Python人工智能

讲师：覃秉丰

交叉熵

二次代价函数

二次代价函数：

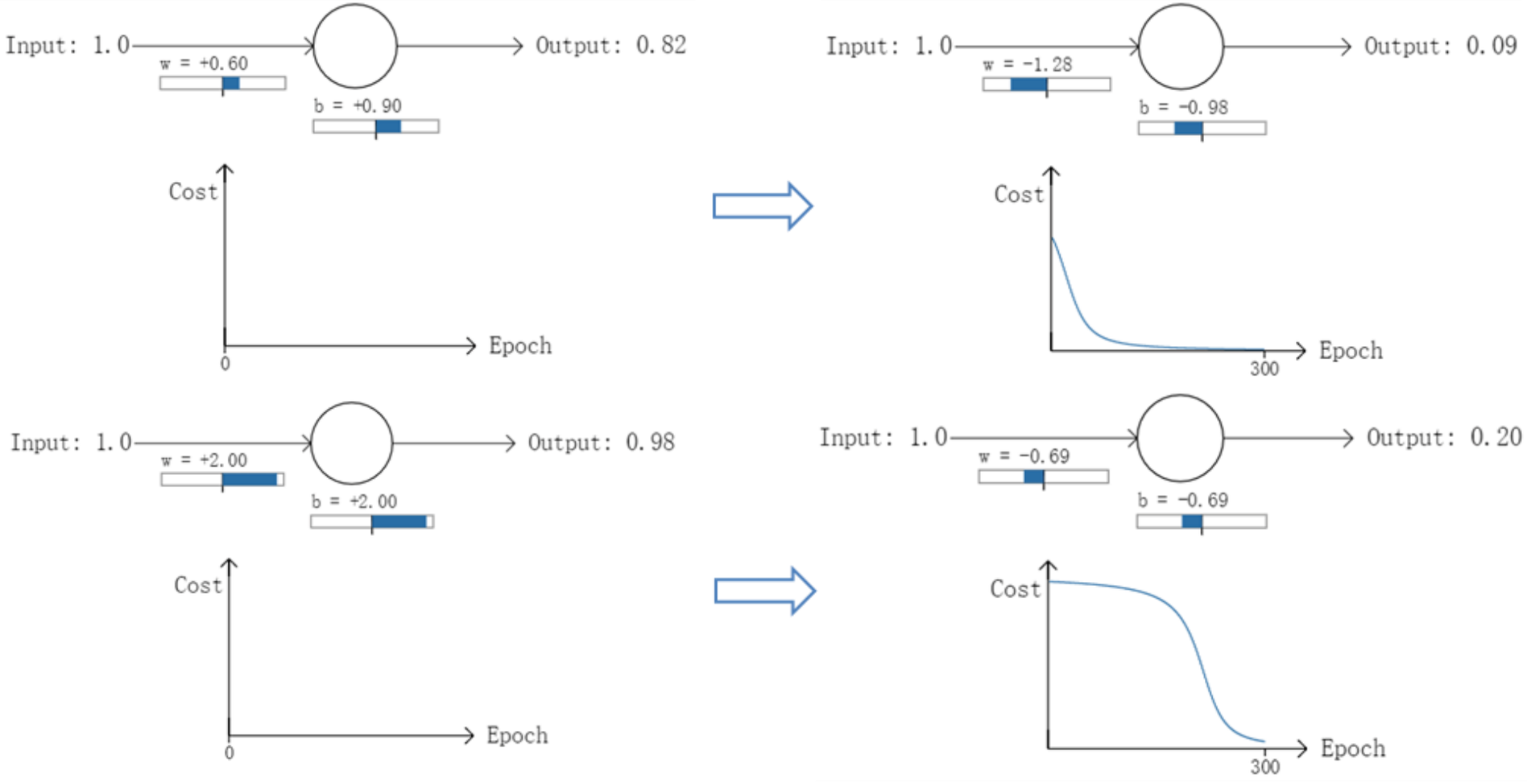
$$E = \frac{1}{2}(t-y)^2$$

$$\frac{\partial E}{\partial w} = (y-t)\,f'(z)\,x \qquad z = WX$$

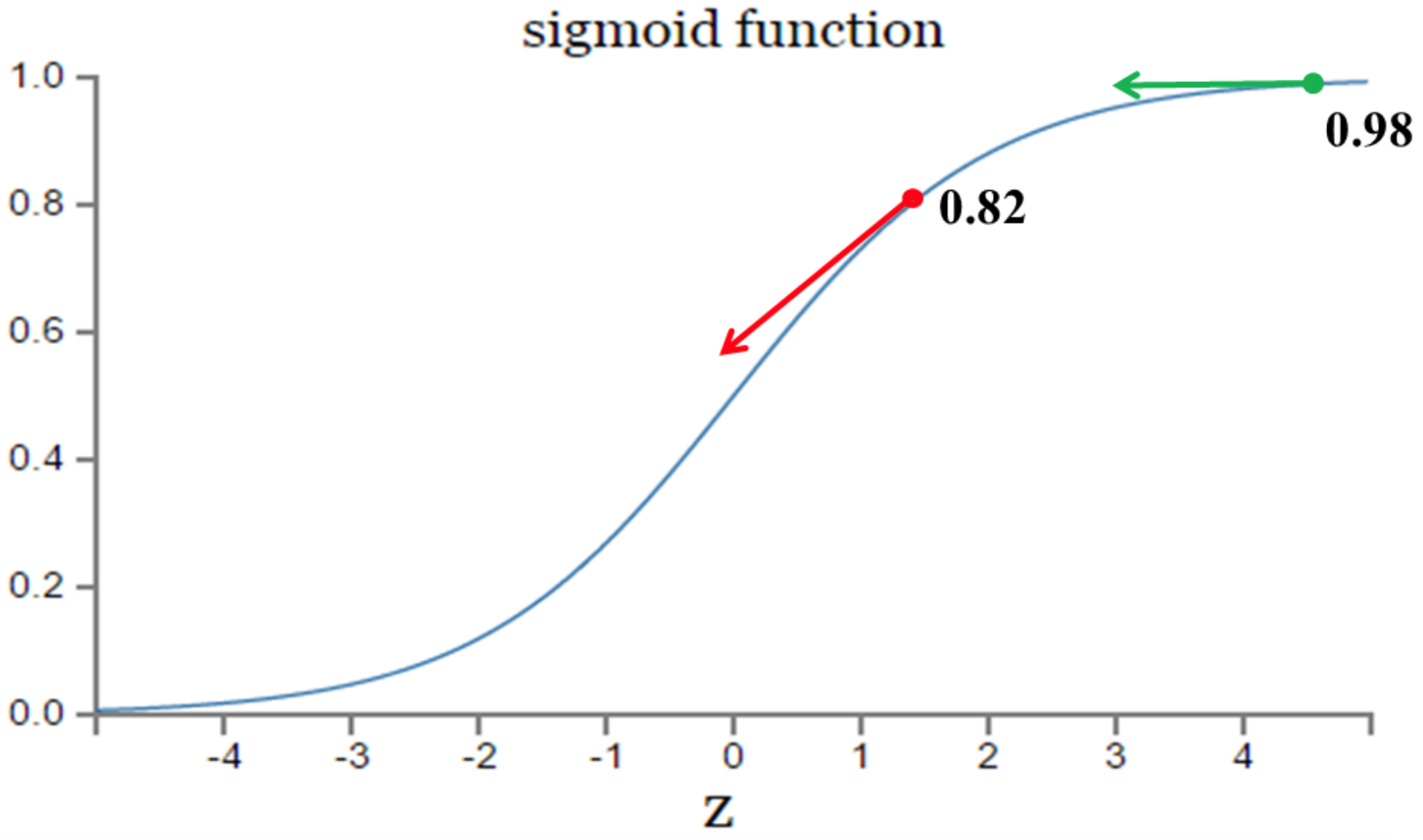
激活函数的梯度f’ (z)越大，w的大小调整得越快，训练收敛得就越快。激活函数的梯度f’ (z)越小，w的大小调整得越慢，训练收敛得就越慢。

二次代价函数

以一个二分类问题为例，进行两组实验。输入同一个样本数据x=1.0，该样本对应的分类为y=0，使用sigmoid激活函数。



二次代价函数



交叉熵

换一个思路，我们不改变激活函数，而是改变代价函数，该用交叉熵代价函数：

$$E = -\left(t \ln y + (1-t) \ln (1-y)\right)$$

$$\begin{aligned}
\frac{\partial E}{\partial w} &= -\left(\frac{t}{f(z)} - \frac{1-t}{1-f(z)}\right) \frac{\partial f}{\partial w} \\
&= -\left(\frac{t}{f(z)} - \frac{1-t}{1-f(z)}\right) f'(z)\, x \\
&= \frac{f'(z)\, x}{f(z)\,(1-f(z))} (f(z) - t) \\
&= x\,(f(z) - t)
\end{aligned}$$

对于sigmoid函数：

$$f'(z) = f(z)\,(1-f(z))$$

对数似然代价函数(log-likelihood cost)

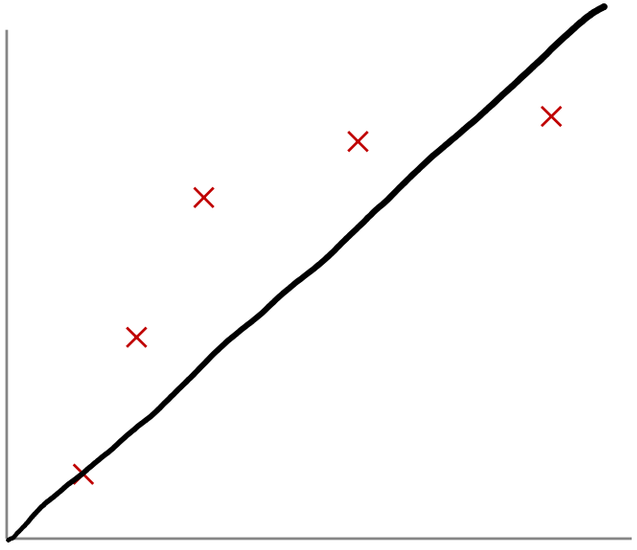
- 对数释然函数常用来作为softmax回归的代价函数。
- 如果输出层神经元是sigmoid函数，可以采用交叉熵代价函数。
- 深度学习中更普遍的做法是将softmax作为最后一层，此时常用的代价函数是对数释然代价函数。
- 对数似然代价函数与softmax的组合和交叉熵与sigmoid函数的组合非常相似。对数释然代价函数在二分类时可以化简为交叉熵代价函数的形式。
- 在tensorflow中使用：

tf.nn.sigmoid_cross_entropy_with_logits()来表示跟sigmoid搭配使用的交叉熵
tf.nn.softmax_cross_entropy_with_logits()来表示跟softmax搭配使用的交叉熵

