AI300学院

从神经网络到深度学习

罗浩
浙江大学

AI300学院

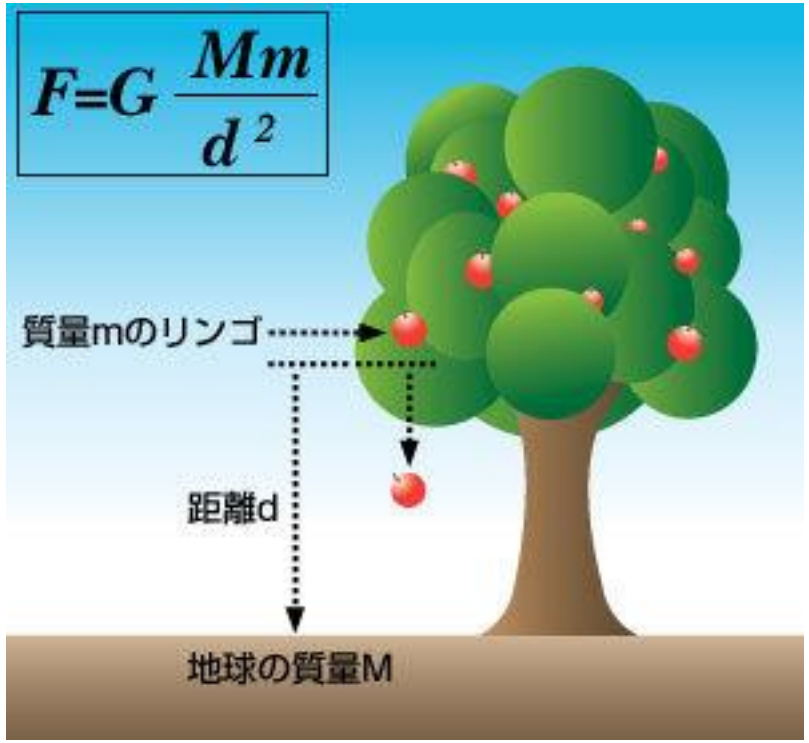
大纲

- 机器学习与神经网络
- 深度学习与卷积神经网络
- 深度学习框架

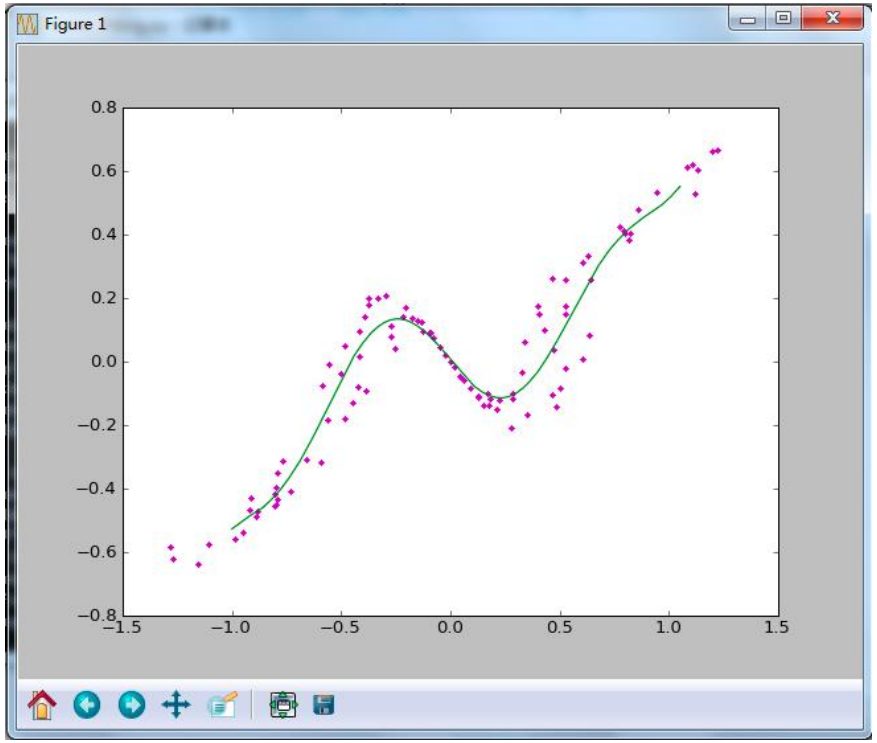
AI300学院

机器学习

- 模型驱动



- 数据驱动



AI300学院

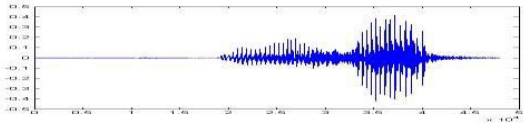
机器学习

AI300学院

机器学习

机器学习≈寻找一个映射

- Speech Recognition

$f($  $) = "How\ are\ you"$

- Image Recognition

$f($  $) = "cat"$

- AlphaGo

$f($  $) = "5-5"$

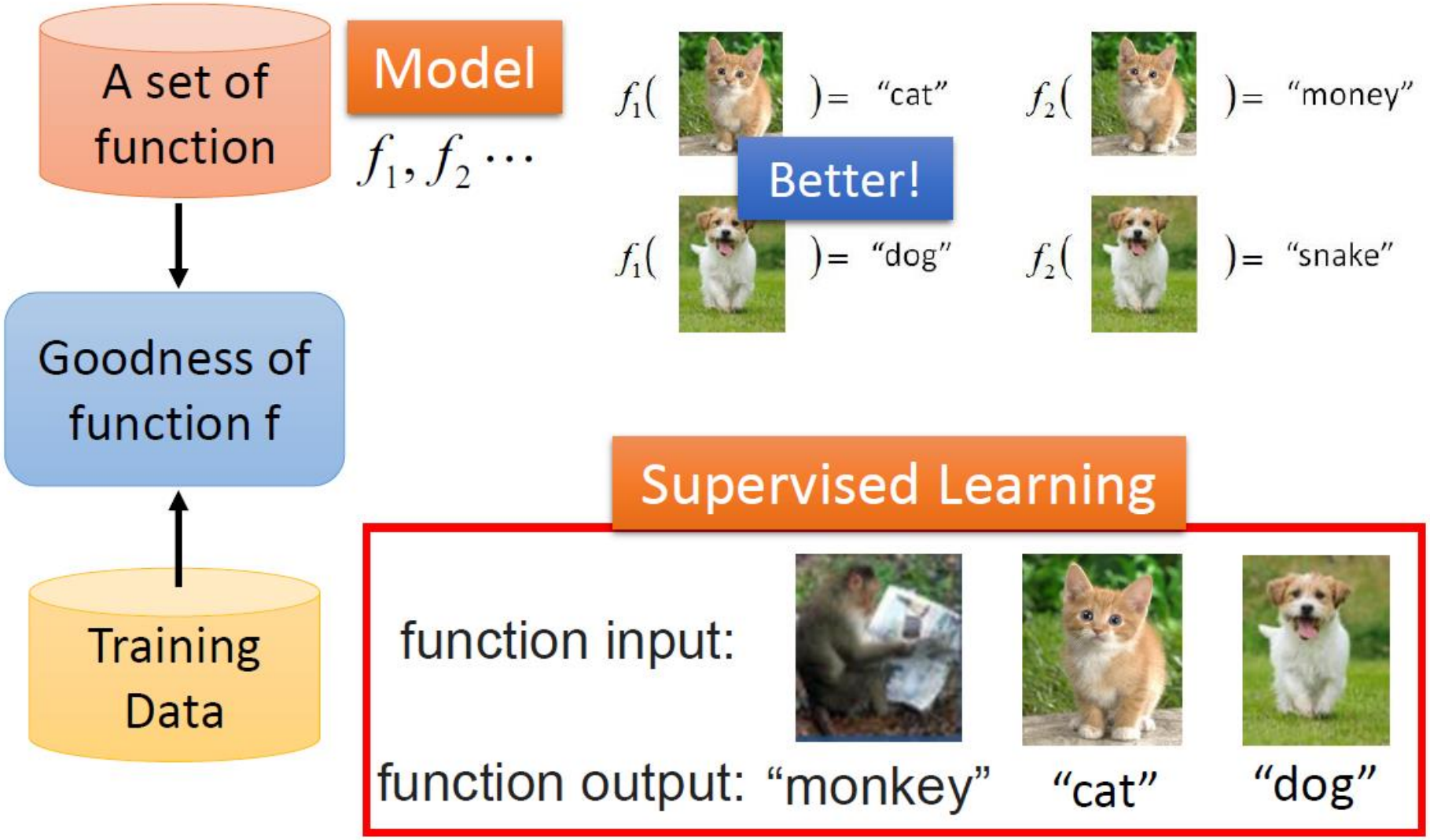
- Dialogue System

$f($ $"Hi"$ $) = "Hello"$

(what the user said) (system response)

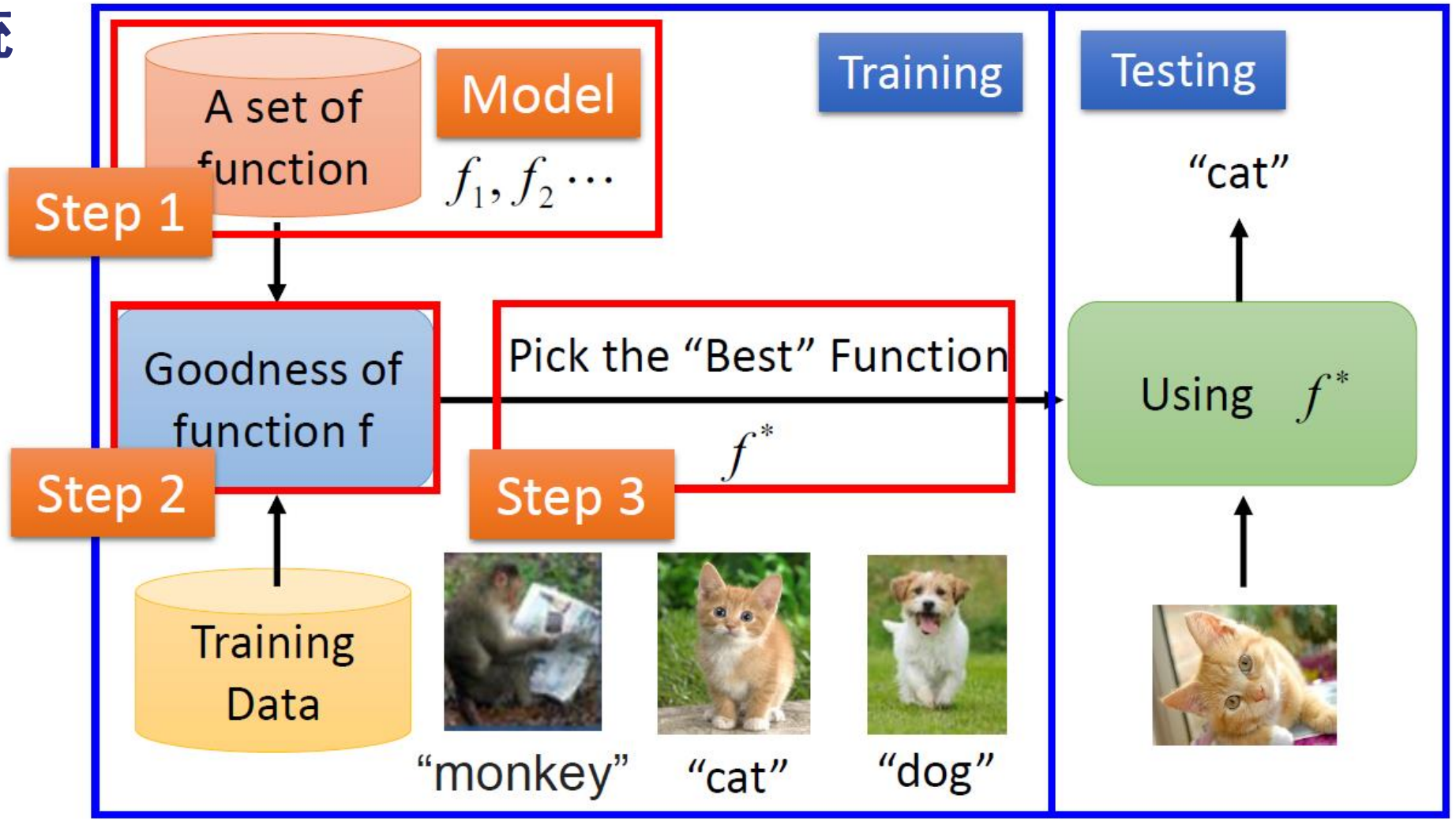
机器学习

机器学习


A set of function
Model
$f_1, f_2 \cdots$
$f_1($ $)=$ "cat"
$f_2($ $)=$ "money"
Better!
$f_1($ $)=$ "dog"
$f_2($ $)=$ "snake"
Goodness of function f
Training Data
Supervised Learning
function input:
function output: "monkey" "cat" "dog"

机器学习

机器学习系统


Training
Testing
A set of function
Model
$f_1, f_2 \cdots$
Step 1
Goodness of function f
Pick the "Best" Function
f^*
Step 2
Step 3
Training Data
"monkey"
"cat"
"dog"
"cat"
Using f^*

机器学习

数据集

•训练数据(training data)

顾名思义，训练数据用于训练学习模型，通常比例不低于总数据量的一半。

•交叉验证数据(cross validation data)

交叉验证数据用于衡量训练过程中模型的好坏，因为机器学习算法大部分都不是通过解析法得到的，而是通过不断迭代来慢慢优化模型，所以交叉验证数据就可以用来监视模型训练时候的性能变化。

•测试数据(testing data)

在模型训练好了之后，测试数据用于衡量最终模型的性能好坏，这也是模型性能好坏的衡量指标，交叉验证的指标只能用于监视和辅助模型训练，不能用来代表模型好坏，所以哪怕交叉验证的准确度是100%而测试数据的准确度是10%，那么模型也是不能被认可的。通常交叉验证和测试数据的比例各占一小半。

机器学习

监督学习

训练集的差异

监督学习　　半监督学习　　无监督学习

机器学习

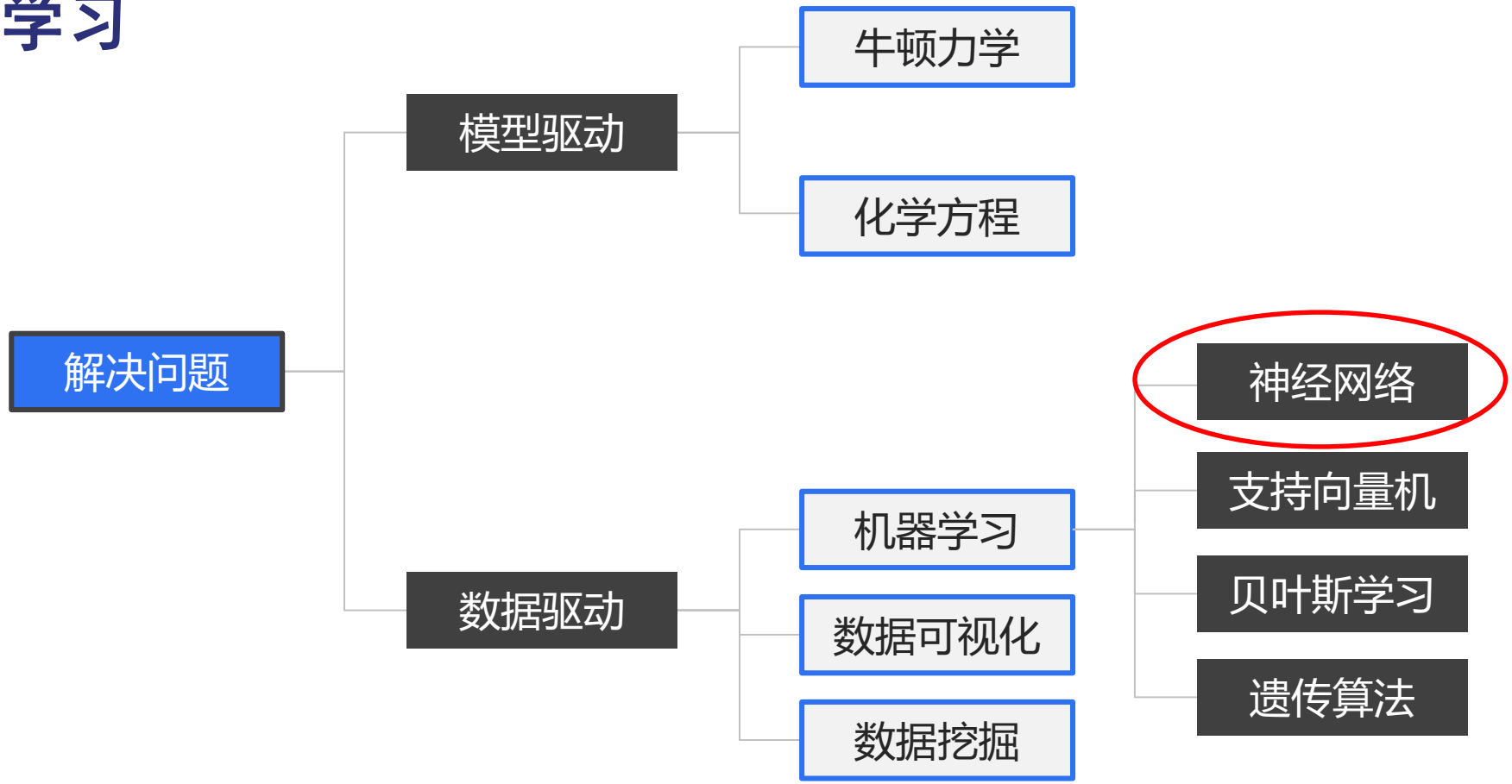
迁移学习

源域→目标域

- **数据分布自适应：**通过一些变换，将不同的数据分布的距离拉近
- **特征选择：**假设源域和目标域中均含有一部分公共的特征
- **子空间学习：**假设源域和目标域数据在变换后的子空间中会有着相似的分布

