人工神经网络2

向量运算

- 向量默认列向量

$$\boldsymbol{a}=\begin{bmatrix}\boldsymbol{a}_1\\ \boldsymbol{a}_2\\ \vdots\\ \boldsymbol{a}_{n-1}\\ \boldsymbol{a}_n\end{bmatrix},\quad \boldsymbol{a}^T=[\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_{n-1},\boldsymbol{a}_n]$$

- 向量的模长: $\|\boldsymbol{a}\|_2^2=<\boldsymbol{a},\boldsymbol{a}>=\boldsymbol{a}^T\boldsymbol{a}$

- 向量间的欧式距离:

$$\|\boldsymbol{a}-\boldsymbol{b}\|_2^2=<\boldsymbol{a}-\boldsymbol{b},\boldsymbol{a}-\boldsymbol{b}>=(\boldsymbol{a}-\boldsymbol{b})^T(\boldsymbol{a}-\boldsymbol{b})$$

向量求梯度

- $\nabla\|\boldsymbol{x}\|_2^2 = \nabla(\boldsymbol{x}^T\boldsymbol{x}) = 2\boldsymbol{x}$
  - 证明一（直接计算）：$\dfrac{\partial\|\boldsymbol{x}\|_2^2}{\partial x_i} = \dfrac{\partial \sum_j x_j^2}{\partial x_i} = \dfrac{\partial x_i^2}{\partial x_i} = 2x_i$

$$\nabla \| \boldsymbol{x}-\boldsymbol{b} \|_2^2 = \nabla < \boldsymbol{x}-\boldsymbol{b}, \boldsymbol{x}-\boldsymbol{b} >$$
$$= \boldsymbol{x}-\boldsymbol{b}+\boldsymbol{x}-\boldsymbol{b} = 2(\boldsymbol{x}-\boldsymbol{b})$$

$$\| \boldsymbol{A}\boldsymbol{x}-\boldsymbol{b} \|_2^2 = < \boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}, \boldsymbol{A}\boldsymbol{x}-\boldsymbol{b} >$$
$$= (\boldsymbol{A}\boldsymbol{x})^T \boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{b}$$
$$= \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{b}$$

$$\nabla \| \boldsymbol{A}\boldsymbol{x}-\boldsymbol{b} \|_2^2 = \nabla(\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{b})$$
$$= 2(\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{A})$$

Laplace矩阵

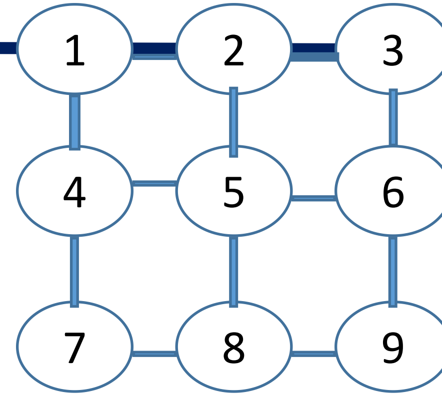
L aplace算子离散公式：

$$\boxed{\begin{aligned}\Delta f &= f_{xx} + f_{yy} = [(\Delta f)_{ij}] \\ &= [f_{i+1j} - 4f_{ij} + f_{i-1j} + f_{ij+1} + f_{ij-1}]\end{aligned}}$$

Δf：离散后的Δ算子生成矩阵 L:

$$\Delta f = [(\Delta f)_{ij}] = \mathrm{L}\begin{bmatrix} f_{11} \\ f_{12} \\ \vdots \\ f_{33} \end{bmatrix},\quad \mathrm{L}\text{ 为}\boldsymbol{Laplace}\text{矩阵}$$

等值边界条件



$$\boldsymbol{\Delta f}=[(\boldsymbol{\Delta f})_{ij}]=\mathrm{L}\begin{bmatrix}\boldsymbol{f}_{11}\\ \boldsymbol{f}_{12}\\ \vdots\\ \boldsymbol{f}_{33}\end{bmatrix},$$

L 为***Laplace***矩阵

$$\begin{pmatrix}-4*f_1+f_2+f_4+f_1+f_1\\ -4*f_2+f_1+f_3+f_5+f_2\\ -4*f_3+f_2+f_6+f_3+f_3\\ -4*f_4+f_1+f_7+f_5+f_4\\ -4*f_5+f_2+f_4+f_8+f_6\\ -4*f_6+f_5+f_3+f_9+f_6\\ -4*f_7+f_8+f_4+f_7+f_7\\ -4*f_8+f_5+f_7+f_9+f_8\\ -4*f_9+f_8+f_6+f_9+f_9\end{pmatrix}=\begin{bmatrix}-2&1&0&1&0&0&0&0&0\\ 1&-3&1&0&1&0&0&0&0\\ 0&1&-2&0&0&1&0&0&0\\ 1&0&0&-3&1&0&1&0&0\\ 0&1&0&1&-4&1&0&1&0\\ 0&0&1&0&1&-3&0&0&1\\ 0&0&0&1&0&0&-2&1&0\\ 0&0&0&0&1&0&1&-3&1\\ 0&0&0&0&0&1&0&1&-2\end{bmatrix}\begin{bmatrix}f_1\\ f_2\\ f_3\\ f_4\\ f_5\\ f_6\\ f_7\\ f_8\\ f_9\end{bmatrix}$$

ADMM算法：

1.初始化：$\boldsymbol{p}^k=0, \boldsymbol{\lambda}^k=0, \boldsymbol{k}=0$

2. do

求解u - 子问题 $\boldsymbol{u}^{k+1}$

$$-\boldsymbol{\beta}\Delta\boldsymbol{u}+\boldsymbol{\alpha}\boldsymbol{I}\boldsymbol{u}=-\boldsymbol{\beta}\operatorname{div}(\boldsymbol{p}^k)-\operatorname{div}(\boldsymbol{\lambda}^k)+\boldsymbol{\alpha}\boldsymbol{u}^0$$

求解p - 子问题 $\boldsymbol{p}^{k+1}$

$$\boldsymbol{p}_{ij}=\begin{cases}(1-\dfrac{1}{\boldsymbol{\beta}\,|\,\boldsymbol{c}_{ij}\,|})\,|\,\boldsymbol{c}_{ij}\,|, \boldsymbol{\beta}\,|\,\boldsymbol{c}_{ij}\,|\geq 1\\ 0, \boldsymbol{else}\end{cases}$$

更新$\boldsymbol{\lambda}$：

$$\boldsymbol{\lambda}^{\mathrm{k}+1}=\boldsymbol{\lambda}^k+\boldsymbol{\beta}(\boldsymbol{p}^{k+1}-\nabla\boldsymbol{u}^{k+1})$$

$$\boldsymbol{while}(\|\boldsymbol{u}^{k+1}-\boldsymbol{u}^k\|_2^2\geq 0.000001)$$

拟合存在问题

- 一元拟合问题：x为一维数据

  ①最小二乘拟合方法

  ②线性拟合，二次曲线拟合，三次曲线拟合及多项式拟合。

  $y = ax+b$；$y = ax^2+bx+c$

  ③a，b, c 称为参数，

  ④x, x^2 称为拟合所需的特征(feature)

- 多元拟合问题：即 x为高维数据

  二次拟合

  $y = ax^2+bx+c$

$$h_\theta(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2$$

- 50*50像素的灰度图片：大约包含上百万个特征，
- 彩色图片，特征会增加至上千万

- 当数据库包含上万甚至上亿张图片时，**拟合方法**很难去拟合这些数据
- 寻找一种数学模型， 能够拟合量大，特征维度高的数据
- **即： 人工神经网络模型**

- 大脑可视作为1000多亿神经元组成的神经网络

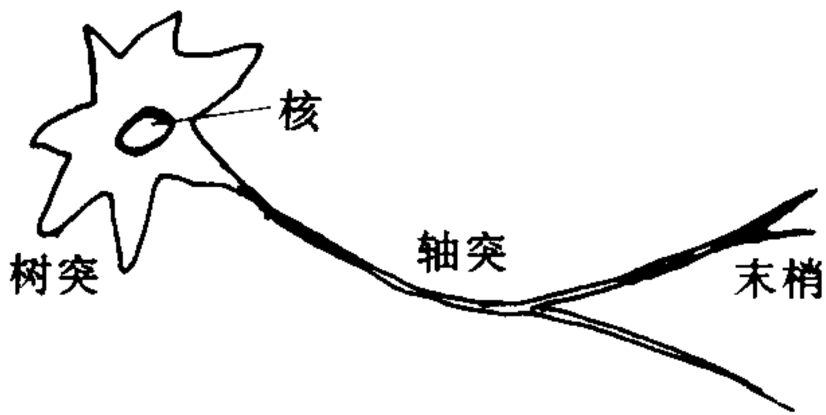
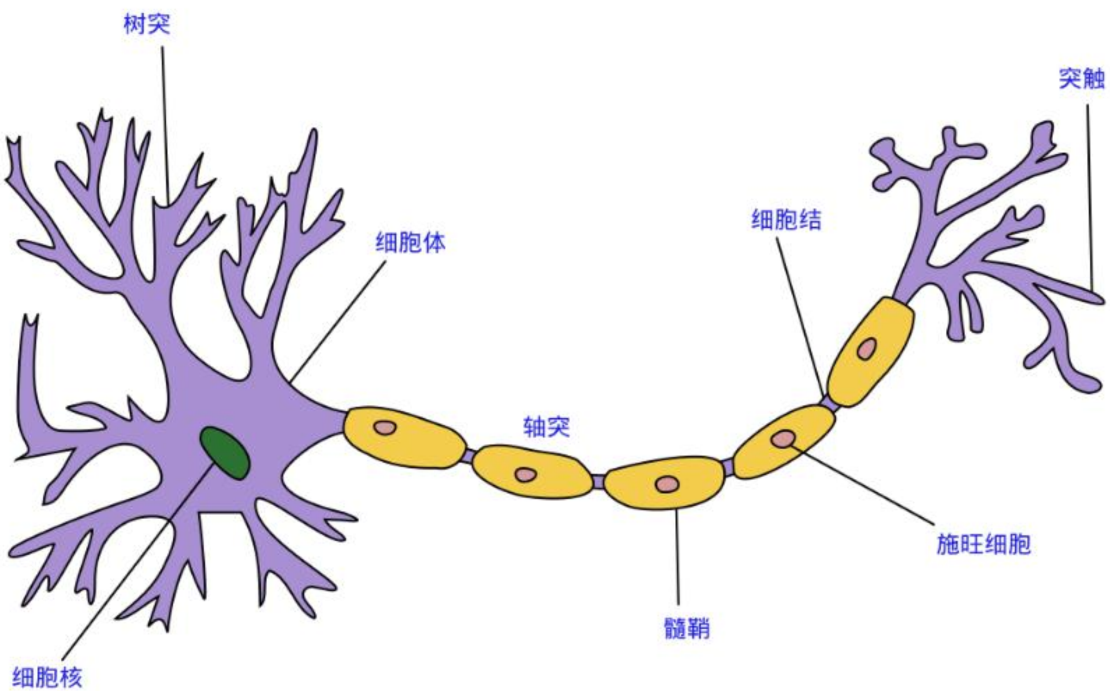


图3 神经元的解剖图

1.1 神经元

- **1904年生物学家就已经知晓了神经元的组成结构**
  - 多个树突，主要用来接受传入信息；
  - 一条轴突，轴突尾端有许多轴突末梢，给其他多个神经元传递信息。
  - 突触，轴突末梢跟其他神经元的树突产生连接，从而传递信号。
  - 人脑中的神经元形状：



- 神经元的信息传递和处理是一种电化学活动.
- 树突由于电化学作用接受外界的刺激；通过胞体内的活动体现为轴突电位，当轴突电位达到一定的值则形成神经脉冲或动作电位；
- 再通过轴突末梢传递给其它的神经元.
- 从控制论的观点来看；这一过程可以看作一个多输入单输出非线性系统的动态过程

神经网络研究的两个方面

- 从生理上、解剖学上进行研究
- 从工程技术上、算法上进行研究

- 人工神经网络

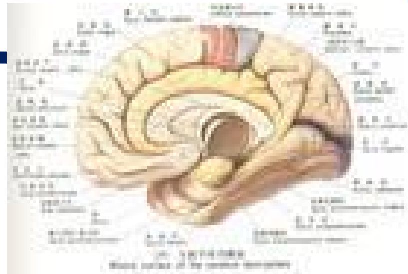
①Artificial Neural Network，ANN，简称神经网络（Neural Network，NN）。

②是一种模仿生物神经网络（动物的中枢神经系统，特别是大脑）的结构和功能的数学模型或计算模型，用于对函数进行**估计或近似**。

③模拟人类的大脑，造出会思考的机器

④神经元是构成神经网络的最基本的单元

人工神经网络的基本特点



（1）可处理非线性

（2）并行结构．对神经网络中的每一个神经元来说；其运算都是同样的．这样的结构最便于计算机并行处理．

（3）具有学习和记忆能力．一个神经网络可以通过训练学习判别事物；学习某一种规律或规则．神经网络可以用于联想记忆．

（4）对数据的可容性大．在神经网络中可以同时使用量化数据和质量数据（如好、中、差、及格、不及格等）．

（5）神经网络可以用大规模集成电路来实现．如美国用256个神经元组成的神经网络组成硬件用于识别手写体的邮政编码．

人工神经网络研究进展

- 1943 提出神经元模型
- 1949 提出学习算法及规则
- 1957 提出感知器-人工神经元模型
- 1969 <感知器>发表,串行计算机全胜
- 1982 建立人工神经网络模型
- 1986 提出BP神经网络算法
- 2006 深度信念网络

MP神经元模型

- MP神经元模型

  1943年，参考了生物神经元的结构，Warren McCulloch和Walter Pitts提出MP神经元模型

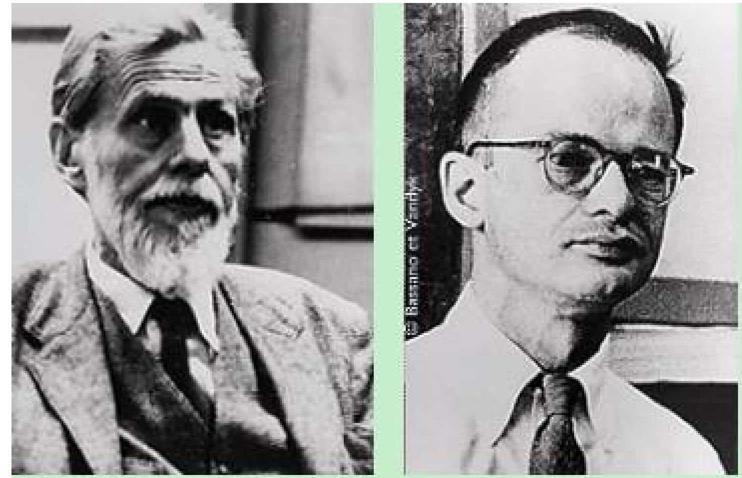


图5 Warren McCulloch（左）和 Walter Pitts（右）

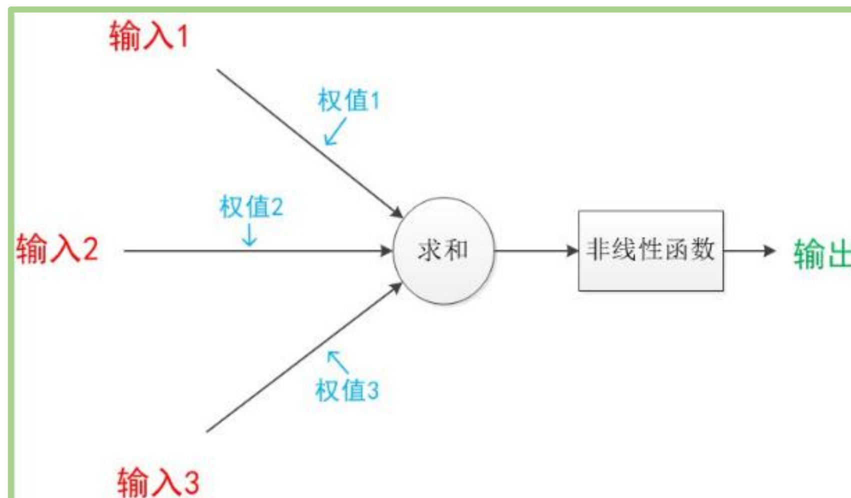
- MP神经元结构：

  输入： 可以类比为神经元的树突，

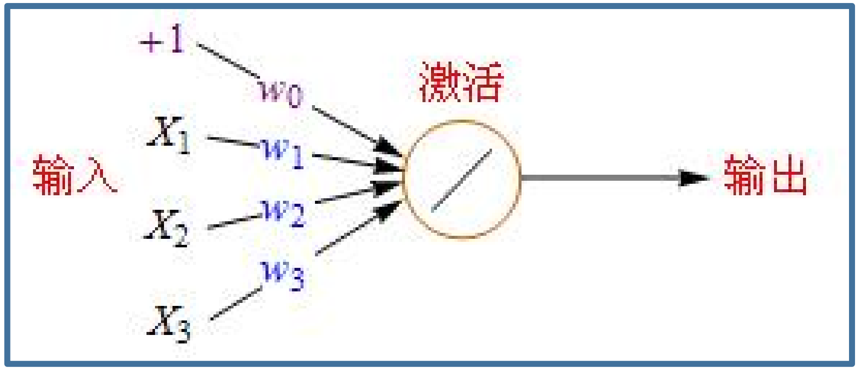
  输出： 可以类比为神经元的轴突，

  计算： 则可以类比为细胞核

- 箭头线：“连接”具有‘权值’。

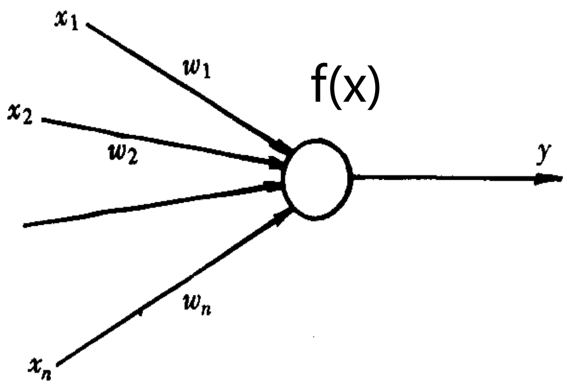


- **神经元构成：**



- +1代表偏移值(偏置项, Bias Units)；
- X1,X2,X2代表初始特征；
- w0,w1,w2,w3代表权重(Weight)，即参数，是特征的缩放倍数；特征经过缩放和偏移后全部累加起来，
- 此后还要经过一次激活运算然后再输出

神经元的数学模型



- 其中x＝ （x_1， …x_m）^{T}输入向量，y为输出，w_i是权系数；
- 输入与输出具有如下关系:

$$y = f\left(\sum_{i=1}^{m} w_i x_i - \theta\right)$$

θ 为阈值，f（X）是激活函数；它可以是线性函数，也可以是非线性函数.

激活函数作用

- 神经网络要引入激活函数来给神经网络增加一些非线性的特性，
- 如果没有激活函数，

  多层神经网络退化为一个多层的线性回归模型，

  难以学习如图像、音频、文本等复杂数据的特征。

- 常见的激活函数大多是非线性函数

  Sigmoid(S形曲线)

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

双曲正切函数

$$y = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

激活函数的性质

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

$$f'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = f(x)*(1-f(x))$$

例，

$$z = \sum_{i=1}^{m} w_i x_i - \theta$$

取激活函数为符号函数

$$\operatorname{sgn}(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} > 0 \\ 0, & \boldsymbol{x} = 0 \\ -1, & \boldsymbol{x} < 0 \end{cases}$$

则

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{z}) = \begin{cases} 1, & \sum_{i=1}^{m} \boldsymbol{w}_i \boldsymbol{x}_i > \boldsymbol{\theta}, \\ -1, & \sum_{i=1}^{m} \boldsymbol{w}_i \boldsymbol{x}_i < \boldsymbol{\theta}, \end{cases}$$

MP神经元影响

- **影响**

  ①1943年发布的MP模型，简单，建立了神经网络的地基。

  然而，MP模型中，权重的值都是预先设置的，因此不能学习

  ②1949年心理学家Hebb提出了Hebb学习率，

  人脑神经细胞的突触（也就是连接）上的强度上可以变化的。

  于是科学家们开始考虑用调整权值的方法来让机器学习。

  ③限于当时的计算机能力，近10年后，第一个真正意义的神经网络才诞生。

- 1958年，计算科学家Rosenblatt提出了由**两层神经元**组成的神经网络。

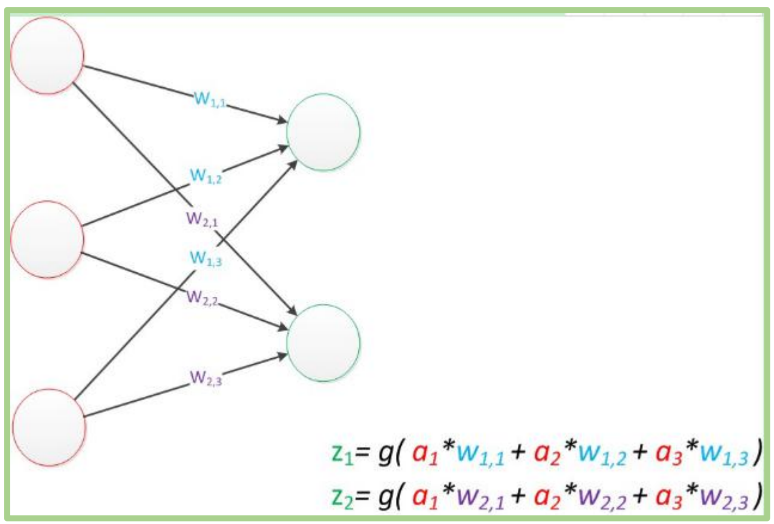
  “感知器”（Perceptron）

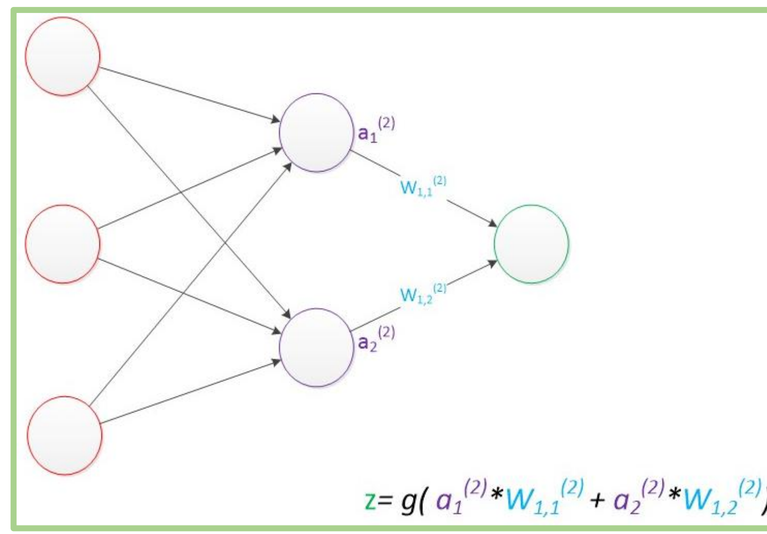
2 感知器：Perceptron

- 感知器：多**层神经元**组成的神经网络

  首个可以学习的人工神经网络

- 感知器的结构：
  - 单层感知器(左):多个M-P模型的累叠
  - **多层感知器(右):其中一层是固定权值的**





感知机分类例子：如何分辨香蕉和苹果

- **香蕉、苹果特征： 颜色p_1和形状 p_2，** 1代表苹果，-1代表香蕉

  输入p_1，p_2：苹果红色(1),香蕉黄色(-1); 苹果圆形(1),香蕉弯形(-1)

