生物信息学- 第3章，人工智能导论

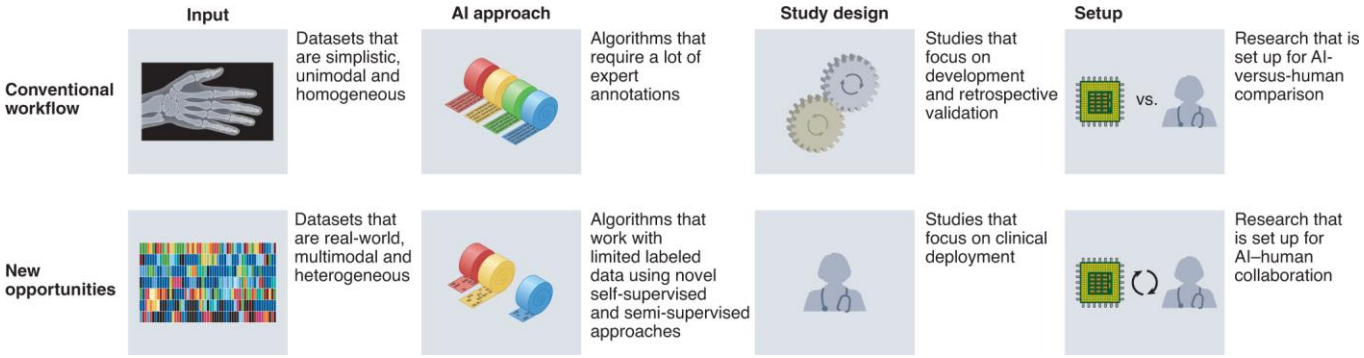
吕晖

2024年

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

人工智能技术

人工智能涉及计算机算法的开发，以执行通常与人类智能相关的任务。人工智能包括但不限于机器学习、表征学习、深度学习和自然语言处理。无论具体技术如何，这些技术在医学上的总体目标是使用计算机算法从数据中发现相关信息并协助临床决策。

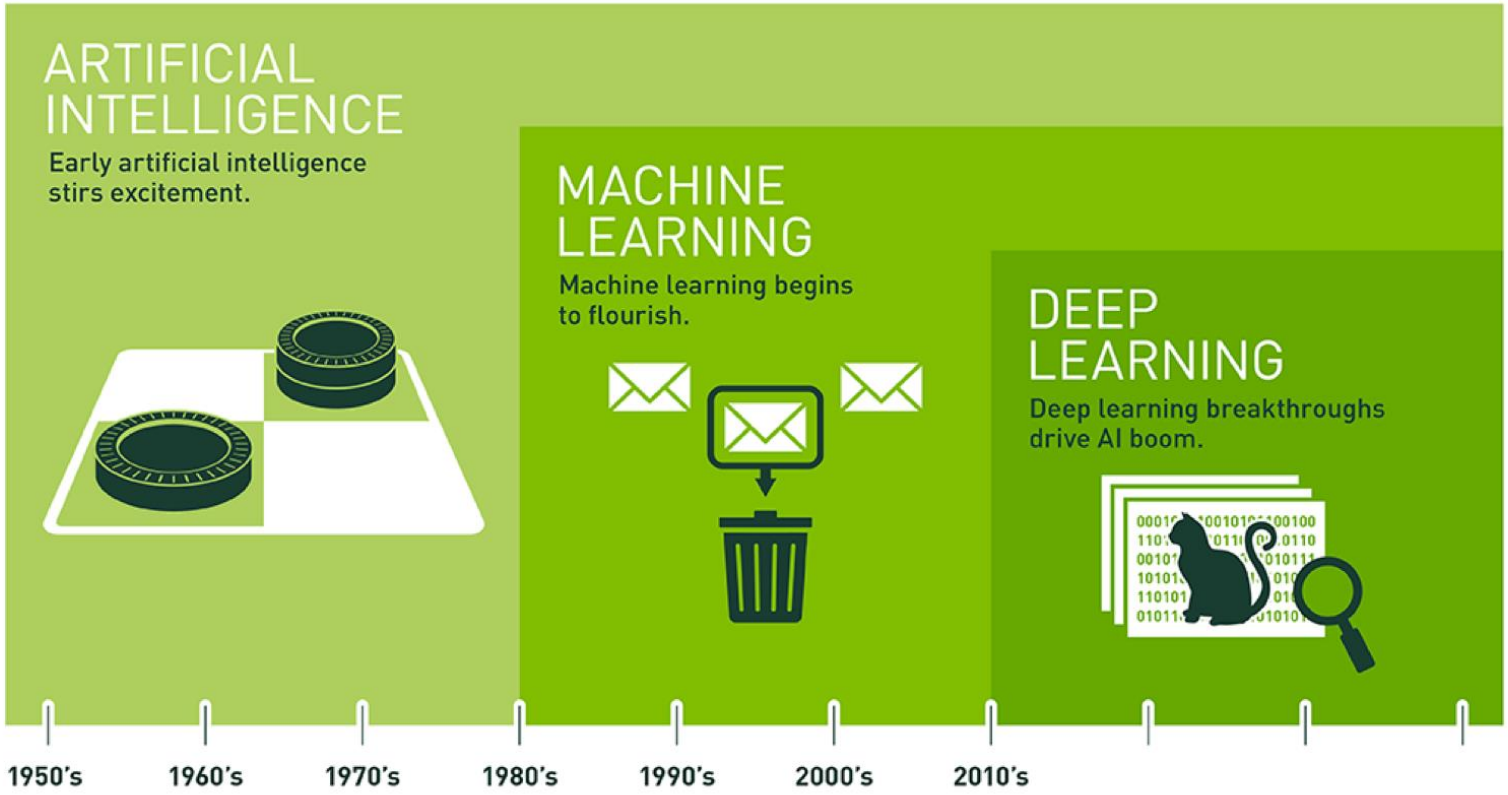


Rajpurkar, P., Chen, E., Banerjee, O., & Topol, E. J. (2022). AI in health and medicine. Nature medicine, 28(1), 31-38.

AI 简介

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Artificial Intelligence



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

Intelligence 智能

What is intelligence?

Wikipedia：Intelligence has been defined in many ways to include the capacity for logic, understanding, self-awareness, learning, emotional knowledge, reasoning, planning, creativity, and problem solving. It can be more generally described as the ability to perceive or infer information, and to retain it as knowledge to be applied towards adaptive behaviors within an environment or context.

（包括：逻辑，理解，自我意识，学习，情感，溯源，计划，创造，问题解决）

脑机接口

自动驾驶

机器人

AI生成

什么是人工智能 Artificial Intelligence?

- 对照智能的定义，在某种程度上接近或者超过人的能力
- 常见情景：预测，判断，语言，下棋，推理，交流，数学推导，… …
- 如何把智能活动数学化

语言与智能的关系

人工智能历史

标志性事件：1956年的 Dartmouth 会议，十人左右参加，历时两个月。

当时创立的计算机系通常分三个领域：理论，系统，AI。

AI领域显示度较高的项目：下棋。从1957年开始，每十年都预言十年内机器能下赢国际象棋世界冠军。一直等了40年，到1997才真正实现。

- 量变到质变：1995年卡斯帕罗夫批评计算机下棋没有悟性/直觉(Insights)；1997年失败的时候还在猜是不是计算机作弊（有人类象棋大师群在背后集体决策）
- 从这里可以推论悟性/直觉指的是什么

机器会思考吗?

乔姆斯基回答 “潜艇会游泳吗？”

思考题

- 在chatGPT 可以进行类人创作和问答之后，还有什么是智能体/大模型理论上做不到的?
- 1
- 2
- 3

Automated Theorem Proving

- Russell & Whitehead, 1910s, ***Principia Mathematica,*** derive all mathematical truth using axioms and inference rules of formal logic.
- "Logic Theorist" could prove a subset in ***Principia Mathematica.***
- 1959, Hao Wang proved all theorems in ***Principia Mathematica*** on an IBM 704.
- 1976, four color theorem.
- 2016, Boolean Pythagorean Triple Problem, 200TB. There are 10^2355 possible solutions, compare with 10^80 atoms in the university and 10^18 seconds since the big bang.

***54·43.** $\vdash :. \alpha, \beta \in 1 . \supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

Dem.

$\vdash . *54{\cdot}26 . \supset \vdash :. \alpha = \iota' x . \beta = \iota' y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$

[*51·231] $\equiv . \iota' x \cap \iota' y = \Lambda .$

[*13·12] $\equiv . \alpha \cap \beta = \Lambda \quad (1)$

$\vdash . (1) . *11{\cdot}11{\cdot}35 . \supset$

$\vdash :. (\exists x, y) . \alpha = \iota' x . \beta = \iota' y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda \quad (2)$

$\vdash . (2) . *11{\cdot}54 . *52{\cdot}1 . \supset \vdash . \text{Prop}$

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

Expert System

- 1960s, the first expert system, DENTRAL.
- 1970s, MYCIN, identify bacteria causing severe infections and recommend antibiotics.
- 1980s, XCON, assist in the ordering of DEC's VAX computer systems by automatically selecting the computer system components based on the customer's requirements.

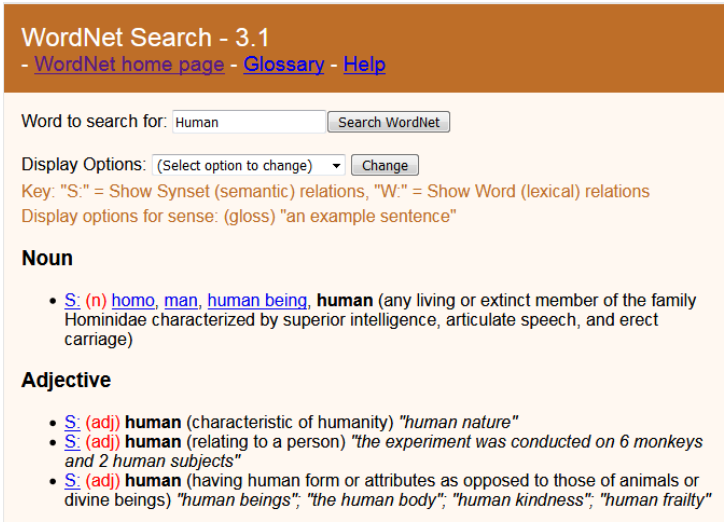
- Now still used in credit investigation, risk management, business rules engine, etc.

Knowledge Representation

- Logics:

$$(\forall x)Man(x) \supset Mortal(x) \;\&\; Man(Socrates) \rightarrow Mortal(Socrates)$$

- WordNet:



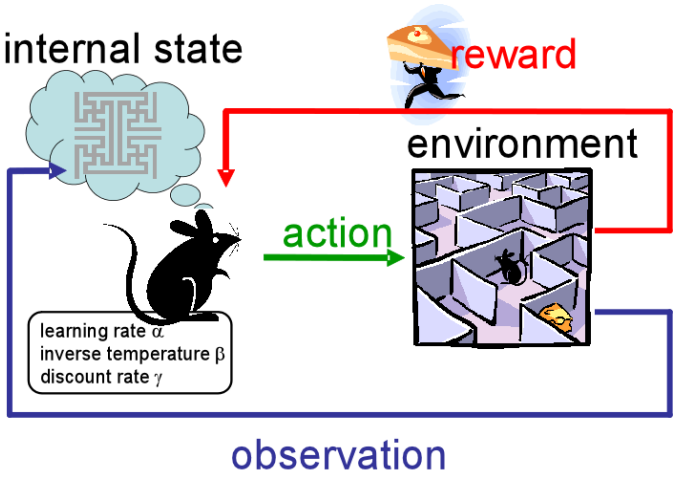
- Conceptual graph, Semantic web
- Knowledge graph

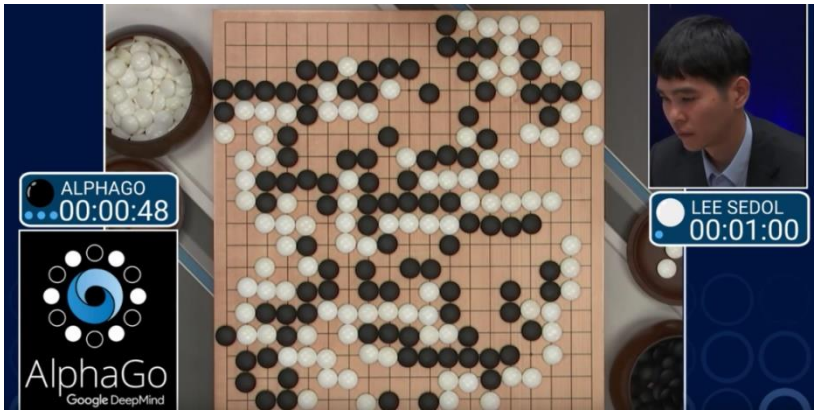
IBM Watson

- Question answering: input, query, output.
- 2011, IBM Watson win in the *Jeopardy!* .
  It has a 4TB knowledge database, including encyclopedias, dictionaries, thesauri, newswire articles, and literary works.
  It used 90 IBM Power 750 servers, 720 cores, 16TB RAM.
- Watson healthcare, financial services, education…

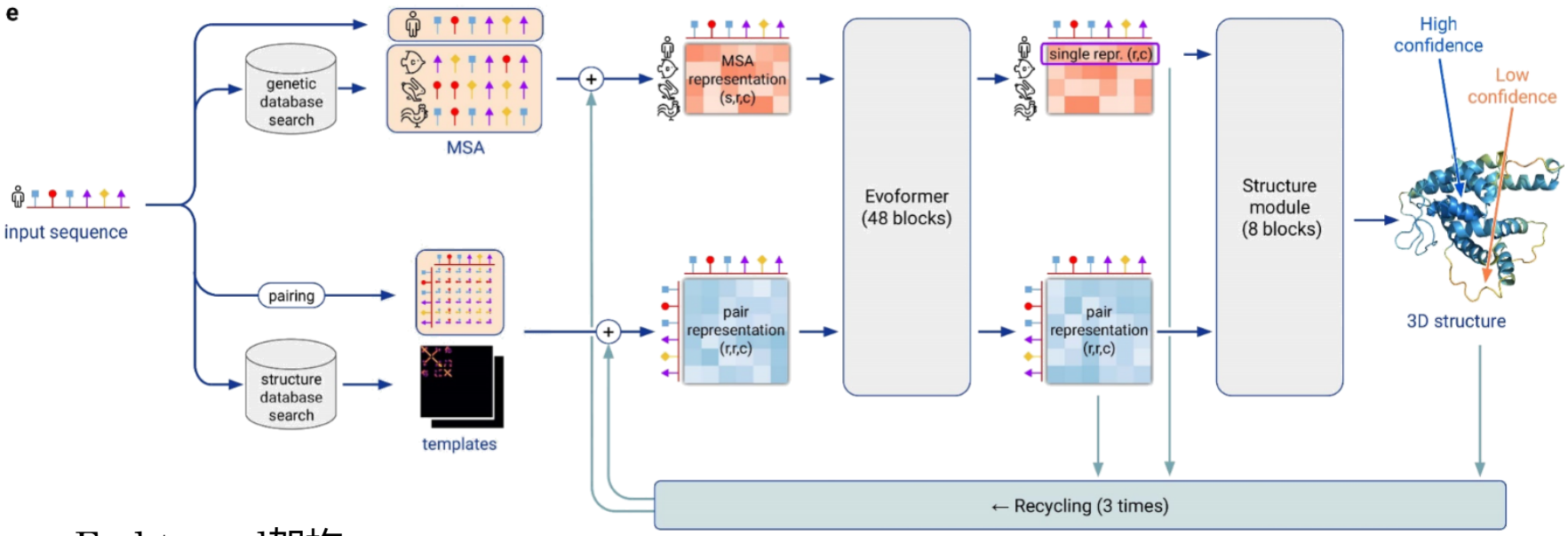
强化学习 AlphaGO

- AlphaGO, made reinforcement learning become popular overnight.
- RL mimics how animals adapt to the environment.



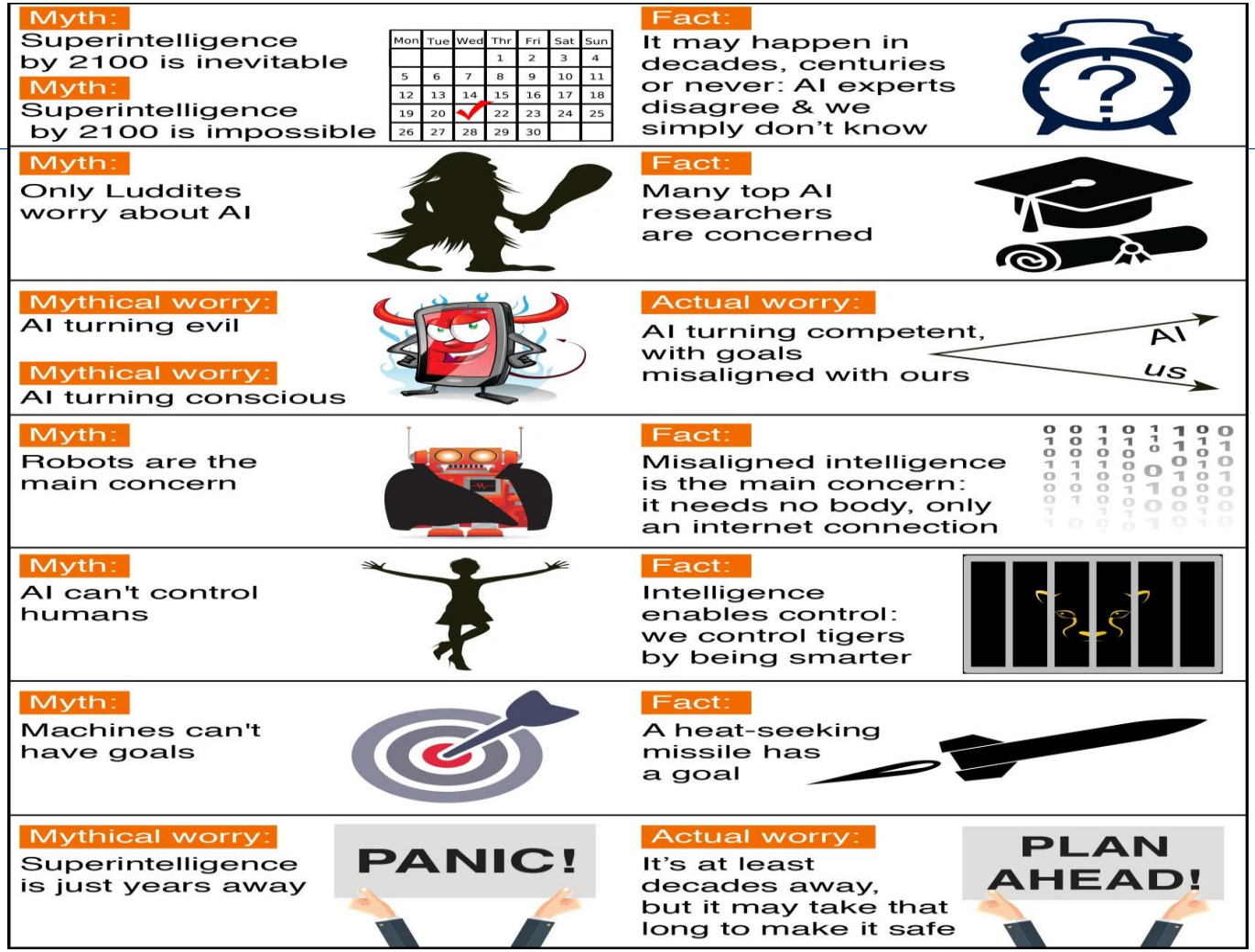


Alphafold2 的整体架构：组合创新



- End-to-end架构
- 1D与2D信息之间使用了Attention
- 3D Equivariant（等变） Structure Module
- 架构上采用了 Recycling
- Attention 模块的细节：Evoformer
- Structure module的多个创新：IPA, residue gas

Jumper, J, et al. Nature, 2021



对AI的担心

https://futureoflife.org/background/benefits-risks-of-artificial-intelligence/?cn-reloaded=1

人工智能流派

AI 开始阶段的两大流派:

Neuroscience inspired 模拟神经系统:

- ➪ neural nets learning from examples for artificial perception

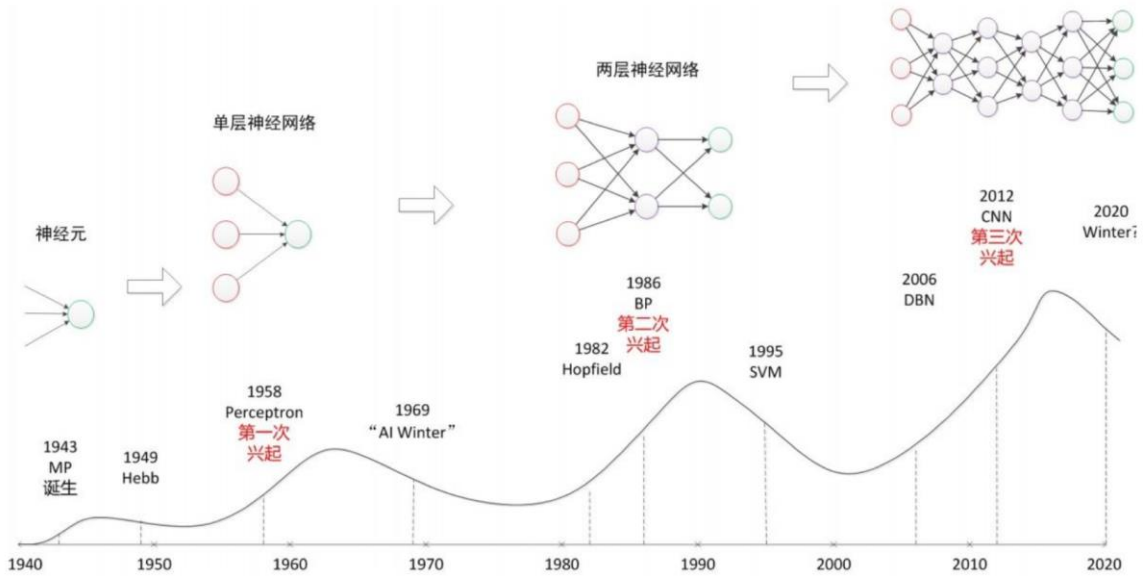
Classical symbolic AI 符号逻辑学派:

- Primacy of logical reasoning capabilities
- ➪ No learning (humans coding rules)
- ➪ poor handling of uncertainty
- Got eventually fixed (Bayes Nets...)

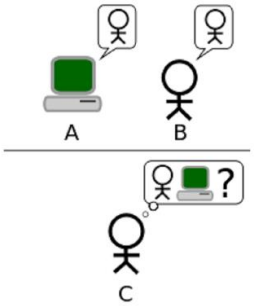
AI 现有的几大类学派

- 符号学派：知识库，逻辑推断
- 联结学派：Perceptron，神经网络
- 进化学派：Genetic Algorithm
- 贝叶斯学派: 先验概率
- 类推学派：KNN

目前是概率模型加上神经网络占了上风



人工智能里程碑事件



1）1950年：艾伦·图灵提出了著名的图灵测试，这是评估机器是否具备智能的基本方法。

2）1956年：达特茅斯会议（Dartmouth Conference）在美国举行，标志着人工智能作为一个独立学科的起点。

3）1980年代：专家系统成为人工智能的热门领域，通过将专家知识转化为规则和推理引擎，实现了某些领域的智能决策。

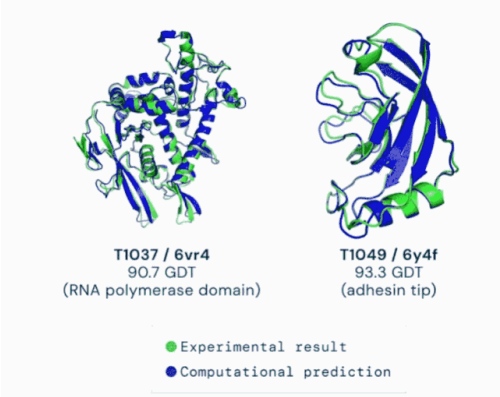
4）1990年代：神经网络和机器学习技术得到了重新关注和发展，为人工智能的进一步发展奠定了基础。

5）1997年：IBM的Deep Blue超级计算机战胜国际象棋世界冠军加里·卡斯帕罗夫，引发了对机器智能的关注。

6）2012年，CNN夺冠ImageNet

7）2016年，AlphaGo战胜人类顶尖围棋选手

8）2020年，GPT-3发布，它是迄今为止最大的语言模型。

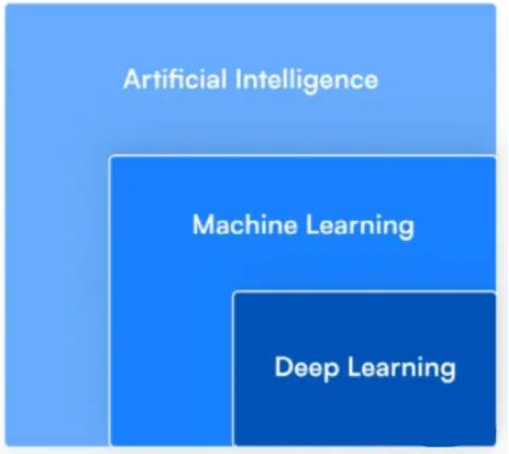


9）2021年，DeepMind的AlphaFold人工智能解决了蛋白质折叠问题。

10）2022年，OpenAI通过GPT-3.5系列大型语音模型微调而成的，全新对话式AI模型ChatGPT正式发布。

人工智能：机器学习

“机器学习是让计算机像人类一样学习和行动的科学，通过以观察和现实世界互动的形式向他们提供数据和信息，以自主的方式改善他们的学习。”



人工智能研究的范围非常广，包括演绎、推理和解决问题、知识表示、学习、运动和控制、数据挖掘等众多领域。

机器学习是实现人工智能的一种方式，而深度学习是机器学习算法的一个子集，在处理涉及非结构化数据的问题(如图像识别和自然语言)方面显示出优越的性能。

经典的机器学习过程

机器学习定义

机器学习是这样一门学科：通过计算的手段，学习经验（也可以说是利用经验）来改善系统的性能。

> Mitchell 于1997给出了一个更形式化的定义：
> 假设用P（Performace）来评估计算机程序在某类任务T（Task）上的性能，若一个程序通过利用经验E（Experience）在T中任务上获得了性能改善，则我们就说关于T和P，该程序对E进行了学习。

在生物医学信息学的背景下，这些对象可以是

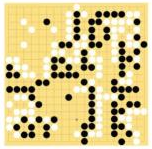
- DNA and protein sequences
- Bimolecular Structures
- Biological network motifs
- Data from microarrays and mass spectrometry experiment
- Clinical outcome
- Diagnostics

机器学习

机器学习主要通过计算的手段，利用经验来改善系统自身的性能在计算机系统中，"经验"通常以"数据"形式存在。机器学习所研究的主要内容，是关于在计算机上从数据中产生“模型” (model) 的算法，即“学习算法” (learning algorithm)

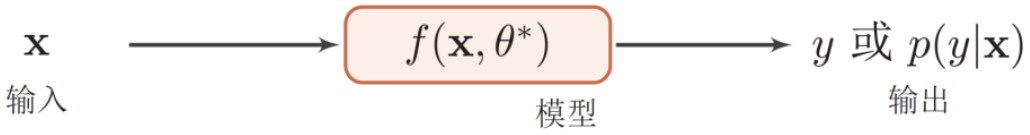
语音识别 $f($  $)=$ “你好”

图片识别 $f($  $)=$ “9”

围棋 $f($  $)=$ “6-5” （落子位置）

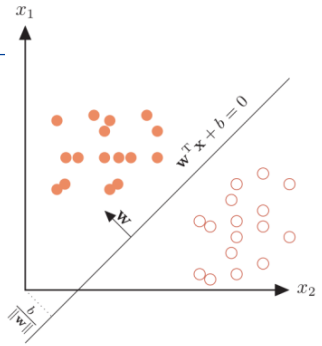
翻译 $f($ “你好！” $)=$ “Hello!”

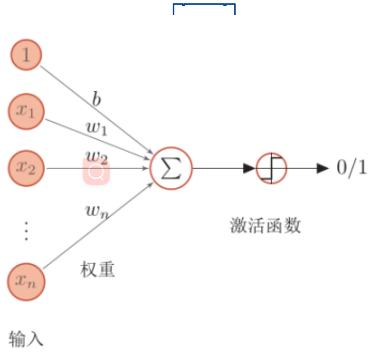
机器学习 ≈ 构建一个映射函数



机器学习的三要素

- 模型
  - 线性方法：$f(\mathbf{x},\theta)=\mathbf{w}^{\mathrm{T}}\mathbf{x}+b$
  - 广义线性方法：$f(\mathbf{x},\theta)=\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})+b$
    - 如果$\phi(x)$为可学习的非线性基函数，$f(\mathbf{x},\theta)$就等价于神经网络。





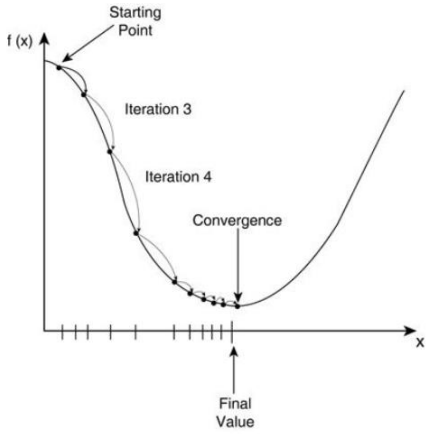
- 学习准则
  - 期望风险

$$\mathcal{R}(f)=\mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}[\mathcal{L}(f(\mathbf{x}),y)],$$

- 优化
  - 梯度下降

输入: 训练集$\mathcal{D}=\{(\mathbf{x}^{(n)},y^{(n)})\},n=1,\cdots,N$，验证集$\mathcal{V}$，学习率$\alpha$

随机初始化θ;
repeat
  对训练集$\mathcal{D}$中的样本随机重排序;
  for $n=1\cdots N$ **do**
    从训练集$\mathcal{D}$中选取样本$(\mathbf{x}^{(n)},y^{(n)})$;
    // 更新参数
    $\theta\leftarrow\theta-\alpha\dfrac{\partial\mathcal{L}(\theta;x^{(n)},y^{(n)})}{\partial\theta}$;
  end
until 模型$f(\mathbf{x},\theta)$在验证集$\mathcal{V}$上的错误率不再下降;
输出: θ



常见的机器学习算法

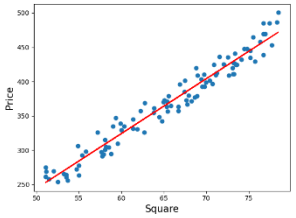
一、线性模型

给定由 d个属性描述的示例$x = (x_1; x_2; \ldots; x_d)$，其中 x_1是 x在第 i个属性上的取值，线性模型 (linear model)试图学得一个通过属性的线性组合来进行预测的函数，即：

$$f(x) = \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_d x_d + b$$

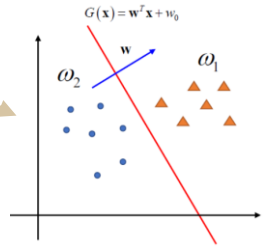
经典线性模型：

1. 线性回归模型



2. 对数几率回归

3. 线性判别分析

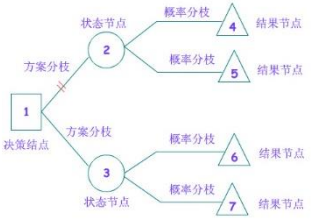


4. 多分类学习

……

二、决策树

决策树 (decision tree) 是一类常见的机器学习方法。以二分类任务为例，希望从给定训练数据集学得一个模型用以对新示例进行分类，这个把样本分类的任务，可看作对“当前样本属于正类吗?”这个问题的“决策”或“判定”过程。



输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
属性集 $A = \{a_1, a_2, \ldots, a_d\}$.
过程： 函数 TreeGenerate(D, A)
1: 生成结点 node;
2: **if** D 中样本全属于同一类别 C **then**
3: 将 node 标记为 C 类叶结点; **return**
4: **end if**
5: **if** $A = \varnothing$ **OR** D 中样本在 A 上取值相同 **then**
6: 将 node 标记为叶结点, 其类别标记为 D 中样本数最多的类; **return**
7: **end if**
8: 从 A 中选择最优划分属性 a_*;
9: **for** a_* 的每一个值 a_*^v **do**
10: 为 node 生成一个分支; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
11: **if** D_v 为空 **then**
12: 将分支结点标记为叶结点, 其类别标记为 D 中样本最多的类; **return**
13: **else**
14: 以 TreeGenerate(D_v, $A \setminus \{a_*\}$)为分支结点
15: **end if**
16: **end for**
输出： 以 node 为根结点的一棵决策树

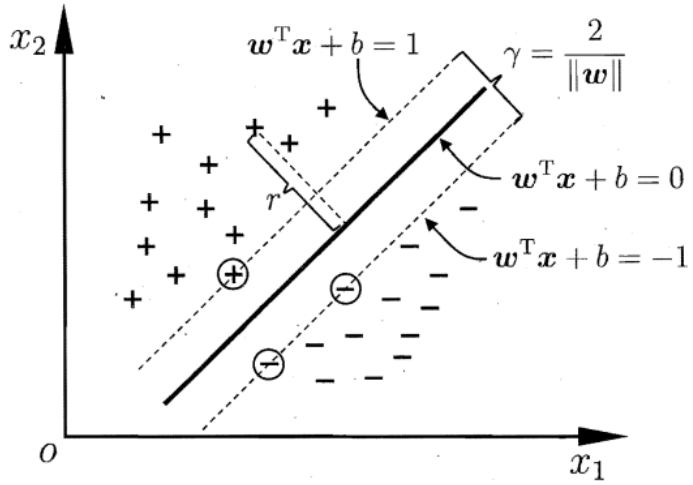
常见的机器学习算法

三、SVM（Support Vector Machines, SVM）

寻找到一个超平面使样本分成两类，并且间隔最大。

SVM分类，就是找到一个平面，让两个分类集合的支持向量或者所有的数据（LSSVM）离分类平面最远；

SVR回归，就是找到一个回归平面，让一个集合的所有数据到该平面的距离最近。



四、贝叶斯分类器

贝叶斯决策论 (Bayesian decision theory)是概率框架下实施决策的基本方法。

在贝叶斯公式计算出后验概率的基础上，进一步做归属的决定——分类

朴素贝叶斯分类算法

条件独立（conditional independence）

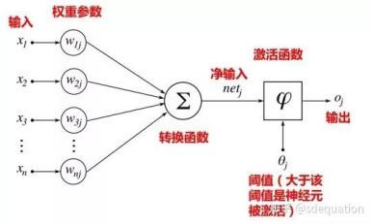
当给定类时，假定属性A_i 之间是独立的:

- $P(A_1, A_2, \ldots, A_n \mid C) = P(A_1 \mid C_j)\, P(A_2 \mid C_j) \ldots P(A_n \mid C_j)$
- Can estimate $P(A_i \mid C_j)$ for all A_i and C_j.
- New point is classified to C_j if $P(C_j)$ Õ $P(A_i \mid C_j)$ is maximal.

