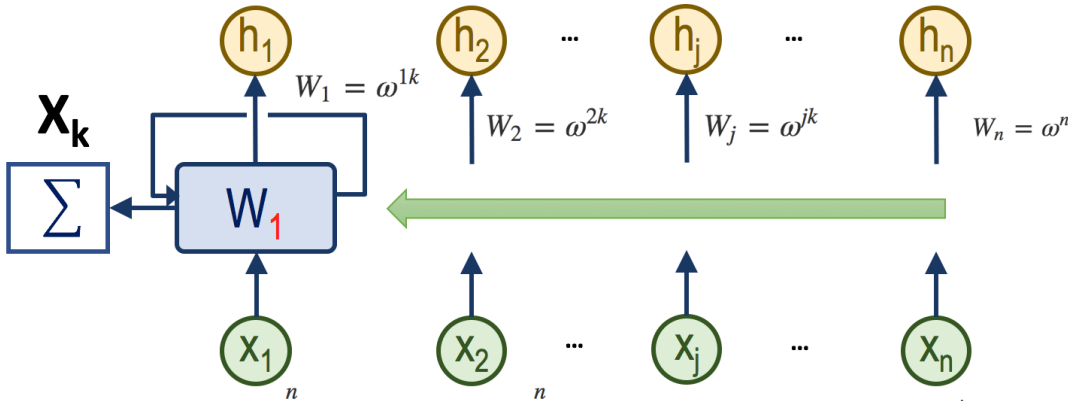
- 单个频域信号的生成 k

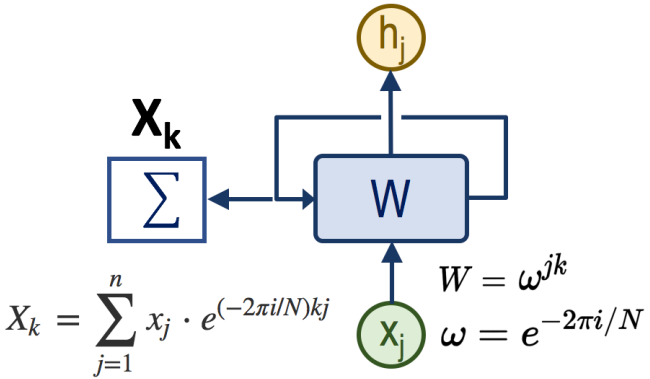

X_k
$\sum$
h_1
h_2
h_j
h_n
$W_1 = \omega^{1k}$
$W_2 = \omega^{2k}$
$W_j = \omega^{jk}$
$W_n = \omega^{nk}$
W_1
W_2
W_j
W_n
x_1
x_2
x_j
x_n
$X_k = \sum_{j=1}^{n} x_j \cdot e^{-i2\pi kj/N} = \sum_{j=1}^{n} x_j \cdot e^{(-2\pi i l/N)kj}$
$\omega = e^{-2\pi i/N}$

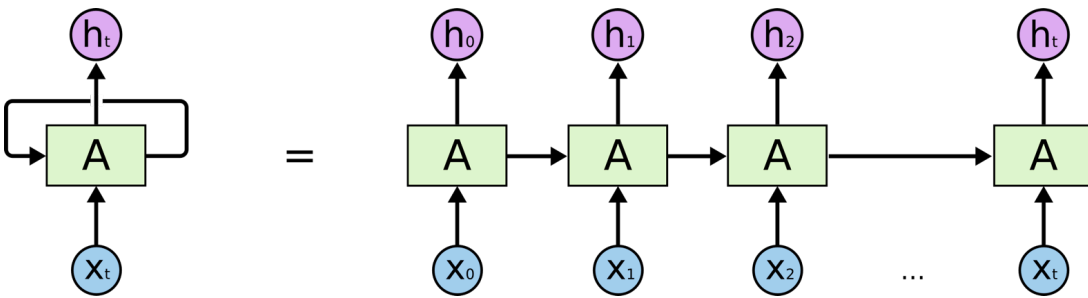


$$X_k = \sum_{j=1}^{n} x_j \cdot e^{-i2\pi kj/N} = \sum_{j=1}^{n} x_j \cdot e^{(-2\pi i/N)kj} \qquad \omega = e^{-2\pi i/N}$$


