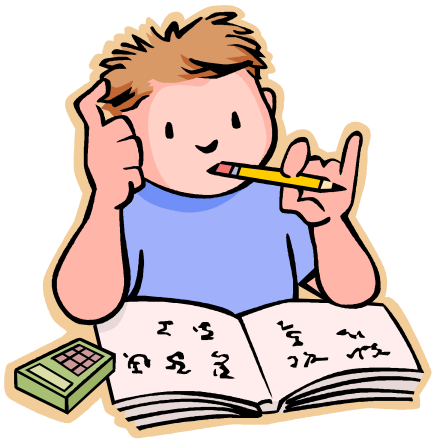
上次课回顾

- 数据挖掘技术的产生
- 数据挖掘概念
- 数据挖掘技术的发展趋势
- 数据挖掘技术的分类问题
- 数据挖掘常用的知识表示模式与方法
- 不同数据存储形式下的数据挖掘问题
- 粗糙集方法及其在数据挖掘中的应用
- 数据挖掘的应用分析



数据挖掘技术的产生

大量数据、强大算力、实际需求

data → information ↔ knowledge

数据挖掘技术举例——十大算法

一、 C4.5：分类决策树算法,其核心算法是ID3 算法

二、k-means：k-平均算法是解决聚类问题

三、SVM：支持向量机

四、Apriori：挖掘布尔关联规则频繁项集的算法

五、EM ：最大期望算法

六、PageRank：网页排名、搜索引擎

七、AdaBoost：自适应增强

八、KNN：k-近邻算法

九、Naive Baye：朴素贝叶斯分类器

十、CART：分类回归树

投 票(可多选)

- 网上购物时，用户推荐系统可能用到以下哪些方法？

A、决策树

B、k-means

C、SVM

D、Apriori

E、EM

F、KNN

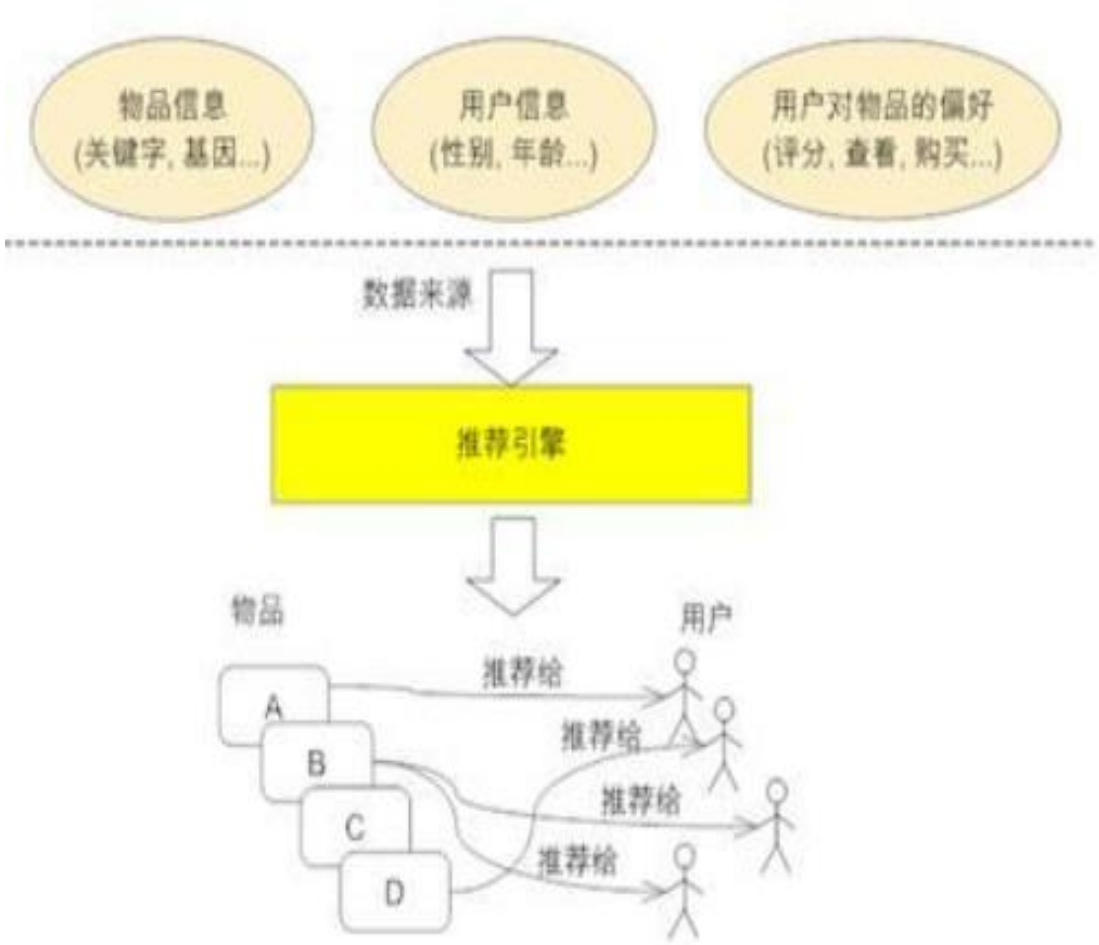
G、Naive Baye

H、其它（填写）_____.

购物推荐

个性化推荐系统

是建立在海量数据挖掘基础上的一种高级商务智能平台，以帮助电子商务网站为其顾客购物提供完全个性化的决策支持和信息服务。



购物推荐

2、推荐系统有三个重要的模块：用户建模模块、推荐对象建模模块、推荐算法模块。

3、主要推荐方法：基于内容推荐、协同过滤推荐、基于关联规则推荐、基于效用推荐、基于知识推荐、组合推荐

推荐系统

发展历程：

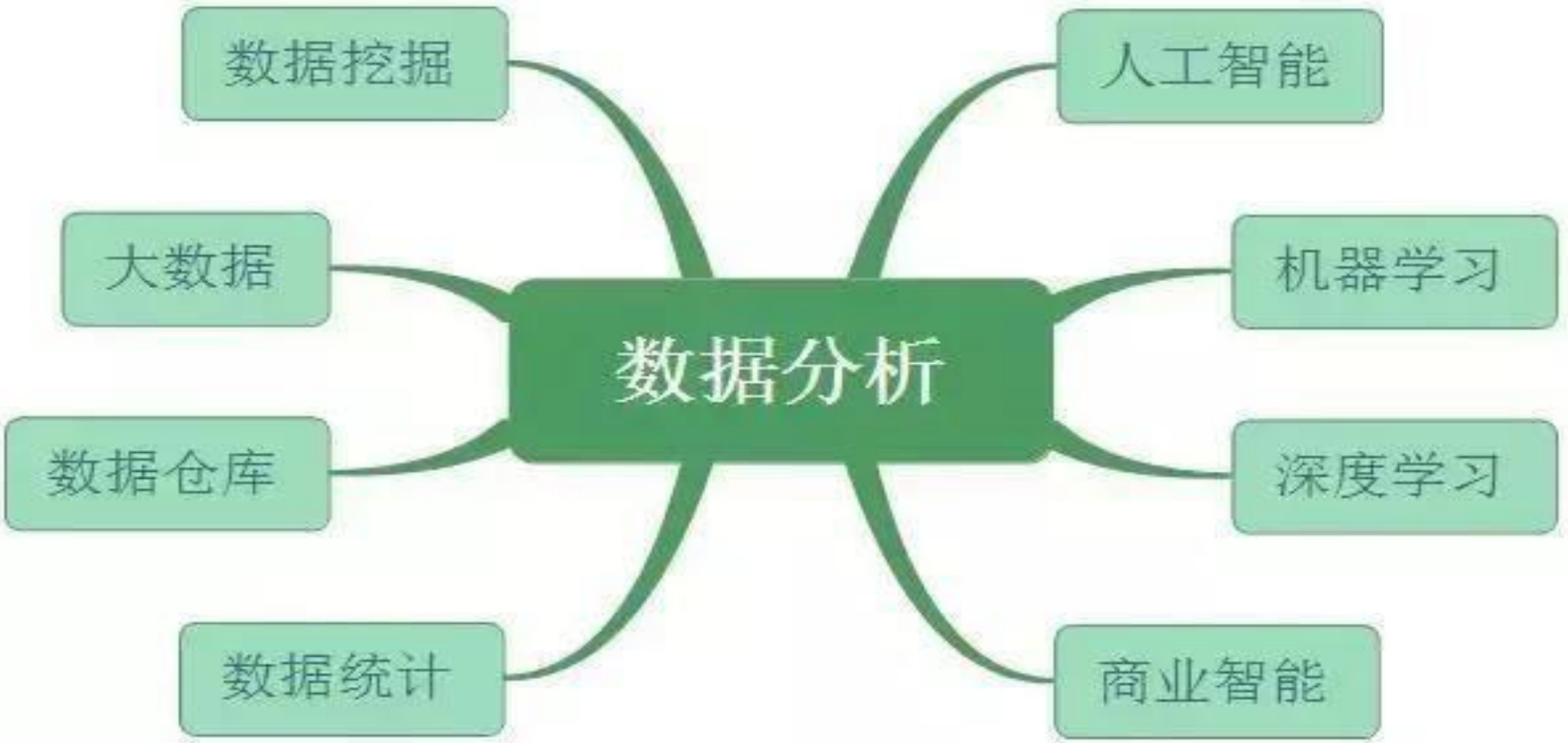
1995年3月，卡耐基.梅隆大学提出了个性化导航系统Web Watcher；斯坦福大学推出了个性化推荐系统LIRA；

