

**软件学院学生实验报告**

**实验课程名称：** 智能算法设计 **教师： 肖如良**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验项目名称** | **背包问题的分布估计算法** | | | **实验成绩** |  |
| **学生姓名** | **高强** | **学 号** | **123032014010** | **年级专业班级** | **软三** |
| **小组成员** |  | | | **实验日期** | **2016.11.16** |

1. **实验目的**

背包问题的分布估计算法：给定n种物品和一个背包物品i的质量是ci，其价值为 pi，背包的质量容量为C，如何选择物品，使得装入背包中物品的总价值最大。用分布估计算法来实现最优化问题。

1. **实验软硬件环境**

Windows,PyCharm,python

1. **实验内容及要求**

选择一种熟悉的语言实现关于背包问题的分布估计算法，要求给定质量容量的背包选择一组合适的商品序列放入背包内是的装入背包中的物品的总价值最大。实验过程要记录其最终结果，并将结果可视化。

1. **实验记录**

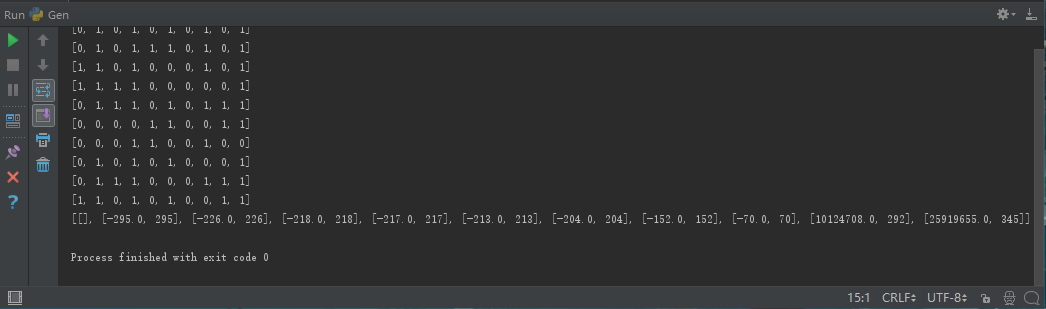
**1.分布估计算法（EDA）代码实现**

# estimation of distribution to solve package question  
**import** numpy **as** np  
**import** math  
**from** numpy **import** random  
  
price = np.array([55, 10, 47, 5, 4, 50, 8, 61, 85, 87])  
weight = np.array([95, 4, 60, 32, 23, 72, 80, 62, 65, 46])  
# 初始化种群数量，可以改进用input()  
popSize = 200  
# 初始化物品数量  
size = 10  
# 背包的容量，可以动态指定  
Count = 269  
# 记录代数的全局变量，初始化为0代  
generation = 0  
# 打开两个文件用于存储在进化过程中每一代的最大值和平均值  
fileMax = open("C:\\Users\gq\Desktop\\max.txt", "a")  
fileAvg = open("C:\\Users\gq\Desktop\\average.txt", "a")  
# 初始化价格数组  
**def getPrive**(path):  
 price1 = np.genfromtxt(path, dtype=float, delimiter=",")  
 **return** price1  
  
**def getWeight**(path):  
 weight1 = np.genfromtxt(path, dtype=float, delimiter=",")  
 **return** weight1  
  
# 求适应度函数  
**def fitness**(goods):  
 li = list()  
 w = 0.0  
 p = 0.0  
 M = 5000 # 参数M  
  
 **for** i **in** range(len(goods)):  
 **if** goods[i] == 1:  
 li.append(i)  
 **for** j **in** range(len(li)):  
 index = li[j]  
 p += float(price[index])  
 w += float(weight[index])  
 cw = Count - w  
 **if** cw > 0:  
 cw = 0  
 **else**:  
 cw = M \* math.pow(cw, 2)  
 fit1 = p \* -1 + cw  
 **return** fit1  
  
# 初始化种群函数  
**def createPop**(popsize,size):  
 pro = np.zeros(size)  
 goodlist = np.zeros((popsize, size))  
  
 # 初始化概率均为0.5  
 pro.fill(0.5)  
 # 初始化种群的商品列表  
 **for** j **in** range(popsize):  
 **for** k **in** range(size):  
 **if** np.random.rand() > 0.5:  
 goodlist[j][k] = 1  
  
