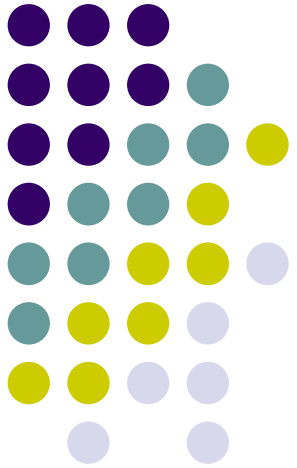
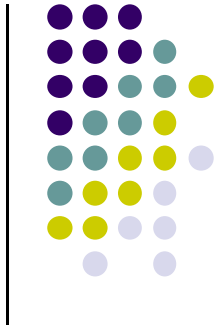
面向科学问题求解的编程实践



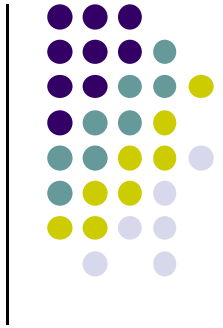
报告要求



没有字数要求，报告文档请转换成pdf格式，和附件一起打包成压缩文件提交，你所提交的压缩文件中应包括：

1. 报告文档(pdf格式，必需)
2. 源代码文件(必需)
3. 其他文件，如运行视频等，如果过大的文件可以发网上链接

报告文档格式

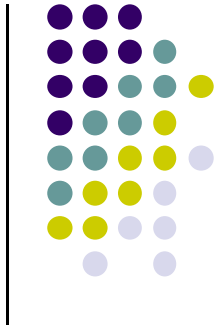


- 实验题目：
- 背景介绍：介绍本次实验的学科背景
- 实验目的：介绍本次实验期望达成的目标
- 实验环境：介绍实验所使用到的开发环境，运行环境，工具，库等
- 实验内容：实验的具体环节，应当包括具体的实验设计，算法的流程等详细信息
- 实验结果：实验结果，应该有相应的数据，运行结果截图等信息
- 总结：实验总结与收获
- 参考资料及文献

Q&A

- 5月24日是否可以交完整报告？
  - 可以
- 5月24日是否需要完成报告？
  - 不需要
- 选题相同是否可以？
  - 可以，不影响评分。但报告不能雷同。
- 是否有答辩环节？
  - 有。不在课上举行，可选择参加。

Logistic回归模型



- 直接预测一个事件的概率

- https://scikit-learn.org/stable/

scikit-learn
Machine Learning in Python

  - Simple and efficient tools for predictive data analysis
  - Accessible to everybody, and reusable in various contexts
  - Built on NumPy, SciPy, and matplotlib
  - Open source, commercially usable - BSD license

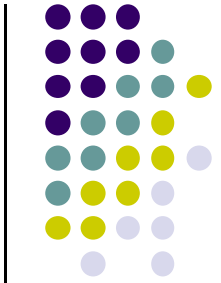
从“泰坦尼克”号生还

- 1912年4月15日清晨，沉没在北大西洋
- 1046位乘客信息
  - 舱位等级、年龄、性别、是否生还、姓名

数据

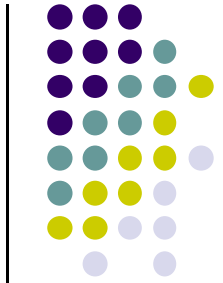
1,29.0,F,1,Allen, Miss. Elisabeth Walton
1,0.92,M,1,Allison, Master. Hudson Trevor
1,2.0,F,0,Allison, Miss. Helen Loraine
1,30.0,M,0,Allison, Mr. Hudson Joshua Creighton
1,25.0,F,0,Allison, Mrs. Hudson J C (Bessie Waldo Daniels)
1,48.0,M,1,Anderson, Mr. Harry
1,63.0,F,1,Andrews, Miss. Kornelia Theodosia
1,39.0,M,0,Andrews, Mr. Thomas Jr
1,53.0,F,1,Appleton, Mrs. Edward Dale (Charlotte Lamson)
1,71.0,M,0,Artagaveytia, Mr. Ramon
1,47.0,M,0,Astor, Col. John Jacob
1,18.0,F,1,Astor, Mrs. John Jacob (Madeleine Talmadge Force)
1,24.0,F,1,Aubart, Mme. Leontine Pauline
1,26.0,F,1,Barber, Miss. Ellen ""Nellie""
1,80.0,M,1,Barkworth, Mr. Algernon Henry Wilson
1,24.0,M,0,Baxter, Mr. Quigg Edmond
1,50.0,F,1,Baxter, Mrs. James (Helene DeLaudeniere Chaput)
1,32.0,F,1,Bazzani, Miss. Albina
1,36.0,M,0,Beattie, Mr. Thomson
1,37.0,M,1,Beckwith, Mr. Richard Leonard
1,47.0,F,1,Beckwith, Mrs. Richard Leonard (Sallie Monypeny)
1,26.0,M,1,Behr, Mr. Karl Howell
1,42.0,F,1,Bidois, Miss. Rosalie
1,29.0,F,1,Bird, Miss. Ellen