2003年，Google开创了AdWords盈利模式，通过用户搜索的关键词来提供相关的广告。AdWords的点击率很高，是Google广告收入的主要来源。

2007年，雅虎推出了SmartAds广告方案。雅虎掌握了海量的用户信息，如用户的性别、年龄、收入水平、地理位置以及生活方式等，再加上对用户搜索、浏览行为的记录，使得雅虎可以为用户呈现个性化的横幅广告。

- 数据挖掘技术的产生与概念
- 数据挖掘的发展趋势
- 数据挖掘技术的分类问题
- 数据挖掘常用的知识表示模式与方法
- 不同数据存储形式下的数据挖掘问题
- 粗糙集方法及其在数据挖掘中的应用
- 数据挖掘的应用分析

- 数据分析是一个交叉学科

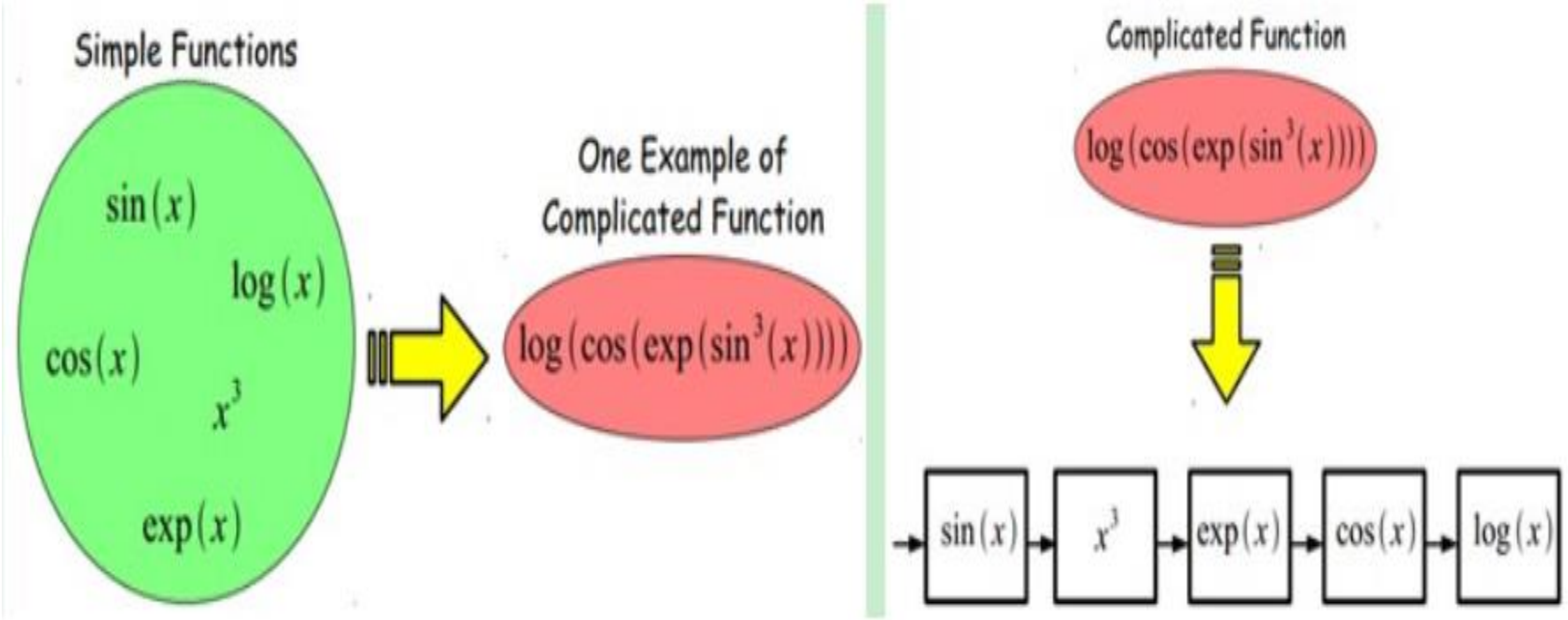


深度学习三大模型

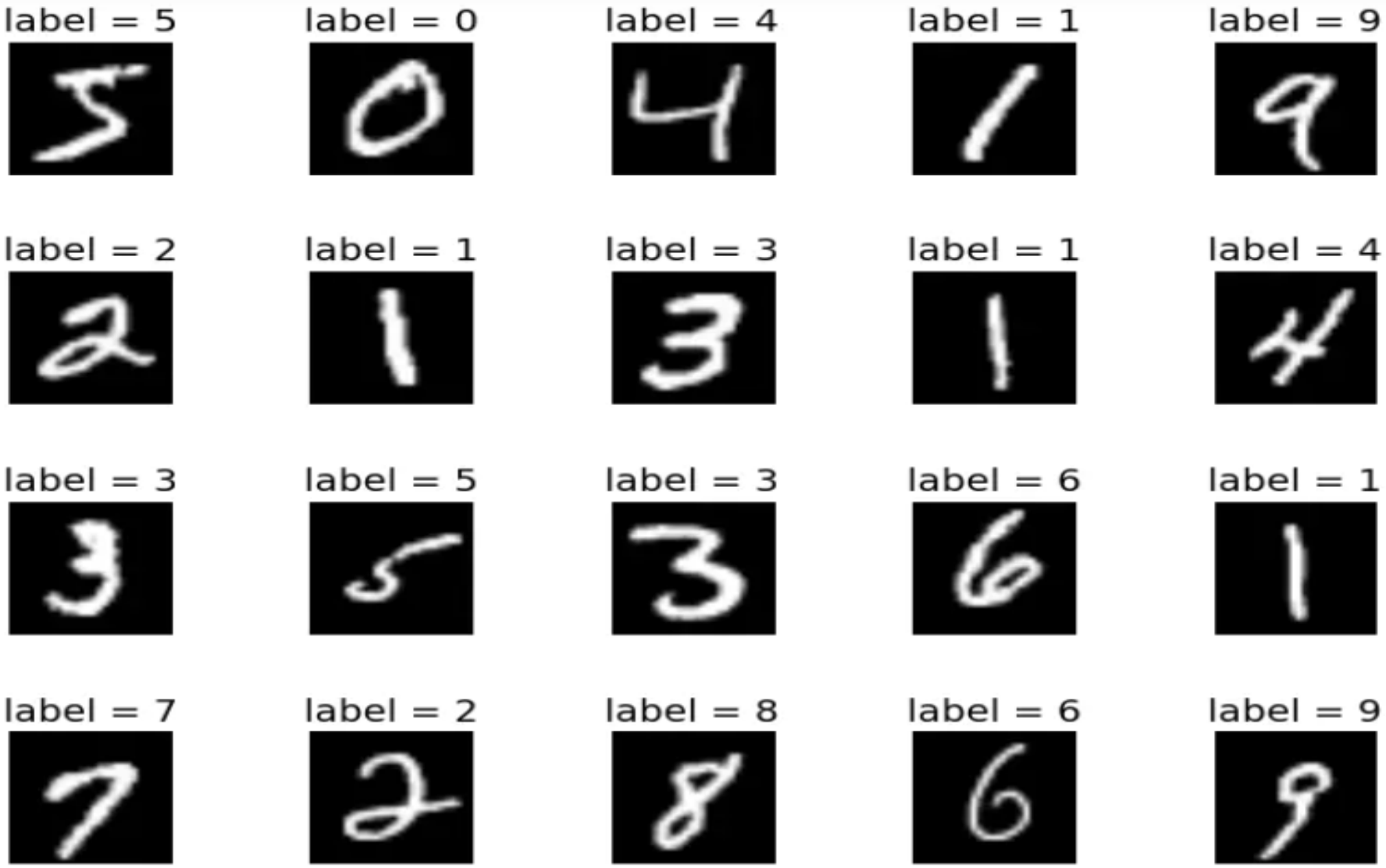
1、DBN：Deep Belief Network, 深度信念网络

2、CNN：Convolution Neural Network卷积神经网络

3、SAE：Sparse Auto Encoder，稀疏自编码



手写数字体识别


label = 5
label = 0
label = 4
label = 1
label = 9
label = 2
label = 1
label = 3
label = 1
label = 4
label = 3
label = 5
label = 3
label = 6
label = 1
label = 7
label = 2
label = 8
label = 6
label = 9