  预设权重： $w_1=w_2=1$，$b=0$

  输出z: $z=w_1*p_1+w_2*p2+b$

  对苹果的鉴别结果： $z = 1*1+1*1 = 2$;

  对香蕉的鉴别结果： $z = -1*1+1*(-1)= -2$;

- 对结果z进行处理，即可实现对二者进行归类
- **问题：**

  这里的权重如果换一个其它值，则影响分类结果

- **香蕉、苹果特征：颜色p_1和形状 p_2，1代表苹果，-1代表香蕉**

  输入p_1: 苹果红色(1),香蕉黄色(-1); p_2苹果圆形(1),香蕉弯形(-1)

  预设权重： 取$w_1=1,w_2=-1,b=0$,

  输出z: $z=w_1*p_1+w_2*p2+b$

  对苹果的鉴别结果： $z = 1*1-1*1 = 0$;

  对香蕉的鉴别结果： $z = -1*1+1*(1)= 0$;

- 无法进行分类

  对于随意选取的参数，如何使输出值依旧正确？

  感知器的学习功能

感知器的学习规则：

- 修改感知器的权值w_i与偏置b

$$w_{new} = w_{old} + e * p, \quad b_{new} = b_{old} + e$$

其中e误差，$e = t - a$，　t为期望值，a为实际输出

- 学习规则：计算误差，　e = t - z
- 代入以上公式重新计算权重w_i与偏置b
- 重新计算输出值
- 依次迭代

感知机总结

- **M-P模型：一个神经元结构，但是没有参数学习的过程**
- 单层感知机引入损失函数，并提出了学习的概念，学习能力有限
- **可以解决简单的线性可分问题，无法处理非线性问题**
- 多层感知机通过增加层数解决非线性问题

  需要人为固定一层参数，只能训练其中一层。
- 1986年Hinton提出了反向传播算法，使得训练多层网络成为可能。

应用例子

- 以下两类蚊子，15个样本数据，样本特征为3，15个输出结果

翼长	触角长	类别	目标值
1.78	1.14	Apf	0.9
1.96	1.18	Apf	0.9
1.86	1.20	Apf	0.9
1.72	1.24	Af	0.1
2.00	1.26	Apf	0.9
2.00	1.28	Apf	0.9
1.96	1.30	Apf	0.9
1.74	1.36	Af	0.1

翼长	触角长	类别	目标t
1.64	1.38	Af	0.1
1.82	1.38	Af	0.1
1.90	1.38	Af	0.1
1.70	1.40	Af	0.1
1.82	1.48	Af	0.1
1.82	1.54	Af	0.1
2.08	1.56	Af	0.1

建立神经网络

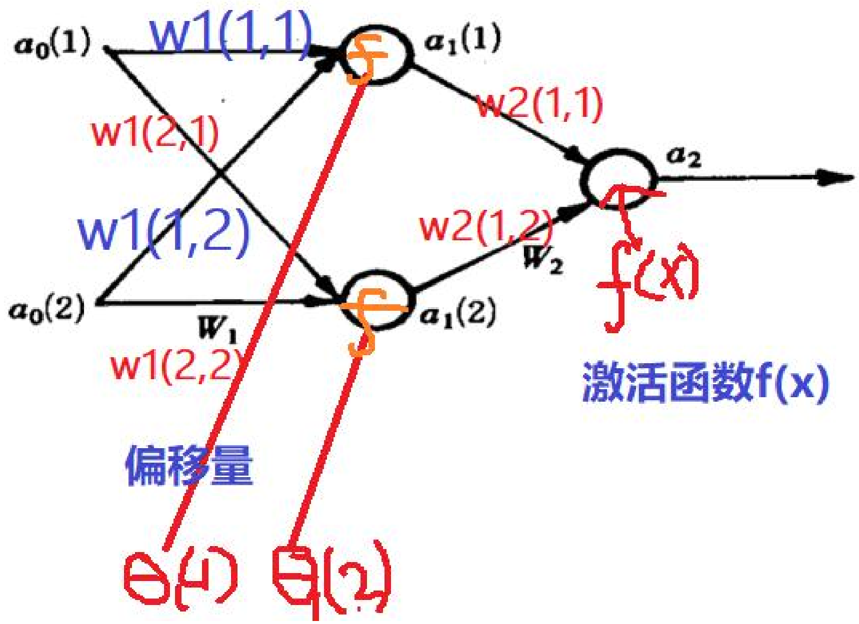
- 输出目标：t = 0.9时为Apf， t = 0.1时为Af
- 设权重系数为:

$$W_1 = \begin{bmatrix} w_1(1,1) & w_1(1,2) & w_1(1,3) \\ w_1(2,1) & w_1(2,2) & w_1(2,3) \end{bmatrix}$$

$$W_2 = \begin{bmatrix} w_2(1,1) & w_2(1,2) & w_2(1,3) \end{bmatrix}$$

- 偏移量为:

$$\boldsymbol{\theta}_1(1), \boldsymbol{\theta}_1(2), \boldsymbol{\theta}_2(1),$$



- 第一层权重与特征求和计算公式:

$$u_1(1) = w_1(1,1)a_0(1) + w_1(1,2)a_0(2) - \theta_1(1)$$
$$u_1(2) = w_1(2,1)a_0(1) + w_1(2,2)a_0(2) - \theta_1(2)$$

- 第一层激活函数计算

$$a_1(1) = f(u_1(1)) \qquad a_1(2) = f(u_1(2))$$

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

$$a_1(i) = f(u_1(i)) = \frac{1}{1+\exp(-u_1(i))} \qquad i = 1,2$$

- 第二层权重与第一层输出结果求和

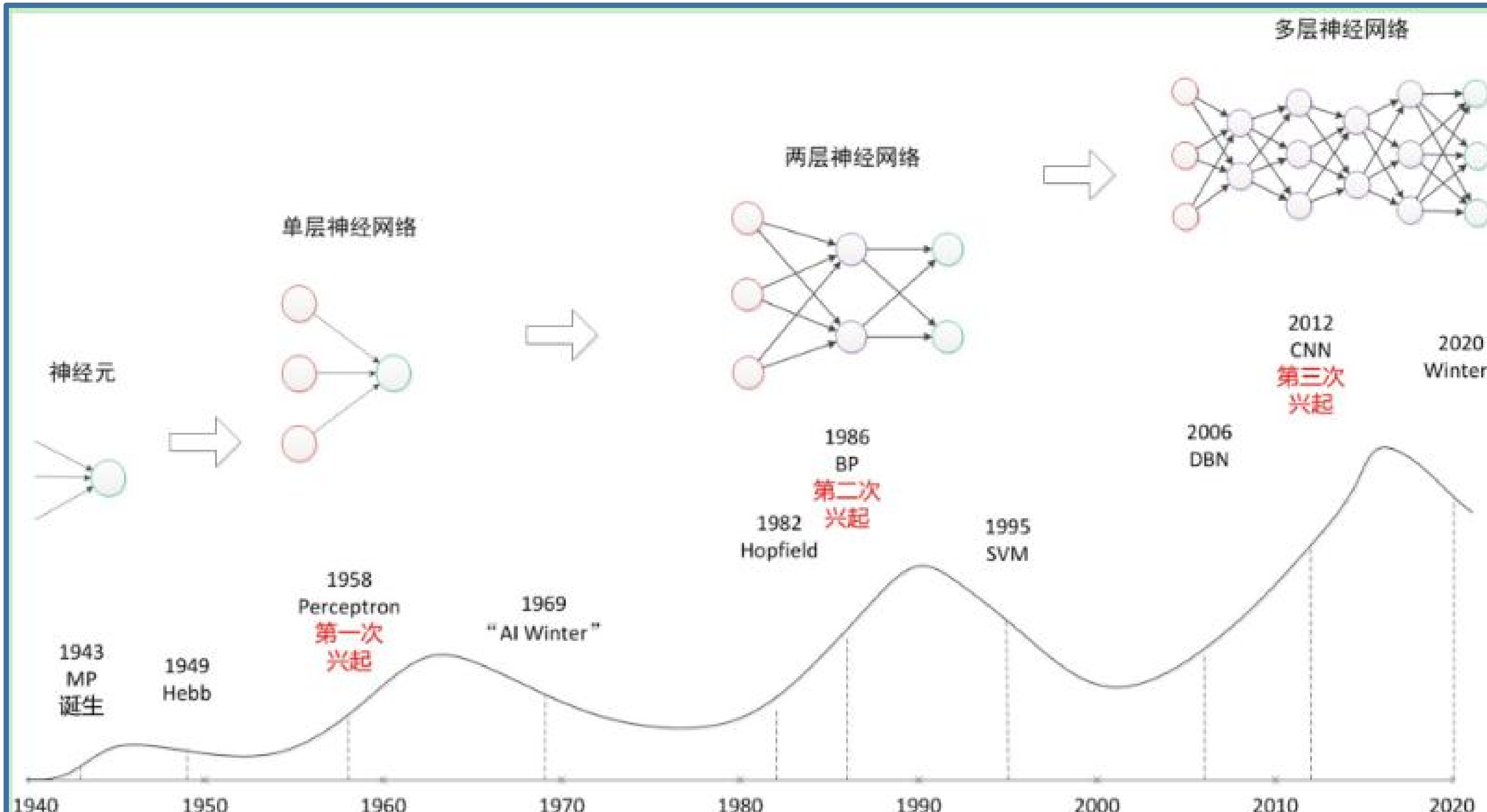
$$\boldsymbol{u}_2(1) = \boldsymbol{w}_2(1,1)\boldsymbol{a}_1(1) + \boldsymbol{w}_2(1,2)\boldsymbol{a}_1(2) - \boldsymbol{\theta}_2(1),$$

- 激活函数运算

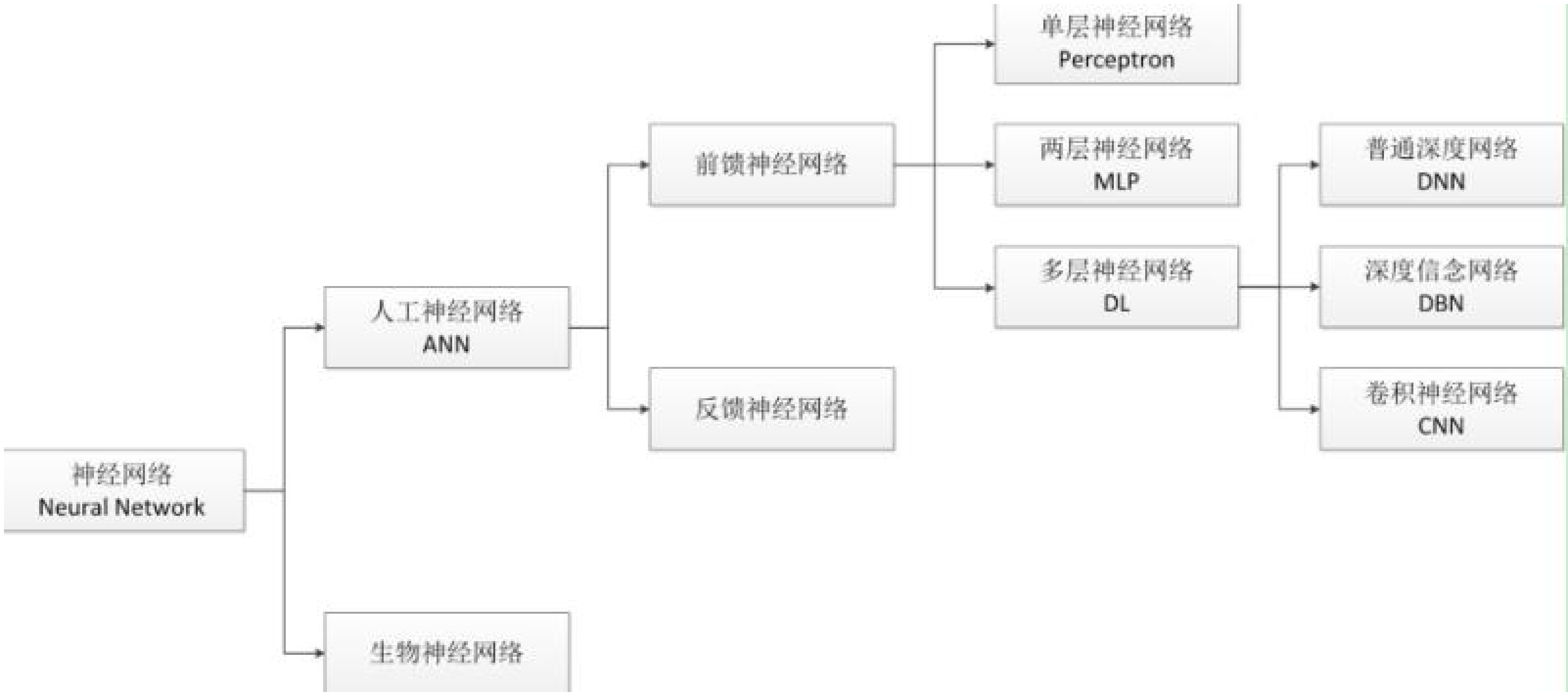
$$\boldsymbol{a}_2(1) = \boldsymbol{f}(\boldsymbol{u}_2(1)),$$

3 神经网络

- 大量感知器进行组合
- 本质：通过参数与激活函数来拟合特征与目标之间的真实函数关系
- 神经网络发展历史



神经网络分类



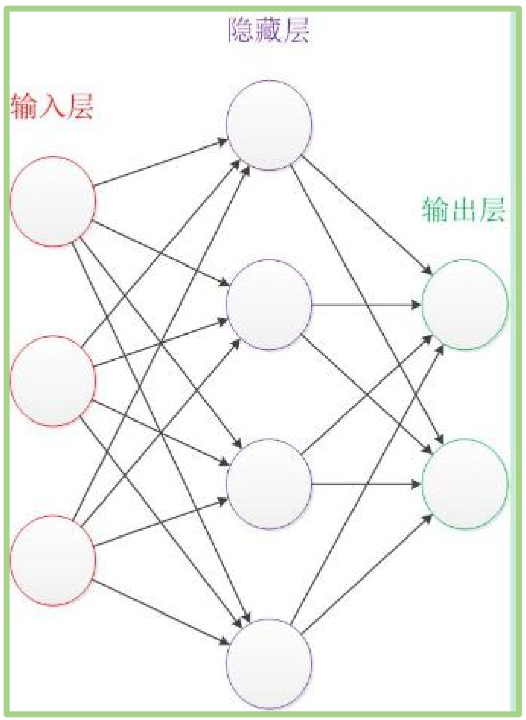
神经网络
Neural Network
人工神经网络
ANN
生物神经网络
前馈神经网络
反馈神经网络
单层神经网络
Perceptron
两层神经网络
MLP
多层神经网络
DL
普通深度网络
DNN
深度信念网络
DBN
卷积神经网络
CNN

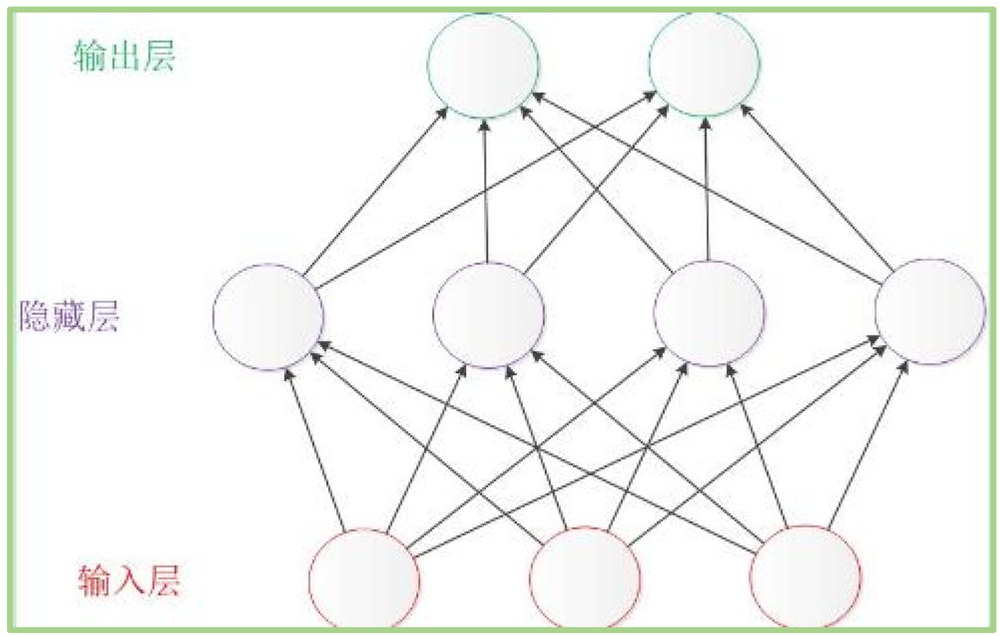
- 经典神经网络的构成：三个部分(从左至右，从下至上两种结构)

  ①红色的是输入层，3个输入单元

  ②绿色的是输出层，2个单元

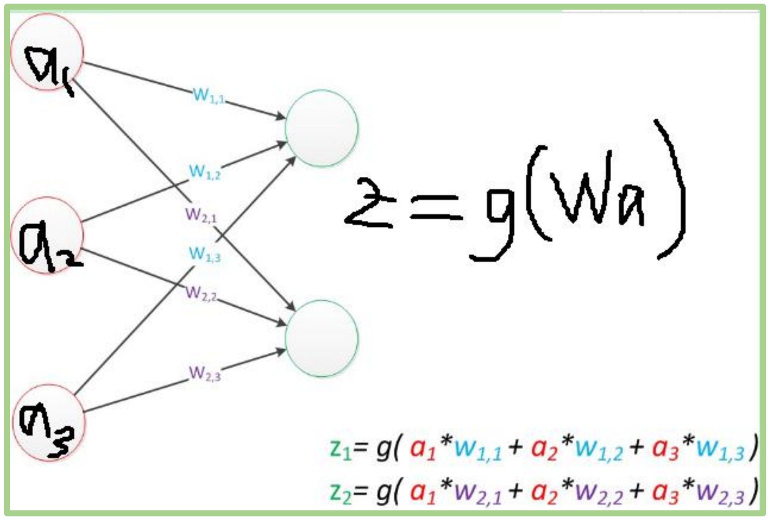
  ③紫色的是中间层（也叫隐藏层），4个单元。

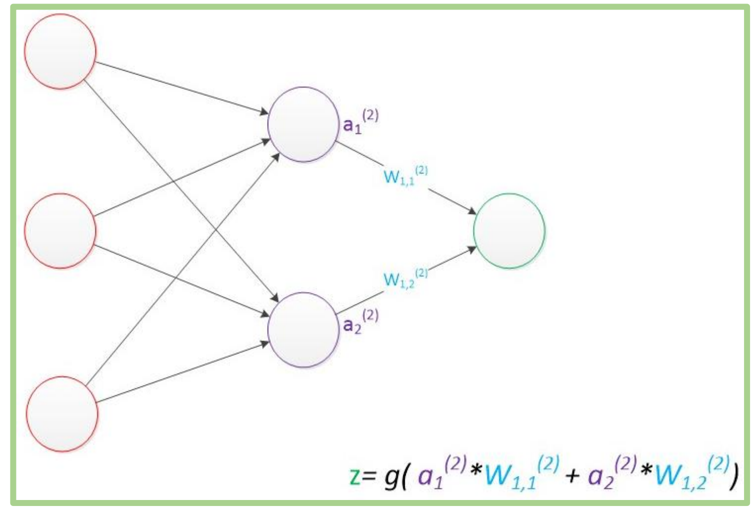




4 两层神经网络

- 结构：输入层，输出层，中间层



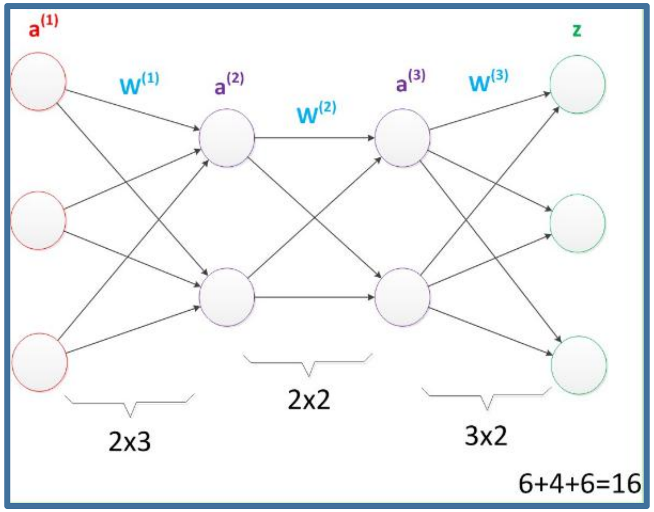


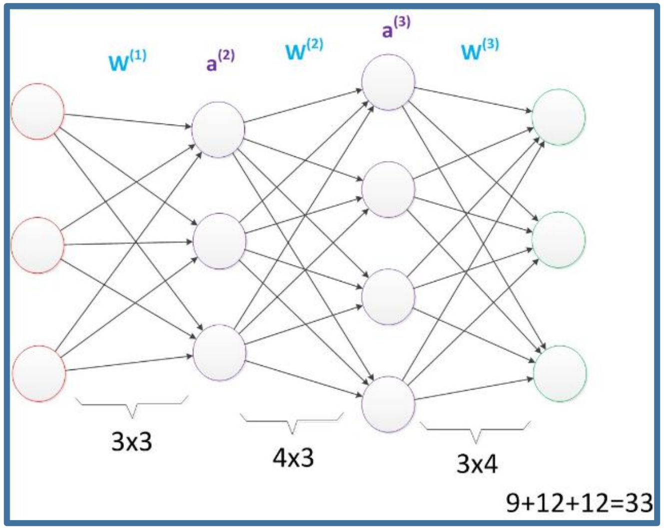
权重矩阵，$a^2 = g(W^1 a)$,　　　$z = g(W^2 a^2)$