Passenger类定义



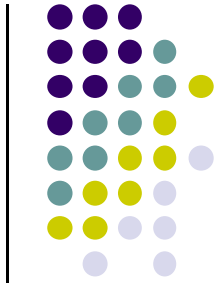
```
#Figure 24.18
class Passenger(object):
    features = ('C1', 'C2', 'C3', 'age', 'male gender')
    def __init__(self, pClass, age, gender, survived, name):
        self.name = name
        self.featureVec = [0, 0, 0, age, gender]
        self.featureVec[pClass - 1] = 1
        self.label = survived
        self.cabinClass = pClass
    def distance(self, other):
        return minkowskiDist(self.veatureVec, other.featureVec, 2)
    def getClass(self):
        return self.cabinClass
    def getAge(self):
        return self.featureVec[3]
    def getGender(self):
        return self.featureVec[4]
    def getName(self):
        return self.name
    def getFeatures(self):
        return self.featureVec[:]
    def getLabel(self):
        return self.label
```

读取数据、做预处理



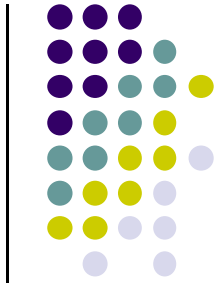
```
#Figure 24.19 (corrected)
def getTitanicData(fname):
    data = {}
    data['class'], data['survived'], data['age'] = [], [], []
    data['gender'], data['name'] = [], []
    f = open(fname)
    line = f.readline()
    while line != '':
        split = line.split(',')
        data['class'].append(int(split[0]))
        data['age'].append(float(split[1]))
        if split[2] == 'M':
            data['gender'].append(1)
        else:
            data['gender'].append(0)
        data['survived'].append(int(split[3])) #1 = survived
        data['name'].append(split[4:])
        line = f.readline()
    return data

def buildTitanicExamples(fileName):
    data = getTitanicData(fileName)
    examples = []
    for i in range(len(data['class'])):
        p = Passenger(data['class'][i], data['age'][i],
                      data['gender'][i], data['survived'][i],
                      data['name'][i])
        examples.append(p)
    return examples
```

测试生还模型



```
#Figure 24.20
def testModels(examples, numTrials, printStats, printWeights):
    stats, weights = [], [[], [], [], [], []]
    for i in range(numTrials):
        training, testSet = divide80_20(examples)
        xVals, yVals = [], []
        for e in training:
            xVals.append(e.getFeatures())
            yVals.append(e.getLabel())
        xVals = pylab.array(xVals)
        yVals = pylab.array(yVals)
        model = sklearn.linear_model.LogisticRegression().fit(xVals,
                                                               yVals)
        for i in range(len(Passenger.features)):
            weights[i].append(model.coef_[0][i])
        truePos, falsePos, trueNeg, falseNeg =\
                           applyModel(model, testSet, 1, 0.5)
        auroc = buildROC(model, testSet, 1, None, False)
        tmp = getStats(truePos, falsePos, trueNeg, falseNeg, False)
        stats.append(tmp + (auroc,))
    print('Averages for', numTrials, 'trials')
    if printWeights:
        for feature in range(len(weights)):
            featureMean = sum(weights[feature])/numTrials
            featureStd = stdDev(weights[feature])
            print(' Mean weight of', Passenger.features[feature],
              '=', str(round(featureMean, 3)) + ',',
              '95% confidence interval =', round(1.96*featureStd, 3))
    if printStats:
        summarizeStats(stats)
```