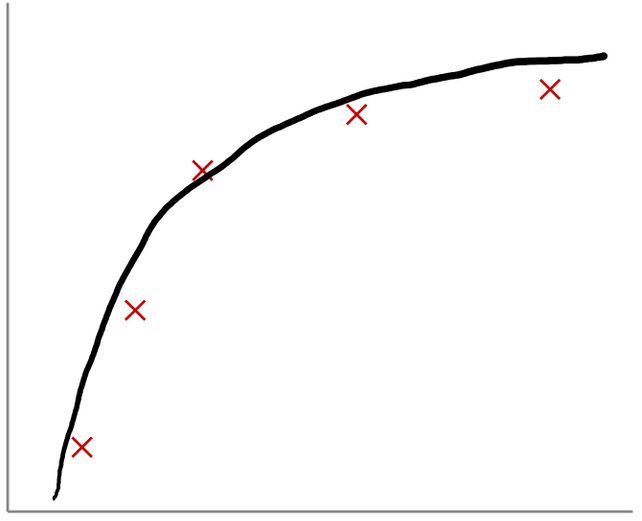
交叉熵(cross-entropy)程序

- 1.交叉熵

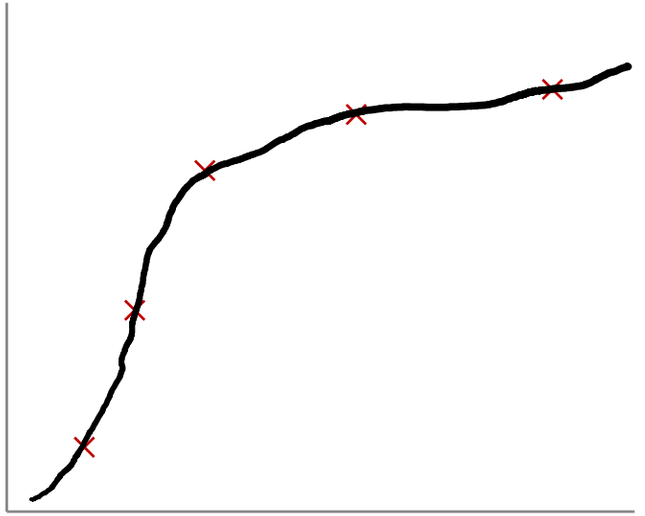
拟合



欠拟合(Underfitting)

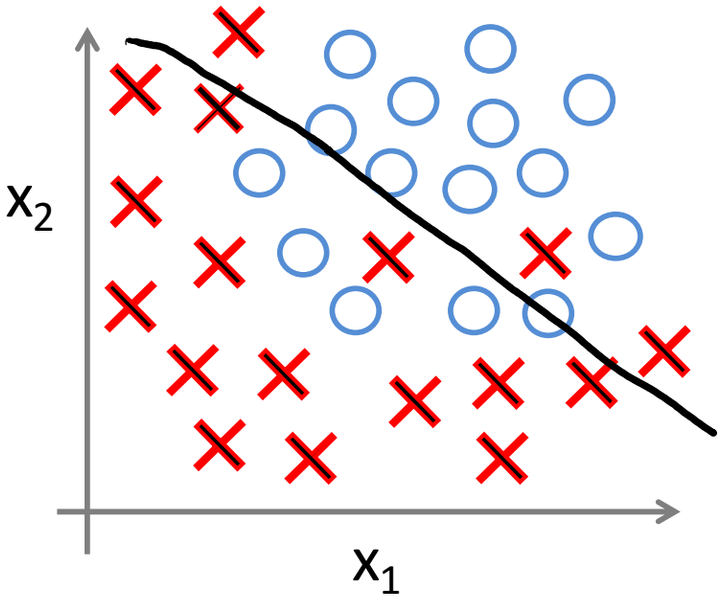


正确拟合(Just right)

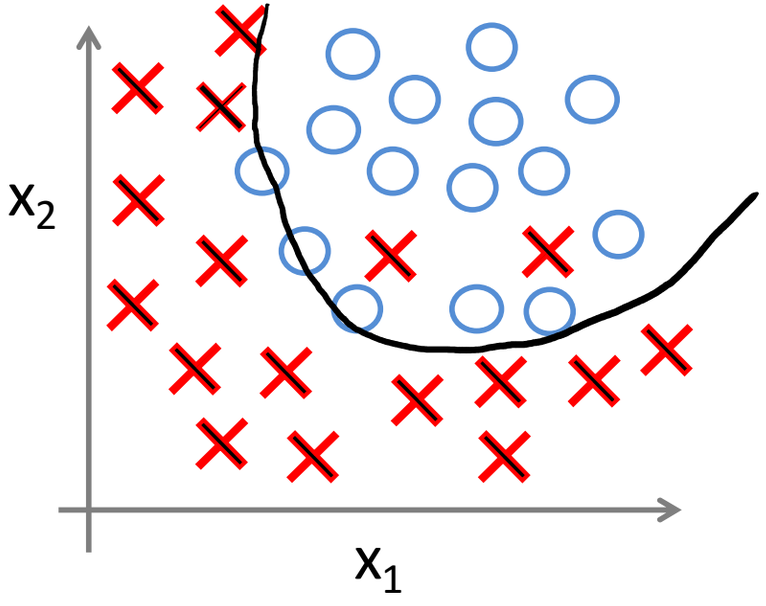


过拟合(Overfitting)

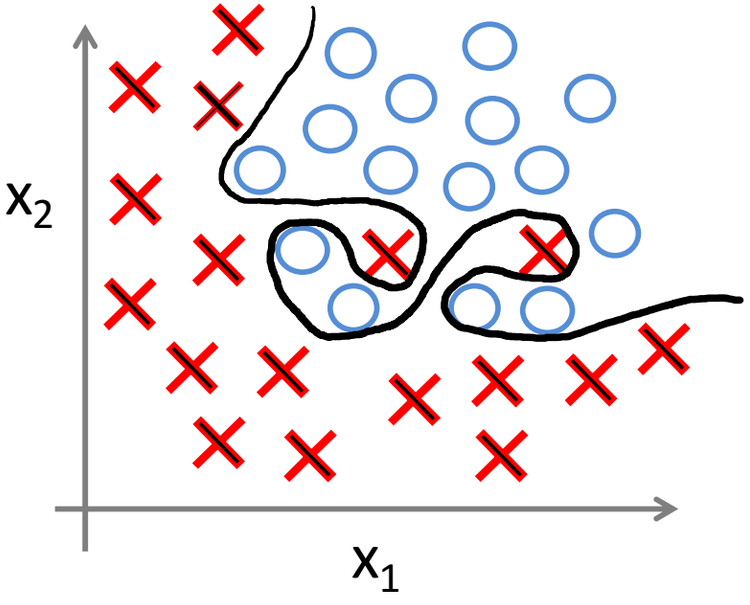
拟合



欠拟合(Underfitting)

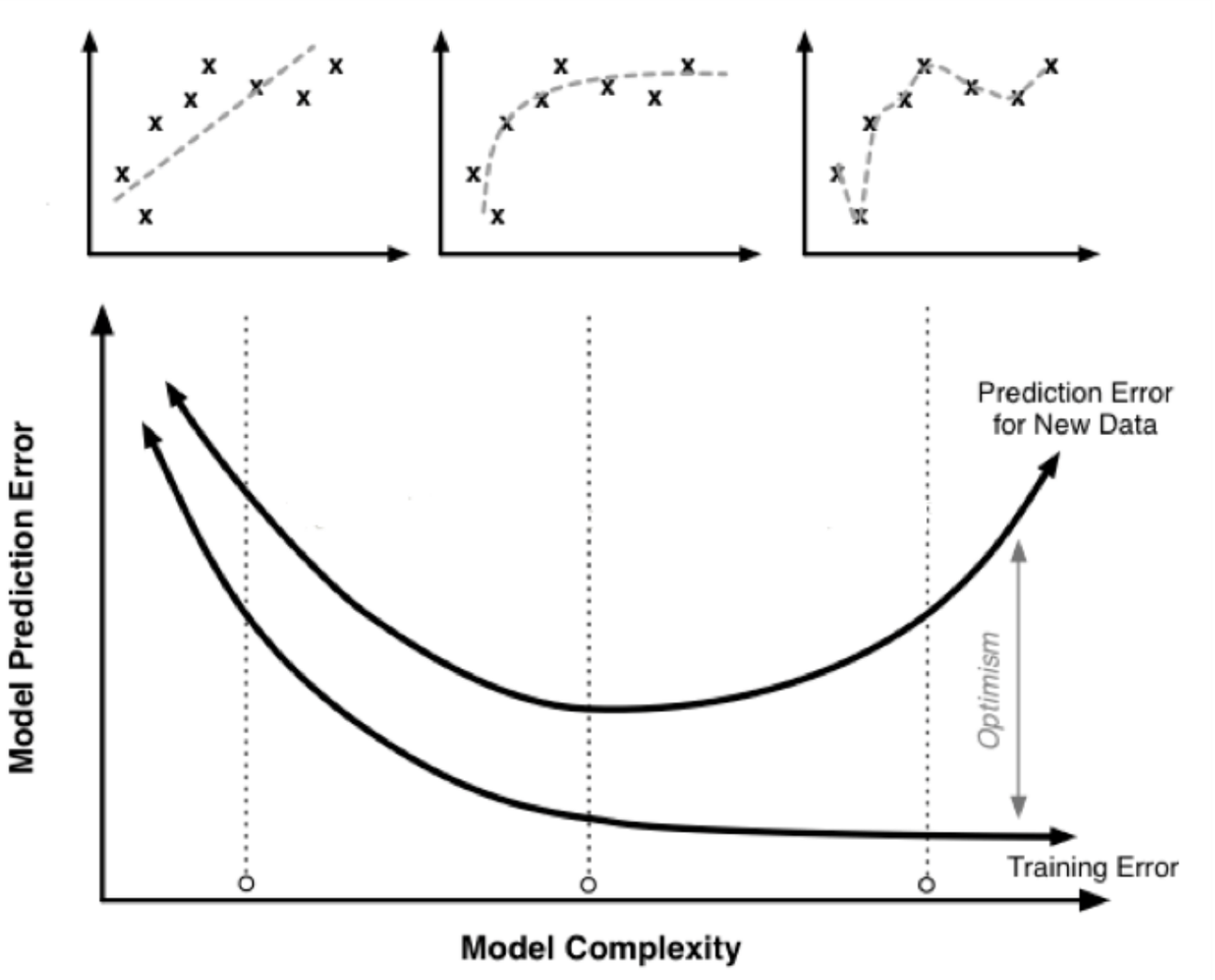


正确拟合(Just right)



过拟合(Overfitting)

过拟合导致测试误差变大



防止过拟合

增大数据集

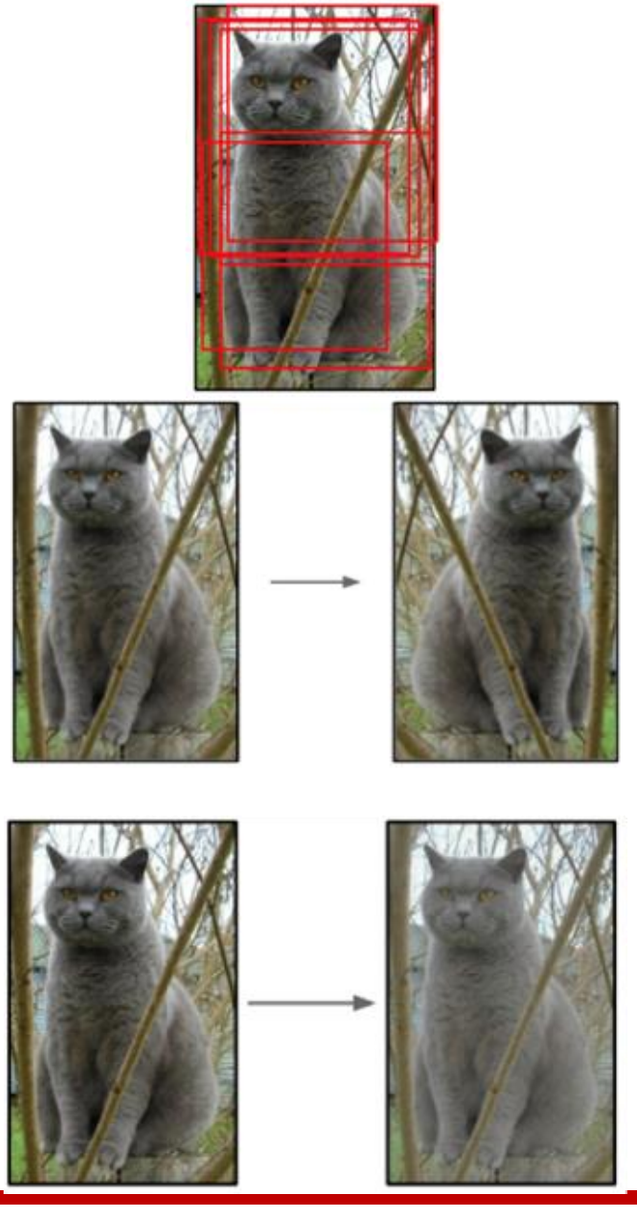
数据挖掘领域流行着这样一句话，“有时候拥有更多的数据胜过一个好的模型”。一般来说更多的数据参与训练，训练得到的模型就越好。如果数据太少，而我们构建的神经网络又太复杂的话就比较容易产生过拟合的现象。