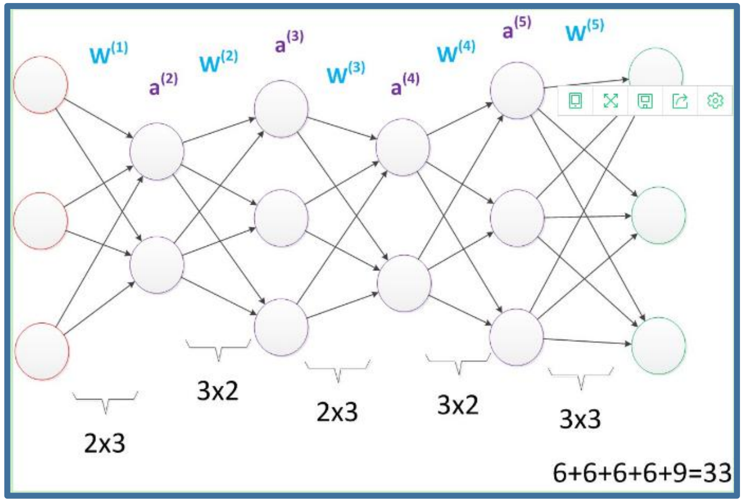
5 :多层神经网络与参数

- 多层神经网络中的层数增加了很多。

  具有更深入的表示特征，以及更强的函数模拟能力。







参数的意义及求解方案

- **神经网络：**模拟特征与目标之间的真实关系的方法，
  更多的参数意味着其模拟的函数可以更加的复杂
  在参数数量一样的情况下，
  更深的网络往往具有比浅层的网络更好的识 别效率
- 学习过程：
  不断的修改w、b两个参数值，使最终的误差达到最小。

- 如何有效的修改这些参数，使误差最小化
  80年代，**误差反向传播算法(BP算法)的提出**，提供了有效的解决方案

链式法则

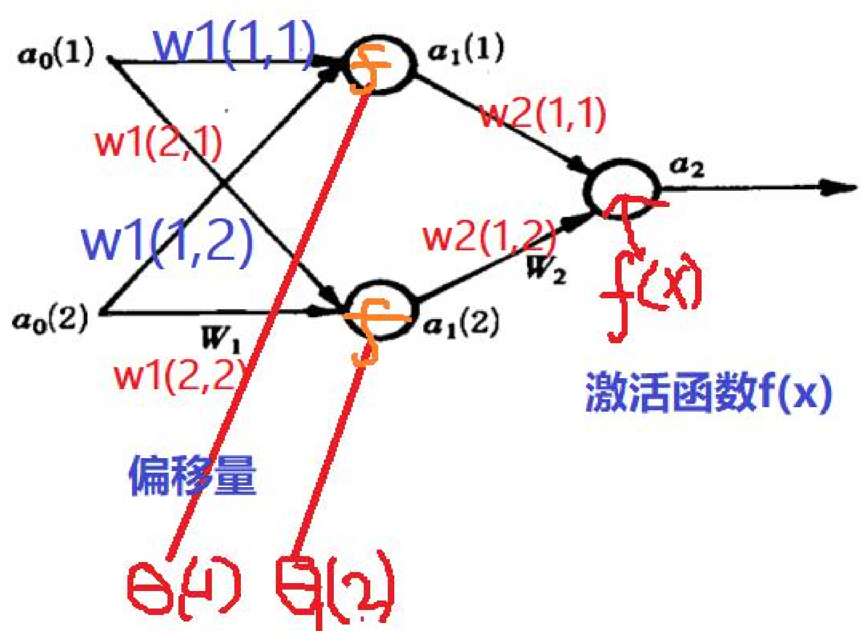
定理　如果函数$u=\phi(t)$及$v=\psi(t)$都在点t可导，函数$z=f(u,v)$在对应点(u,v)具有连续偏导数，则复合函数$z=f[\phi(t),\psi(t)]$在对应点t可导，且其导数可用下列公式计算：

$$\frac{dz}{dt}=\frac{\partial z}{\partial u}\frac{du}{dt}+\frac{\partial z}{\partial v}\frac{dv}{dt}.$$

- 本质：梯度下降法修改权重信息

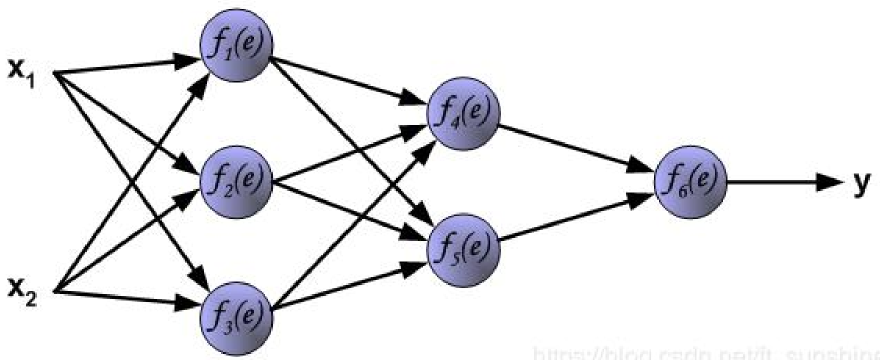
$$\Delta\omega_{j\kappa} = -\eta\frac{\partial E}{\partial\omega_{j\kappa}} j = 0, 1, 2, \ldots, m; \qquad \kappa = 1, 2, \ldots, \ell$$

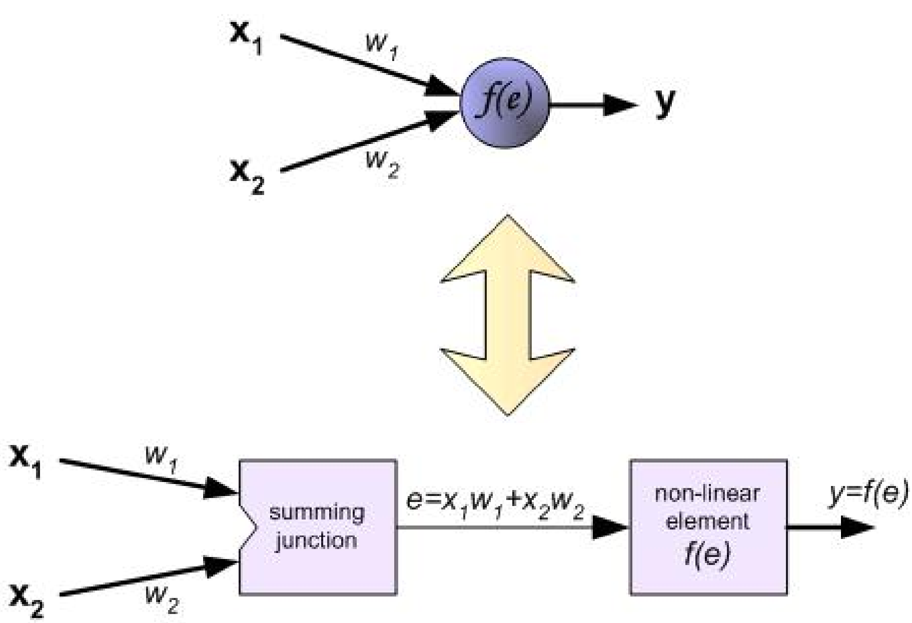
$$\Delta v_{ij} = -\eta\frac{\partial E}{\partial v_{ij}} i = 0, 1, 2, \ldots, n; \qquad j = 1, 2, \ldots, m$$



BP算法推导过程

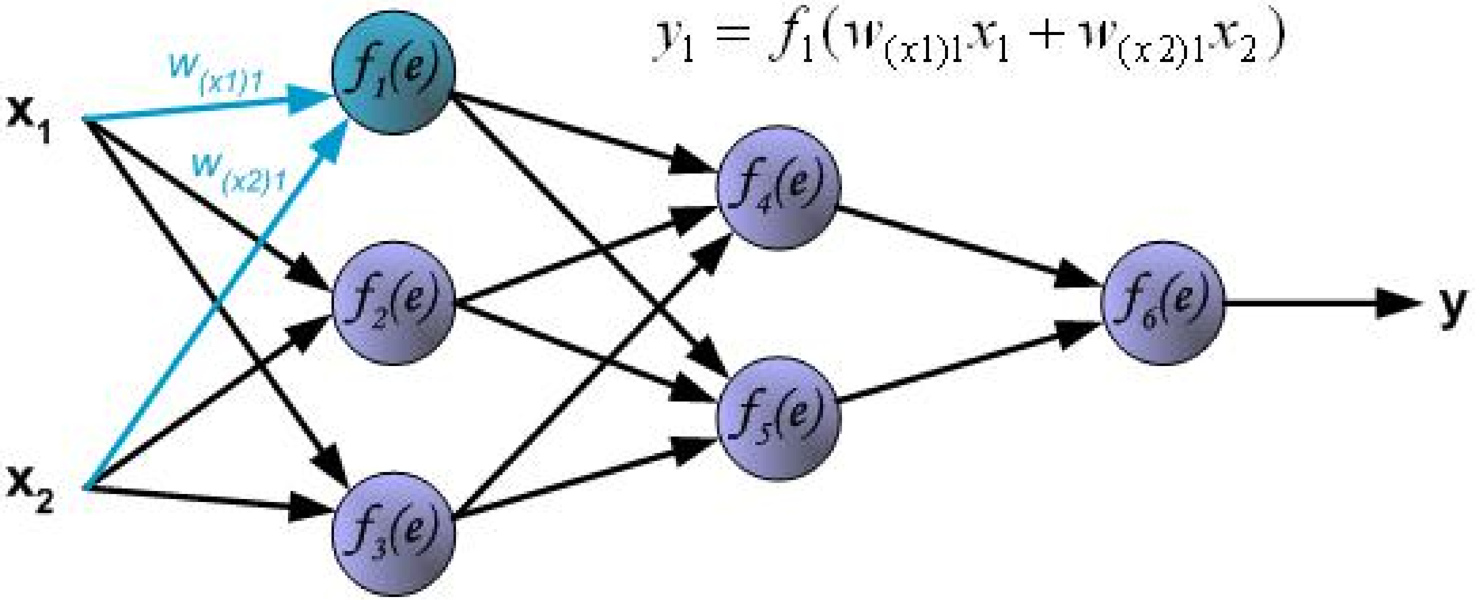
- 针对以下神经网络，y=f(e)，为输出结果，f为激活函数





- 第一步计算前向传播过程

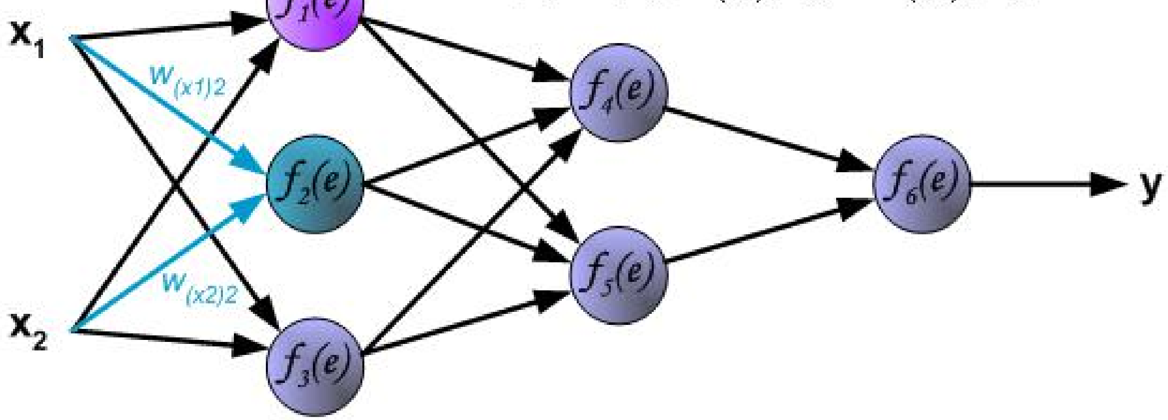
下面是**前向传播**过程：



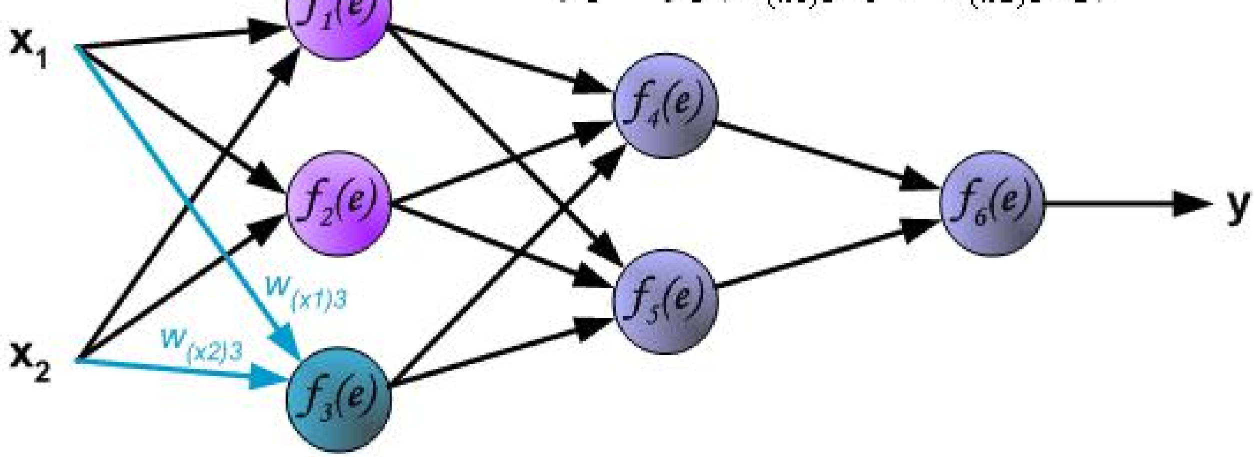
$$y_1 = f_1(w_{(x1)1} x_1 + w_{(x2)1} x_2)$$

- 计算第一层

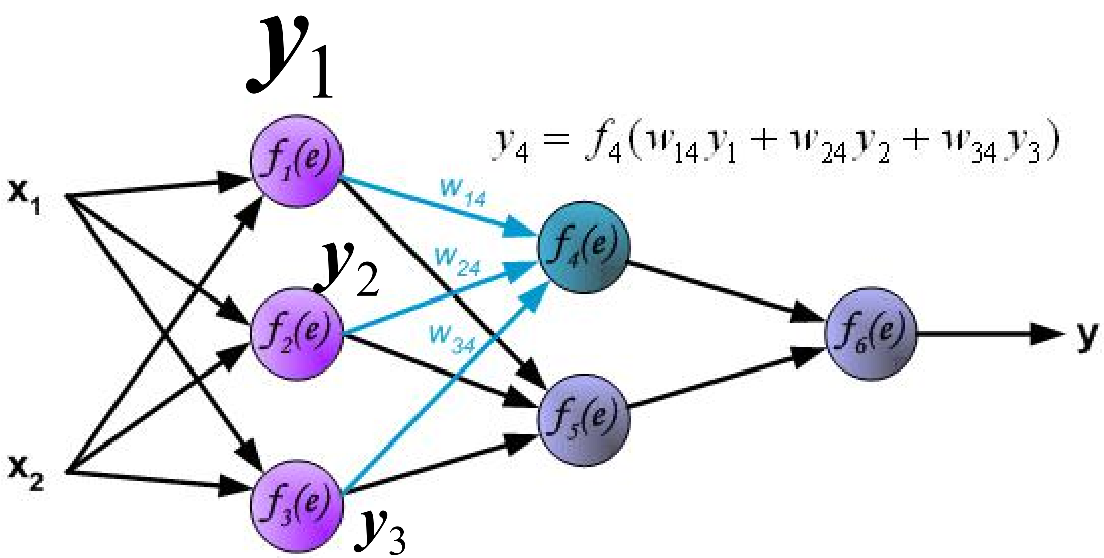
$$y_2 = f_2(w_{(x1)2}x_1 + w_{(x2)2}x_2)$$



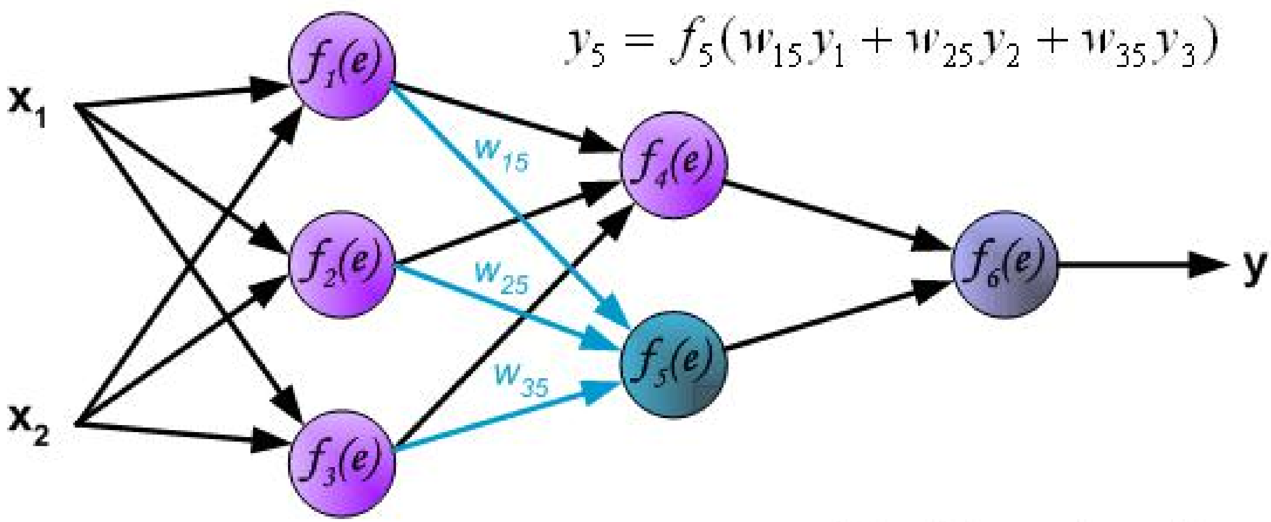
$$y_3 = f_3(w_{(x1)3}x_1 + w_{(x2)3}x_2)$$



计算第二层，第一层结果为输入

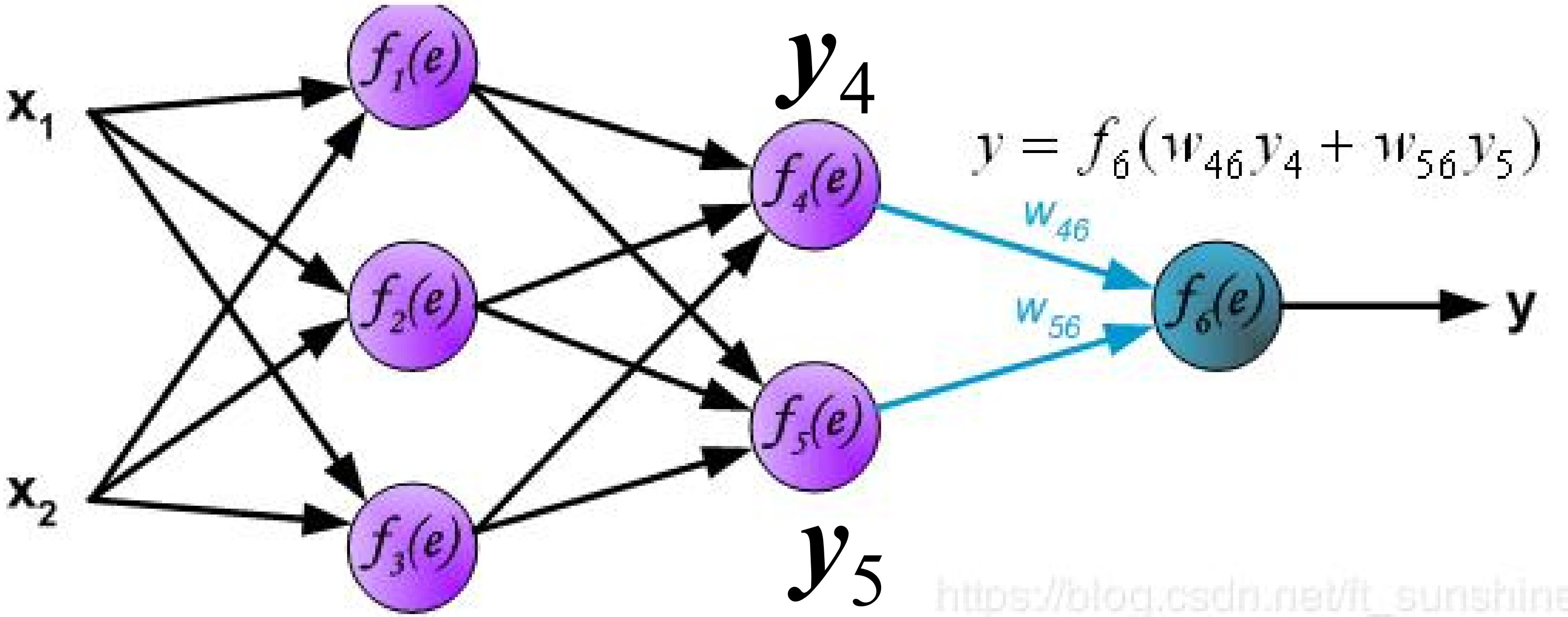


$$y_4 = f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3)$$



$$y_5 = f_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3)$$

- 第二层的结果为输入，输出正向结果



- 计算损失函数：正向结果y与目标结果z差的平方

$$E=(y-z)^2=(f_6(w_{46}y_4+w_{56}y_5)-z)^2$$

$$y_1=f_1(w_{(x1)1}x_1+w_{(x2)1}x_2)$$

$$y_2=f_2(w_{(x1)2}x_1+w_{(x2)2}x_2)$$

$$y_3=f_3(w_{(x1)3}x_1+w_{(x2)3}x_2)$$

$$y_4=f_4(w_{14}y_1+w_{24}y_2+w_{34}y_3)$$

$$y_5=f_5(w_{15}y_1+w_{25}y_2+w_{35}y_3)$$

$$f_i(x)=\frac{1}{1+e^{-x}},i=1,2,3,4,5,6$$

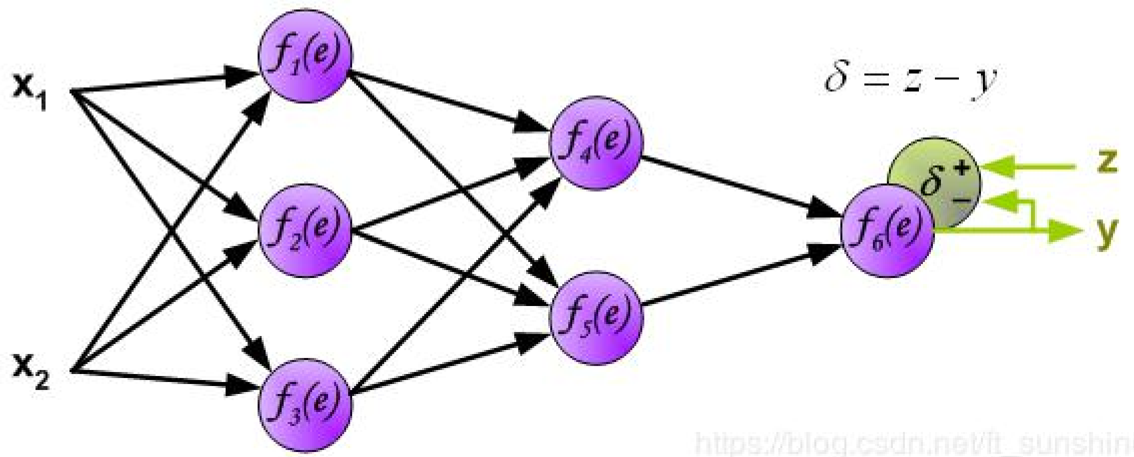
- 因此损失函数E为关于权重的复合函数

$$E(w,f_1,\cdots,f_6)$$

- 当E等于0时，说明权重合适，不需要更新
- 当E不等于0 时，结果与实际目标不符，需要调整权重，重新计算输出结果
- 更新权重方法，梯度下降法：

$$w_{ij}^{k+1}=w_{ij}^{k}-\eta\frac{\partial E}{\partial w_{ij}}$$

- 计算反向误差



记 $e = w_{46}y_4 + w_{56}y_5$

$$E = (z-y)^2 = (z - f_6(w_{46}y_4 + w_{56}y_5))^2 = (z - f_6(e))^2$$

$$\frac{\partial E}{\partial w_{46}} = -(z-y)\frac{\partial f_6(e)}{\partial e}y_4$$

$$\frac{\partial E}{\partial w_{56}} = -(z-y)\frac{\partial f_6(e)}{\partial e}y_5$$