X_k
h_1
h_2
...
h_j
...
h_n
$W_1 = \omega^{1k}$
$W_2 = \omega^{2k}$
$W_j = \omega^{jk}$
$W_n = \omega^{nk}$
$\sum$
W_1
W_1
W_1
W_1
X
X
X
x_1
x_2
...
x_j
...
x_n
$X_k = \sum_{j=1}^{n} x_j \cdot e^{-i2\pi kj/N} = \sum_{j=1}^{n} x_j \cdot e^{(-2\pi i/N)kj}$
$\omega = e^{-2\pi i/N}$


h1
h2
...
hj
...
hn
$W_1 = \omega^{1k}$
$W_2 = \omega^{2k}$
$W_j = \omega^{jk}$
$W_n = \omega^{nk}$
Xk
∑
W1
x1
x2
...
xj
...
xn
$X_k = \sum_{j=1}^{n} x_j \cdot e^{-i2\pi kj/N} = \sum_{j=1}^{n} x_j \cdot e^{(-2\pi i/N)kj}$
$\omega = e^{-2\pi i/N}$


h_j
X_k
$\sum$
W
$W = \omega^{jk}$
$X_k = \sum_{j=1}^{n} x_j \cdot e^{(-2\pi i/N)kj}$
x_j
$\omega = e^{-2\pi i/N}$


h_t
A
x_t
=
h_0
h_1
h_2
h_t
A
A
A
A
x_0
x_1
x_2
...
x_t

- 历史上的傅里叶网络 [Raul Rojas, 1996, “Neural Networks”]

