北京师范大学
人工智能学院

数据挖掘课程 习题讲解 三

课堂练习5 讲解

&

实验4 实验5 讲解

讲解人：王舒扬

课堂练习5

完成时间：2023年4月28日

01 课堂练习5

2.单选题 (1分)

以下关于ID3算法局限性或者特性的说法,错误的选项是()

A 没有考虑连续特征

B 采用信息增益大的特征优先建立决策树的节点

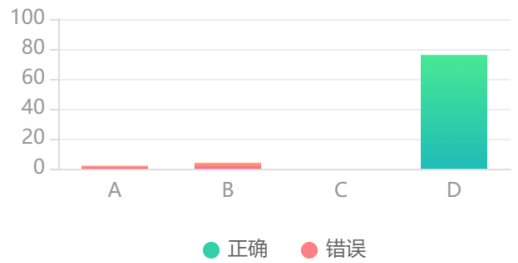
C 没有考虑缺失值的情况

D 考虑了过拟合的问题

正确答案：D

答题统计 (统计数据中的人数，为已交卷人数)

82/95	86.32%	92.68%
完成人数	完成率	正确率



ID3是决策树的常用算法。

有以下缺点：

1. ID3没有**考虑连续特征**，比如长度，密度都是连续值，无法在ID3运用。（A）
2. ID3**采用信息增益大的特征优先建立决策树的节点。**（B）在相同条件下，比如一个特征有2个分支，另一个特征为3个分支，取值比较多的特征比取值少的特征信息增益大。其实他们都是求不确定的变量，但是3个取值的比2个取值的信息增益大。
3. ID3算法**对于缺失值的情况没有做考虑**（C）
4. **没有考虑过拟合的问题**（D）

01 课堂练习5

4.单选题 (1分)

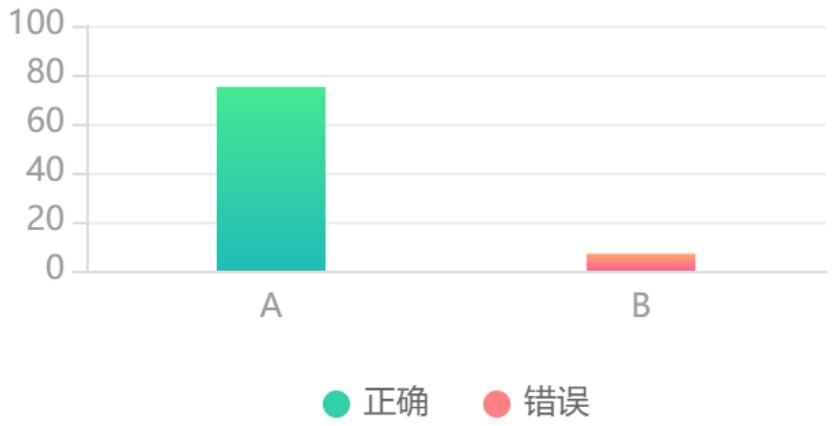
一般情况,CART需要计算χ2值来判断分类条件和类别的相关程度,决定是否停止分裂

A 正确

B 错误

正确答案：A

82/95 完成人数 | 86.32% 完成率 | 91.46% 正确率



CART：一种非常重要的二叉决策树，属于Top10 Machine Learning Algorithm。

CART算法既可以用于创建分类树（Classification Tree），也可以用于创建回归树（Regression Tree）、模型树（Model Tree），两者在建树的过程稍有差异。

✓什么时候节点就可以停止分裂了？

●直观的情况：当节点包含的数据记录都属于同一个类别时就可以终止分裂了。这只是一个特例。

●更一般的情况：计算χ2值来判断分类条件和类别的相关程度。当χ2很小时说明分类条件和类别是独立的，即按照该分类条件进行分类是没有道理的，此时节点停止分裂。注意这里的“分类条件”是指按照GINI_Gain最小原则得到的“分类条件”。

01 课堂练习5

9.单选题 (1分)