- 第二层误差

$$E=(y-z)^2=(f_6(w_{46}y_4+w_{56}y_5)-z)^2$$

$$y_4=f_4(w_{14}y_1+w_{24}y_2+w_{34}y_3)$$
$$y_5=f_5(w_{15}y_1+w_{25}y_2+w_{35}y_3)$$

$$f_i(x)=\frac{1}{1+e^{-x}},i=1,2,3,4,5,6$$

$$\frac{\partial E}{\partial w_{14}}=-(z-y)\frac{\partial f_6(e)}{\partial e}w_{46}\frac{\partial f_4(e)}{\partial e}y_1$$
$$\frac{\partial E}{\partial w_{24}}=-(z-y)\frac{\partial f_6(e)}{\partial e}w_{46}\frac{\partial f_4(e)}{\partial e}y_2$$
$$\frac{\partial E}{\partial w_{34}}=-(z-y)\frac{\partial f_6(e)}{\partial e}w_{46}\frac{\partial f_4(e)}{\partial e}y_3$$

$$\frac{\partial E}{\partial w_{15}}=-(z-y)\frac{\partial f_6(e)}{\partial e}w_{56}\frac{\partial f_5(e)}{\partial e}y_1$$
$$\frac{\partial E}{\partial w_{25}}=-(z-y)\frac{\partial f_6(e)}{\partial e}w_{56}\frac{\partial f_5(e)}{\partial e}y_2$$
$$\frac{\partial E}{\partial w_{35}}=-(z-y)\frac{\partial f_6(e)}{\partial e}w_{56}\frac{\partial f_5(e)}{\partial e}y_3$$

- 第一层误差

$$E=(y-z)^2=(f_6(w_{46}y_4+w_{56}y_5)-z)^2$$

$$y_1=f_1(w_{(x1)1}x_1+w_{(x2)1}x_2)$$

$$y_2=f_2(w_{(x1)2}x_1+w_{(x2)2}x_2)$$

$$y_3=f_3(w_{(x1)3}x_1+w_{(x2)3}x_2)$$

$$y_4=f_4(w_{14}y_1+w_{24}y_2+w_{34}y_3)$$

$$y_5=f_5(w_{15}y_1+w_{25}y_2+w_{35}y_3)$$

$$\frac{\partial E}{\partial w_{(x1)1}}=-(z-y)\frac{\partial f_6(e)}{\partial e}(w_{46}\frac{\partial f_4(e)}{\partial e}w_{14}\frac{\partial f_1(e)}{\partial e}x_1+w_{56}\frac{\partial f_5(e)}{\partial e}w_{15}\frac{\partial f_1(e)}{\partial e}x_1)$$

$$\frac{\partial E}{\partial w_{(x2)1}}=-(z-y)\frac{\partial f_6(e)}{\partial e}(w_{46}\frac{\partial f_4(e)}{\partial e}w_{14}\frac{\partial f_1(e)}{\partial e}x_2+w_{56}\frac{\partial f_5(e)}{\partial e}w_{15}\frac{\partial f_1(e)}{\partial e}x_2)$$

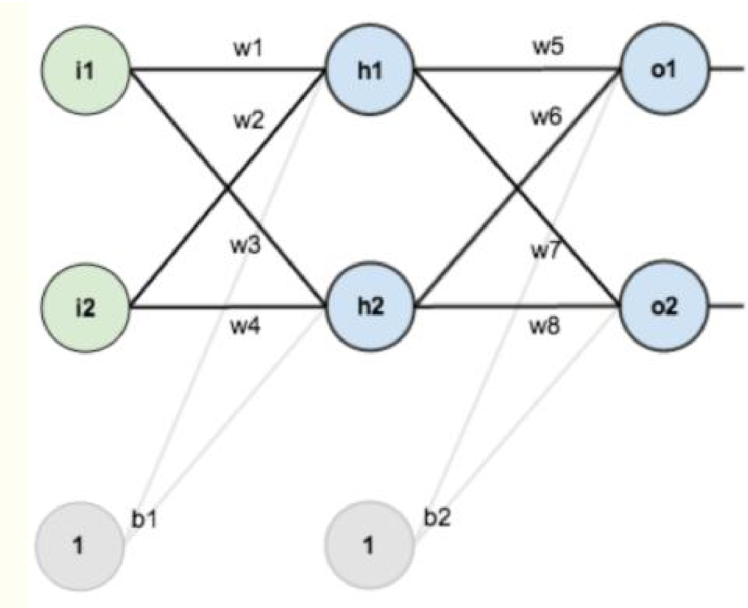
$$\frac{\partial E}{\partial w_{(x1)2}}=-(z-y)\frac{\partial f_6(e)}{\partial e}(w_{46}\frac{\partial f_4(e)}{\partial e}w_{24}\frac{\partial f_2(e)}{\partial e}x_1+w_{56}\frac{\partial f_5(e)}{\partial e}w_{25}\frac{\partial f_2(e)}{\partial e}x_1)$$

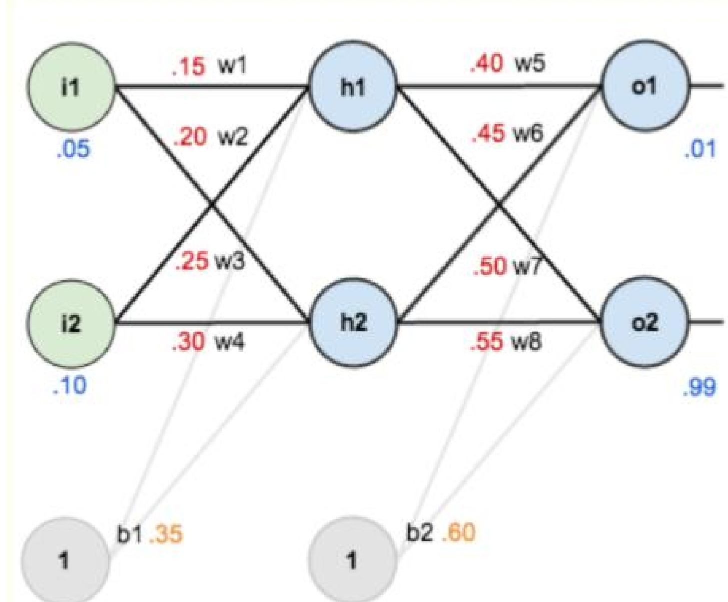
$$\frac{\partial E}{\partial w_{(x2)2}}=-(z-y)\frac{\partial f_6(e)}{\partial e}(w_{46}\frac{\partial f_4(e)}{\partial e}w_{24}\frac{\partial f_2(e)}{\partial e}x_2+w_{56}\frac{\partial f_5(e)}{\partial e}w_{25}\frac{\partial f_2(e)}{\partial e}x_2)$$

$$\frac{\partial E}{\partial w_{(x1)3}}=-(z-y)\frac{\partial f_6(e)}{\partial e}(w_{46}\frac{\partial f_4(e)}{\partial e}w_{34}\frac{\partial f_3(e)}{\partial e}x_1+w_{56}\frac{\partial f_5(e)}{\partial e}w_{35}\frac{\partial f_3(e)}{\partial e}x_1)$$

$$\frac{\partial E}{\partial w_{(x2)3}}=-(z-y)\frac{\partial f_6(e)}{\partial e}(w_{46}\frac{\partial f_4(e)}{\partial e}w_{34}\frac{\partial f_3(e)}{\partial e}x_2+w_{56}\frac{\partial f_5(e)}{\partial e}w_{35}\frac{\partial f_3(e)}{\partial e}x_2)$$

具体计算例子





其中，输入数据 i1=0.05，i2=0.10;

输出数据 o1=0.01,o2=0.99;

初始权重 w1=0.15,w2=0.20,w3=0.25,w4=0.30;

w5=0.40,w6=0.45,w7=0.50,w8=0.55

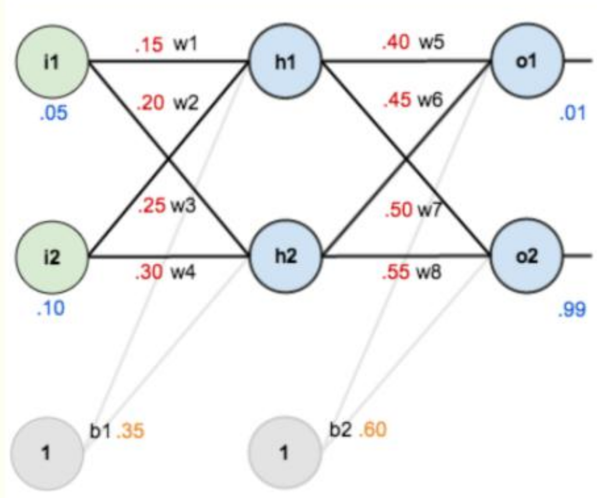
Step 1 前向传播

1.输入层---->隐含层:

计算神经元h1的输入加权和:

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

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



神经元h1的输出o1:(此处用到激活函数为sigmoid函数):

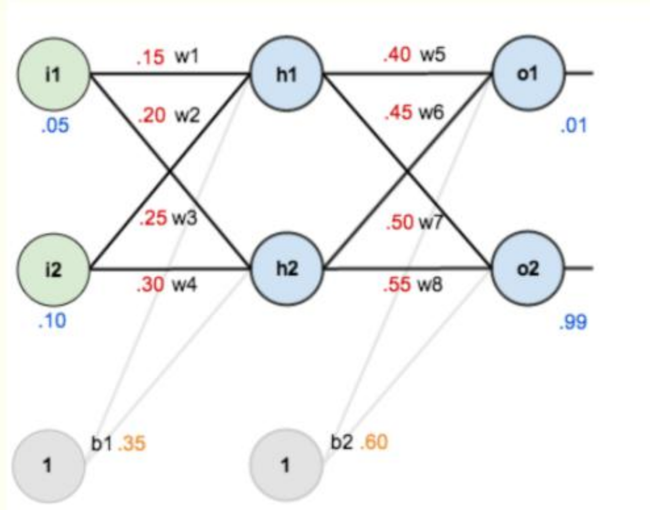
$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

同理，可计算出神经元h2的输出o2:

$$out_{h2} = 0.596884378$$

2.隐含层---->输出层:

计算输出层神经元o1和o2的值:



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

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

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

$$out_{o2} = 0.772928465$$

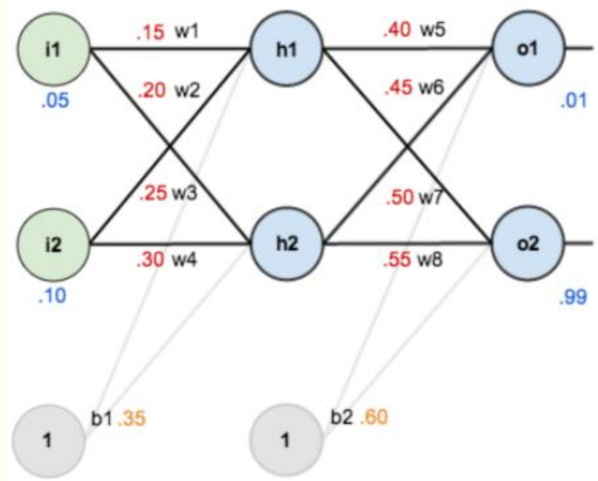
我们得到输出值为[0.75136079 , 0.772928465]，与实际值[0.01 , 0.99]相差还很远，

现在我们对误差进行反向传播，更新权值，重新计算输出。

Step 2 反向传播

1.计算总误差

总误差：(square error)



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

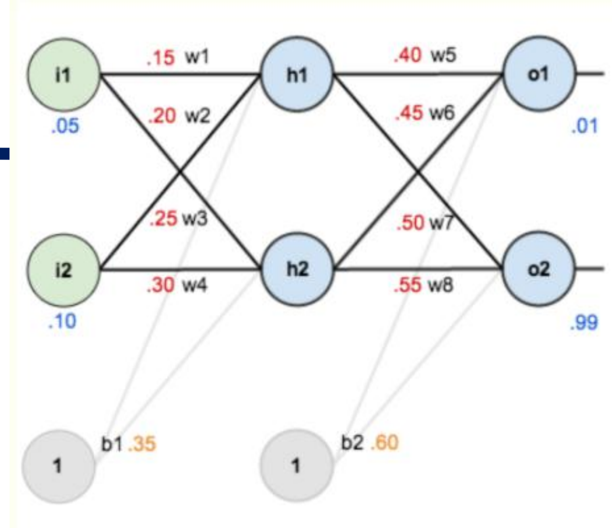
但是有两个输出，所以分别计算o1和o2的误差，总误差为两者之和：

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

$$E_{o2} = 0.023560026$$

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

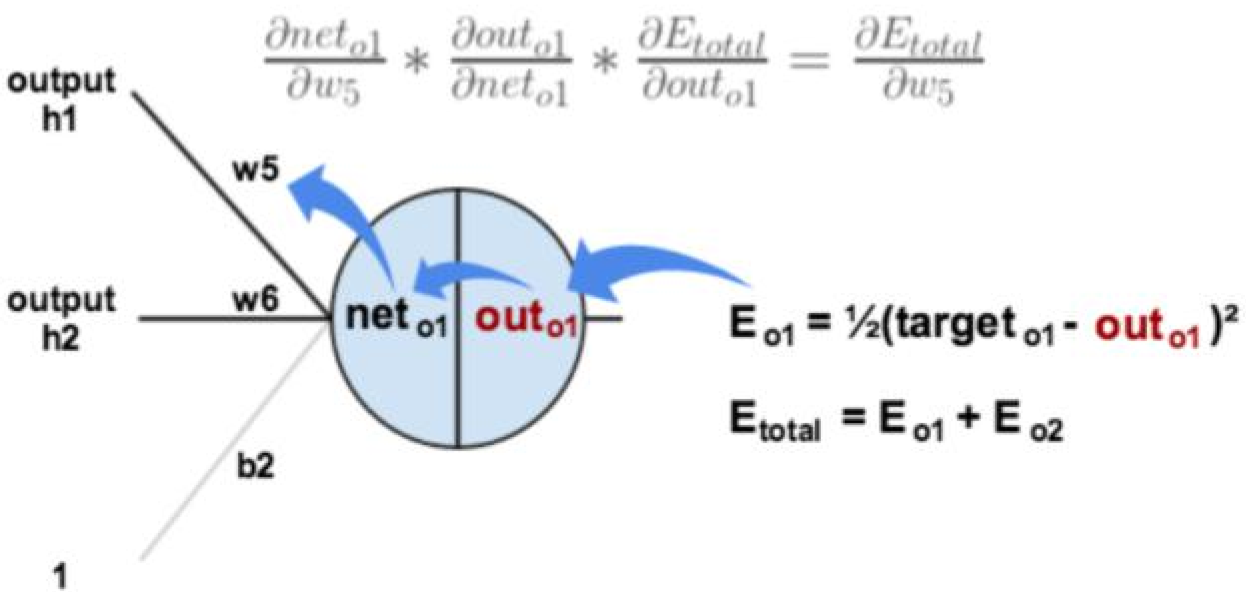
2.隐含层---->输出层的权值更新：



以权重参数w5为例，

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

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



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

计算 $\frac{\partial E_{total}}{\partial out_{o1}}$：

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

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

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

计算 $\frac{\partial out_{o1}}{\partial net_{o1}}$：

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

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

计算 $\frac{\partial net_{o1}}{\partial w_5}$：

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

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

最后三者相乘:

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

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

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

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

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

因此，整体误差E(total)对w5的偏导公式可以写成:

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

最后我们来更新w5的值：

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

同理，可更新w6,w7,w8：

$$w_6^+ = 0.408666186$$

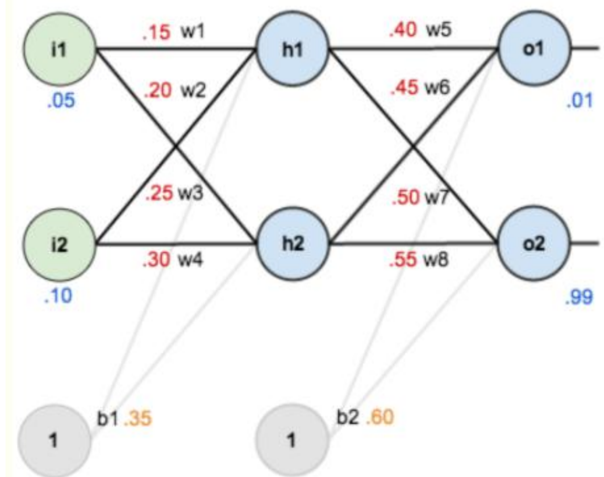
$$w_7^+ = 0.511301270$$

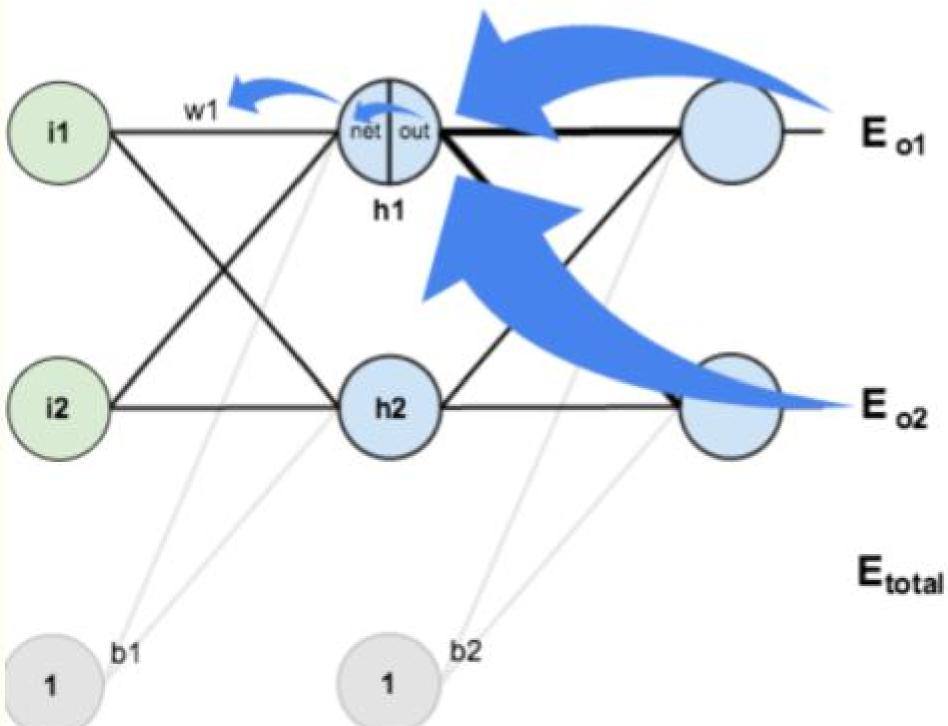
$$w_8^+ = 0.561370121$$

3.隐含层---->隐含层的权值更新：

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

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





$E_{total} = E_{o1} + E_{o2}$

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

$$\downarrow$$

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

计算 $\frac{\partial E_{total}}{\partial out_{h1}}$:

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

先计算 $\frac{\partial E_{o1}}{\partial out_{h1}}$:

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

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

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

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

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

同理，计算出：

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

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

再计算 $\frac{\partial out_{h1}}{\partial net_{h1}}$ ：

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

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1}) = 0.59326999(1 - 0.59326999) = 0.2413007$$