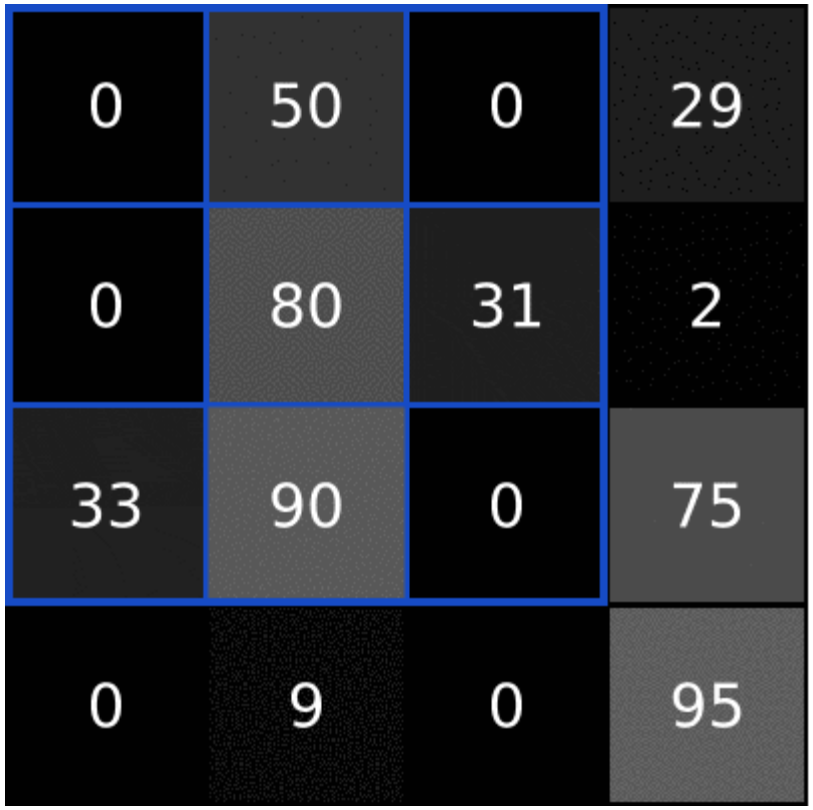
卷积操作

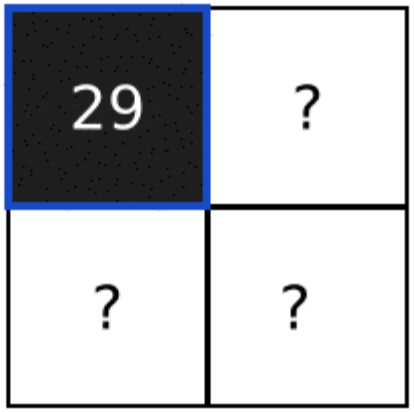
在重叠的图像和滤波器元素之间逐个进行乘法运算，按照从左向右、从上到下的顺序。

0	50	0	29
0	80	31	2
33	90	0	75
0	9	0	95

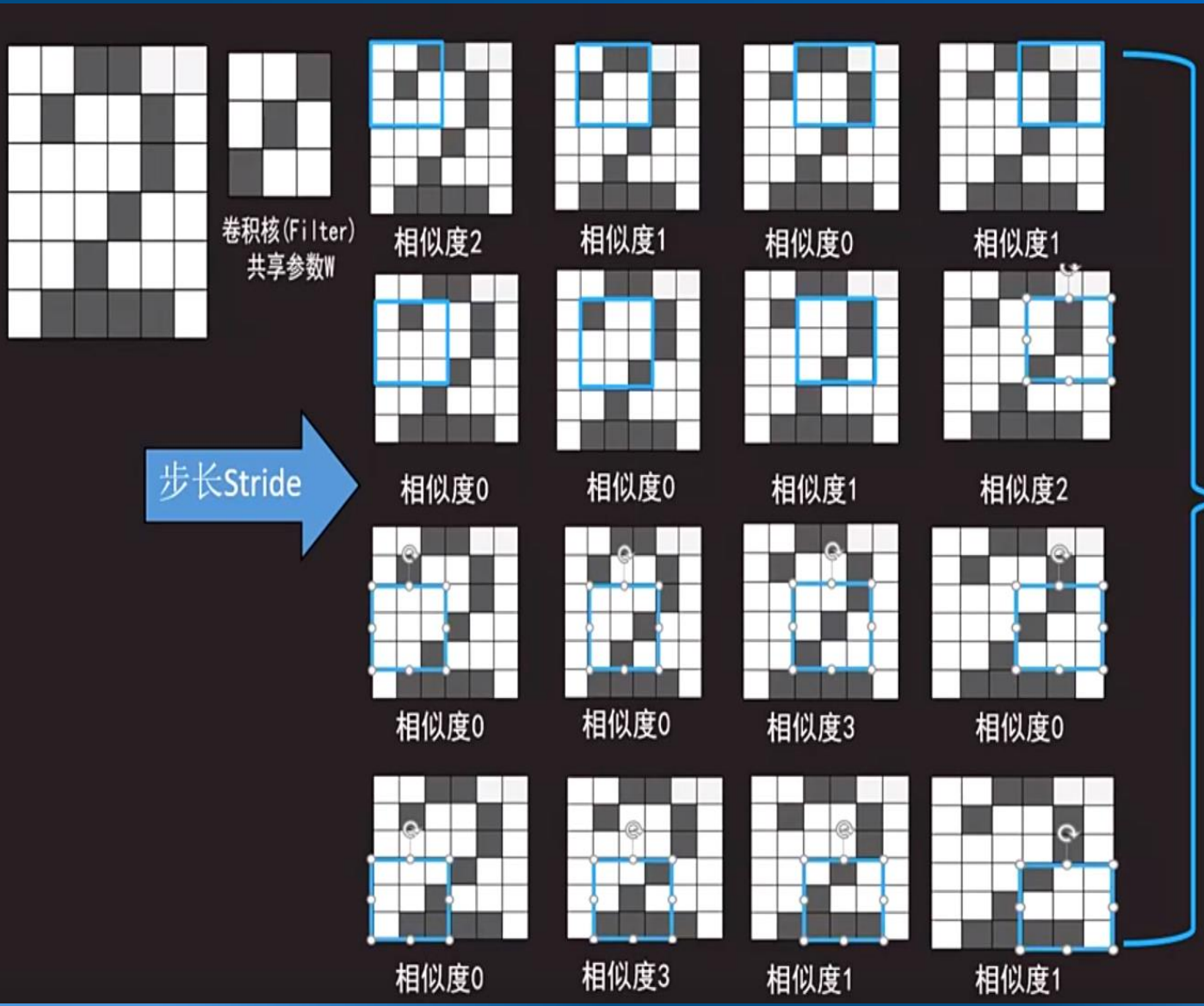
-1	0	1
-2	0	2
-1	0	1

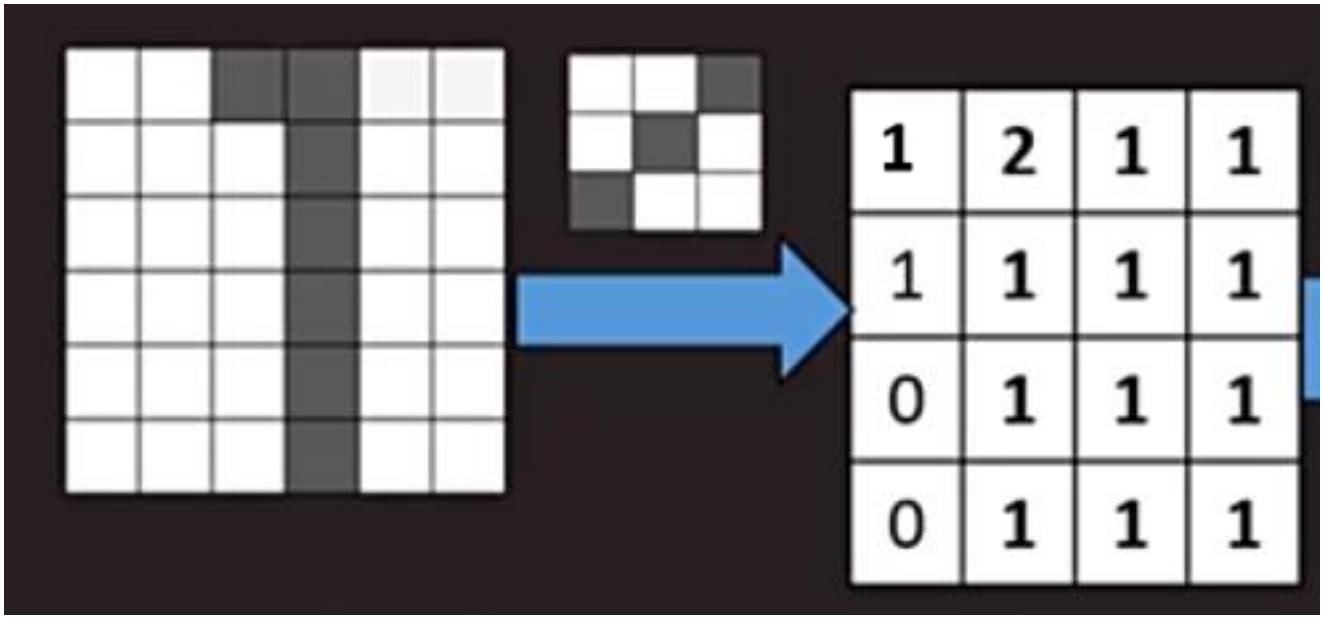
29	-192
-35	-22

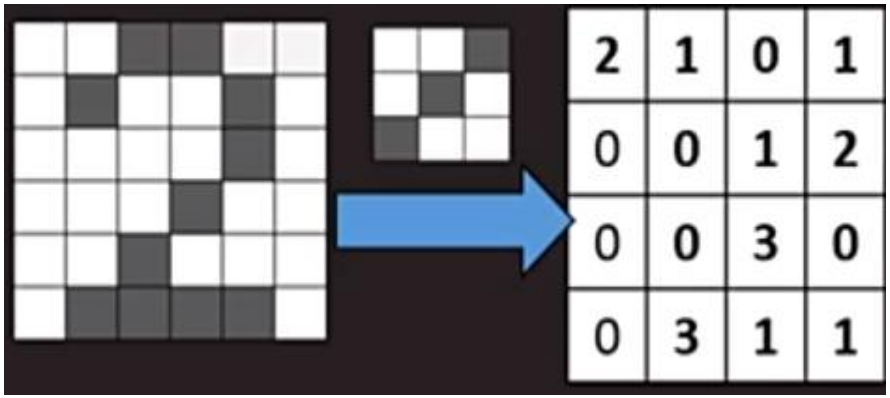

0
50
0
29
0
80
31
2
33
90
0
75
0
9
0
95

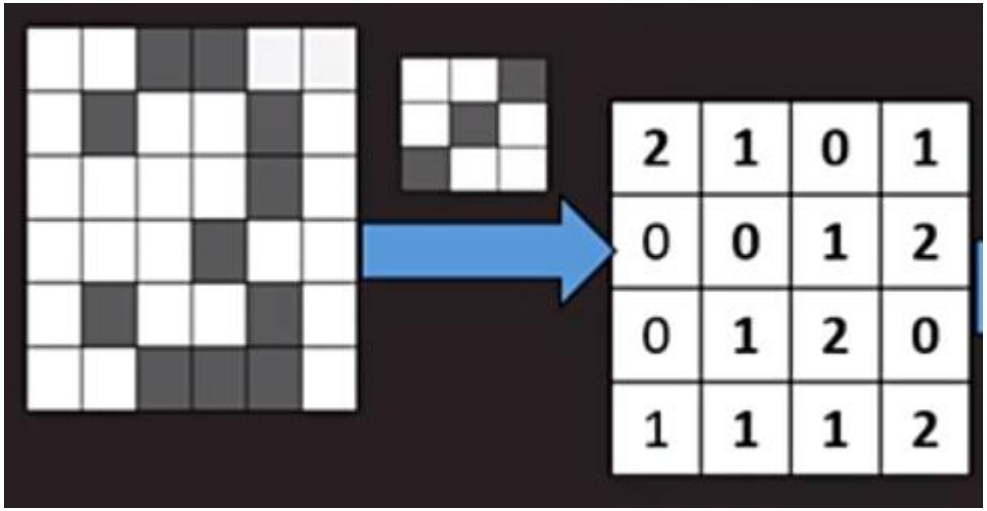

29
?
?
?