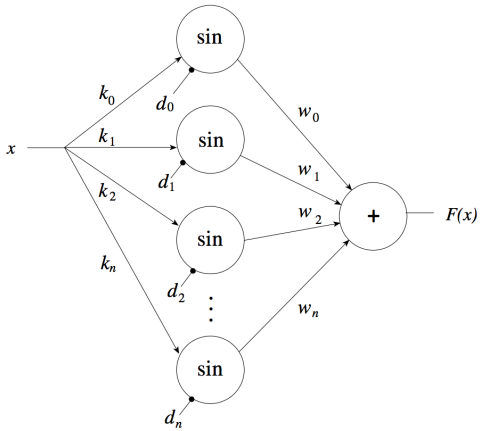


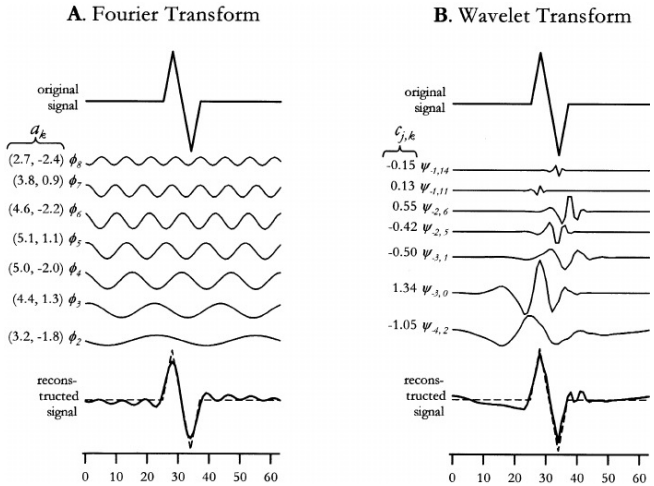
Fig. 1.17. A Fourier network

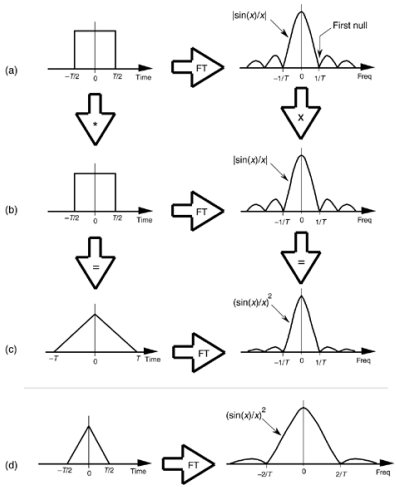
很明显这种浅层网络难以逐层学习，难以有表示学习的优势！

- 深度递归网络具有傅里叶变化的能力

1. 频谱化波动信号
2. 高维频谱等价长时迭代，低维频谱等价短时迭代
3. 越深越好

- 卷积网络和小波变换

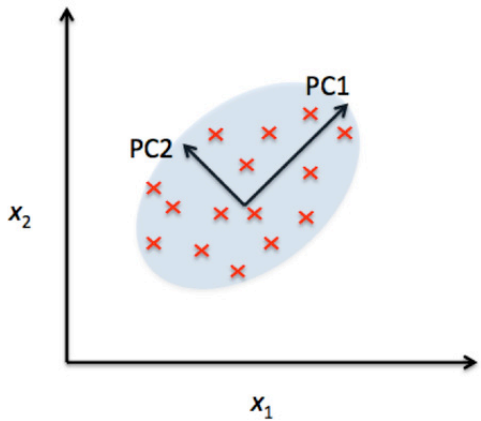

A. Fourier Transform
original signal
a_k
(2.7, -2.4) ϕ_8
(3.8, 0.9) ϕ_7
(4.6, -2.2) ϕ_6
(5.1, 1.1) ϕ_5
(5.0, -2.0) ϕ_4
(4.4, 1.3) ϕ_3
(3.2, -1.8) ϕ_2
recons-tructed signal
0 10 20 30 40 50 60
B. Wavelet Transform
original signal
$c_{j,k}$
-0.15 $\psi_{-1,14}$
0.13 $\psi_{-1,11}$
0.55 $\psi_{-2,6}$
-0.42 $\psi_{-2,5}$
-0.50 $\psi_{-3,1}$
1.34 $\psi_{-3,0}$
-1.05 $\psi_{-4,2}$
recons-tructed signal
0 10 20 30 40 50 60