机器学习

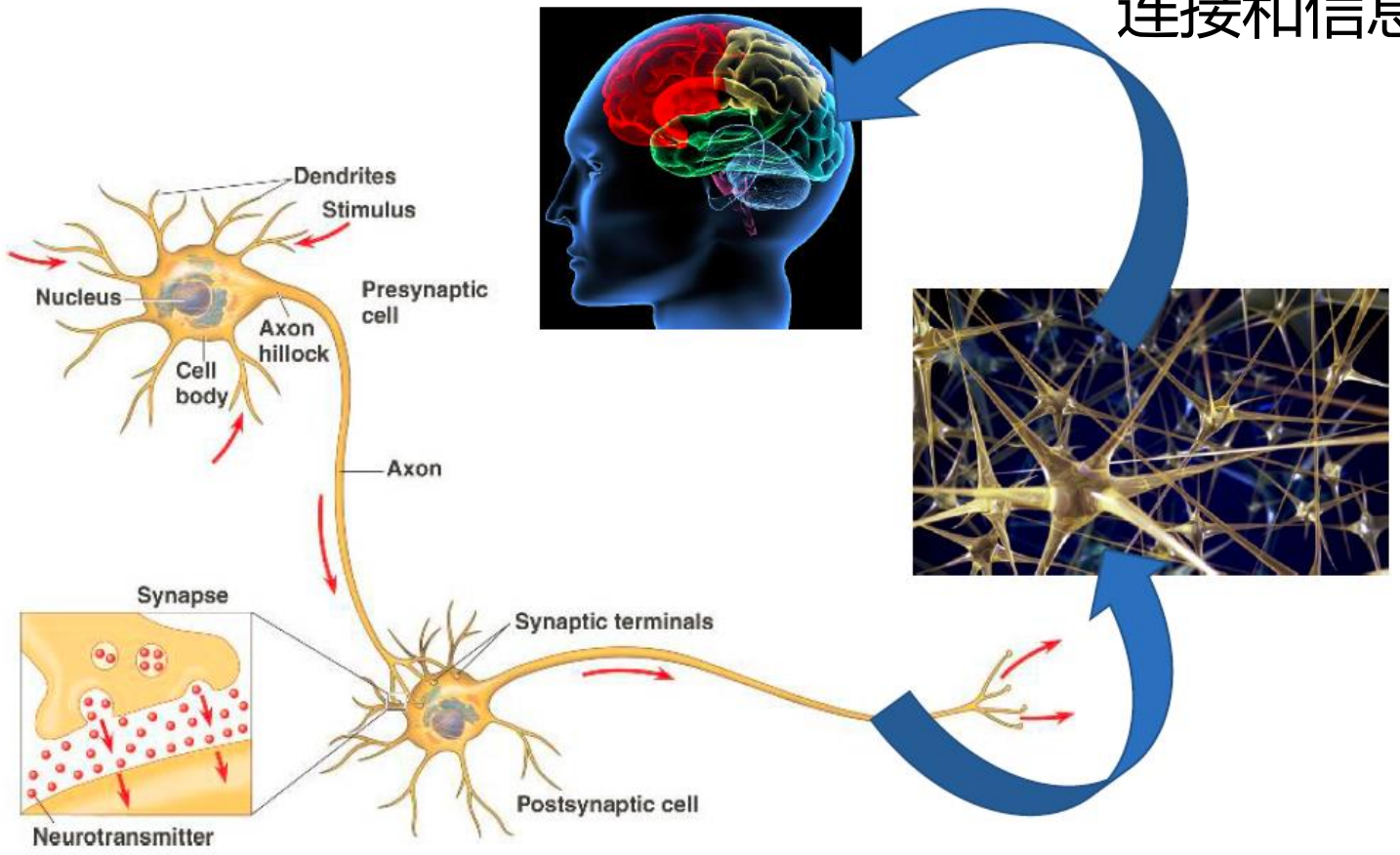
机器学习


解决问题
模型驱动
牛顿力学
化学方程
数据驱动
机器学习
神经网络
支持向量机
贝叶斯学习
遗传算法
数据可视化
数据挖掘

神经网络

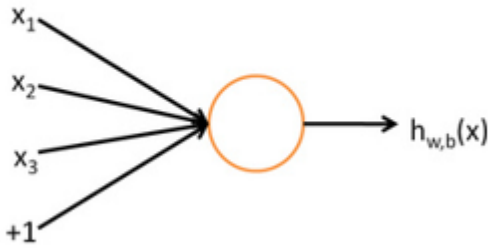
神经元—神经网络的基本运算单位

连接和信息传递



神经网络

单层感知机（神经元）



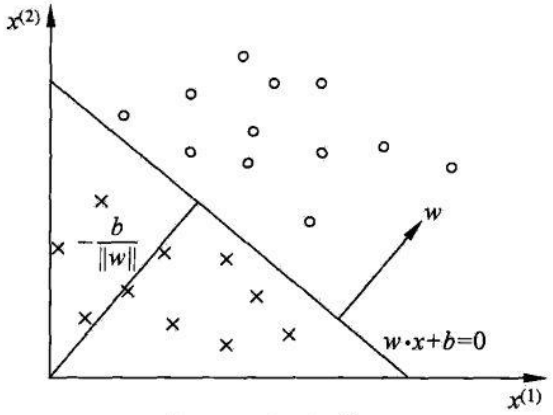


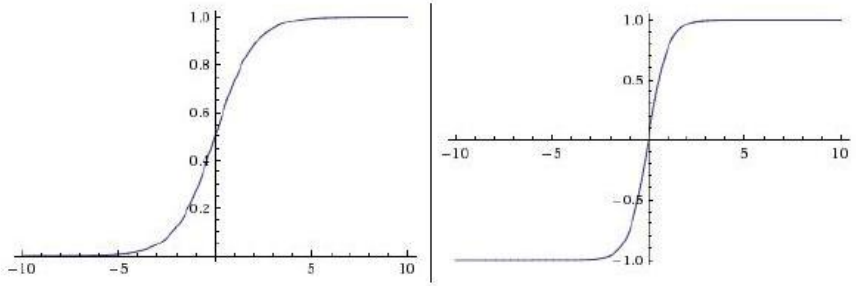
图 2.1　感知机模型

非线性

激活函数

$$f_{sigmoid}(z) = \frac{1}{1+e^{-z}}$$

$$f_{tanh}(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$



Left: Sigmoid non-linearity squashes real numbers to range between [0,1] **Right:** The tanh non-linearity squashes real numbers to range between [-1,1].

- 输入：　$X = [x_1, x_2, x_3, \ldots, x_i]^T$
- 权重：　$W = [w_1, w_2, w_3, \ldots, w_i]$
- 偏置：　b
- 输出：　$y = sign(\sum w_i x_i + b) = sign(WX + b)$

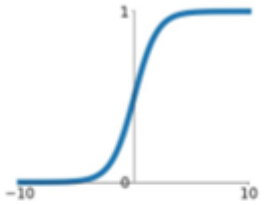
$$f'_{sigmoid}(z) = \frac{1}{1+e^{-z}} - \frac{1}{(1+e^{-z})^2} = f_{sigmoid}(z)[1 - f_{sigmoid}(z)]$$

神经网络

常用激活函数

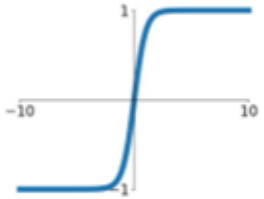
Sigmoid

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



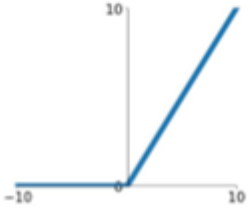
Tanh

$$\tanh(x)$$



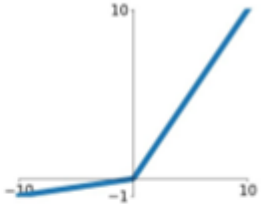
ReLU

$$\max(0, x)$$



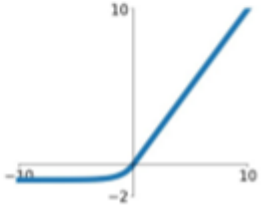
Leaky ReLU

$$\max(0.1x, x)$$



ELU

$$\begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases}$$



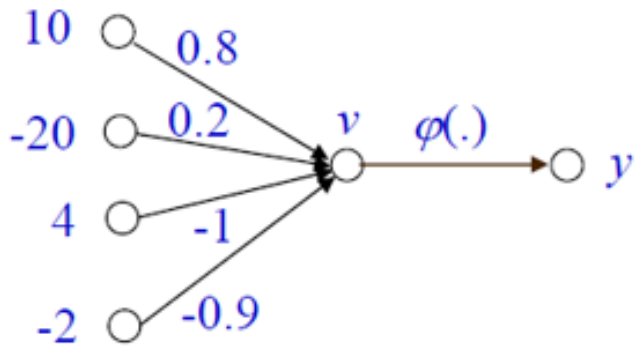
神经网络

神经元运算举例

一个神经元接收四个其他神经元的输入，每一个的激励水平是10，-20，4以及-2。每一个神经元的突触权重是0.9，0.2，-1.0以及-0.9。于是神经元的输出可以计算如下：

$$v_j = \sum_{i=1}^{4} w_{ji} x_i$$

$$v_j = 1.8$$



假设激励函数是Sigmoid, 偏置为0.1:

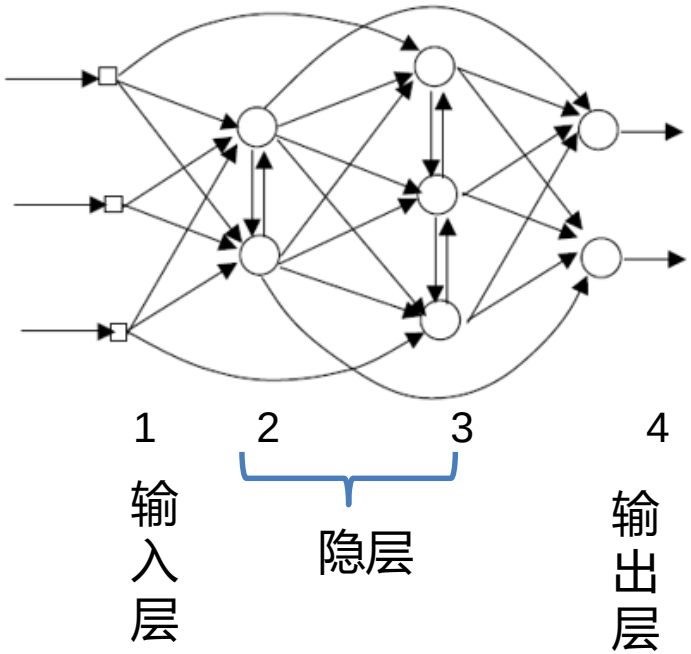
$$y_j = \varphi(v_j + 0.1) = \frac{1}{1 + e^{-(v_j + 0.1)}} = \frac{1}{1 + e^{-(1.8 + 0.1)}} = 0.8699$$

神经网络

神经元->神经网络

单个神经元的计算能力有限，无法处理大规模的运算

多个神经元的集合叫层

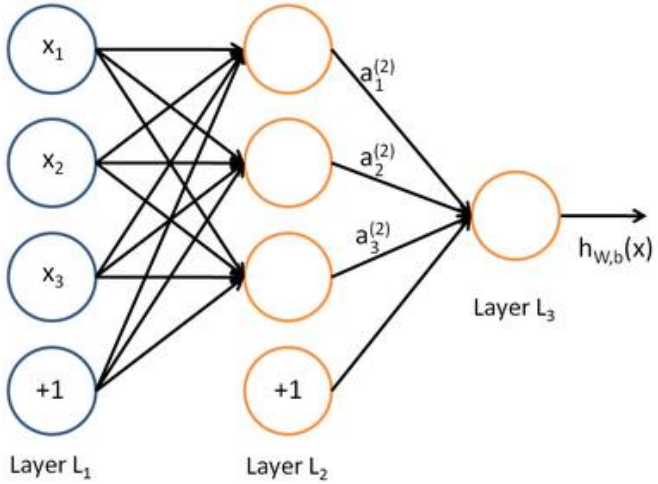


单向的

层内也可以存在连接

神经网络

多层前馈神经网络



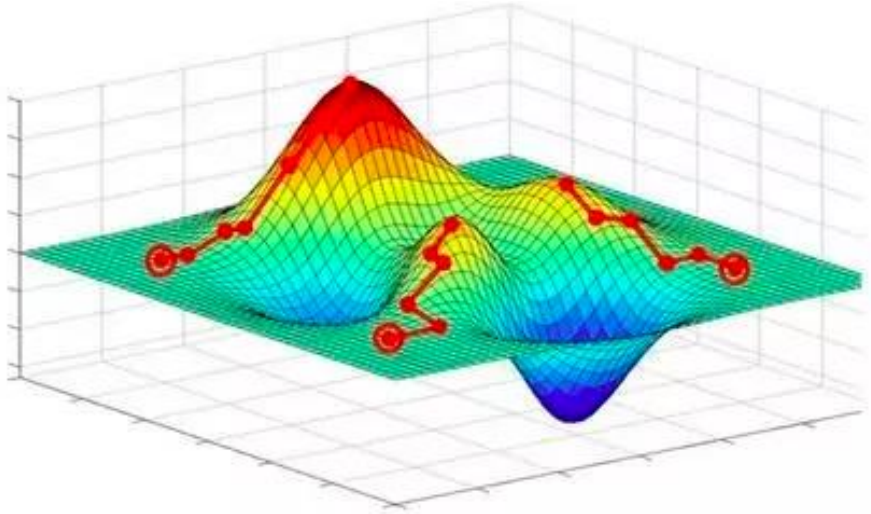
怎么训练? → 预测≈标签 → 损失函数 → min $J(W,b,x,y)$

- 前馈: $z_i^{(2)} = \sum w_{ij}^{(1)} x_j + b^{(1)}$

  $a_i^{(2)} = f(z_i^{(2)}) = f(\sum w_{ij}^{(1)} x_j + b^{(1)})$

- 输出: $h_{W,b}(x) = a_1^{(3)} = f(\sum w_{1j}^{(2)} a_j^{(2)} + b^{(2)})$

- 通用: $a_i^{(l+1)} = f(z_i^{(l+1)}) = f(\sum w_{ij}^{(l)} a_j^{(l)} + b^{(l)})$



神经网络

反向传播(Backward Propagation)

通过梯度来更新每一层的网络的参数**W**

$f(a,b,c) = (a+b)c$

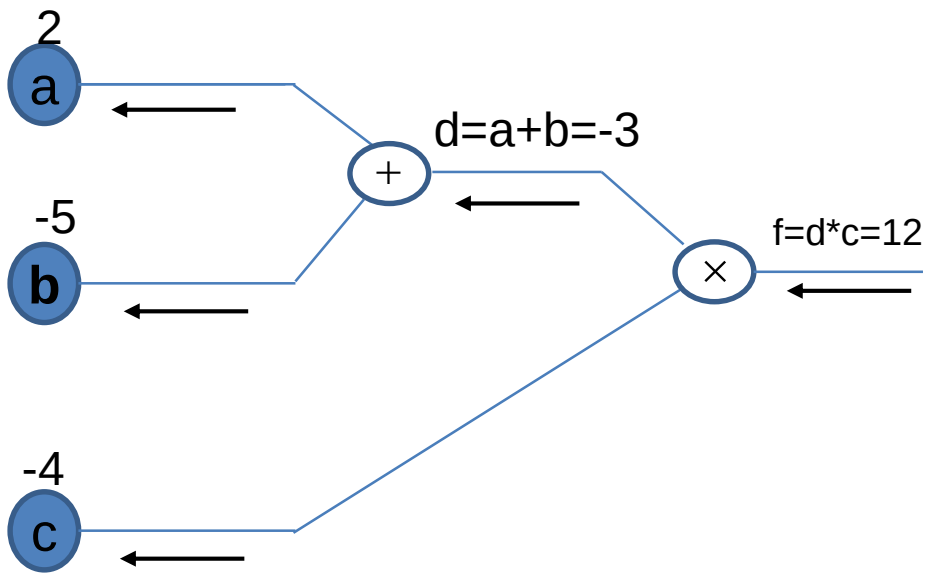
e.g.