增大图片数据集

1.随机裁剪

2.水平翻转

3.光照颜色抖动

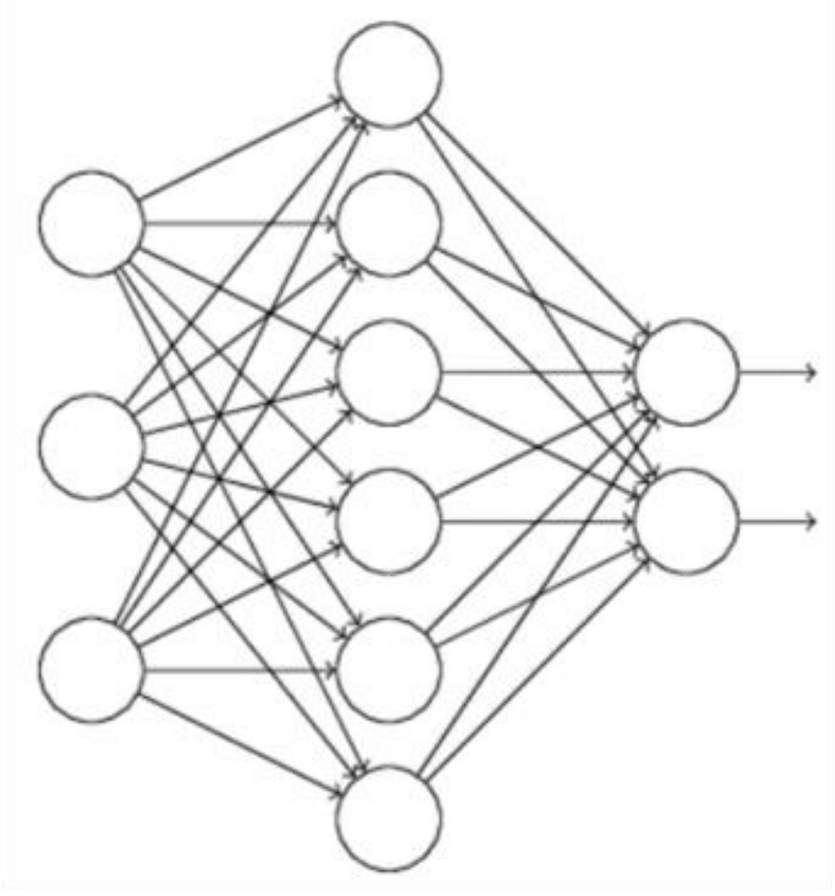


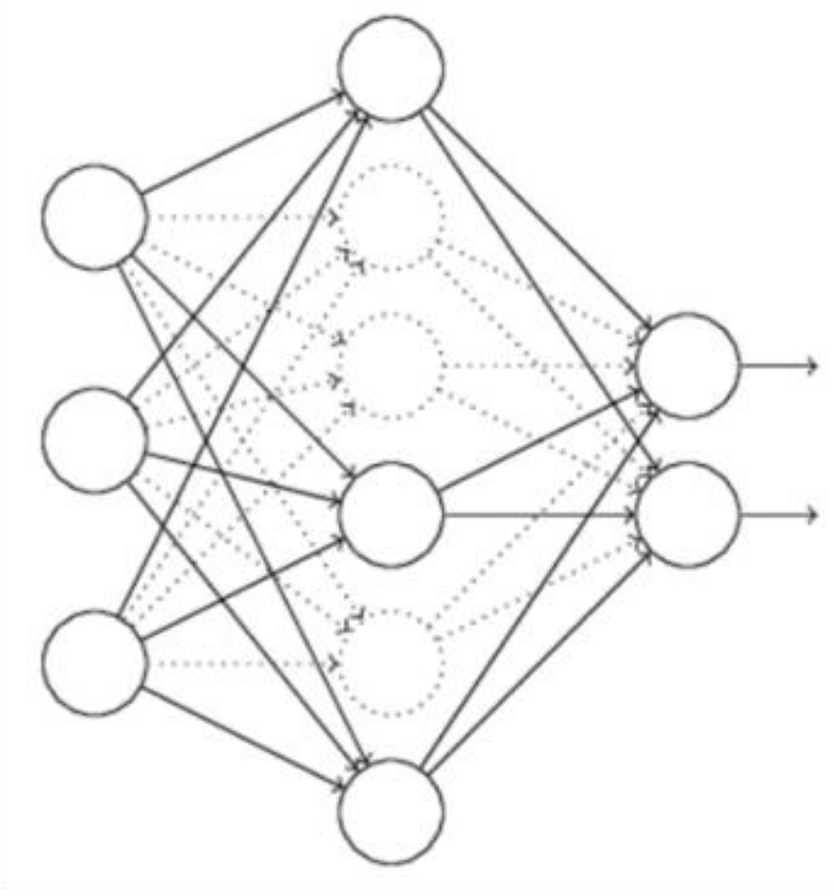
Early stopping

•在训练模型的时候，我们往往会设置一个比较大的迭代次数。Early stopping便是一种提前结束训练的策略用来防止过拟合。

•一般的做法是记录到目前为止最好的validation accuracy，当连续10个Epoch没有达到最佳accuracy时，则可以认为accuracy不再提高了。此时便可以停止迭代了（Early Stopping）。