再计算 $\frac{\partial net_{h1}}{\partial w_1}$：

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

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

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

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

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

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

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

最后，更新w1的权值：

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

同理，额可更新w2,w3,w4的权值：

$$w_2^+ = 0.19956143$$

$$w_3^+ = 0.24975114$$

$$w_4^+ = 0.29950229$$

最后我们再把更新的权值重新计算，不停地迭代，

总误差E(total)由0.298371109下降至0.291027924。

常见的几种神经网络

图像的卷积

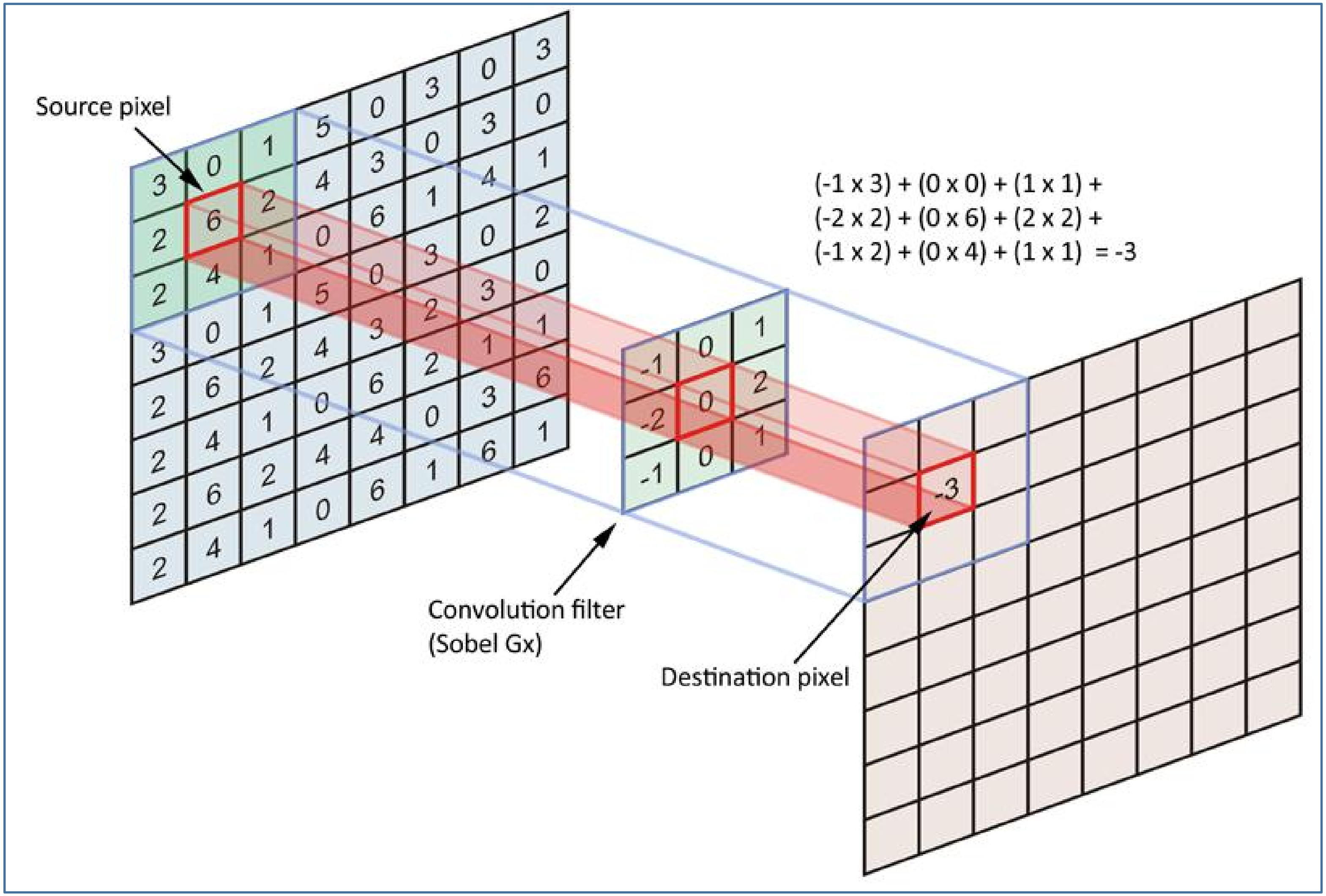
- 卷积
  - 提取图片的特征
- 卷积核
  - 通常为较小尺寸的矩阵，3 * 3， 5 * 5
- 卷积运算
  - 使用卷积核自上而下、自左向右在图像上滑动，将卷积核矩阵的各个元素与它在图像上覆盖的对应位置元素相乘，求和

$$\begin{bmatrix} \mathbf{I}_{x-1,y-1} & \mathbf{I}_{x-1,y} & \mathbf{I}_{x-1,y+1} \\ \mathbf{I}_{x,y-1} & \mathbf{I}_{x,y} & \mathbf{I}_{x,y+1} \\ \mathbf{I}_{x+1,y-1} & \mathbf{I}_{x+1,y} & \mathbf{I}_{x+1,y+1} \end{bmatrix}, \text{卷积核} = \begin{bmatrix} \mathbf{-1} & \mathbf{-2} & \mathbf{-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{2} & \mathbf{1} \end{bmatrix}$$

(x, y)处的卷积结果为：

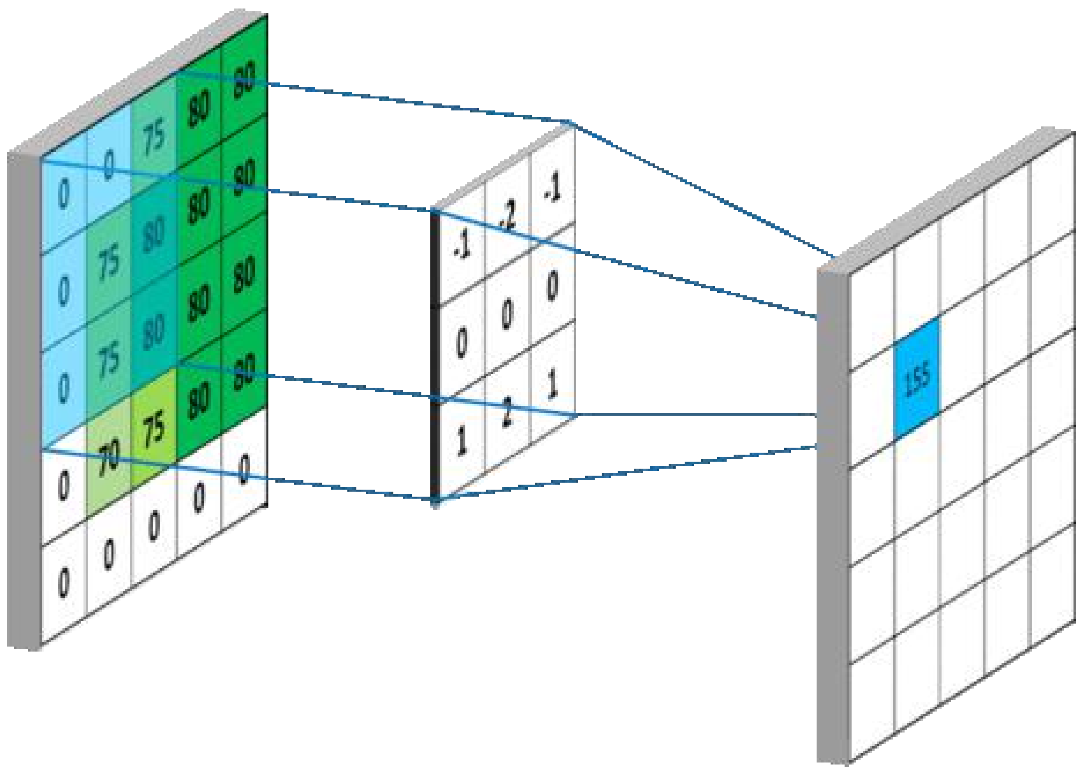
$$-\mathbf{I}_{x-1,y-1} - 2\mathbf{I}_{x-1,y} - \mathbf{I}_{x-1,y+1} + \mathbf{I}_{x+1,y-1} + 2\mathbf{I}_{x+1,y} + \mathbf{I}_{x+1,y+1}$$

卷积运算例子



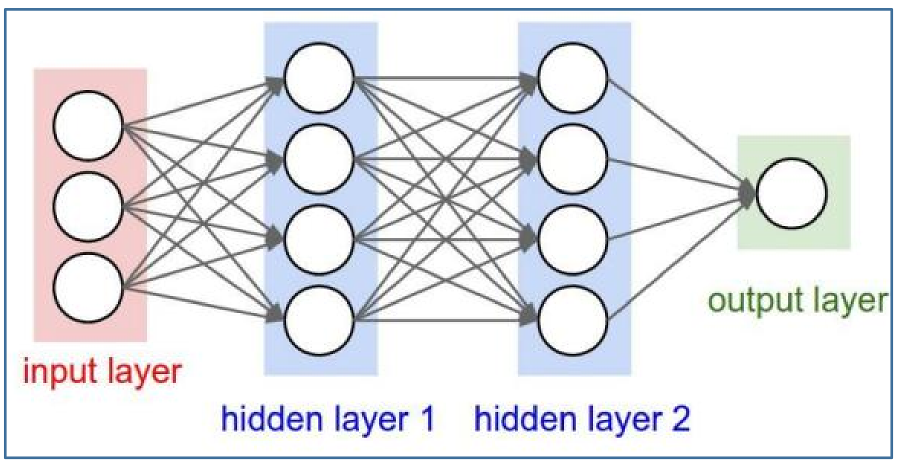
Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