CNN——手写数字体识别

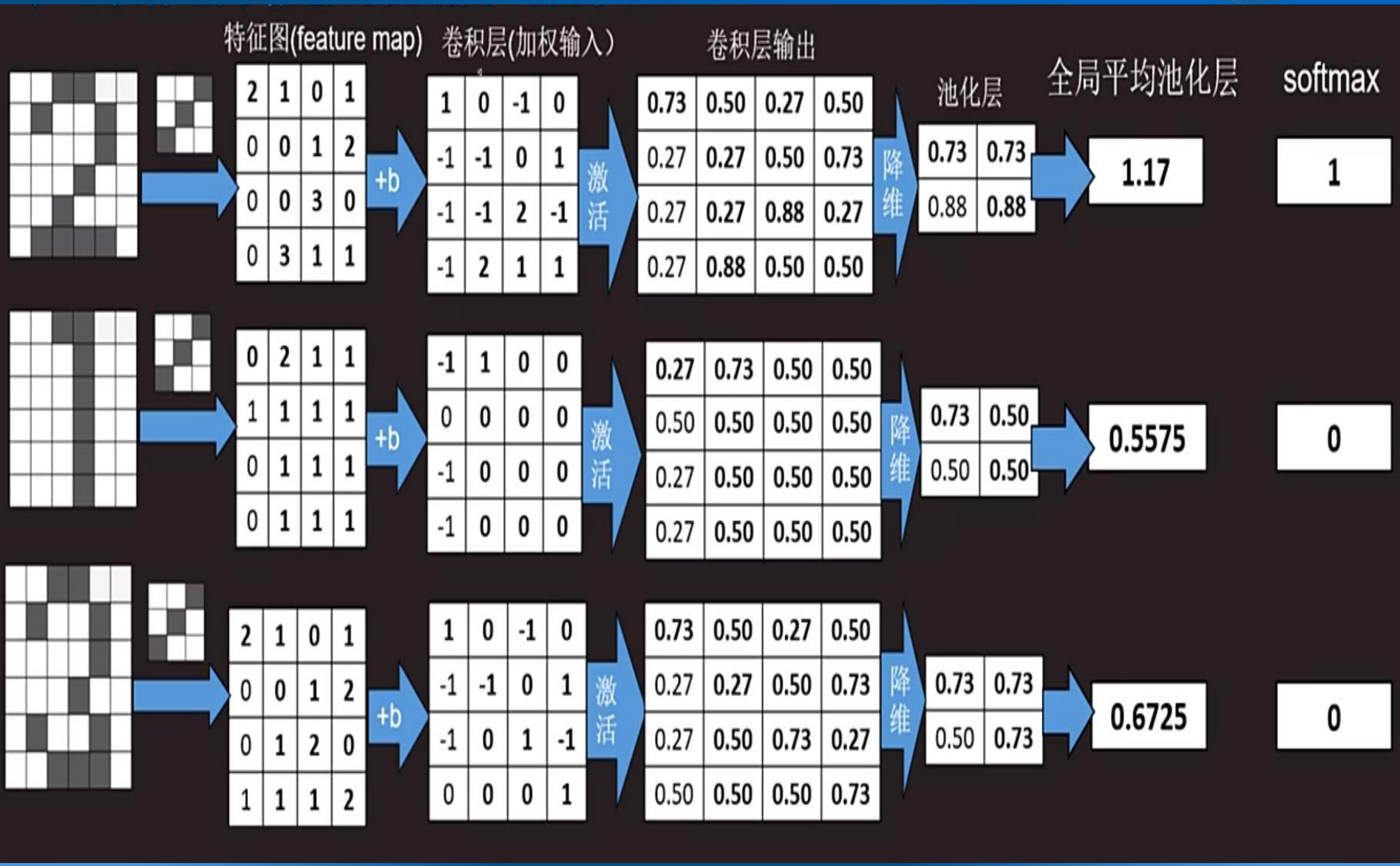

卷积核(Filter)
共享参数W
步长Stride
相似度2
相似度1
相似度0
相似度1
相似度0
相似度0
相似度1
相似度2
相似度0
相似度0
相似度3
相似度0
相似度0
相似度3
相似度1
相似度1
特征图(feature map)
2 1 0 1
0 0 1 2
0 0 3 0
0 3 1 1
b