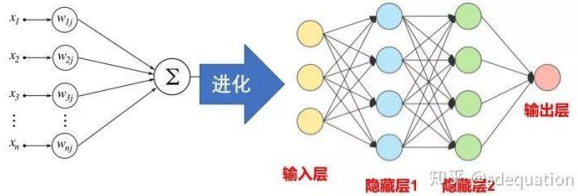
常见的机器学习算法

五、神经网络

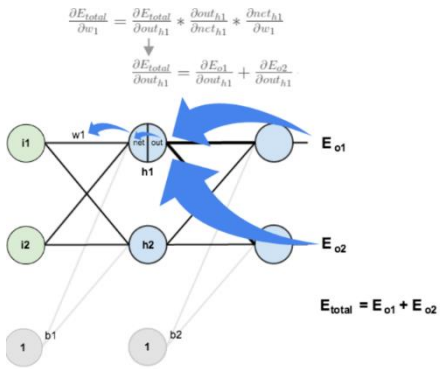
1. 人工神经网络的首次提出可以追溯到上世纪40 年代，美国神经科学家 Warren McCulloch 和 Walter Pitts 提出的**神经逻辑**计算模型（Neural Logical Calculus Model）。



2. 1959 年，加拿大神经科学家 David Hubel 和瑞典神经科学家 Torsten Wiesel发现视觉神经系统中存在两种不同细胞，复杂细胞和简单细胞，分别位于大脑的不同部位。受此启发，人们便在原来的神经网络中加入**隐藏层**以体现复杂细胞可以跨层次存在。



3. 直到 1984 年，深度学习之父，杰弗里 辛顿（Geoffrey Hinton）等人将所谓的**反向迭代算法**（Backpropagation）引入神经网络中，神经网络的研究方才再次步入正规。例如如今炙手可热的玻尔兹曼机 、卷积和长短期记忆等，实际上在上世纪 80-90 年代就出现了。



神经网络与其他算法

SVM支持向量机的优点：

- 可以解决高维问题，即大型特征空间；
- 解决小样本下机器学习问题；
- 能够处理非线性特征的相互作用；
- 无局部极小值问题；（相对于神经网络等算法）
- 无需依赖整个数据；

SVM支持向量机的缺点：

- 当观测样本很多时，效率并不是很高；
- 对非线性问题没有通用解决方案，有时候很难找到一个合适的核函数；
- 对于核函数的高维映射解释力不强，尤其是径向基函数；
- 常规SVM只支持二分类；

人工神经网络的优点：

•分类的准确度高；

•并行分布处理能力强,分布存储及学习能力强，

•对噪声神经有较强的鲁棒性和容错能力；

•具备联想记忆的功能，能充分逼近复杂的非线性关系；

•可处理批量巨大、多元复杂的数据，完成更复杂的任务

人工神经网络的缺点：

•神经网络需要大量的参数，如网络拓扑结构、权值和阈值的初始值；

•黑盒过程，不能观察之间的学习过程，输出结果难以解释，会影响到结果的可信度和可接受程度；

•学习时间过长，有可能陷入局部极小值，甚至可能达不到学习的目的。

神经网络简介

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

人工智能算法

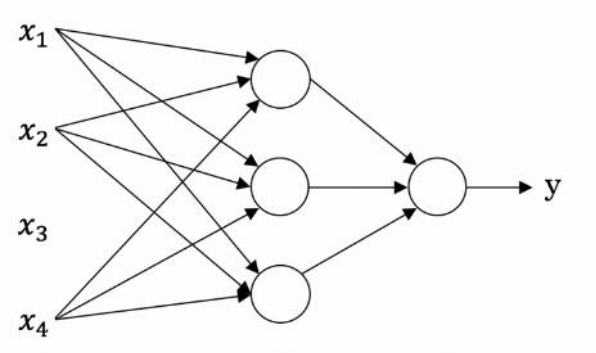
- 基本算法
  - 神经网络
  - 深度学习
  - 原则上各种机器学习算法都可以使用

- 拓展算法
  - 迁移学习
  - 增强学习
  - 对抗学习
  - 小样本学习
  - 因果推断

Neural Network (NN)

- How does neural network work?

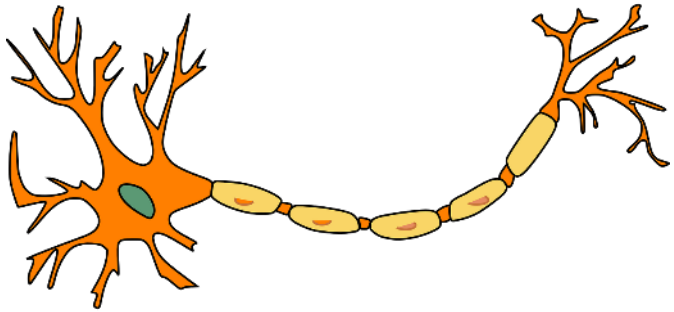
节点，连接



Standard NN

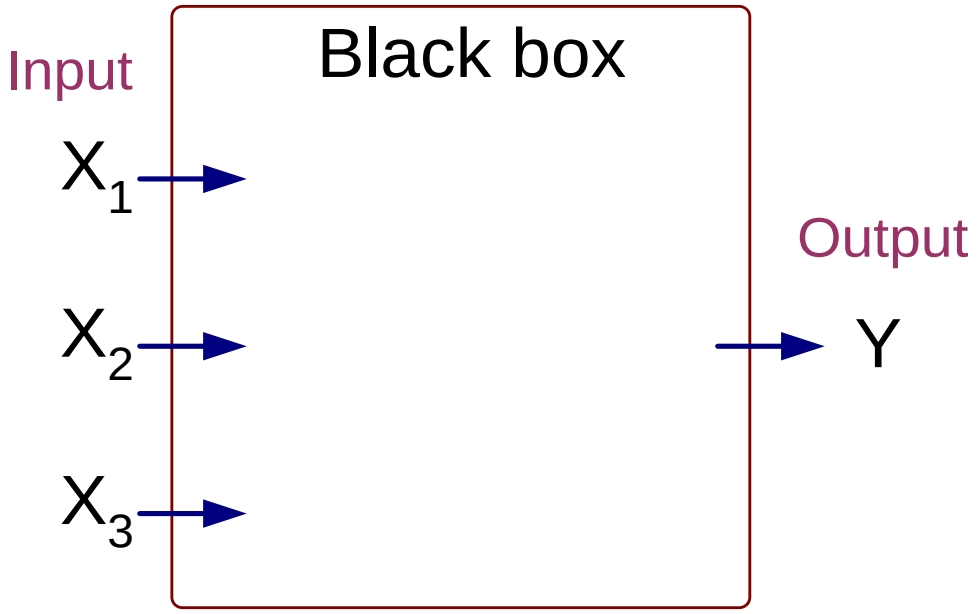
- Mimic human neural system

千亿神经元，每个约1000个连接



Artificial Neural Networks (ANN)

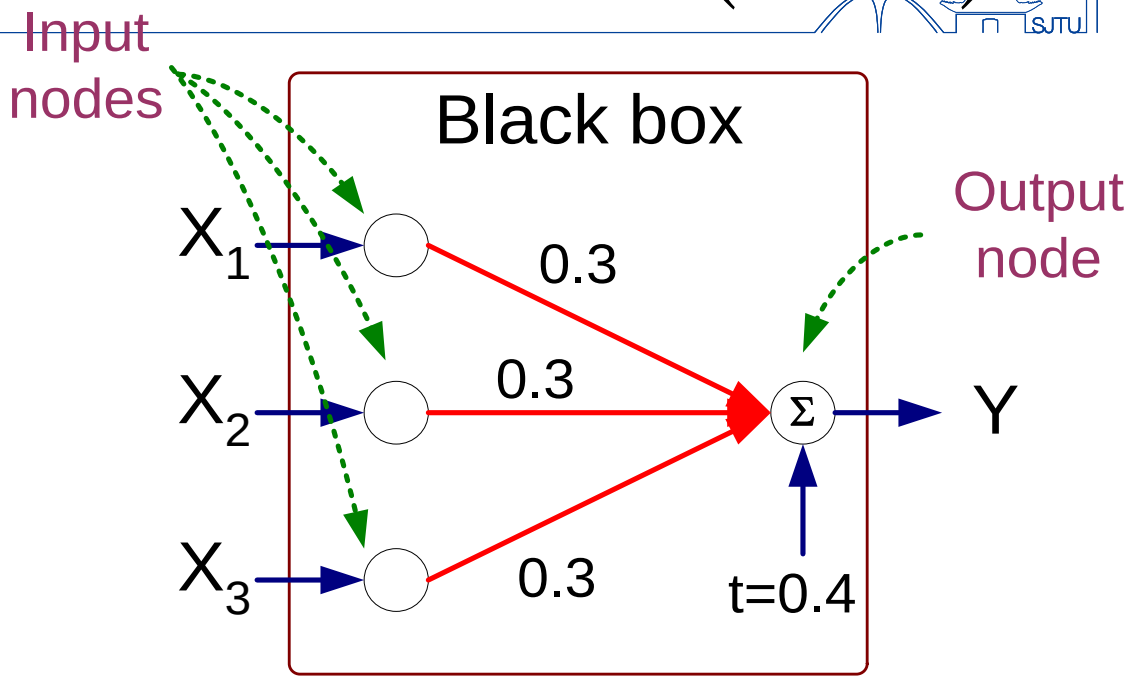
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

Artificial Neural Networks (ANN)

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

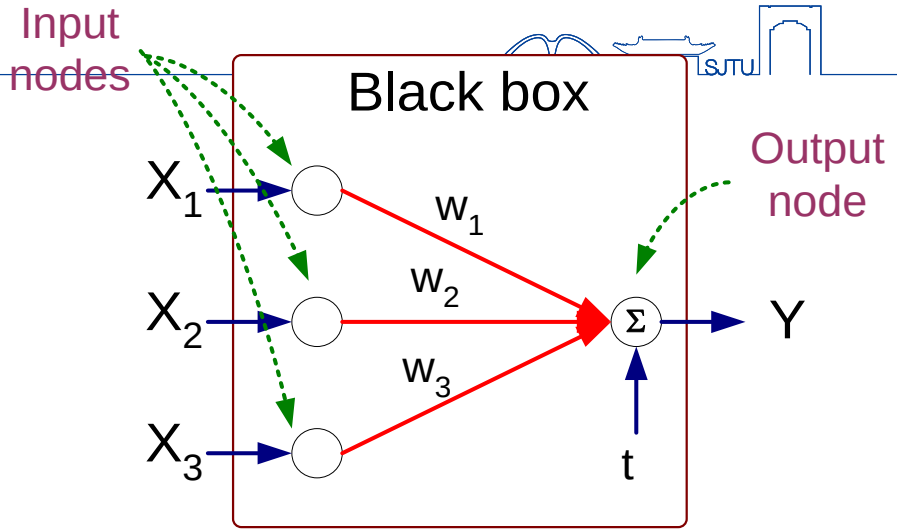


$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Artificial Neural Networks (ANN)

- Model is an assembly of inter-connected nodes and weighted links
- Output node sums up each of its input value according to the weights of its links
- Compare output node against some threshold t

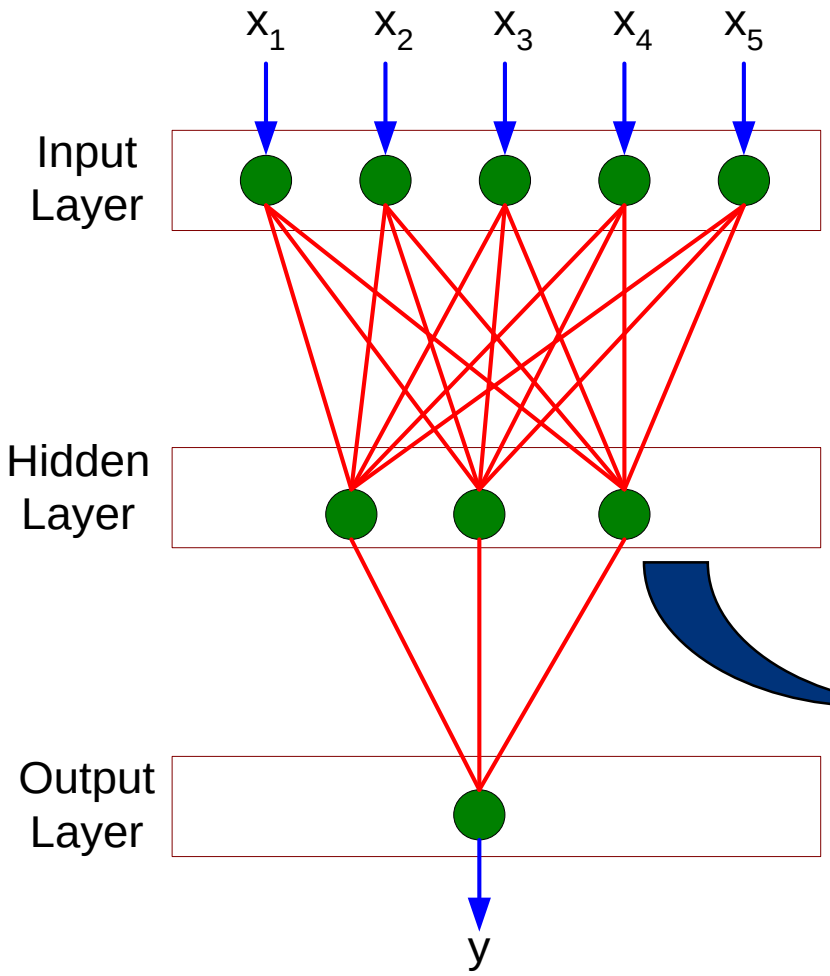


Perceptron Model

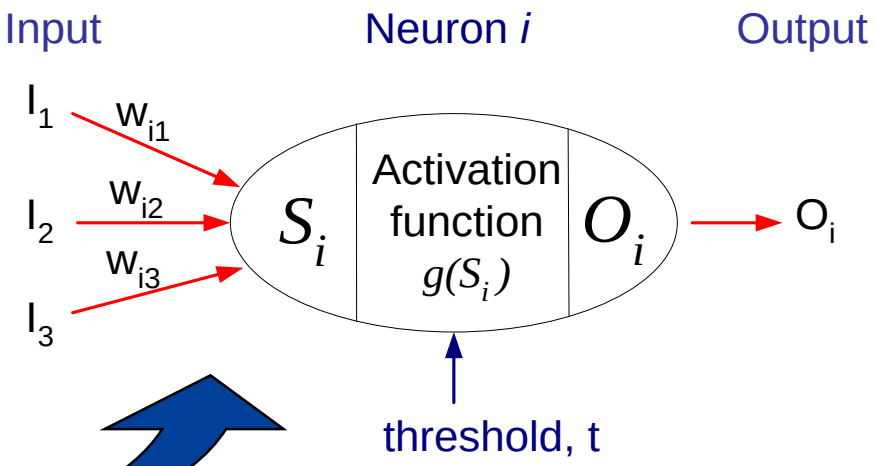
$$Y = I(\sum_i w_i X_i - t) \quad \text{or}$$

$$Y = sign(\sum_i w_i X_i - t)$$

General Structure of ANN



Training ANN means learning the weights of the neurons

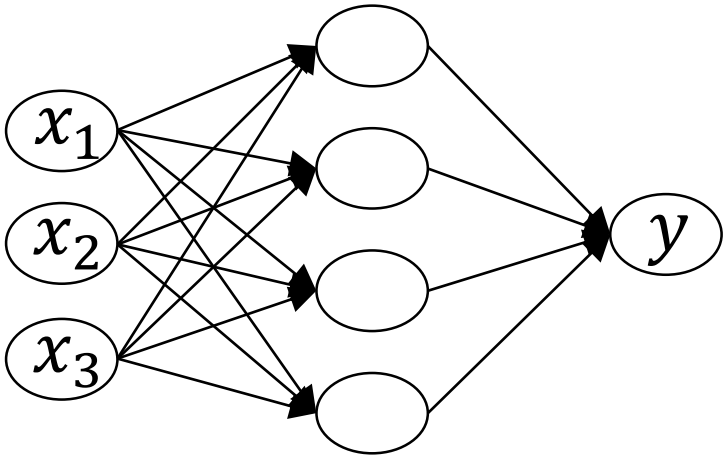


深度学习算法基础

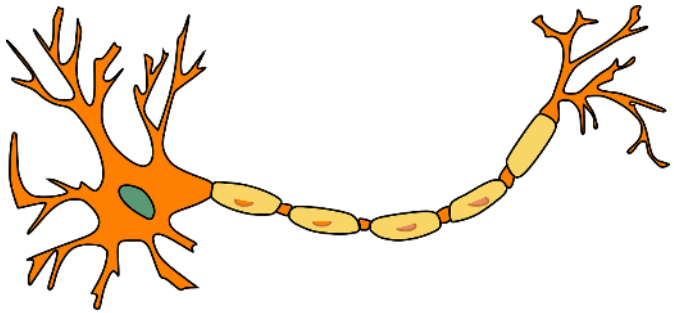
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Neural Network (NN)

- Basic structure of neural network

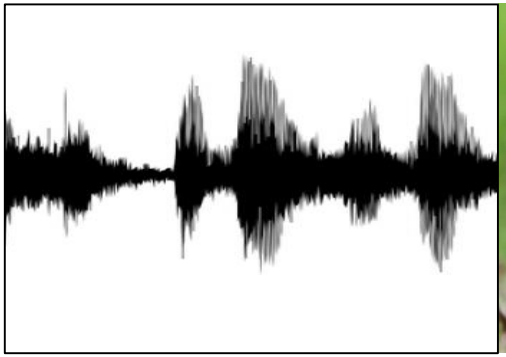


- Mimic human neural system



Unstructured Data

- Structured data vs. Unstructured data



From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thy self thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
Pity the world, or else this glutton be,
To eat the world's due, by the grave and thee.

Audio Image Text

- Deep learning helps machine understand unstructured data better

How to construct a NN

- Start from simple image recognition:

Blue
Green
Red

		255	134	93	22
	255	134	202	22	2
255	231	42	22	4	30
123	94	83	2	192	124
34	44	187	92	34	142
34	76	232	124	94	
67	83	194	202		

$$x = \begin{vmatrix} 255 \\ \vdots \\ 142 \end{vmatrix} \qquad y = \begin{cases} 1 \text{ (cat)} \\ 0 \text{ (non cat)} \end{cases}$$

$$\hat{y} = P(y = 1|x), \qquad 0 \le y \le 1$$

$$\hat{y} = wx + b \quad ?$$

构建神经网络需解决的问题：

layer模块：
包括了数据输入层、全连接层、激活函数层、损失函数层等。

- 前馈神经网络结构及BP算法原理

function_for_layer模块：
定义了不同的激活函数、损失函数

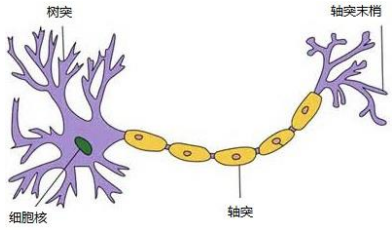
- 如何选择激活函数以及损失函数
- 权重与参数初始化方法

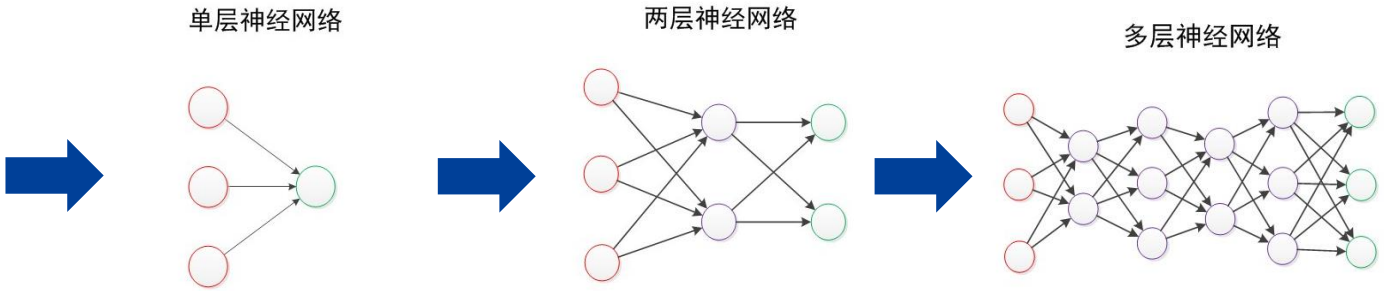
update_method模块：
定义了学习率、梯度的更新方法。

- 如何选择学习率更新机制以及梯度更新方法

Net模块：
定义了神经网络的结构

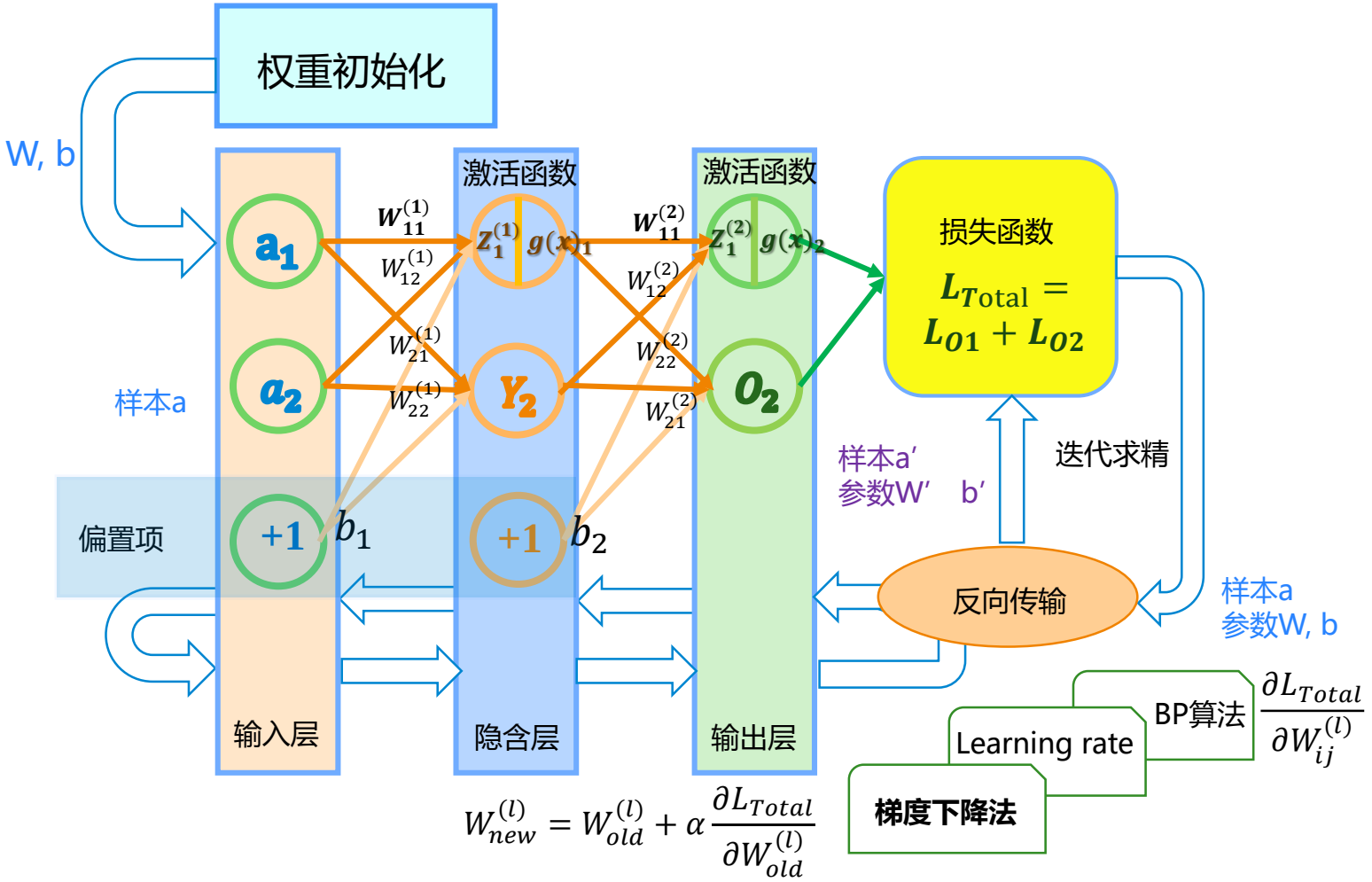
- 如何选择隐藏层层数与每一层节点数





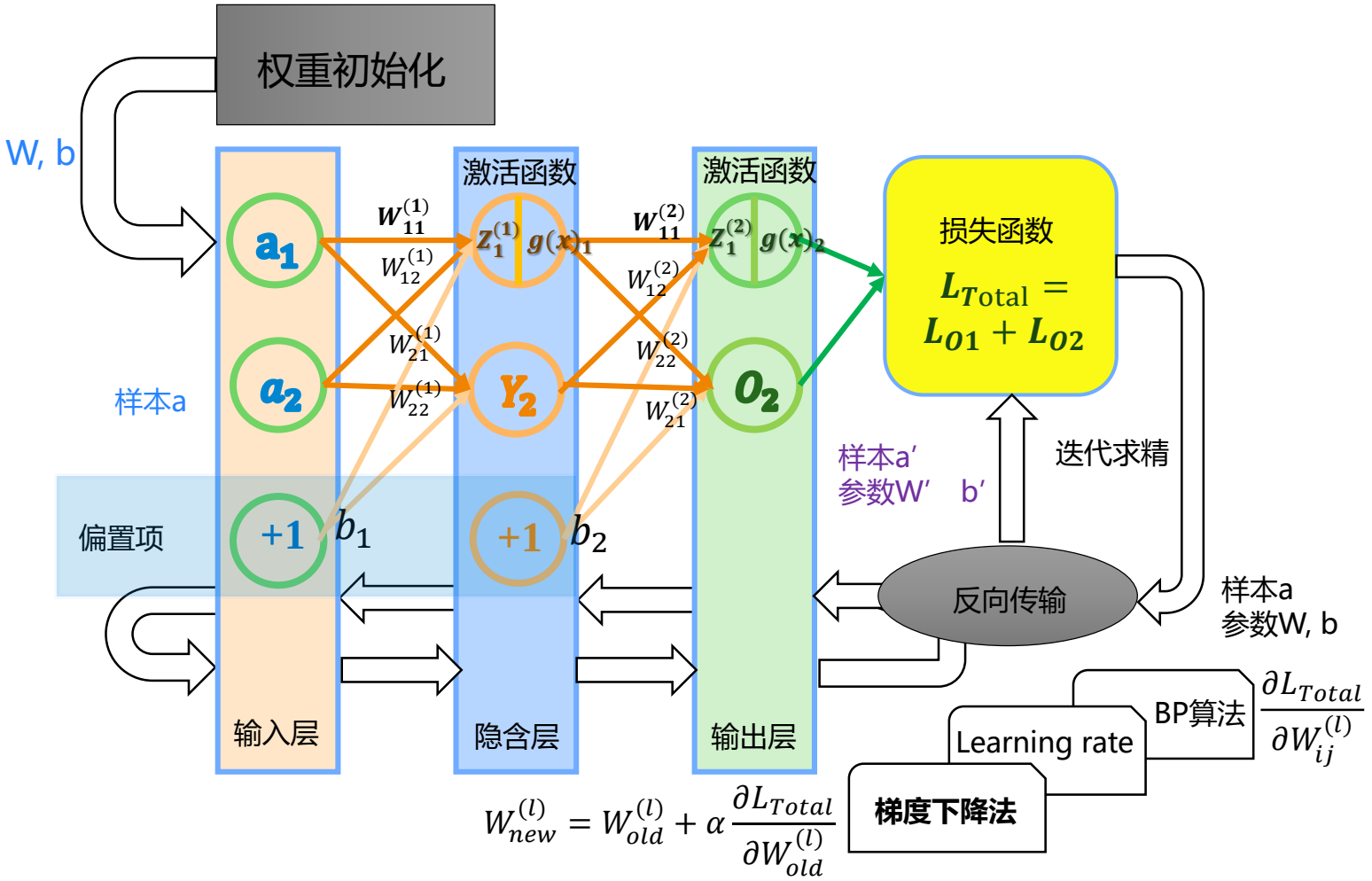
前馈神经网络：

layer模块：
包括了数据输入层、全连接层、激活函数层、损失函数层等。



前向传输：

layer模块：
包括了数据输入层、全连接层、激活函数层、损失函数层等。

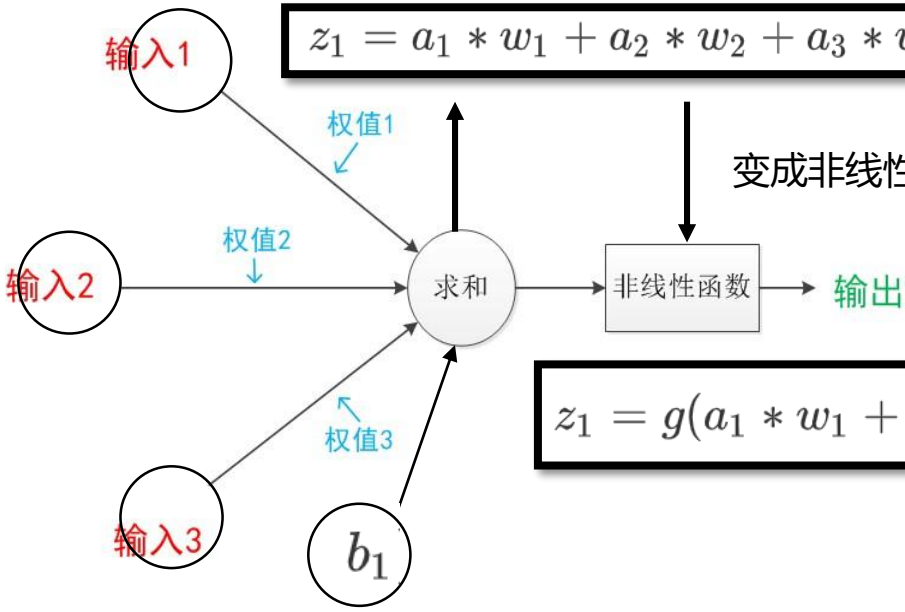


前馈神经网络训练过程：

layer模块：
包括了数据输入层、全连接层、激活函数层、损失函数层等。

DL中的神经元由五部分组成：

1. 输入：一个感知器可以接收多个输入a_1, a_2, a_3
2. 权重：每一个输入都有一个权重w_1, w_2, w_3
3. 偏置项：b_1
4. 激活函数：$g(x)$也叫做非线性单元，神经网络的激活函数有很多种类
5. 输出：$z_1 = g(a_1 * w_1 + a_2 * w_2 + a_3 * w_3 + b_1)$



$$z_1 = a_1 * w_1 + a_2 * w_2 + a_3 * w_3 + b_1$$

变成非线性

$$z_1 = g(a_1 * w_1 + a_2 * w_2 + a_3 * w_3 + b_1)$$

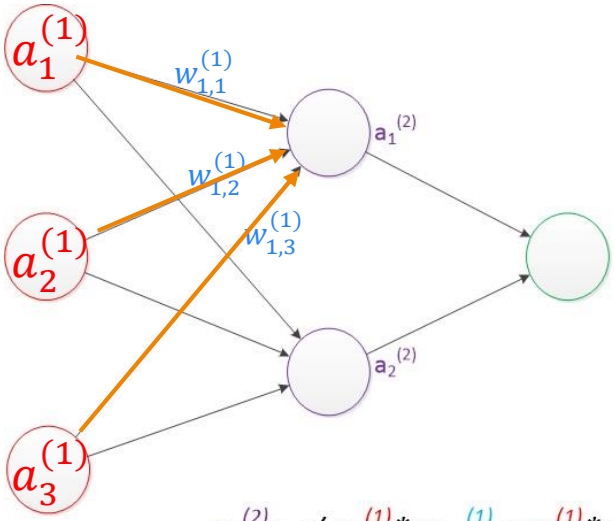
$$\begin{bmatrix} z_1^{(l)} \\ \vdots \\ z_i^{(l)} \\ \vdots \\ z_{s_l}^{(l)} \end{bmatrix} = \begin{bmatrix} W_{11}^{(l-1)} & \cdots & W_{1j}^{(l-1)} & \cdots & W_{1s_{l-1}}^{(l-1)} \\ \vdots & & \vdots & & \vdots \\ W_{i1}^{(l-1)} & \cdots & W_{ij}^{(l-1)} & \cdots & W_{is_{l-1}}^{(l-1)} \\ \vdots & & \vdots & & \vdots \\ W_{s_l 1}^{(l-1)} & \cdots & W_{s_l j}^{(l-1)} & \cdots & W_{s_l s_{l-1}}^{(l-1)} \end{bmatrix} \begin{bmatrix} a_1^{(l-1)} \\ \vdots \\ a_i^{(l-1)} \\ \vdots \\ a_{s_{l-1}}^{(l-1)} \end{bmatrix} + \begin{bmatrix} b_1^{(l-1)} \\ \vdots \\ b_i^{(l-1)} \\ \vdots \\ b_{s_{l-1}}^{(l-1)} \end{bmatrix}$$

$$\begin{bmatrix} a_1^{(l)} \\ \vdots \\ a_i^{(l)} \\ \vdots \\ a_{s_l}^{(l)} \end{bmatrix} = \begin{bmatrix} f\left(z_1^{(l)}\right) \\ \vdots \\ f\left(z_i^{(l)}\right) \\ \vdots \\ f\left(z_{s_l}^{(l)}\right) \end{bmatrix}$$

前馈神经网络训练过程：

layer模块：
包括了数据输入层、全连接层、激活函数层、损失函数层等。

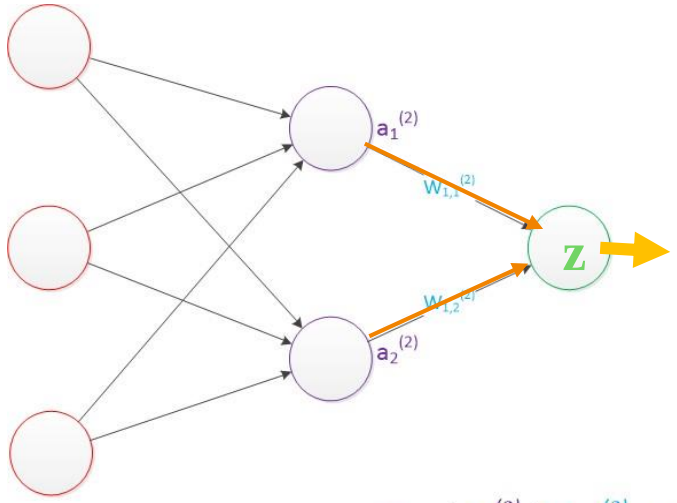
前向传播：单方向



$$a_1^{(2)} = g(a_1^{(1)} * w_{1,1}^{(1)} + a_2^{(1)} * w_{1,2}^{(1)} + a_3^{(1)} * w_{1,3}^{(1)})$$

$$a_2^{(2)} = g(a_1^{(1)} * w_{2,1}^{(1)} + a_2^{(1)} * w_{2,2}^{(1)} + a_3^{(1)} * w_{2,3}^{(1)})$$

(1)Input层向hidden层传递

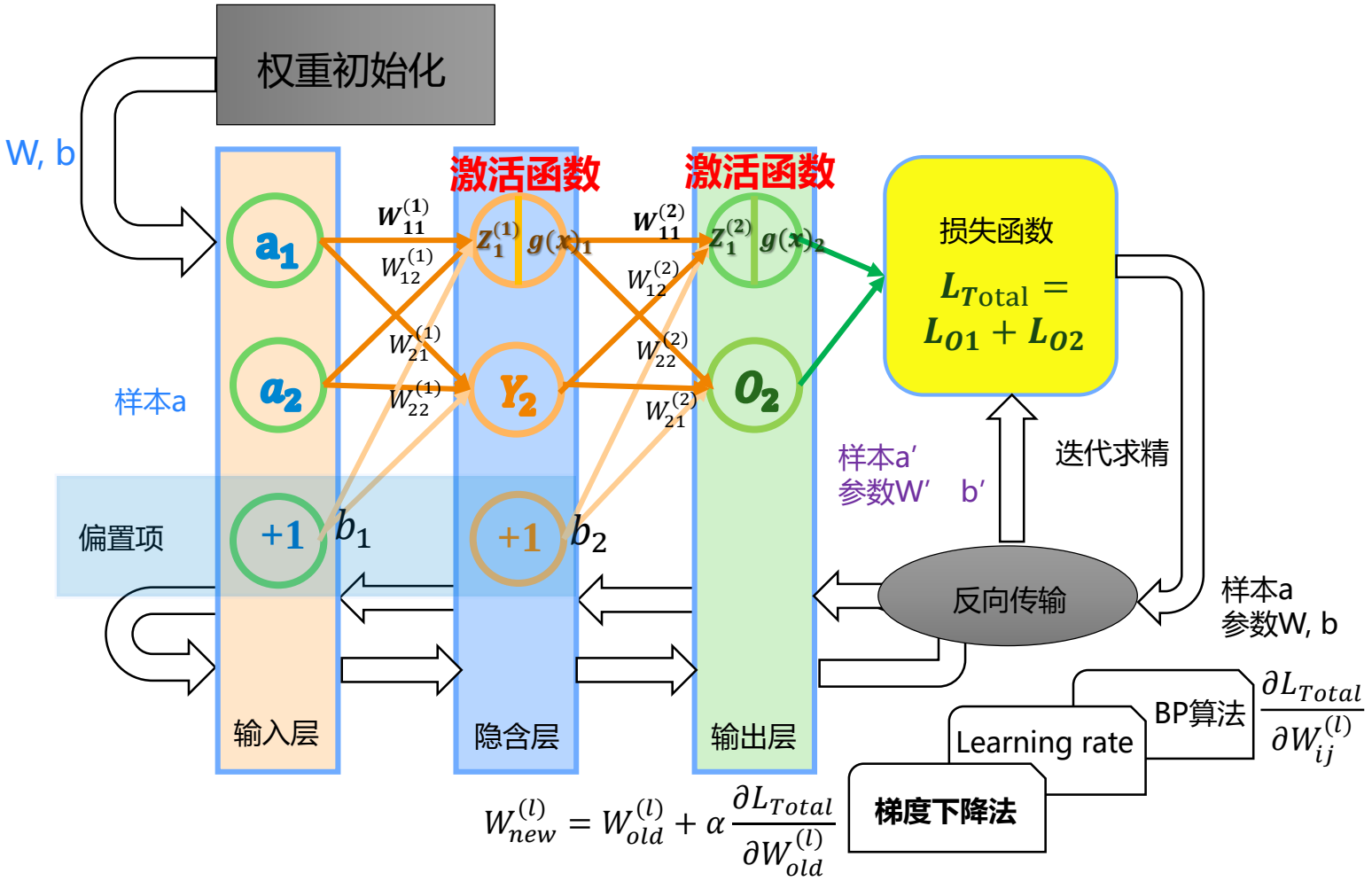


$$z = g(a_1^{(2)} * w_{1,1}^{(2)} + a_2^{(2)} * w_{1,2}^{(2)})$$

(2) hidden层向下一层传递

激活函数：

layer模块：
包括了数据输入层、全连接层、激活函数层、损失函数层等。



function_for_layer
模块：
定义了不同的激活函数、损失函数

激活函数

什么是激活函数?