以下哪一项关于决策树的说法是错误的（）

A 冗余属性不会对决策树的准确率造成不利的影响

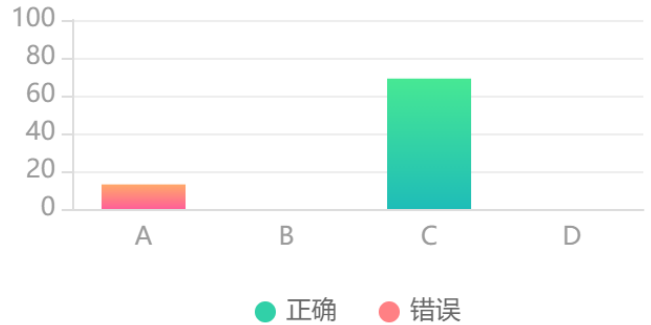
B 子树可能在决策树中重复多次

C 决策树算法对于噪声的干扰非常敏感

D 寻找最佳决策树是NP完全问题

正确答案：C

82/95	86.32%	84.15%
完成人数	完成率	正确率



决策树的特点：

1. **冗余属性不会对决策树的准确率造成不利影响。**一个数据如果在数据中它与另一个属性是强相关的，那么它是冗余的。在两个冗余的属性中，如果已经选择其中一个作为划分属性，则另一个将会被忽略。（A）

2. **决策树对于噪声的干扰有较好的鲁棒性**，采用避免过拟合的方法之后尤其如此（C）

3. **找到最佳决策树是NP完全问题。**许多决策树算法都采取启发式的方法指导对假设空间的搜索。我们常用一种贪心的、自顶向下的递归划分策略建立决策树。（D）

4. **子树可能在决策树中重复多次。**当决策树的每个内部结点都依赖单个属性测试条件时，就会出现这种情形。由于大多数的决策树算法都采用分治划分策略，因此在属性空间的不同部分可以使用相同的测试条件，从而导致子树重复问题。（B）

01 课堂练习5

11.单选题 (1分)

术语“假正(false positive)”的含义是()

A 分类器正确标记的正元组

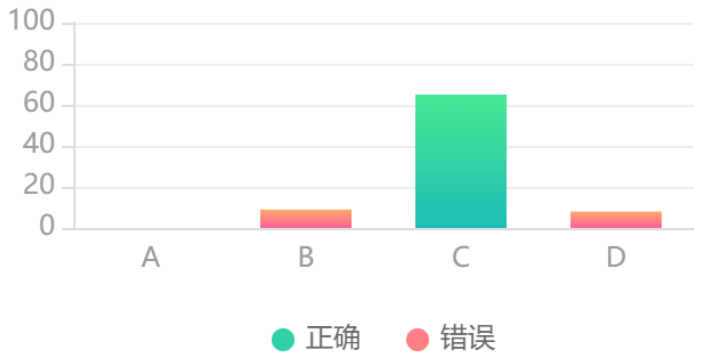
B 分类器正确标记的负元组

C 错误标记的负元组

D 错误标记的正元组

正确答案：C

82/95	86.32%	79.27%
完成人数	完成率	正确率



几个术语

正元组（感兴趣的主类元组，如buy_computer = yes）

负元组（如 buy_computer = no）

真正（true positive，TP）：分类器正确标记的正元组

真负（true negative, TN）：分类器正确标记的负元组

假正（false positive, FP）：错误标记的负元组

如： buy_computer = no 的元组，分类器预测为 buy_computer = yes

假负（false negative,FN）：错误标记的正元组

如： buy_computer = yes的元组，分类器预测为 buy_computer = no

PART 02

实验4 + 实验5

实验四：频繁模式挖掘

- 实验内容：掌握频繁模式挖掘的一些定义，知道如何运用Python计算支持度，置信度等；掌握mlxtend库调用Apriori算法和FP-Growth算法

实验四：第1、2问

- 第一问：在“啤酒与尿布”数据集中计算bread ⟶ milk的置信度
- 参考答案：

```
flag_ante = df["item"].apply(is_subset, setB=({'milk', 'bread'}))
count_ante = df[flag_ante].shape[0]
flag_conq = df["item"].apply(is_subset, setB=({'bread'}))
count_conq = df[flag_conq].shape[0]
conf = count_ante/count_conq
print("Confidence is {:.2f}".format(conf))
```

```
Confidence is 0.75
```