 **return** pro, goodlist  
  
# 调用 createPop()函数，初始化种群  
pro, goodlist = createPop(popSize, size)  
  
**def changePop**(): #种群进化  
 n = int(popSize \* 0.4)  
 select = np.zeros((n, size))  
 order = np.zeros(popSize, dtype=[('x', int), ('y', float)])  
 index = 0  
 **global** pro  
  
 # 首先计算每个个体的适应度，然后按照适应度大小对个体进行排名  
 **for** l **in** range(popSize):  
 order[l][0] = l  
 order[l][1] = fitness(goodlist[l])  
 order.sort(order='y')  
  
 **for** i **in** order.getfield(np.int):  
 select[index] = goodlist[i]  
 index += 1  
 **if** index >= n:  
 **break** # 根据 评估每个变量的概率，重新生成变量的概率值  
 pro = select.sum(axis=0)/n  
  
# 进化生成下一代种群函数  
**def evolution**():  
 goods = np.zeros((size))  
 # 根据新的概率，生成下一代种群  
 **for** j **in** range(popSize):  
 **for** i **in** range(size):  
 **if** random.rand() < pro[i]:  
 goods[i] = 1  
 **else**:  
 goods[i] = 0  
 goodlist[j] = goods  
 money = price.dot(goodlist.T)  
 max = money.max()  
 average = money.sum()/popSize  
  
 **global** generation  
 generation += 1  
  
 fileMax.write(str(generation)+',' + str(max))  
 fileAvg.write(str(generation)+',' + str(average))  
 fileMax.write("\n")  
 fileAvg.write("\n")  
  
**def eda**(genes):  
 **for** g **in** range(genes):  
 changePop()  
 evolution()  
 # 将最后一代种群中个体的商品列表写到文件中  
 goodlist[popSize - 1].tofile("C:\\Users\gq\Desktop\goods.txt", sep=",")  
  
**if** \_\_name\_\_ == '\_\_main\_\_':  
 eda(20)  
 fileMax.close()  
 fileAvg.close()

1. **实验结果可视化部分代码**

**import** matplotlib.pyplot **as** plt  
  
**def plotMax**():  
 file = open('C:\\Users\gq\Desktop\\max.txt', 'r')  
 lineslist = file.readlines()  
 lineslist = [line.strip().split(",") **for** line **in** lineslist]  
 file.close()  
 x = [x[0] **for** x **in** lineslist]  
 print(x)  
 y = [x[1] **for** x **in** lineslist]  
 print(y)  
 plt.plot(x, y, 'b\*')  
 plt.plot(x, y, 'r')  
 plt.legend()  
  
**def plotMean**():  
 file = open('C:\\Users\gq\Desktop\\average.txt', 'r')  
 linesList = file.readlines()  
 linesList = [line.strip().split(",") **for** line **in** linesList]  
 file.close()  
 x = [x[0] **for** x **in** linesList]  
 print(x)  
 y = [x[1] **for** x **in** linesList]  
 print(y)  
 plt.plot(x, y, 'b\*')  
 plt.plot(x, y, 'r')  
 plt.xlabel("generation --x")  
 plt.ylabel("average and max of value when N=1000")  
 plt.ylim(0, 400)  
 plt.title('EDA algorithm to solve package question')  
 plt.legend()  
 plotMax()  
 plt.show()  
**def plotGen**():  
 file = open('C:\\Users\gq\Desktop\\gen.txt', 'r')  
 line = file.readline()  
 y = line.strip().split(",")  
 ylist = []  
 #for j in range(len(y) - 1):  
 # ylist.append(int(y[j]))  
 file.close()  
 ylist.sort(reverse=**True**)  
 num = len(ylist)  
 xlist = [x **for** x **in** range(num)]  
 plt.plot(xlist, ylist, 'b\*')  
 plt.plot(xlist, ylist, 'r')  
 plt.xlabel("generation --x")  
 plt.ylabel("average and max of value when N=200")  
 plt.ylim(0, 400)  
 plt.title('gen algorithm to solve package question')  
 plt.legend()  
 plotMax()  
 plt.show()  
  