$a = 2, b = -5, c = -4$

$d = a + b \quad \frac{\partial d}{\partial a} = 1, \frac{\partial d}{\partial b} = 1$

$f = dc \quad \frac{\partial d}{\partial a} = c, \frac{\partial f}{\partial c} = 1$

计算 $\frac{\partial f}{\partial a}, \frac{\partial f}{\partial b}, \frac{\partial f}{\partial c}$



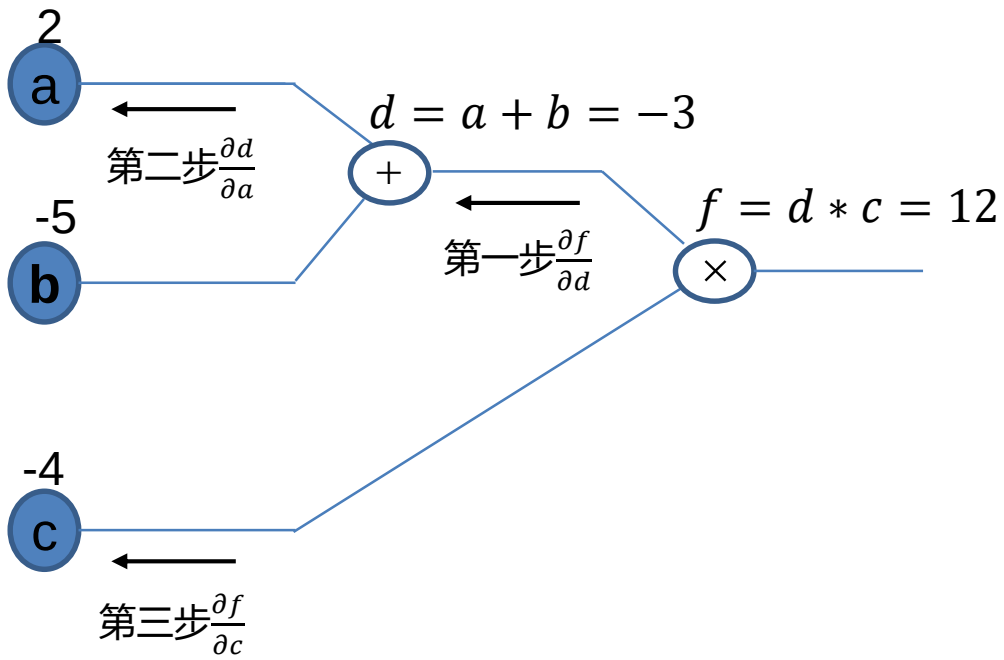
神经网络

链式法则

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial d} * \frac{\partial d}{\partial a} = c * 1 = c$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial d} * \frac{\partial d}{\partial b} = c * 1 = c$$

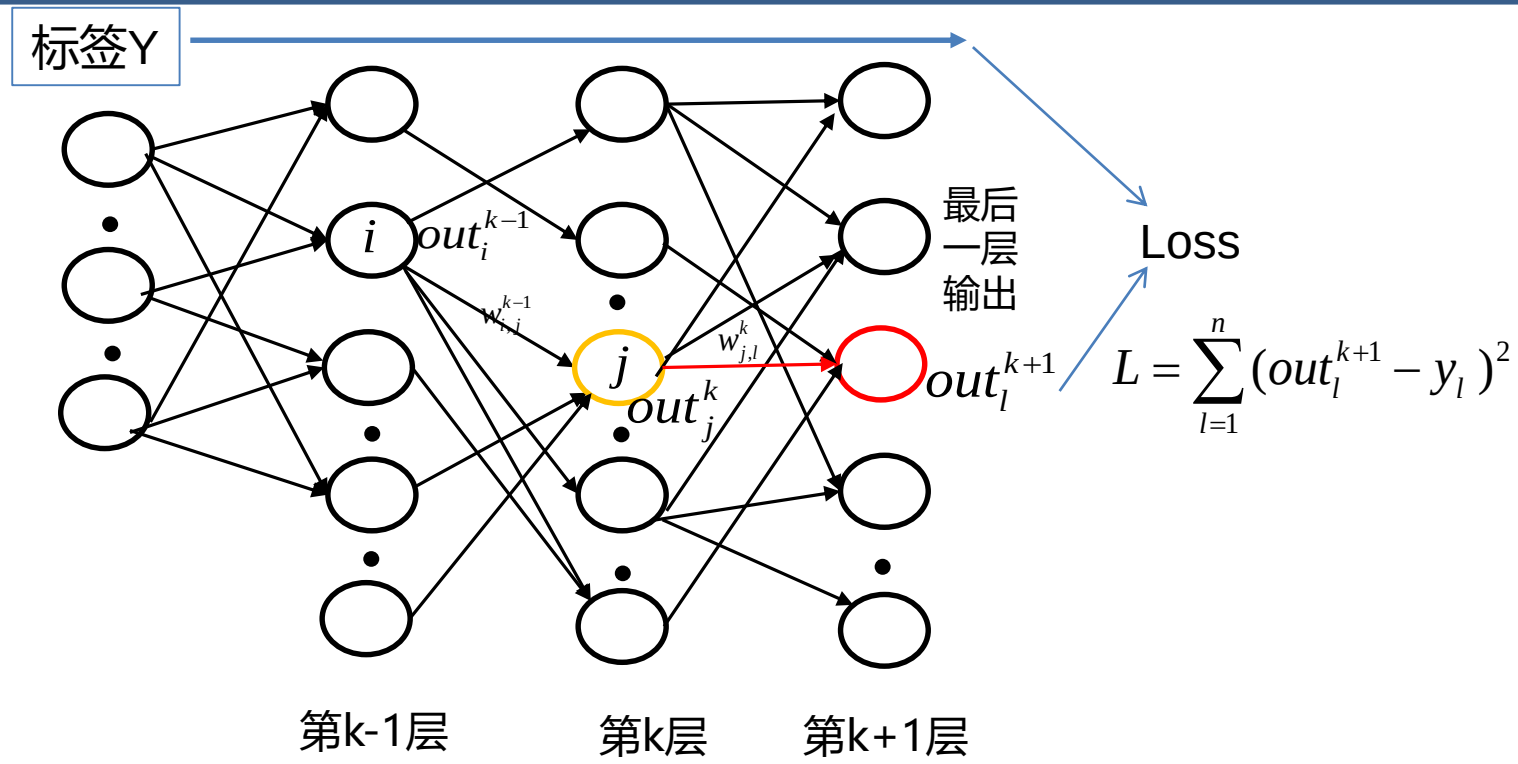
$$\frac{\partial f}{\partial c} = 1$$



反传意味着计算关于网络参数的导数

神经网络

梯度反传



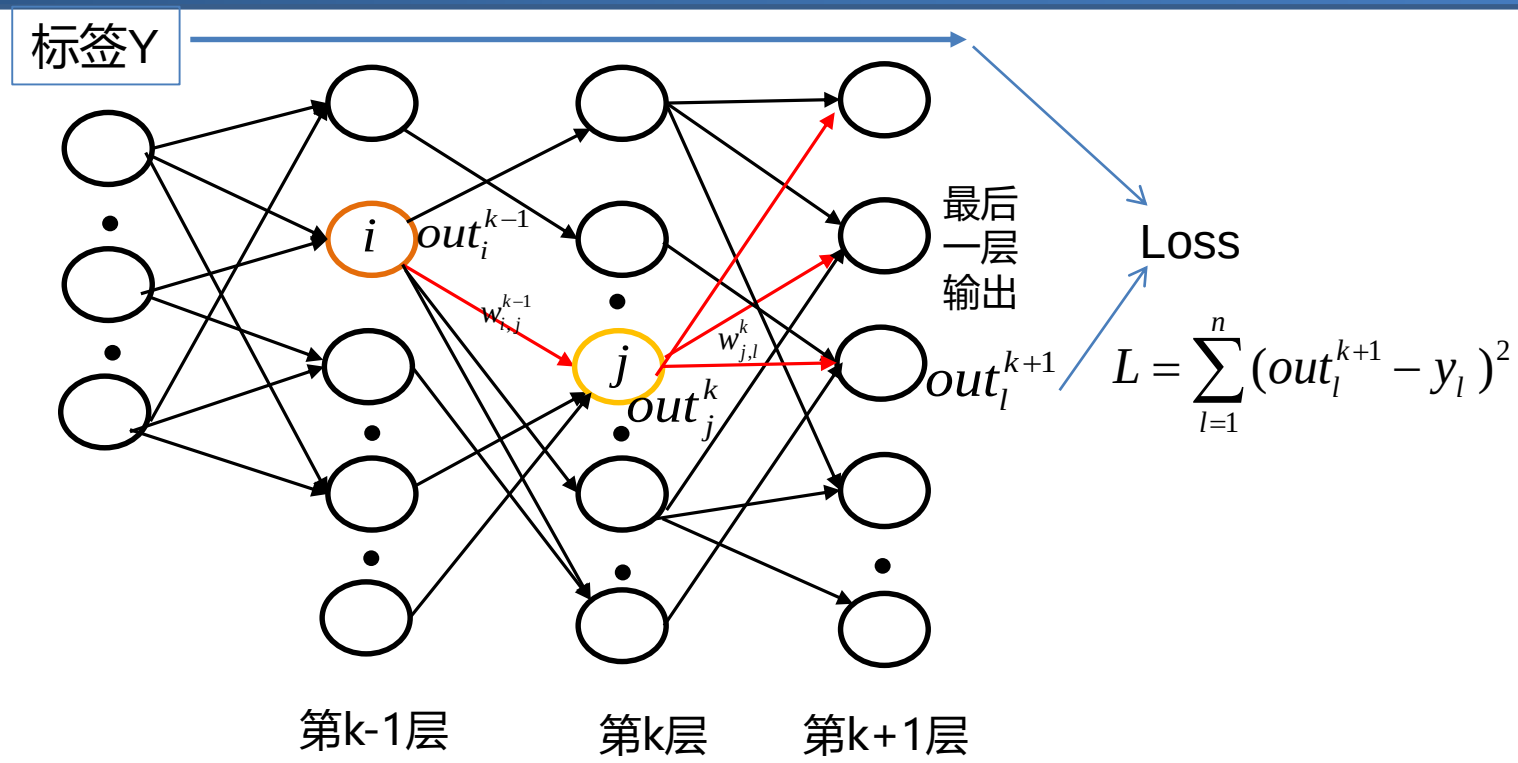
计算损失函数L关于输出层的梯度：

$$\frac{\partial L}{\partial out_l^{k+1}}=2\left(y_l-out_l^{k+1}\right) \qquad out_l^{k+1}=f(in^{k+1},w_l)$$

$$\frac{\partial L}{\partial w_{j,l}^k}=\frac{\partial L}{\partial out_l^{k+1}}\frac{\partial out_l^{k+1}}{\partial w_{j,l}^k}$$

神经网络

梯度反传



关于每一个输出节点：$$\frac{\partial L}{\partial out_j^k} = \sum_{l=1}^{n_{k+1}} w_{j,l}^k f'(in_l^{k+1}) \frac{\partial L}{\partial out_l^{k+1}}$$

关于每一层网络权重：$$\frac{\partial L}{\partial w_{i,j}^{k-1}} = \frac{\partial L}{\partial out_j^k} f'(in_j^k) out_i^{k-1}$$

AI300学院

神经网络

反向传播推导示例——梯度下降

$$J(W,b,x,y) = \frac{1}{2}\left\|y - h_{w,b}(x)\right\|^2 = \frac{1}{2}\left\|y - a^{(l+1)}\right\|^2 = \frac{1}{2}\left\|y - f(z_i^{(l+1)})\right\|^2$$

$$f(z_i^{(l+1)}) = f(\sum w_{ij}^{(l)} a_j^{(l)} + b^{(l)})$$

$$\frac{\partial J}{\partial w_{ij}^{(l)}} = \frac{\partial J}{\partial z_i^{(l+1)}}\frac{\partial z_i^{(l+1)}}{\partial w_{ij}^{(l)}} = \delta^{(l+1)}\frac{\partial z_i^{(l+1)}}{\partial w_{ij}^{(l)}}$$

$$= \delta^{(l+1)}\frac{\partial \sum w_{ij}^{(l)} a_j^{(l)} + b^{(l)}}{\partial w_{ij}^{(l)}}$$

$$= \delta^{(l+1)} a_j^{(l)}$$

$$\frac{\partial J}{\partial b^{(l)}} = \frac{\partial J}{\partial z_i^{(l+1)}}\frac{\partial z_i^{(l+1)}}{\partial b^{(l)}} = \delta^{(l+1)}\frac{\partial z_i^{(l+1)}}{\partial b^{(l)}}$$

$$= \delta_i^{(l+1)}\frac{\partial \sum w_{ij}^{(l)} a_j^{(l)} + b^{(l)}}{\partial b^{(l)}}$$

$$= \delta_i^{(l+1)}$$

其中

$$\delta_i^{(l+1)} = \frac{\partial J}{\partial z_i^{(l+1)}}$$

$$= \frac{\partial \frac{1}{2}\left\|y - f(z_i^{(l+1)})\right\|^2}{\partial z_i^{(l+1)}}$$

$$= -(y - f(z_i^{(l+1)}))f'(z_i^{(l+1)})$$

$$f'(z_i^{(l+1)}) = f(z_i^{(l+1)})\ [1 - f(z_i^{(l+1)}]$$

$$\frac{\partial J}{\partial w_{ij}^{(l-1)}} = \frac{\partial J}{\partial z_i^{(l+1)}}\frac{\partial z_i^{(l+1)}}{\partial z_k^{(l)}}\frac{\partial z_k^{(l)}}{\partial w_{kj}^{(l-1)}}$$

$$= \delta_i^{(l+1)}\frac{\partial z_i^{(l+1)}}{\partial z_k^{(l)}}\frac{\partial z_k^{(l)}}{\partial w_{kj}^{(l-1)}}$$

$$= \delta_i^{(l+1)}\frac{\partial \sum w_{ij}^{(l)} a_j^{(l)} + b^{(l)}}{\partial z_k^{(l)}}\frac{\partial z_k^{(l)}}{\partial w_{kj}^{(l-1)}}$$

$$= \delta_i^{(l+1)}\frac{\partial \sum w_{ik}^{(l)} f(z_k^{(l)}) + b^{(l)}}{\partial z_k^{(l)}}\frac{\partial z_k^{(l)}}{\partial w_{kj}^{(l-1)}}$$