Dropout





正则化项

C0代表原始的代价函数，n代表样本的个数，λ就是正则项系数，权衡正则项与C0项的比重。

L1正则化：

$$C = C_0 + \frac{\lambda}{n}\sum_w |w|$$

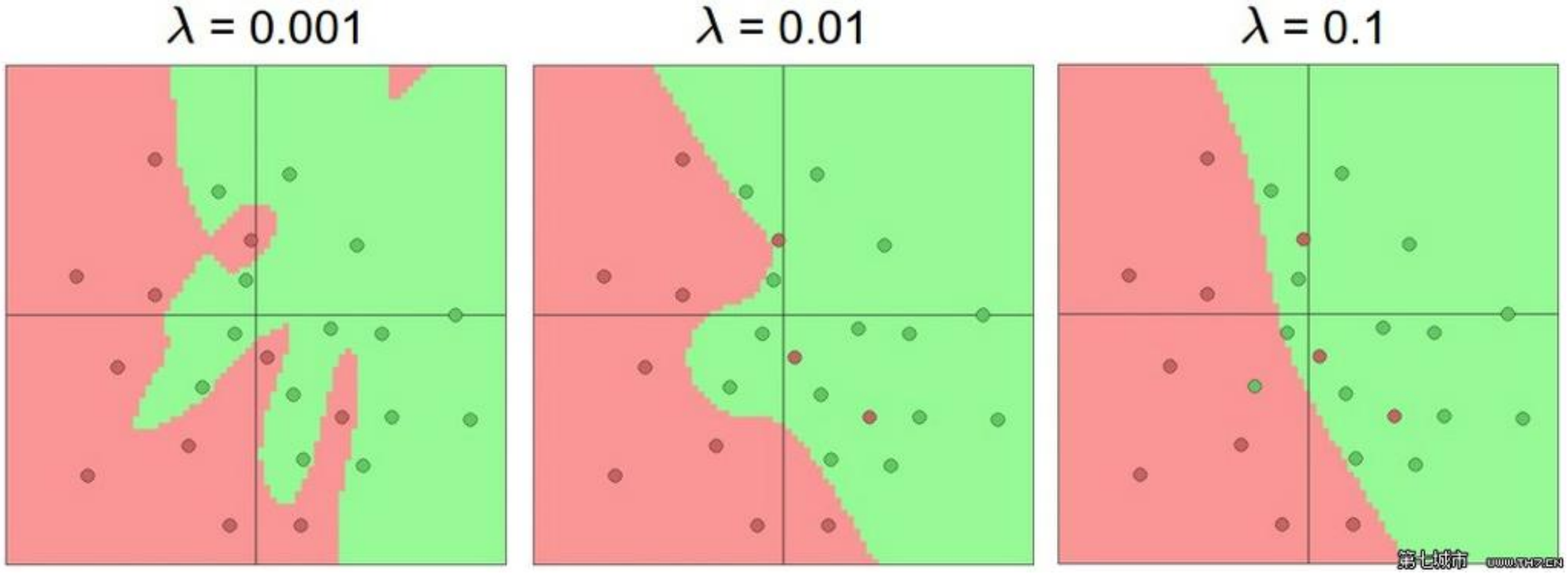
L1正则化可以达到模型参数稀疏化的效果

L2正则化：

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2$$

L2正则化可以使得模型的权值衰减，使模型参数值都接近于0。

正则化项



Dropout和正则化程序

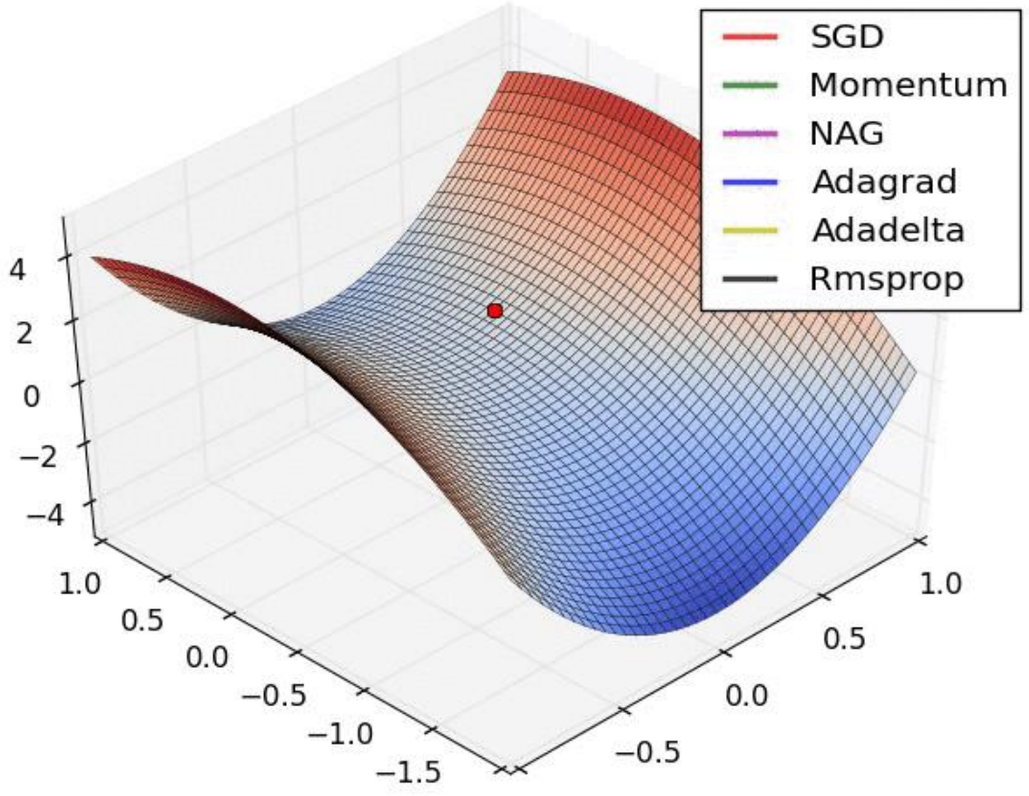
- 2.Dropout
- 3.正则化

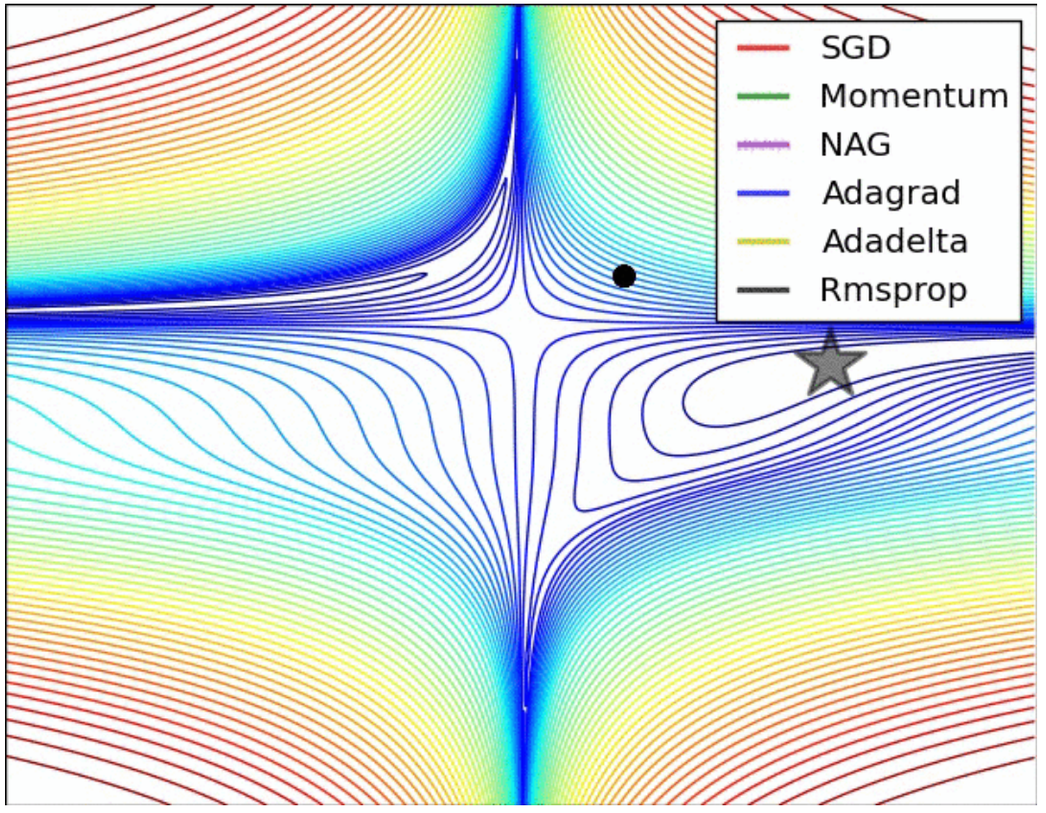
优化器

Optimizer

tf.train.GradientDescentOptimizer
tf.train.AdadeltaOptimizer
tf.train.AdagradOptimizer
tf.train.AdagradDAOptimizer
tf.train.MomentumOptimizer
tf.train.AdamOptimizer
tf.train.FtrlOptimizer
tf.train.ProximalGradientDescentOptimizer
tf.train.ProximalAdagradOptimizer
tf.train.RMSPropOptimizer

优化器


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop

优化器程序

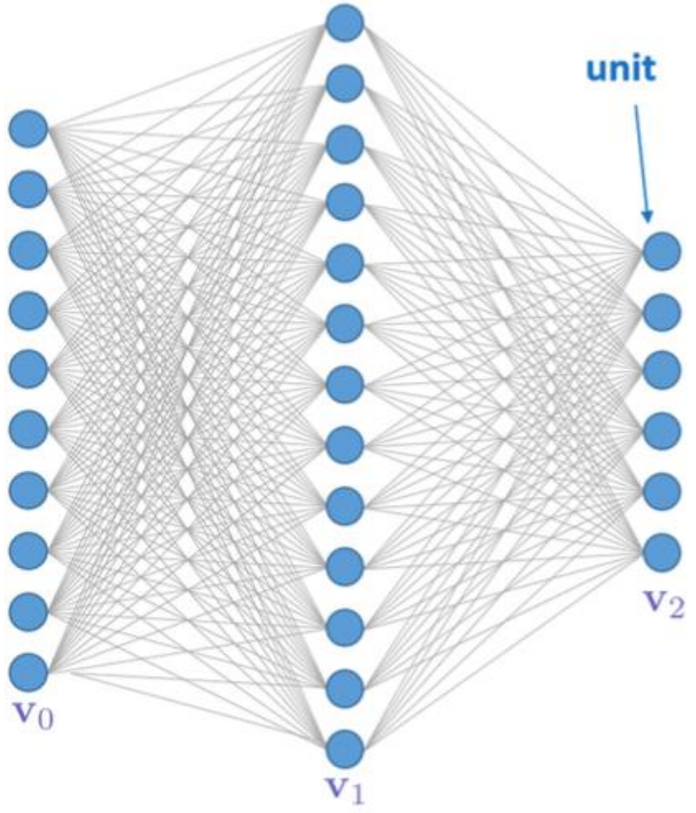
- 4.优化器
- 5.手写数字识别程序优化

卷积神经网络

CNN

卷积神经网络是近年发展起来，并广泛应用于图像处理和图像，NLP等领域的一种多层神经网络。

传统BP处理图像时的问题：
1.权值太多，计算量太大
2.权值太多，需要大量样本进行训练。

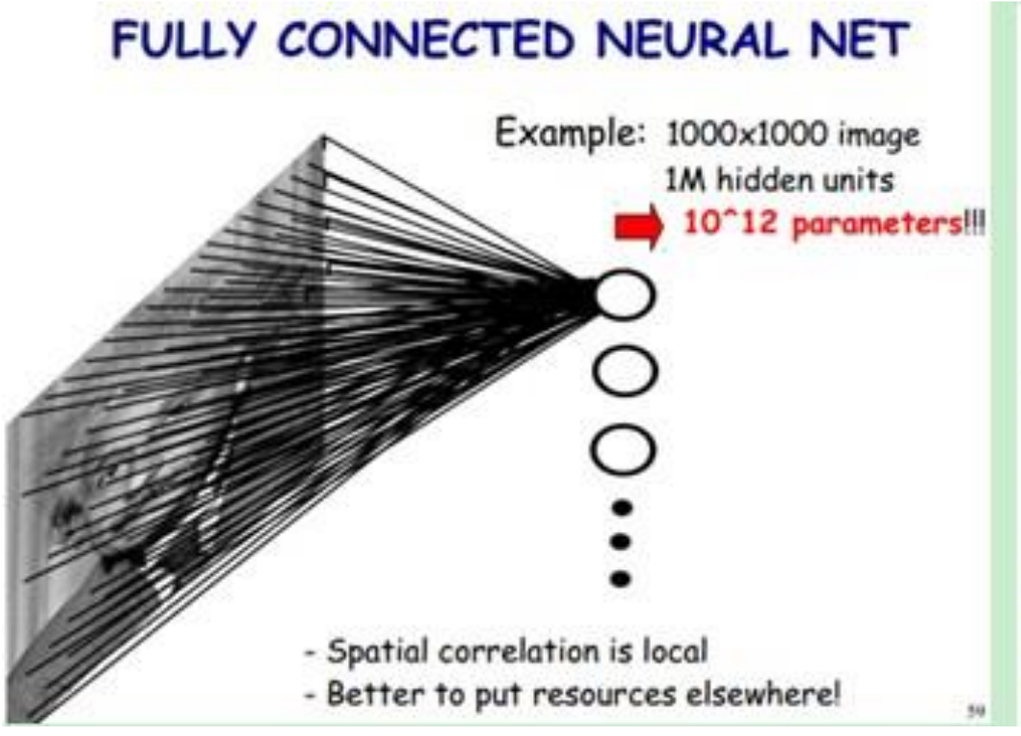


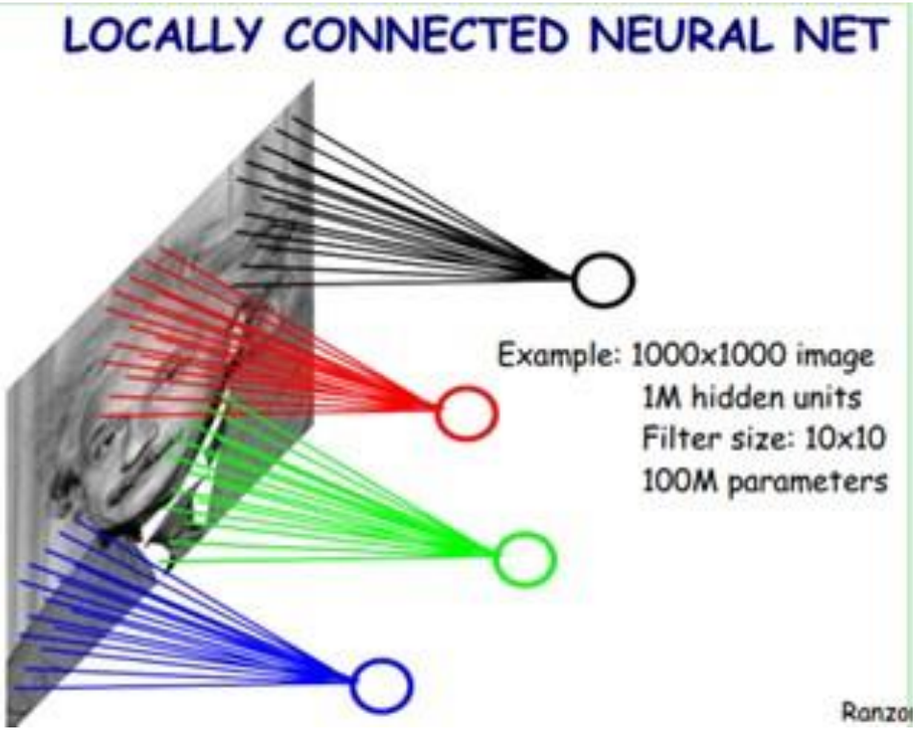
局部感受野

1962年哈佛医学院神经生理学家Hubel和Wiesel通过对猫视觉皮层细胞的研究，提出了感受野(receptive field)的概念，1984年日本学者Fukushima基于感受野概念提出的神经认知机(neocognitron)可以看作是卷积神经网络的第一个实现网络，也是感受野概念在人工神经网络领域的首次应用。