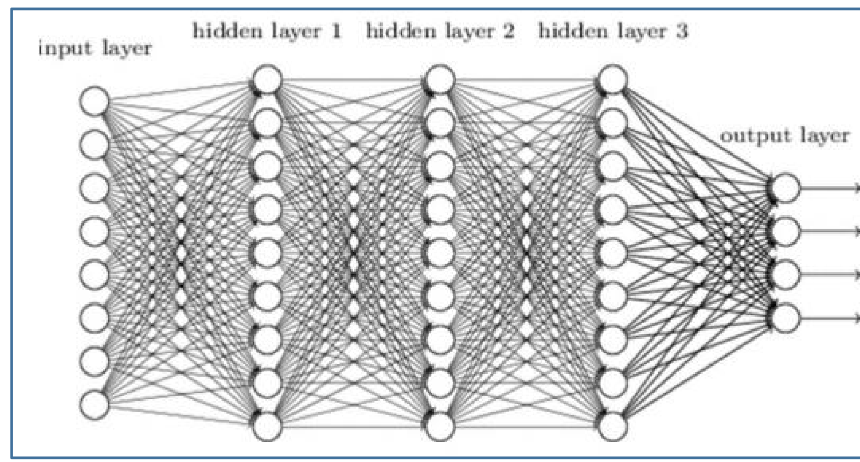
不同的卷积核



神经网络与深度神经网络

- 神经网络：模拟特征与目标之间的真实关系函数的方法
- 多层神经网络：更多的参数意味着其模拟的函数可以更加的复杂表示能力大幅度增强





卷积神经网络：Convolutional Neural Networks，CNN

- 卷积滤波器和神经网络两个思想结合起来
- 若干底层特征组成上一层特征，最终通过多个层级的组合做出分类
- 是一种多层神经网络，擅长处理图像相关的机器学习问题

- 卷积神经网络结构：

  ①数据输入层/ Input layer：数据预算理，去均值，归一化，PCA降维

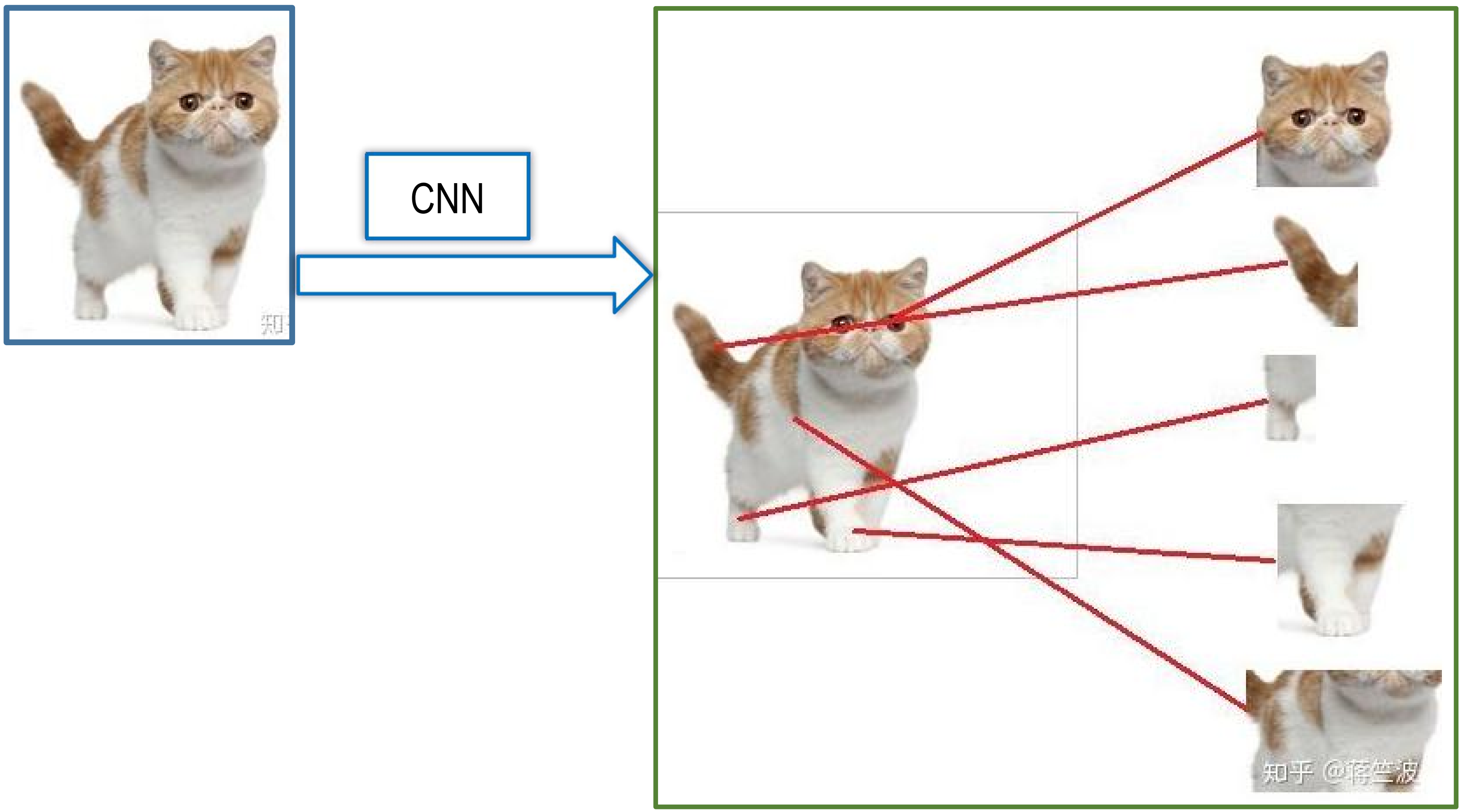
  ②卷积计算层/ CONV layer：最重要的一个层次

  ③ReLU激励层 / ReLU layer：把卷积层输出结果做非线性映射

  ④池化层 / Pooling layer：用于压缩数据和参数的量，减小过拟合

  ⑤全连接层 / FC layer：两层之间所有神经元都有权重连接

卷积神经网络示例

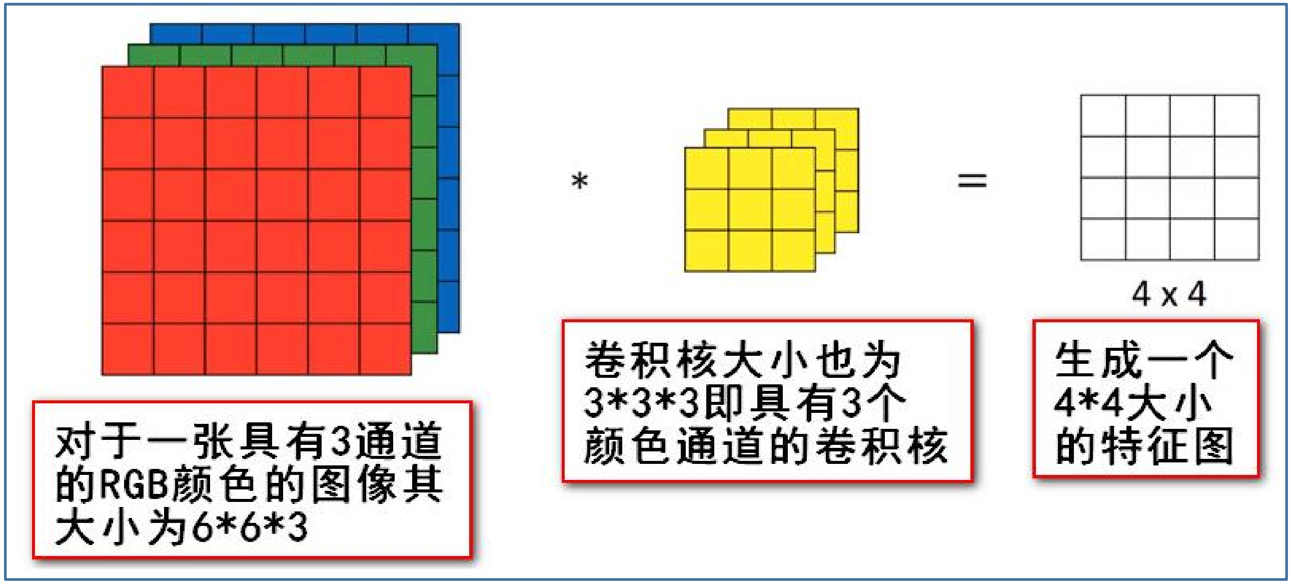


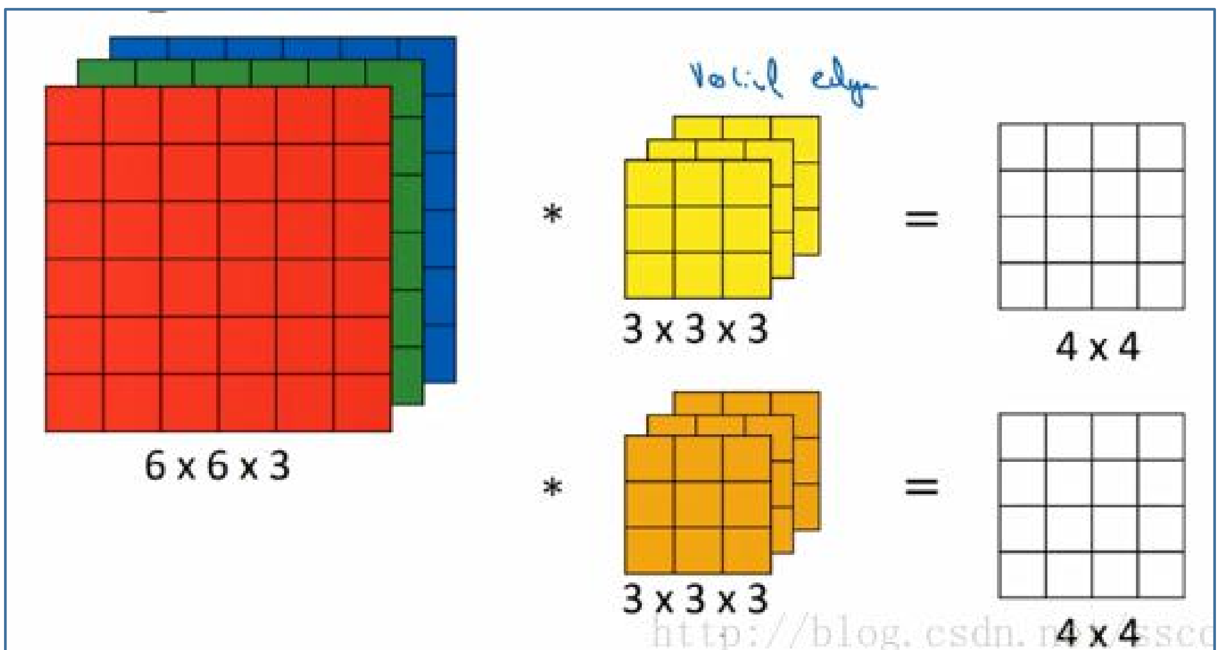
卷积层：提取图像特征

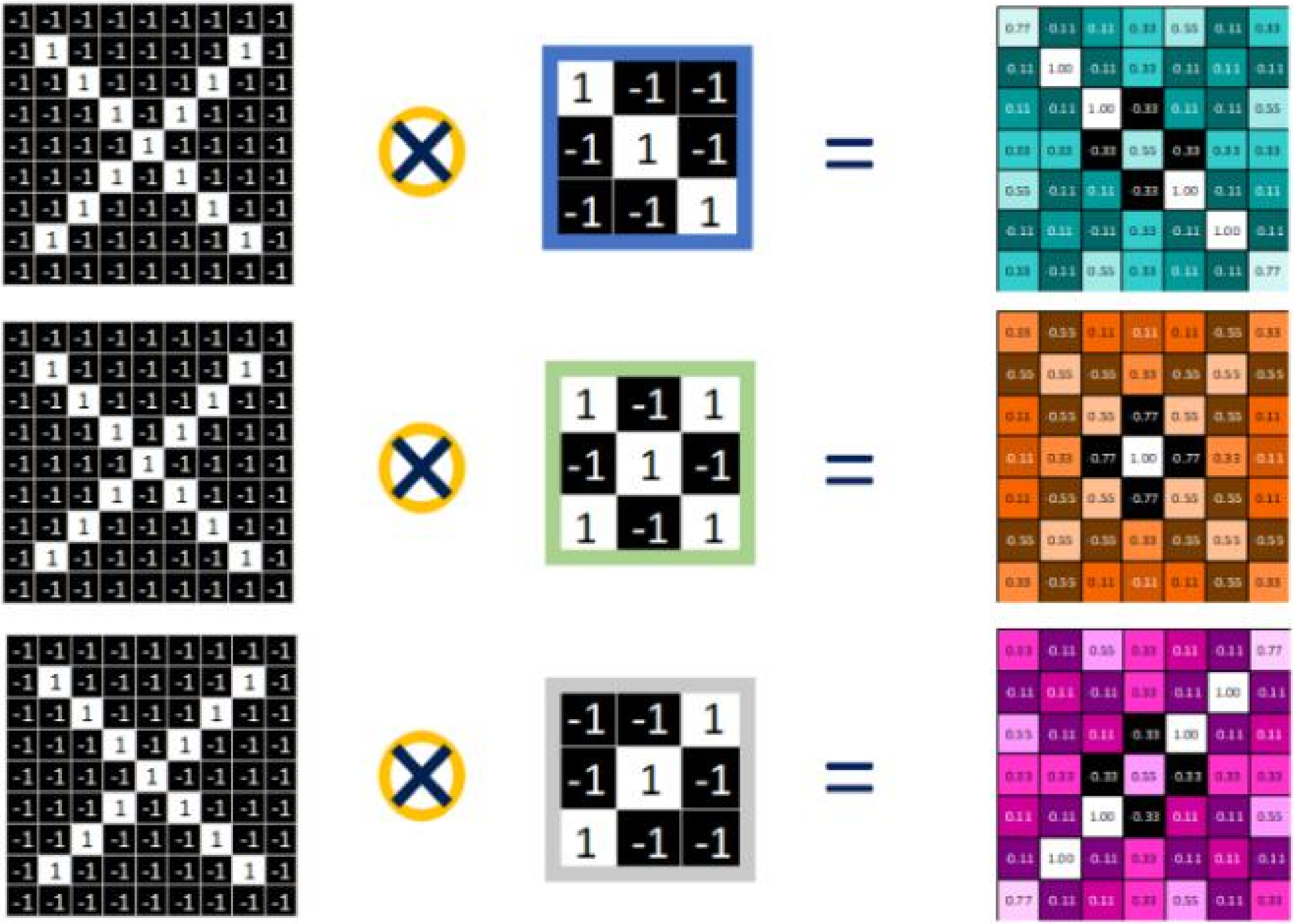
- 卷积
  - 设计特定的卷积核，与图像做卷积，提取图片的特征

- 卷积核
  - 通常为较小尺寸的矩阵，3 * 3，5 * 5

卷积核的颜色通道与输入图像颜色通道一致



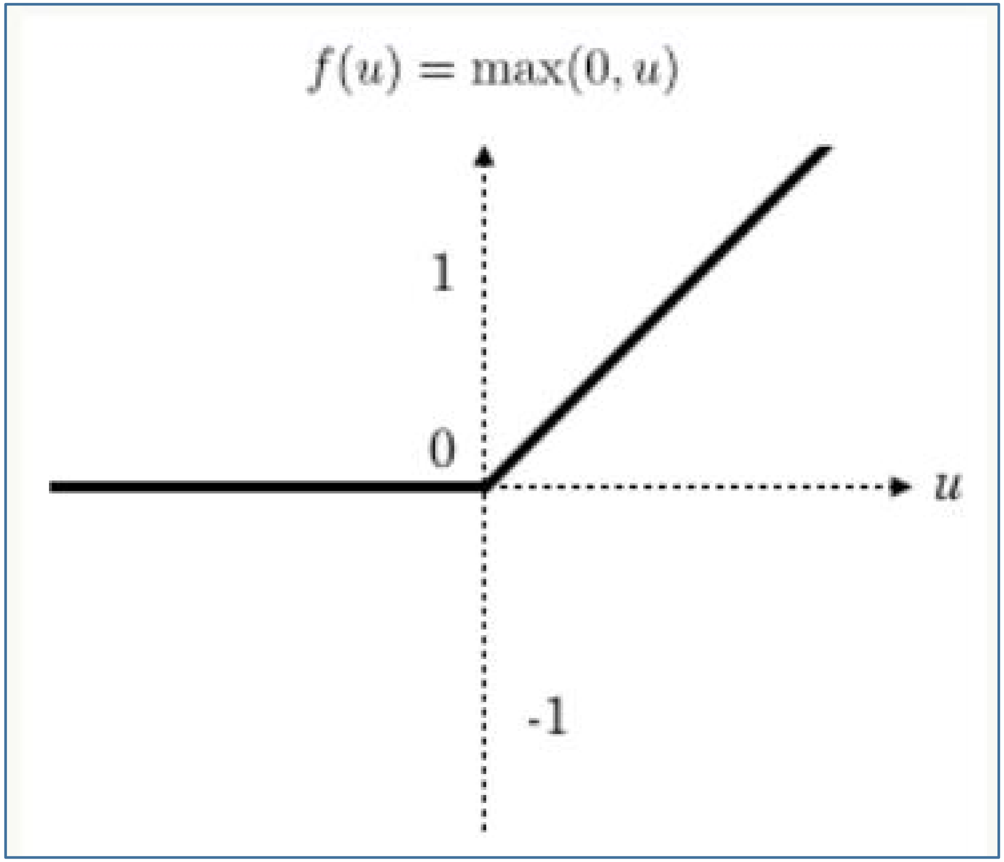



1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1

ReLU激励层：非线性激活

- ReLU函数（Rectified Linear Units）

$$f(x)=\max\{0,x\}$$



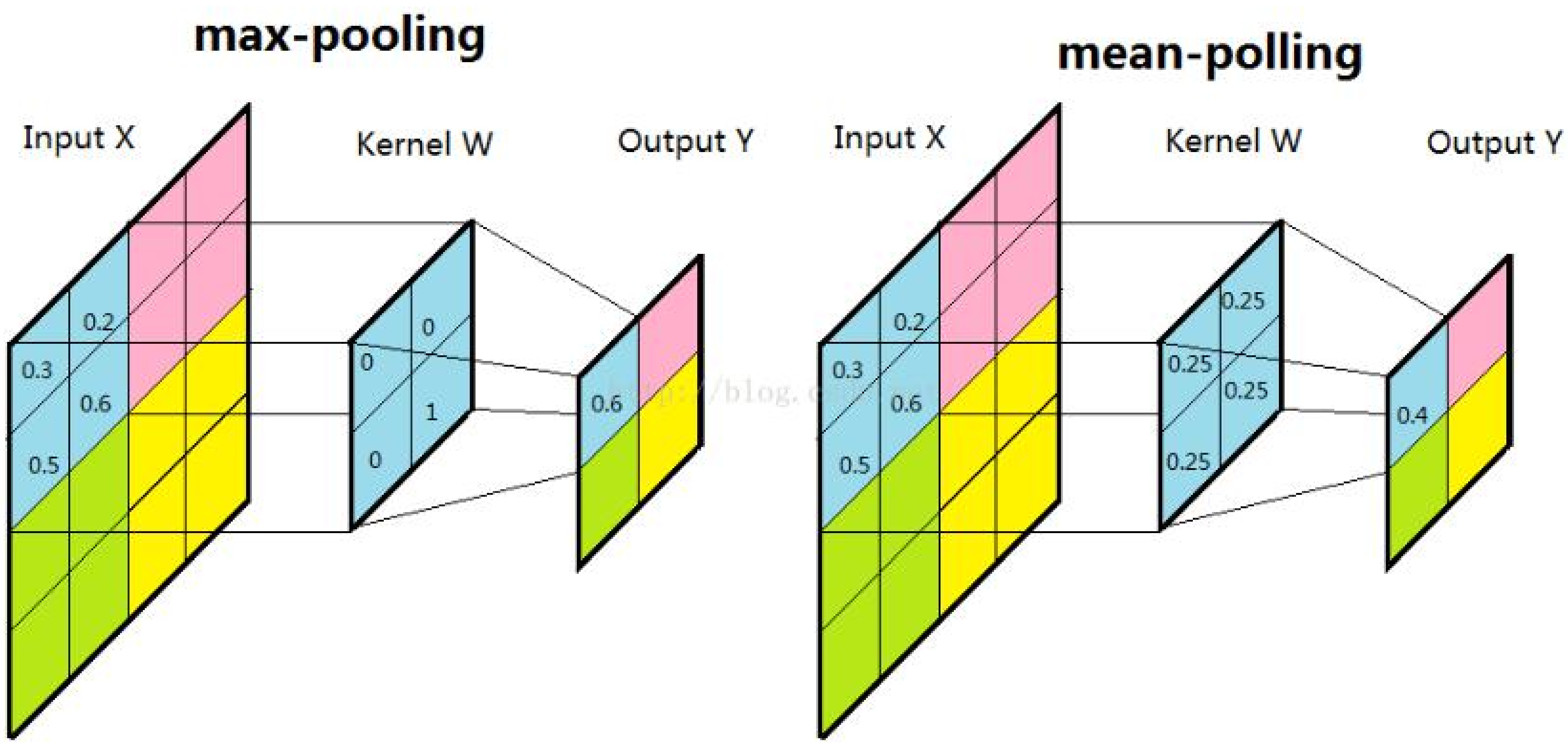
卷积后产生的特征图中的值，
越靠近1表示与该特征越关联
越靠近-1表示越不关联，
进行特征提取时，
为了使得数据更少，
操作更方便，就直接舍弃掉
那些不相关联的数据。

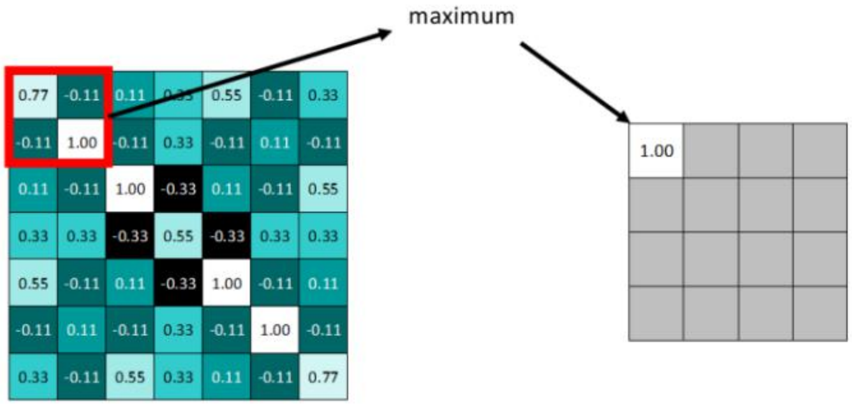
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

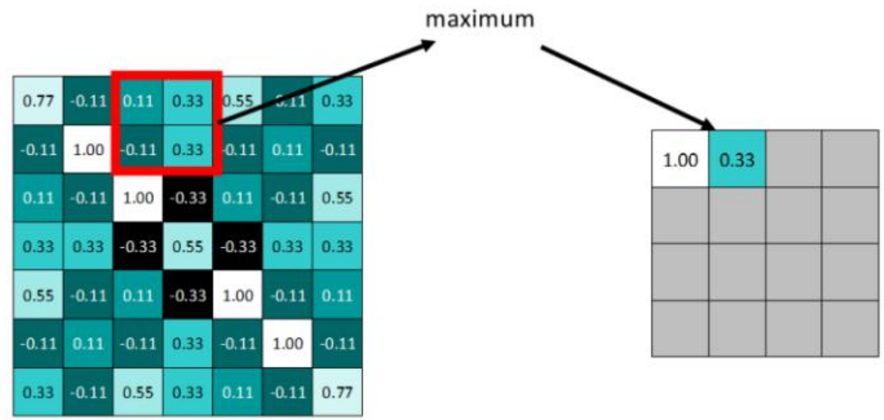
0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

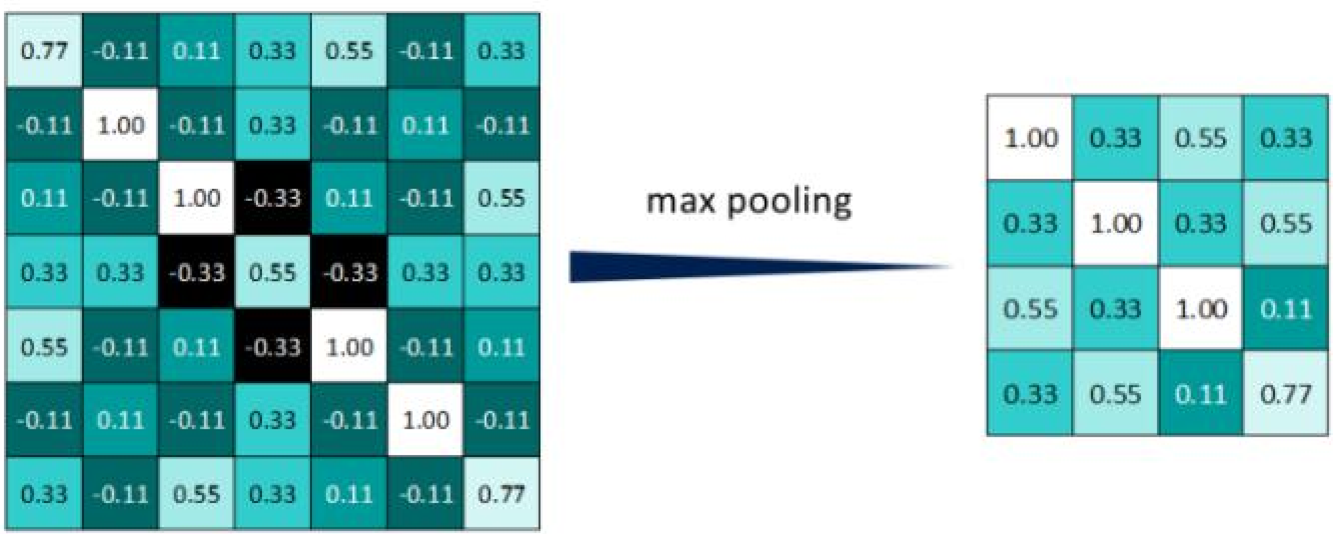
池化层：下采样降维

- 卷积操作后，得到了有着不同值的feature map，尽管数据量比原图少了很多，但还是过于庞大，因此使用接下来的池化操作减少数据量。
- 最大池化：某一个区域用最大值代替
- 平均池化：某一个区域用平均值代替



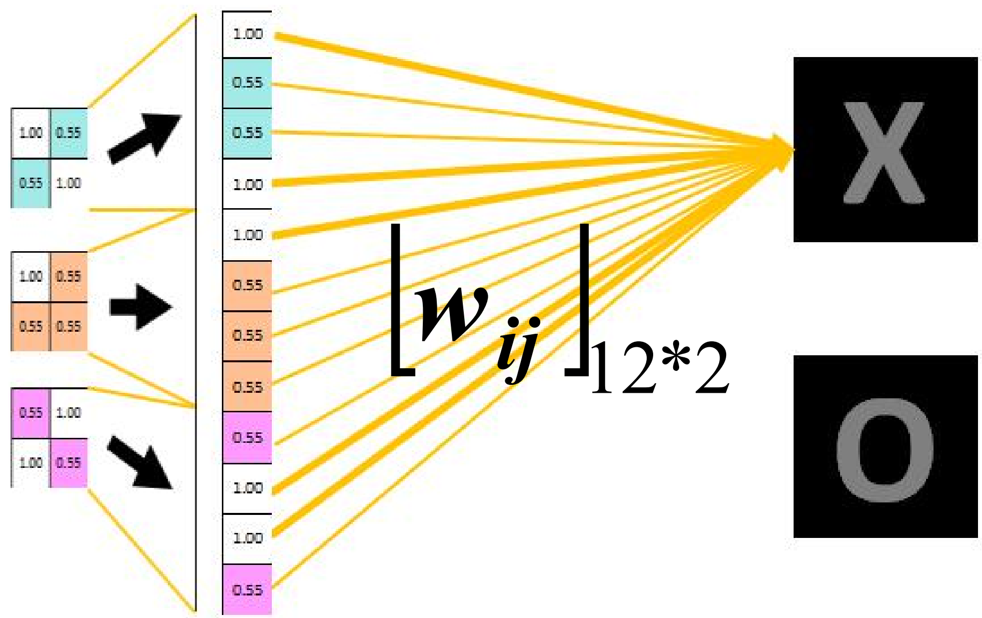

maximum
1.00


maximum
1.00 0.33

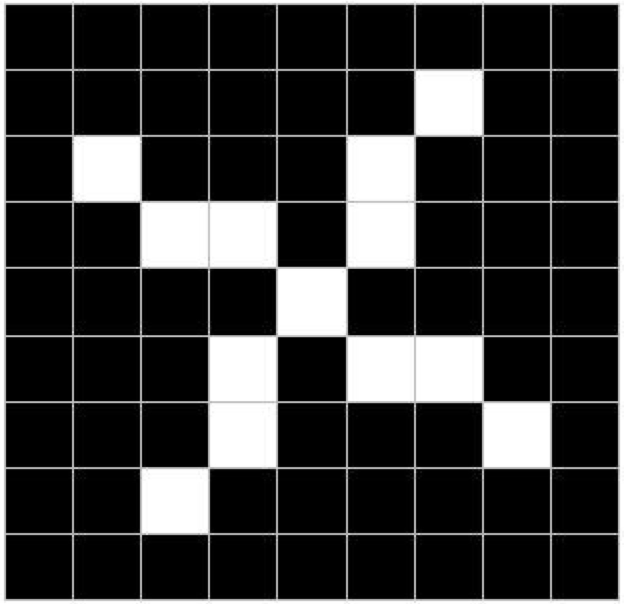

0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
max pooling
1.00 0.33 0.55 0.33
0.33 1.00 0.33 0.55
0.55 0.33 1.00 0.11
0.33 0.55 0.11 0.77

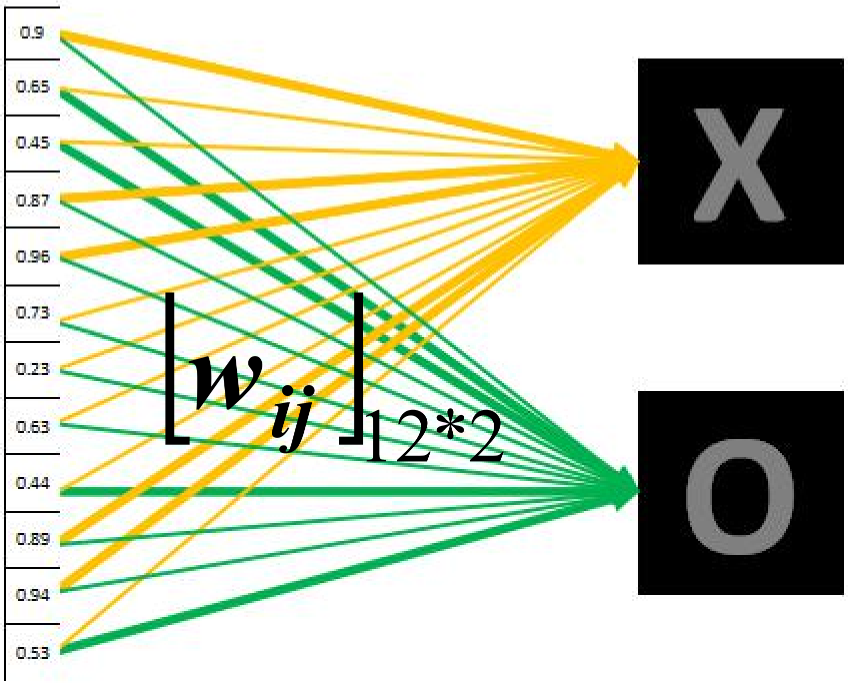
全连接层：

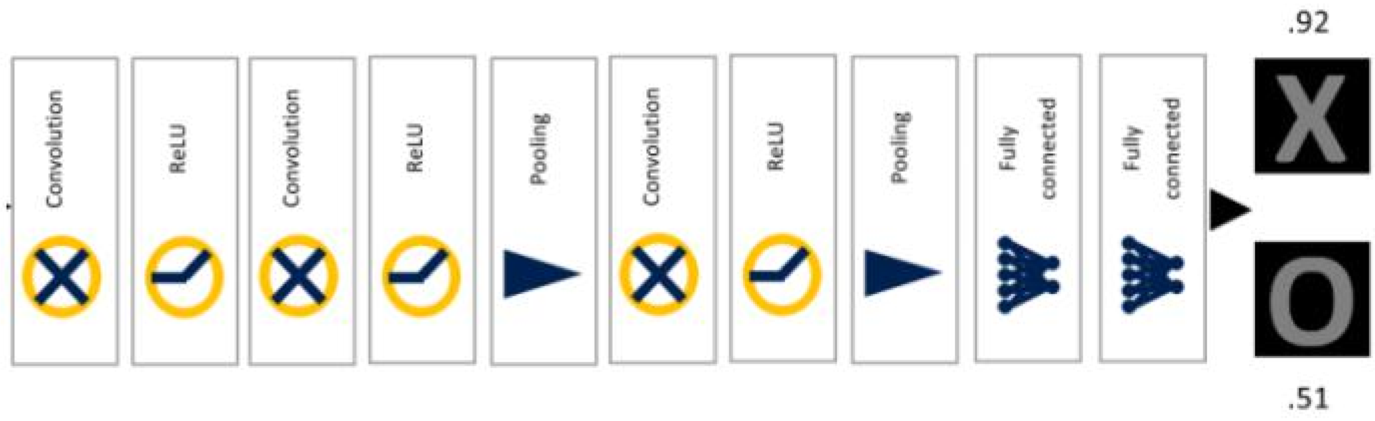
- 以上过程可以重复使用多层，最将终图片压缩为2*2大小
- 全连接：将图片展开对分类目标一一对应，最终得到的数字为概率