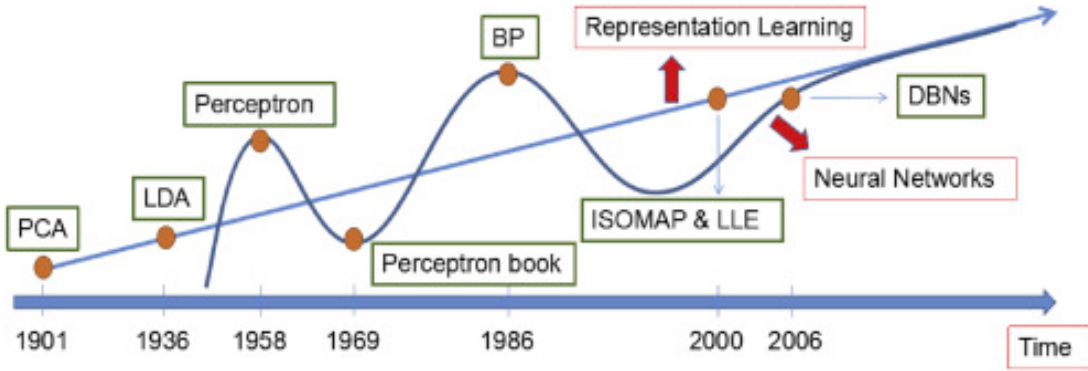
卷积还可以看成是小波变换

4.3 自动编码器 AE 的数学类比

- 主成分分析 PCA



- 为什么要理解 PCA：表征学习

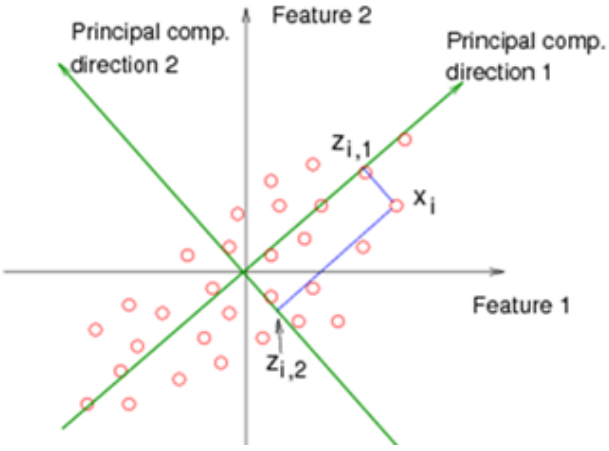


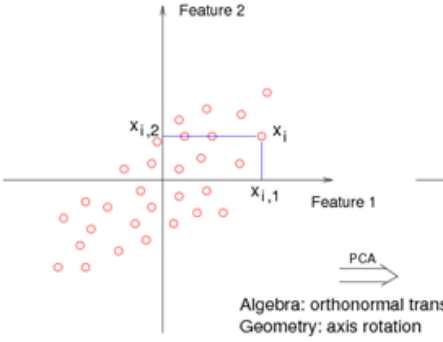
- PCA 的 4 种理解

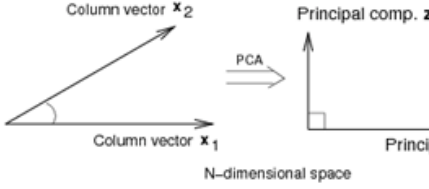
1. 最大方差投影
2. 最小重建误差
3. 高斯先验误差
4. 线性流形对齐

这 4 种理解，是逐层递进深入的。尤其第 4 种理解，基于流行的理解，是深度学习祖师 Hinton 的原创 (1997)，并且收录到 Goodfellow 的《深度学习》的书里。为 t-SNE(t-Distributed Stochastic Neighbor Embedding, 2008) 算法的发明打下了基石。

- PCA 的最大方差投影理解







投影：$\mathbf{w}^T\boldsymbol{x}$

方差：$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^T\boldsymbol{x}_i)^2 = \mathbf{w}^T S\mathbf{w}$

$$S = \frac{1}{n}\sum_i \boldsymbol{x}_i\boldsymbol{x}_i^T$$

拉格朗日乘数法：

$$L = \mathbf{w}^T S\mathbf{w} + \lambda(1 - \mathbf{w}^T\mathbf{w})$$

$$\frac{\partial L}{\partial \mathbf{w}} = 2S\mathbf{w} - 2\lambda\mathbf{w}$$

$$S\mathbf{w} = \lambda\mathbf{w}$$

最大方差：

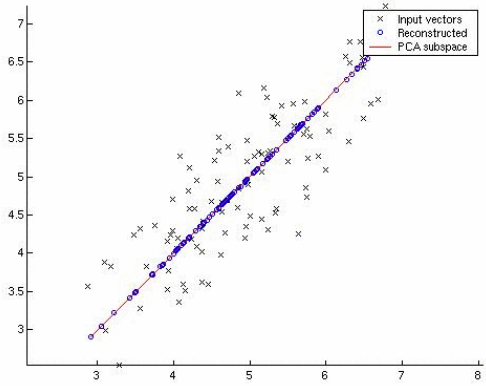
$$\max_{\mathbf{w}} \quad \mathbf{w}^T S\mathbf{w}$$
$$s.t. \quad \|\mathbf{w}\| = 1$$