CNN

CNN通过局部感受野和权值共享减少了神经网络需要训练的参数个数。



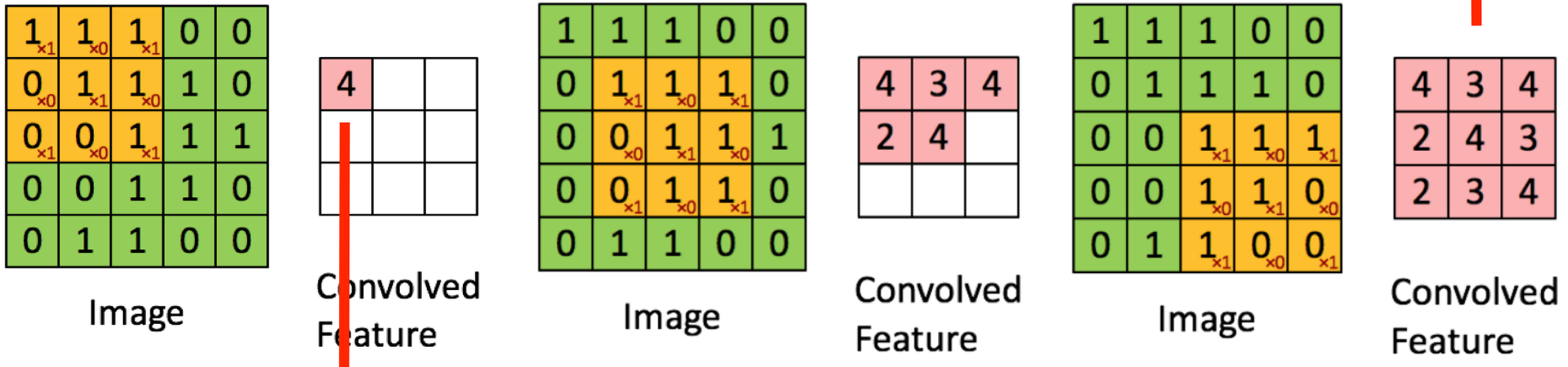


卷积计算

卷积核/滤波器

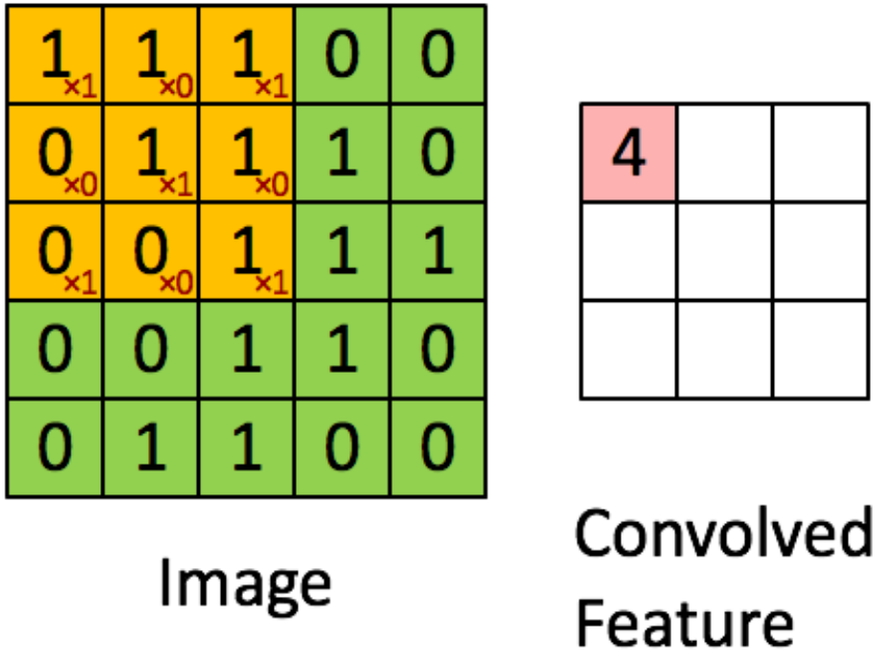
1	0	1
0	1	0
1	0	1

特征图:feature map

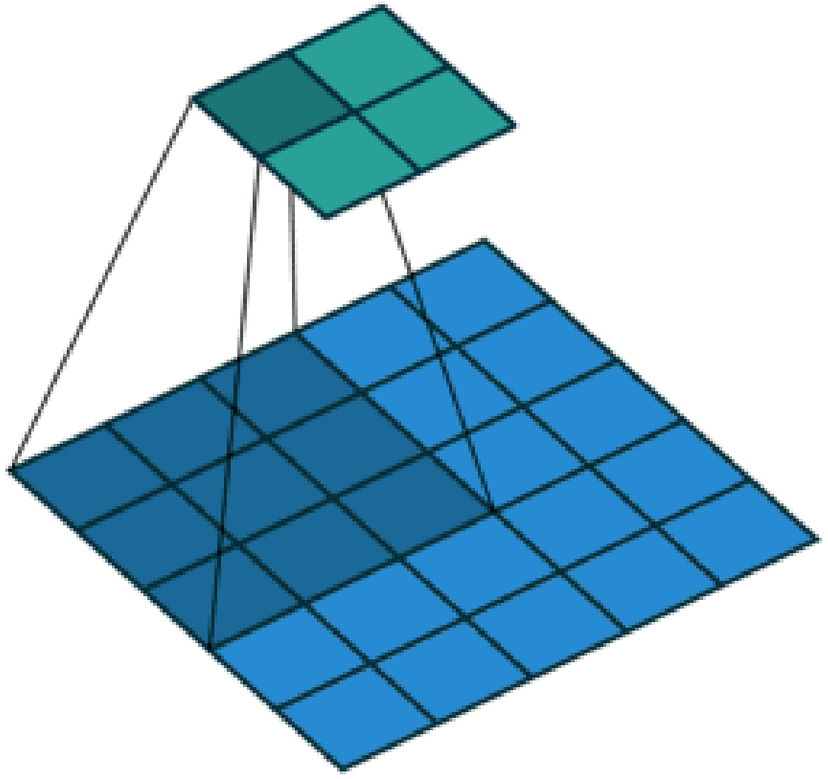


$$1*1+1*0+1*1+0*0+1*1+1*0+0*1+0*0+1*1 = 4$$

不同步长的卷积

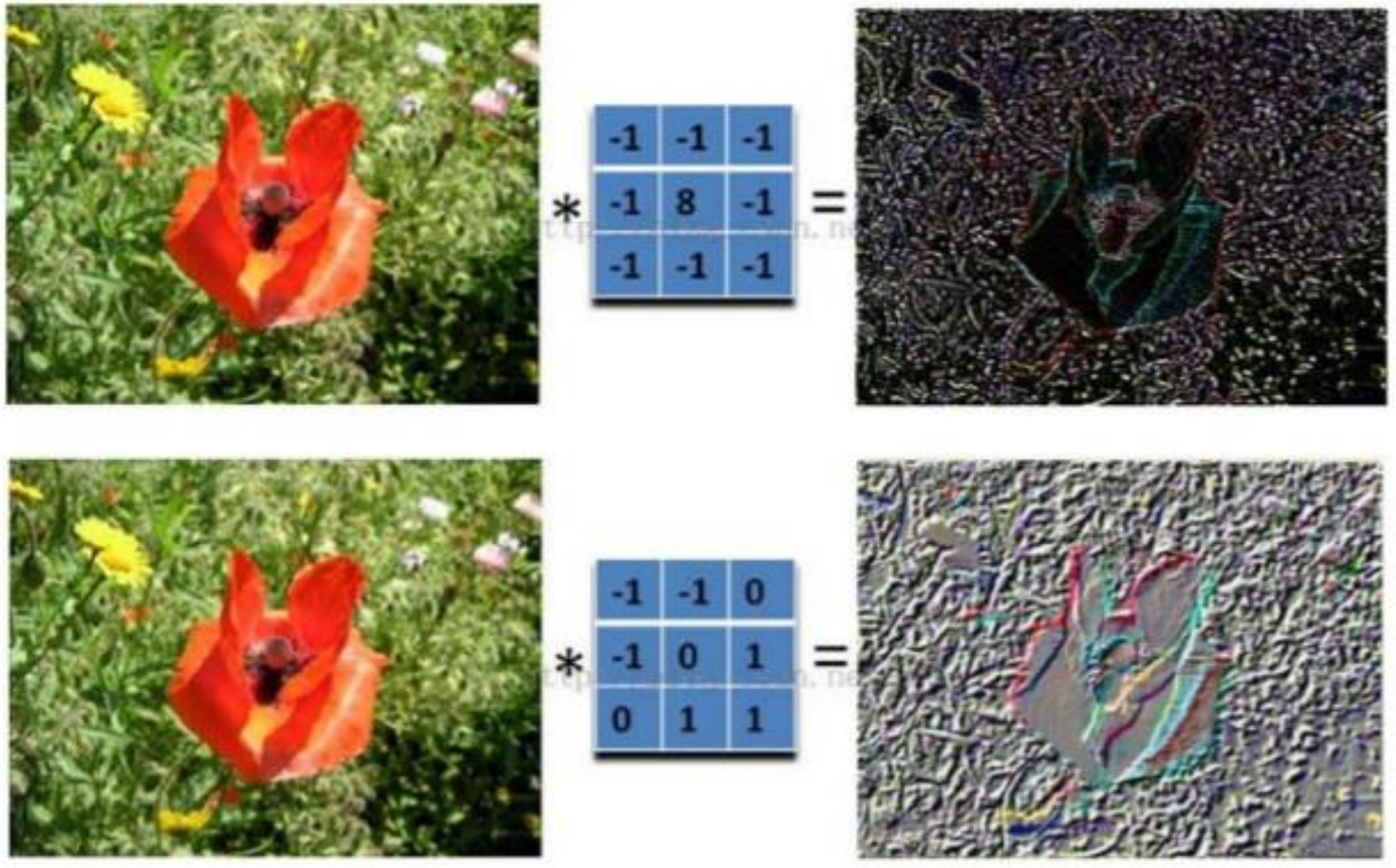


步长为1



步长为2

滤波器



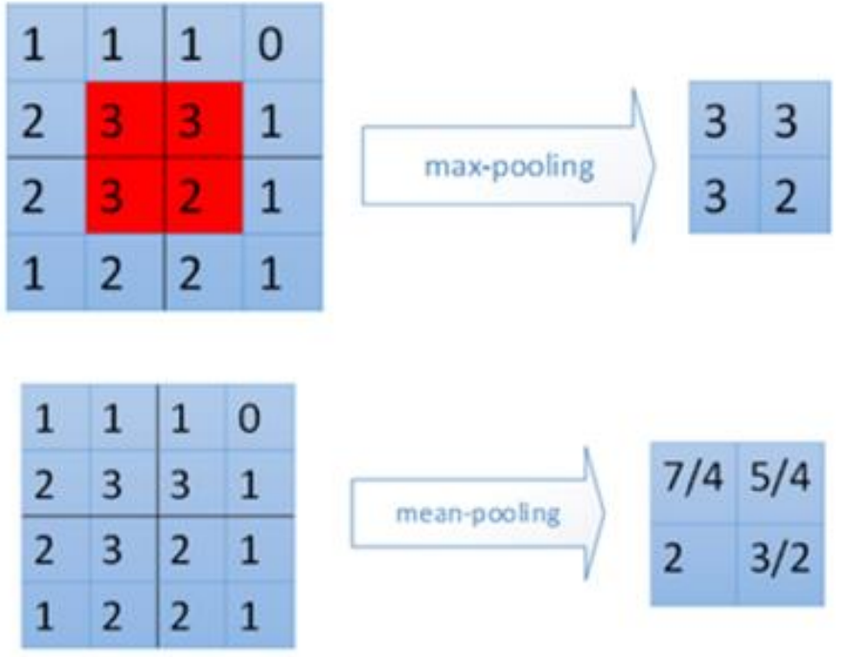
池化(Pooling)