$$= (\sum w_{ik}^{(l)}\delta_i^{(l+1)})f'(z_k^{(l)})\frac{\partial z_k^{(l)}}{\partial w_{kj}^{(l-1)}}$$

$$= (\sum w_{ik}^{(l)}\delta_i^{(l+1)})f'(z_k^{(l)})a_j^{(l-1)}$$

$$= \delta_i^{(l-1)} a_j^{(l-1)}$$

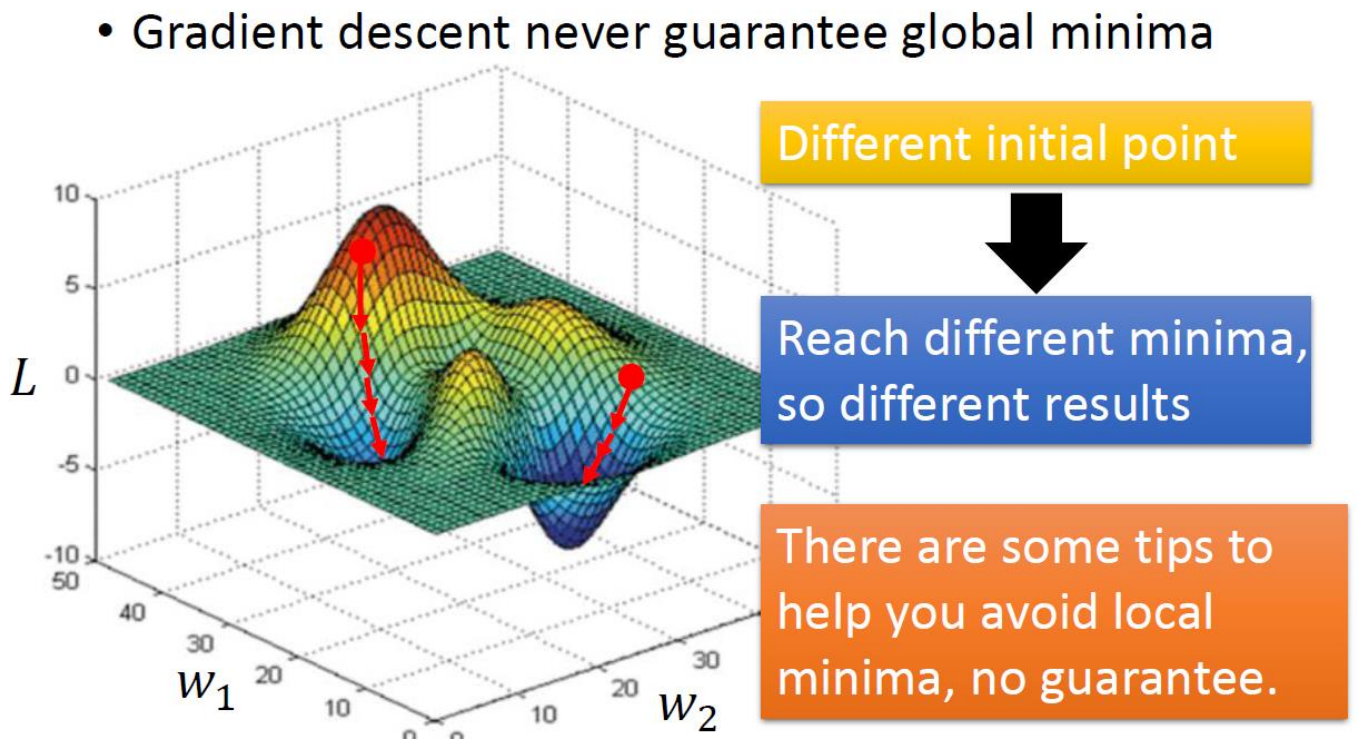
Update

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \alpha\frac{\partial J}{\partial w_{ij}^{(l)}} \qquad b^{(l)} = w_{ij}^{(l)} - \alpha\frac{\partial J}{\partial b^{(l)}}$$

神经网络

神经网络的缺陷

- 局部最优



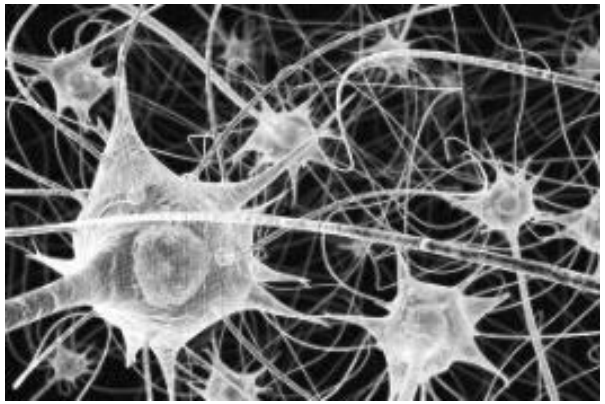
- 梯度弥散
- 参数量指数级增加

层数过多无法训练

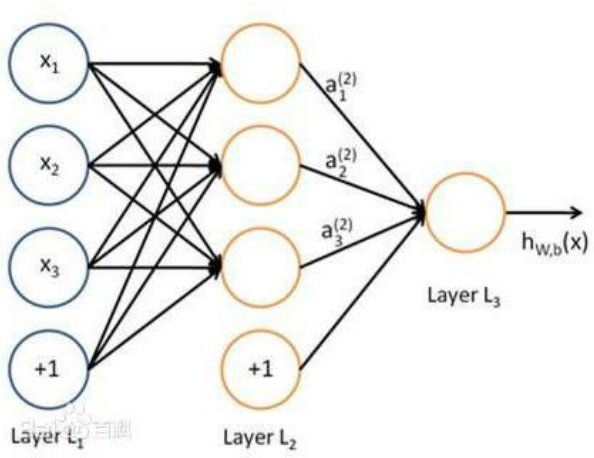
神经网络

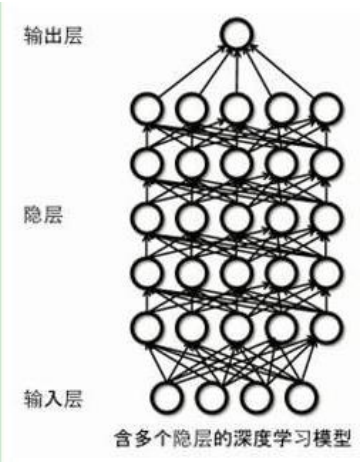
神经网络的优缺点

神经元细胞



神经网络





- 可以拟合任何非线性模型，实现分类、回归
- 基于梯度训练，方便编程实现扩展

- 层数过多会出现梯度弥散，无法训练
- 为了降低输入的维度，通常需要对原始数据进行人工特征提取
- …

神经网络

神经网络训练过程

为了得到所有点的导数：

1.前向传递计算每一个节点的输出

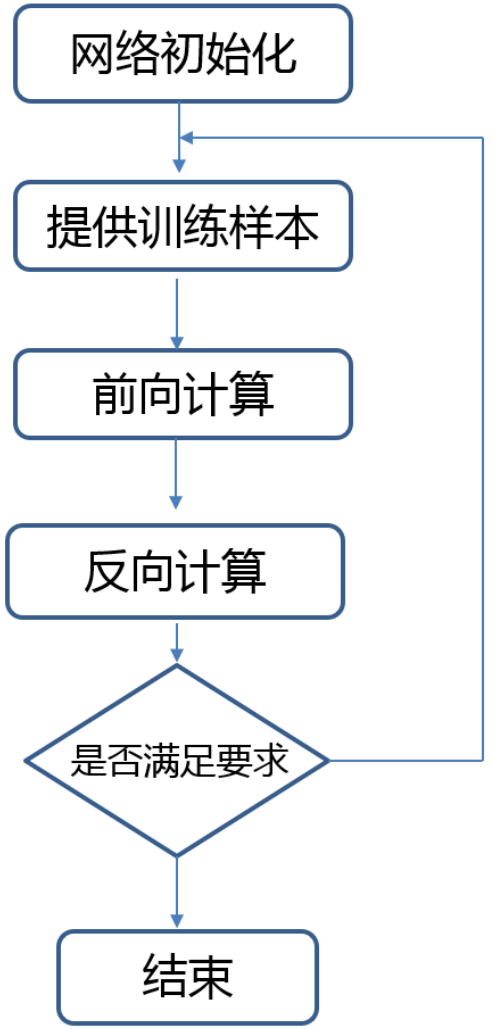
2.从输出层反向依次计算每一个节点的梯度

3.利用梯度下降来更新连接权重：

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

η 是学习率，也就是步长

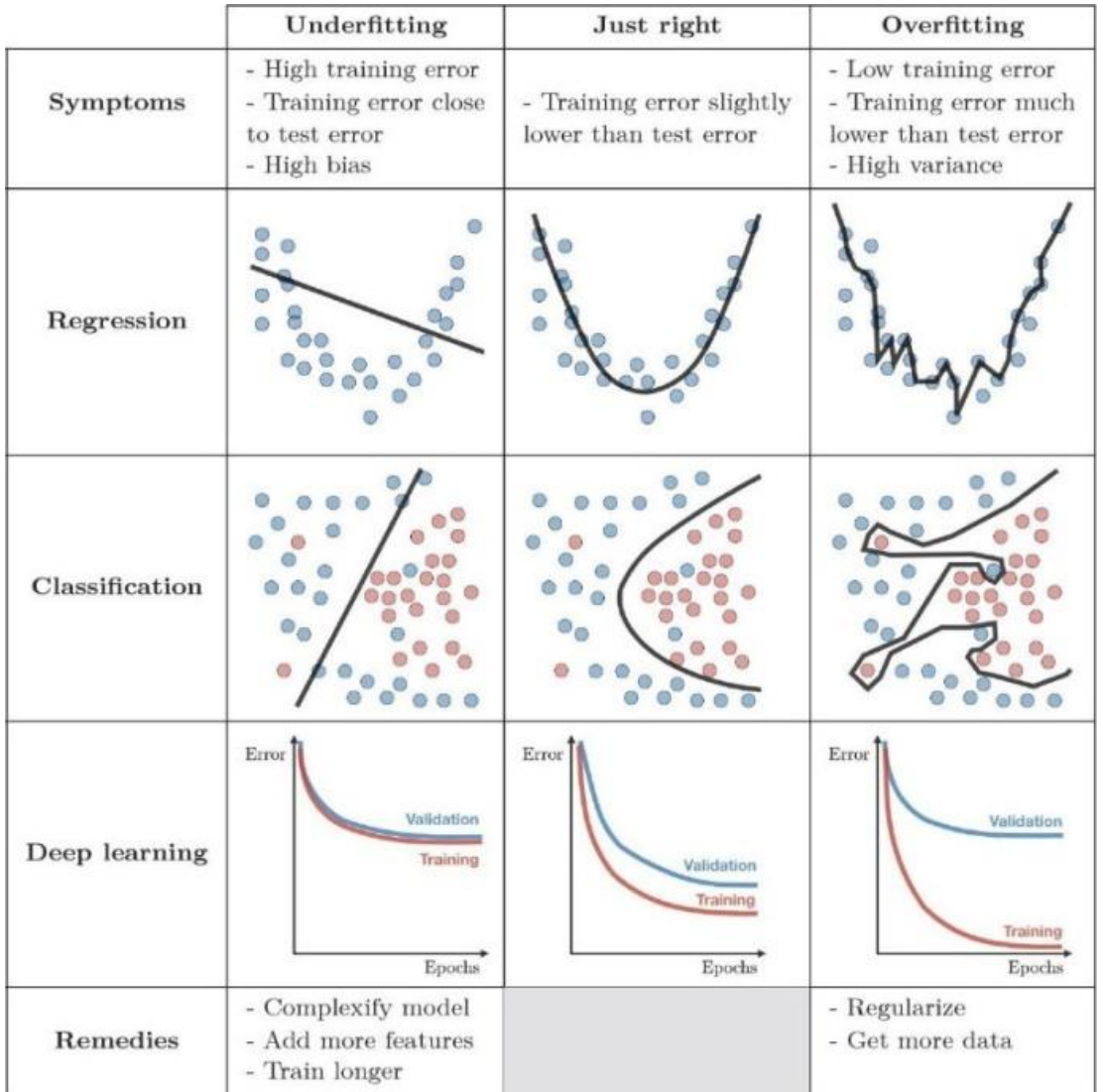
$\frac{\partial L}{\partial w}$ 是梯度下降方向



神经网络

过拟合与欠拟合

- 抑制过拟合的方式
  - Dropout
  - 正则化
  - 数据增广
  - 减小网络;
  - Early stopping
  - ...



AI300学院

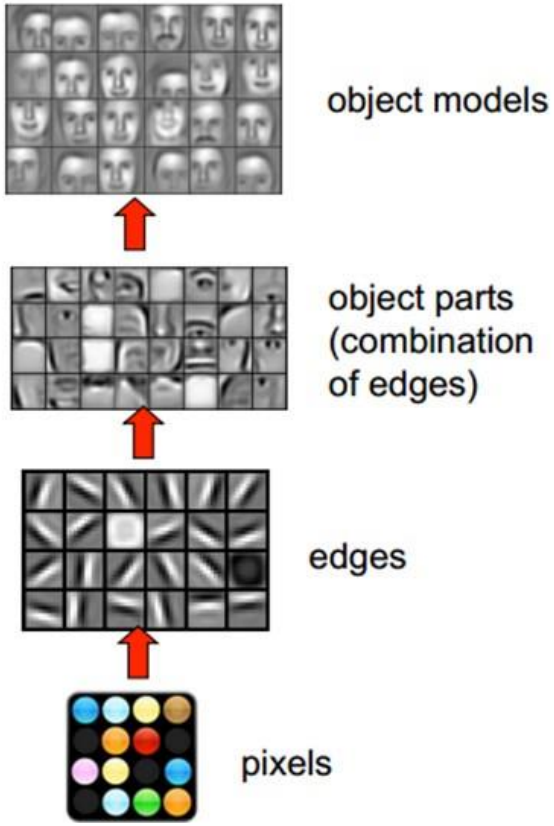
神经网络

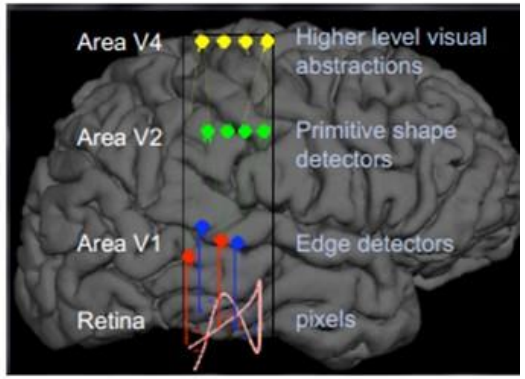
到底是什么限制了神经网络的发展?

深度学习

为什么要深度?

层数越多，表达能力越强





深度学习

深度学习的发展

2006

诞生（DNN, DBN）

Hinton 发表了第一篇深度学习论文

A fast learning algorithm for deep belief nets

2012

首次应用（CNN）

Hinton学生在2012年ImageNet比赛中大幅刷新准确度

ImageNet Classification with Deep Convolutional Neural Networks.

2014

时序列（RNN）

谷歌利用 Recurrent neural network 实现图文转换.

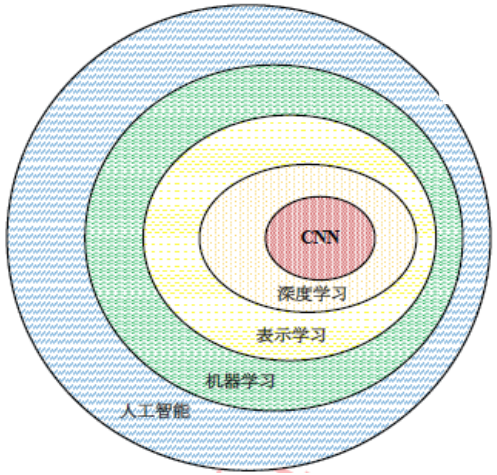
2016

爆发（Alpha Go）

谷歌旗下DeepMind公司的围棋机器人Alpha Go 击败李世石，后以Master身份连续60场不败.