方差：

$$\mathbf{w}^T S\mathbf{w} = \mathbf{w}^T\lambda\mathbf{w} = \lambda$$

理解，PCA 中，矩阵特征值和特征向量的由来！

- PCA 的最小重建误差理解



基于特征值的认知后，可以舍弃最小特征值，从而重建数据。

- PCA 的最小重建误差数学建模

正交基：$\mathbf{u}_1, \cdots, \mathbf{u}_D$

原始数据：$\boldsymbol{x}_i = \sum_{j=1}^{D} \alpha_{ij} \mathbf{u}_j$

基坐标：$\alpha_{ij} = \mathbf{u}_j^T \boldsymbol{x}_i$

降维重建：$\hat{\boldsymbol{x}}_i = \sum_{j=1}^{d} \alpha_{ij} \mathbf{u}_j$

减一维重建：

$$d = D - 1$$

对应最小特征值：

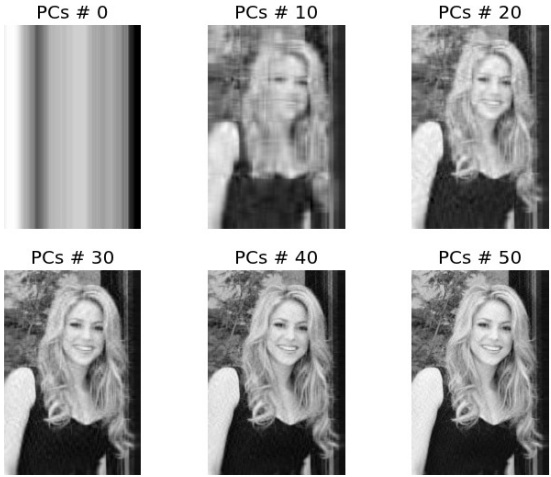
$$\mathbf{w}_D^T S \mathbf{w}_D = \lambda_D$$

- PCA 的最小重建误差数学推理

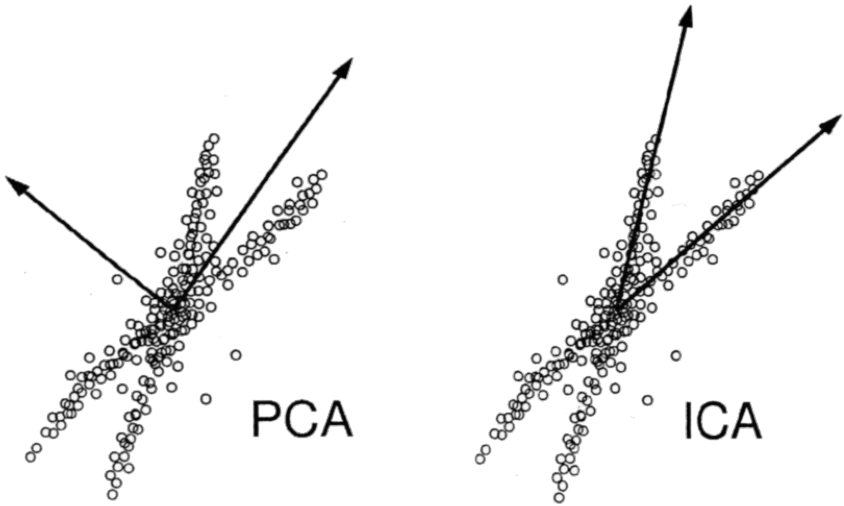
$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i-\hat{\boldsymbol{x}}_i\|^2 &= \frac{1}{n}\sum_{i=1}^{n}\|\sum_{j=1}^{D}\alpha_{ij}\mathbf{u}_j-\sum_{j=1}^{d}\alpha_{ij}\mathbf{u}_j\|^2\\ &= \frac{1}{n}\sum_{i=1}^{n}\|\sum_{j=d+1}^{D}\alpha_{ij}\mathbf{u}_j\|^2\\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=d+1}^{D}\alpha_{ij}^2\\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=d+1}^{D}\mathbf{u}_j^T\boldsymbol{x}_i\boldsymbol{x}_i^T\mathbf{u}_j\\ &= \sum_{j=d+1}^{D}\mathbf{u}_j^T S\mathbf{u}_j \quad \text{等价方差最小}\end{aligned}$$