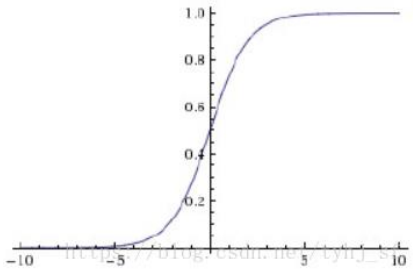
在多层神经网络中，上层节点的输出和下层节点的输入之间具有一个函数关系，这个函数称为激活函数（又称激励函数）。

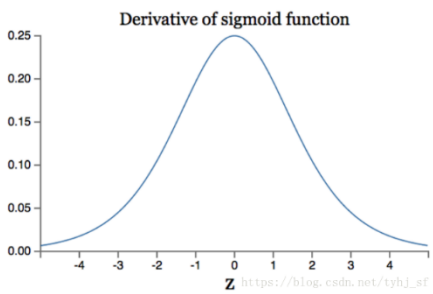
激活函数具有的特性

- 非线性：激活函数是非线性的。
- 计算简单：神经元都要经过激活运算的，在随着网络结构越来越庞大、参数量越来越多，激活函数如果计算量小就节约了大量的资源。
- f (x) ≈ x：在向前传播时，如果参数的初始化是随机量的最小值，神经网络的训练很高效。
- 可微：因为神经网络要通过反向传播来跟新参数，如果激活函数不可微，就无法根据损失函数对权重求偏导，也就无法更新权重。
- 非饱和性：饱和函数有Sigmoid、Tanh等，非饱和函数ReLU等
- 单调性：当激活函数单调时，单层网络保证是凸函数。
- 输出值的范围： 当激活函数输出值是有限的时候，基于梯度的优化方法会更加稳定，因为特征的表示受有限权值的影响更显著。

function_for_layer
模块：
定义了不同的激活函数、损失函数

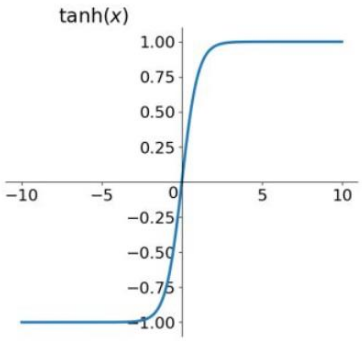
激活函数：种类与选择

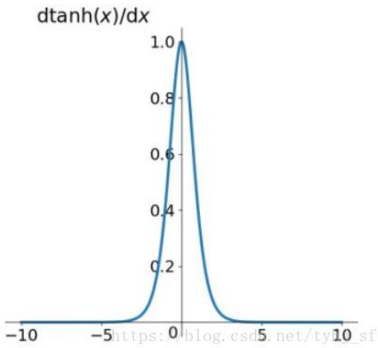




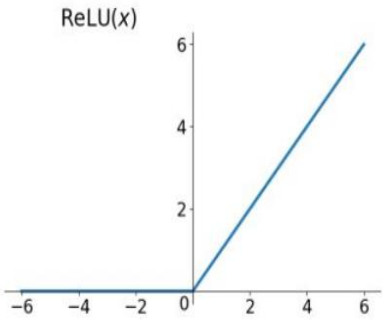
Sigmoid函数 $f(z) = \frac{1}{1+e^{-z}}$

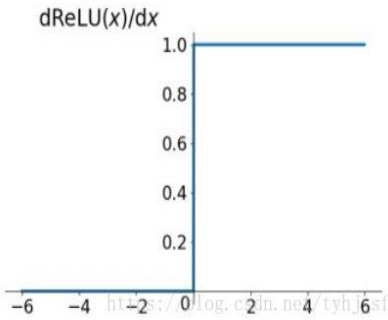
特点：
缺点1，2，3





tanh函数 $tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$



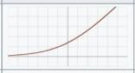
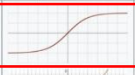
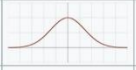


Relu函数 $Relu = max(0, x)$

取最大值函数，注意这并不是全区间可导

function_for_layer
模块：
定义了不同的激活函数、损失函数

激活函数：种类与选择

Name	Plot	Function, $f(x)$	Derivative of f, $f'(x)$	Range	Order of continuity
Softplus[8]		$\ln(1+e^x)$	$\frac{1}{1+e^{-x}}$	$(0,\infty)$	C^∞
Sigmoid linear unit (SiLU,[4] Sigmoid shrinkage,[13] SiL,[14] or Swish-1[15])		$\frac{x}{1+e^{-x}}$	$\frac{1+e^{-x}+xe^{-x}}{(1+e^{-x})^2}$	$[-0.278\ldots,\infty)$	C^∞
Scaled exponential linear unit (SELU)[10]		$\lambda\begin{cases}\alpha(e^x-1) & \text{if } x<0\\ x & \text{if } x\geq 0\end{cases}$ with parameters $\lambda=1.0507$ and $\alpha=1.67326$	$\lambda\begin{cases}\alpha e^x & \text{if } x<0\\ 1 & \text{if } x\geq 0\end{cases}$	$(-\lambda\alpha,\infty)$	C^0
Rectified linear unit (ReLU)[7]		$\begin{cases}0 & \text{if } x\leq 0\\ x & \text{if } x>0\end{cases}$ $=\max\{0,x\}=x\mathbf{1}_{x>0}$	$\begin{cases}0 & \text{if } x<0\\ 1 & \text{if } x>0\\ \text{undefined} & \text{if } x=0\end{cases}$	$[0,\infty)$	C^0
Parameteric rectified linear unit (PReLU)[12]		$\begin{cases}\alpha x & \text{if } x<0\\ x & \text{if } x\geq 0\end{cases}$ with parameter α	$\begin{cases}\alpha & \text{if } x<0\\ 1 & \text{if } x\geq 0\end{cases}$	$(-\infty,\infty)$[2]	C^0
Mish [16]		$x\tanh(\ln(1+e^x))$	$\frac{(e^x(4e^{2x}+e^{3x}+4(1+x)+e^x(6+4x)))}{(2+2e^x+e^{2x})^2}$	$[-0.308\ldots,\infty)$	C^∞
Logistic, sigmoid, or soft step		$\sigma(x)=\frac{1}{1+e^{-x}}$[1]	$f(x)(1-f(x))$	$(0,1)$	C^∞
Leaky rectified linear unit (Leaky ReLU)[11]		$\begin{cases}0.01x & \text{if } x<0\\ x & \text{if } x\geq 0\end{cases}$	$\begin{cases}0.01 & \text{if } x<0\\ 1 & \text{if } x\geq 0\end{cases}$	$(-\infty,\infty)$	C^0
Identity		x	1	$(-\infty,\infty)$	C^∞
Hyperbolic tangent (tanh)		$\tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$	$1-f(x)^2$	$(-1,1)$	C^∞
Gaussian Error Linear Unit (GELU)[4]		$\frac{1}{2}x\left(1+\operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$ $=x\Phi(x)$	$\Phi(x)+x\phi(x)$	$(-0.17\ldots,\infty)$	C^∞
Gaussian		e^{-x^2}	$-2xe^{-x^2}$	$(0,1]$	C^∞
Exponential linear unit (ELU)[9]		$\begin{cases}\alpha(e^x-1) & \text{if } x\leq 0\\ x & \text{if } x>0\end{cases}$ with parameter α	$\begin{cases}\alpha e^x & \text{if } x<0\\ 1 & \text{if } x>0\\ 1 & \text{if } x=0 \text{ and } \alpha=1\end{cases}$	$(-\alpha,\infty)$	$\begin{cases}C^1 & \text{if } \alpha=1\\ C^0 & \text{otherwise}\end{cases}$
Binary step		$\begin{cases}0 & \text{if } x<0\\ 1 & \text{if } x\geq 0\end{cases}$	$\begin{cases}0 & \text{if } x\neq 0\\ \text{undefined} & \text{if } x=0\end{cases}$	$\{0,1\}$	C^{-1}

选择怎样的activation function不在于它能否模拟真正的神经元，而在于能否便于优化整个深度神经网络。

Sigmoid和tanh存在梯度消失问题。

现在最受欢迎的激活函数是ReLU。

function_for_layer
模块：
定义了不同的激活函数、损失函数

Relu函数及变种

Relu函数的问题：

-- 由于负数部分恒为0，会导致一些神经元无法激活（可通过设置小学习率部分解决）

-- 输出不是以0为中心的。

Leaky ReLU函数/PReLU函数 $LeakyRelu(x) = \begin{cases} x(x \geq 0) \\ \alpha x(x < 0)) \end{cases}$

RReLU函数 $f(x) = \begin{cases} x & if \quad x > 0 \\ \lambda x & if \quad x <= 0 \end{cases}$

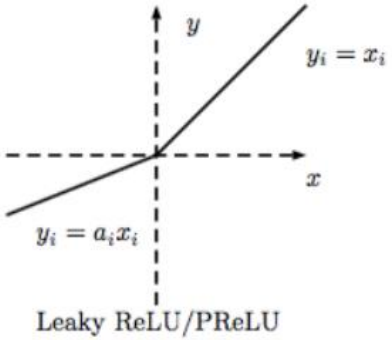
Bounded ReLU $BReLU(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq n \\ n & x > n \end{cases}$

ELU/SELU

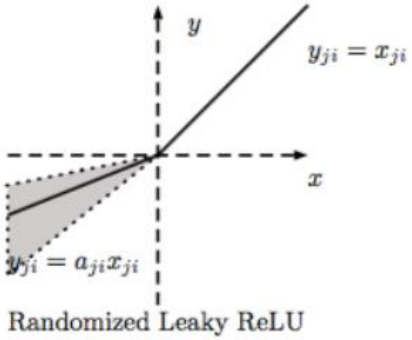
λ

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

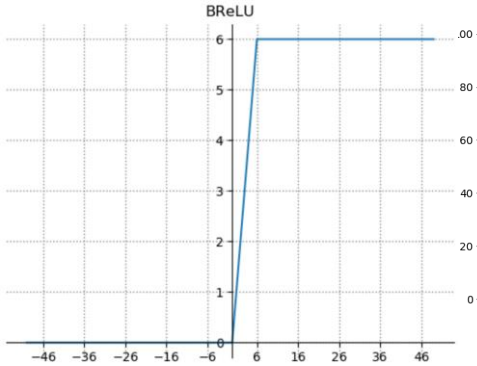
$$f'(x) = \begin{cases} 1 & if \quad x > 0 \\ f(x) + \alpha & if \quad x <= 0 \end{cases}$$

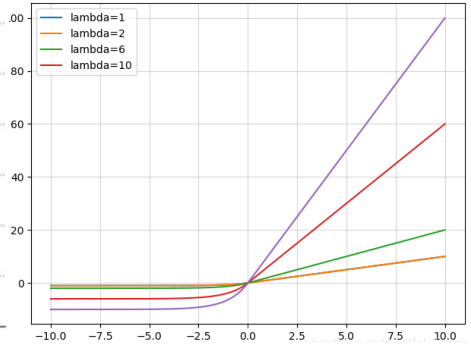


Leaky ReLU/PReLU



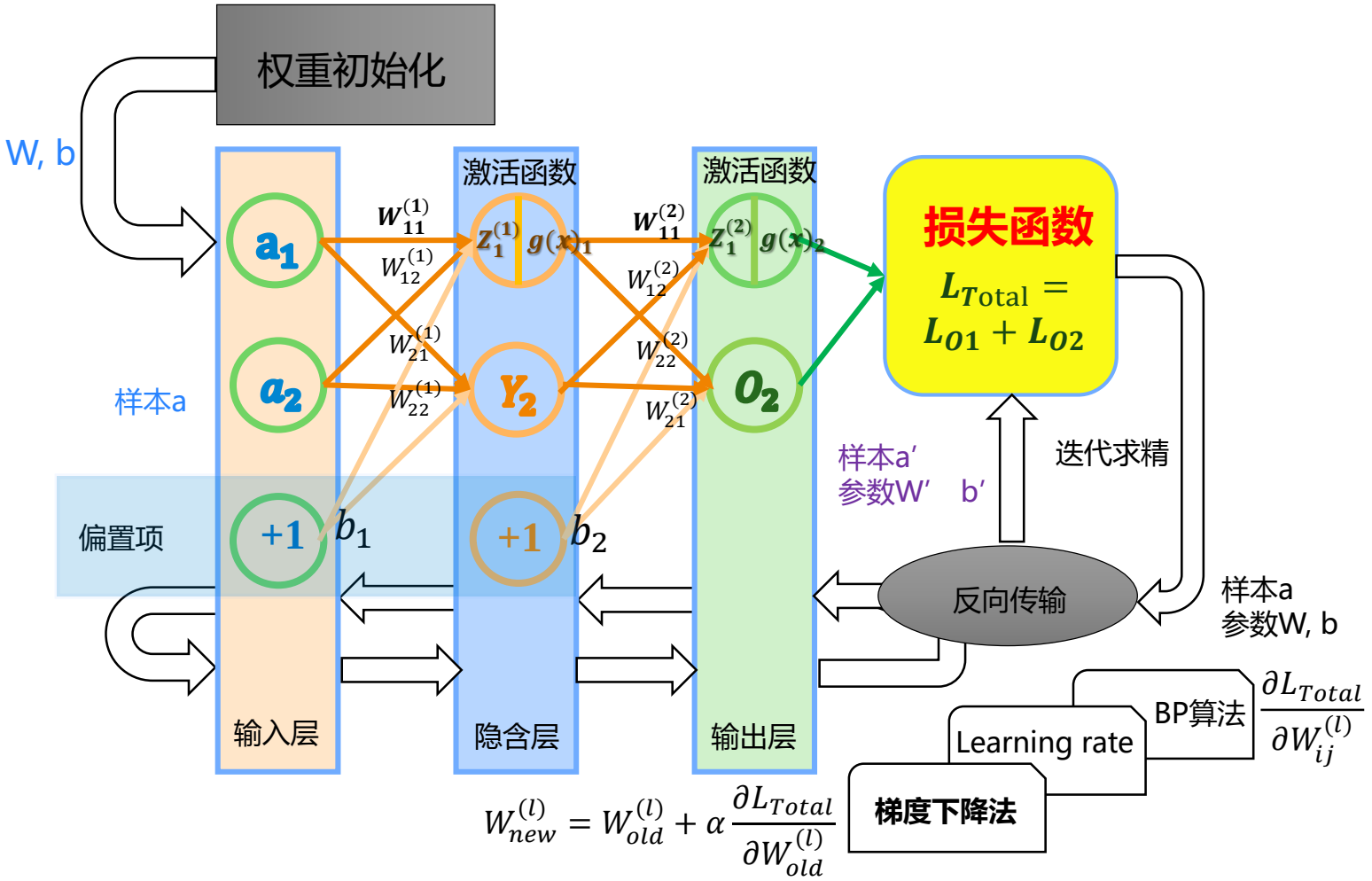
Randomized Leaky ReLU





损失函数：

layer模块：
包括了数据输入层、全连接层、激活函数层、损失函数层等。



function_for_layer
模块：
定义了不同的激活函数、损失函数

损失函数Loss

损失函数（Loss Function）直接作用于单个样本，用来表达样本的误差
代价函数（Cost Function）作用于整个训练集，是整个样本集的平均误差，对所有损失函数值的平均
目标函数（Object Function）是我们最终要优化的函数，也就是代价函数+正则化函数（经验风险+结构风险）

$$\theta^* = \arg\min_{\theta} \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i;\theta)) + \lambda\,\Phi(\theta)$$

如何根据问题选择损失函数	
回归问题	均方误差MSE，平均绝对误差MAE，Huber Loss
分类问题	铰链损失（hinge loss），交叉熵损失（Cross Entropy）
排序问题	Pointwise Ranking Loss，Pairwise Ranking Loss，Listwise Ranking Loss

function_for_layer
模块：
定义了不同的激活函数、损失函数

交叉熵损失函数

信息熵

$$H(\mathbf{X}) = -\sum_{i=1}^{n} P(x_i)\log(P(x_i))) \qquad (\mathbf{X} = x_1, x_2, x_3 ..., x_n)$$

交叉熵损失（Cross Entropy)

单个样本的交叉熵函数：

$$L = -[ylog\,\hat{y} + (1-y)log\,(1-\hat{y})]$$

交叉熵公式表示：

$$H(p,q) = -\sum_{i=1}^{n} p(x_i)\,log\,(q(x_i))$$

*	猫	狗	马
Label	0	1	0
Pred	0.2	0.7	0.1

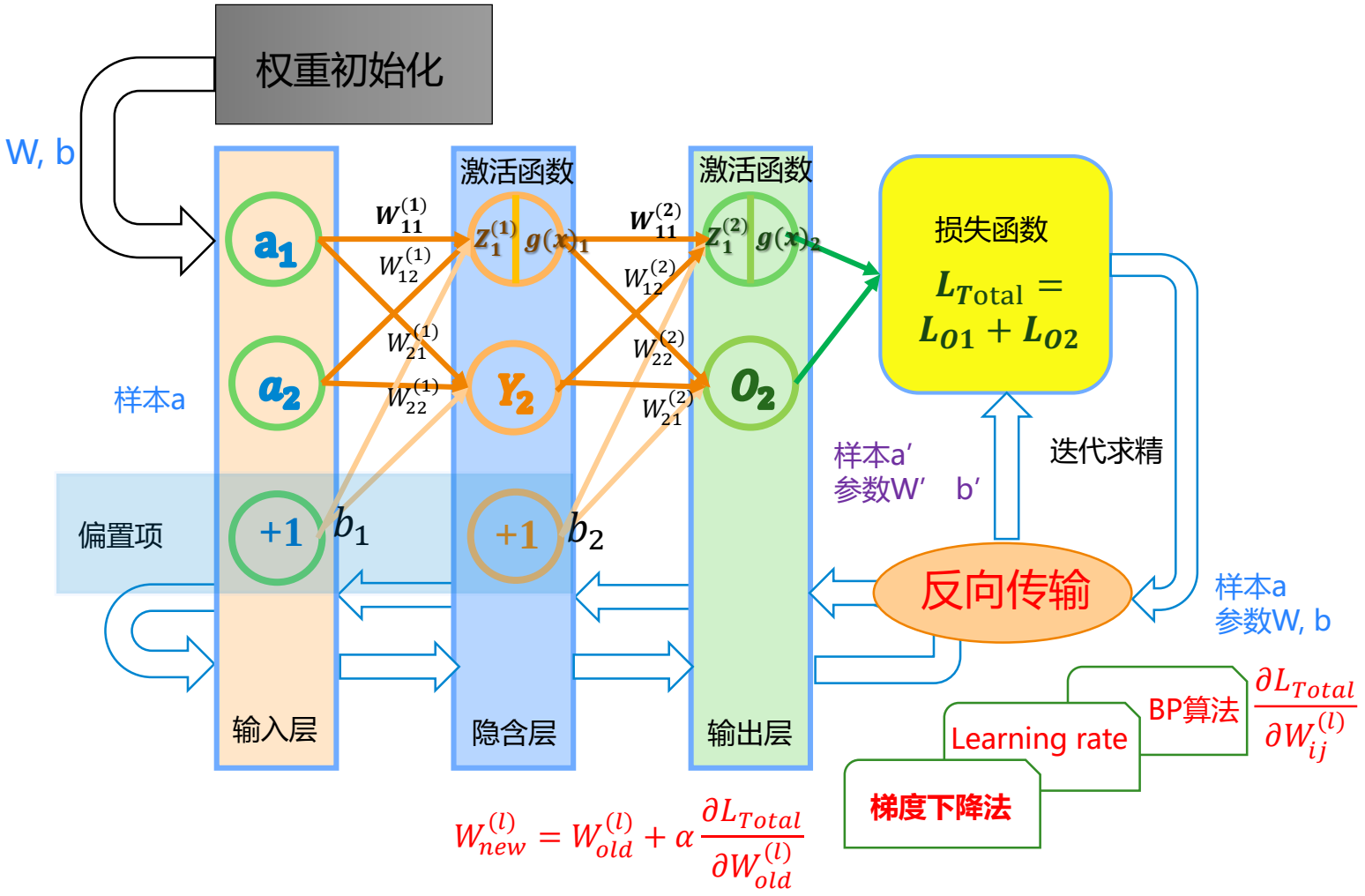
$$loss = -(0 * \log(0.2) + 1 * \log(0.7) + 0 * \log(0.1)) = 0.36$$

$$loss = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n} p(x_{ij})\,log\,(q(x_{ij}))$$

根据不同应用需求，交叉熵loss选择

反向传输：

layer模块：
包括了数据输入层、全连接层、激活函数层、损失函数层等。



为什么需要反向传输?

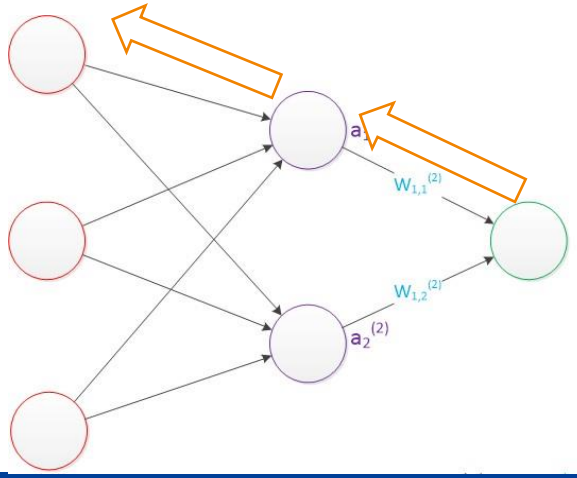
layer模块:
包括了数据输入层、全连接层、激活函数层、损失函数层等。

反向传输（BP）算法的作用是什么?

对于神经网络模型的优化实质上就是对**整体损失函数 L（成本函数）** 的优化

$$\cos t = L(W, B \mid (x_1, y_1) \cdots (x_N, y_N)) = \frac{1}{N} \sum_{i=1}^{N} L_i(W, B \mid (x_i, y_i))$$

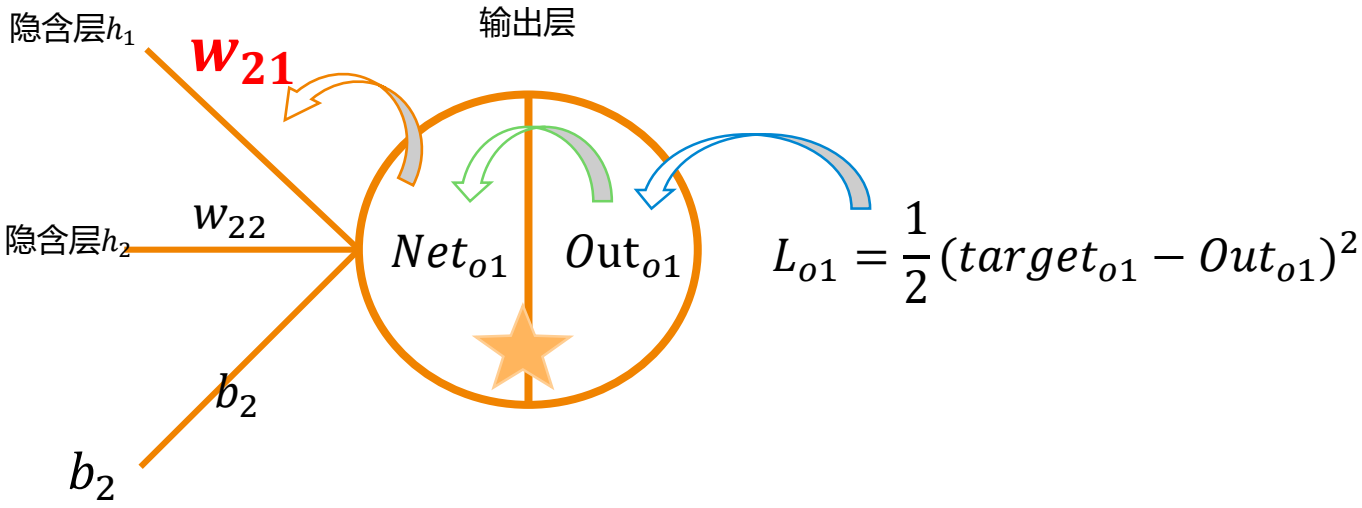
深度网络可能有上亿的参数需要优化，如何高效的求解出 L 对这上亿参数的偏导数，便成为一个难题，**反向传输（BP）算法即用来高效的计算这些参数的偏导数，进而得出成本函数（损失函数L）的梯度。下一步，便可以很自然地才用梯度下降算法对权重更新，以向最小值点前进。**



BP推导前置知识:

- 计算图
- 激活函数
- 梯度下降
- 链式求导
- 张量求导

隐含层—输出层的权值更新



$$L_{o1} = \frac{1}{2}(target_{o1} - Out_{o1})^2$$

以权重参数W_{21}为例，如果我们想知道W_{21}对整体误差产生了多少影响，可以用整体误差对W_{21}求偏导求出：（链式法则）

$$\frac{\partial L_{O1}}{\partial W_{21}} = \frac{\partial L_{O1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial W_{21}}$$

分别计算三个偏导的值再相乘

隐含层—输出层的权值更新

$$\frac{\partial L_{O1}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - Out_{o1})^{2-1} * -1 + 0 \ = -(target_{o1} - Out_{o1})$$

$$net_{o1} = W_{21} * out_{h1} + W_{22} * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial W_{21}} = 1 * out_{h1} * W_{21}^{(1-1)} + 0 + 0 = out_{h1}$$

$$out_{01} = \frac{1}{1 - e^{-net_{o1}}}$$

$$\frac{\partial out_{01}}{\partial net_{o1}} = out_{01}(1 - out_{01})$$

$$\frac{\partial L_{Total}}{\partial W_{21}} = \frac{\partial L_{O1}}{\partial out_{o1}} * \frac{\partial out_{01}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial W_{21}}$$

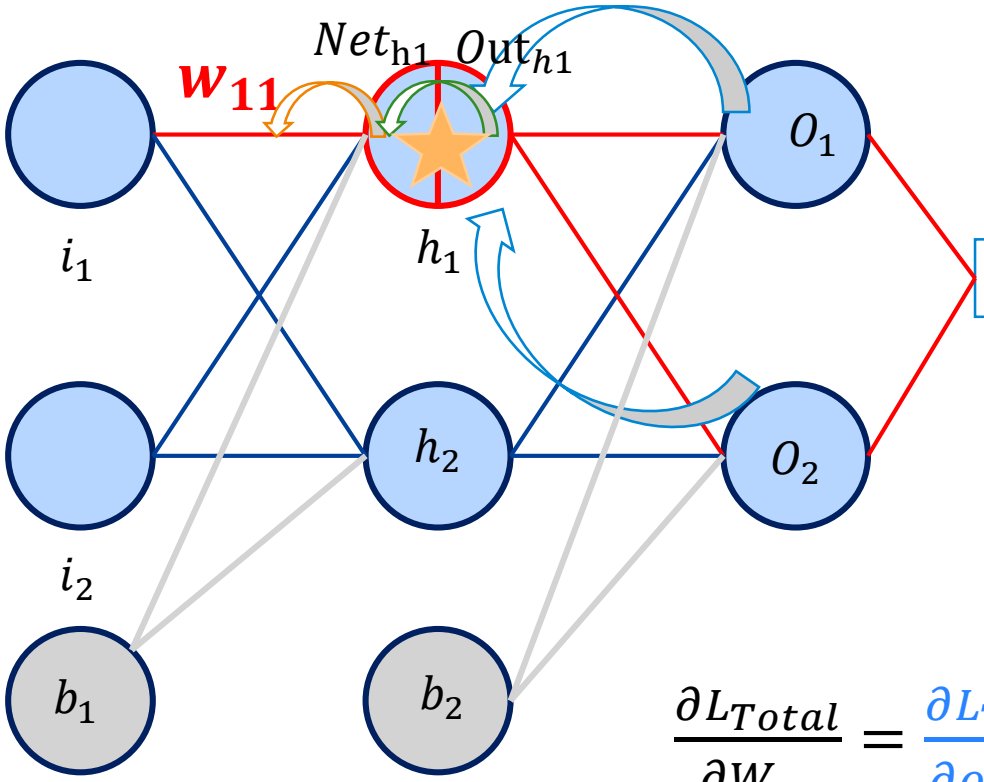
$$= -(target_{o1} - Out_{o1}) * out_{01}(1 - out_{01}) * out_{h1}$$

令 $\delta_{o1} = -(target_{o1} - Out_{o1}) * out_{01}(1 - out_{01})$

整体损失函数L对W_{21}的偏导公式为：$\frac{\partial L_{O1}}{\partial W_{21}} = (\pm)\delta_{o1} * out_{h1}$

最后我们来更新W_{21}的值：$W_{21 \cdot new}^{(l)} = W_{21}^{(l)} - \alpha \frac{\partial L_{O1}}{\partial W_{21}}$

隐藏层—隐藏层权值更新



$$L_{o1} = \frac{1}{2}(target_{o1} - Out_{o1})^2$$

$$L_{Total} = L_{o1} + L_{o2}$$

$$L_{o2} = \frac{1}{2}(target_{o2} - Out_{o2})^2$$

$$\frac{\partial L_{Total}}{\partial W_{11}} = \frac{\partial L_{Total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial W_{11}}$$

$$\frac{\partial L_{Total}}{\partial out_{h1}} = \frac{\partial L_{O1}}{\partial out_{h1}} + \frac{\partial L_{O2}}{\partial out_{h1}}$$

隐藏层—隐藏层权值更新

$$\frac{\partial L_{total}}{\partial out_{h1}} = \frac{\partial L_{O1}}{\partial out_{h1}} + \frac{\partial L_{O2}}{\partial out_{h1}}$$

$$\frac{\partial L_{O1}}{\partial out_{h1}} = \frac{\partial L_{O1}}{\partial net_{01}} * \frac{\partial net_{O1}}{\partial out_{h1}} = \left(\frac{\partial L_{O1}}{\partial out_{01}} * \frac{\partial out_{O1}}{\partial net_{01}}\right) * \frac{\partial net_{O1}}{\partial out_{h1}} = \left(\frac{\partial L_{O1}}{\partial out_{01}} * \frac{\partial out_{O1}}{\partial net_{01}}\right) * W_{11}$$