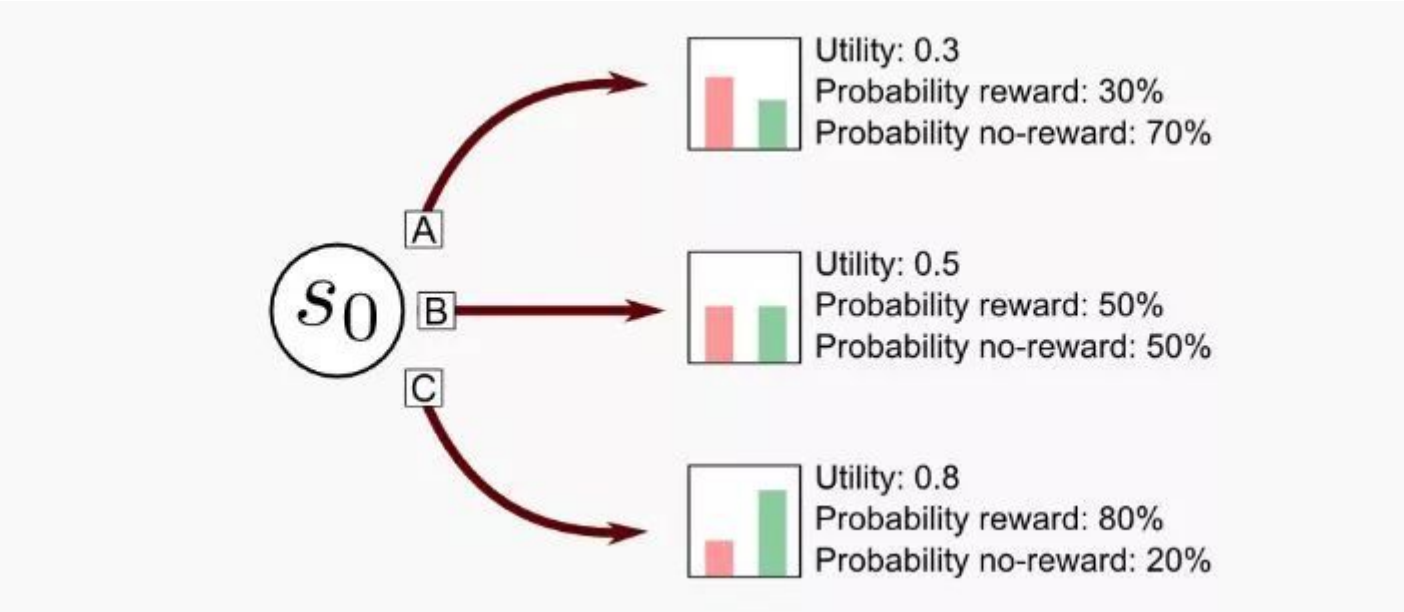
打印分类器的统计量



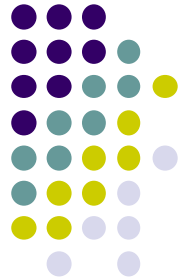
```
#Figure 24.21
def summarizeStats(stats):
    """assumes stats a list of 5 floats: accuracy, sensitivity,
       specificity, pos. pred. val, ROC"""
    def printStat(X, name):
        mean = round(sum(X)/len(X), 3)
        std = stdDev(X)
        print('Mean', name, '=', str(mean) + ',',
              '95% confidence interval =', round(1.96*std, 3))
    accs, sens, specs, ppvs, aurocs = [], [], [], [], []
    for stat in stats:
        accs.append(stat[0])
        sens.append(stat[1])
        specs.append(stat[2])
        ppvs.append(stat[3])
        aurocs.append(stat[4])
    printStat(accs, 'accuracy')
    printStat(sens, 'sensitivity')
    printStat(accs, 'specificity')
    printStat(sens, 'pos. pred. val.')
    printStat(aurocs, 'AUROC')

examples = buildTitanicExamples('TitanicPassengers.txt')
testModels(examples, 100, True, False)
```

多臂老虎机问题

多臂老虎机问题


s_0
A
B
C
Utility: 0.3
Probability reward: 30%
Probability no-reward: 70%
Utility: 0.5
Probability reward: 50%
Probability no-reward: 50%
Utility: 0.8
Probability reward: 80%
Probability no-reward: 20%

秘书问题



- 经典秘书问题的场景如下：雇主想要从n个候选人中雇佣一个秘书，这n个候选人有一个全局的序关系。他们以均匀随机的顺序到来，即n!种可能的到来顺序 出现的概率是相同的。
- 每面试一个候选人，雇主能看到此人在目前面试过的所有入中的排名情况，然后必须立即决定是否录取此人，并且做出决定后不可反悔。
- 雇主的目标是最大化录取到的最优候选人的概率。

秘书问题

- 一个策略：对前(r-l) 个人都拒绝，然后对剩下的(n-r+1) 个人进行面试，如果任何一个面试者比之前面试的人都优秀那么就录取这个人。

n	1	2	3	4	5	6	7	8	9
r	1	1	2	2	3	3	3	4	4
P	1.000	0.500	0.500	0.458	0.433	0.428	0.414	0.410	0.406