理解，PCA 中，数据压缩的由来！

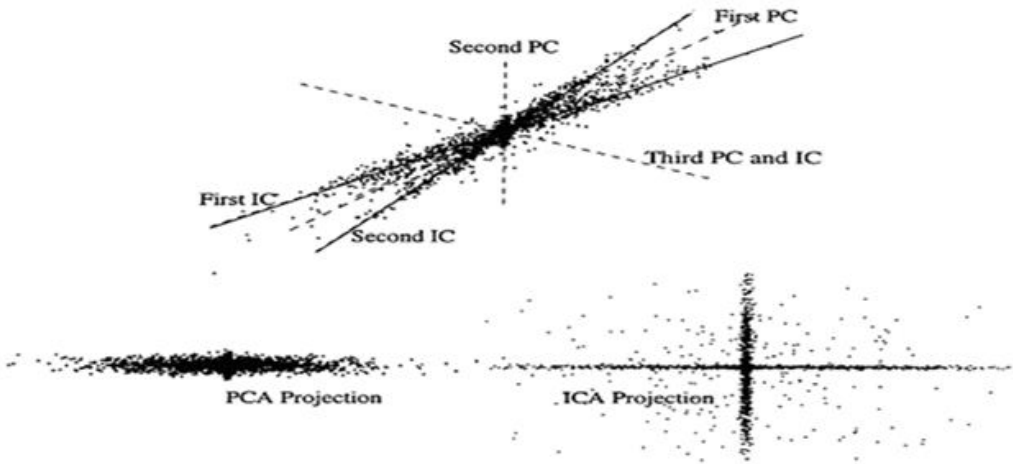
数据重建的思想使得 PCA 常常被用来做图像特征提取！



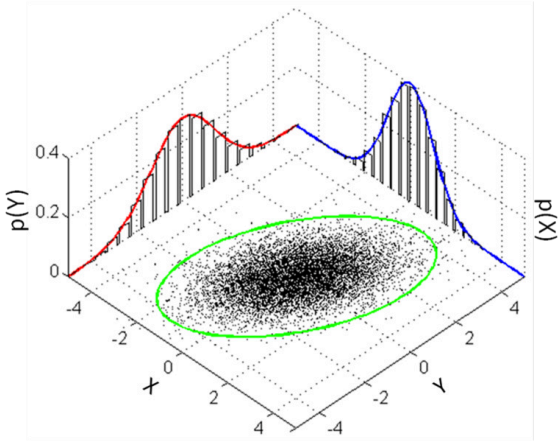
这种认知也决定了 PCA 和其他方法，譬如 ICA 的区别！


PCA
ICA

- PCA 和 ICA 的不同投影



- PCA 的高斯先验误差理解



基于重建的思想要求最小均方误差！最小均方误差，等价于高斯分布先验下的最大似然估计 MLE。

- 高斯先验的 MLE 等价于最小二乘

假设$X_1,...,X_n$ $Y_1,...,Y_n$ 满足如下：

$$y_i = \alpha + \beta x_i + \epsilon_i \Rightarrow \epsilon_i = y_i - (\alpha + \beta x_i)$$ 其中误差满足正态分布：$\varepsilon_i \sim N(0, \sigma^2)$

那么根据**MLE**，我们得到：

$$L(\alpha, \beta, \sigma^2 | \mathbf{y}, \mathbf{x}) = \frac{1}{\sqrt{(2\pi\sigma^2)^n}} \exp - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - (\alpha + \beta x_i))^2$$