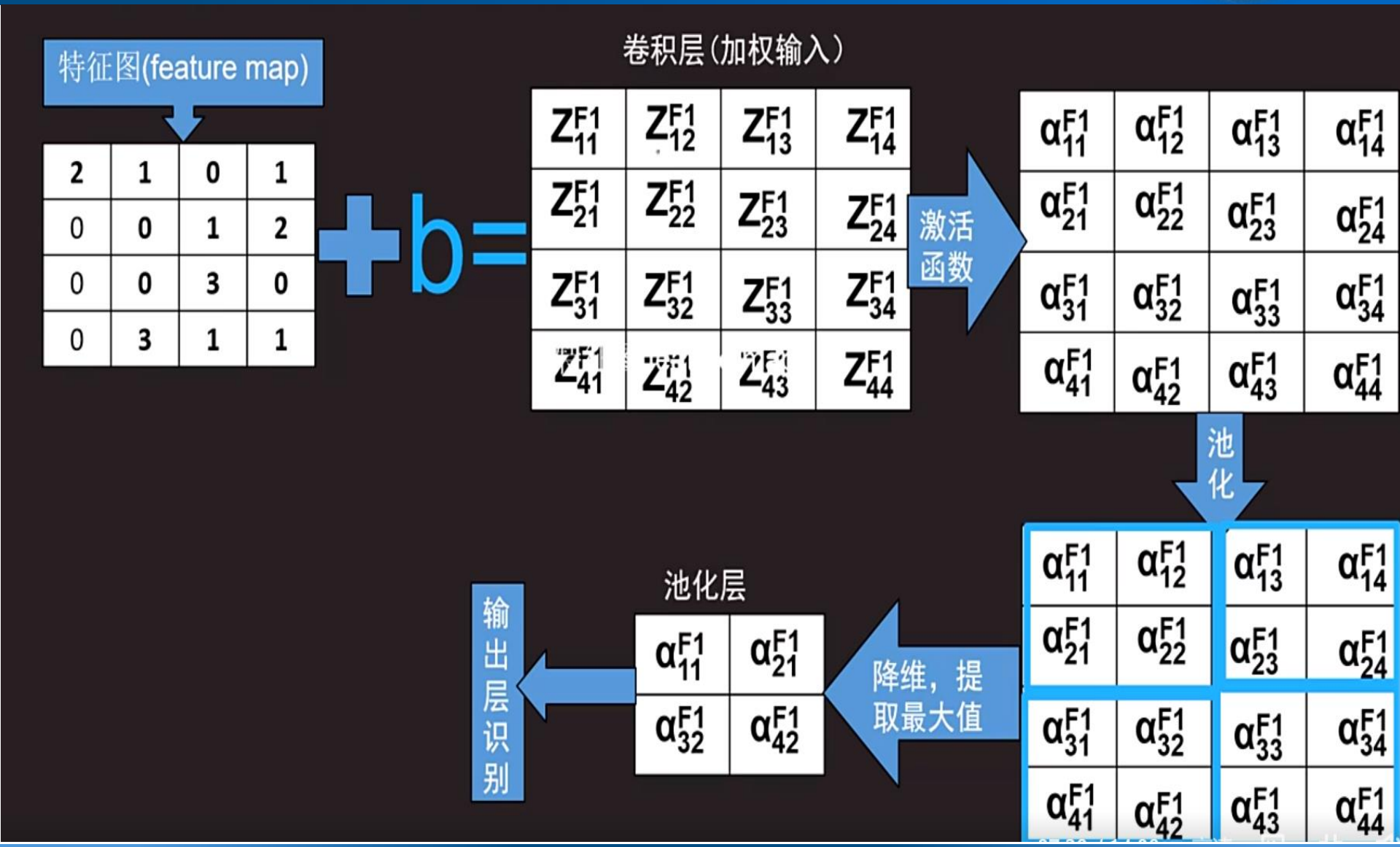

1 2 1 1
1 1 1 1
0 1 1 1
0 1 1 1


2 1 0 1
0 0 1 2
0 0 3 0
0 3 1 1

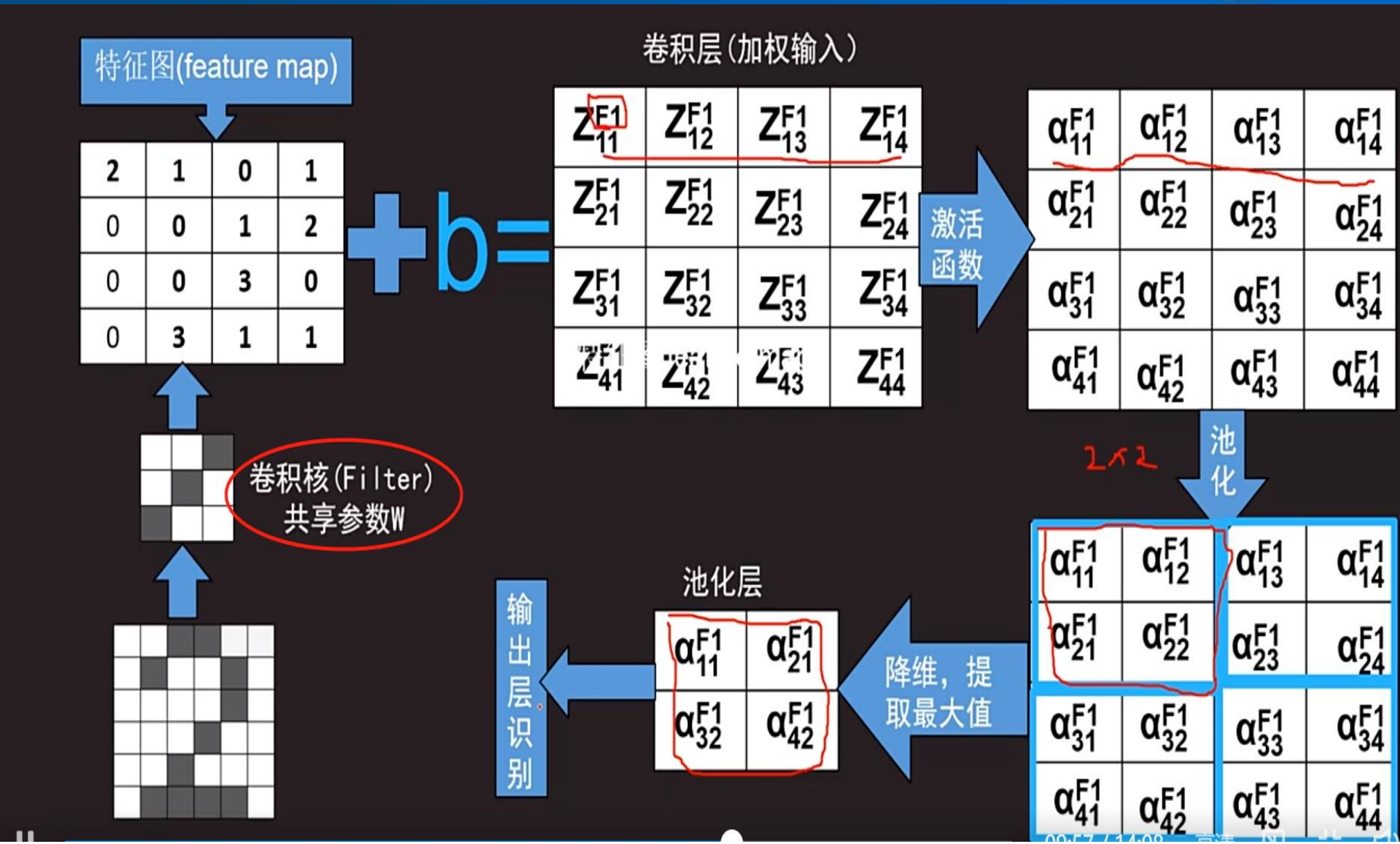

2 1 0 1
0 0 1 2
0 1 2 0
1 1 1 2

CNN——手写数字体识别


特征图(feature map)
卷积层(加权输入)
卷积层输出
池化层
全局平均池化层
softmax
+b
激活
降维
2 1 0 1
0 0 1 2
0 0 3 0
0 3 1 1
1 0 -1 0
-1 -1 0 1
-1 -1 2 -1
-1 2 1 1
0.73 0.50 0.27 0.50
0.27 0.27 0.50 0.73
0.27 0.27 0.88 0.27
0.27 0.88 0.50 0.50
0.73 0.73
0.88 0.88
1.17
1
0 2 1 1
1 1 1 1
0 1 1 1
0 1 1 1
-1 1 0 0
0 0 0 0
-1 0 0 0
-1 0 0 0
0.27 0.73 0.50 0.50
0.50 0.50 0.50 0.50
0.27 0.50 0.50 0.50
0.27 0.50 0.50 0.50
0.73 0.50
0.50 0.50
0.5575
0
2 1 0 1
0 0 1 2
0 1 2 0
1 1 1 2
1 0 -1 0
-1 -1 0 1
-1 0 1 -1
0 0 0 1
0.73 0.50 0.27 0.50
0.27 0.27 0.50 0.73
0.27 0.50 0.73 0.27
0.50 0.50 0.50 0.73
0.73 0.73
0.50 0.73
0.6725
0


特征图(feature map)
2 1 0 1
0 0 1 2
0 0 3 0
0 3 1 1
+b=
卷积层(加权输入)
Z^{F1}_{11} Z^{F1}_{12} Z^{F1}_{13} Z^{F1}_{14}