Pooling常用的三种方式：

1.max-pooling

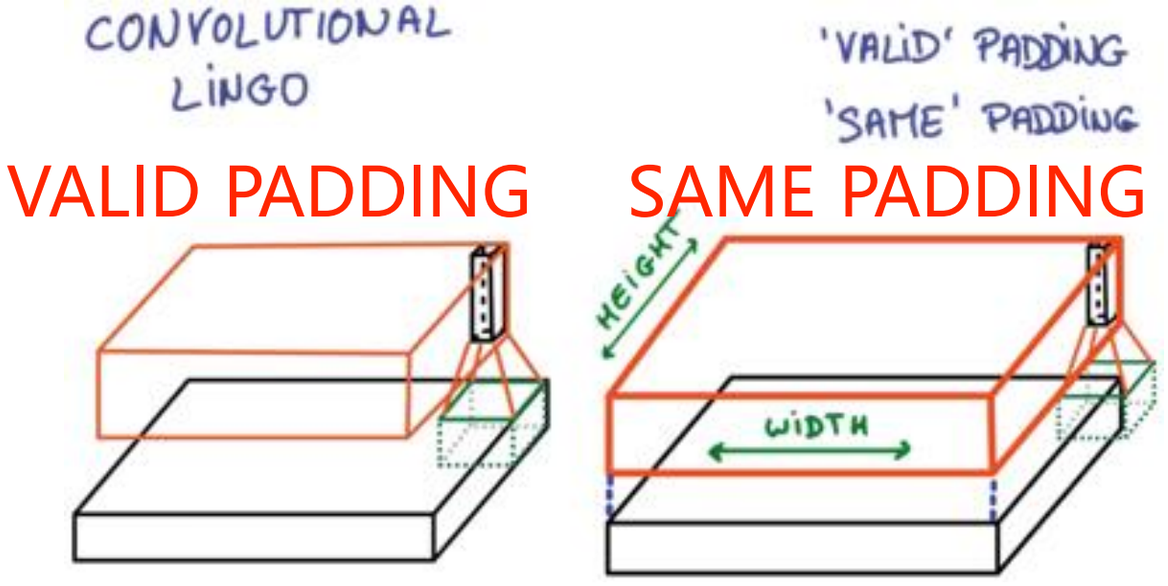
2.mean-pooling

3.stochastic pooling



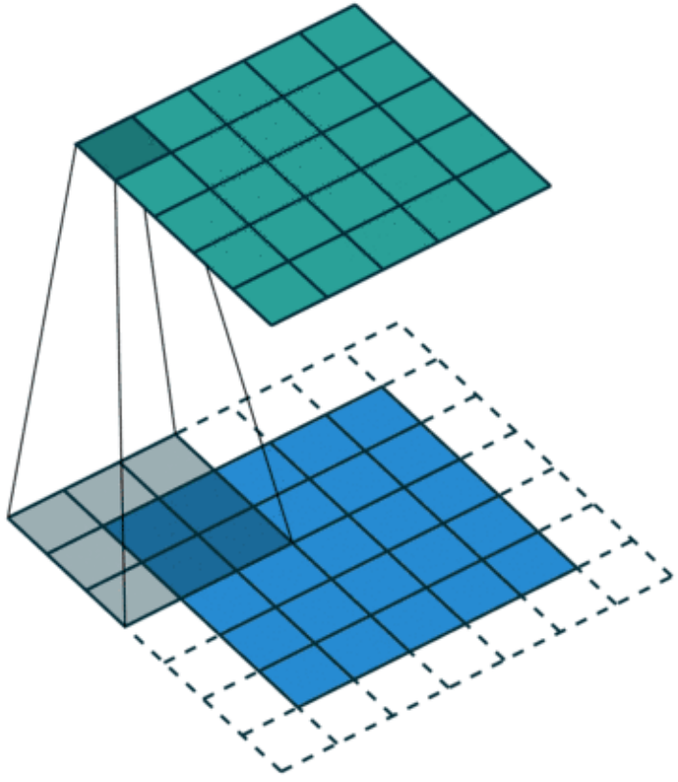
卷积Padding

SAME PADDING:
给平面外部补0
卷积窗口采样后得到一个跟原来大小相同的平面
VALID PADDING:
不会超出平面外部
卷积窗口采样后得到一个比原来平面小的平面

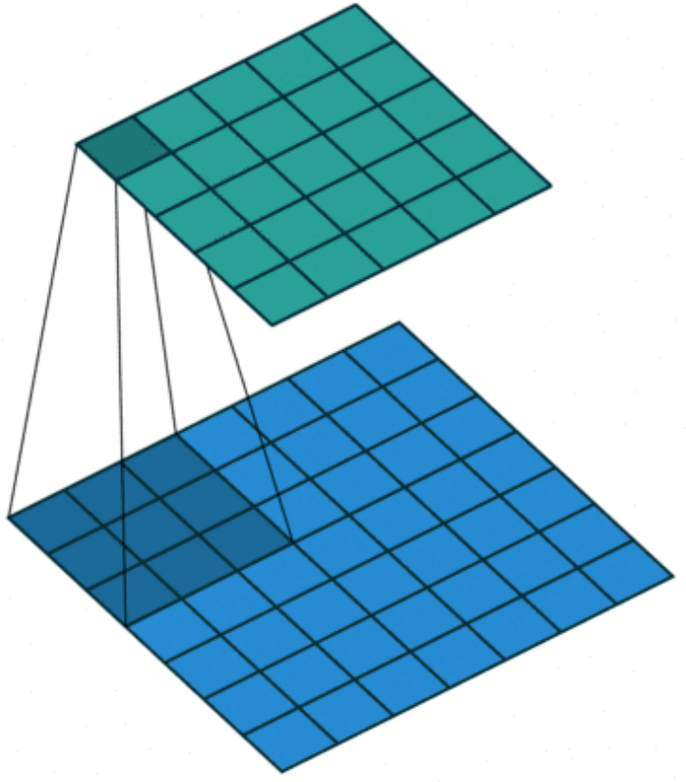


卷积Padding

SAME PADDING



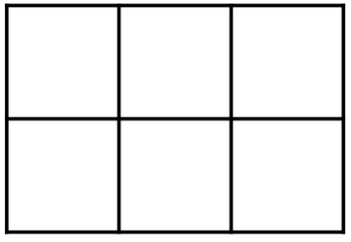
VALID PADDING



池化Padding

SAME PADDING:可能会给平面外部补0
VALID PADDING:不会超出平面外部
假如有一个28*28的平面，用2*2步长为2的窗口对其进行pooling操作
使用SAME PADDING的方式，得到14*14的平面
使用VALID PADDING的方式，得到14*14的平面

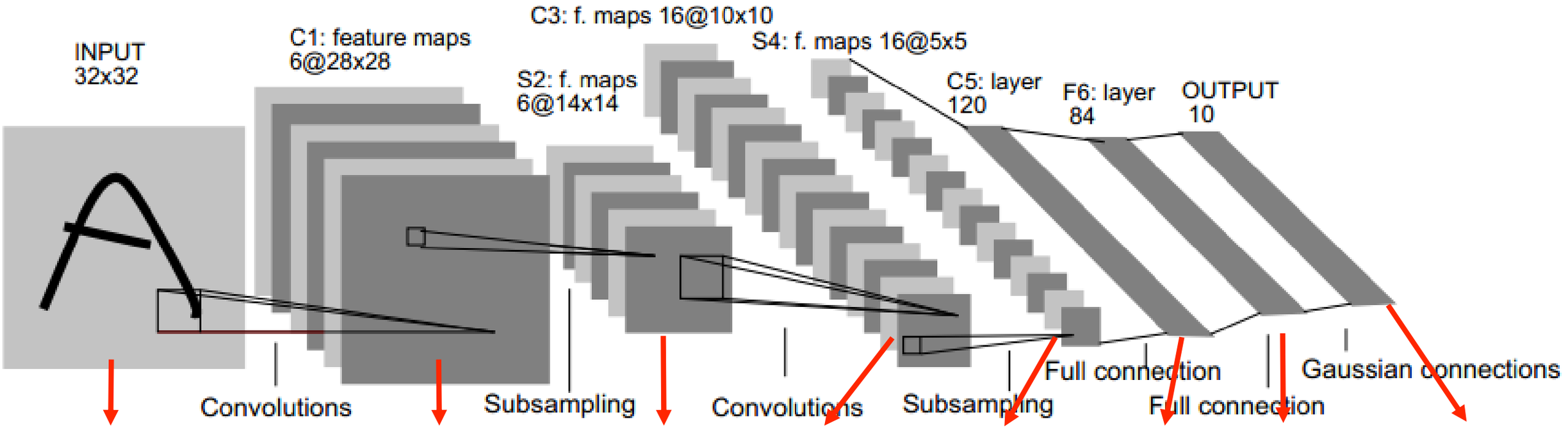
假如有一个2*3的平面，用2*2步长为2的窗口对其进行pooling操作



使用SAME PADDING的方式，得到1*2的平面
使用VALID PADDING的方式，得到1*1的平面

LeNET-5

LeNET-5是最早的卷积神经网络之一，曾广泛用于美国银行。手写数字识别正确率在99%以上。



输入层 卷积层 池化层 卷积层 池化层 全链接层 全链接层 输出层

卷积窗口:5*5
步长:1

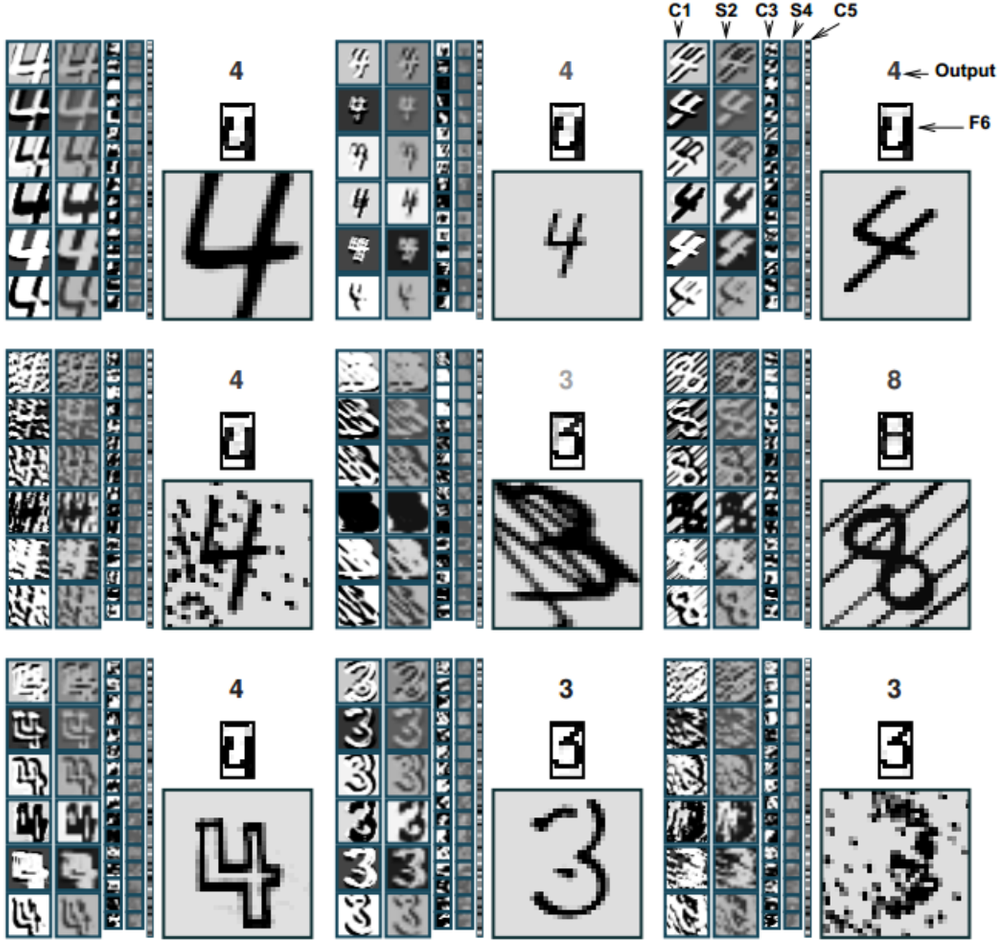
池化窗口:2*2
步长:2

卷积窗口:5*5
步长:1

池化窗口:2*2
步长:2

LeNET-5

LeNET-5可视化



卷积神经网络程序

- 6.卷积神经网络应用于MNIST数据集分类

ImageNet

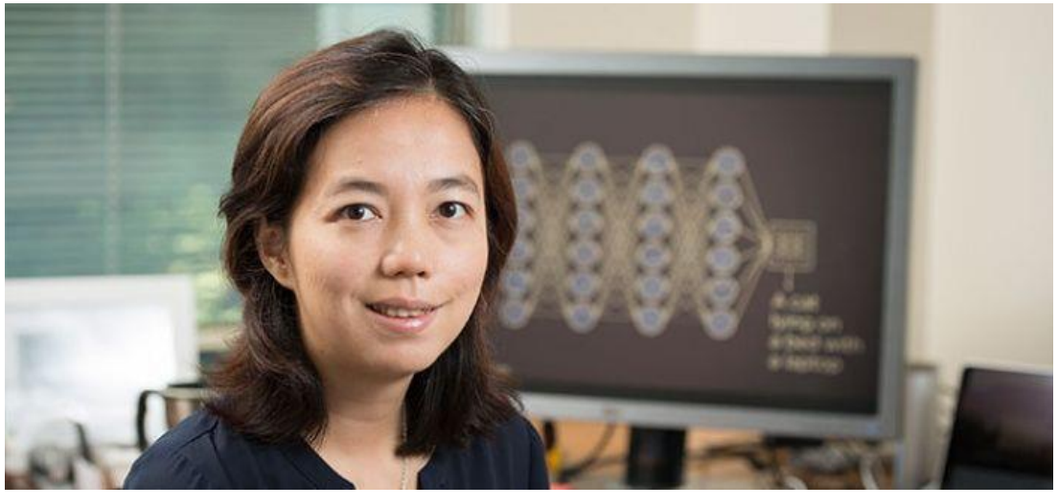
ImageNet

ImageNet是一个计算机视觉系统识别项目，是目前世界上图像识别最大的数据库。一共有1500万张左右的图片，被分为22000个左右的类。是由斯坦福教授李飞飞领导建立的。