分别计算”milk, bread”和”bread”的支持度

计算比值得到置信度

- 第二问：sklearn中哪个模块可以用来创建one-hot编码？
- 参考答案：OneHotEncoder

实验四：第3问

- 第三问：movies.csv是近年来冯小刚导演的电影主演演员表，请用频繁模式挖掘的方法回答，冯小刚导演喜欢合作的两位演员分别是谁？喜欢启用的演员组合是哪个？
- 参考答案：数据预处理：将数据转化为one-hot形式

```
df = pd.read_csv('movies.csv').T
data = df.values
data = list(data)
for i in range(len(data)):
    data[i] = list(data[i])
    while np.nan in data[i]:
        data[i].remove(np.nan)
    for j in range(len(data[i])):
        if ' ' in data[i][j]:
            data[i][j] = data[i][j].replace(' ', '')
data[-3:]
```

读取数据
删除“nan”数据
将名字中的空格删除

```
[['徐帆', '冯远征', '修宗迪', '叶京', '梁丹妮', '马玉良', '秦焰'],
 ['范伟'],
 ['赵宝刚', '温海涛', '徐帆', '刘小宁']]
```

```
# 转化为onehot
te = TransactionEncoder()
data = te.fit_transform(data)
df_apriori = pd.DataFrame(data, columns=te.columns_)
df_apriori[:3]
```

转化为one-hot形式，多个函数可以实现，选一种即可

	于和伟	任泉	何冰	保罗·马祖斯基	修宗迪	傅彪	关之琳	冯远征	刘威	刘小宁
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False

实验四：第3问

- 参考答案：修改最小支持度得到最爱用的演员和组合

```
# 根据最小支持度得到所需结果，徐帆最爱用，（徐帆，葛优）这个组合最爱用
frequent_itemsets = apriori(df_apriori, min_support=0.2, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

调用函数，设置最小支持度，得到结果

	support	itemsets	length
0	0.590909	(徐帆)	1
1	0.500000	(葛优)	1
2	0.272727	(葛优, 徐帆)	2

实验四：常见问题

- 第三问中对数据进行预处理的时候，需要将数据中演员名字中的空格删去，否则结果发生错误，下图为不删除空格时的结果

```
frequent_itemsets = apriori(df_apriori, min_support=0.05, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets[frequent_itemsets['length']==1]
```

	support	itemsets	length
0	0.136364	(修宗迪)	1
1	0.090909	(傅彪)	1
2	0.136364	(刘蓓)	1
3	0.090909	(廖凡)	1
4	0.181818	(张国立)	1
5	0.136364	(张涵予)	1
6	0.090909	(徐帆)	1
7	0.500000	(徐帆)	1
8	0.136364	(李诚儒)	1
9	0.090909	(王宝强)	1
10	0.090909	(秦焰)	1

个人感想优秀案例

1. 掌握了关联分析的一些定义，知道如何运用 python 计算支持度，置信度等，同时了解了 Apriori 算法和 FP-Growth 算法。对关联分析有了更清晰的认知。

2. 在做第二题的时候，没有用到 sklearn 中的 OneHotEncoder 模块，主要是不能将各个词单独分出来进行独热代码的构建，于是使用 CountVectorizer 解决了该问题。

3. 最后一题本来是想将两问分开做，因为第一问的最初思路是遍历统计次数，后来一想，其实就是项集的长度不同而已，本质上没有什么区别，这样思路就比较清晰了。

实验五：人工神经网络

- 实验内容： 理解神经网络基本原理，学会使用pytorch利用神经网络分类

实验五：第1问

- 第一问：列举三种常用的激活函数，写出其表达式，并编写程序，计算它们的导数值
- 参考答案：

```python
import math
# 1. Sigmoid函数
def Sigmoid(x):
    return 1 / (1 + math.exp(- x))

def Sigmoid_grad(x):
    return Sigmoid(x)*(1-Sigmoid(x))

# 3. ReLU函数
def ReLU(x):
    return max(0, x)

def ReLU_grad(x):
    return 1 if x >0 else 0
```

```python
# 2. Tanh函数
def Tanh(x):
    return (math.exp(x) - math.exp(-x))/(math.exp(x) + math.exp(-x))

def Tanh_grad(x):
    return 1 - Tanh(x) ** 2
```

激活函数的更详细的资料：
https://www.cnblogs.com/saseng/p/13647606.html

实验五：第2问

- 第二问：构造双隐层神经网络，对FashionMNIST数据进行分类，迭代20次，查看在预测集上的精度
- 参考答案：