同理可得 $\frac{\partial L_{O2}}{\partial out_{h1}} = \left(\frac{\partial L_{O2}}{\partial out_{01}} * \frac{\partial out_{O1}}{\partial net_{01}}\right) * \frac{\partial net_{O1}}{\partial out_{h1}}$

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

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

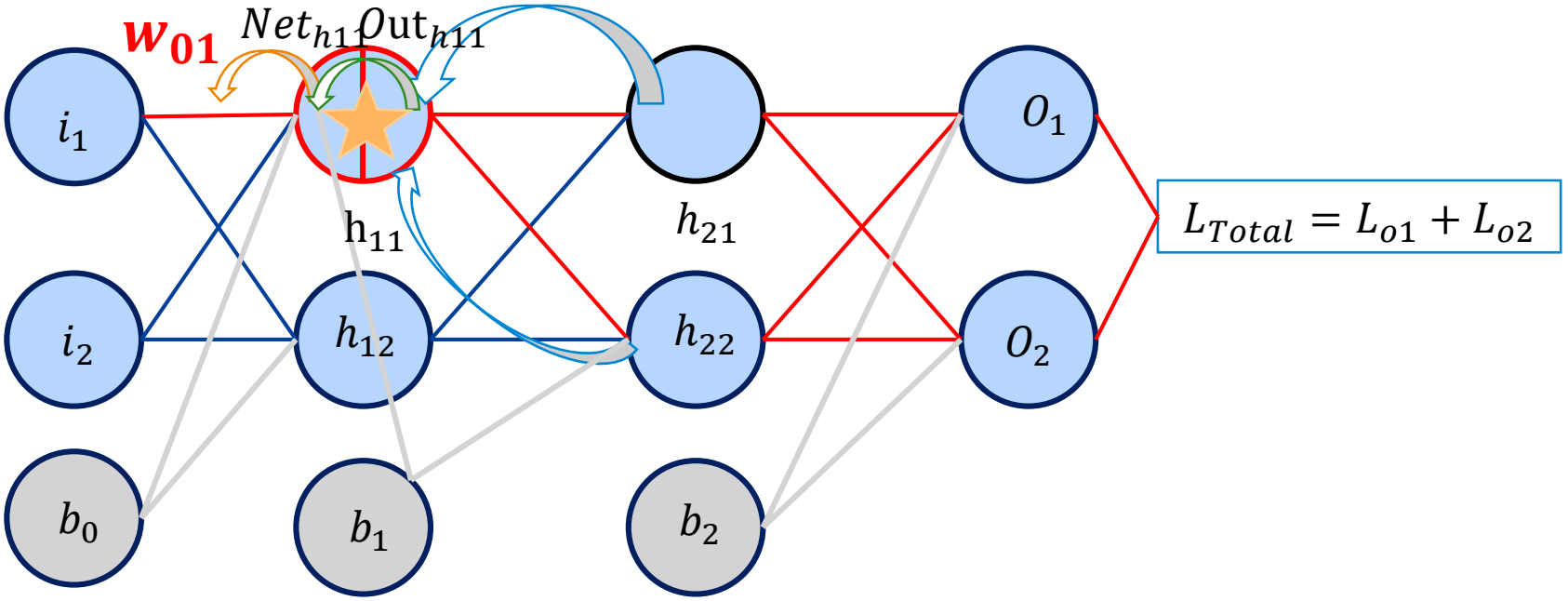
$$net_{h1} = W_{11} * i_1 + W_{12} * i_2 + b_1 * 1$$

$$\frac{\partial net_{h1}}{\partial W_{11}} = i_1 + 0 + 0$$

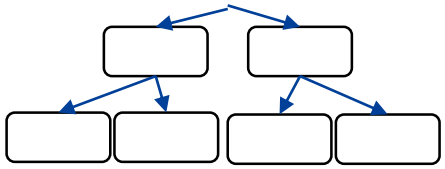
$$\frac{\partial L_{total}}{\partial out_{h1}} = \left(\sum_O \left(\frac{\partial L_{total}}{\partial out_O} * \frac{\partial out_O}{\partial net_O} * \frac{\partial net_O}{\partial out_{h1}}\right) * \frac{\partial out_{h1}}{\partial net_{h1}}\right) * \frac{\partial net_{h1}}{\partial W_{11}}$$

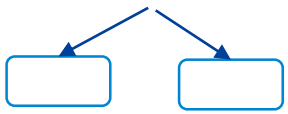
最后我们来更新W_{11}的值：$W_{11\cdot new}^{(l)} = W_{11}^{(l)} - \alpha \frac{\partial L_{Total}}{\partial W_{11}}$

隐藏层—输入层权值更新



$$\frac{\partial L_{Total}}{\partial W_{01}} = \frac{\partial L_{Total}}{\partial out_O} * \frac{\partial out_{O2}}{\partial out_{h11}} * \frac{\partial out_{h11}}{\partial net_{h11}} * \frac{\partial net_{h11}}{\partial W_{01}}$$





梯度计算（数学）

$$\frac{\partial L_{Total}}{\partial W_{01}} = \frac{\partial L_{Total}}{\partial out_O} * \frac{\partial out_{O2}}{\partial out_{h11}} * \cdots * \frac{\partial out_{h11}}{\partial net_{h11}} * \frac{\partial net_{h11}}{\partial W_{01}}$$

有两种计算梯度的方法：一种缓慢、近似但简单的方法（**数值梯度**），
以及一种快速、精确但更容易出错的方法，需要微积分（**解析梯度**）。

1 有限差分计算数值梯度

$$\frac{d}{d\theta}J(\theta) = \lim_{\varepsilon \to 0} \frac{J(\theta+\varepsilon) - J(\theta-\varepsilon)}{2\varepsilon} \qquad \frac{d}{d\theta}J(\theta) = \frac{J(\theta+epsilon) - J(\theta-epsilon)}{2epsilon}$$

$epsilon$通常被设置为一个很小的常量，比如$10^{(-4)}$

2 微积分计算梯度

需要微积分的更多知识

update_method模块：
定义了学习率、梯度的更新方法。

梯度下降法（回顾）

梯度下降法(Gradient Descent)是一种常见的、用于寻找**函数极小值的一阶迭代**优化算法，又称为**最速下降**(Steepest Descent)

$$\theta := \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$$

单变量线性回归(Univariate Linear Regression)模型 $y = wx + b$

Loss: $Z(w,b) = \frac{1}{2}(y_{w,b}(x^i) - y^i)^2 = \frac{1}{2}(wx^i + b - y^i)^2$

目标函数最小的参数值: $(\hat{w}, \hat{b}) = \arg\min_{w,b} Z(w,b)$

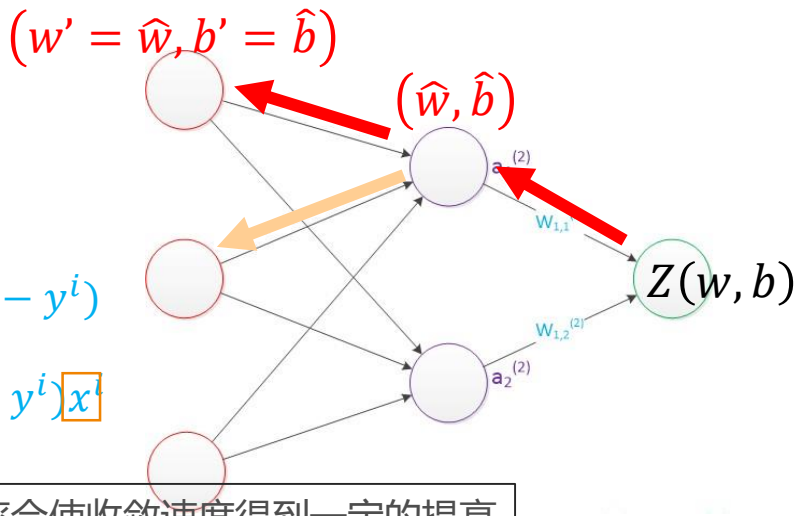
梯度求解+参数更新:

（偏导数的求导）

$$\hat{w} = w - \alpha \frac{\partial Z(w,b)}{\partial w} = w - \alpha(wx + bx^i - y^i)$$

$$\hat{b} = b - \alpha \frac{\partial Z(w,b)}{\partial b} = b - \alpha(w + bx^i - y^i)x^i$$

更新项:



学习率 η （步长）决定了梯度下降的**收敛速度**，合适的学习率会使收敛速度得到一定的提高。

update_method模块：定义了学习率、梯度的更新方法。

梯度下降法种类

批梯度下降法 BGD

批梯度下降法根据整个数据集求取损失函数的梯度，来更新参数θ 的值。它是指在每一次迭代时使用**所有**样本来进行梯度的更新。公式如下：

$$\theta := \theta - \alpha \frac{1}{m} \sum_{i=1}^{m} \frac{\partial J(\theta, x^i, y^i)}{\partial \theta}$$

Loss:

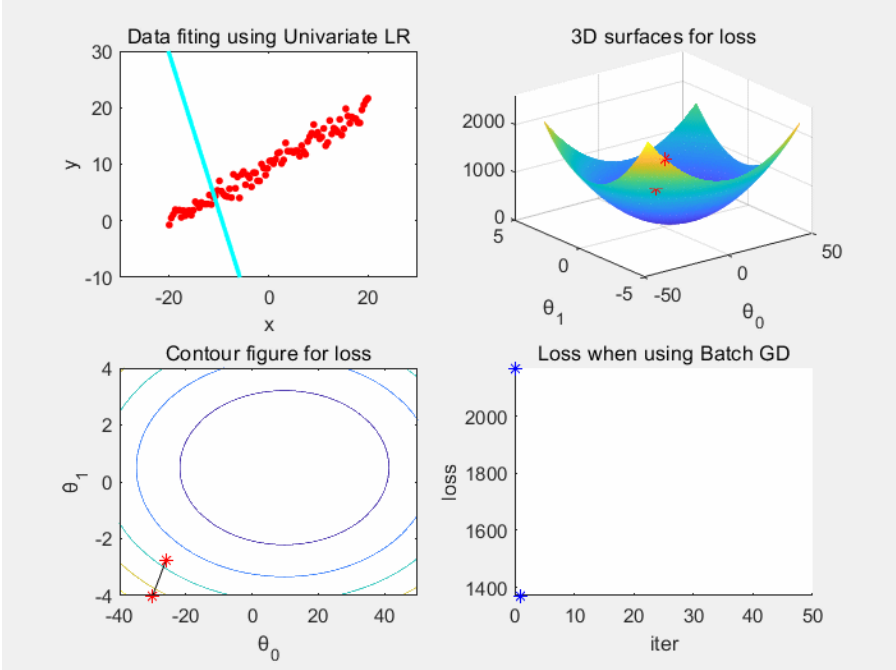
$$Z(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (y_{w,b}(x^i) - y^i)^2$$

$$= \frac{1}{2\mathrm{m}} \sum_{i=1}^{m} (wx^i + b - y^i)^2$$

梯度求解+参数更新：

$$\hat{w} = \mathrm{w} - \frac{\partial Z(w,b)}{\partial w} = w - \alpha \frac{1}{m} \sum_{i=1}^{m} (w + bx^i - y^i)$$

$$\hat{b} = b - \frac{\partial Z(w,b)}{\partial b} = b - \alpha \frac{1}{m} \sum_{i=1}^{m} (w + bx^i - y^i) x^i$$



优点：考虑到了全局数据，更新稳定，震荡小，每次更新都会朝着正确的方向。

缺点：如果数据集比较大，会造成大量**时间和空间开销。**

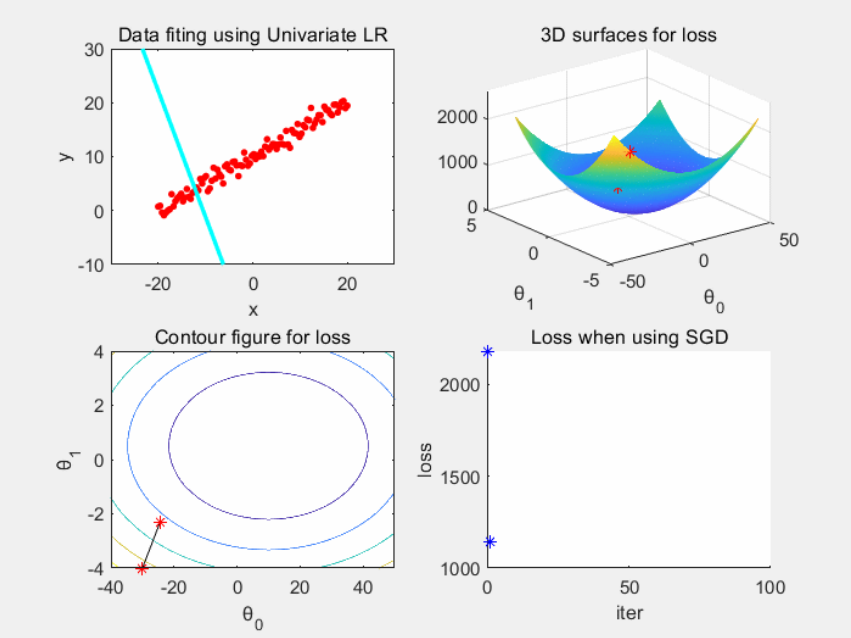
update_method模块：
定义了学习率、梯度的更新方法。

梯度下降法种类

随机梯度下降法 SGD

因为批梯度下降所需要的时间空间成本大，我们考虑随机抽取一个样本来获取梯度，更新参数 θ 。

$$\theta := \theta - \alpha \cdot \frac{\partial J(\theta, x^i, y^i)}{\partial \theta}$$



步骤如下：

- 随机打乱数据集。
- 对于每个样本点，依次迭代更新参数 θ 。
- 重复以上步骤直至损失足够小或到达迭代阈值。

优点：

（1）由于不是在全部训练数据上的损失函数，而是在每轮迭代中，随机优化某一条训练数据上的损失函数，这样每一轮参数的更新**速度大大加快**。

缺点：

（1）**准确度下降**。由于即使在目标函数为强凸函数的情况下，SGD仍旧无法做到线性收敛。
（2）可能会收敛到**局部最优**，由于单个样本并不能代表全体样本的趋势。
（3）**不易于并行实现**。

update_method模块：
定义了学习率、梯度的更新方法。

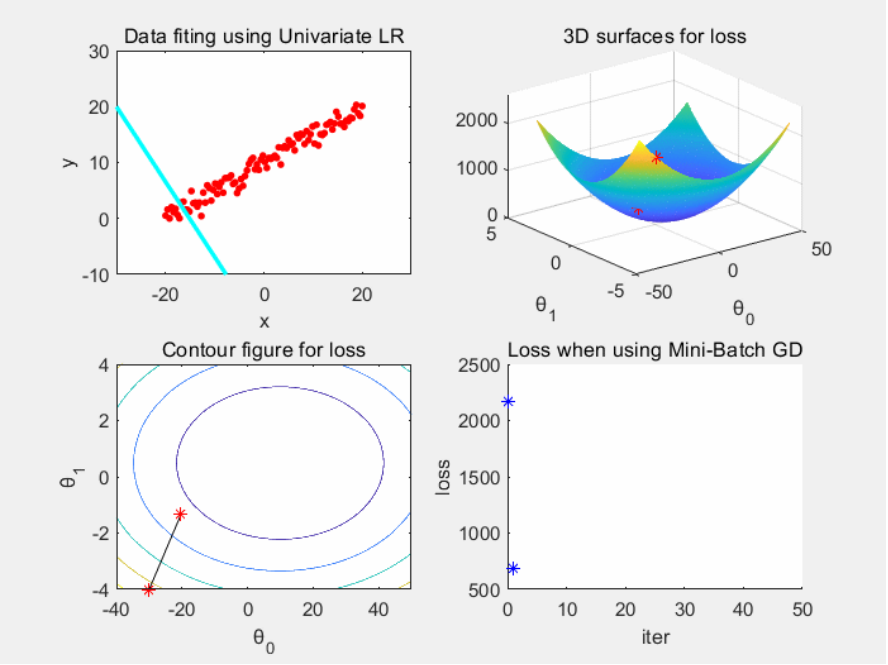
梯度下降法种类

小批量（随机）梯度下降法 mini-batch GD

随机梯度下降收敛速度快，但是波动较大，在最优解处出现波动很难判断其是否收敛到一个合理的值。而批梯度下降稳定但是时空开销很大。

引入小批量梯度下降对二者进行折衷，在每轮迭代过程中使用n个样本来训练参数：

$$\theta := \theta - \alpha \frac{1}{n} \sum_{i=1}^{n} \frac{\partial J(\theta, x^i, y^i)}{\partial \theta}$$



优点：

（1）通过矩阵运算，每次在一个batch上优化神经网络参数并不会比单个数据慢太多。

（2）每次使用一个batch可以大大减小收敛所需要的迭代次数，同时可以使收敛到的结果更加接近梯度下降的效果。（比如随机梯度下降迭代的30W，设置batch_size=100时，需要迭代3000次，远小于SGD的30W次）

（3）可实现并行化。

缺点：

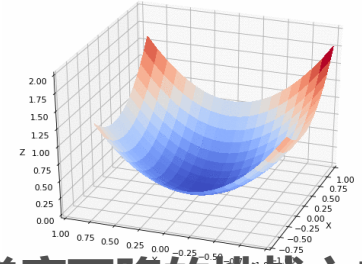
（1）batch_size的不当选择可能会带来一些问题。

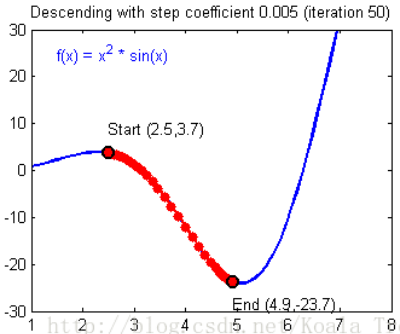
（2）局部震荡问题需要引入学习率-动量概念，跳出局部最小值。

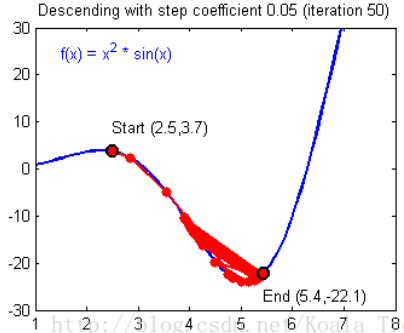
如何加快收敛速度?

梯度下降算法的调优方法

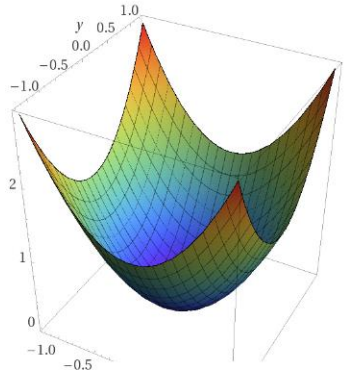
梯度下降的挑战之一：局部震荡，抖动

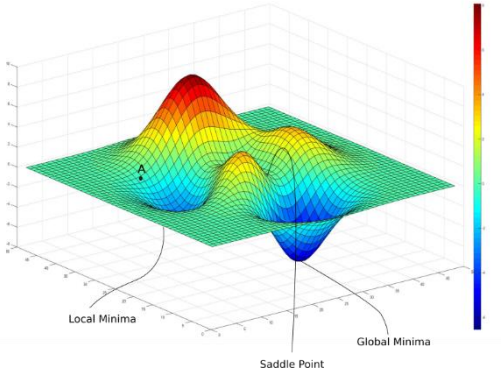


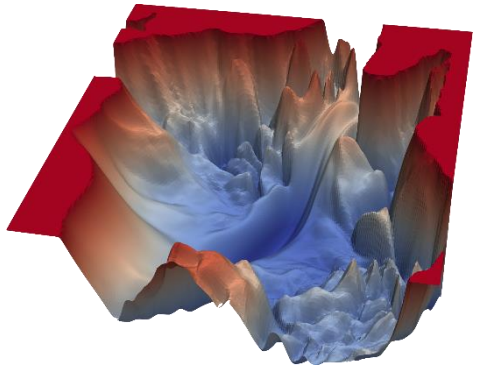




梯度下降的挑战之二：局部极小值

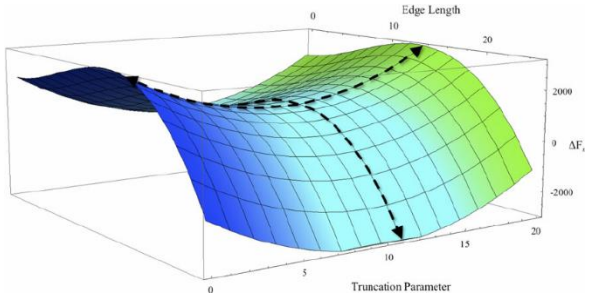






梯度下降的挑战之三：鞍点

鞍点处，梯度在一个方向（X）上是极小值，在另一个方向上则是极大值。如果沿着x方向的等值曲面比较平，那么梯度下降会沿着Y方向来回振荡，造成收敛于最小值的错觉。



梯度下降算法的调优方法

梯度下降的挑战之四：梯度消失和梯度爆炸

神经网络的参数采用随机初始化的方法，W取值往往小于1，梯度$\frac{\partial L}{\partial w_1}$随着网络深度的增加，传递到第i层时，数值小于第$i+1$层的$\frac{1}{4}$。如果网络非常深，就可能会使得前几层网络参数的**梯度接近于0**，也就发生了梯度消失现象。如果初始化参数非常大，就可能会使得前几层网络参数的**梯度非常大**，也就发生了梯度爆炸现象。

梯度下降的挑战之五：梯度不精确

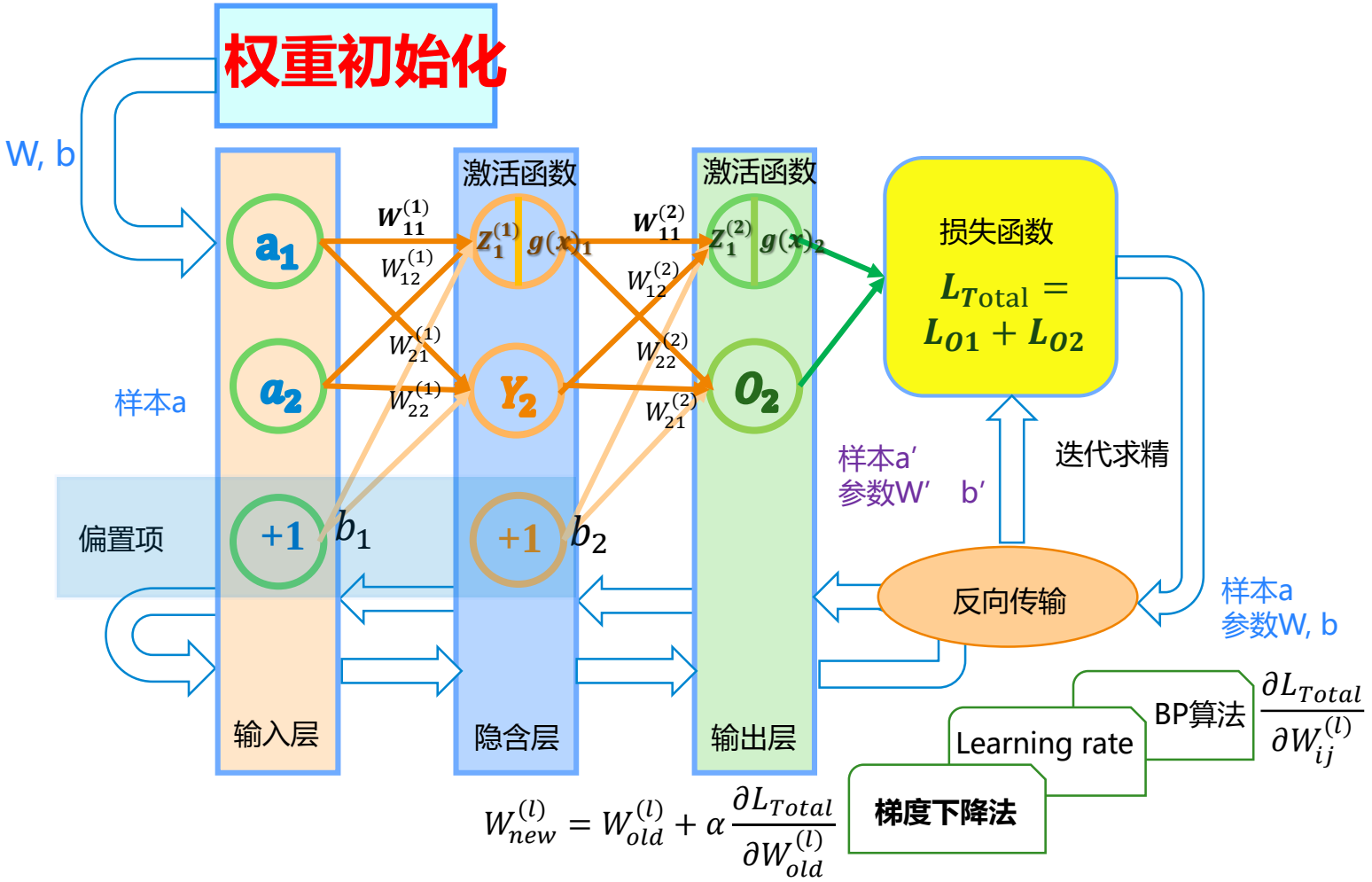
随机梯度下降与小批量梯度下降这种**不精确的梯度**，也就导致了训练的稳定性较差，但梯度不精确有时也可以看作是防止过拟合以及逃离局部最优或鞍点的方法。机器学习终究不是一个最优化问题，但其却要依靠优化手段来完成机器学习任务。

梯度下降的挑战之六：优化理论的局限性

一些理论结果显示，很多针对神经网络而设计的**优化算法有着局限性**，但在实践中，这些理论结果却很少对神经网络产生影响。这也是相比于其他机器学习算法而言，神经网络更像是一个黑盒，神经网络中存在着大量的训练技巧，这也使得训练神经网络更像是艺术而非科学。

权重初始化：

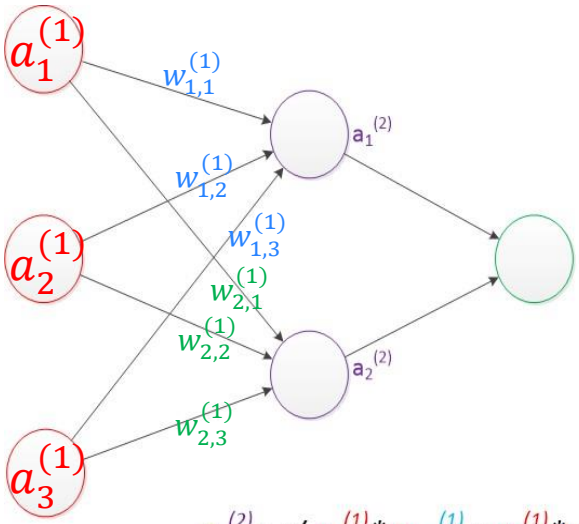
layer模块：
包括了数据输入层、全连接层、激活函数层、损失函数层等。



function_for_layer
模块：
定义了不同的激活函数、损失函数

参数初始化方法

为什么要给网络参数赋初值？ w_0 和 b_0



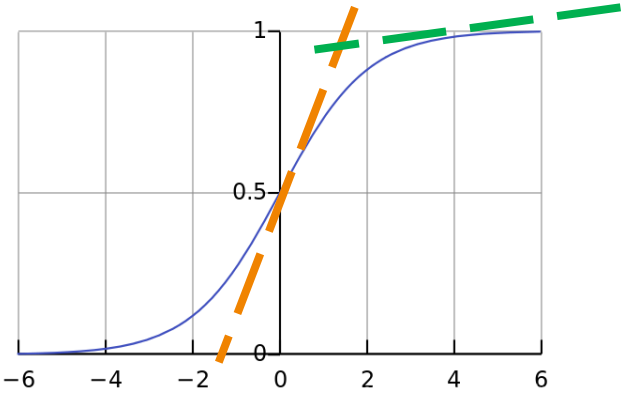
$$a_1^{(2)} = g(a_1^{(1)} * w_{1,1}^{(1)} + a_2^{(1)} * w_{1,2}^{(1)} + a_3^{(1)} * w_{1,3}^{(1)})$$

随机初始化参数的一个问题是如何选择随机初始化的区间。

“对称性”问题：

- 假设每个w初始都为0或者1（相同的恒等值）
- 则a_1, a_2相等，
- 向前传递到z_i也是0
- 调整后的梯度也相等
- 第二次迭代也同理

有100个单元与1个单元没有差异。
由此，一般将参数设置为**随机值**。



function_for_layer
模块：
定义了不同的激活函数、损失函数

参数初始化要求

常见的网络参数赋初值方法：

参数初始化的几点要求

（1）参数不能全部初始化为0，也不能全部初始化同一个值，为什么，请参见“对称问题”；

（2）最好保证参数初始化的均值为0，正负交错，正负参数大致上数量相等(似乎就是随机？)；

（3）初始化参数不能太大或者是太小，参数太小会导致特征在每层间逐渐缩小而难以产生作用，参数太大会导致数据在逐层间传递时逐渐放大而导致梯度消失发散，不能训练。

初始化一个深层神经网络时，一个比较好的初始化策略是保持每个神经元输入和输出的*方差*一致。

function_for_layer
模块：
定义了不同的激活函数、损失函数

参数初始化方法

1.基于固定方差的参数初始化

- 高斯分布初始化：参数从一个固定均值(比如0)和固定方差(比如0.01)的高斯分布进行随机初始化。
- 均匀分布：在[r,-r]内采用均匀分布初始化，其方差为$Var(x) = \frac{(b-a)^2}{12}$，得到$r = \sqrt{3\sigma^2}$

2. 基于方差缩放的参数初始化

- 2.1 Xavier初始化（也称为Glorot初始化，因为发明人为Xavier Glorot）
- 2.2 He初始化（也称为Kaiming初始化）

3. 正交初始化

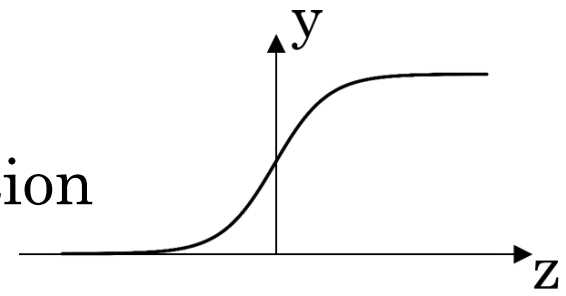
为了避免梯度消失或爆炸，希望误差项在反向传播具有范数保持性（Norm-Preserving），即 $||\delta^{(l-1)}||^2 = ||\delta^{(l)}|| = ||(W^{(l)})^T\delta^{(l)}||^2$

将$W^{(l)}$初始化为正交矩阵，即 $W^{(l)}(W^{(l)})^T = I$，这种方法称为正交初始化（Orthogonal Initialization）

Loss function

$\hat{y} = \sigma(wx + b)$

where $\sigma(z) = \frac{1}{1+e^{-z}}$, called **sigmoid** function



Given $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

Loss (error) function:

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\left(y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})\right)$$

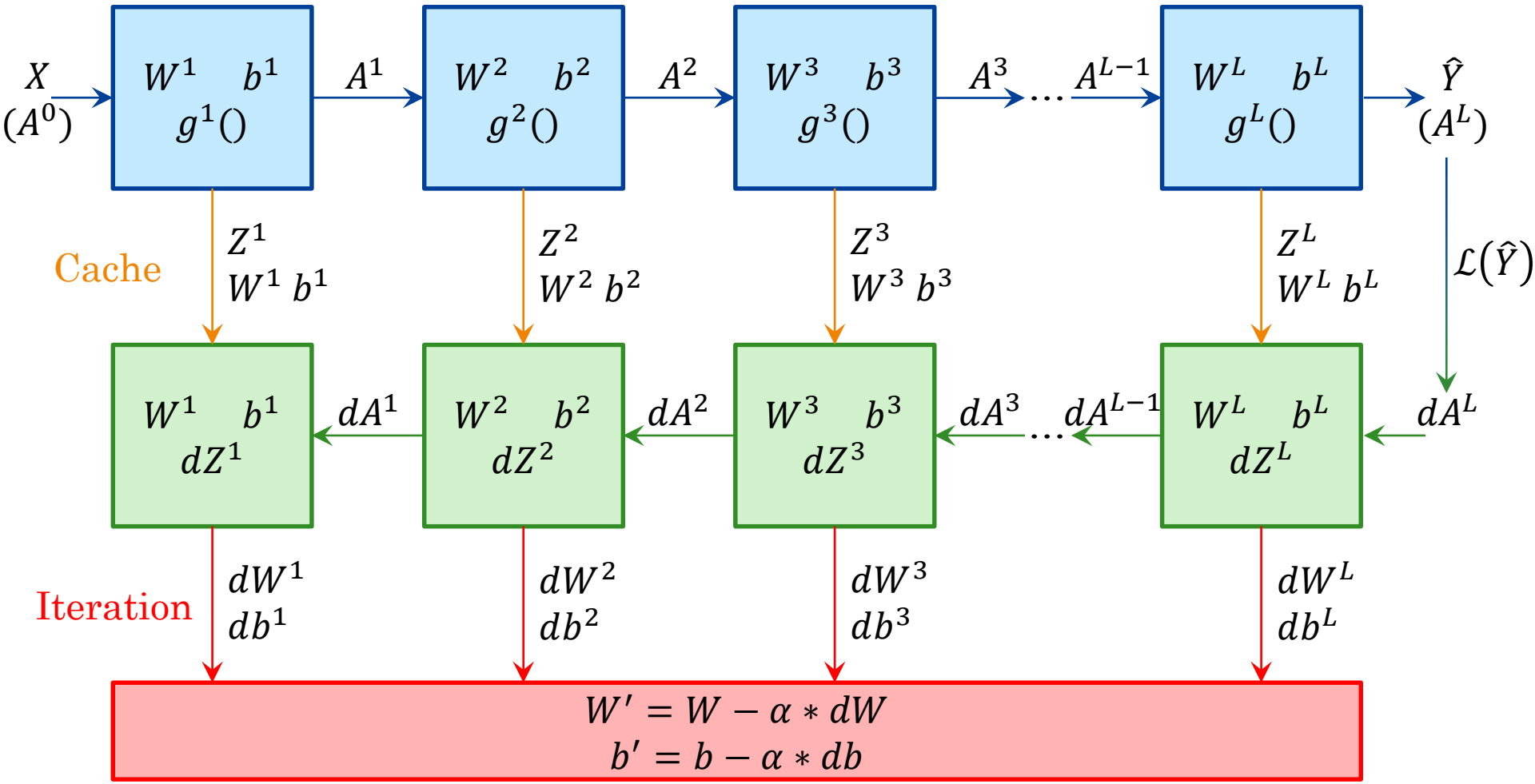
Loss function of samples

Cost function:

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

$$= -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Implement FP and BP



Hyperparameters

- Learning rate
- Iterations
- Hidden layers
- Hidden Units
- Activation function

Deep learning is a very empirical process!