TED演讲：我们怎么教计算机理解图片？

李飞飞

1976年出生于北京，长在四川，16岁随父母移居美国新泽西州。
1999年毕业于普林斯顿大学，2005年获得加州理工学院电子工程博士。
2009年加入斯坦福大学担任助理教授，并于2012年担任副教授（终生教授），和斯坦福人工智能实验室与视觉实验室主任。
2017年1月入职Google，担任谷歌云首席科学家。



ILSVRC

ILSVRC:ImageNet Large Scale Visual Recognition Challenge

ImageNet Challenge

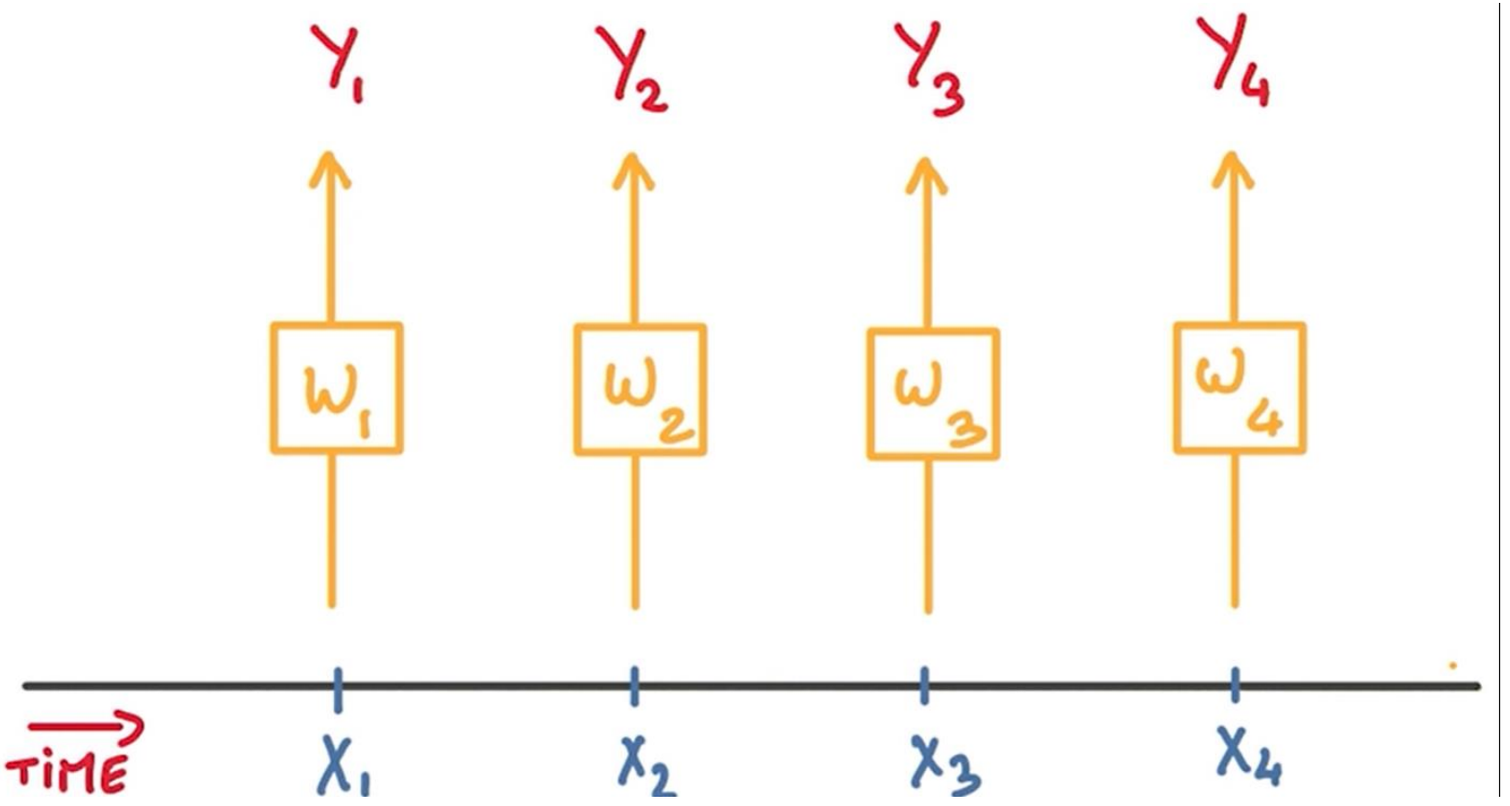
IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

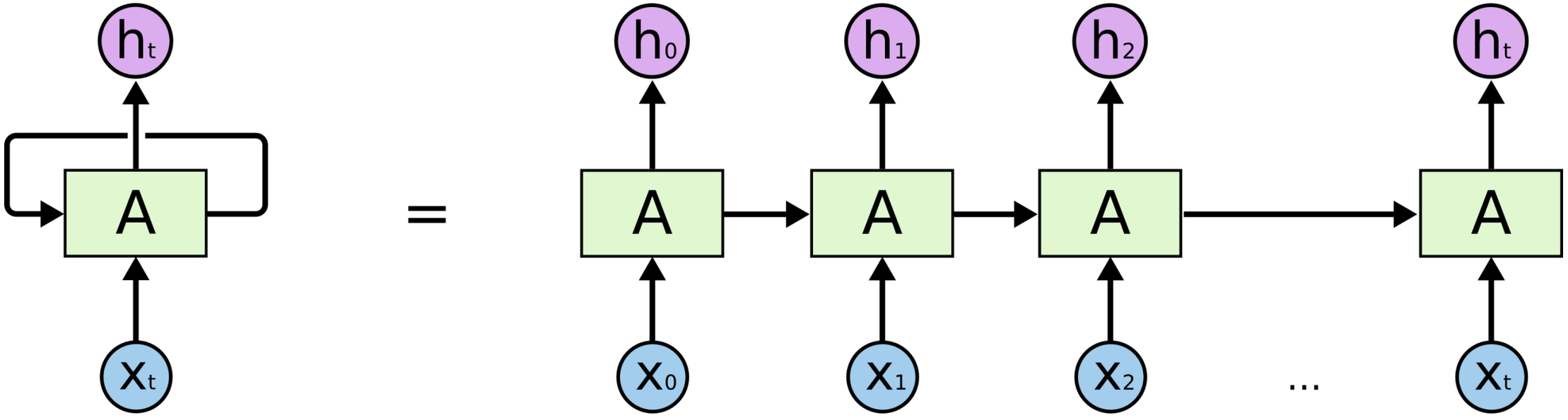
递归神经网络

RNN（Recurrent Neural Network）：

RNN称为递归神经网络。在过去几年RNN在语音识别，自然语言处理，机器翻译等领域有着非常多的应用。



RNN（Recurrent Neural Network）：



RNN（Recurrent Neural Network）：

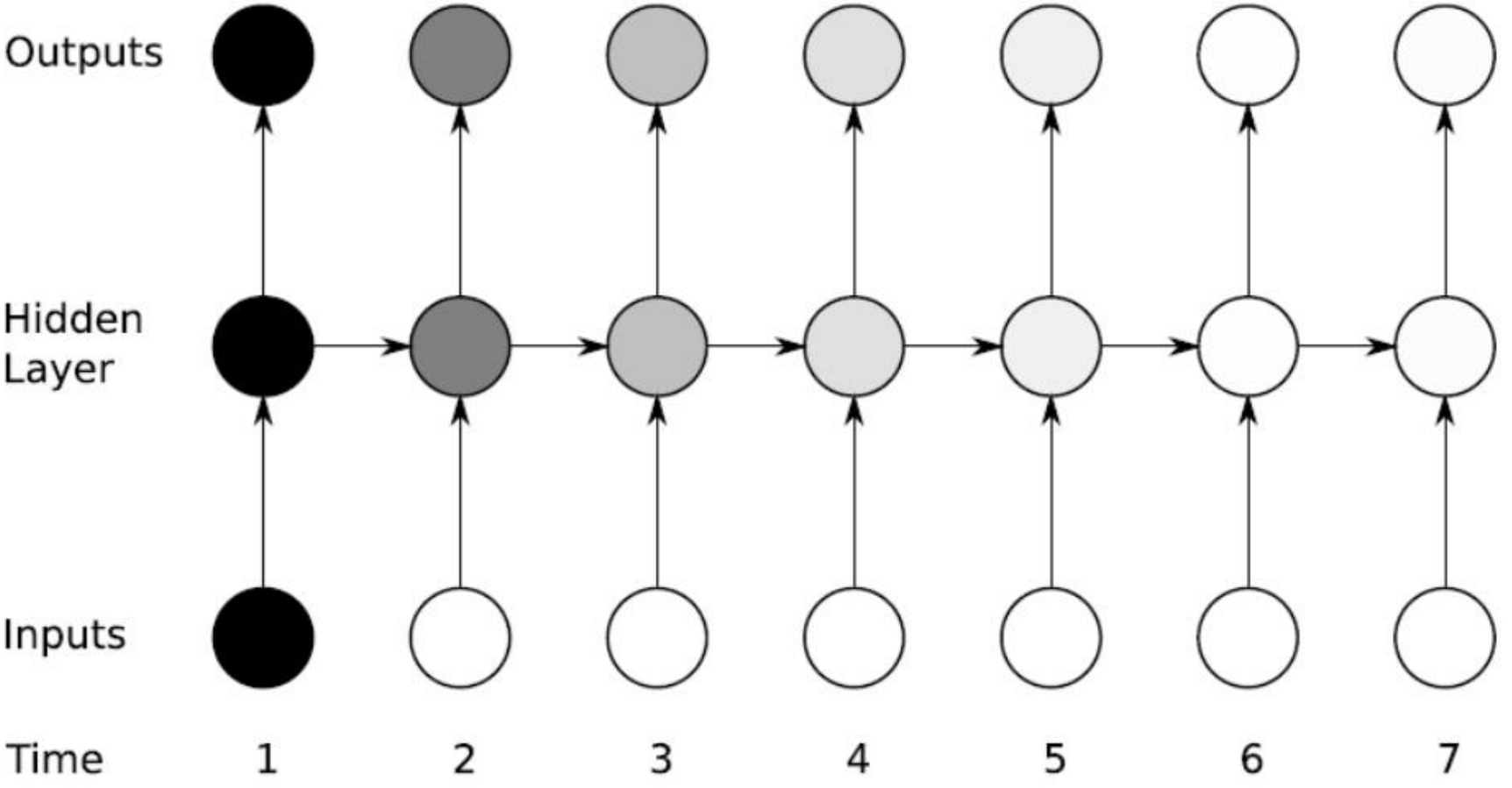
RNN一个重要的用法就是通过之前的信息来决策当前的问题。

(比如就像我们看电影，我们要根据电影之前的情节，才能理解现在的情节。)