卷积神经网络

深度学习



- 解决问题
  - 模型驱动
    - 牛顿力学
    - 化学方程
  - 数据驱动
    - 机器学习
      - 神经网络 → 深度学习
      - 支持向量机
      - 贝叶斯学习
      - 遗传算法
    - 数据可视化
    - 数据挖掘

卷积神经网络

卷积神经网络——卷积

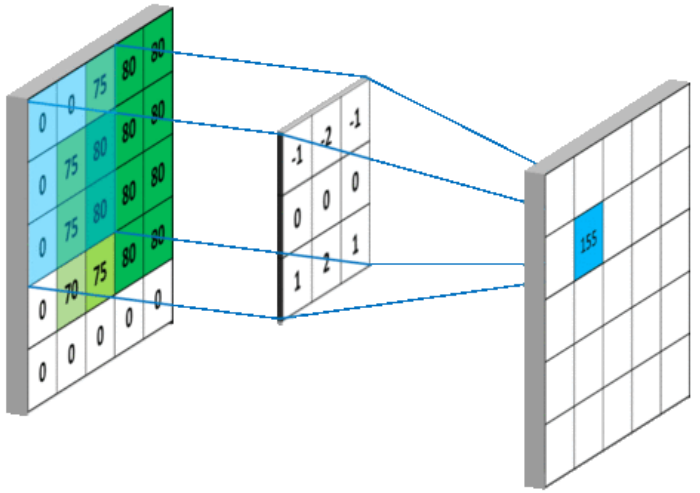
$$s(n) = \sum_{m=0}^{N-1} f(m)g(n-m) \quad n = 0, 1, \ldots, N+M-2$$

例：假设$f(n) = [1, 2, 3]$；$g(n) = [2, 3, 1]$；则：

$$\begin{aligned}
s(0) &= f(0)g(0-0) + f(1)g(0-1) + f(2)g(0-2) = 1*2 + 2*0 + 3*0 = 2 \\
s(1) &= f(0)g(1-0) + f(1)g(1-1) + f(2)g(1-2) = 1*3 + 2*2 + 3*0 = 7 \\
s(2) &= f(0)g(2-0) + f(1)g(2-1) + f(2)g(2-2) = 1*1 + 2*3 + 3*2 = 13 \\
s(3) &= f(0)g(3-0) + f(1)g(3-1) + f(2)g(3-2) = 1*0 + 2*1 + 3*3 = 11 \\
s(4) &= f(0)g(4-0) + f(1)g(4-1) + f(2)g(4-2) = 1*0 + 2*0 + 3*1 = 3
\end{aligned}$$

所以最后的卷积结果为：

$$s(n) = [2, 7, 13, 11, 3]$$



一维卷积

二维卷积

卷积神经网络

卷积神经网络——卷积

原图

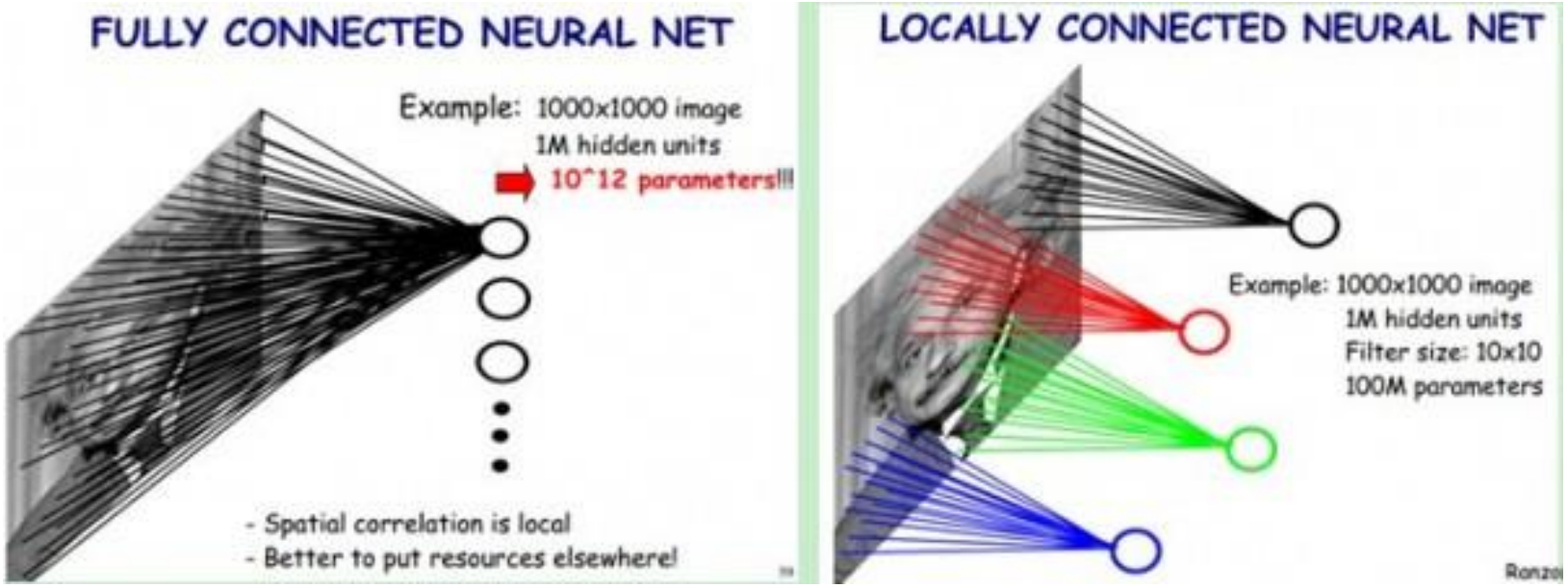
整体边缘滤波器

横向边缘滤波器

纵向边缘滤波器

卷积神经网络

卷积神经网络三特性——局部连接

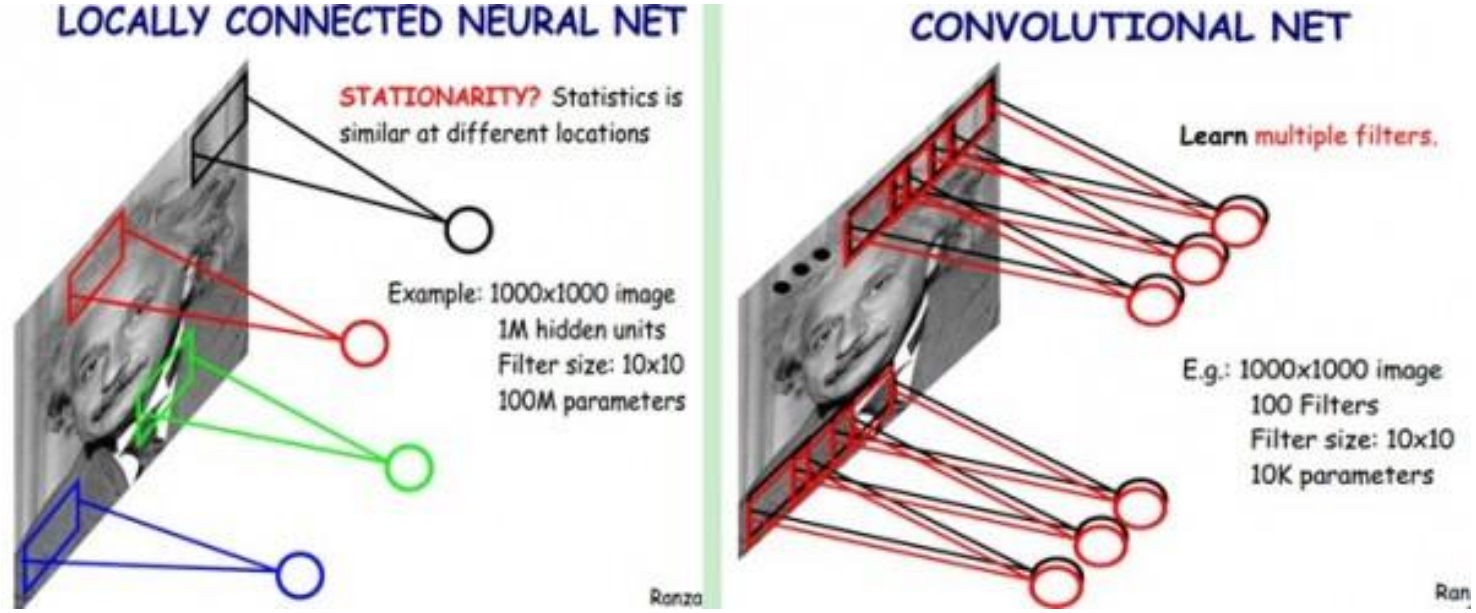


- 人类神经元
- 感受野
- 减少参数

根据视觉神经相关研究的表明，我们的视觉神经元是有层次感，低层的视觉神经元更加关注具体的局部细节（例如边缘，纹理等），而高层视觉神经元更加关注高层特征等（例如轮廓、空间关系等）。而低层神经元的实现就是通过局部连接的思想实现。

卷积神经网络

卷积神经网络三特性——权值共享

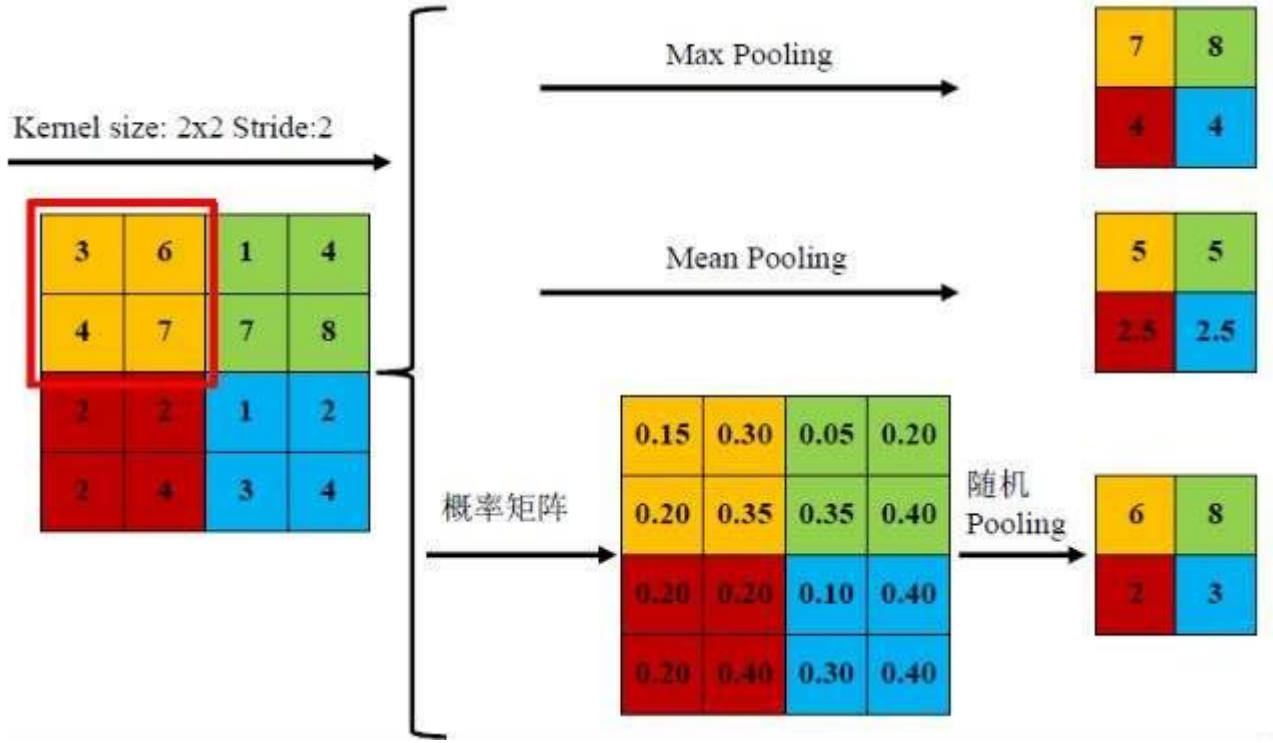


- 减少参数
- 提取特征

权值共享不仅仅是直接的参数量大大缩小，它也是拥有充分论证的物理意义的，在图像处理领域一次卷积操作就是一次特征提取

卷积神经网络

卷积神经网络三特性——池化采样

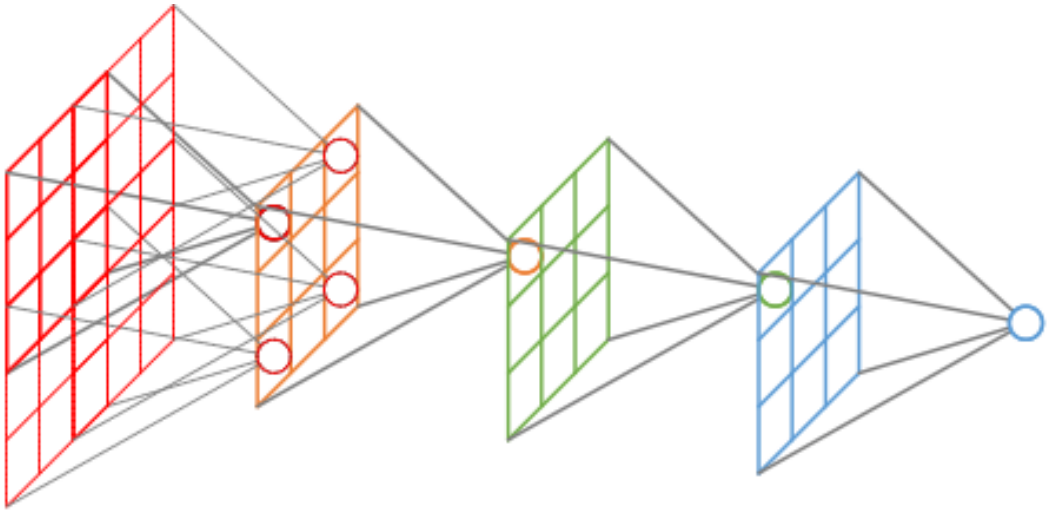


- 扩大感受野
- 层次特征

浅层关注的是局部的细节特征——纹理、颜色
高层关注的是更加抽象的全局特征——轮廓、类别

卷积神经网络

感受野



感受野是指一个神经元与输入层连接神经元组成的区域

- 感受野的计算

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

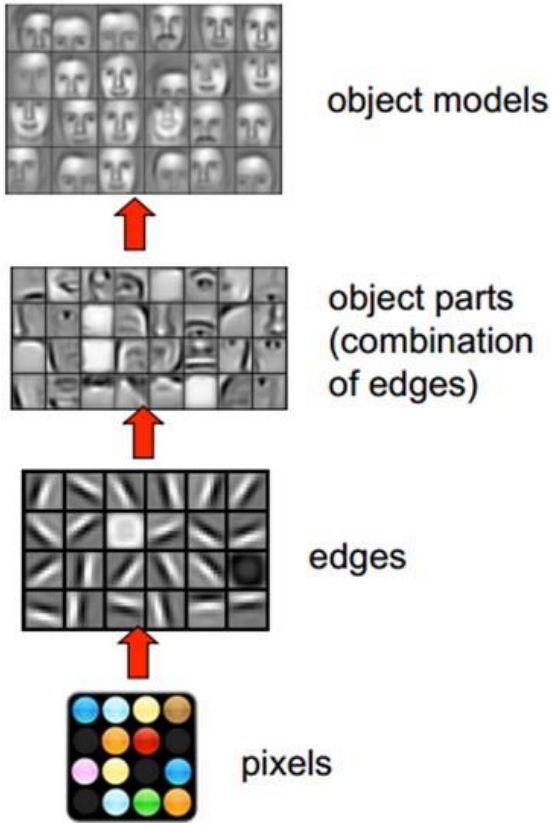
n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

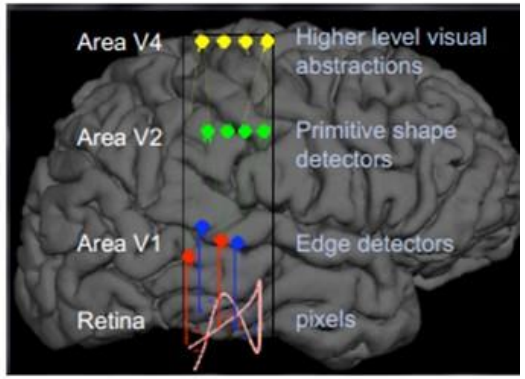
$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$
$$j_{out} = j_{in} * s$$
$$r_{out} = r_{in} + (k-1) * j_{in}$$
$$start_{out} = start_{in} + \left(\frac{\mathrm{k}-1}{2} - \mathrm{p}\right) * j_{in}$$

卷积神经网络

为什么要深度?