```
# 定义网络结构
self.fc = nn.Sequential(nn.Linear(self.num_inputs, self.num_hiddens),
                        nn.ReLU(),
                        nn.Linear(self.num_hiddens, self.num_hiddens),
                        nn.ReLU(),
                        nn.Linear(self.num_hiddens, self.num_outputs))
```

在源代码中多添加一个中间层

```
# 加载模型
path = "model-{}".format(num_epochs)
net = torch.load(path)
acc_sum, n = 0.0, 0
net.eval()
with torch.no_grad():
    for X, y in test_iter:
        acc_sum += (net(X).argmax(dim=1) == y).float().sum().item()
        n += len(y)
print("test_acc:", acc_sum / n)
```

```
test_acc: 0.8747
```

```
train(net, train_iter, test_iter, loss, num_epochs, optimizer)
```

```
epoch 1, loss 0.0044, train acc 0.566
epoch 2, loss 0.0021, train acc 0.797
epoch 3, loss 0.0018, train acc 0.828
epoch 4, loss 0.0016, train acc 0.846
epoch 5, loss 0.0016, train acc 0.853
epoch 6, loss 0.0014, train acc 0.864
epoch 7, loss 0.0013, train acc 0.873
epoch 8, loss 0.0013, train acc 0.877
epoch 9, loss 0.0012, train acc 0.883
epoch 10, loss 0.0012, train acc 0.886
epoch 11, loss 0.0012, train acc 0.889
epoch 12, loss 0.0011, train acc 0.892
epoch 13, loss 0.0011, train acc 0.896
epoch 14, loss 0.0011, train acc 0.898
epoch 15, loss 0.0010, train acc 0.901
epoch 16, loss 0.0010, train acc 0.902
epoch 17, loss 0.0010, train acc 0.906
epoch 18, loss 0.0010, train acc 0.908
epoch 19, loss 0.0009, train acc 0.908
epoch 20, loss 0.0009, train acc 0.913
```

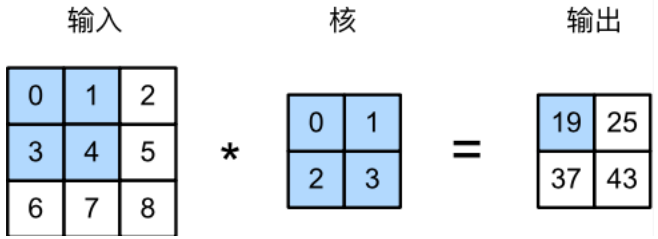
实验五：第三问

- 3. 查阅关于卷积神经网络的资料，回答并完成下面的问题及练习：
- 什么是卷积核?
- 为什么使用卷积核?
- 利用经典的VGG网络结构对FashionMNIST数据集进行分类

实验五：第三问

- 参考答案：

1. 卷积核也称为filter，类似数学中的卷积操作，是一个数组，用于数据特征提取。



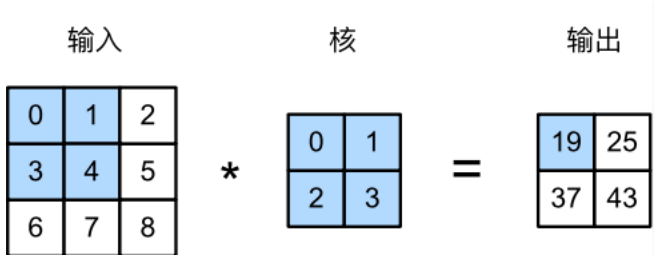
$$
\begin{aligned}
0\times 0+1\times 1+3\times 2+4\times 3&=19,\\
1\times 0+2\times 1+4\times 2+5\times 3&=25,\\
3\times 0+4\times 1+6\times 2+7\times 3&=37,\\
4\times 0+5\times 1+7\times 2+8\times 3&=43.
\end{aligned}
$$

实验五：第三问

- 参考答案：

2. 全连接网络的参数过多，卷积核通过对图像局部信息的加权求和与权重共享的策略（使用同一个核），有效的降低了参数。

以下面的计算为例，输入3×3的数组，希望得到2×2的输出，那么对于全连接层来说需要$3 \times 3 \times 2 \times 2 = 36$个参数，而卷积只需要4个。



实验五：第三问

- 3.