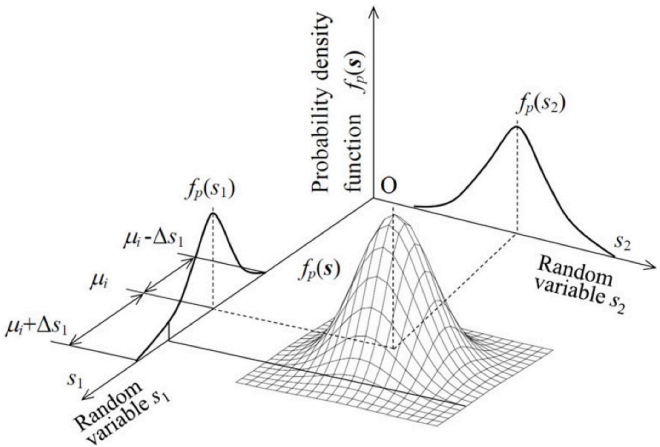
$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{\epsilon_1^2}{2\sigma^2} \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{\epsilon_2^2}{2\sigma^2} \cdots \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{\epsilon_n^2}{2\sigma^2} = \frac{1}{\sqrt{(2\pi\sigma^2)^n}} \exp - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \epsilon_i^2$$

求最大值 $\ln L(\alpha, \beta, \sigma^2 | \mathbf{y}, \mathbf{x}) = -\frac{n}{2}(\ln 2\pi + \ln \sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - (\alpha + \beta x_i))^2$ 求最小值

$$SS(\alpha.\beta) = \sum_{i=1}^{n} (y_i - (\alpha + \beta x_i))^2$$ ← 这个刚好是**LSE**的表达式

因此看成满足高斯先验的数据，在最大似然估计下的重建！

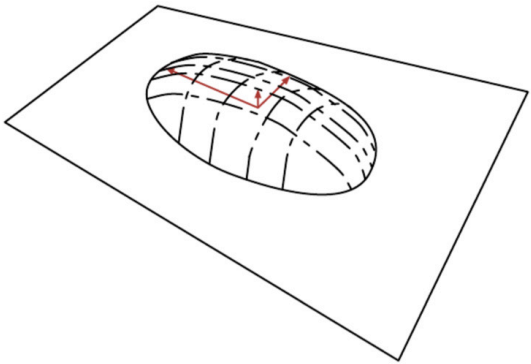
- PCA 的线性流形对齐理解



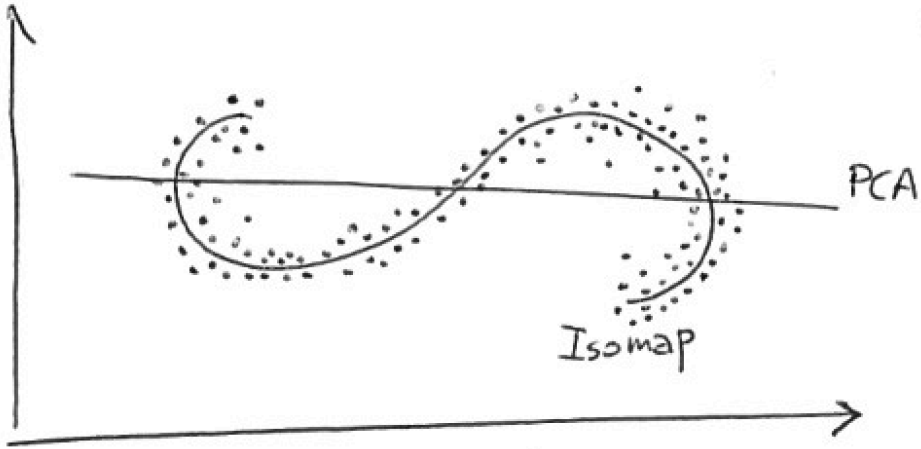
如果把高斯分布的思想，应用到数据的边缘分布满足高斯分布，然后求线性流形。

– 辛顿的烙饼说 Pancake

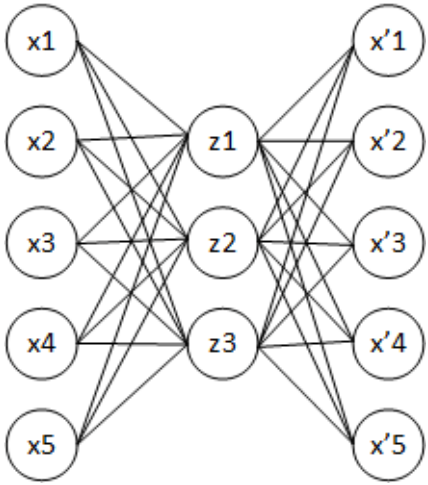
$$\frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i-\hat{\boldsymbol{x}}_i\|^2$$



