《计算机视觉-openCV应用技术》

第九章 基于OpenCV的神经网络

李策

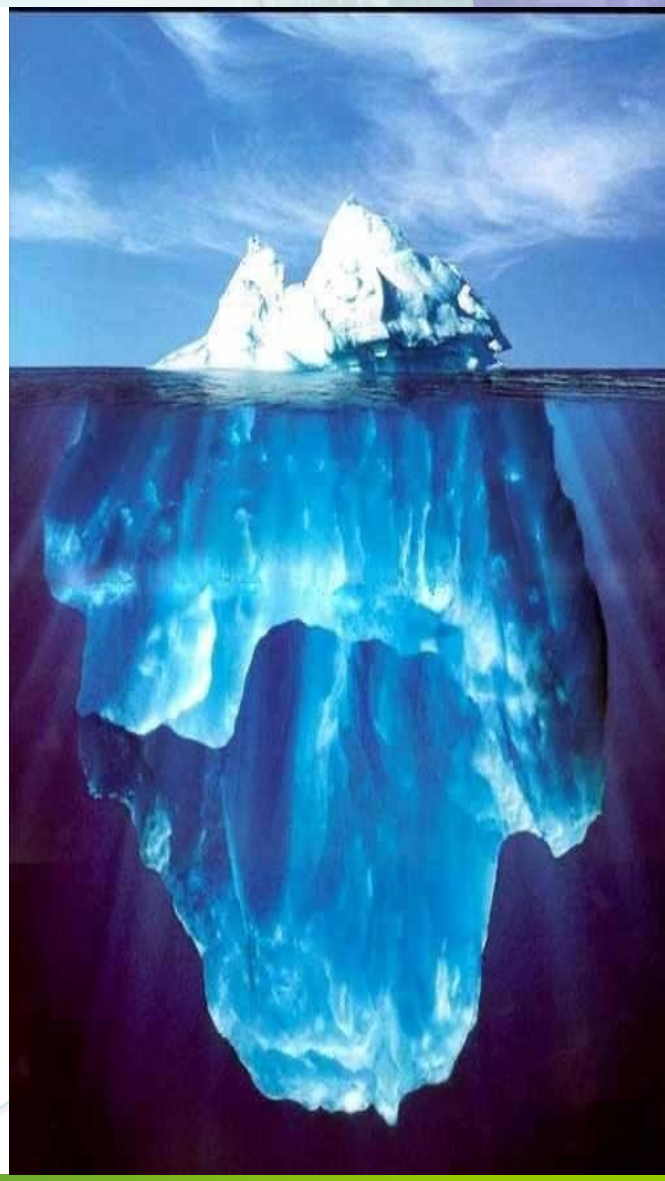
中国矿业大学（北京）计算机科学与技术系

E-mail: celi@cumtb.edu.cn

提纲

第九章 基于OpenCV的神经网络

- 9.1 人工神经网络
- 9.2 人工神经网络的结构
  - 9.2.1 网络层级示例
  - 9.2.2 学习算法
- 9.3 OpenCV中的ANN
  - 9.3.1 基于ANN的动物分类
  - 9.3.2 训练周期
- 9.4 用人工神经网络进行手写数字识别
- 9.5 可能的改进和潜在的应用
- 9.6 总结