- Momentum term
- Mini-batch size
- Regularization parameters

深度学习算法基础 – CNN

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

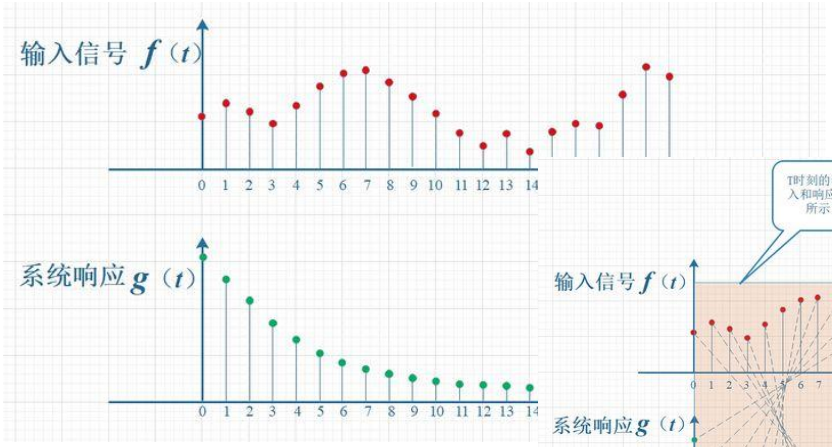
Computer vision

Cat? (0/1)

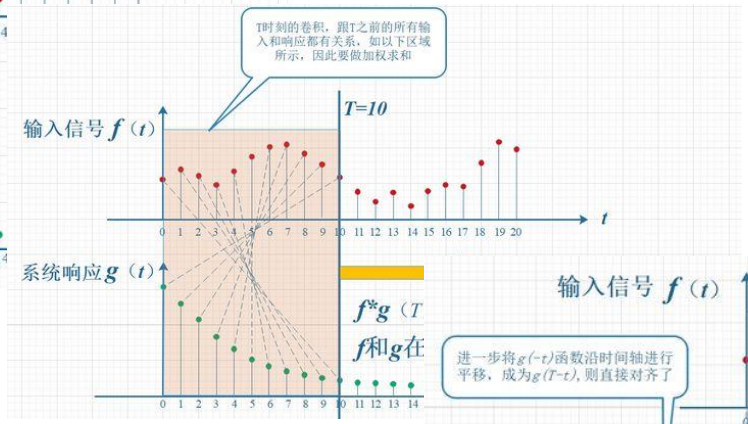
64×64

1000×1000

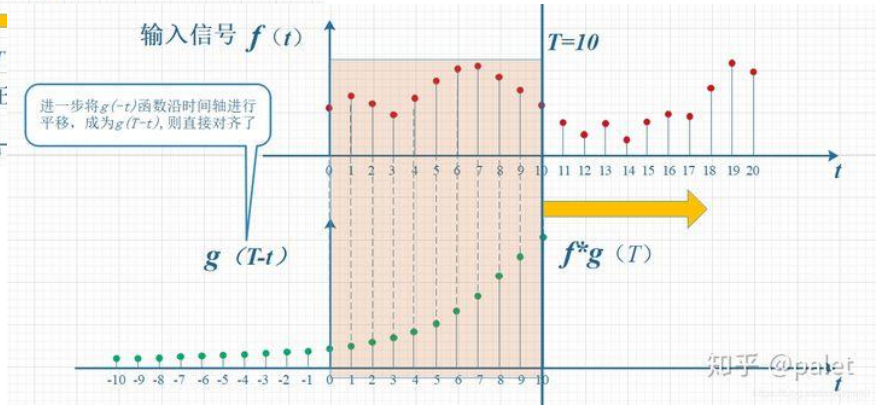
卷积的定义



输入信号$f(t)$，系统响应$g(t)$
$t=0$时刻的输入$f(0)$的值在$t=T$时刻将衰减为$f(0)g(T)$

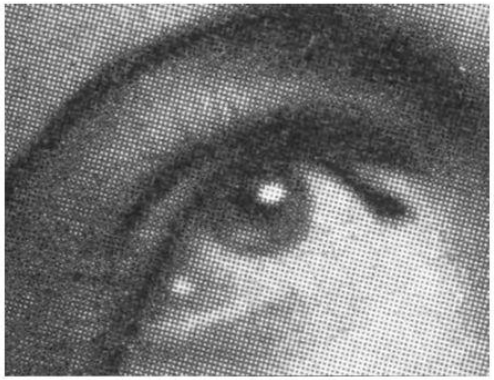


求$t=10$时刻的输出的信号值



卷积的定义

- 二维离散卷积：$(f*g)(u,v)=\sum_i\sum_i f(i,j)g(u-i,v-j)=\sum_i\sum_j a_{i,j}b_{u-i,v-j}$



$$\Longrightarrow\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \cdots & a_{0,n} \\ a_{1,0} & a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,0} & a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m,0} & a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

$$f=\begin{bmatrix} a_{u-1,v-1} & a_{u-1,v} & a_{u-1,v+1} \\ a_{u,v-1} & a_{u,v} & a_{u,v+1} \\ a_{u+1,v-1} & a_{u+1,v} & a_{u+1,v+1} \end{bmatrix}$$

$$g=\begin{bmatrix} b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & b_{0,0} & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{bmatrix}$$

$$g'=\begin{bmatrix} b_{1,1} & b_{1,0} & b_{1,-1} \\ b_{0,1} & b_{0,0} & b_{0,-1} \\ b_{-1,1} & b_{-1,0} & b_{-1,-1} \end{bmatrix}$$

$$f*g(u,v)=a_{u-1,v-1}\times b_{1,1}+a_{u-1,v}\times b_{1,0}+a_{u-1,v+1}\times b_{1,-1}$$

$$+a_{u,v-1}\times b_{0,1}+a_{u,v}\times b_{0,0}+a_{u,v+1}\times b_{0,-1}$$

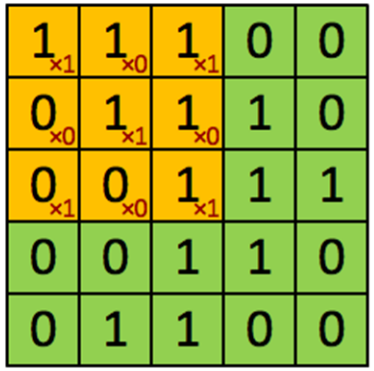
$$+a_{u+1,v-1}\times b_{-1,1}+a_{u+1,v}\times b_{-1,0}+a_{u+1,v+1}\times b_{-1,-1}$$

卷积层

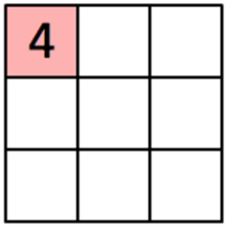
用卷积核在原图上滑动，进行卷积运算，得到特征图feature map。
卷积的本质：将原图中符合卷积核特征的特征提取出来，展示在feature map里面。

1	0	1
0	1	0
1	0	1

过滤器/卷积核
(filter/kernel)



Image



Convolved
Feature

卷积层

边界检测（edge detection）：
图的中间两个颜色的分界线就是我们要检测的边界

1 ×1	1 ×0	1 ×-1	0	0	0
1 ×1	1 ×0	1 ×-1	0	0	0
1 ×1	1 ×0	1 ×-1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0

6×6

*

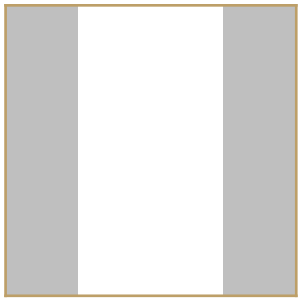
1	0	-1
1	0	-1
1	0	-1

过滤器/卷积核
(filter/kernel)

=

0	3	3	0
0	3	3	0
0	3	3	0
0	3	3	0

特征图
(feature map)



卷积层

检测纵向边缘

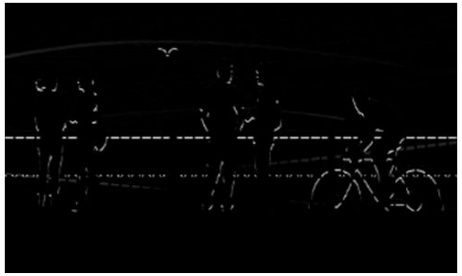
过滤器/卷积核
(filter/kernel)

1	0	-1
1	0	-1
1	0	-1

特征图
(feature map)

检测横向边缘

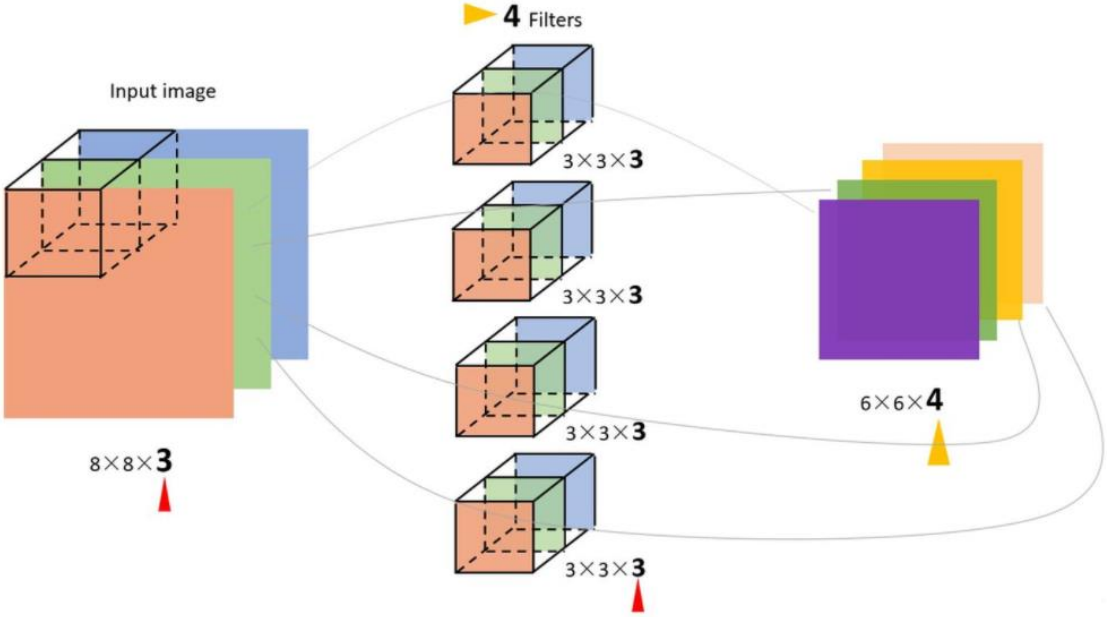
1	1	1
0	0	0
-1	-1	-1



卷积核的大小和数字都不局限于上面两种，所以可以检测出各种各样的特征，并且特征也不局限于我们肉眼可以观察出的特征。

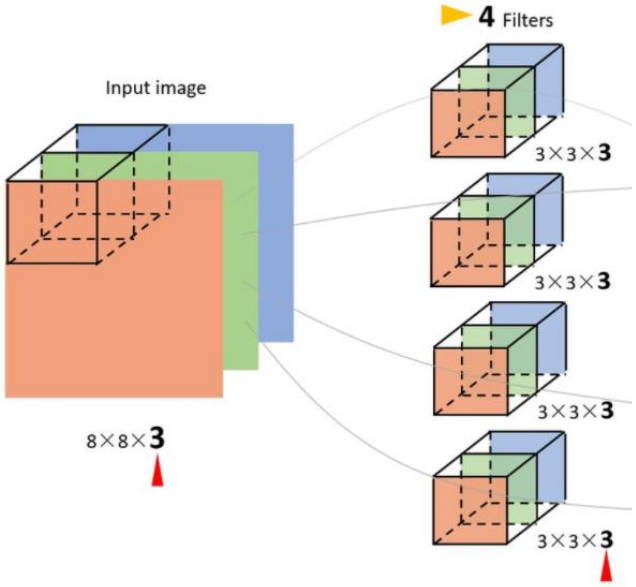
卷积层

彩色图像，一般都是红、绿、蓝三个图层的叠加的，我们把每一个图层叫一个通道（channel）。因此输入数据的维度一般有三个：（长，宽，通道）。 比如一个8×8的RGB图片，维度就是(8,8,3)。多图层意味着每个图层都可以提取特征，每一个卷积核也有多个通道，每一个通道提取一个图层的特征。



卷积层

彩色图像，一般都是红、绿、蓝三个图
此输入数据的维度一般有三个：（长，
多图层意味着每个图层都可以提取特征
征。



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	1	1	0	2	0
0	2	2	2	2	1	0
0	1	0	0	2	0	0
0	0	1	1	0	0	0
0	1	2	0	0	2	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	1	0	2	2	0	0
0	0	0	0	2	0	0
0	1	2	1	2	1	0
0	1	0	0	0	0	0
0	1	2	1	1	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	2	0	0	0
0	1	0	0	1	0	0
0	0	2	1	0	1	0
0	0	1	2	2	2	0
0	2	1	0	0	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	1	0
0	1	0
0	1	1

w0[:,:,1]

-1	-1	0
0	0	0
0	-1	0

w0[:,:,2]

0	0	-1
0	1	0
1	-1	-1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

1	1	-1
-1	-1	1
0	-1	1

w1[:,:,1]

0	1	0
-1	0	-1
-1	1	0

w1[:,:,2]

-1	0	0
-1	0	1
-1	0	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

6	7	5
3	-1	-1
2	-1	4

o[:,:,1]

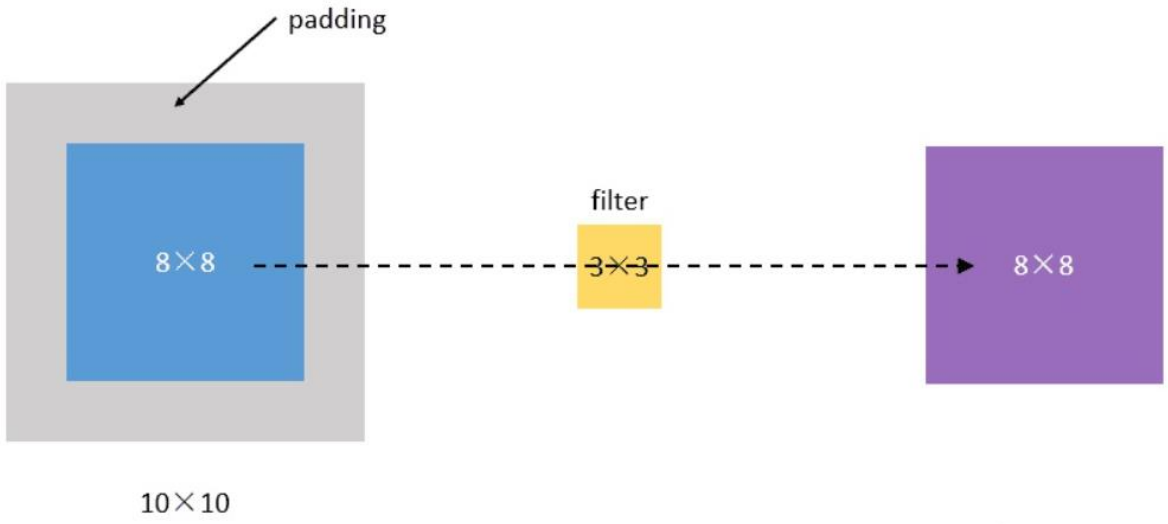
2	-5	-8
1	-4	-4
0	-5	-5

toggle movement

步长(stride)

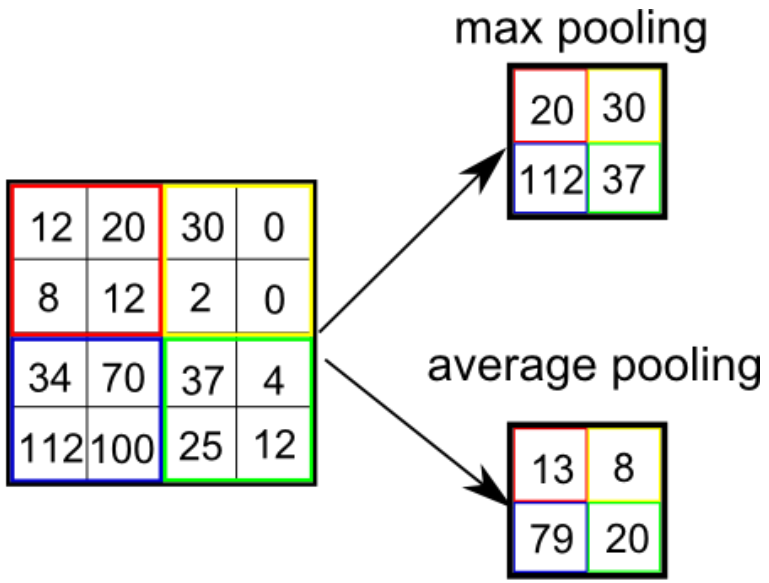
Padding:防止卷积边缘信息丢失

每次卷积前，先给图片周围都补一圈空白，能投让卷积之后图片跟原来一样大，同时，原来的边缘也被计算了更多次。



池化层

池化（Pooling）也叫做下采样（subsampling），用一个像素代替原图上邻近的若干像素，在保留feature map特征的同时压缩其大小。



优点：

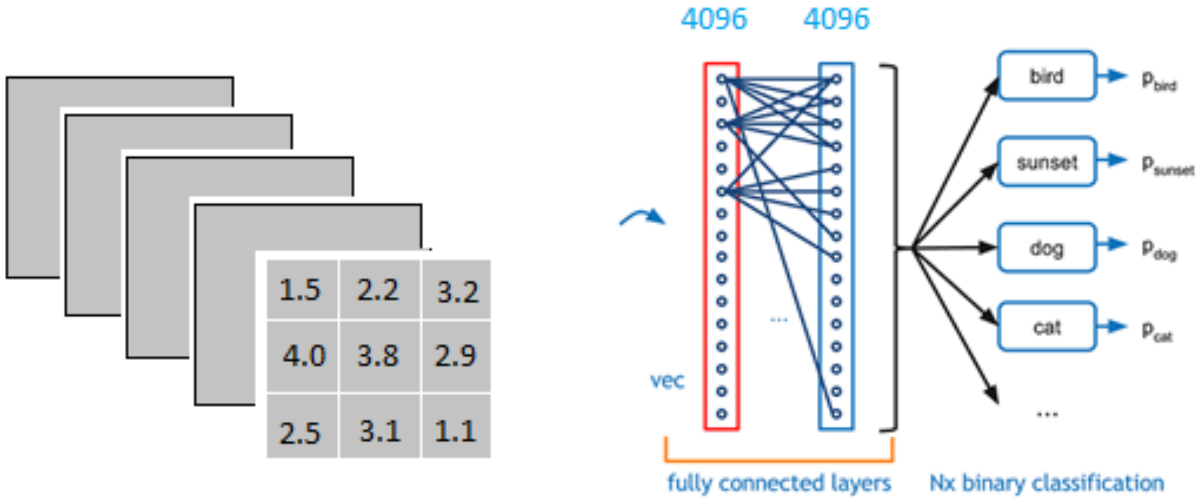
- 降低图像尺寸
- 一定程度上的平移、旋转不变性（在图像识别中，重要的不是显著特征的绝对位置而是相对的位置。防止过拟合、过学习）

- 均值池化：主要用来抑制邻域值之间差别过大，造成的方差过大。
- 最大池化：能够抑制网络参数误差造成的估计均值偏移的现象。在计算机视觉中：对纹理的提取较好

全连接层（Fully connected layers, FC 层）

全连接层之前的作用是**提取特征**，全理解层的作用是**分类**

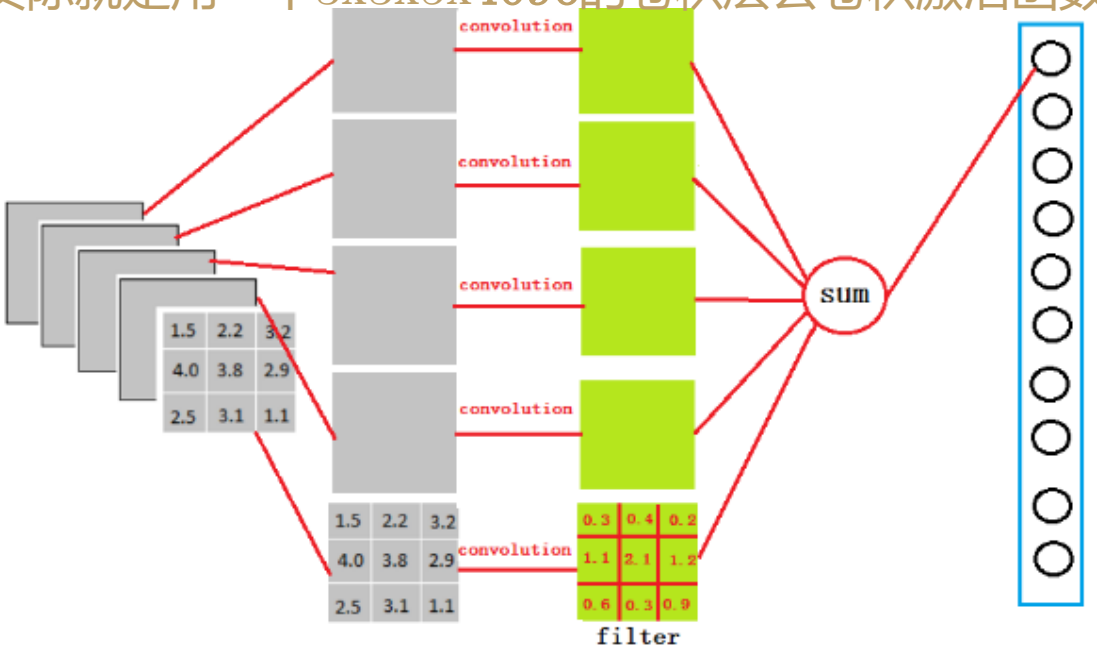
把输出转化为一维的形式



把3x3x5的输出，转换成1x4096的形式

全连接层

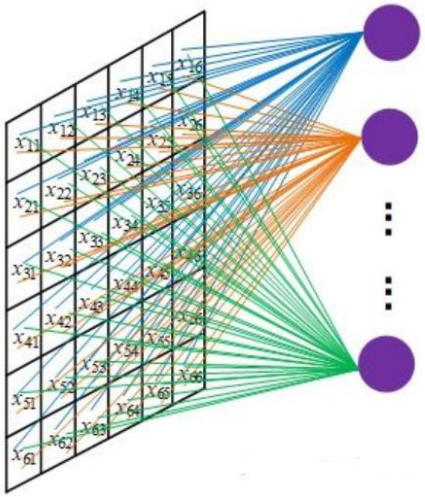
我们用一个3x3x5的filter 去卷积激活函数的输出，得到的结果就是一个fully connected layer 的一个神经元的输出，这个输出就是一个值，因为我们有4096个神经元，我们实际就是用一个3x3x5x4096的卷积层去卷积激活函数的输出。



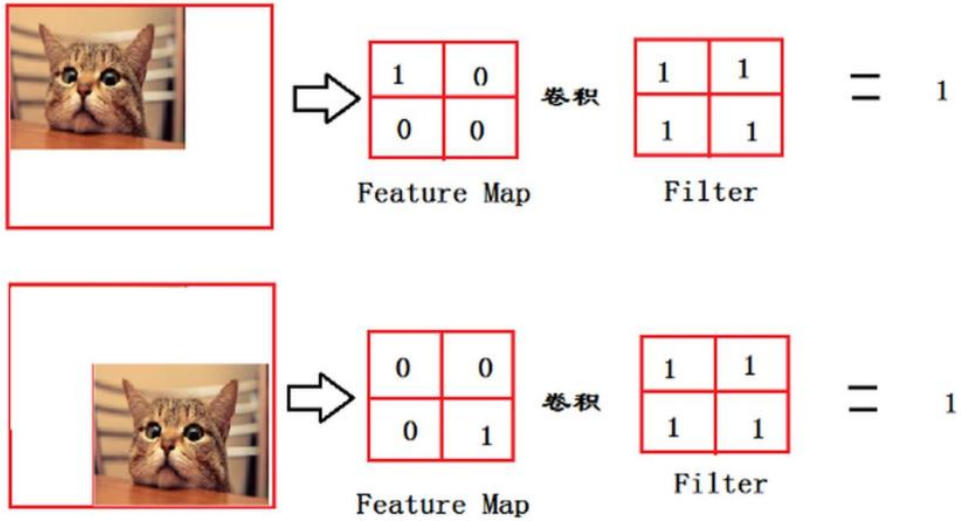
全连接层参数很多

卷积神经网络 VS. 传统神经网络

1.参数共享机制（parameters sharing）



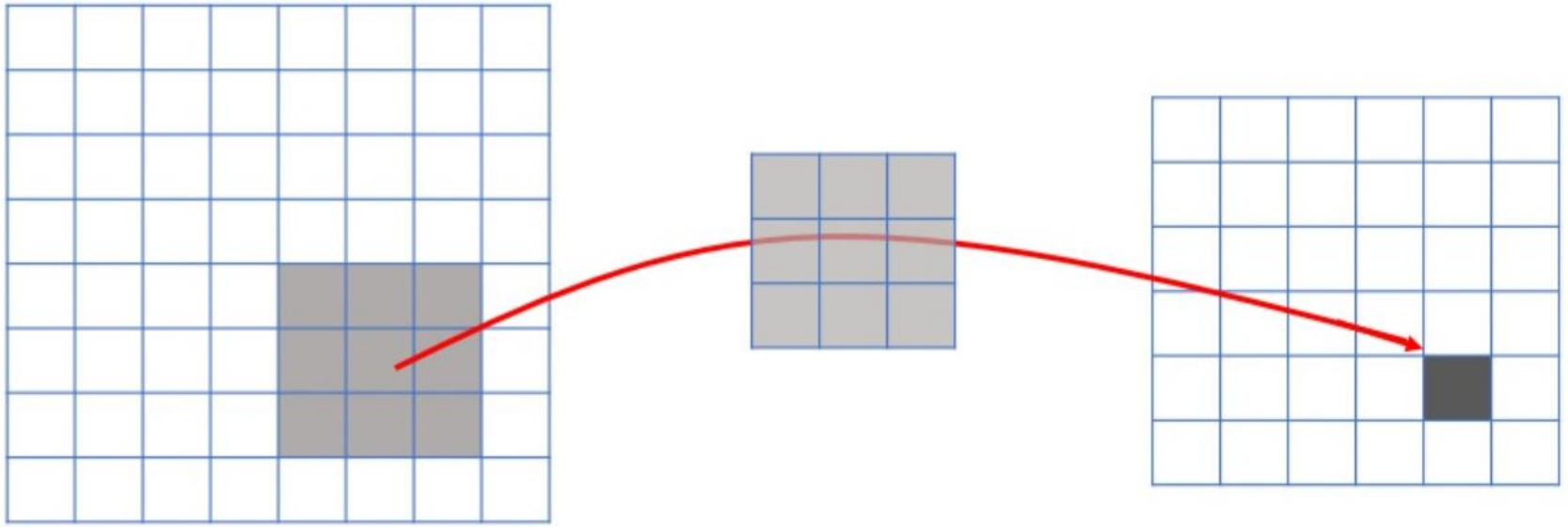
传统神经网络：6*6*n个参数，n为输出



让我们的网络的参数数量大大地减少。这样，我们可以用较少的参数，训练出更加好的模型，典型的事半功倍，而且可以有效地 **避免过拟合**。 同样，由于卷积核的参数共享，即使图片进行了一定的平移操作，我们照样可以识别出特征，这叫做 **“平移不变性”**。

卷积神经网络 VS. 传统神经网络

2.连接的稀疏性（sparsity of connections）

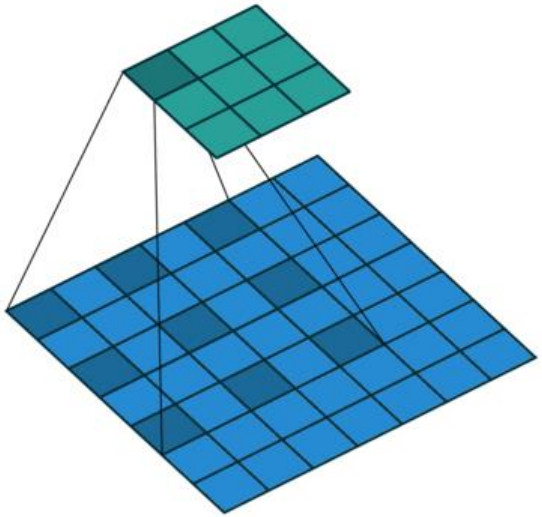


由卷积的操作可知，输出图像中的任何一个单元，只跟输入图像的一部分有关系，而传统神经网络中，由于都是全连接，所以输出的任何一个单元，都要受输入的所有的单元的影响。这样无形中会对图像的识别效果大打折扣。

卷积层

扩张卷积（Dilated Convolution）

解决问题：计算资源有限时处理大图片



扩张卷积（Dilated Convolution）也被称为空洞卷积或者膨胀卷积，是在标准的卷积核中注入空洞，以此来增加模型的感受野（reception field）。

相比原来的正常卷积操作，除了卷积核大小，步长和填充外，扩张卷积多了一个参数：扩张率（dilation rate），指的是卷积核的点的间隔数量，比如常规的卷积操作dilatation rate为1。扩张率为2的3x3卷积核与5x5卷积核具有相同的感受野，而仅使用9个参数。

优点：

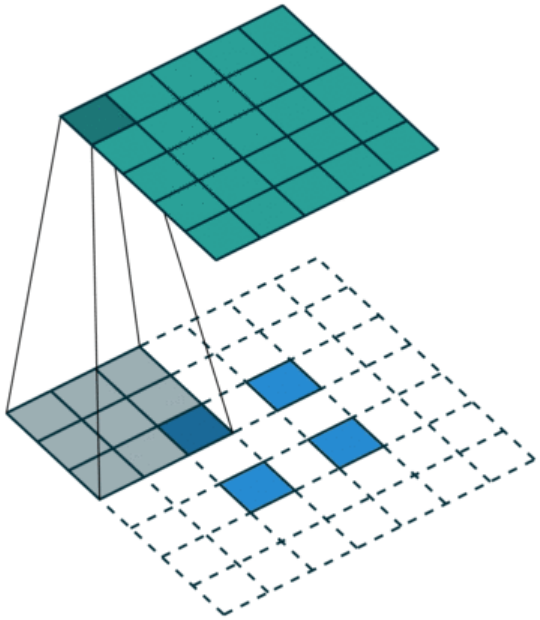
1. **不丢失分辨率（像素数目不变）的情况下扩大感受野。**
2. **调整扩张率获得多尺度信息。**

当网络层需要较大的感受野，但计算资源有限而无法提高卷积核数量或大小时，可以考虑空洞卷积。
不适合小图片

卷积层

转置卷积（transposed Convolutions）

解决问题：需要扩大图片尺寸，提高分辨率



$$input = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad kernel = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Stride=2

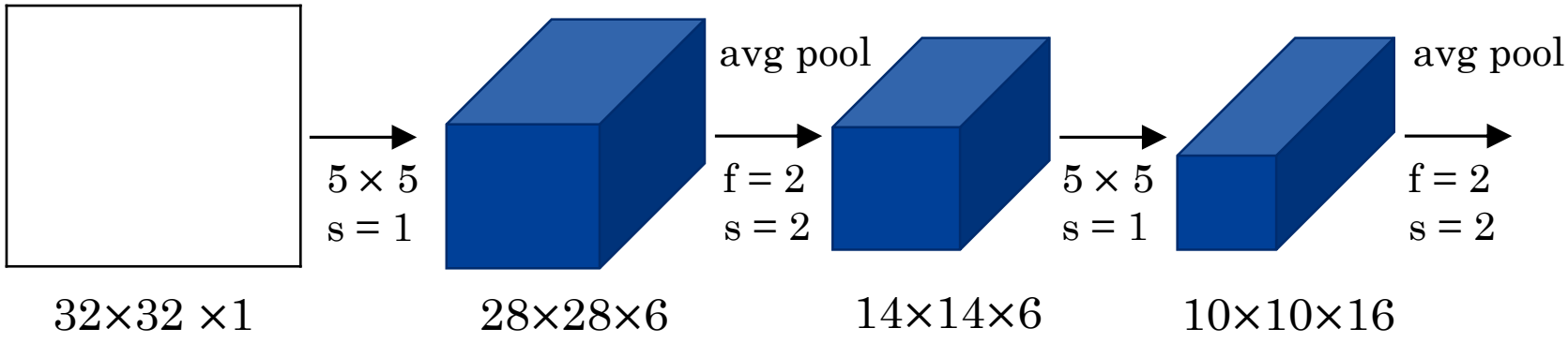
$$input_{pad} = \begin{bmatrix} 1 & 0 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 5 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 8 & 0 & 9 \end{bmatrix} \quad output = \begin{bmatrix} 1 & 0 & 2 & 0 & 3 \\ 0 & 6 & 0 & 8 & 0 \\ 4 & 0 & 5 & 0 & 6 \\ 0 & 12 & 0 & 14 & 0 \\ 7 & 0 & 8 & 0 & 9 \end{bmatrix}$$

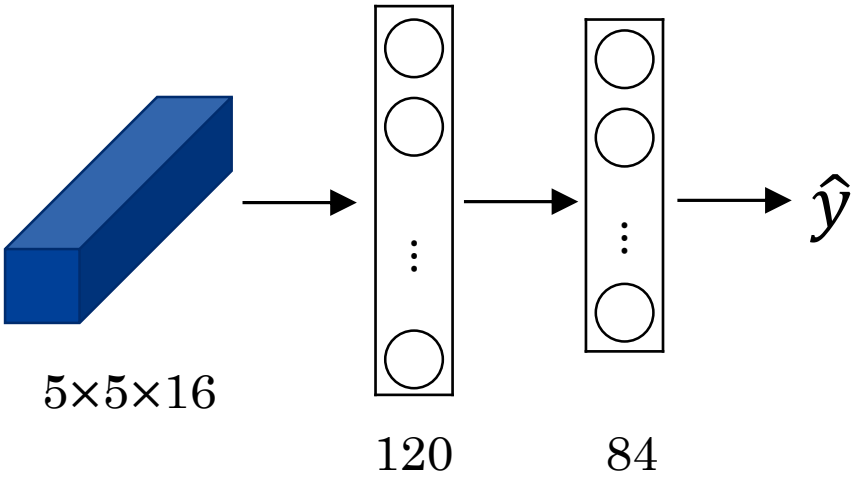
优点：可以实现上采样

有时我们需要将图像恢复到原来的尺寸以便进行进一步的计算(e.g.:图像的语义分割)，这个采用扩大图像尺寸，实现图像由小分辨率到大分辨率的映射的操作，叫做上采样(Upsample)。