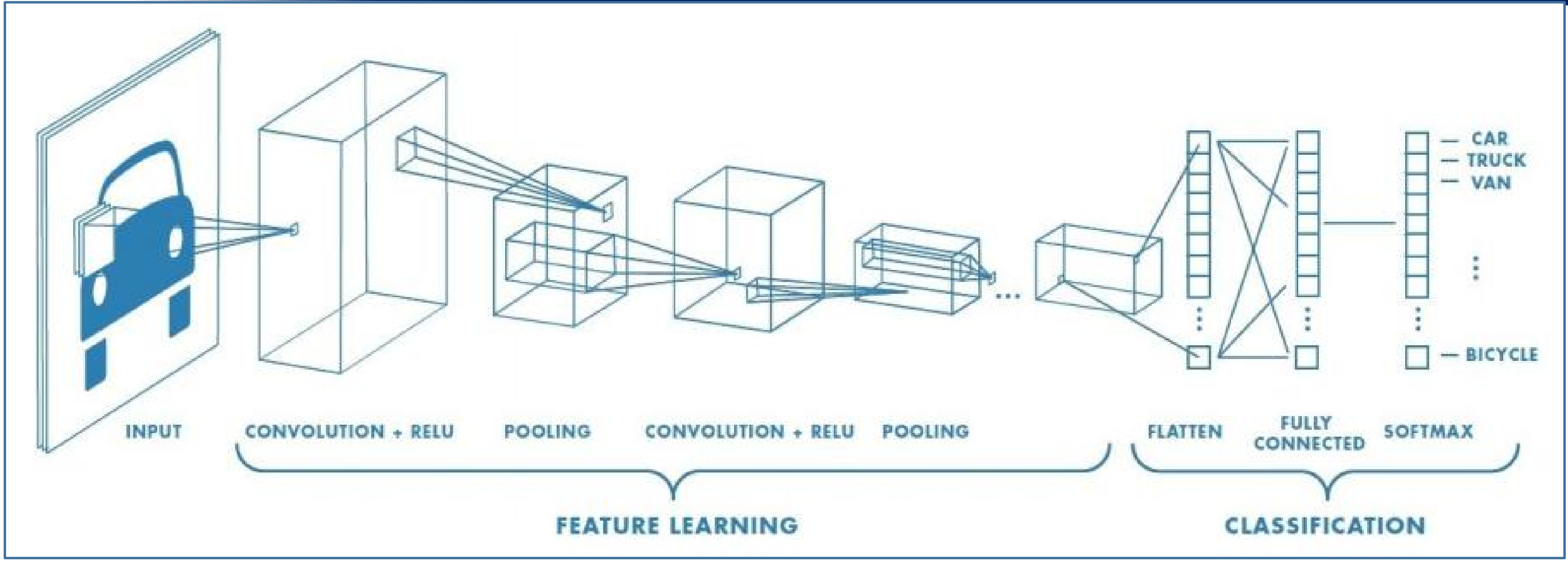
对于不太精确的样本进行归类：

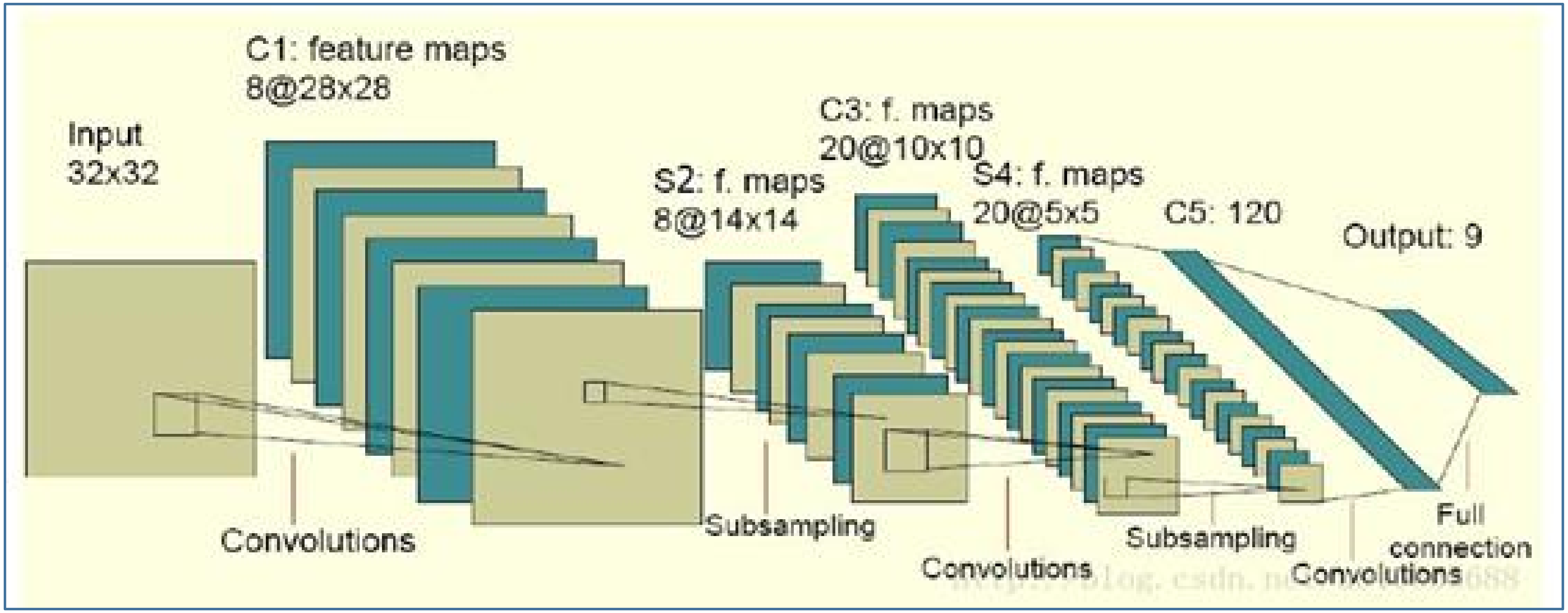


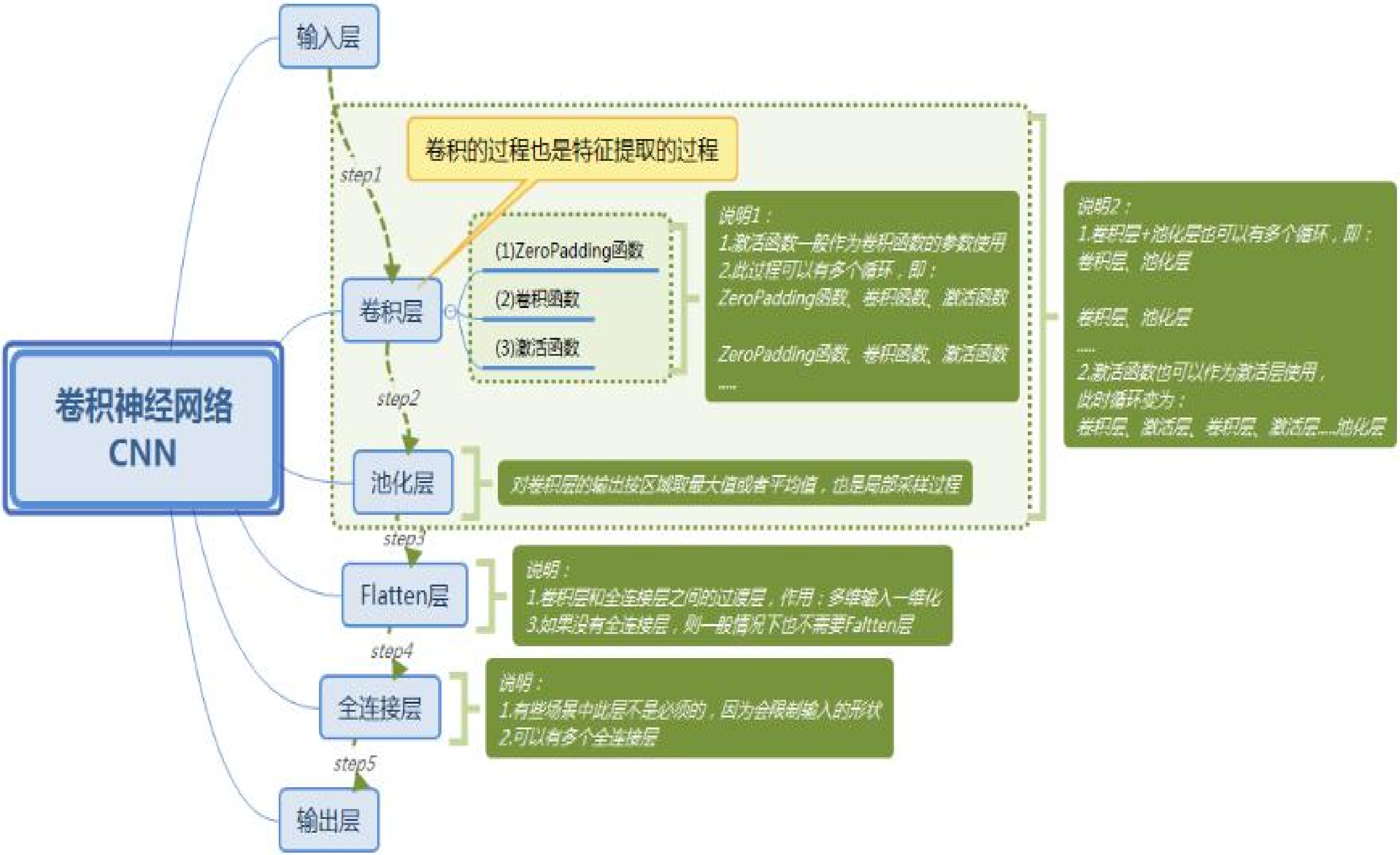




完整CNN框架


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION


C1: feature maps
8@28x28
Input
32x32
C3: f. maps
20@10x10
S2: f. maps
8@14x14
S4: f. maps
20@5x5
C5: 120
Output: 9
Convolutions
Subsampling
Convolutions
Subsampling
Full
connection
Convolutions


卷积神经网络
CNN
输入层
step1
卷积的过程也是特征提取的过程
卷积层
(1)ZeroPadding函数
(2)卷积函数
(3)激活函数
说明1：
1.激活函数一般作为卷积函数的参数使用
2.此过程可以有多个循环，即：
ZeroPadding函数、卷积函数、激活函数
ZeroPadding函数、卷积函数、激活函数
.....
说明2：
1.卷积层+池化层也可以有多个循环，即：
卷积层、池化层
卷积层、池化层
.....
2.激活函数也可以作为激活层使用，
此时循环变为：
卷积层、激活层、卷积层、激活层....池化层
step2
池化层
对卷积层的输出按区域取最大值或者平均值，也是局部采样过程
step3
Flatten层
说明：
1.卷积层和全连接层之间的过渡层，作用：多维输入一维化
3.如果没有全连接层，则一般情况下也不需要Faltten层
step4
全连接层
说明：
1.有些场景中此层不是必须的，因为会限制输入的形状
2.可以有多个全连接层
step5
输出层

卷积神经网络优缺点

- 优点
  - 共享卷积核，对高维数据处理无压力
  - 无需手动选取特征，训练好权重，分类效果好
- 缺点
  - 需要调参，需要大样本量，训练最好要GPU
  - 物理含义不明确（也就说，我们并不知道每个卷积层到底提取到的是什么特征，而且神经网络本身就是一种难以解释的“黑箱模型”：无法给出理论上严格的解释)

循环神经网络：RNN

- 经典的神经网络：如CNN,

  所有的输出都是独立的，对于数据具有依赖性的，效果不太理想

  解决了计算机的视觉问题，让机器具备视觉上识别的能力
- 缺陷：

  单靠机器的视觉能力，并不能实现自主的智能，还有其它能力也很重要
- 例如，人类的分析能力

  人类可以根据一个故事的开头猜到一个故事的结尾；

  可以根据对方说的话，揣测他背后的目的；

  智者往往处理事情有理有据，层次分明，

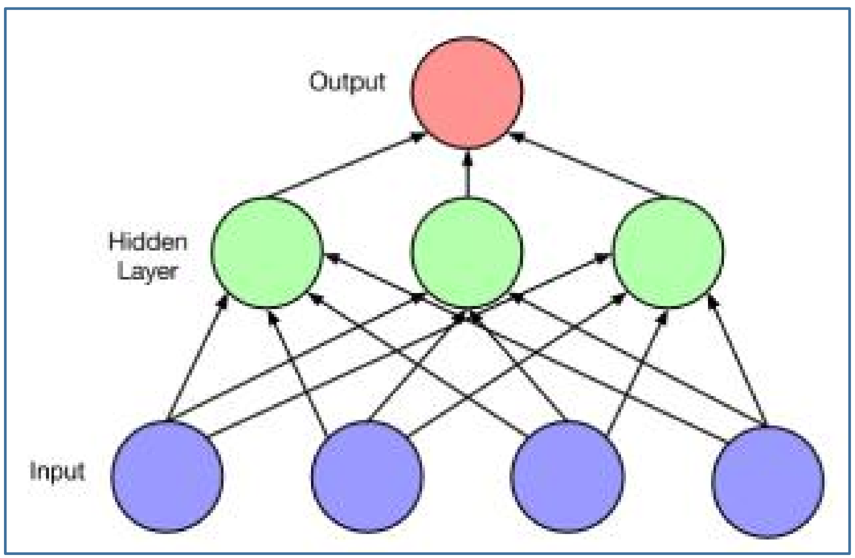
  我们期待计算机也有这样的能力。

所以学者们设计了神奇的**循环神经网络。**

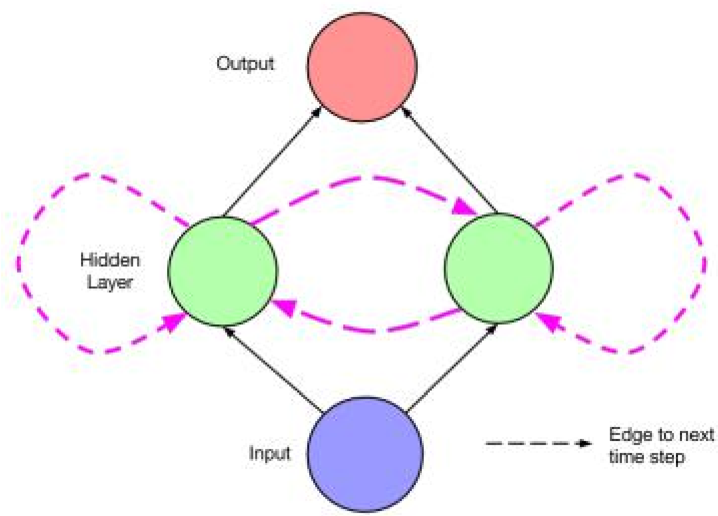
- 循环神经网络：

  是一类以序列（sequence）数据为输入，在序列的演进方向进行递归（recursion）且所有节点按链式连接的递归神经网络

- 目的：使机器具备分辨因果的能力，具有记忆功能
- 应用：机器翻译系统，语音识别



CNN



RNN

简单RNN的结构

- 简单RNN：输入层，隐藏层、输出层
- 如下面左图所示，右图为对应展开图
- 输出层与隐藏层计算方法

x_t ：输入向量
h_t ：输出向量
s_t ：隐藏层向量
U, V, A：权重矩阵

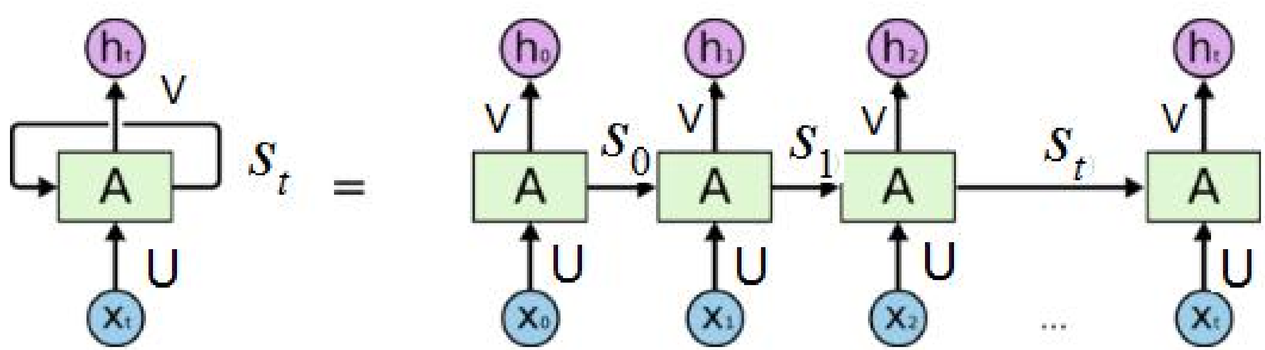
循环层： $\boldsymbol{s_t} = \boldsymbol{f}(\boldsymbol{U}\boldsymbol{x_t} + \boldsymbol{W}\boldsymbol{s_{t-1}})$

输出层： $\boldsymbol{h_t} = \boldsymbol{g}(\boldsymbol{V}\boldsymbol{s_t}),$

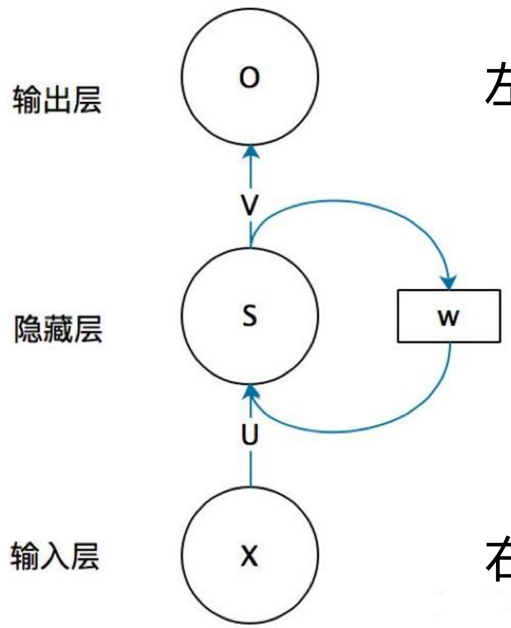
$\boldsymbol{f}, \boldsymbol{g}$为激活函数

输出值h_t受前面的输入值影响x_t

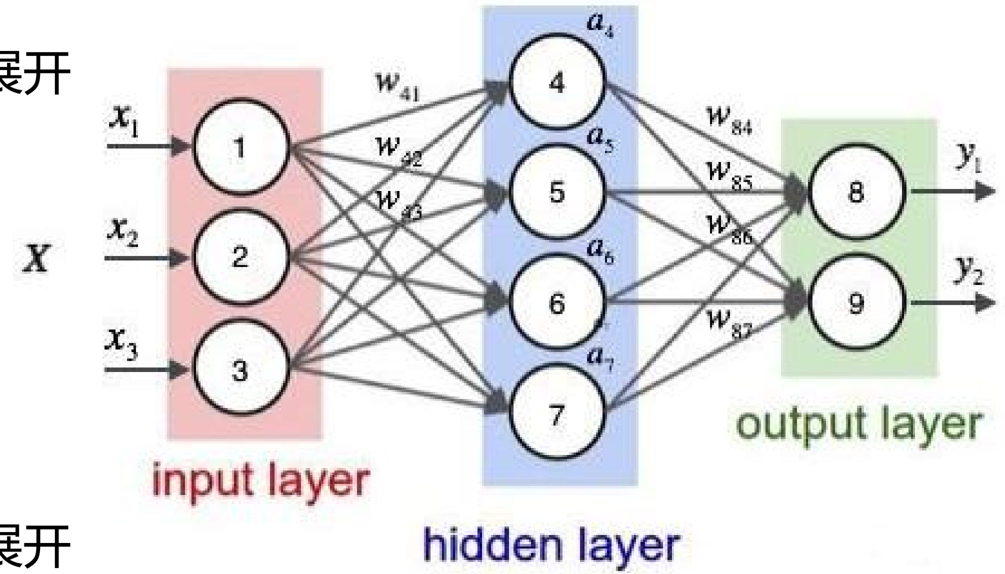
$$h_t = g(Vs_t) = g(Vf(Ux_t + Ws_{t-1}))$$
$$= g(Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Wf(Ux_{t-3} + \ldots))))$$



RNN结构展开

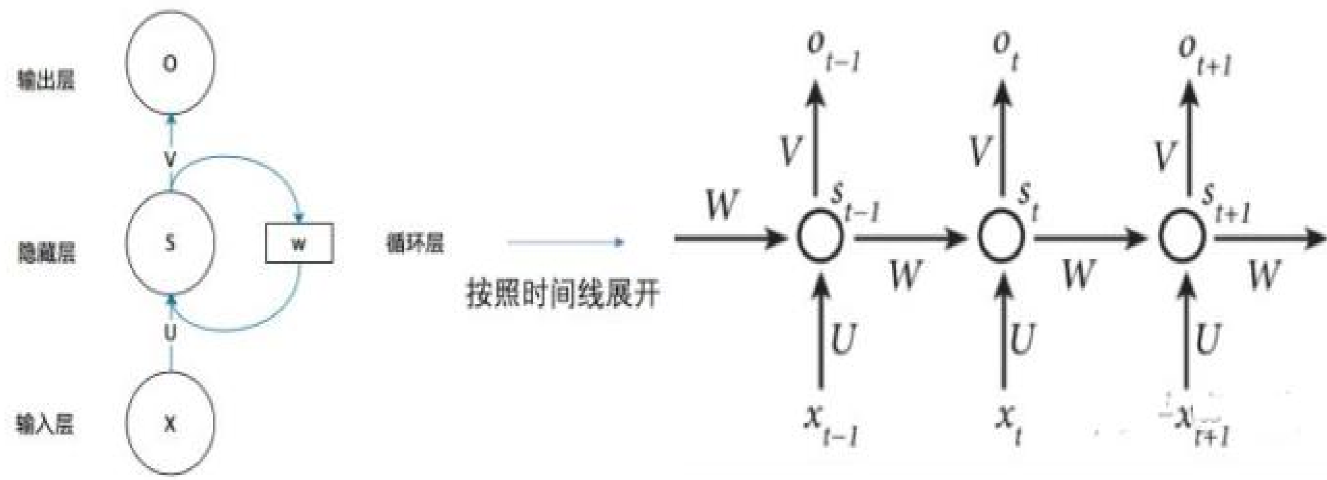


左部展开



右部展开

$$O_t = g(V \cdot S_t)$$
$$S_t = f(U \cdot X_t + W \cdot S_{t-1})$$



RNN结构例子

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}2\\2\end{bmatrix}$ ……

所有权重的值都为1且没有偏差

当我们输入第一个序列，【1,1】

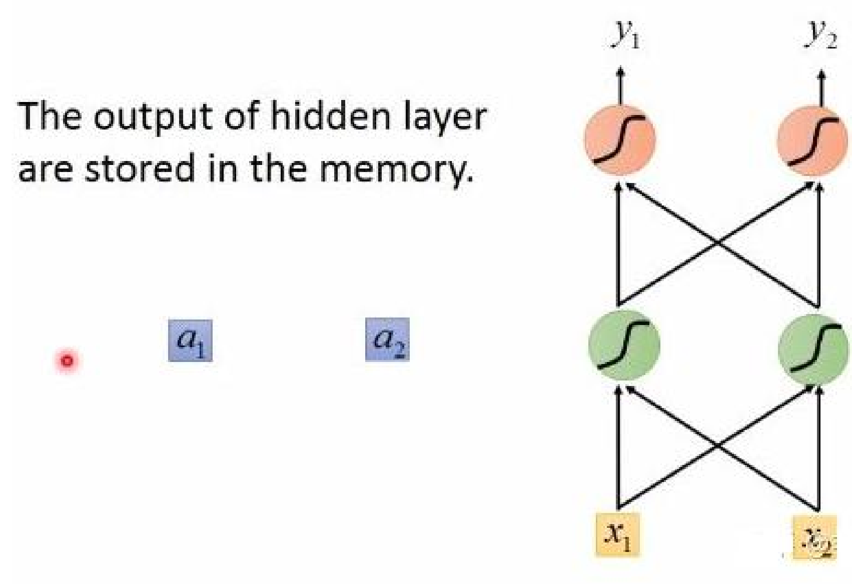
$$S_t = f\left(U \cdot X_t + W \cdot S_{t-1}\right)$$