9.1 人工神经网络

什么是人工神经网络（ANN)？

ANN是一个统计模型

统计模型是一对元素：

- 空间S（观测数据集）
- 概率P

ANN是如何改进简单统计模型的？生成该数据集的函数很可能需要大量的输入怎么办？

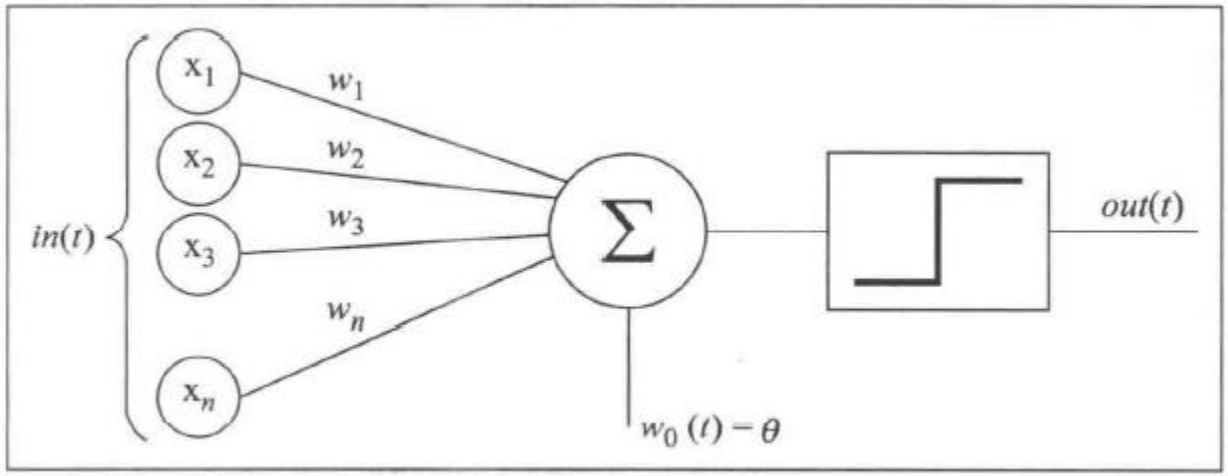
- 将任务分配给许多神经元
- 每一个神经元都能够“近似”生成输入的函数

9.1 人工神经网络

神经元与感知器

感知器是接收许多输入并输出一个值的函数

每个输入都有一个与之相关联的权重。Sigmoid函数会输出一个值：



Sigmoid函数用来指示该函数的值是0还是1。

判断条件是一个阈值，输入的权重和大于某一阈值，输出为1，否则为0

9.2.1 网络层级示例

如图所示

神经网络有三层：

- 输入层

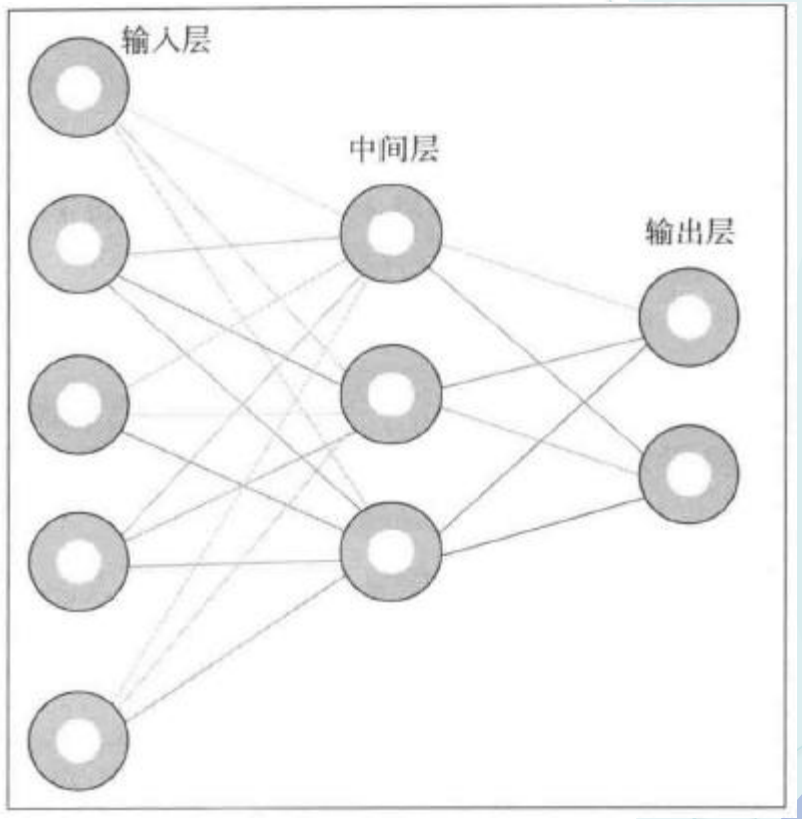
  网络的输入数目

- 输出层

  数目与定义的类别数相同

- 隐藏层（或中间层）

  包含感知器



9.2.1 网络层级示例

创建ANN最常见的规则如下：

- 隐藏层神经元数应该介于输入层的大小与输出层的大小之间
- 对于相对较小的隐藏层，隐藏层神经元数最好是输入层和输出层大小之和的三分之二，或者小于输入层大小的两倍

需要注意的现象：过拟合

当待分类训练数据提供的信息无意义，而隐藏层又含有太多信息就会发生过拟合

9.2.2 学习算法

人工神经网络可使用多种学习算法，有三个主要的算法：

- 监督算法：从ANN获得函数，描述标记的数据
- 非监督算法：与监督学习不同，这类算法的数据没有标记
- 强化学习：系统接受输入，决策机制决定决策行为，执行该机制并给出相应的评分，输入和动作要与评分相匹配

9.3 OpenCV 中的 ANN

- 创建ANN

```
ann = cv2.ml.ANN_MLP_create()
```

- 创建完网络之后，设置相应的拓扑结构：

```
ann.setLayerSizes(np.array([9, 5, 9], dtype=np.uint8))
```

训练采用反向传播方式：

- BACKPOP
- RPROP

这两种算法可用在有监督学习的场合

9.3.1 基于ANN的动物分类

根据统计量（体重、长度、牙齿）对动物进行分类

- 导入模块randint

  from random import randint

- 创建ANN，设定train函数为弹性反向传播，激活函数为sigmoid函数

  animals_net = cv2.ml.ANN_MLP_create()