```
class FlattenLayer(nn.Module):
    def __init__(self):
        super(FlattenLayer, self).__init__()

    def forward(self, input):
        return input.reshape(input.shape[0], -1)

class VGG_Block(nn.Module):
    def __init__(self, num_conv, in_channels, out_channels, p=0):
        super(VGG_Block, self).__init__()
        self.net = nn.ModuleList()
        self.net.append(
            nn.Conv2d(in_channels=in_channels, out_channels=out_channels, kernel_size=(3, 3), padding=(1, 1),
                      stride=(1, 1)))
        self.net.append(nn.BatchNorm2d(out_channels))
        self.net.append(nn.Tanh())
        self.net.append(nn.Dropout(p))
        for num in range(1, num_conv):
            self.net.append(
                nn.Conv2d(in_channels=out_channels, out_channels=out_channels, kernel_size=(3, 3), padding=(1, 1),
                          stride=(1, 1)))
            self.net.append(nn.BatchNorm2d(out_channels))
            self.net.append(nn.Tanh())
            self.net.append(nn.Dropout(p))
        self.net.append(nn.MaxPool2d(kernel_size=(2, 2), stride=(2, 2)))

    def forward(self, input):
        for net in self.net:
            input = net(input)
        return input
```

将图片拉平，例如64x3x32x32的图片被拉平为64x3072

添加卷积核

实验五：第三问

```
class VGG(nn.Module):
    def __init__(self, fc_features, fc_hidden_unites=4096, p=0):
        super(VGG, self).__init__()
        self.conv = nn.Sequential(VGG_Block(2, 1, 32, p),
                                  VGG_Block(2, 32, 128, p),
                                  VGG_Block(8, 128, 512, p))
        self.fc = nn.Sequential(FlattenLayer(),
                                nn.Linear(fc_features, fc_hidden_unites),
                                nn.Tanh(),
                                nn.Linear(fc_hidden_unites, fc_hidden_unites),
                                nn.Tanh(),
                                nn.Linear(fc_hidden_unites, 10),
                                nn.Tanh(),
                                nn.Softmax())

    def forward(self, input):
        input = self.conv(input)
        input = self.fc(input)
        return input
```

一共三个Block，分别有2，2，8个卷积，共12个

三层全连接层用于分类

实验五：第三问

```
train(net, train_iter, test_iter, loss, num_epochs, optimizer)
```

```
/root/miniconda3/lib/python3.8/site-packages/torch/nn/modules/container
x has been deprecated. Change the call to include dim=X as an argument.
  input = module(input)
epoch 1, loss 0.0080, train acc 0.774
epoch 2, loss 0.0079, train acc 0.875
epoch 3, loss 0.0078, train acc 0.895
epoch 4, loss 0.0078, train acc 0.907
epoch 5, loss 0.0078, train acc 0.916
epoch 6, loss 0.0078, train acc 0.921
epoch 7, loss 0.0078, train acc 0.928
epoch 8, loss 0.0078, train acc 0.932
epoch 9, loss 0.0078, train acc 0.938
epoch 10, loss 0.0078, train acc 0.940
epoch 11, loss 0.0078, train acc 0.945
epoch 12, loss 0.0078, train acc 0.946
epoch 13, loss 0.0077, train acc 0.951
epoch 14, loss 0.0077, train acc 0.953
epoch 15, loss 0.0077, train acc 0.956
epoch 16, loss 0.0077, train acc 0.958
epoch 17, loss 0.0077, train acc 0.959
epoch 18, loss 0.0077, train acc 0.963
epoch 19, loss 0.0077, train acc 0.964
epoch 20, loss 0.0077, train acc 0.965
```

```python
# 加载模型
path = "model-{}".format(num_epochs)
net = torch.load(path)
acc_sum, n = 0.0, 0
net.eval()
with torch.no_grad():
    for X, y in test_iter:
        X = X.to('cuda:0')
        acc_sum += (net(X).cpu().argmax(dim=1) == y).float().sum().item()
        n += len(y)
print("test_acc:", acc_sum / n)
```

```
test_acc: 0.9194
```

常用的深度学习平台：Kaggle，AIC平台（学院）