– 线性 VS 非线性



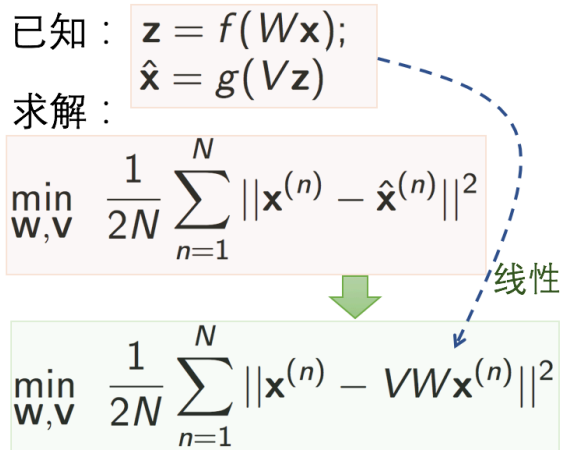
– 自动编码器 AE 的 PCA 理解



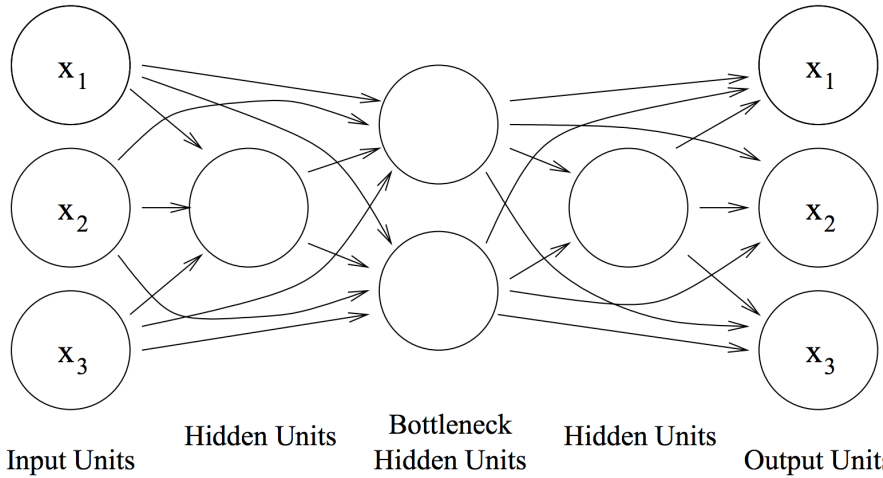
Encoder: $z = f(Wx)$

Decoder:$\hat{x} = g(Vz)$

– 线性假设下的 AE


已知：
$\mathbf{z} = f(W\mathbf{x})$;
$\hat{\mathbf{x}} = g(V\mathbf{z})$
求解：
$\min_{\mathbf{W},\mathbf{V}} \frac{1}{2N} \sum_{n=1}^{N} ||\mathbf{x}^{(n)} - \hat{\mathbf{x}}^{(n)}||^2$
线性
$\min_{\mathbf{W},\mathbf{V}} \frac{1}{2N} \sum_{n=1}^{N} ||\mathbf{x}^{(n)} - VW\mathbf{x}^{(n)}||^2$

– 自动编码器的非线性



当隐藏层变得更深或者非线性激活函数！

5 TensorFlow 入门

5.1 各大深度学习平台简介

5.2 TensorFlow 基础

感谢 Stanford, CMU, MIT 等网上公开课程和资料！

AI2ML