层数越多，表达能力越强





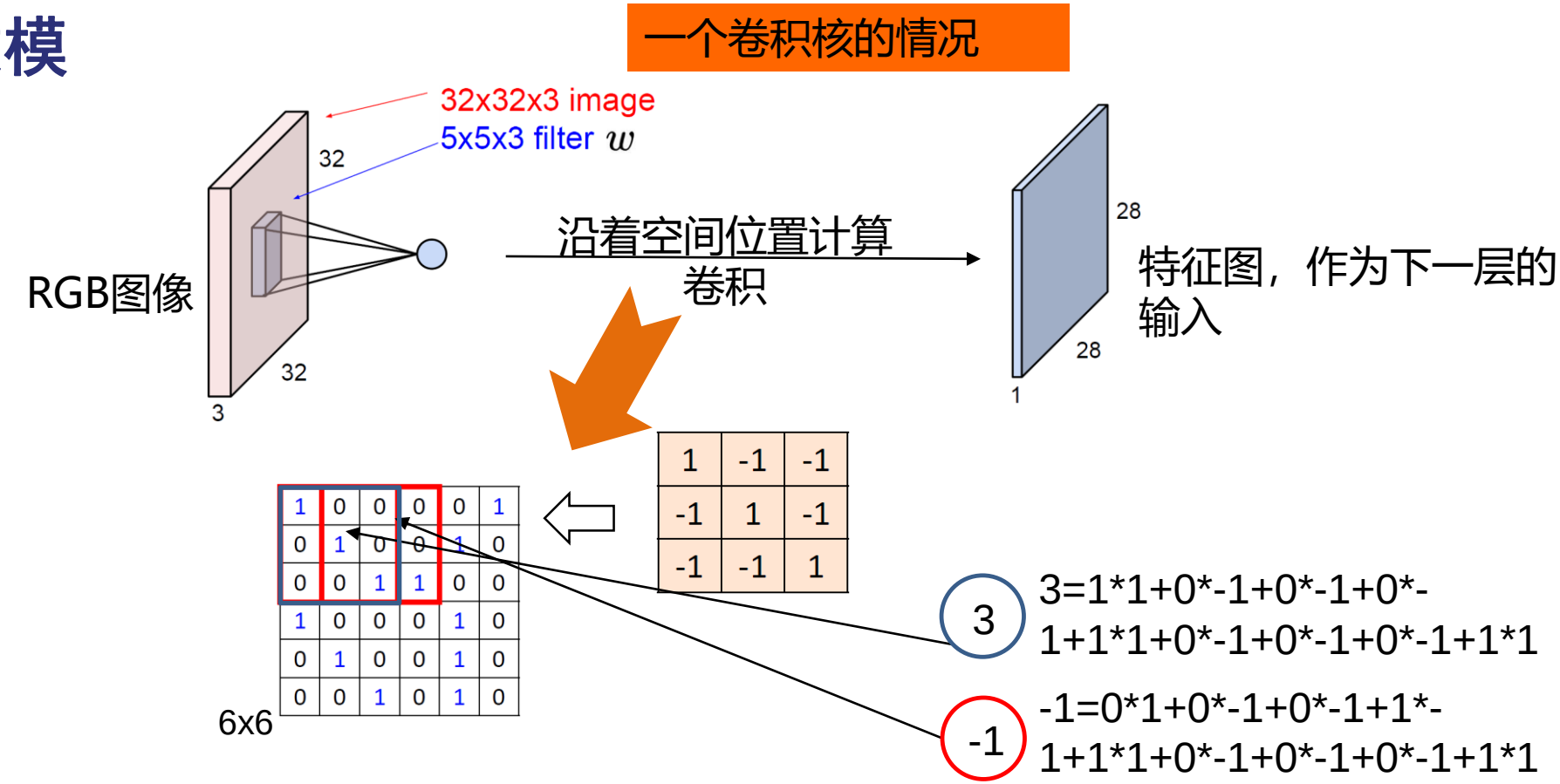
卷积神经网络

深度卷积网络的构成

- 卷积层(convolution layer)
- 池化层(pooling layer)
- 全连接层(fully connected layer)
- 辅助层(support layer)