Z^{F1}_{21} Z^{F1}_{22} Z^{F1}_{23} Z^{F1}_{24}
Z^{F1}_{31} Z^{F1}_{32} Z^{F1}_{33} Z^{F1}_{34}
Z^{F1}_{41} Z^{F1}_{42} Z^{F1}_{43} Z^{F1}_{44}
激活函数
α^{F1}_{11} α^{F1}_{12} α^{F1}_{13} α^{F1}_{14}
α^{F1}_{21} α^{F1}_{22} α^{F1}_{23} α^{F1}_{24}
α^{F1}_{31} α^{F1}_{32} α^{F1}_{33} α^{F1}_{34}
α^{F1}_{41} α^{F1}_{42} α^{F1}_{43} α^{F1}_{44}
池化
α^{F1}_{11} α^{F1}_{12} α^{F1}_{13} α^{F1}_{14}
α^{F1}_{21} α^{F1}_{22} α^{F1}_{23} α^{F1}_{24}
α^{F1}_{31} α^{F1}_{32} α^{F1}_{33} α^{F1}_{34}
α^{F1}_{41} α^{F1}_{42} α^{F1}_{43} α^{F1}_{44}
降维，提取最大值
池化层
α^{F1}_{11} α^{F1}_{21}
α^{F1}_{32} α^{F1}_{42}
输出层识别

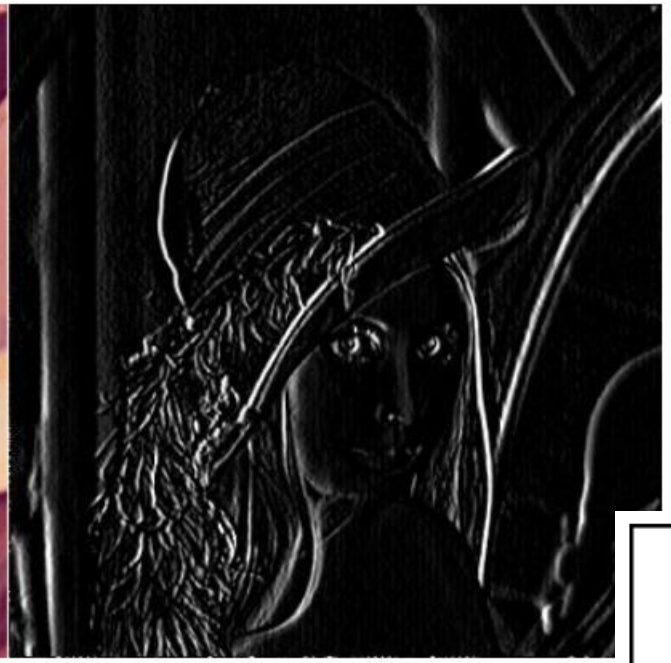
参数共享


特征图(feature map)
2 1 0 1
0 0 1 2
0 0 3 0
0 3 1 1
卷积核(Filter)
共享参数W
+b=
卷积层(加权输入)
激活函数
池化
2×2
池化层
输出层识别
降维，提取最大值

不同的卷积核，可以实现边缘检测、图像锐化、快速均值模糊、高斯模糊

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

求卷积有何用?