只能恢复尺寸，不能恢复数值

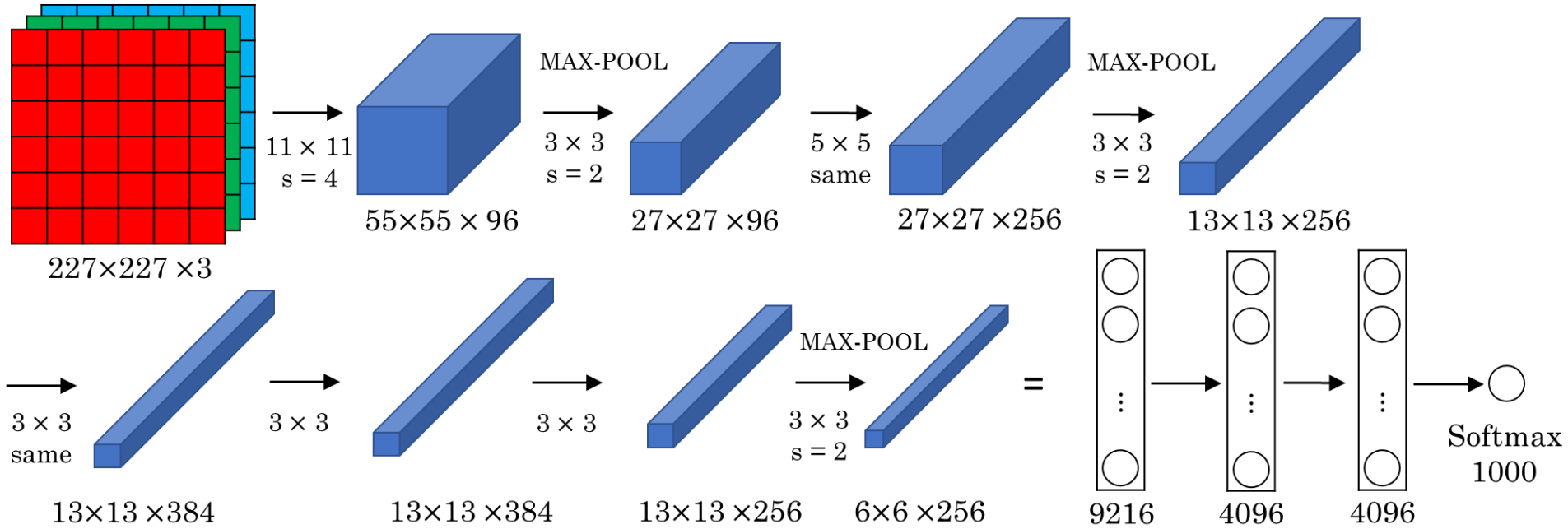
LeNet-5





60k parameters

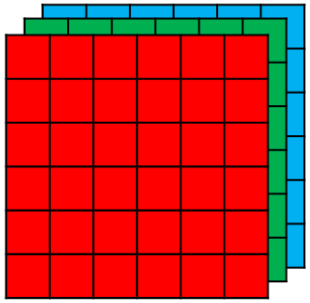
AlexNet



60M parameters

VGG-16

CONV = 3×3 filter, s = 1, same MAX-POOL = 2×2 , s = 2



224×224 ×3

→ [CONV 64] ×2 → 224×224×64 → POOL → 112×112 ×64 → [CONV 128] ×2 → 112×112 ×128 → POOL → 56×56 ×128

→ [CONV 256] ×3 → 56×56 ×256 → POOL → 28×28 ×256 → [CONV 512] × 3 → 28×28 ×512 → POOL → 14×14×512

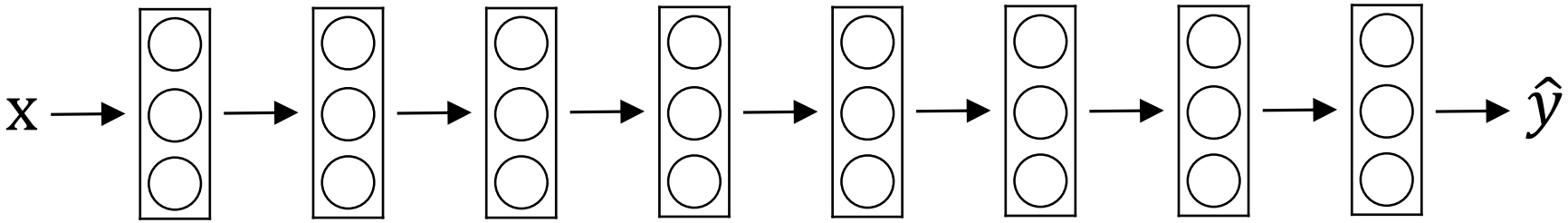
→ [CONV 512] ×3 → 14×14 ×512 → POOL → 7×7×512 → FC 4096 → FC 4096 → Softmax 1000

138M parameters

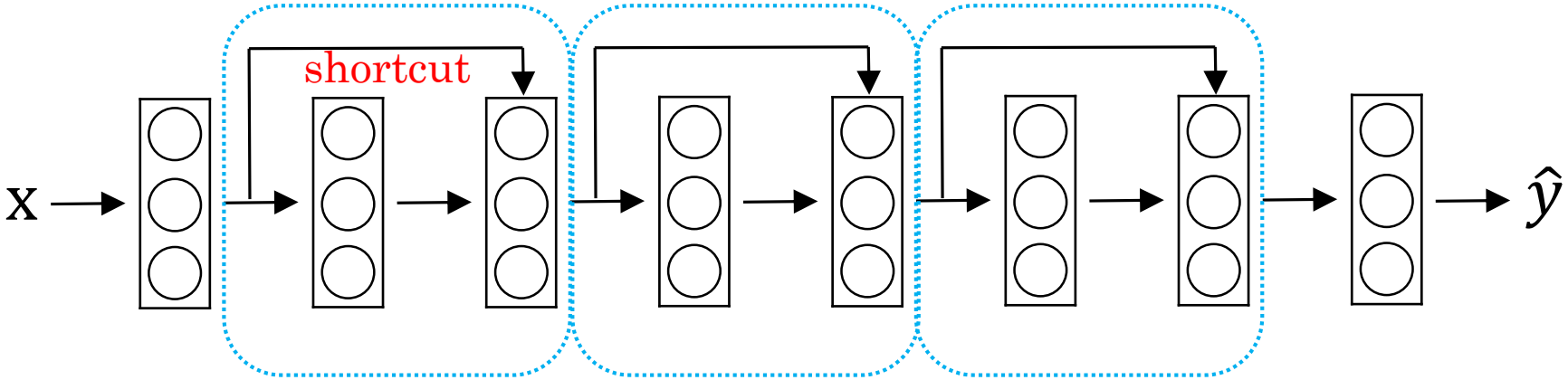
ResNet

2015, Kaiming He *et al.* (Microsoft)
ResNet wins in the ILSVRC 2015 detection, classification and localization tasks

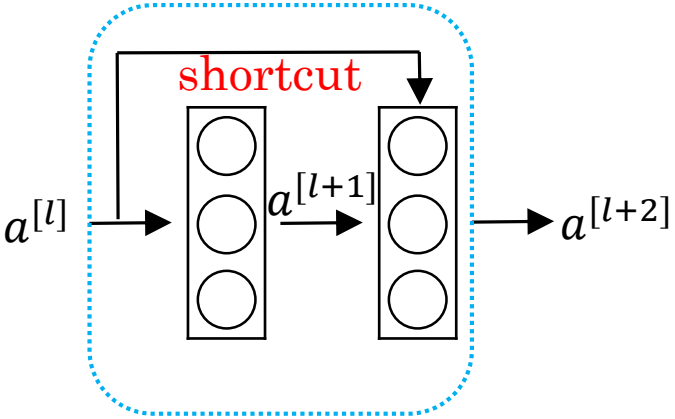
Plain: hard to train when go deeper



ResNet: introduce shortcut connections



Residual block



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

$$a^{[l+1]} = g(z^{[l+1]})$$

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

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

$z^{[l+2]}$ now is residual of $a^{[l]}$
if $z^{[l+2]} = 0$, then $a^{[l+2]} = a^{[l]}$

It is easier to learn the residual parameters.

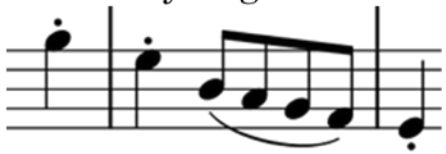
ResNet helps build very deep NN.

→ It's better $z^{[l+2]}$ and $a^{[l]}$ have same dimension.

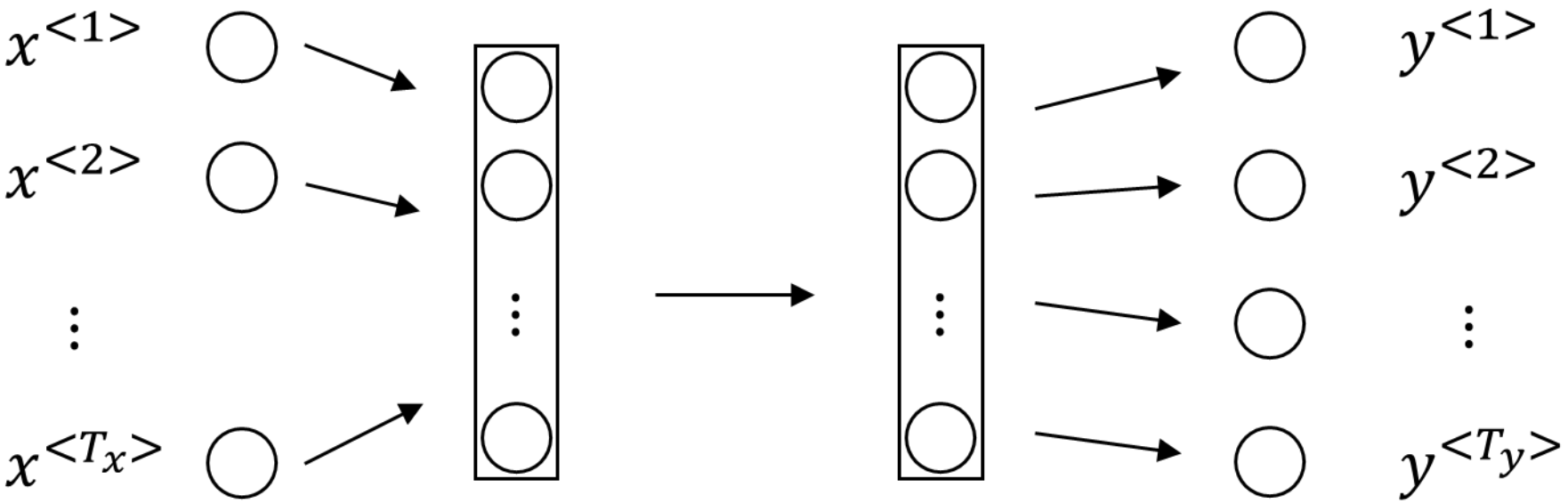
深度学习基础算法: Recurrent NN

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Examples of sequence data

Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation	Ø	→	
Sentiment classification	"There is nothing to like in this movie."	→	★☆☆☆☆
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

Problem with NN

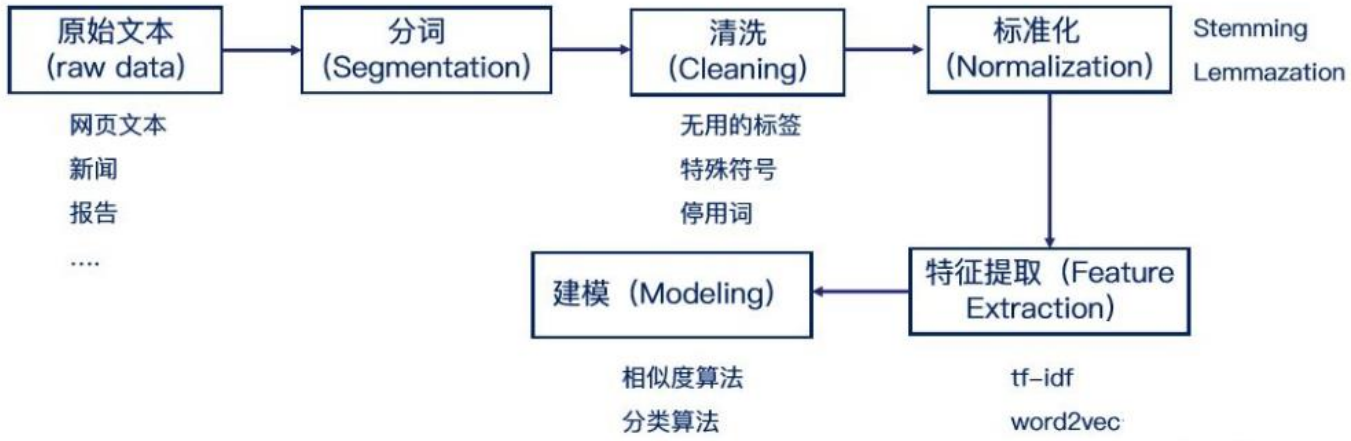


Problems:

- Inputs, outputs can be different lengths in different examples.
- Doesn't share features learned across different positions of text.

以自然语言处理为例

自然语言处理 (NLP) 是计算机程序理解人类口头和书面语言的能力。使用NLP的过程中通常有以下三个主要步骤：1. **文本预处理**（接收自然语言。自然语言就是大家平时在生活中常用的表达方式，如自然语言：这次做实验养的细胞比较散 (非自然语言：细胞分布的密度比较低)。）2. **文本表示**（转译自然语言）3. **分析和建模**（分析自然语言并输出结果）



Word representation

- One-hot vocabulary:

$x^{\langle 1\rangle}$ $x^{\langle 2\rangle}$ $x^{\langle 3\rangle}$ $x^{\langle 4\rangle}$ $x^{\langle 5\rangle}$

The cat is beautiful.

$$\begin{bmatrix} a \\ abundant \\ acclaim \\ \vdots \\ baby \\ \vdots \\ is \\ \vdots \\ \vdots \\ zoo \\ < \text{UNK} > \\ < \text{EOS} > \end{bmatrix} \quad \begin{bmatrix}0\\0\\0\\\vdots\\0\\0\\0\\1\\\vdots\\0\\0\\0\end{bmatrix} \quad \begin{bmatrix}0\\0\\0\\\vdots\\0\\1\\0\\0\\\vdots\\0\\0\\0\end{bmatrix} \quad \begin{bmatrix}0\\0\\0\\\vdots\\0\\0\\1\\0\\\vdots\\0\\0\\0\end{bmatrix} \quad \begin{bmatrix}0\\0\\0\\\vdots\\1\\0\\0\\0\\\vdots\\0\\0\\0\end{bmatrix} \quad \begin{bmatrix}0\\0\\0\\\vdots\\0\\0\\0\\0\\\vdots\\0\\0\\1\end{bmatrix}$$

Find noun:

$x^{\langle 1\rangle}$ $x^{\langle 2\rangle}$ $x^{\langle 3\rangle}$ $x^{\langle 4\rangle}$ $x^{\langle 5\rangle}$

The cat is beautiful.

↓ ↓ ↓ ↓ ↓

$y^{\langle 1\rangle}$ $y^{\langle 2\rangle}$ $y^{\langle 3\rangle}$ $y^{\langle 4\rangle}$ $y^{\langle 5\rangle}$

0 1 0 0 0

NLP工作流-文本表示

为了使用机器学习和深度学习方法分析文本，我们要把文字表示成计算机能够运算的数字或向量。存在于不同层面上，字母，单词，句子，文本...

例子：

John likes to watch movies. Mary likes too.
John also likes to watch football games.

法1：one-hot编码（独热编码）

```
John: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
likes: [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
...
```

又称一位有效编码，其方法是使用N位状态寄存器来对N个状态进行编码，每个状态都有它独立的寄存器位，并且在任意时候，其中只有一位有效。

优点：独热编码解决了分类器不好处理属性数据的问题，在一定程度上也起到了扩充特征的作用。它的值只有0和1，不同的类型存储在垂直的空间。

缺点1：当单词数量很多时，特征空间会变得非常大。

缺点2：难以体现文本特征：1.不考虑词与词之间的顺序；2.它假设词与词相互独立；3.特征是离散稀疏的（0个数远大于非零个数并且分布无规律，特征值都为整数）。

NLP工作流-文本表示

比如，上面的工作特征，属于离散型特征，共有五个可能取值，不使用独热编码（One-Hot Encoding）的话，其数字表示分别为：

> [演 员，厨 师，公务员，工程师，律 师] = [0，1，2，3，4]

那么不同工作特征之间的距离为：

> d（演 员，厨 师）= 1
> d（厨 师，公务员）= 1
> d（公务员，工程师）= 1
> d（工程师，律 师）= 1
> d（演 员，公务员）= 2
> d（演 员，工程师）= 3
> 以此类推...

NLP工作流-文本表示

例子：

John likes to watch movies. Mary likes too.
John also likes to watch football games.

法2：Bag of Words（词袋表示）

```
John likes to watch movies. Mary likes too.  -->> [1, 2, 1, 1, 1, 0, 0, 0, 1, 1]
John also likes to watch football games.     -->> [1, 1, 1, 1, 0, 1, 1, 1, 0, 0]
```

词袋表示，也称为计数向量表示(Count Vectors)。**句子或文本**的向量表示可以直接用单词的向量进行求和得到。

法3：Bi-gram(N-gram)

Bi-gram将相邻两个单词编上索引，N-gram将相邻N个单词编上索引。

{"John likes": 1, "likes to": 2, "to watch": 3, "watch movies": 4, "Mary likes": 5, "likes too": 6, "John also": 7, "also likes": 8,"watch football": 9, "football games": 10}

为 Bi-gram建立索引

```
John likes to watch movies. Mary likes too.  -->> [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
John also likes to watch football games.     -->> [0, 1, 1, 0, 0, 0, 1, 1, 1, 1]
```

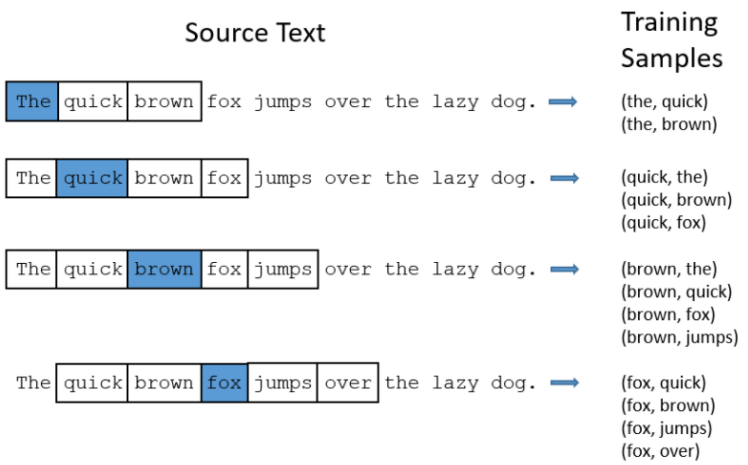
优点：与词袋表示法相比，考虑了词的顺序
缺点1：造成了词向量的急剧膨胀，导致特征空间会变得非常大
缺点2：特征值离散

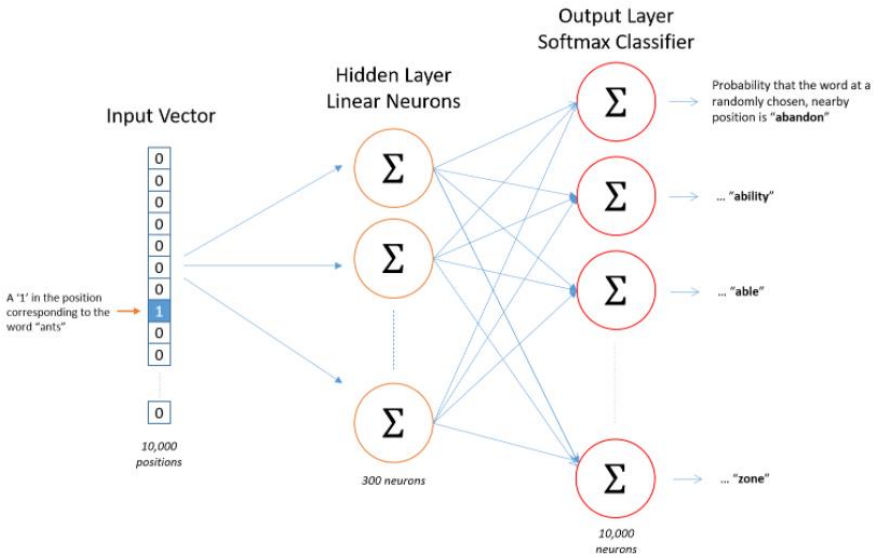
NLP工作流-文本表示

法5：Word2Vec

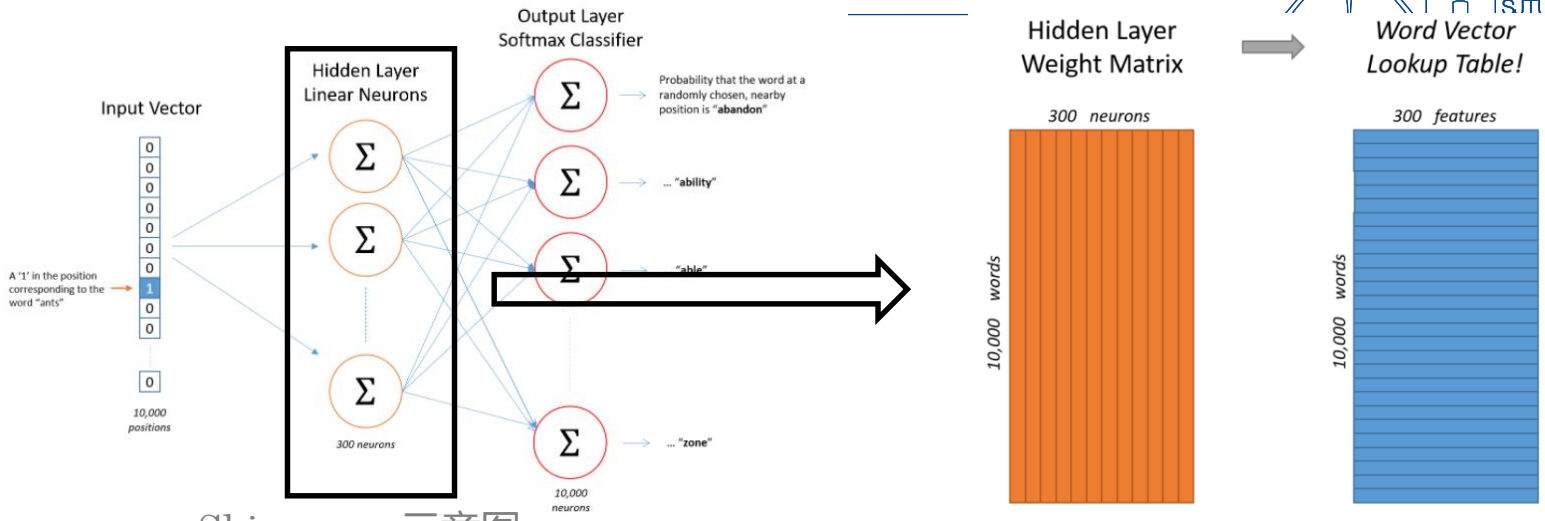
Word2Vec 是一种构建Word Embedding的方法。它可以使用两种方法：Skip Gram 和 Common Bag Of Words (CBOW)。Skip Gram**以每个单词输入，尝试预测上下文**。CBOW**以每个单词的上下文作为输入，尝试预测上下文对应的单词**。Word Embedding在这个过程中产生。

Skip Gram（以每个单词输入，尝试预测上下文）





NLP工作流-文本表示



Skip gram 示意图

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

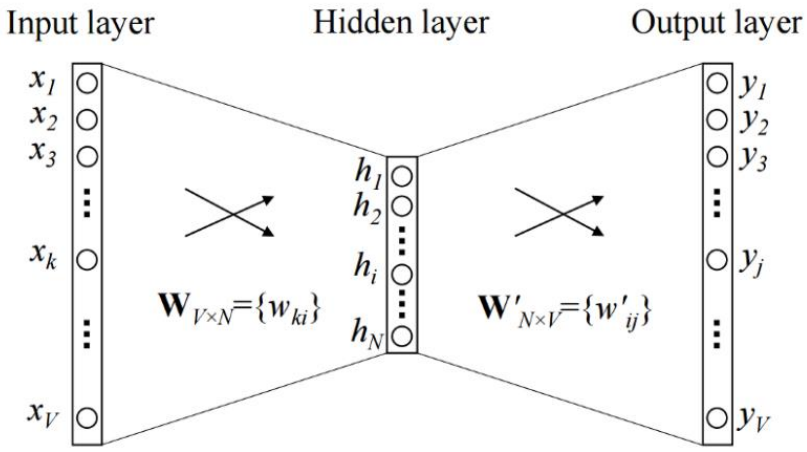
实现了高维特征向量（左）到低维特征（右）向量的转化

NLP工作流-文本表示

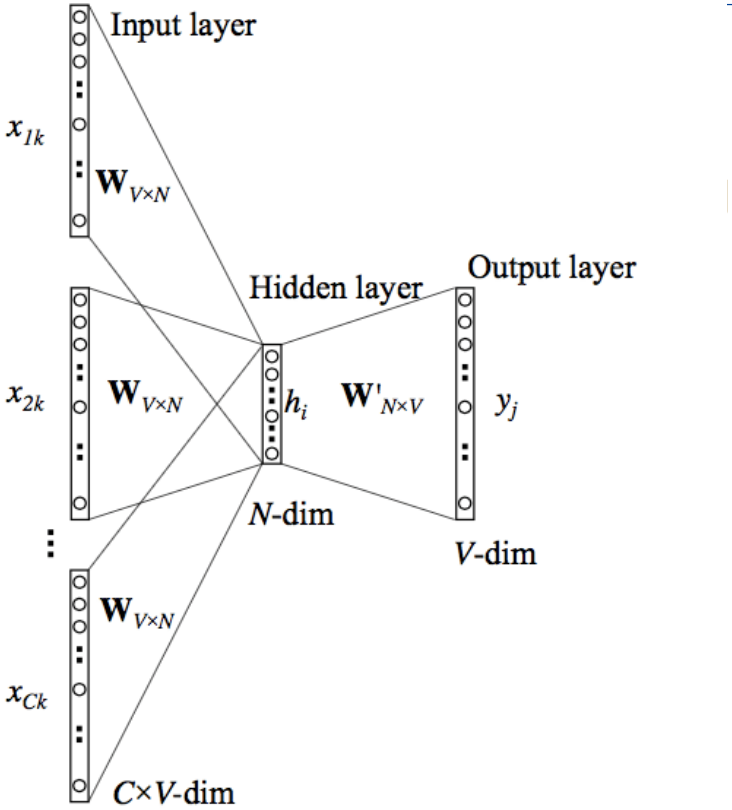
法5：Word2Vec

CBOW（以每个单词的上下文作为输入，尝试预测上

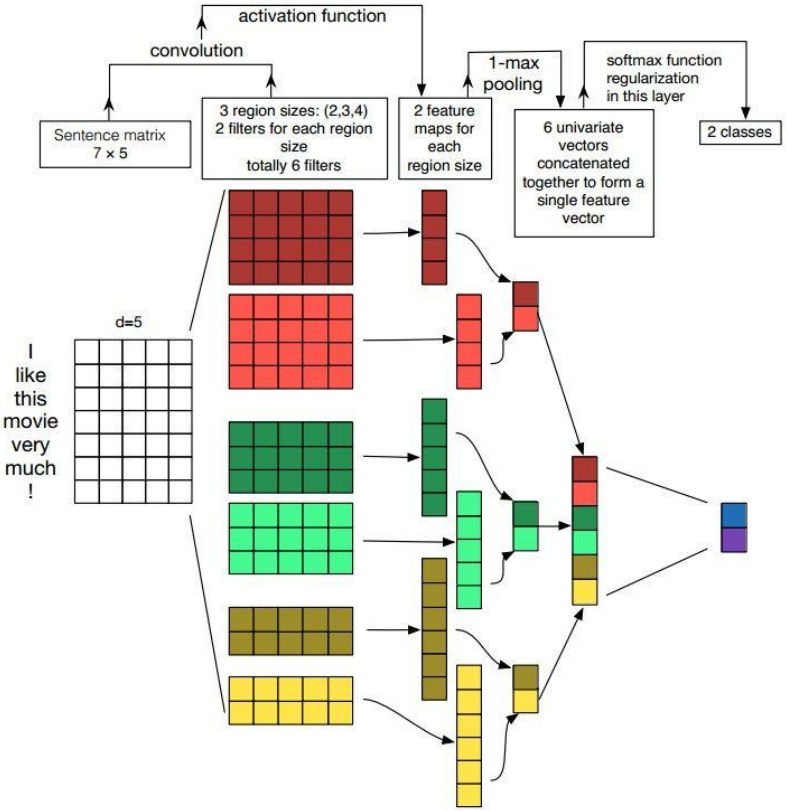
当我们只用上下文中的一个词作为输入时，如，
络示意图如下：



V是文本中单词数量，N是隐藏层中神经元个数



NLP传统深度模型-TextCNN



textCNN是一个应用了CNN网络的文本分类模型。

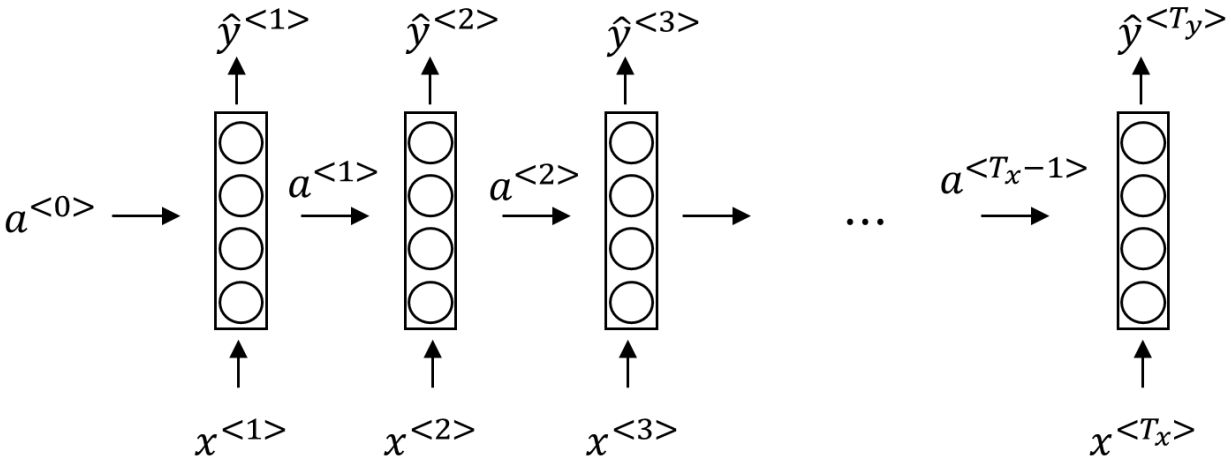
输入层：输入层是一个n×k的矩阵，n为一个句子中的单词数，k为每个词对应的向量。每个词向量可以是预先在其他语料库中训练好的，也可以作为未知的参数由网络训练得到。预先训练的词嵌入可以利用其他语料库得到更多的先验知识，而由当前网络训练的词向量能够更好地抓住与当前任务相关联的特征。

卷积层：卷积核和词向量的宽度一致，但高度不同，可以提取**不同范围内词的关系，获得的是纵向的差异信息。**

池化层：如图中所示的网络采用了1-Max池化

优点：模型简单, 训练速度快，效果不错。
缺点：模型可解释型不强。

Recurrent NN

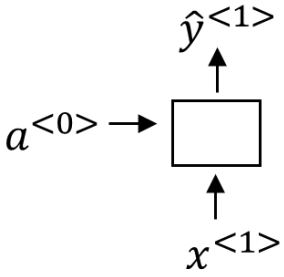


$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

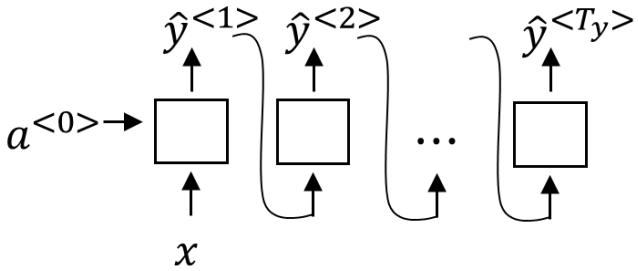
$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

$$\mathcal{L} = \sum_t \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

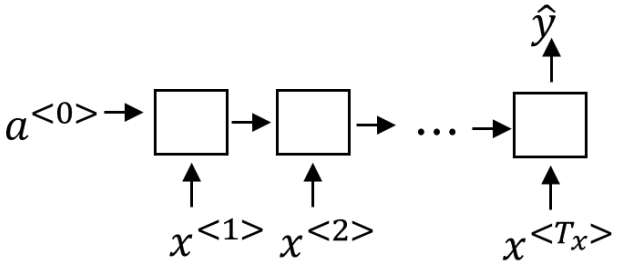
Types of RNN



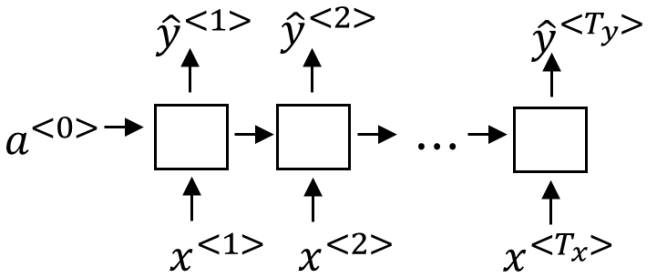
One to one



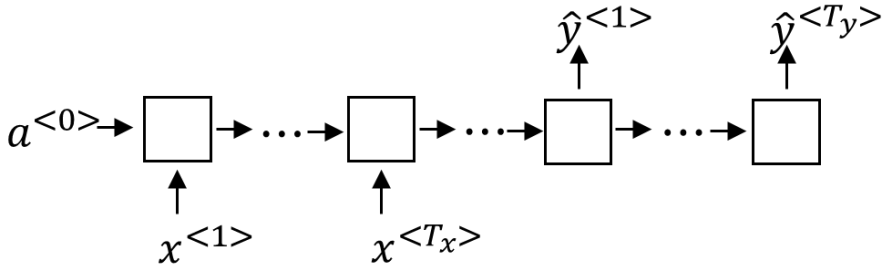
One to many



Many to one



Many to many



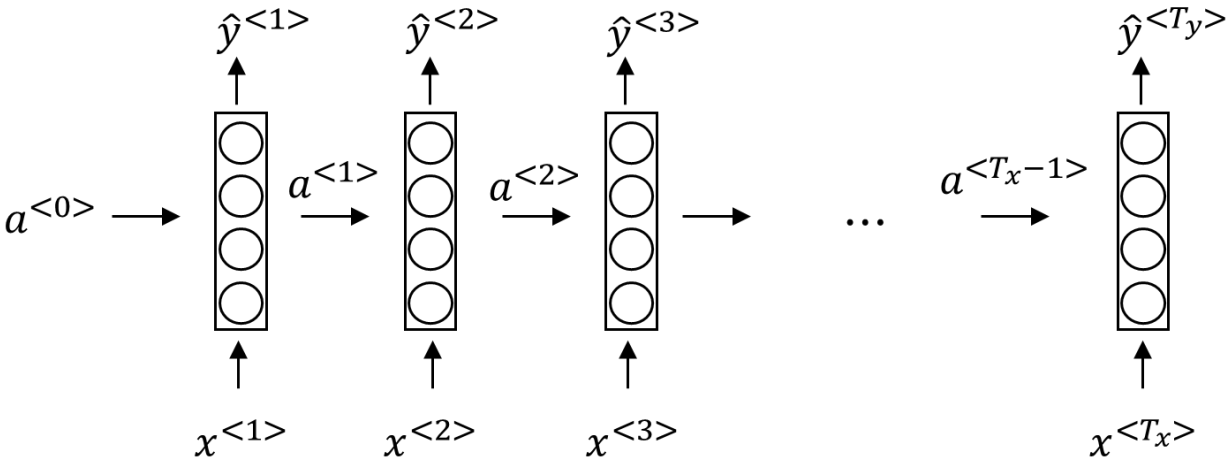
Many to many

思考：各举几个应用的例子

Vanishing Gradients

The cat, which …, was full.