  setTrainMethod

  setActivationFunction

- 指定ANN的终止条件：

  setTermCriteria

- 定义四种创建样本的函数和四种分类函数

9.3.2 训练周期

另一个ANN训练的重要概念：周期

一个训练周期表示训练数据的一次迭代，在这之后，是对数据进行分类测试

- 前面的例子进行修改
- 从dog类开始测试
- 重复所有类，输出结果

此例使用了无实用价值的数据，并且只考虑了训练数据的大小/训练迭代次数

通过结果可以看到ANN对哪些类产生了过拟合

9.4 用人工神将网络进行手写数字识别

9.4.1 MINIST - 手写数字数据库

MINIST数据库是Web上非常流行的OCR和手写字符识别和分类器训练资源，通过它可以创建用于识别手写数字的人工网络的程序。

9.4.2 定制训练数据

- 创建自己的训练数据
- 收集大量的手写数字
- 创建含有单个数字的图像
- 创建一个机制来保证训练样本与预期分类同步

9.4.3 初始数据

A.　输入层

MINIST数据库

每幅图像大小为28×28，即784像素

B.　隐藏层

经过多次尝试，50至60个节点会得到最好的结果

C.　输出层

大小为10（0~9），需要10个节点

9.4.4 迭代次数

最初使用MINIST提供的整个训练数据集（60000个手写图像）

只需要一次迭代就能得到可接受的高检测精度

现在，将由用户反复训练基于同一训练数据的神将网络，在训练过程中对精度进行测试

9.4.5 其他参数

使用sigmoid激活函数、弹性反馈，并延长终止条件

通过设置cv2.TermCriteria将每次计算的迭代次数由10次变为20次

9.4.6 迷你库

为了尽可能地自动执行，需要建立一个迷你库，用来封装ANN在OpenCV中的原始实现，这样会使重新训练神经网络变得容易。

这里是一个封装的例子：见书上P172

代码思路：

Load_data、wrap_data和vectorized_result函数

都包含在用于加载pickle文件的代码中

加载的数据分为

- Train：训练ANN
- Test：评估ANN的准确性

都含有两类数据：数据本身和类标签

9.4.6 迷你库

该函数的重要部分：

- 将单独的训练记录分解为train数据和相应的类标签
- 传递到 ANN
- 利用numpy数组的ravel()函数，获得任意形状的数组
- 返回network和test这两组数据
- 封装 ANN 的 predict() 函数