卷积神经网络

卷积层建模

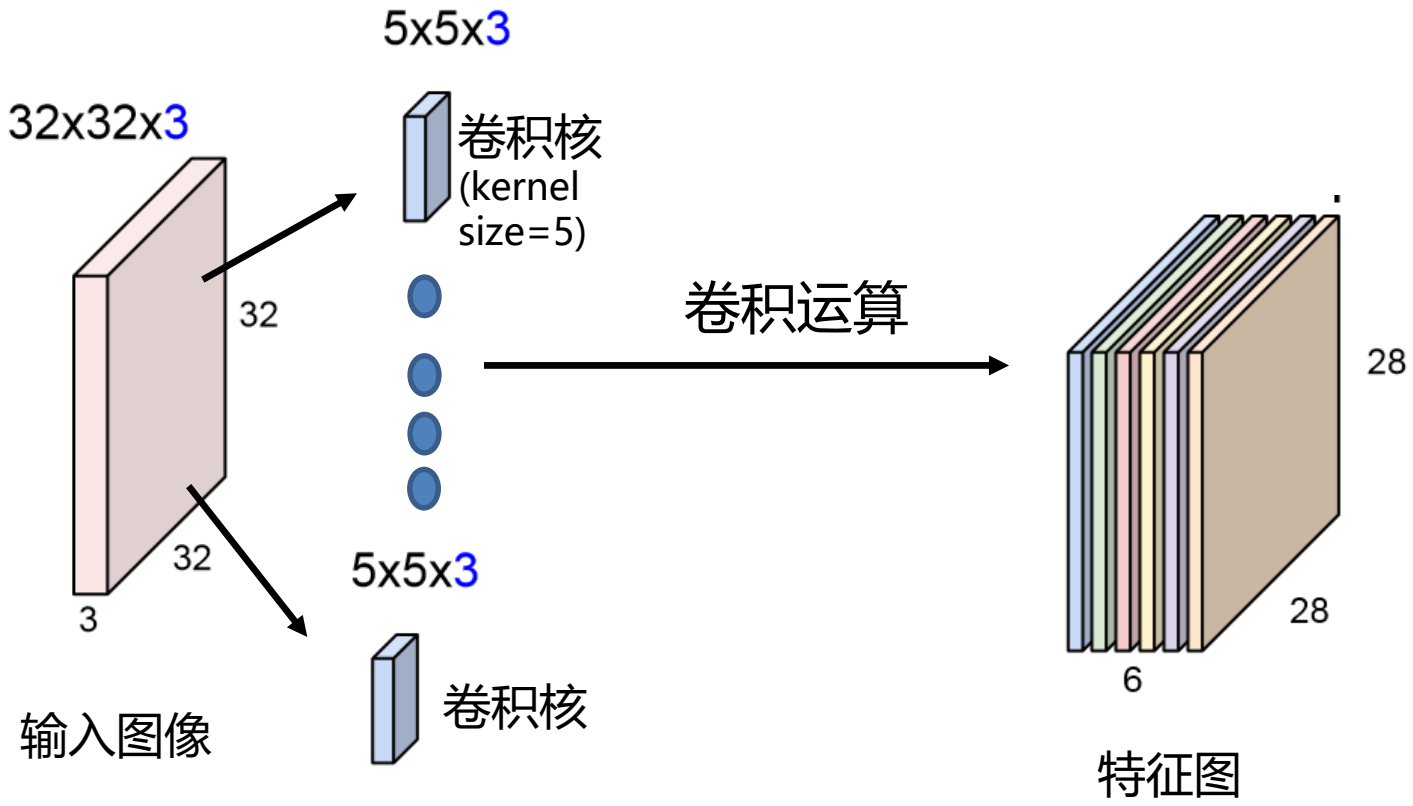


数学表达式：$M = \varphi(I * f + b)$

卷积神经网络

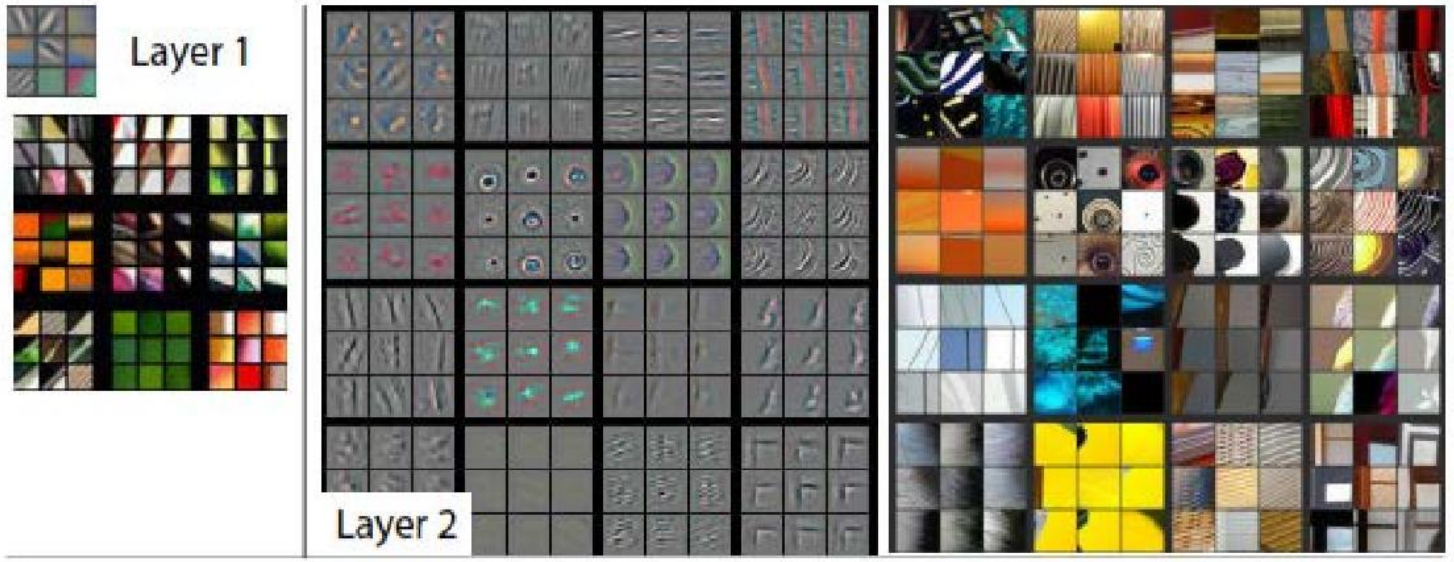
卷积层建模

多个卷积核的情况



卷积神经网络

卷积层可视化

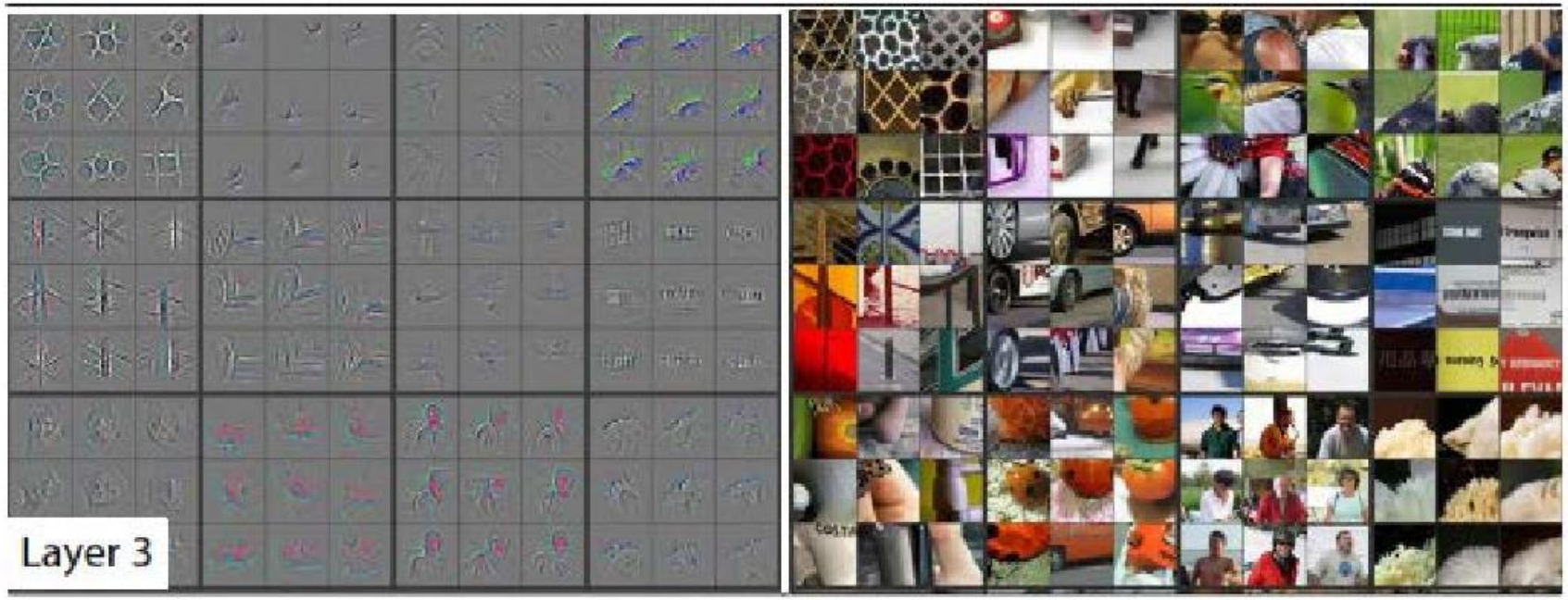


第一层和第二层：角点，边缘后者颜色信息

http://web.cs.ucdavis.edu/~yjlee/teaching/ecs289g-fall2016/jason.pdf

卷积神经网络

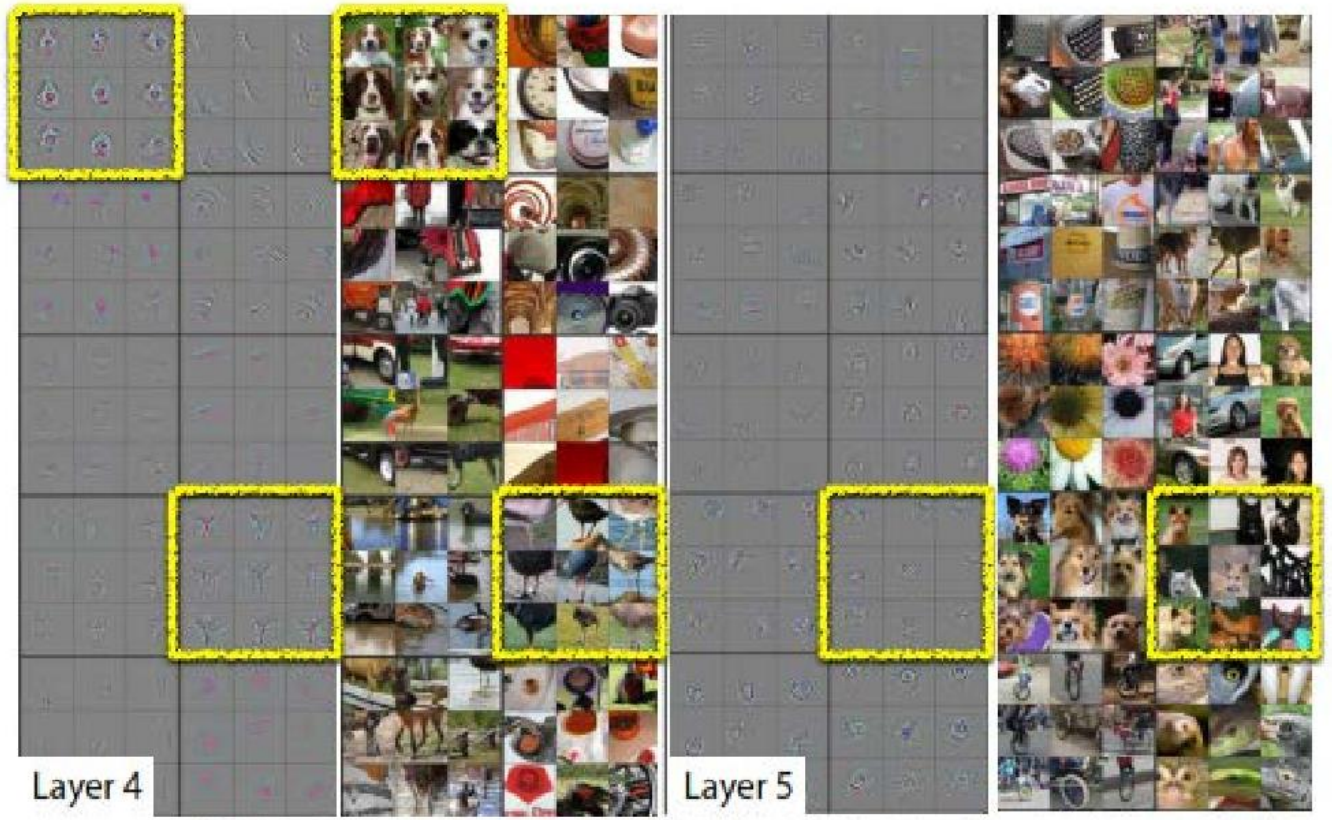
卷积层可视化



第三层：　　相似的纹理

卷积神经网络

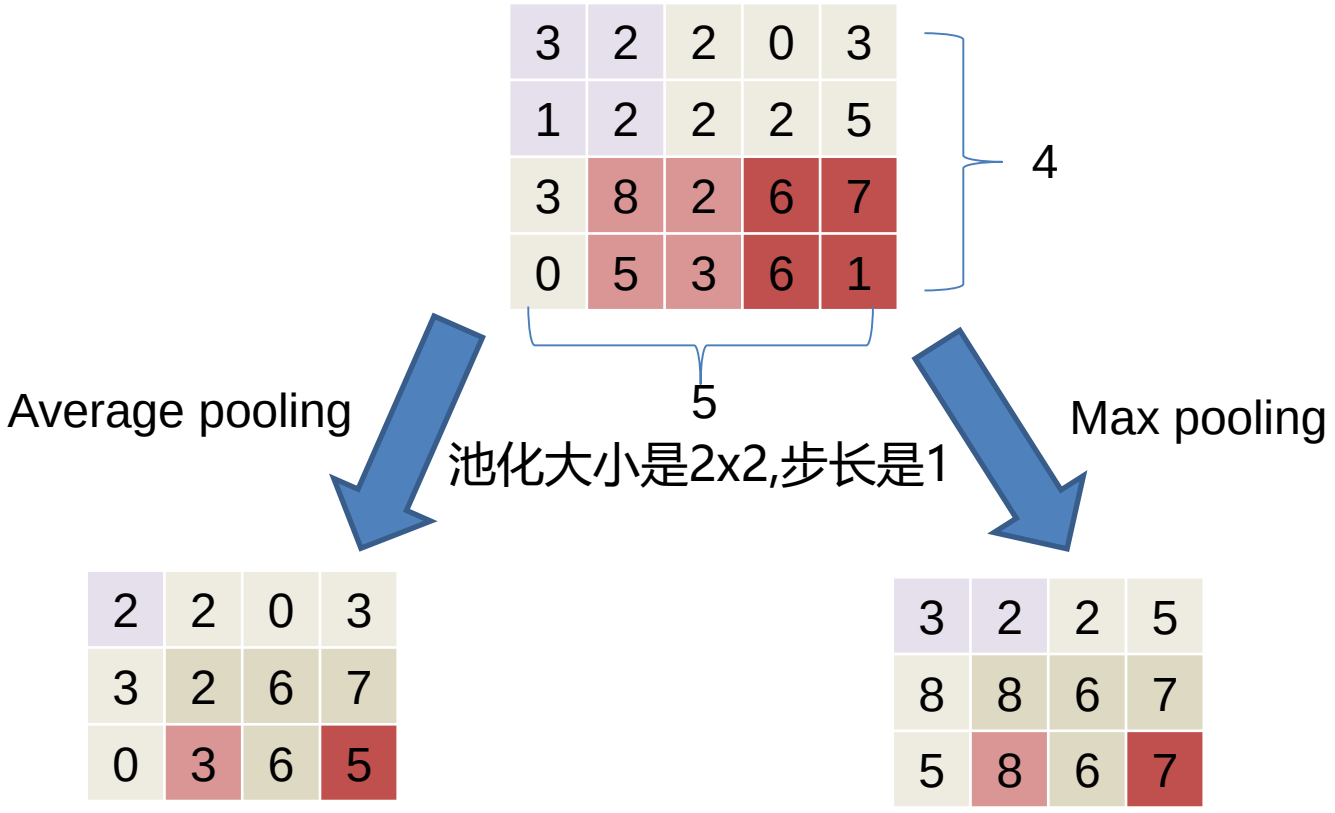
卷积层可视化



第四层,第五层卷积层可视化

卷积神经网络

池化层



池化层没有任何需要学习的参数

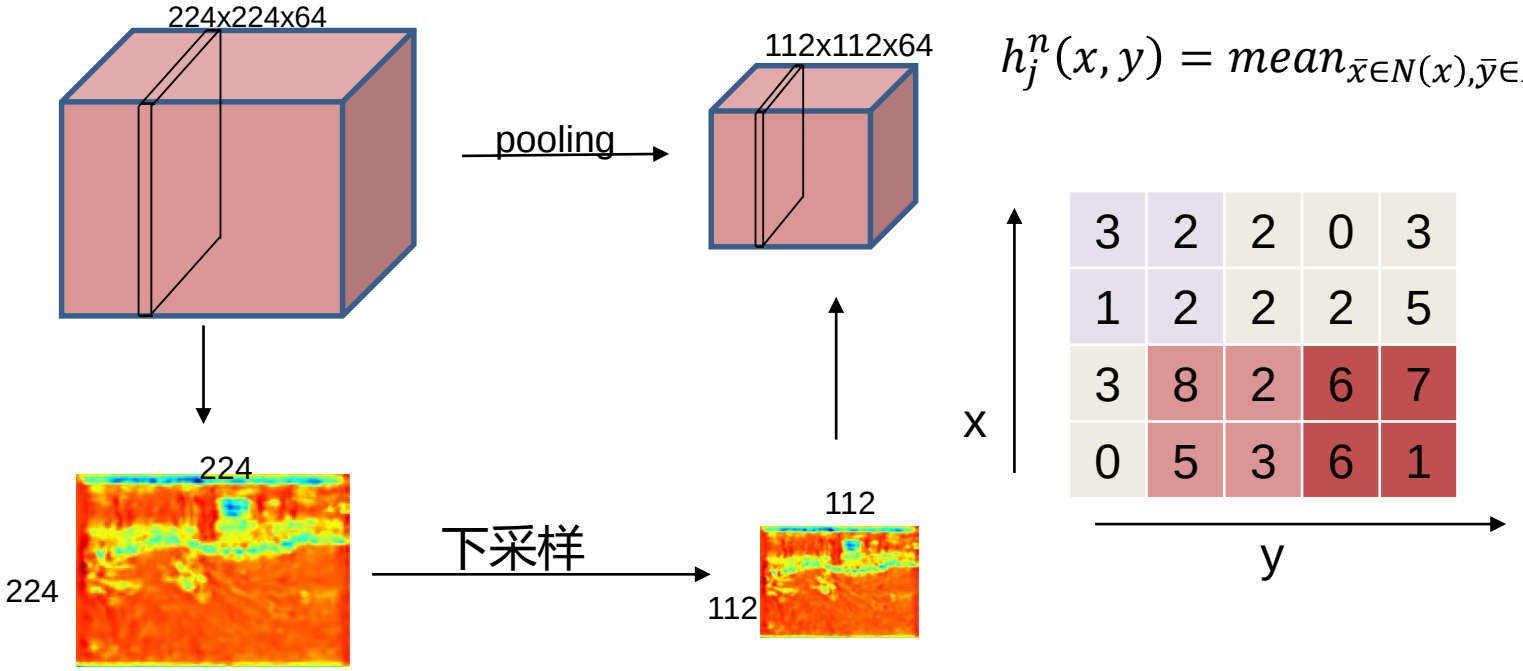
输入尺寸大小：$W1xH1xD1$

需要设置的超参数
池化范围：F
步长：S

池化后的尺寸：
$W_2 = (W_1 - F)/S + 1$
$H_2 = (H_1 - F)/S + 1$
$D_2 = D_1$

卷积神经网络

池化层



$$h_j^n(x,y) = mean_{\bar{x}\in N(x),\bar{y}\in N(x)} h_j^{n-1}(\bar{x},\bar{y})$$

Average pooling, max pooling比较：

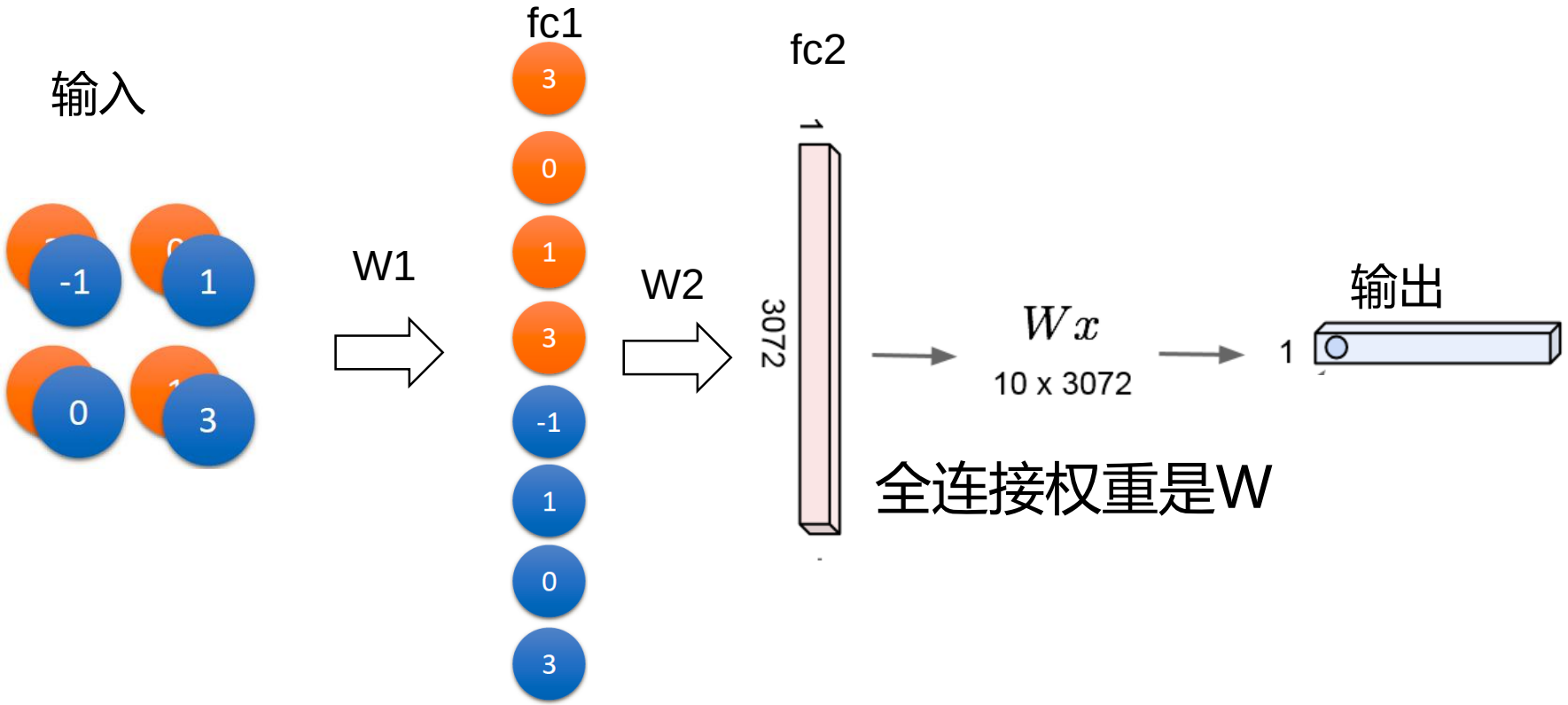
$$h_j^n(x,y) = max_{\bar{x}\in N(x),\bar{y}\in N(x)} h_j^{n-1}(\bar{x},\bar{y})$$

$$h_j^n(x,y) = mean_{\bar{x}\in N(x),\bar{y}\in N(x)} h_j^{n-1}(\bar{x},\bar{y})$$

卷积层之间一般是max pooling,最后输出层一般解average pooling
特征表达更加紧凑，同时具有位移不变性

卷积神经网络

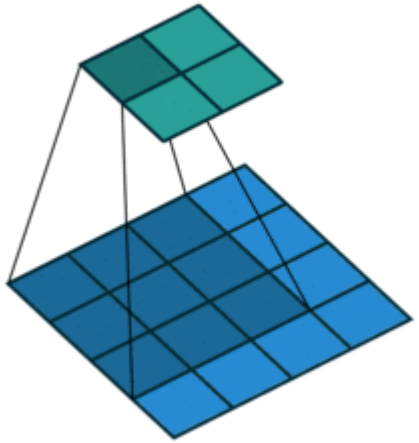
全连接层



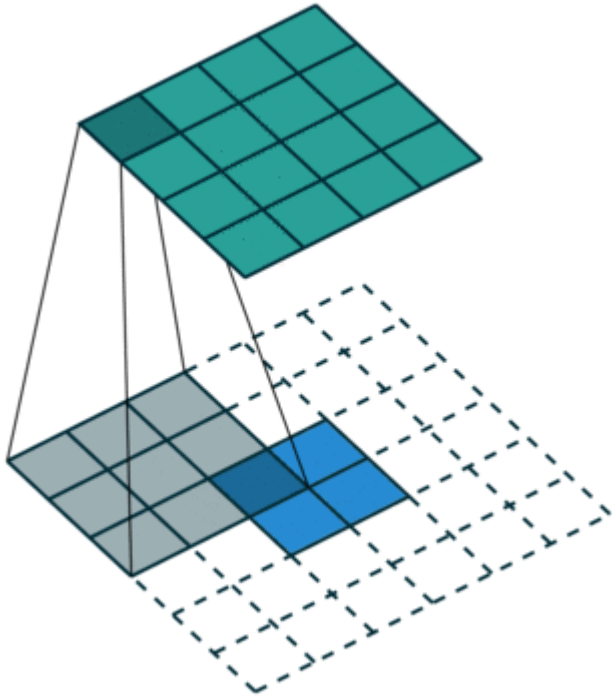
本质上也是卷积层，卷积核的大小和输入数据大小一致
比如W1 = (w*h)*N_{fc1}，N_{fc1}表示第一个全连接层节点个数
全连接层之间，可以看做是1*1的卷积
W2= N_{fc1} * （1*1） * N_{fc2}

卷积神经网络

辅助层：反卷积层



卷积

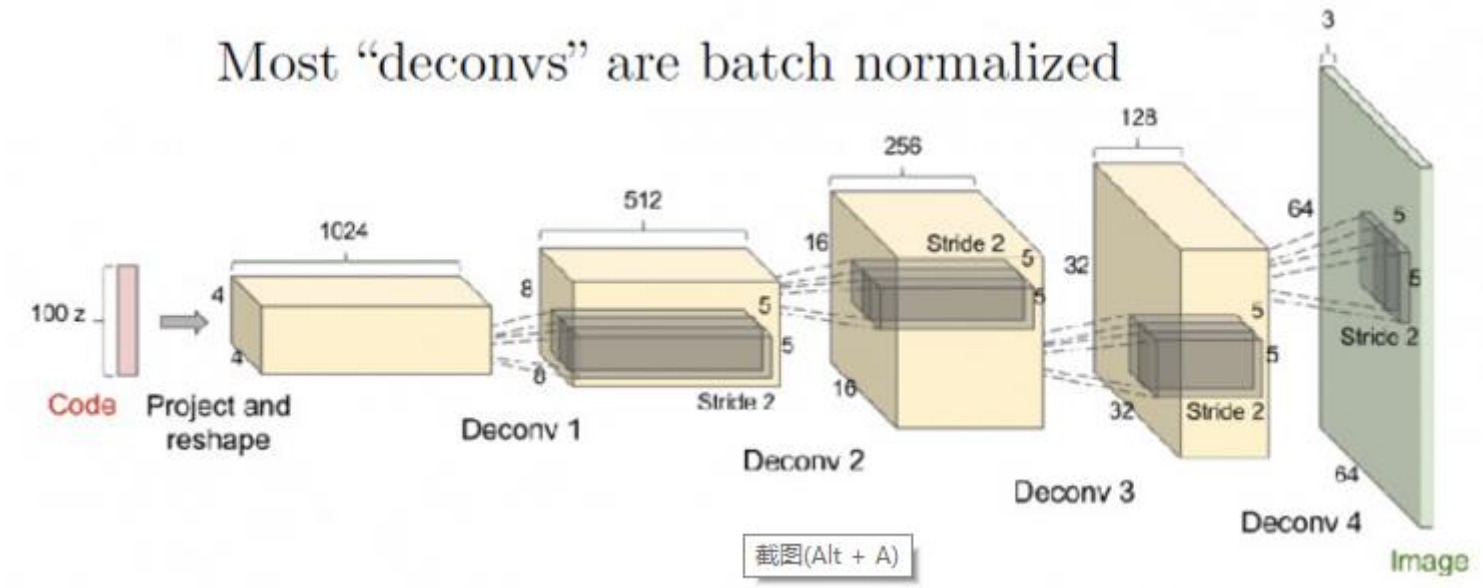


反卷积

可学习的上采样层，在图像分割，图像生成中广泛应用

卷积神经网络

辅助层：反卷积层



Mirza M, Osindero S. Conditional Generative Adversarial Nets[J]. arXiv: Learning, 2014.