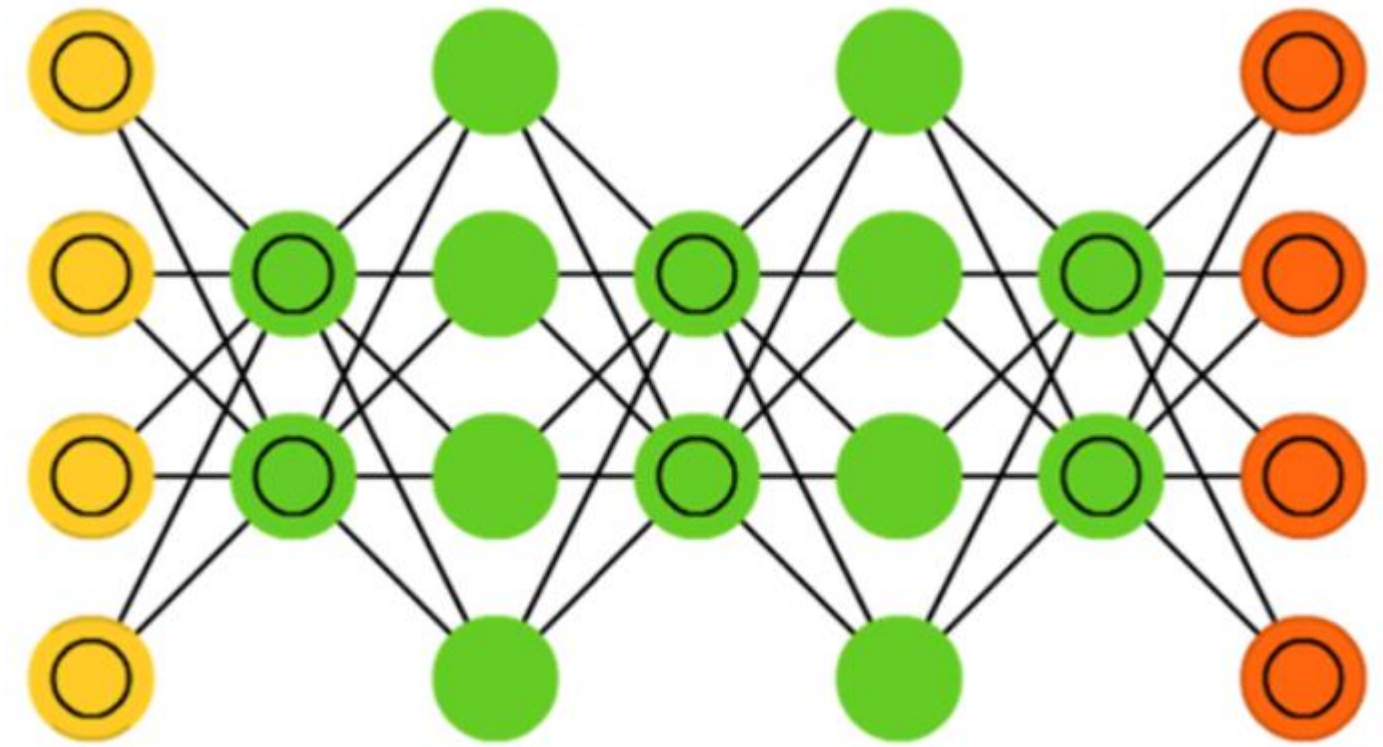
-1	0	1
-2	0	2
-1	0	1

知名的Lena照片

索伯滤波器是是边缘检测器。

深度学习三大模型

1、DBN：Deep Belief Network, 深度信念网络
2、CNN：Convolution Neural Network卷积神经网络
3、SAE：Sparse Auto Encoder，稀疏自编码

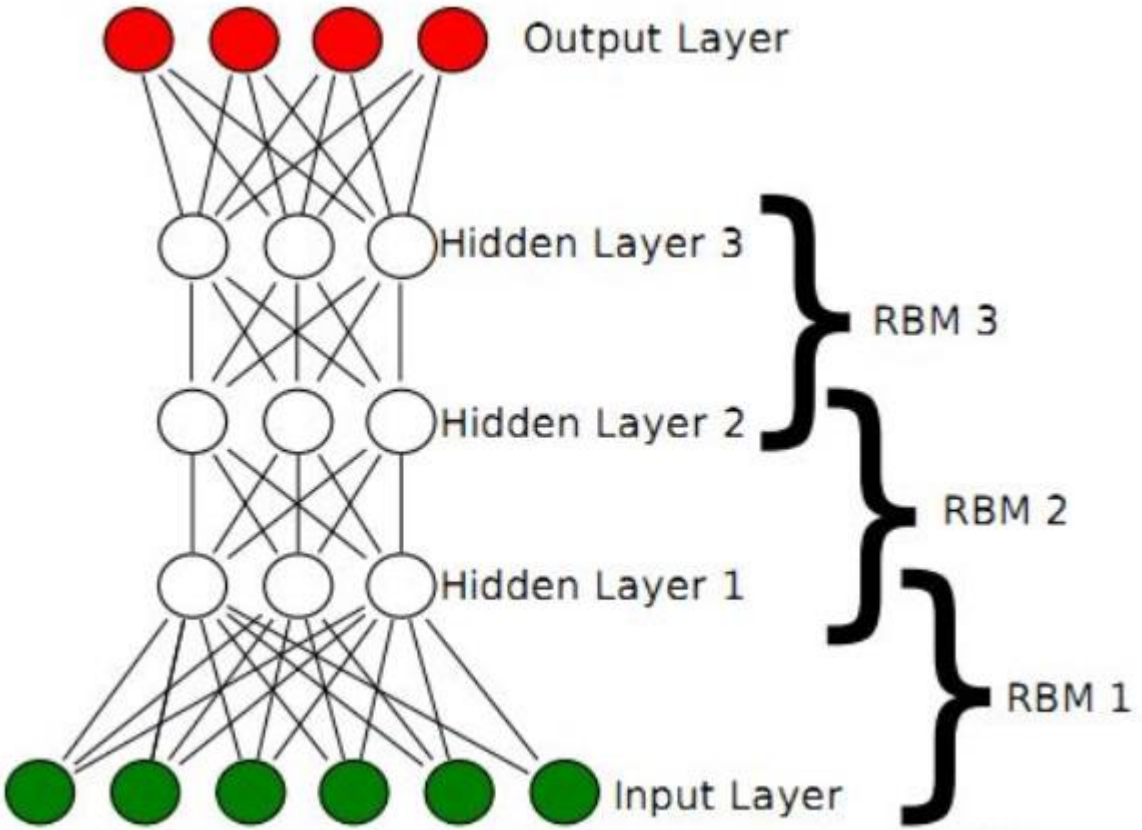


手写数字体识别实验

MNIST数据集中的每个图像都是28x28像素，包含一个居中的灰度数字。

MNIST训练集训练

```
# 28 * 28 = 784
input_nodes = 784
hidden_nodes = 200
output_nodes = 10

# learning rate is 0.3
learning_rate = 0.1

# create instance of neural network
n = neuralNetwork(input_nodes,hidden_nodes,output_nodes,learning_rate)
```


Output Layer
Hidden Layer 3
Hidden Layer 2
Hidden Layer 1
Input Layer
RBM 3
RBM 2
RBM 1

创建的神经网络--mnist手写数字识别

一、加载所需要的库

二、定义神经网络类

三、创建神经网络对象并用MNIST训练集训练

四、用测试集测试准确率

五、参数调优过程记录

六、测试下自己绘制的字体图片识别效果

加载所需要的库

```
# Code for a 3-layer neural network, and code for learni
# Zhouxw@ebscn.com,2018.8  Studying to write neural netw
# license is GPLv2

import numpy
# scipy.special for the sigmoid function expit()
import scipy.special
import matplotlib.pyplot
# ensure the plots are inside this jupyter notebook, not
%matplotlib inline

# helper to load data from PNG image files
import imageio
# glob helps select multiple files using patterns
import glob
```

二、定义神经网络类

```
# neural network class definition  (3 layers)
class neuralNetwork:
    # initialise the neural network
    def __init__(self,inputnodes,hiddennodes,outputnodes,learningrate):
        # set number of nodes in each input,hidden,output layer
        self.inodes = inputnodes
        self.hnodes = hiddennodes
        self.onodes = outputnodes
        # learning rate
        self.lr = learningrate

        # link weight matrices ,wih and who
        # weithg inside the arrays are w_i_j, where link is from node i to node j in the next layer
        # w11 w21
        # w12 w22 etc
        self.wih = (numpy.random.normal(0.0, pow(self.hnodes,-0.5), (self.hnodes,self.inodes) )  )
        self.who = (numpy.random.normal(0.0, pow(self.onodes,-0.5), (self.onodes,self.hnodes) )  )

        # activation function is the sigmoid function
        self.activation_function = lambda x: scipy.special.expit(x)

        pass
```

```
25      def train(self,inputs_list,targets_list):
26          # convert inputs list to 2d array
27          inputs = numpy.array(inputs_list,ndmin=2).T
28          targets = numpy.array(targets_list,ndmin=2).T
29
30          # calculate signals into hidden layer
31          hidden_inputs = numpy.dot(self.wih,inputs)
32          # calculate the signals emerging from hidden layer
33          hidden_outputs = self.activation_function(hidden_inputs)
34
35          # calculate signals into final output layer
36          final_inputs = numpy.dot(self.who, hidden_outputs)
37          # calculate the signals emerging from final output layer
38          final_outputs = self.activation_function(final_inputs)
39
40          # output layer error is the (target-actual)
41          output_errors = targets - final_outputs
42          # hidden layer error is the output_errors,split by weights,recombined at hidden nodes
43          hidden_errors = numpy.dot(self.who.T, output_errors)
44
45          # update the weights for the links between the hidden and output layers
46          self.who += self.lr * numpy.dot((output_errors * final_outputs * (1.0 - final_outputs)), numpy.transpose(hidde
47
48          # update the weights for the links between the input and hidden layers
49          self.wih += self.lr * numpy.dot((hidden_errors * hidden_outputs * (1.0 - hidden_outputs)), numpy.transpose(inp
```

```
53      # query the neural network
54      def query(self,inputs_list):
55          # convert inputs list to 2d array
56          inputs = numpy.array(inputs_list,ndmin=2).T
57
58          # calculate signals into hidden layer
59          hidden_inputs = numpy.dot(self.wih,inputs)
60          # calculate the signals emerging from hidden layer
61          hidden_outputs = self.activation_function(hidden_inputs)
62
63          # calculate signals into final output layer
64          final_inputs = numpy.dot(self.who, hidden_outputs)
65          # calculate the signals emerging from final output layer
66          final_outputs = self.activation_function(final_inputs)
67
68          return final_outputs
```