9.4.7 主文件

完成常规模块加载后，通过digits_ann.py导入所创建的迷你库

A.定义函数，inside（）函数用来确定矩阵是否完全包含在另一个矩阵中：

```
def inside(r1, r2):
```

B.Wrap_digits（）函数获取数字周围的矩形，将其转换为正方形，对其中心化：

```
def wrap_digit(rect):
```

C.创建神经网络：

```
ann, test_data = ANN.train(ANN.create_ANN(58), 20000)
```

9.4.7 主文件

D.迭代多次，直到收敛

```
ann, test_data = ANN.train(ANN.create_ANN(100), 50000, 30)
```

E.使用阈值和形态学操作方法来确保数字能从背景中正确区分出来：

```
ret, thbw = cv2.threshold(bw, 127, 255, cv2.THRESH_BINARY_INV)
thbw = cv2.erode(thbw, np.ones((2,2), np.uint8), iterations = 2)
```

F.识别图像中的轮廓

```
image, cntrs, hier = cv2.findContours(thbw.copy(),
cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

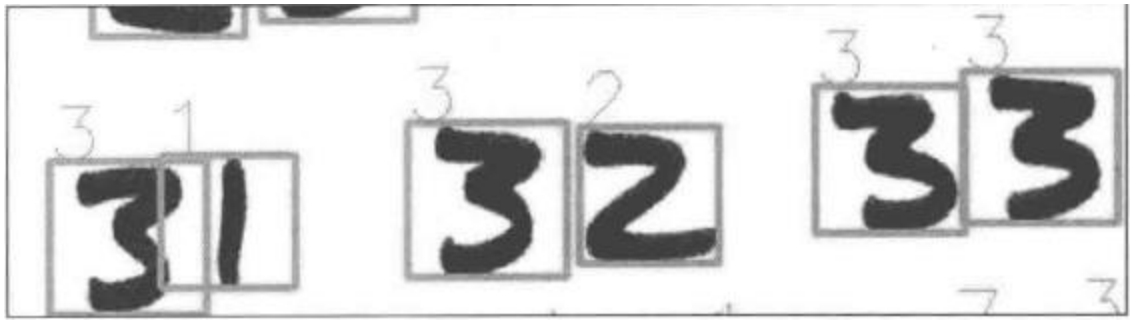
G.通过轮廓迭代，添加不包含在其他矩形中和不超过图像宽度的好的矩形

H.已经识别的矩形定义为感兴趣区域：

```
for r in rectangles:
x,y,w,h = wrap_digit(r)
```

9.4.6 迷你库

最后的结果：



9.5 可能的改进和潜在的应用

改进：

- 扩大数据集，进行多次迭代
- 采用不同的激励函数
- 利用不同的训练标志

实际应用中，软件开发的经验也尤为重要，认真分析应用需求有助于选择最佳的参数

9.5.2 应用

上面的程序仅仅是手写识别应用的基础，还可以应用到：

- 早期的视频和实时手写数字检测
- 训练ANN为成熟的OCR系统识别整个字母表
- 扩展到车牌检测
- 采用ANN和SVM来建立分类器，查看检测效果

9.5.2 总结

本章介绍了人工神经网络的基本原理

- 采用OpenCV实现的ANN
- 初步了解了ANN的结构
- 基于应用需求来设计神经网络的拓扑结构
- 建立了一个手写数字识别应用