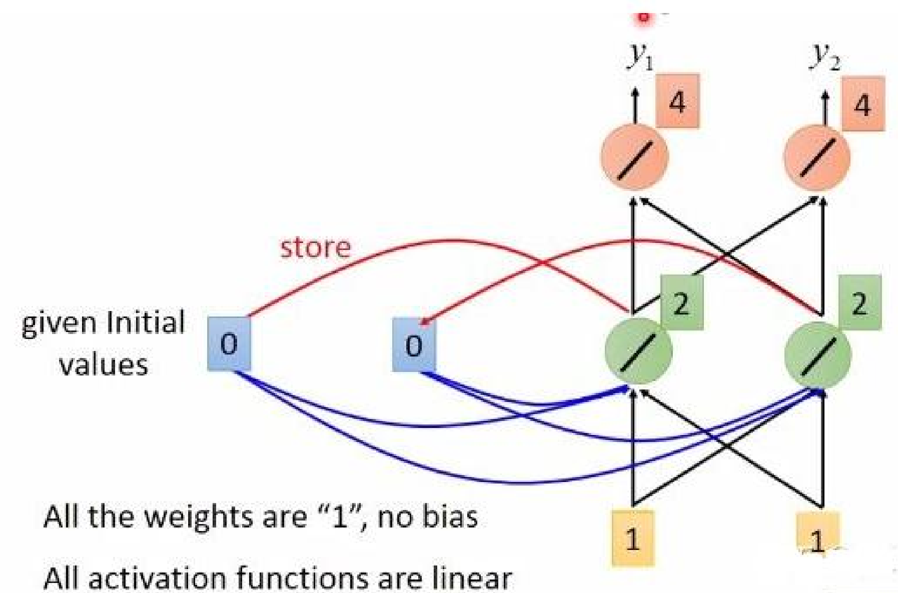
$$1 * 1 + 1 * 1 + 1 * 0 + 1 * 0 = 2$$

$$O_t = g\left(V \cdot S_t\right)$$

$$2 * 1 + 2 * 1 = 4$$

得到输出向量【4,4】





RNN结构例子

输入下一个向量【1,1】

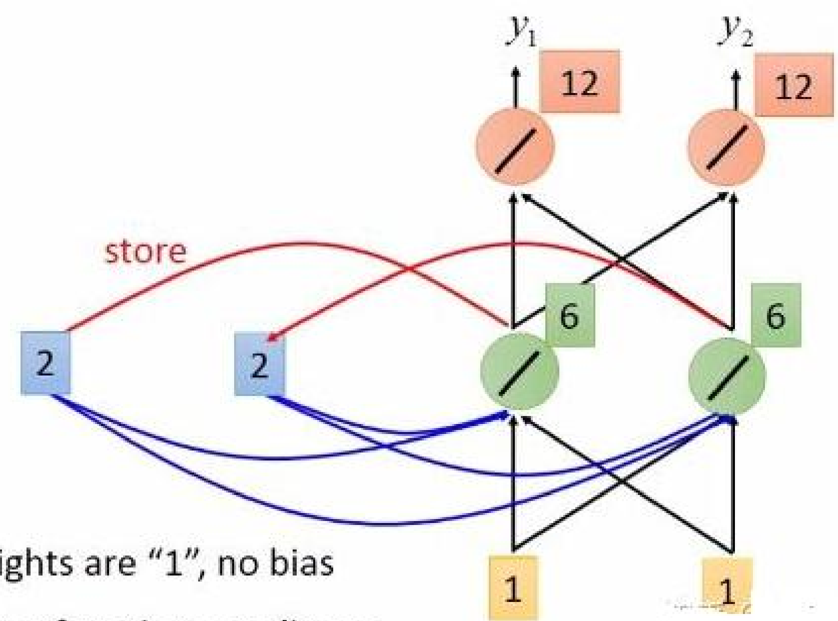
$$S_t = f(U \cdot X_t + W \cdot S_{t-1})$$

$$1*1+1*1+1*2+1*2=6$$

$$O_t = g(V \cdot S_t)$$

$$6*1+6*1=12$$

最终得到输出向量【12,12】



All the weights are "1", no bias

All activation functions are linear

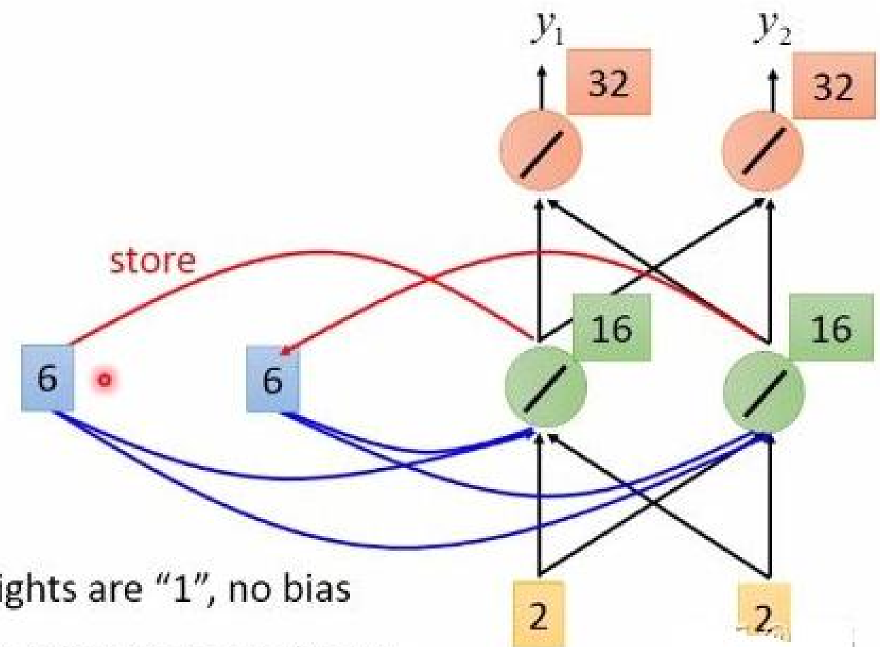
RNN结构例子

a_1, a_2 的值变成了6，

第三个向量【2,2】

得到输出向量【32,32】

output sequence: $\begin{bmatrix}4\\4\end{bmatrix}\begin{bmatrix}12\\12\end{bmatrix}\begin{bmatrix}32\\32\end{bmatrix}$



All the weights are "1", no bias

All activation functions are linear

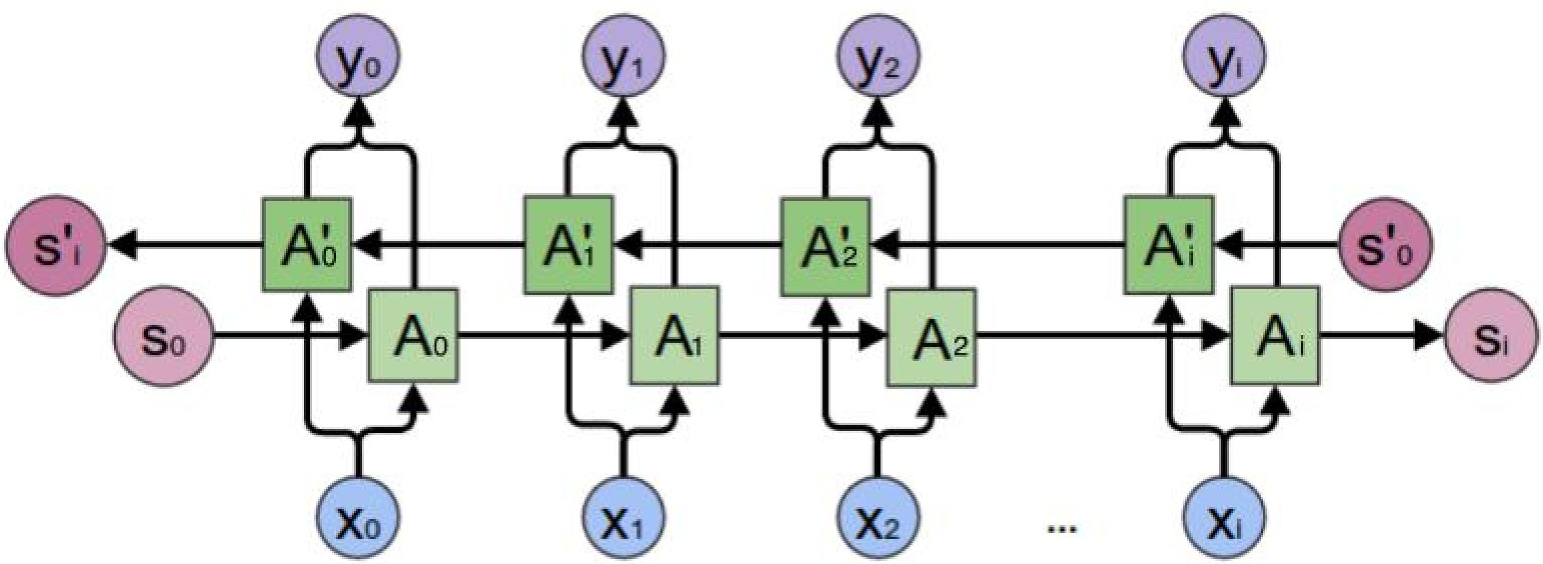
双向循环神经网络

- 隐藏层包含：正向计算与反向计算
- 计算方法为：

$$y_2 = g(VA_2 + V'A_2')$$

$$A_2 = f(WA_1 + Ux_2)$$
$$A_2' = f(W'A_3' + U'x_2)$$

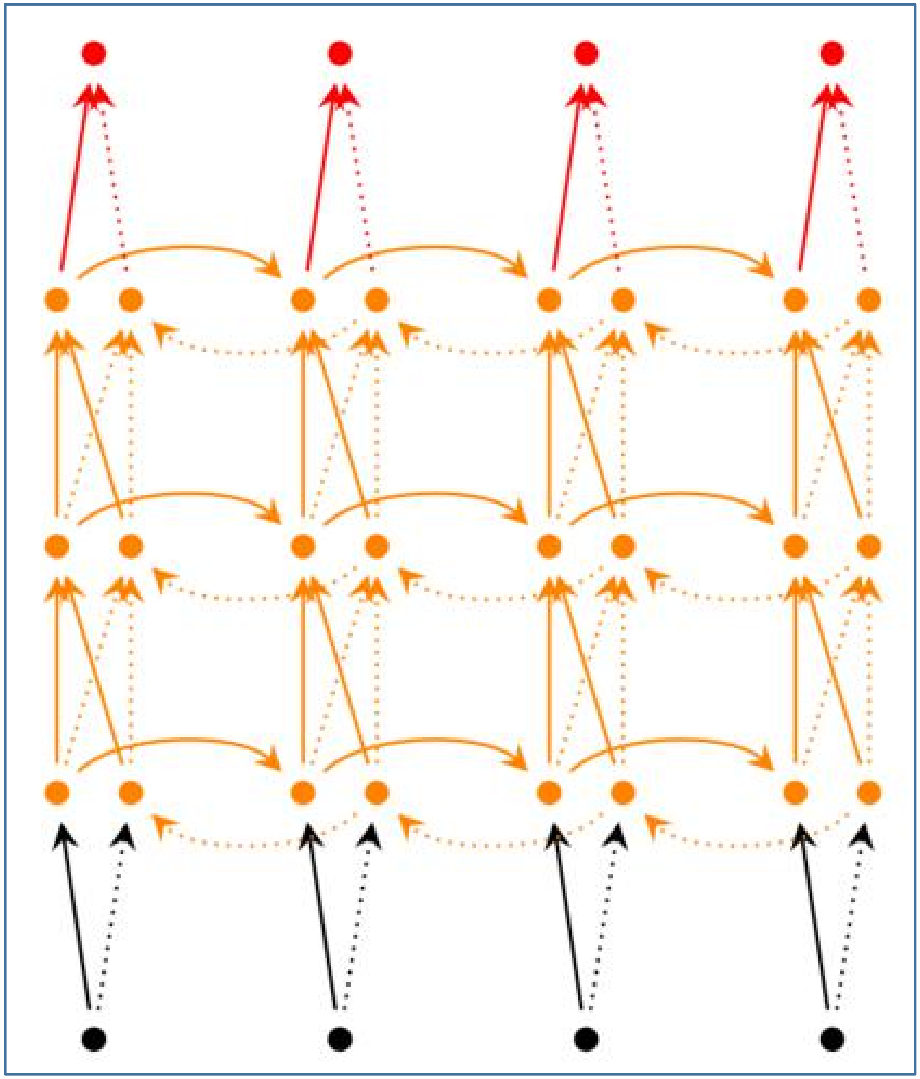
$$y_t = g(Vs_t + V's_t')$$
$$s_t = f(Ux_t + Ws_{t-1})$$
$$s_t' = f(U'x_t + W's_{t+1}')$$



深层RNN

深度循环神经网络计算方式为：

$$
\begin{aligned}
y_t &= g(V^{(i)} s_t^{(i)} + V'^{(i)} s_t'^{(i)}) \\
s_t^{(i)} &= f(U^{(i)} s_t^{(i-1)} + W^{(i)} s_{t-1}) \\
s_t'^{(i)} &= f(U'^{(i)} s_t'^{(i-1)} + W'^{(i)} s'_{t+1}) \\
&\cdots \\
s_t^{(1)} &= f(U^{(1)} x_t + W^{(1)} s_{t-1}) \\
s_t'^{(1)} &= f(U'^{(1)} x_t + W'^{(1)} s'_{t+1})
\end{aligned}
$$



2. 生成对抗网络：GAN，背景分析

- 人类除了识别与分析能力，还具备创造能力；
- 使机器具备创造的能力

  ①通过学习过去的文章，训练一个可以撰写文章人工智能作者

  ②可不可以创造一个人工智能画家，通过从画家过去的作品中学习，然后像任何艺术家一样画画？

- 这些任务止前确实难以自动化，GAN已经使部分任务变成可能
- 所以简单来说，G A N，要获得一个强大的英雄（即生成器generator），需要一个更强大的对手（即鉴别器discriminator）。

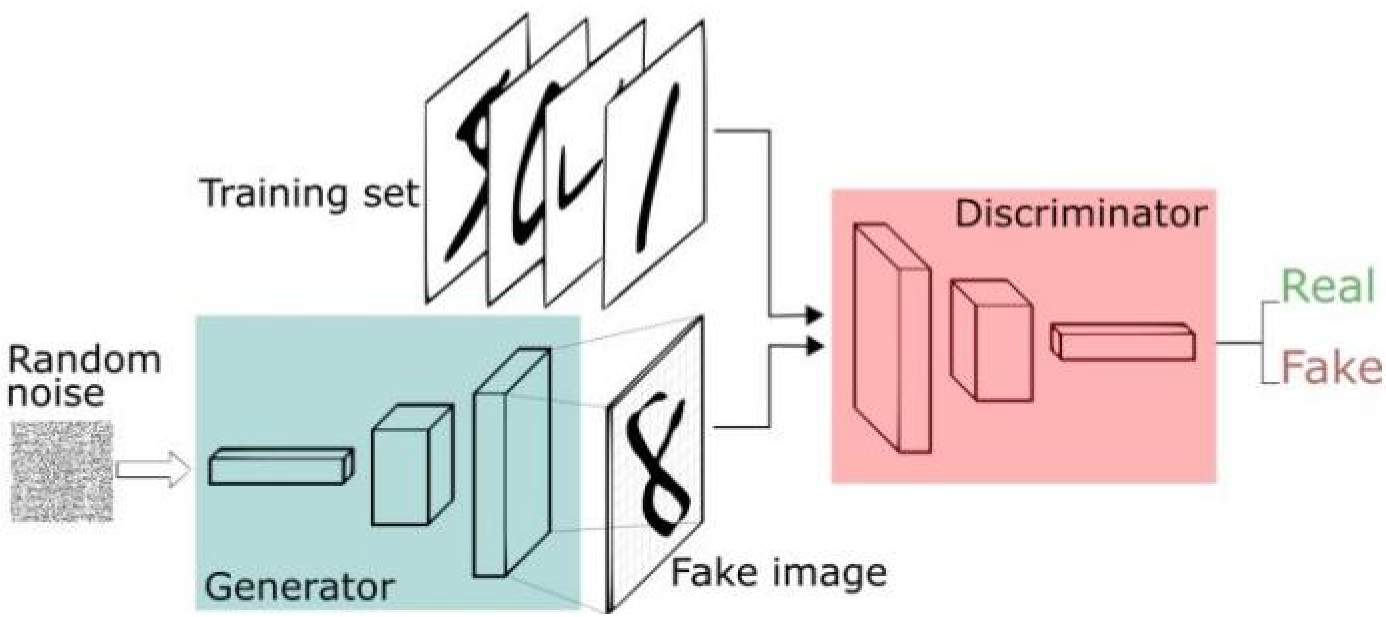
生成对抗网络：GAN

- GAN: Generative Adversarial Networks， 是一种博弈思想；
- 包含两种结构：

  生成式模型G（generative model), 判别式模型D（discriminative model）

①G是生成图片网络，接收随机噪声z，通过这个噪声生成图片，记做G(z)。

②D是判别网络，判别一张图片是不是“真实”。



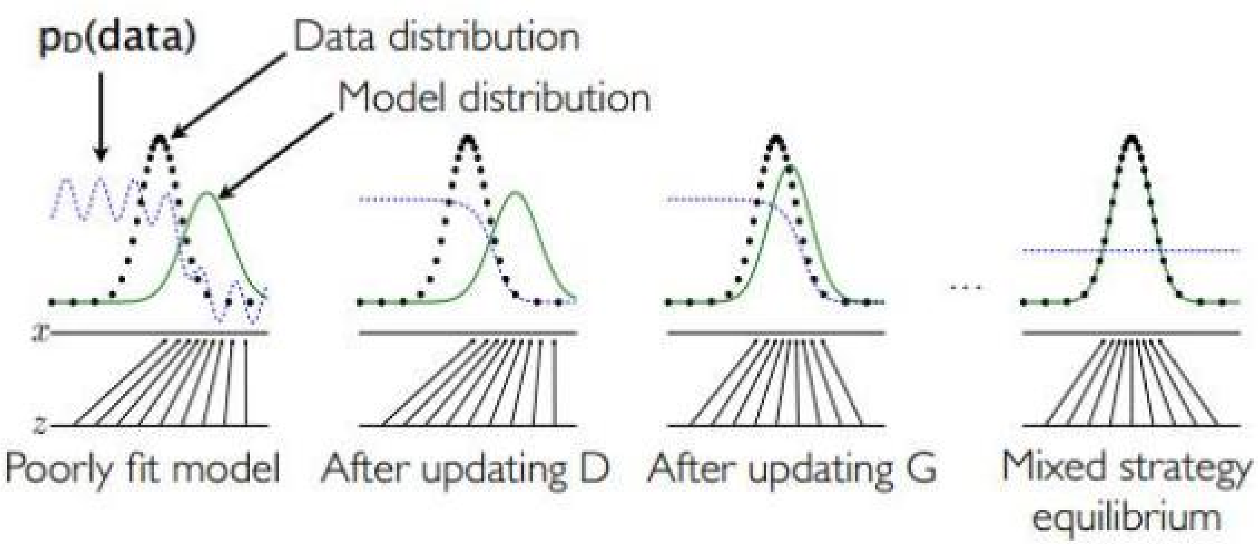
图三 GAN基本结构

GAN:训练

- 训练过程中:

  G的目标就是尽量生成真实的图片去欺骗判别网络D。

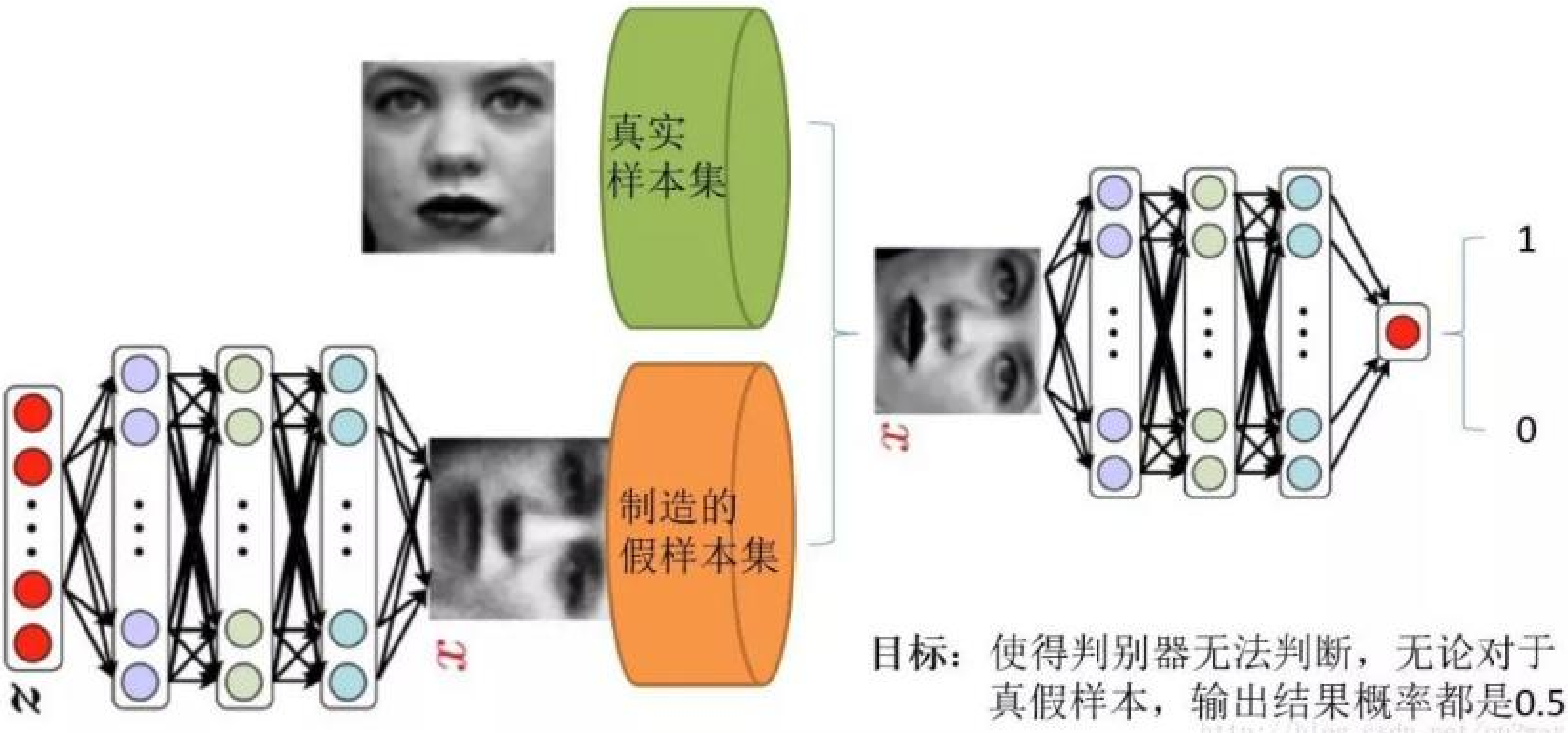
  D的目标就是尽量把G生成的图片和真实的图片分别开来,
- G和D构成了一个动态的**“博弈过程”**。
- 最后博弈的结果是: G可以生成足以“以假乱真”的图片G(z)。



注:图中的**黑色虚线**表示真实的样本的分布情况,**蓝色虚线**表示判别器判别概率的分布情况,**绿色实线**表示生成样本的分布。Z 表示噪声,Z 到 x 表示通过生成器之后的分布的映射情

GAN例子

- 判别网络(下图右半部分)
- 生成网络(下图左下部分)



GAN数学优化模型

- GAN数学优化问题，

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[log(D(x))] + E_{z\sim p_z(z)}[log(1-D(G(z)))]$$

- 以上问题交替迭代求解，

优化D：

$$\max_D V(D,G) = E_{x\sim p_{data}(x)}[log(D(x))] + E_{z\sim p_z(z)}[log(1-D(G(z)))]$$

优化G：

$$\min_G V(D,G) = E_{z\sim p_z(z)}[log(1-D(G(z)))]$$

GAN：应用

- 图像生成：

  「输入」满足一个输入分布，「输出」满足一个预期的期望 分布

  学习这两种图像之间的映射

卷积神经网络: CNN

卷积核学习特征，让机器具备视觉上的识别能力

循环神经网络: RNN

让机器具备分辨因果的能力和记忆功能，序列问题

生成对抗网络：GAN

让机器具备创造能力