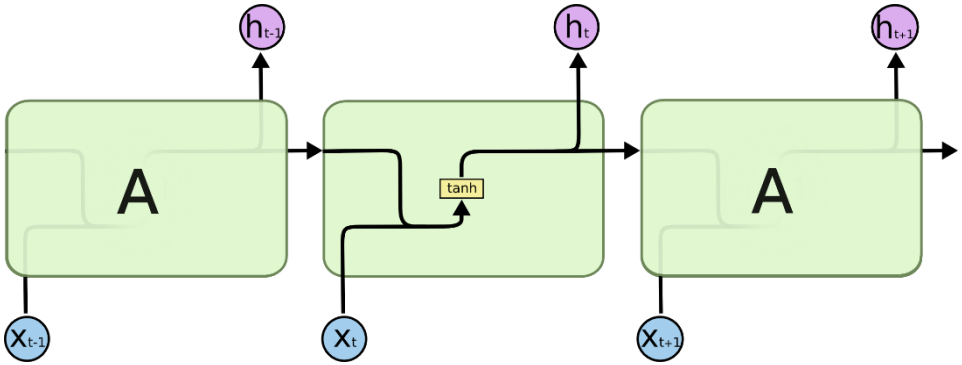
The cats, which …, were full.

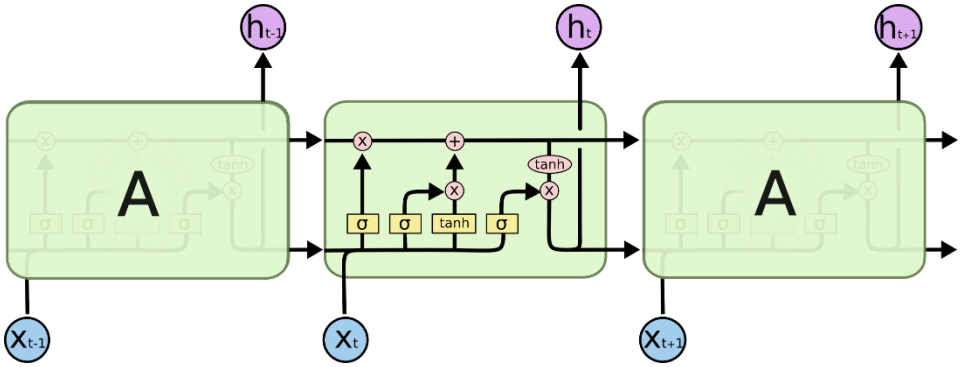


1. vanishing gradients

2. how to memorize information long time ago

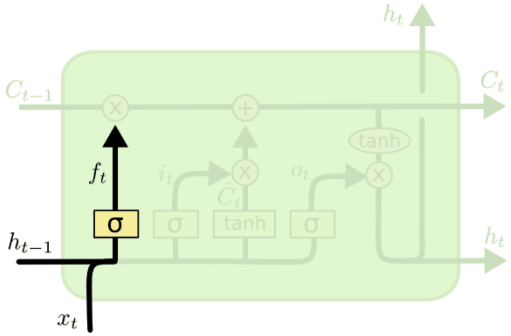
Long Short Term Memory





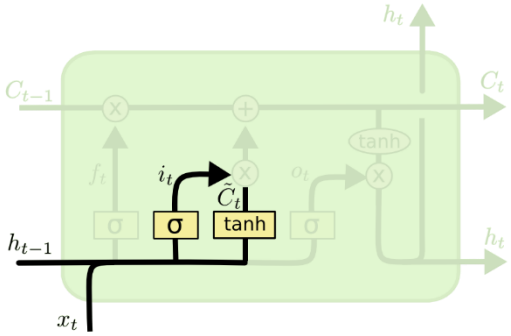
cell state

LSTM



forget gate:
decide what information in cell state will be forgotten

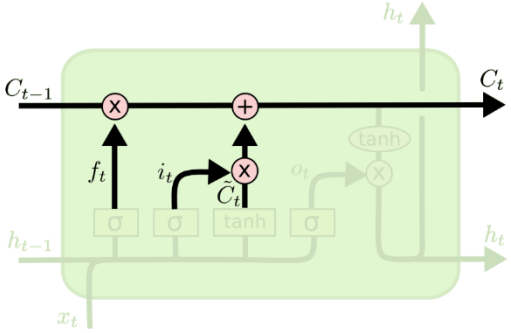
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$



input gate:
decide what information will be added into cell state

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

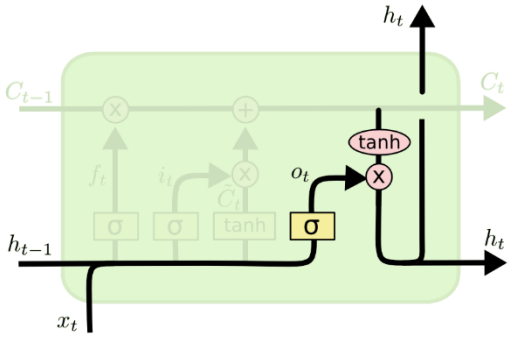
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



update:
give out new cell state

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM

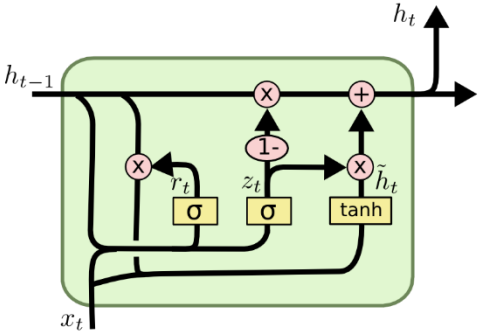


output gate:
decide what information will be output

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Gated Recurrent Unit (GRU): output gate and update gate



$$z_t = \sigma\left(W_z \cdot \left[h_{t-1}, x_t\right]\right)$$

$$r_t = \sigma\left(W_r \cdot \left[h_{t-1}, x_t\right]\right)$$

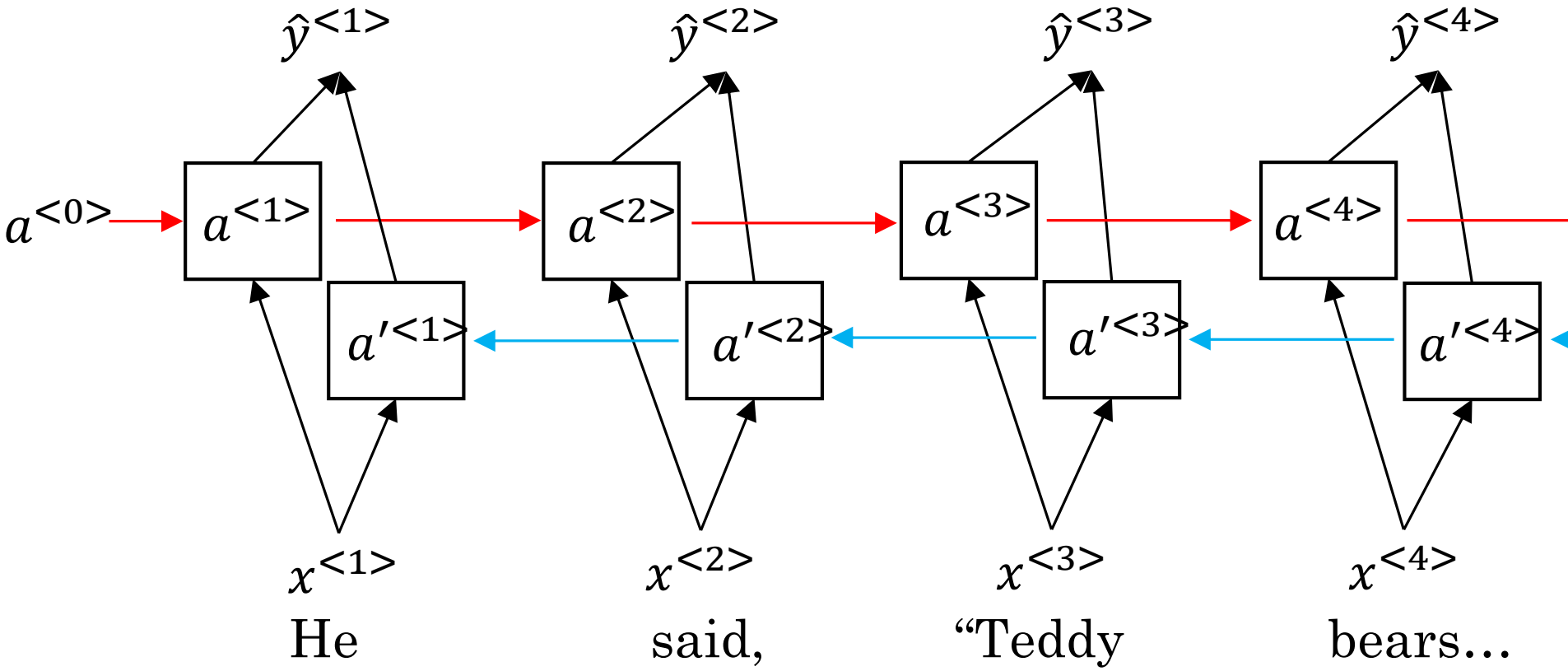
$$\tilde{h}_t = \tanh\left(W \cdot \left[r_t * h_{t-1}, x_t\right]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

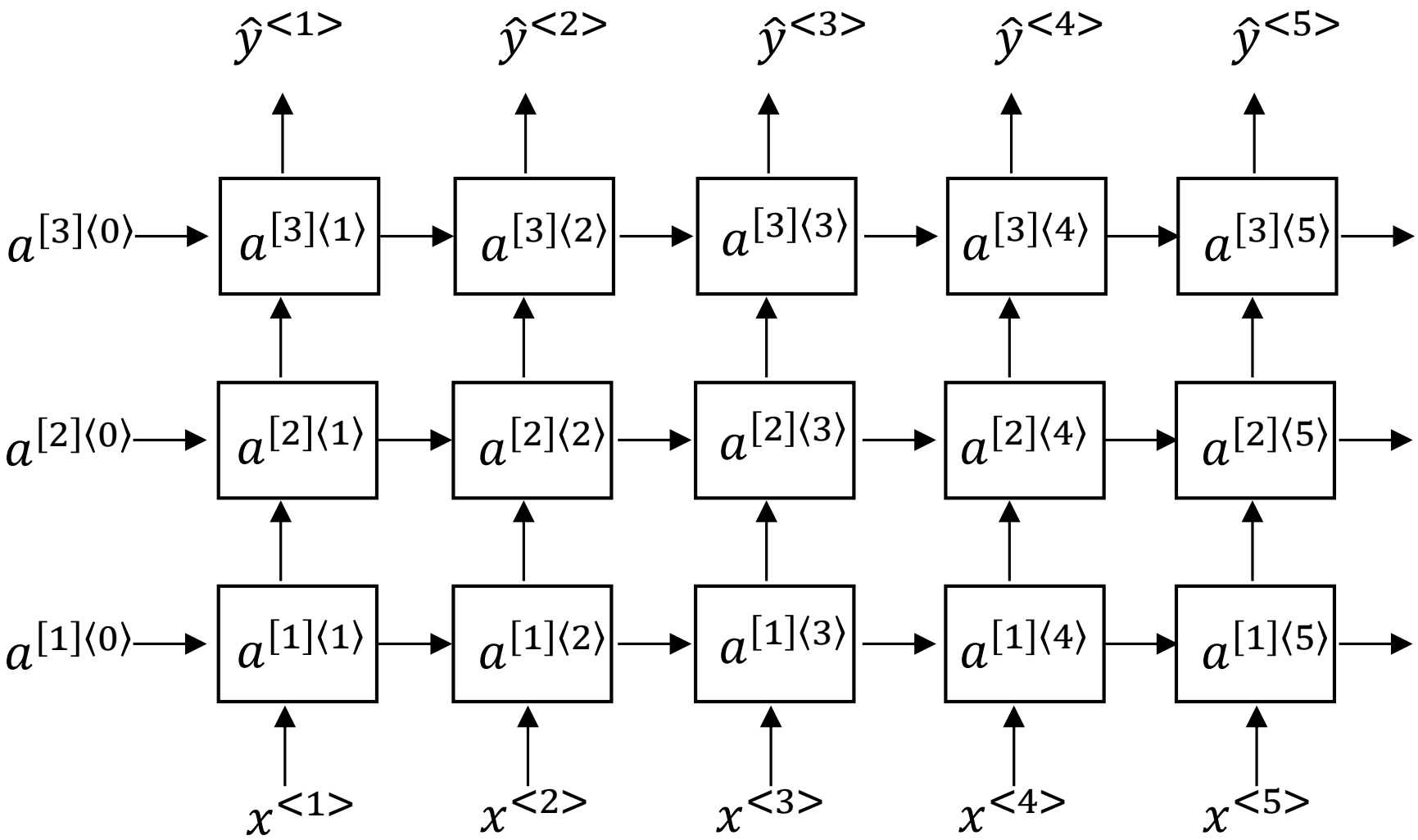
Bidirectional RNN

He said, “Teddy Roosevelt was a great President.”
He said, “Teddy bears are on sale!”

$$\hat{y}^{<t>} = g(W[a^{<t>}, a'^{<t>}] + b)$$



Deep RNN


$\hat{y}^{<1>}$
$\hat{y}^{<2>}$
$\hat{y}^{<3>}$
$\hat{y}^{<4>}$
$\hat{y}^{<5>}$
$a^{[3]\langle 0\rangle}$
$a^{[3]\langle 1\rangle}$
$a^{[3]\langle 2\rangle}$
$a^{[3]\langle 3\rangle}$
$a^{[3]\langle 4\rangle}$
$a^{[3]\langle 5\rangle}$
$a^{[2]\langle 0\rangle}$
$a^{[2]\langle 1\rangle}$
$a^{[2]\langle 2\rangle}$
$a^{[2]\langle 3\rangle}$
$a^{[2]\langle 4\rangle}$
$a^{[2]\langle 5\rangle}$
$a^{[1]\langle 0\rangle}$
$a^{[1]\langle 1\rangle}$
$a^{[1]\langle 2\rangle}$
$a^{[1]\langle 3\rangle}$
$a^{[1]\langle 4\rangle}$
$a^{[1]\langle 5\rangle}$
$x^{<1>}$
$x^{<2>}$
$x^{<3>}$
$x^{<4>}$
$x^{<5>}$

深度学习: Attention

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Word representation

- One-hot vocabulary:

$x^{\langle 1 \rangle}$ $x^{\langle 2 \rangle}$ $x^{\langle 3 \rangle}$ $x^{\langle 4 \rangle}$ $x^{\langle 5 \rangle}$

The cat is beautiful.

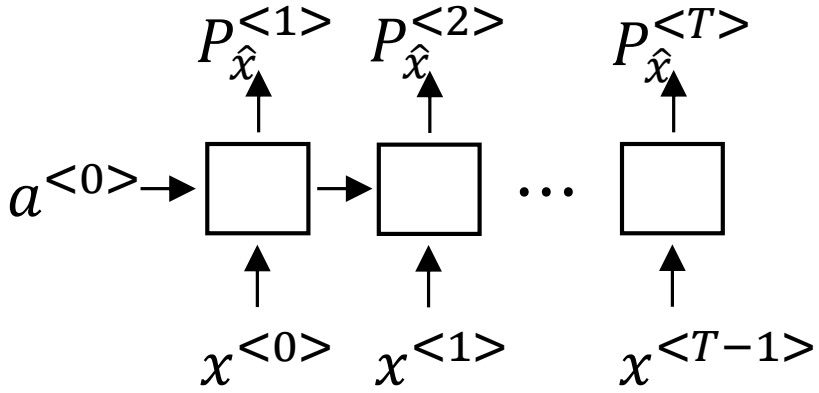
$$\begin{bmatrix} a \\ abundant \\ acclaim \\ \vdots \\ baby \\ \vdots \\ is \\ \vdots \\ \vdots \\ zoo \\ < \text{UNK} > \\ < \text{EOS} > \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

- Feature representation:

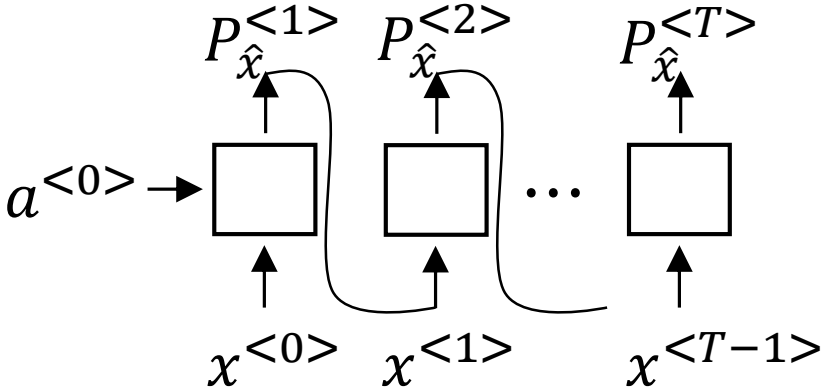
	Man	Woman	King	Queen	Apple	Orange
Gender	−1	1	−0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	−0.01	0.00
Age	0.03	0.02	0.7	0.69	0.03	−0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Size	⋮	⋮	⋮	⋮	⋮	⋮

语言训练模型

- Establish an RNN model of language:



- Generative sampling language sequences:

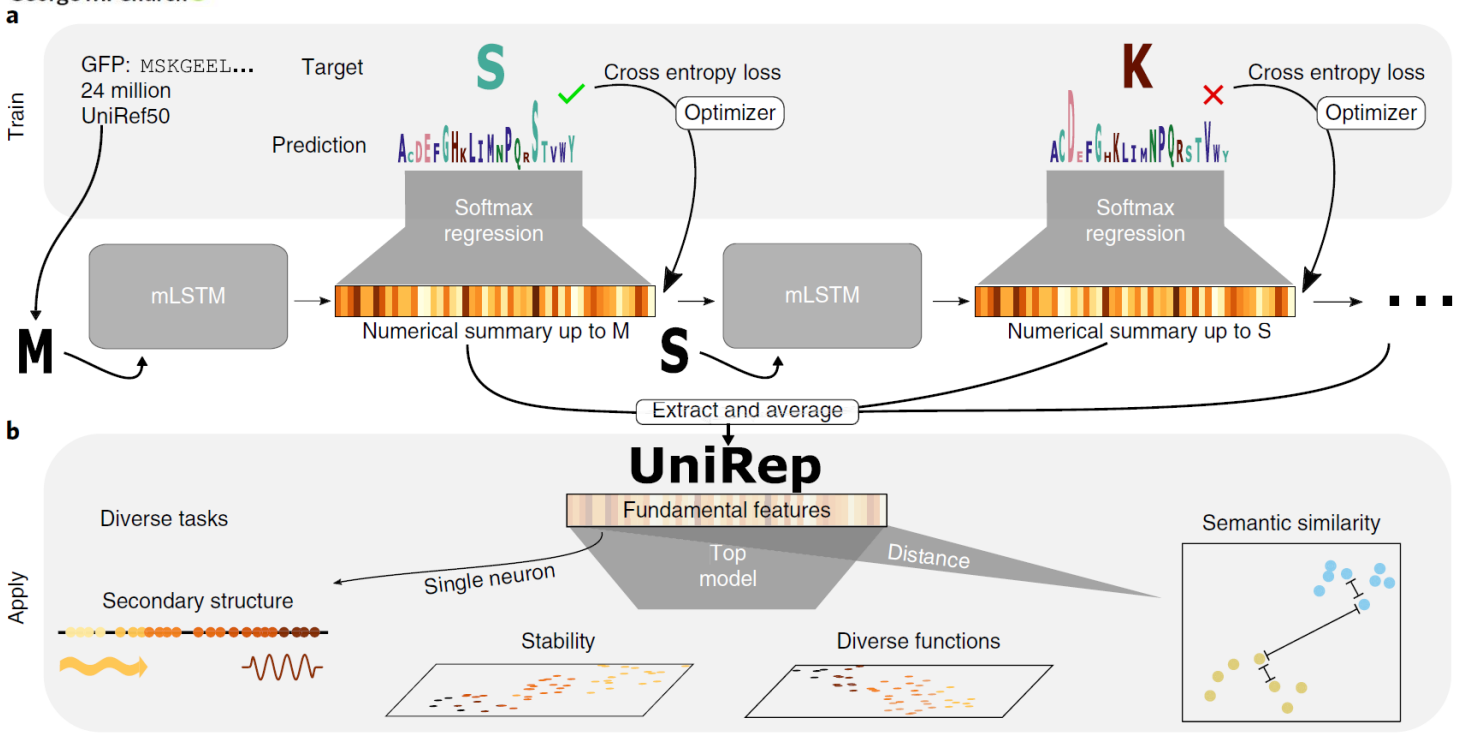


Biological Sequence Modelling

nature methods
ARTICLES
https://doi.org/10.1038/s41592-019-0598-1

Unified rational protein engineering with sequence-based deep representation learning

Ethan C. Alley[1,2,6], Grigory Khimulya[6,7], Surojit Biswas[1,3,6], Mohammed AlQuraishi[4] and George M. Church[1,5*]



Neural Machine Translation

- Seq2seq model was used for neural machine translation:

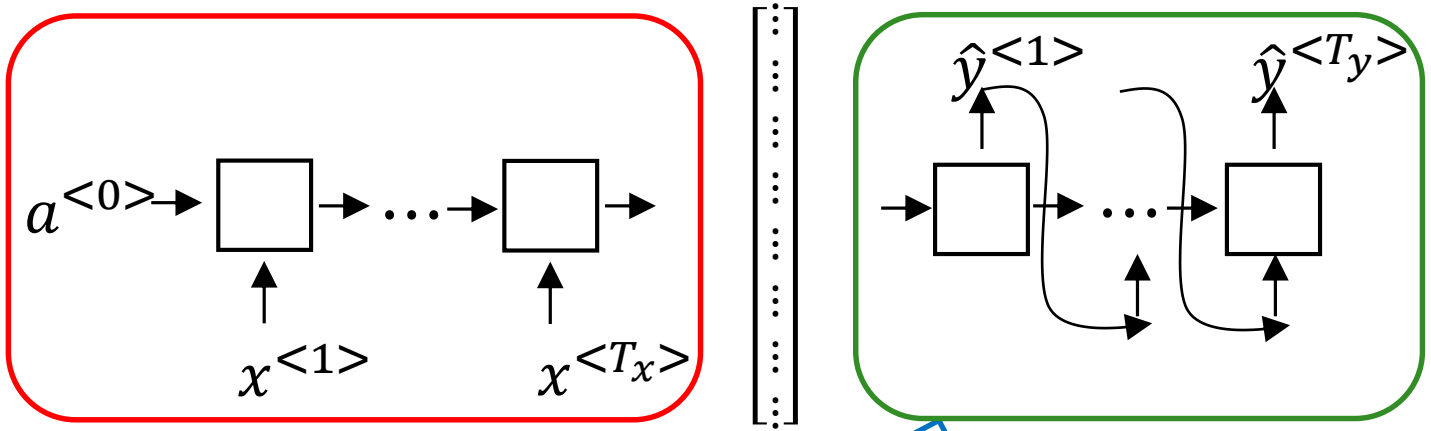


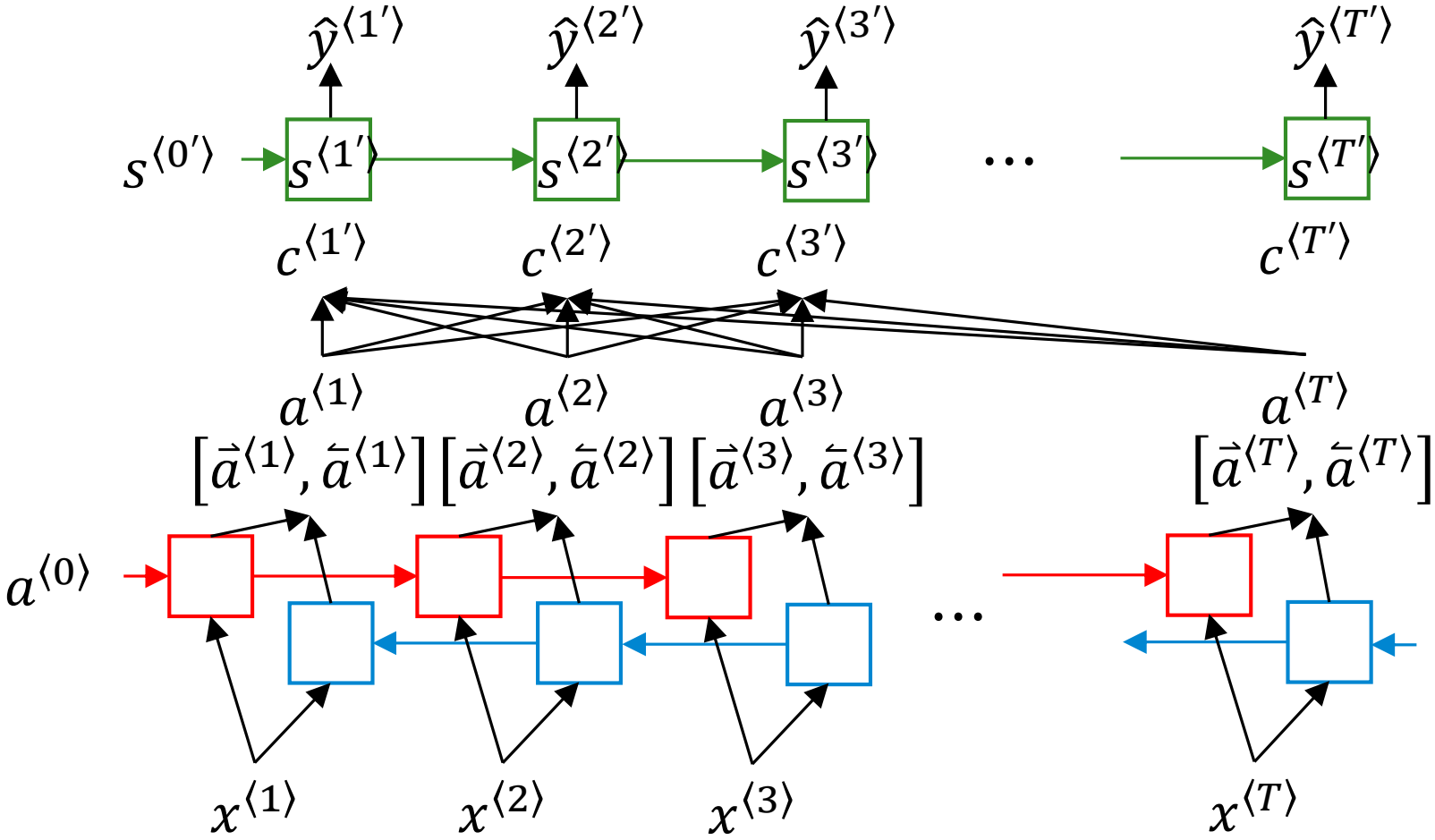
image caption

Probability of whole sentence is evaluated.

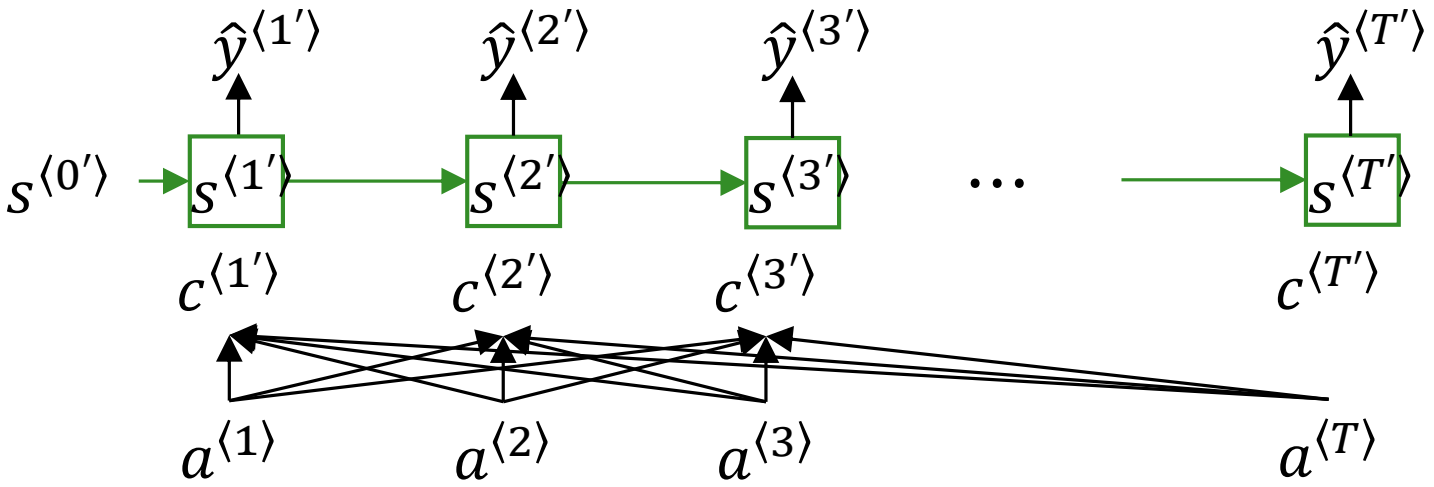
Beam search algorithm can be used to search through all possible sentences.