例子1：有一朵云飘在（ ）

例子2：我从小生长在中国。。。我可以说一口流利的（ ）

RNN（Recurrent Neural Network）：



长短时记忆网络

LSTM(Long Short Term Memory)：

$$\mathbf{i}_t = \sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + \mathbf{b}_i)$$

$$\mathbf{f}_t = \sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + \mathbf{b}_f)$$

$$\tilde{\mathbf{c}}_t = \tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{o}_t = \sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + \mathbf{b}_o)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

i_t 输入门信号

f_t 忘记门信号

$\tilde{c}_t$ 输入信号

c_t Cell输出信号

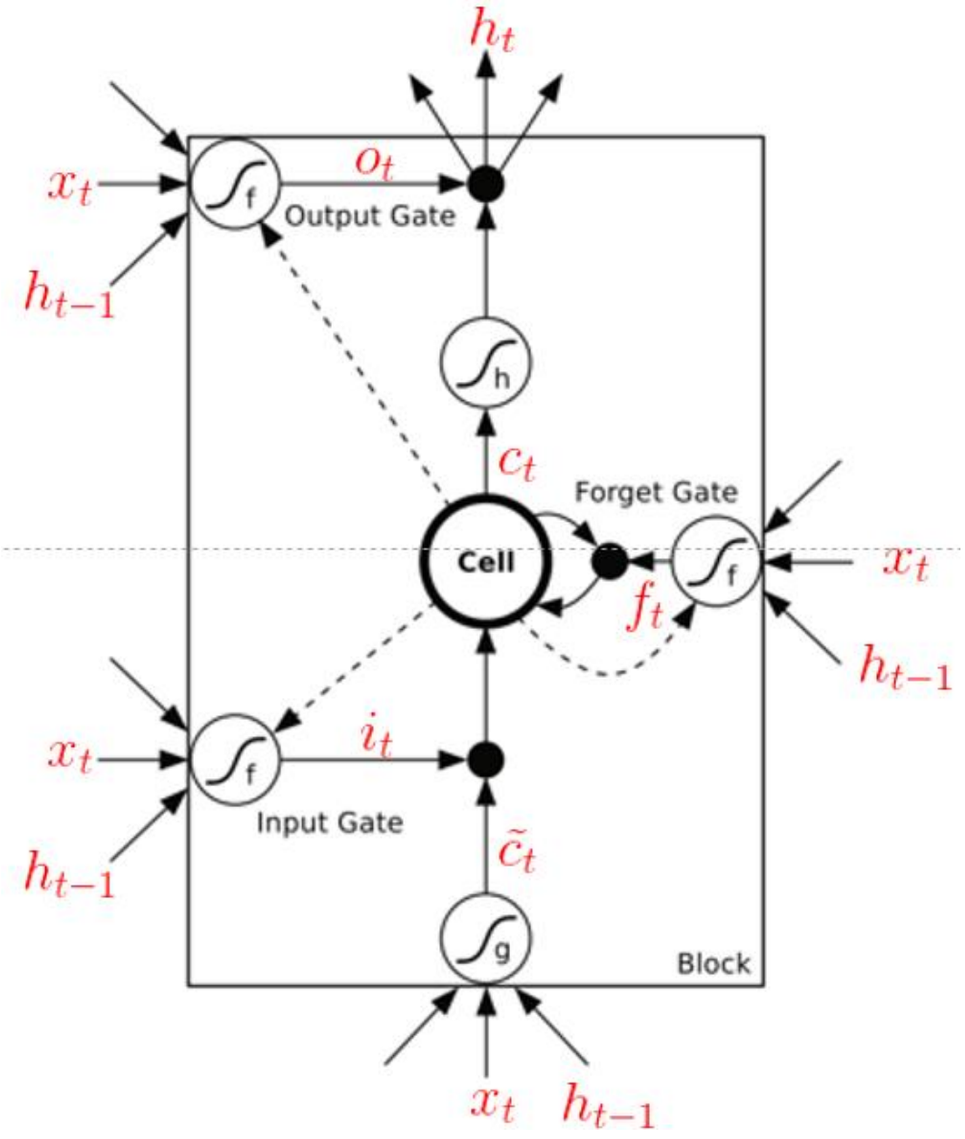
o_t 输出门信号

h_t block输出信号

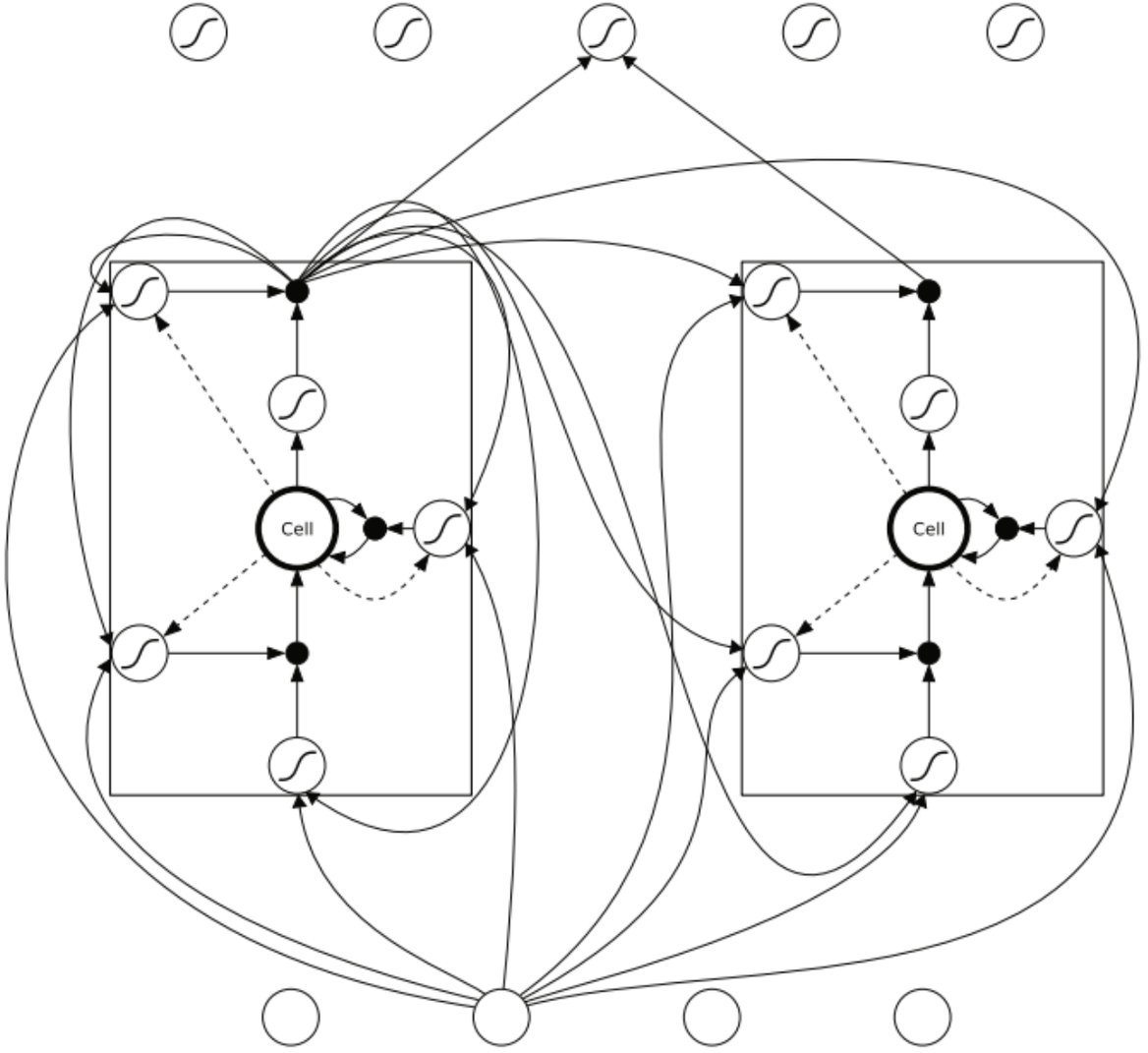
x_t 第t个序列输入

h_{t-1} 第t-1个序列输出

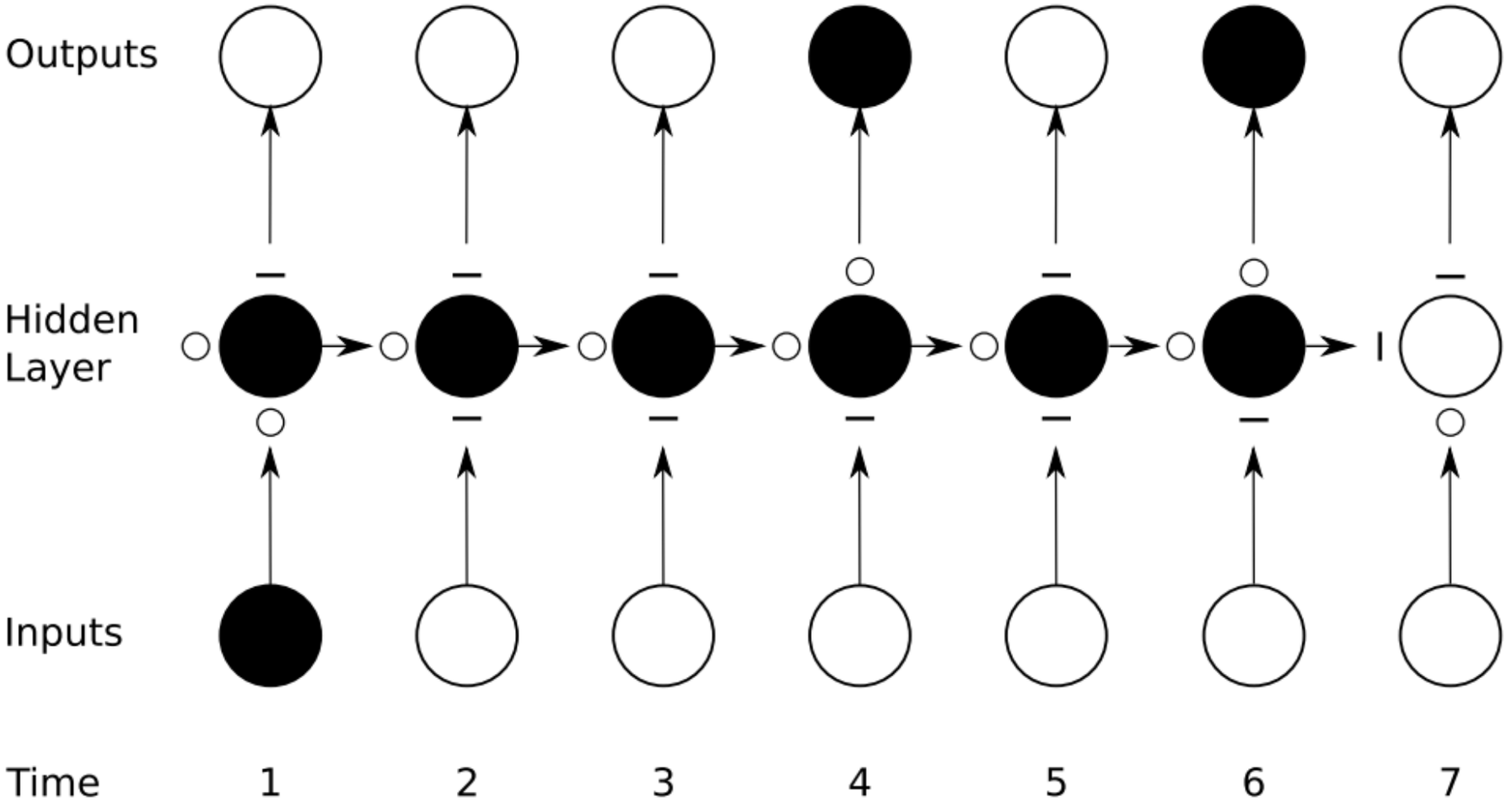
σ sigmoid函数



LSTM(Long Short Term Memory)：



LSTM(Long Short Term Memory)：



LSTM网络程序

- 7.长短时记忆网络LSTM

模型保存和载入

- 8.saver_save
- 9.saver_restore
- 10.保存模型参数与结构
- 11.载入模型参数和结构

使用Inception-v3做图像识别

模型保存和载入

- 12.下载google图像识别网络inception-v3并查看结构
- 13.使用inception-v3做各种图像的识别

训练自己的图像识别模型

Thanks!