卷积神经网络

辅助层：Batch Normalization 层

输入数据归一化：

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}$$

$$\gamma^{(k)^-} = \sqrt{Var[x^{(k)}]}$$

前向传导过程：

$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ //mini-batch mean

$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$ //mini-batch

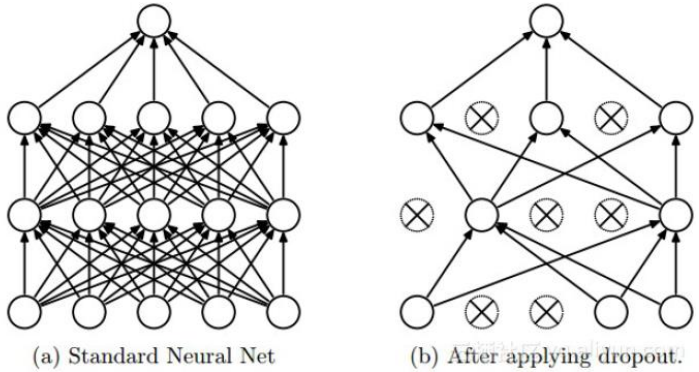
$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$ variance

$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$ //normalize

Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[J]. international conference on machine learning, 2015: 448-456//scale and shift

卷积神经网络

辅助层：Dropout层



在训练阶段中，将假设的投影作为修改的激活函数a(h), D是伯努利分布变量：

输出：$f(h) = D \odot a(h)$

激活值进行缩放(rescle)：$f(k,p) = \begin{cases} p, & k = 1 \\ 1-p, & k = 0 \end{cases}$

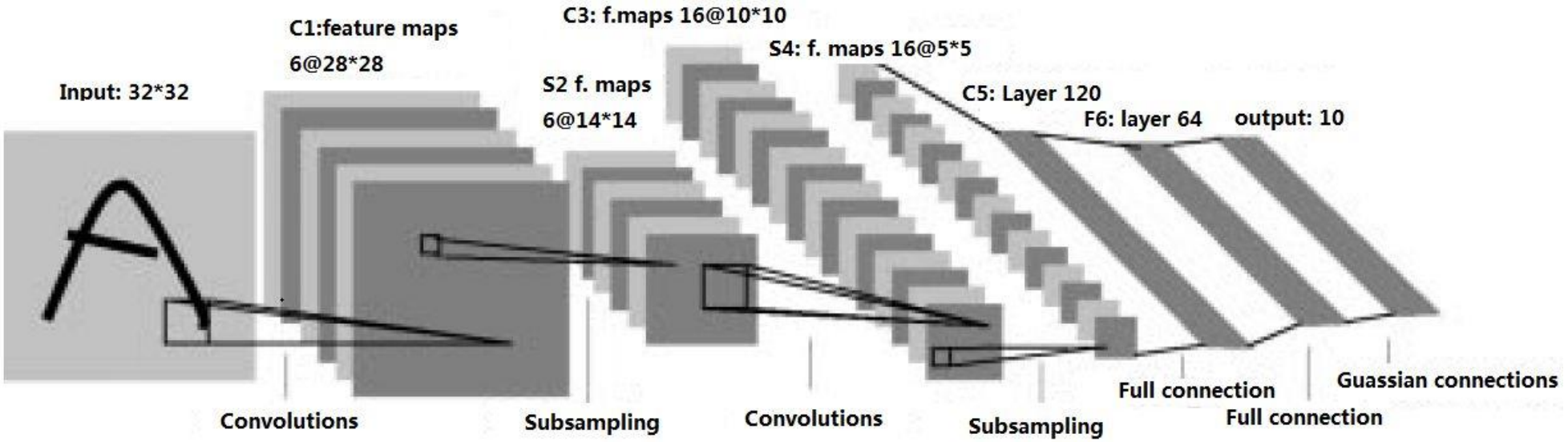
训练阶段：$\frac{1}{1-p} = \frac{1}{p}$, $O_i = X_i a\left(\sum_{k-1}^{d} w_k x_k + b\right) = \begin{cases} a(\sum_{k-1}^{d} w_k x_k + b), X_i < 0 \\ 0, \ X_i = 0 \end{cases}$

测试阶段：$O_i = \frac{1}{q} X_i a\left(\sum_{k-1}^{d_i} w_k x_k + b\right)$

https://yq.aliyun.com/articles/68901

卷积神经网络

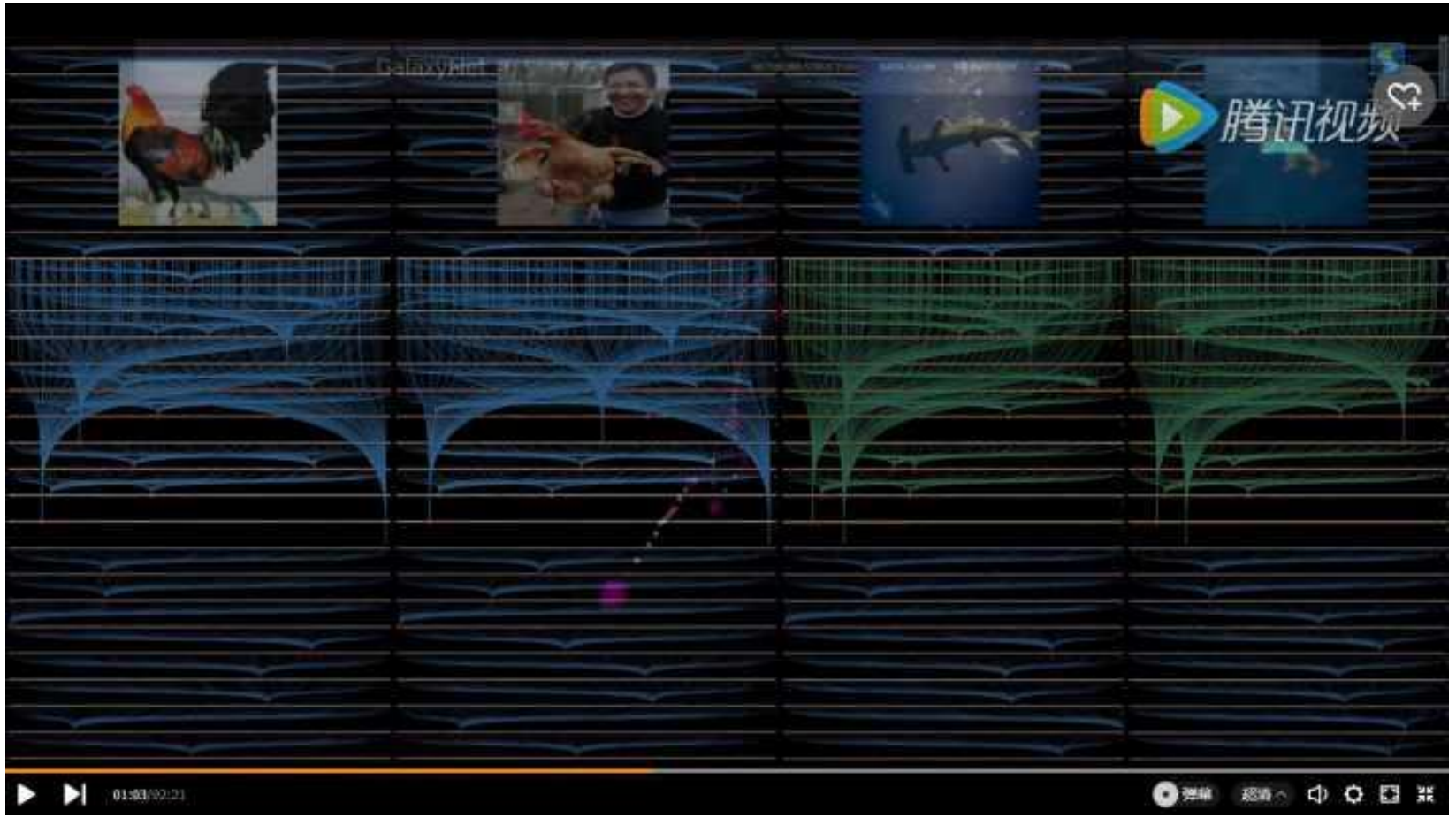
卷积神经网络示例——LeNet




```
model = Sequential()
model.add(Conv2D(6, kernel_size=(5, 5), activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, (5, 5), activation='relu’))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu’))
model.add(Dense(num_classes, activation='softmax'))
```

卷积神经网络

卷积神经网络Demo



AI300学院

卷积神经网络

深度学习好像很厉害，我要怎样自己实现一个CNN呢?

深度学习框架

深度学习平台的成熟程度（GitHub Stars的数量，2018/9/30）

- Tensorflow: 110806
- Keras: 33978
- Caffe: 25702
- Pytorch:19096
- Mxnet: 15303
- CNTK: 15171
- Tflearn:8570
- Theano:8496

深度学习框架

深度学习平台对比

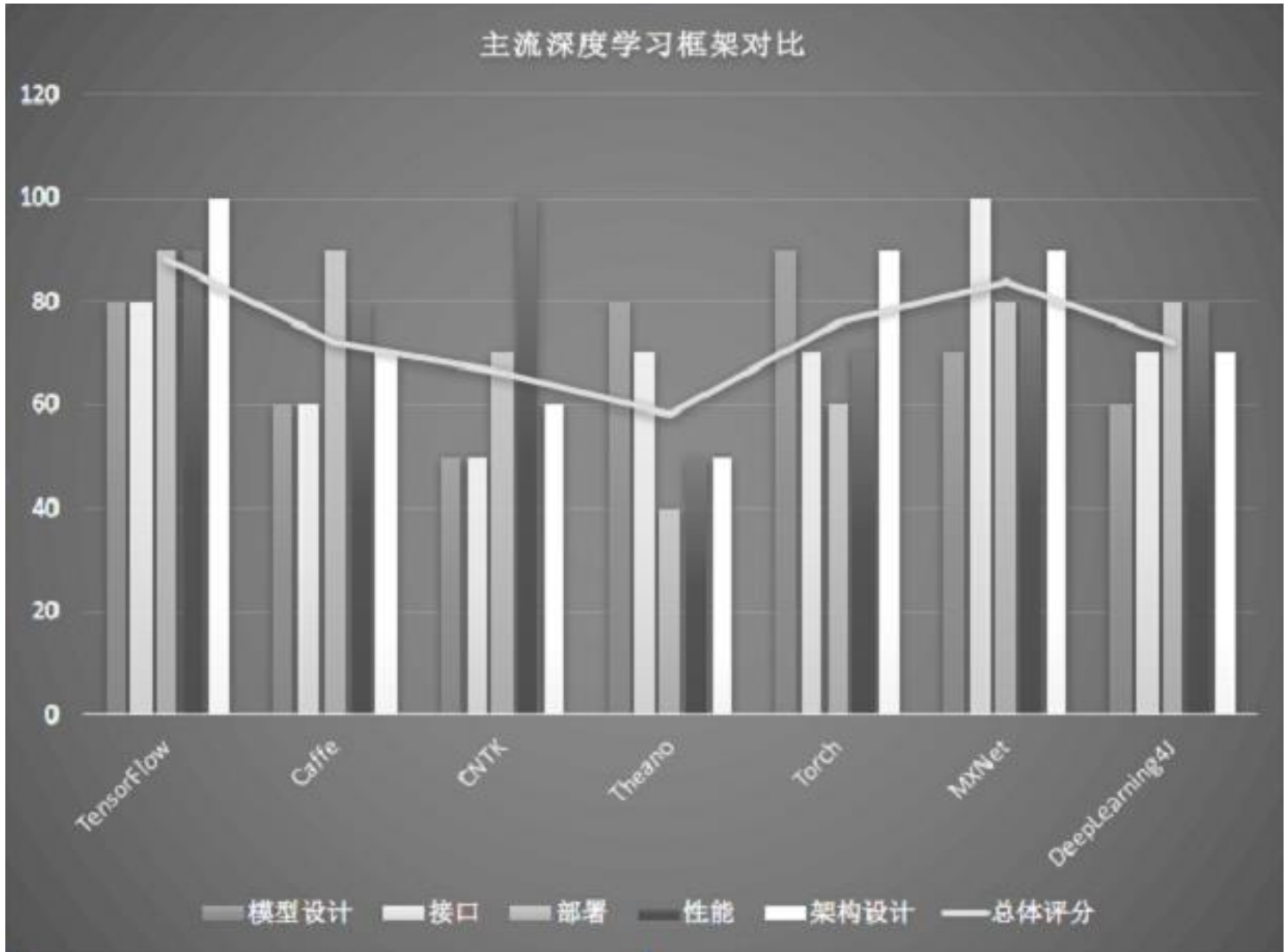
Design Choice	Torch.nn	Theano	Caffe	Chainer	MXNet	Tensor-Flow	PyTorch
NN definition	Script (Lua)	Script* (Python)	Data (protobuf)	Script (Python)	Script (many)	Script (Python)	Script (Python)
Backprop	Through graph	Extended graph	Through graph	Through graph	Extended graph	Extended graph	Through graph
Parameters	Hidden in operators	Separate nodes	Hidden in operators	Separate nodes	Separate nodes	Separate nodes	Separate nodes
Update formula	Outside of graphs	Part of graphs	Outside of graphs	Outside of graphs	Outside of graphs	Part of graphs	Outside of graphs
Graph construction	Static	Static	Static	Dynamic	Static	Static	Dynamic
Graph Optimization	-	Supported	-	-	-	Supported	-
Parallel computation	Multi GPU*	Multi GPU	Multi GPU*	Multi GPU**	Multi node Multi GPU	Multi node Multi GPU	Multi GPU**

表格数据来源于2017/2/5

AI300学院

深度学习框架

深度学习平台对比



AI300学院

深度学习框架

深度学习平台简评

- Tensorflow：谷歌的框架，受众最多，目前支持动态库eager
- Keras:适合新手入门，自定义差，速度最慢，高级阶段基本会逐渐弃用
- Caffe: 工程部署上有点挺多，但是用的protobuf，非脚本风格
- Pytorch:目前的新宠，动态图适合科研调试，潜力很大，但bug还很多
- Mxnet: 性能还不错，但目前还比较小众，等部分功能完善
- CNTK:微软的框架，有比较好的分布式性能，适合工业生产
- Tflearn:TensorFlow的高层封装
- Theano:已经停止维护

AI300学院

课后思考

1. 自行推导神经网络的反向传播
2. 为什么卷积层的导数是翻转180°卷积
3. 用Pytorch实现LeNet

AI300学院

欢迎关注AI300学院