山东大学 计算机科学与技术 学院

机器学习（双语） 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201705130113 | 姓名：黄瑞哲 | | 班级：计科17.3 |
| 实验题目：朴素贝叶斯 | | | |
| 实验学时：2 | | 实验日期： 2019.10.30 | |
| 实验目的：   1. 了解朴素贝叶斯模型，包括假设、求解等。 2. 深化学习并进一步掌握的朴素贝叶斯的原理以及实现方式 3. 根据训练数据建立朴素贝叶斯模型，完成对入学建议的推荐。 | | | |
| 硬件环境：  Intel Core i5-8300H @ 2.3GHz | | | |
| 软件环境：  Windows10 Pro 1903  Python 3.7  Visual Studio Code 1.38. | | | |
| 实验步骤与内容：  朴素贝叶斯模型的核心是贝叶斯公式，因此建立模型前有属性独立性假设，即各个特征之间是独立同分布的。  根据条件概率贝叶斯公式可以得到  因此朴素贝叶斯就是要找到合适的y使得在给定的特征情况下，取当前标签的概率最大，即求得  从计算上来看，我们需要知道，其中我们无从得知，但是由于每一个式子中都带有，在我们最大化时不需要考虑它，因此我们只需要最大即可。这里的一个核心假设在于各个特征之间时相互独立的，因此我们可以使用链式法则  由此可以推导出对数-似然函数    可以使用拉格朗日乘子法求得最大值，因为我们有约束条件    可以解得      根据这个公式便可以建立朴素贝叶斯模型，但是考虑到训练数据中会有一些y没有出现，会导致分母为0，可以加上拉普拉斯平滑解决（认为缺省标签出现的概率相同）。 | | | |
| 结论分析与体会：  对于任务1：  我们根据训练数据计算得到每一个，在预测得时候枚举每一种可能的y并计算出其对应得概率，我们取概率最高得标签y作为该数据的标签。经过测试数据的测试，准确率达90.34%。    对于任务2：  我们进行100次取样，每次取样多选择100个数据，在取样的数据基础上训练朴素贝叶斯模型，并根据这个模型来测试其准确性。    由上图可以看出，随着样本不断增大，模型的准确性在不断提升，最终在90%左右震荡，与任务一的结果相吻合。由此可以看出，训练数据规模越大，模型的准确性越高，因此为了提高贝叶斯分类器的准确性，我们提供的训练数据越多越好。但是较多的训练数据也会使得训练分类器的时间变长。 | | | |

附录：程序源代码

# exp4\_1.py

import numpy as np

def load\_data(file):

feature = []

label = []

with open(file) as f:

for each\_line in f.readlines():

data = list(map(int, each\_line.strip().split()))

feature.append(data[0:-1])

label.append([data[-1]])

return np.mat(feature), np.mat(label)

def fit(feature, label):

n, m = np.shape(feature)

max\_x = np.max(feature)

max\_y = np.max(label)

p = [[[0 for i in range(max\_y + 1)] for i in range(max\_x + 1)] for i in range(m)]

y = [0 for i in range(max\_y + 1)]

for j in range(n):

y[label[j, 0]] += 1

for i in range(m):

p[i][feature[j, i]][label[j, 0]] += 1

for i in range(m):

for j in range(max\_x + 1):

for k in range(max\_y + 1):

p[i][j][k] /= y[k]

for i in range(max\_y + 1):

y[i] /= n

return p, y

def classify(feature, prob, y):

ans = 0

max\_p = 0

m = np.shape(feature)[1]

for i in range(len(y)):

p = y[i]

for j in range(m):

p \*= prob[j][feature[0, j]][i]

if p > max\_p:

max\_p = p

ans = i

return ans

if \_\_name\_\_ == "\_\_main\_\_":

feature, label = load\_data("exp4/data/training\_data.txt")

feature\_test, label\_test = load\_data("exp4/data/test\_data.txt")

p, y = fit(feature, label)

correct = 0

for (f\_test, l\_test) in zip(feature\_test, label\_test):

if classify(f\_test, p, y) == l\_test:

correct += 1

print("accuracy=%.2f%%" % (correct / len(feature\_test) \* 100))

# exp4\_2.py

import numpy as np

import matplotlib.pyplot as plt

laplace = [3, 5, 4, 4, 3, 2, 3, 3]

def load\_data(file):

feature = []

label = []

with open(file) as f:

for each\_line in f.readlines():

data = list(map(int, each\_line.strip().split()))

feature.append(data[0:-1])

label.append([data[-1]])

return np.mat(feature), np.mat(label)

def fit(feature, label):

n, m = np.shape(feature)

max\_x = np.max(feature)

max\_y = np.max(label)

p = [[[0 for i in range(max\_y + 1)] for i in range(max\_x + 1)] for i in range(m)]

y = [0 for i in range(max\_y + 1)]

for j in range(n):

y[label[j, 0]] += 1

for i in range(m):

p[i][feature[j, i]][label[j, 0]] += 1

for i in range(m):

for j in range(max\_x + 1):

for k in range(max\_y + 1):

p[i][j][k] /= (y[k] + laplace[i])

for i in range(max\_y + 1):

y[i] /= n

return p, y

def classify(feature, prob, y):

label = 0

max\_p = 0

m = np.shape(feature)[1]

for i in range(5):

p = y[i]

for j in range(m):

p \*= prob[j][feature[0, j]][i]

if p > max\_p:

max\_p = p

label = i

return label

def random\_test(feature, label, num):

import random

vis = dict()

n = np.shape(feature)[0]

feature\_select = []

label\_select = []

for \_ in range(num):

pos = random.randint(0, n - 1)

while pos in vis:

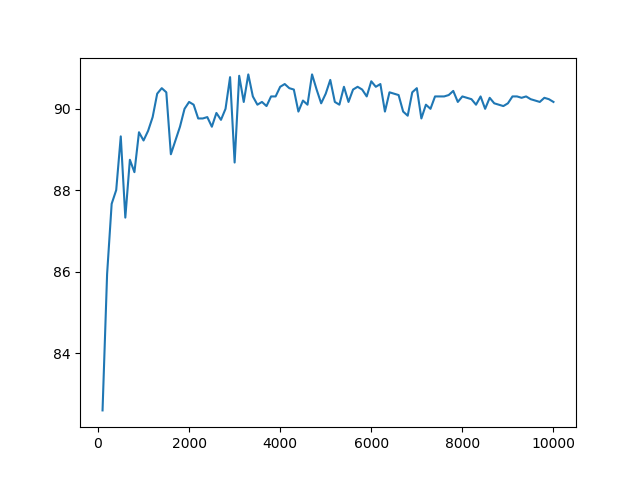
pos = random.randint(0, n - 1)

vis[pos] = 1

feature\_select.append(feature[pos].tolist()[0])

label\_select.append(label[pos].tolist()[0])

return np.mat(feature\_select), np.mat(label\_select)



if \_\_name\_\_ == "\_\_main\_\_":

feature, label = load\_data("exp4/data/training\_data.txt")

feature\_test, label\_test = load\_data("exp4/data/test\_data.txt")

tot = [ i \* 100 for i in range(1, 101)]

rate = []

for cnt in tot:

f, l = random\_test(feature, label, cnt)

p, y = fit(f, l)

correct = 0

for (f\_test, l\_test) in zip(feature\_test, label\_test):

if classify(f\_test, p, y) == l\_test:

correct += 1

rate.append(correct / len(feature\_test) \* 100)

plt.plot(tot, rate)

plt.show()