三、创建神经网络对象并用MNIST训练集训练

```
2  # 28 * 28 = 784
3  input_nodes = 784
4  hidden_nodes = 200
5  output_nodes = 10
6
7  # learning rate is 0.3
8  learning_rate = 0.1
9
10 # create instance of neural network
11 n = neuralNetwork(input_nodes,hidden_nodes,output_nodes,learning_rate)
```

```
# load the mnist training data csv file into a list
training_data_file = open("mnist_dataset/mnist_train.csv",'r')
training_data_list = training_data_file.readlines()
training_data_file.close()

# epochs is the number of times the training data set is used for training
epochs = 5
for e in range(epochs):
    # go through all records in the training data set
    for record in training_data_list:
        all_values = record.split(',')
        # scale and shift the inputs
        inputs = (numpy.asfarray(all_values[1:]) / 255.0 * 0.99) + 0.01
        # create the target output values (all 0.01, except the desired label which is 0.99)
        targets = numpy.zeros(output_nodes) + 0.01
```

四、用测试集测试准确率

```
test_data_file = open("mnist_dataset/mnist_test.csv",'r')
test_data_list = test_data_file.readlines()
test_data_file.close()

# scorecard for how well the network performs,initially empty
scorecard = []
# go through all records in the test data set
for record in test_data_list:
    all_values = record.split(',')
    # correct answer is first value
    correct_label = int(all_values[0])
    # scale and shift the inputs
    inputs = (numpy.asfarray(all_values[1:]) / 255.0 * 0.99) + 0.01
    # query the network
    outputs = n.query(inputs)
    # the index of the highest value corresponds to the label
    label = numpy.argmax(outputs)
#    print("Answer label is:",correct_label," ; ",label," is network's answer"
    # append correct or incorrect to list
    if(label == correct_label):
        # network's answer matches correct answer, add 1 to scorecard
        scorecard.append(1)
    else:
        scorecard.append(0)
    pass
```

五、参数调优过程记录

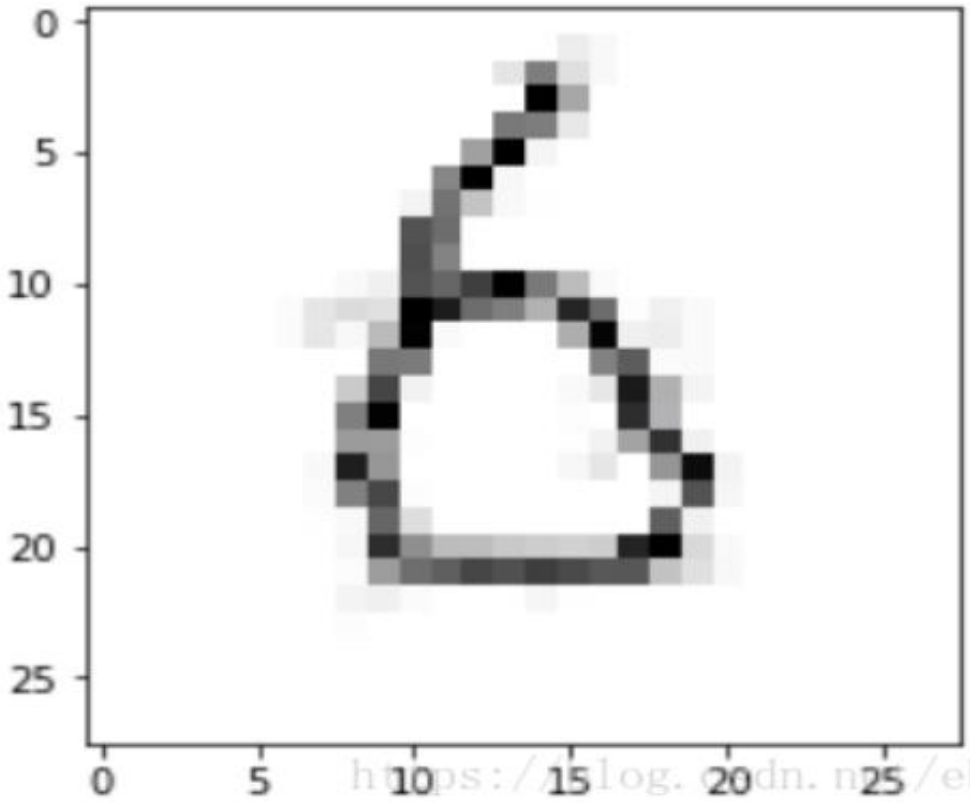
2	有效的参数调优说明				
3					
4	学习率	训练轮数	隐藏层节点	结果准确率	说明
5	0.3	1	100	0.9473	初始经验，效果还不错。
6	0.6	1	100	0.9047	学习率再增加到0.6，测试准确率下
7	0.1	1	100	0.9502	降低学习率到0.1，准确率增加。
8	0.01	1	100	0.9241	更低的学习率也不行，应该是限制
9	0.2	1	100	0.9515	学习率调到0.2为最优
10	0.2	5	100	0.9611	5~7轮迭代是比较好的经验值。测试
11	0.1	5	100	0.9653	增加训练轮数，可适当降低学习率，
12	0.1	5	200	0.9723	增加影藏层节点数，神经网络有更好
13	0.1	5	500	0.9751	这个结果已经非常好了！

六、测试下自己绘制的字体图片识别效果（28*28）

```
our_own_dataset = []

# load the png image data as test data set
for image_file_name in glob.glob('my_own_images/2828_my_own_?.png'):

    # use the filename to set the correct label
    label = int(image_file_name[-5:-4])

    # load image data from png files into an array
    print ("loading ... ", image_file_name)
    img_array = imageio.imread(image_file_name, as_gray=True)

    # reshape from 28x28 to list of 784 values, invert values
    img_data  = 255.0 - img_array.reshape(784)

    # then scale data to range from 0.01 to 1.0
    img_data = (img_data / 255.0 * 0.99) + 0.01
    print(numpy.min(img_data))
    print(numpy.max(img_data))

    # append label and image data  to test data set
    record = numpy.append(label,img_data)
    our_own_dataset.append(record)

    pass
```

结果样子如下：

```
network says  6
Good,match!
```



作业：

MNIST 数据集可在 http://yann.lecun.com/exdb/mnist/ 获取, 它包含了四个部分:

- Training set images: train-images-idx3-ubyte.gz (9.9 MB, 解压后 47 MB, 包含 60,000 个样本)
- Training set labels: train-labels-idx1-ubyte.gz (29 KB, 解压后 60 KB, 包含 60,000 个标签)
- Test set images: t10k-images-idx3-ubyte.gz (1.6 MB, 解压后 7.8 MB, 包含 10,000 个样本)
- Test set labels: t10k-labels-idx1-ubyte.gz (5KB, 解压后 10 KB, 包含 10,000 个标签)

MNIST 数据集来自美国国家标准与技术研究所, National Institute of Standards and Technology (NIST). 训练集 (training set) 由来自 250 个不同人手写的数字构成, 其中 50% 是高中学生, 50% 来自人口普查局 (the Census Bureau) 的工作人员. 测试集(test set) 也是同样比例的手写数字数据.

谢谢！