Attention Mechanism

- Attention was invented in 2015, to help memorize long sentences



Additive Attention



$$c^{\langle t' \rangle} = \alpha_{t',1} a^{\langle 1 \rangle} + \alpha_{t',2} a^{\langle 2 \rangle} + \alpha_{t',3} a^{\langle 3 \rangle} + \cdots = \alpha_{t'} \boldsymbol{a}^{\langle t \rangle}$$

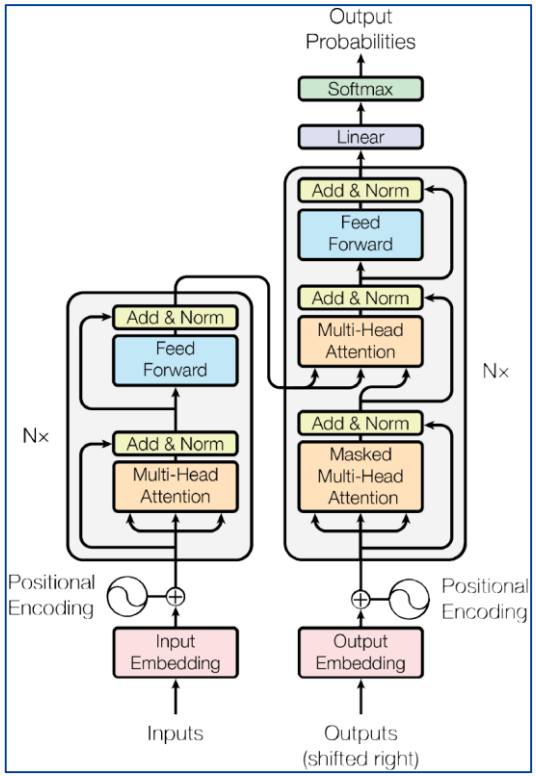
$$\boldsymbol{c}^{\langle t' \rangle} = \boldsymbol{\alpha} \boldsymbol{a}^{\langle t \rangle}$$

$$\alpha_{t'-1,t} = softmax\left(\boldsymbol{V}_\alpha tanh\left(\boldsymbol{W}_\alpha \left[\boldsymbol{s}^{\langle t'-1 \rangle}, \boldsymbol{a}^{\langle t \rangle}\right]\right)\right)$$

$$\begin{cases} \boldsymbol{s}^{\langle t'-1 \rangle} \odot \boldsymbol{a}^{\langle t \rangle} \\ \left(\boldsymbol{s}^{\langle t'-1 \rangle}\right)^T W \, \boldsymbol{a}^{\langle t \rangle} \end{cases}$$

Transformer

- 2017 proposed by Google
- BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-training Transformer)



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

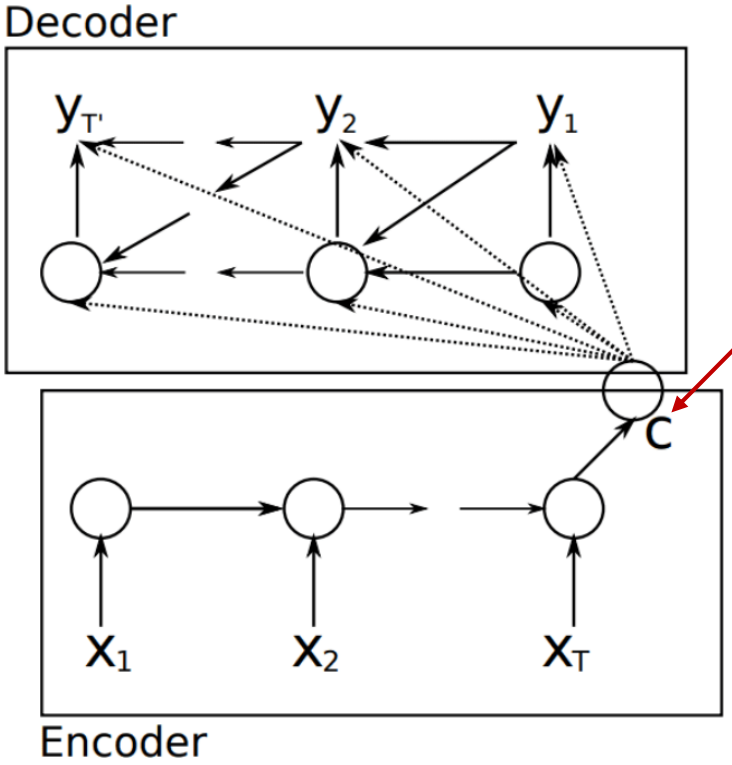
Attention is not *all* you need:
pure attention loses rank doubly exponentially with depth

Yihe Dong Google yihed@google.com
Jean-Baptiste Cordonnier EPFL jean-baptiste.cordonnier@epfl.ch
Andreas Loukas EPFL andreas.loukas@epfl.ch

Transformer

- 三个主要创新点
  - Positional Encoding
  - Attention
  - Self Attention

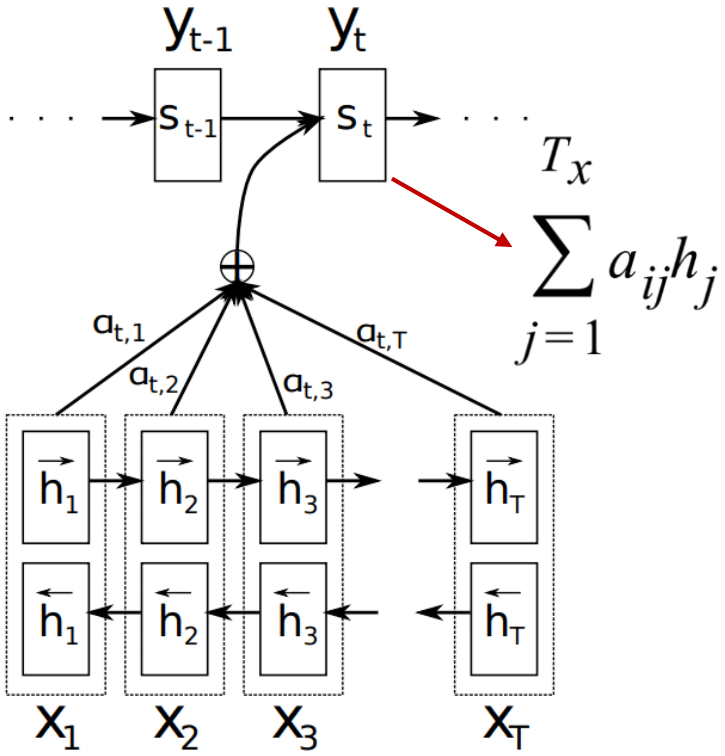
TextRNN Encoder-Decoder架构：平行计算



- Encoder-Decoder神经网络架构使用循环神经网络学习，将变长源序列X编码成定长向量表示c，并将学习到的定长向量c解码成变长目标序列Y。模型的编码器和解码器被联合训练，以最大化给定源序列的目标序列的条件概率。

https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks/

Encoder-Decoder架构改进



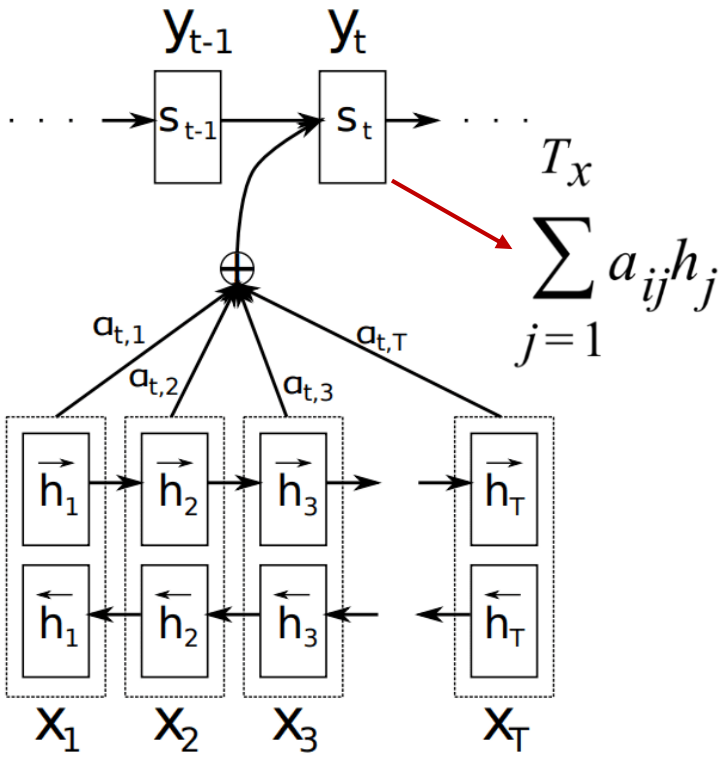
- Decoder输出的每一个y_i都同时考虑了所有的输入单元

$$a_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)}$$

$$e_{ij} = v_a^{\top} \tanh\left(W_a s_{i-1} + U_a h_j\right)$$

https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks/

注意力机制 Attention



https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks/

1. 柔性注意力机制（Soft Attention）

- 在输入信息上计算注意力分布
- 根据注意力分布计算输入信息的加权求和

给定一个和任务相关的查询向量q，用注意力变量z∈[1, N]表示被选择信息的索引位置，即z=i表示选择了第i个输入信息。

$$\alpha_i = p(z = i \mid X, \mathbf{q})$$
$$= softmax\left(s\left(\mathbf{x}_i, \mathbf{q}\right)\right)$$
$$= \frac{\exp\left(s\left(\mathbf{x}_i, \mathbf{q}\right)\right)}{\sum_{j=1}^{N}\exp\left(s\left(\mathbf{x}_j, \mathbf{q}\right)\right)}$$

其中，α_i称为注意力分布，$s\left(\mathbf{x}_i, \mathbf{q}\right)$称为注意力打分函数

注意力机制 Attention

2. 键值对注意力机制（Key-Value Pair Attention Mechanism）

输入信息：$(K, V) = \left[\left(\mathbf{k}_1, \mathbf{v}_1\right), \cdots, \left(\mathbf{k}_N, \mathbf{v}_N\right)\right]$

其中键用来计算注意力分布αi，值用来计算聚合信息

注意力分布：

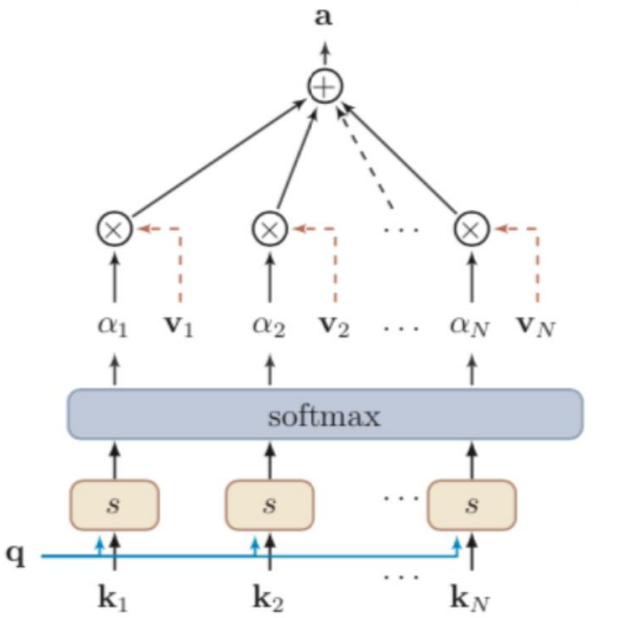
$$\alpha_i = \frac{\exp\left(s\left(\mathbf{k}_i, \mathbf{q}\right)\right)}{\sum_{j=1}^{N}\exp\left(s\left(\mathbf{k}_j, \mathbf{q}\right)\right)}$$

当$K = V$时，键值对注意力机制等价为柔性注意力机制

- **加权求和：**

$$att((K, V), \mathbf{q}) = \sum_{i=1}^{N}\alpha_i\mathbf{v}_i$$

$$= \sum_{i=1}^{N}\frac{\exp\left(s\left(\mathbf{k}_i, \mathbf{q}\right)\right)}{\sum_{j=1}^{N}\exp\left(s\left(\mathbf{k}_j, \mathbf{q}\right)\right)}\mathbf{v}_i$$

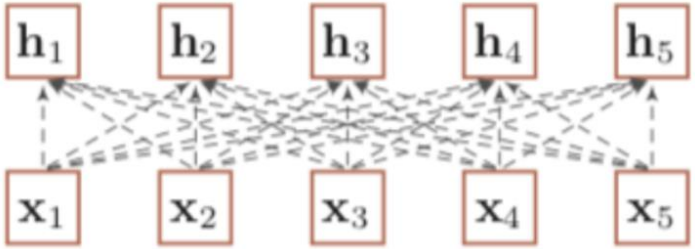


注意力机制 Attention

3. 自注意力模型（Self-Attention Model）

输入序列：$X = \left[\mathbf{x}_1, \cdots, \mathbf{x}_N\right] \in \mathrm{R}^{d_1 \times N}$

输出序列：$H = \left[\mathbf{h}_1, \cdots, \mathbf{h}_N\right] \in \mathrm{R}^{d_2 \times N}$



$$Q = W_Q X \in \mathrm{R}^{d_3 \times N}$$

通过线性变换得到向量序列：$K = W_K X \in \mathrm{R}^{d_3 \times N}$

$$V = W_V X \in \mathrm{R}^{d_2 \times N}$$

其中，Q为查询向量序列，K为键向量序列，V为值向量序列，WQ, WK, WV分别为可学习参数矩阵

若使用缩放点积模型作为打分函数，则输出向量序列：

$$H_{d_2 \times N} = softmax\left(\frac{K^{\top} Q}{\sqrt{d_3}}\right) V_{d_2 \times N}$$

$$= softmax\left(\frac{K^{\top} Q}{\sqrt{d_3}}\right) W_V X$$

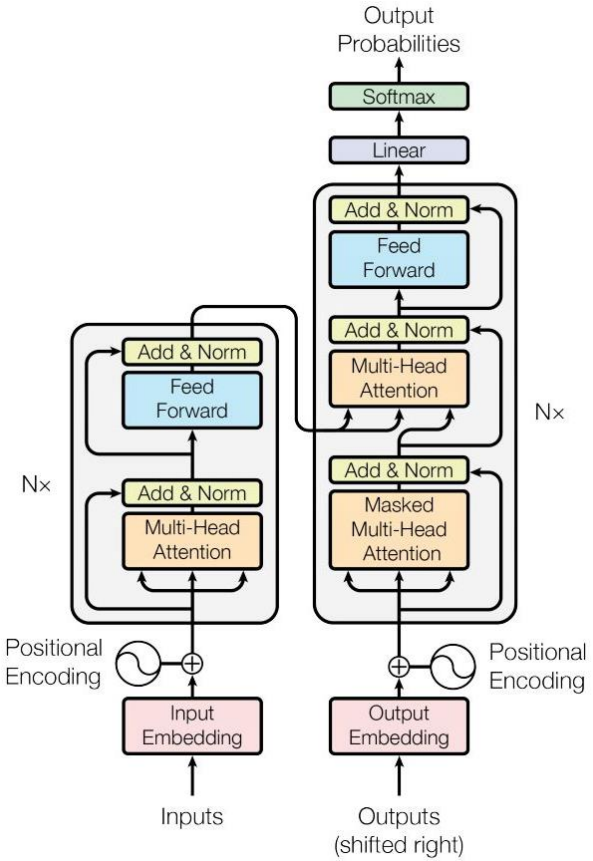
注意力机制 Attention

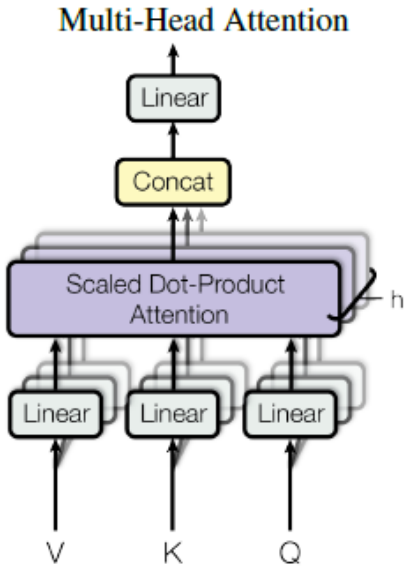
4. 多头注意力机制（Multi-Head Attention Mechanism）

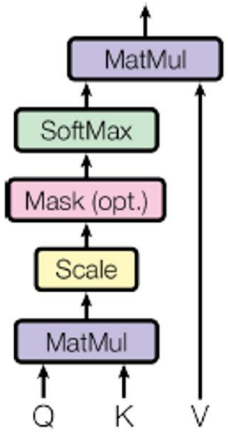
多个查询Q = [q1, ···, qM]平行的计算从输入信息中选取多个信息。每个注意力关注输入信息的不同部分，最后向量拼接。

$$att((K, V), Q) = att\left((K, V), \mathbf{q}_1\right) \oplus \cdots \oplus att\left((K, V), \mathbf{q}_M\right)$$

Transformer模型



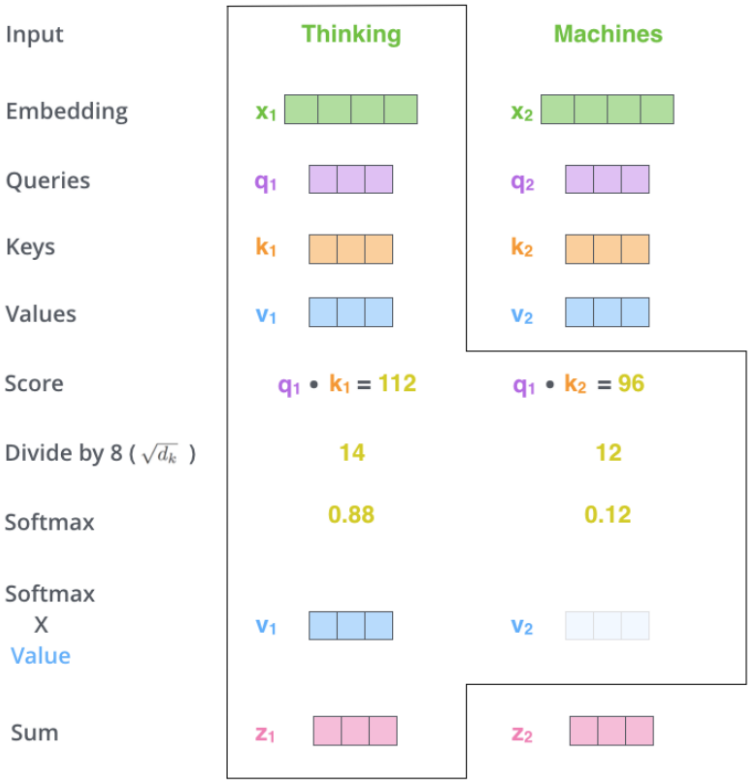


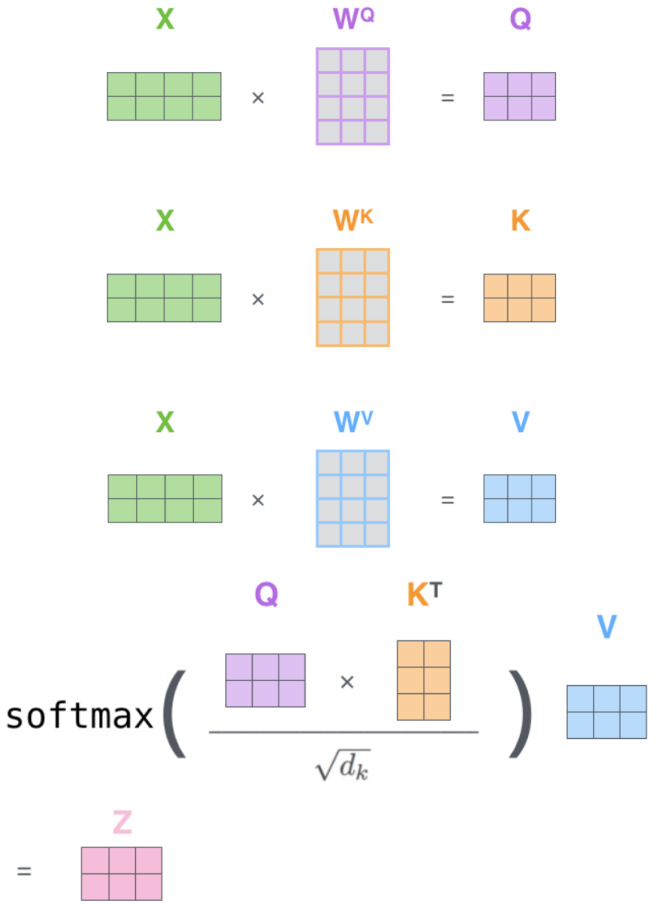


Vaswan et al., Attention Is All You Need
https://arxiv.org/pdf/1706.03762.pdf

在encode 和 decode, 都加入位置信息，可以训练中自动学到次序。

Transformer模型（Self Attention）





https://jalammar.github.io/illustrated-transformer/

人工智能算法进阶

（一）迁移学习

迁移学习的动机：

复杂模型需大量标注数据

大多数模型面对新任务时表现下降

迁移学习可分为
实例迁移（Instance Transfer）
特征迁移（Feature Transfer）
参数迁移（Parameter Transfer）
域泛化（Domain Generalization）

迁移学习的两大核心术语：
域（domain）：
特征空间X和空间上的数据分布$p(x)$

任务（task）：
标签空间Y及目标预测函数$f(\cdot)$

迁移学习之所以能够迁移，是因为源域和目标域不同，这里的不同主要分为域的不同和任务的不同，其中域的不同表现在空间和分布两个层面。

人工智能算法进阶

（一）迁移学习

1. 实例迁移

重新调整源域中实例的权重然后应用于目标域的数据

代表模型是选择性学习算法（Selective Learning Algorithm），
主要思想是构建自编码器并调整误差函数以进行实例选择

人工智能算法进阶

（一）迁移学习

2. 特征迁移

源域和目标域的数据映射到一个分布更加接近的特征空间当中然后提取特征。

代表模型：
迁移成分分析（Transfer Component Analysis, TCA）

将不同分布的数据映射到一个新的隐空间当中，使得数据分布更加接近。

$$Dist\left(\varphi(X_S),\varphi(X_T)\right) = \max_{\phi}\left\|\frac{1}{n_S}\sum_{i=1}^{n_S}\phi\left(\varphi(X_S)\right)-\frac{1}{n_T}\sum_{i=1}^{n_T}\phi\left(\varphi(X_T)\right)\right\|_H$$

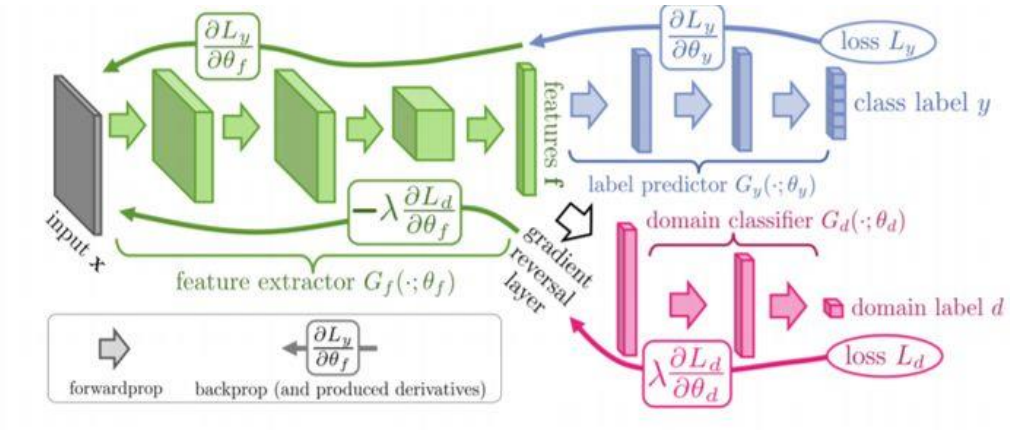
人工智能算法进阶

（一）迁移学习

2. 特征迁移

代表模型：

域对抗神经网络（Domain Adversarial Neural Network, DANN）

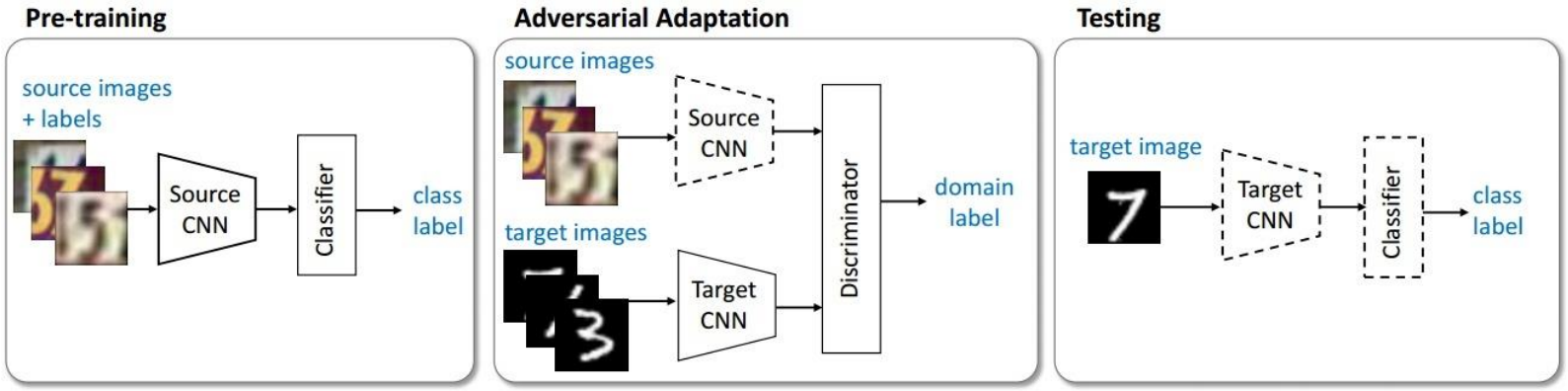


人工智能算法进阶

（一）迁移学习

2. 特征迁移

代表模型：Adversarial Discriminative Domain Adaptation (ADDA)



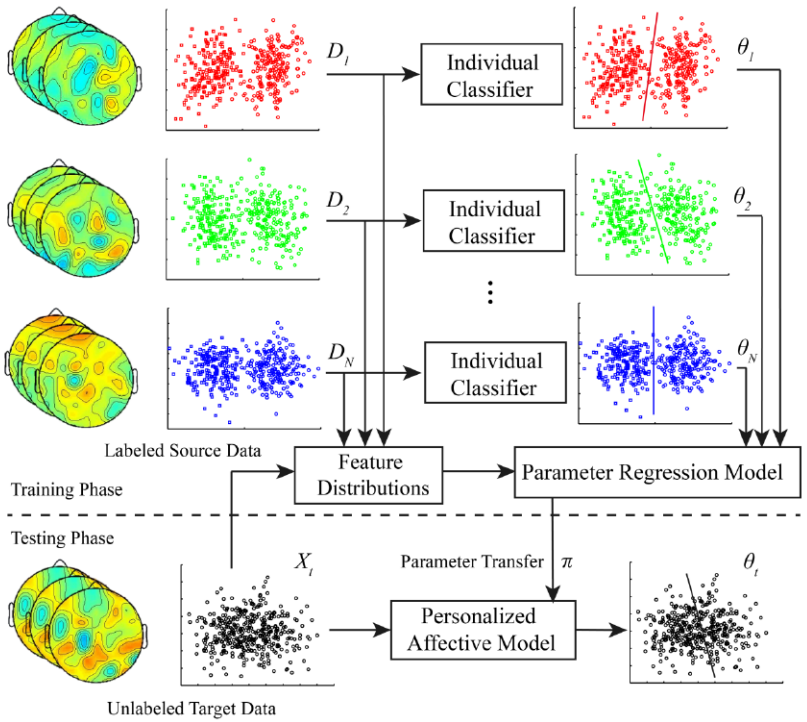
人工智能算法进阶

（一）迁移学习

3. **参数迁移**

根据源域模型学习目标域模型的参数

代表模型是直推式参数迁移（Transductive Parameter Transfer, TPT）



直推式参数迁移示意图

人工智能算法进阶

（一）迁移学习

4. 域泛化（DG）

主要的域泛化方法可分为域对齐、元学习、数据增强、集成学习、自监督学习等。

1) Domain alignment：域对齐方法假设对源域转移不变的特征，当转移到任何未知的目标域时也应具有鲁棒性，其核心思想是最小化源域时间的差异，以学习域不变的特征表示。

2) 元学习：元学习中和DG最相关的算法为MAML，使模型在训练时暴露于domain shift，以期能更好地处理在未知域上的domain shift。

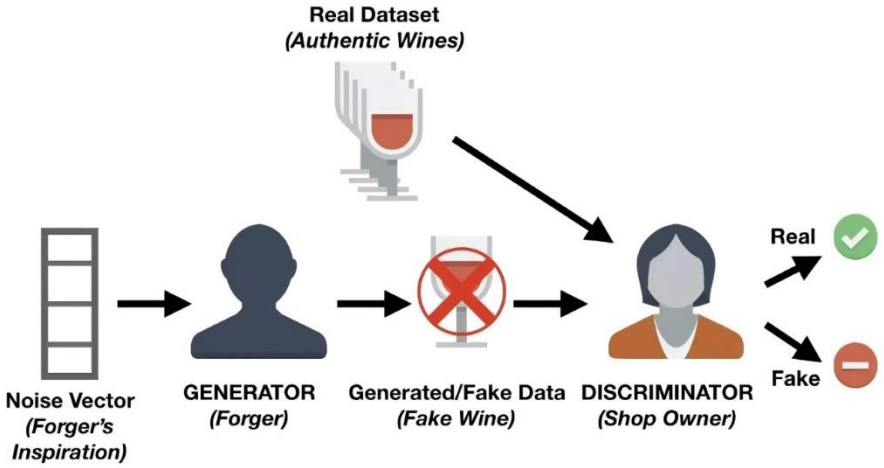
3) 数据增强：以图片为例，对图片进行形状、位置、纹理、灰度、噪声点等处理，从而增强数据；此外，有意地向梯度下降的反方向去增强数据，学习最坏情况下的数据分布，也能提高域泛化的能力。

4) 集成学习：使用不同初始化权重或不同训练数据训练多个相同的模型，来集成预测。

人工智能算法进阶

（二）对抗生成

生成对抗网络，是一种包含两个网络（或网络集合）的深度学习框架。这两个网络在互相对抗，互相学习，并完成一个生成型任务。

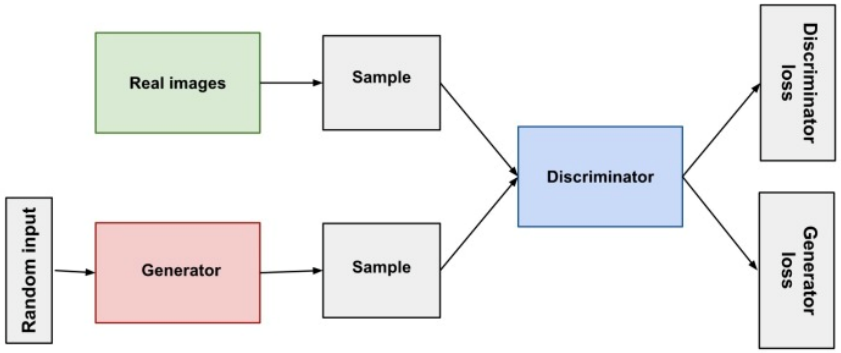


应用：
生成接近真实的图片数据以实现数据扩增；
基于低分辨率图片生成高分辨率图片；
基于某一种图片类型生成出另一种图片类型以完成图片转换；
基于静态图片生成动态图片等。

人工智能算法进阶

（二）对抗生成

1. 模型框架



生成对抗网络框架

（1）首先输入为一串随机的噪音向量Z

（2）随后生成器G会生成假图片。

（3）假图片$\hat{X}$，与真实图片X混合在一起输入到一个判别器D中

（4）根据判别结果和图片真假计算一个损失函数

人工智能算法进阶

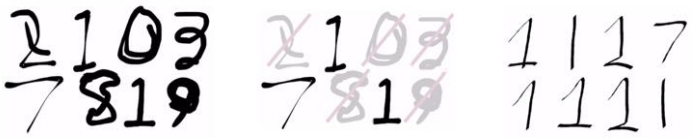
（二）对抗生成

2. 模型训练

模型训练不稳定的问题

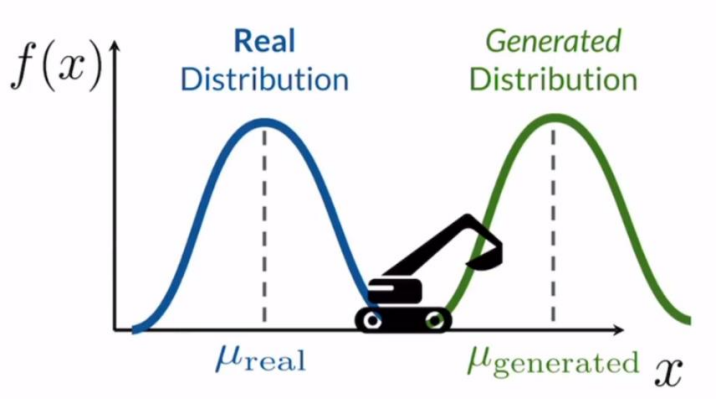
（1）模式崩溃（mode collapse）

（2）梯度消失



优化GAN：损失函数Wasserstein loss

对Earth Mover’s Distance（EMD）的一种近似算法



Earth Mover’s Distance示意图

人工智能算法进阶

（二）对抗生成

3. 模型评估

Fréchet Inception Distance（FID）

其做法是，先随机生成一批图片，然后用常用的一种图片特征提取器如InceptionV3，提取每个生成的图片的特征向量。

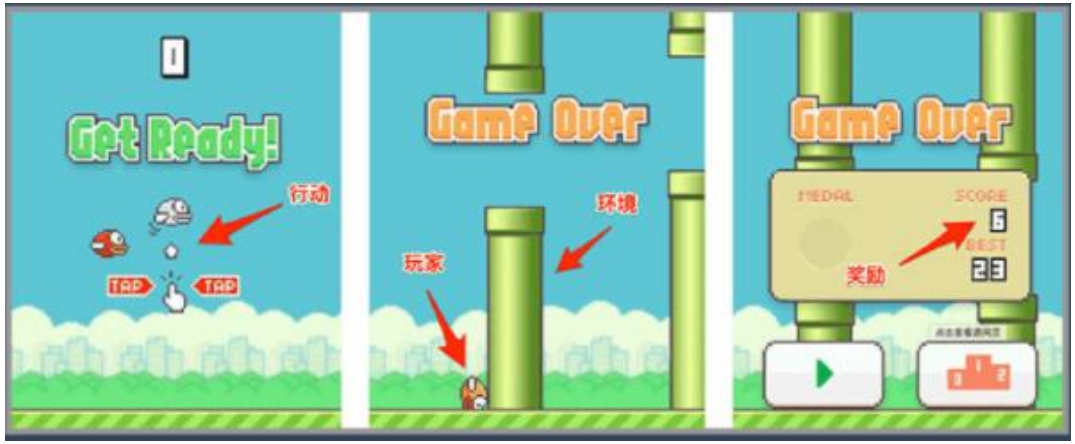
同样地，也提取多个真实图片的特征向量。假设这个特征向量满足多元正态分布，那么就可以用Fréchet Distance计算生成图片的特征向量的分布和真实图片的特征向量的分布之间的距离。

人工智能算法进阶

（三）强化学习

1. 强化学习概述

强化学习讨论的问题是一个智能体(agent) 怎么在一个复杂不确定的环境(environment)里面去极大化它能获得的奖励。



人工智能算法进阶

（三）强化学习

2. 马尔科夫决策过程

MDP 的策略完全取决于当前状态

P[St+1|St]=P[St+1|S1,...St]

- 状态：$s \in S$ 有限状态 state 集合，s 表示某个特定状态
- 动作：$a \in A$ 有限动作 action 集合，a 表示某个特定动作
- 状态转移概率：$T(S,a,S') \sim P_r(s'|sa')$
- 奖励：$R(s,a) = E[R_{t+1}|s,a]$
- 累计回报G

人工智能算法进阶

（三）强化学习

贝尔曼方程

可以用下一状态的动作价值函数来表示当前状态的动作价值函数

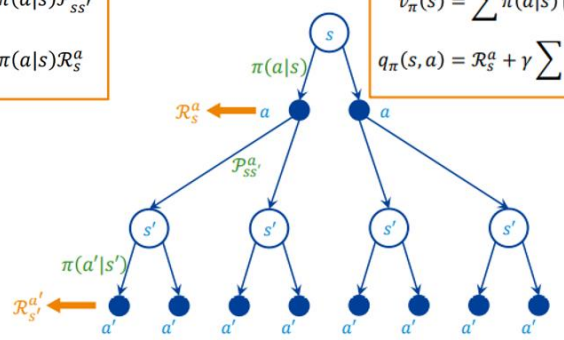
$$
\begin{aligned}
v(s) &= \mathbb{E}[G_t \mid S_t = s] \\
&= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \ldots\right) \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]
\end{aligned}
$$

$$\mathcal{P}^{\pi}_{ss'} = \sum \pi(a|s)\mathcal{P}^{a}_{ss'}$$

$$\mathcal{R}^{\pi}_{s} = \sum \pi(a|s)\mathcal{R}^{a}_{s}$$

$$v_\pi(s) = \sum \pi(a|s)\left(\mathcal{R}^a_s + \gamma \sum \mathcal{P}^a_{ss'} v_\pi(s')\right)$$

$$q_\pi(s,a) = \mathcal{R}^a_s + \gamma \sum \mathcal{P}^a_{ss'} \sum \pi(a'|s') q_\pi(s',a')$$



通过优化两个值函数v，q来找到最优的策略，这时候在MDP上有最好的策略产生。在这里优化的含义是，让价值函数能准确表示出选择这个状态或者动作的时候所能在未来得到的准确的收益。

人工智能算法进阶

（三）强化学习

3. 应用

（1）机器人辅助手术（RAS）

计算机视觉技术（如用于目标检测 / 分割和立体视觉的 CNN）可以根据图像数据重建开放性伤口的样子，然后通过解决路径优化问题生成缝合或打结轨迹，该路径优化问题试图在考虑外部约束（如关节限制和障碍）的同时找到最优轨迹。

（2）追踪/定位问题

关键点在临床的应用里面起到非常重要的作用，包括医生可以通过关键点去找到一些解剖结构，或是标准切面，起到导航（navigate）的作用。同时，关键点也可以用于生物参数的测量（宽度，长度，大小等等），产前里面常测的侧脑室宽度，小脑横径等等，有利于后续的分析诊断。