**if** \_\_name\_\_ == '\_\_main\_\_':  
 #plotMean()  
 plotGen()

1. **实验结果截图**

采用遗传算法的结果 ****

遗传算法结果

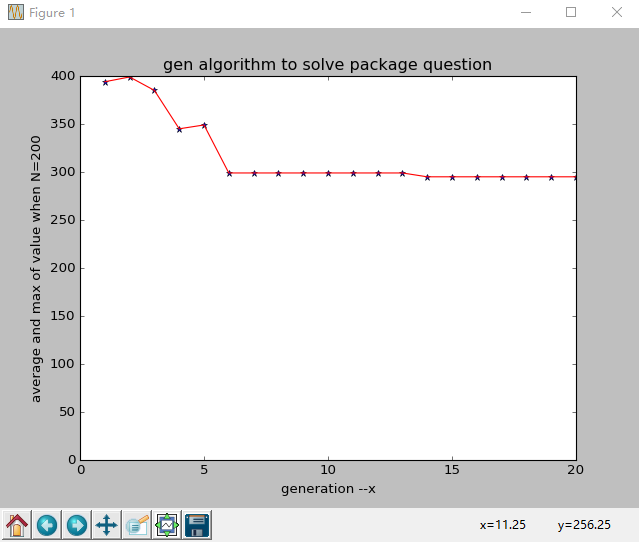


图2

N为100时

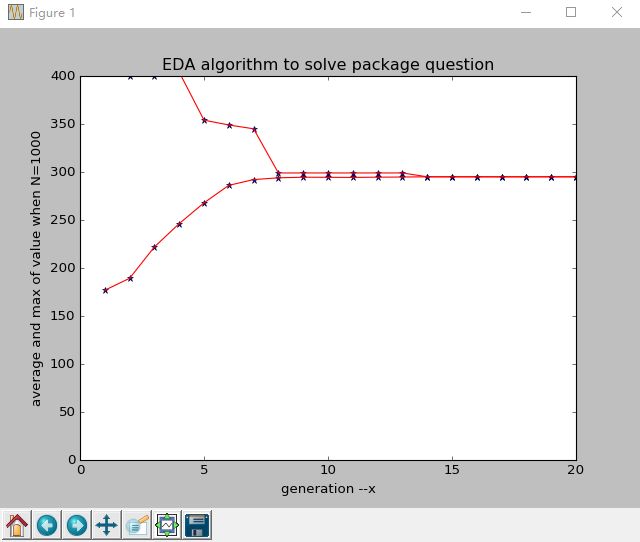
****

图3

当种群较小的时候

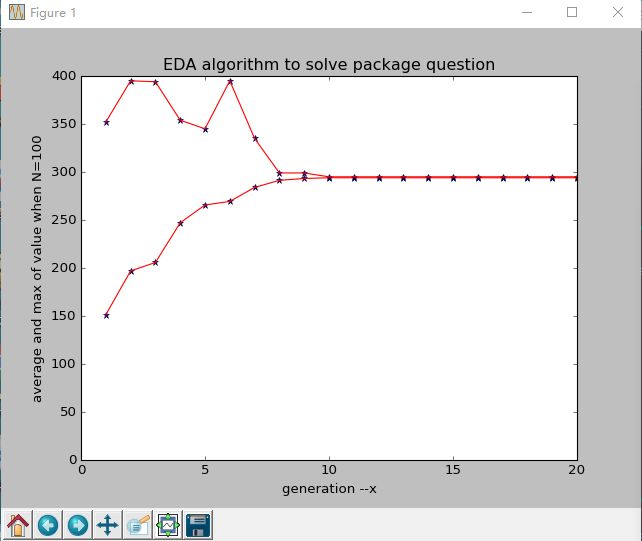
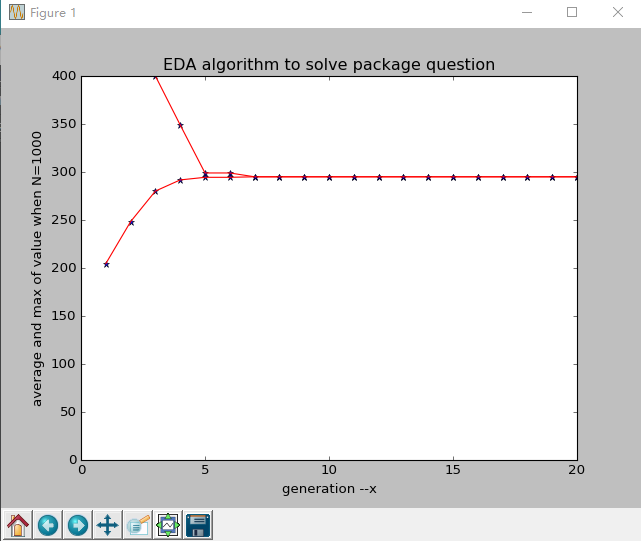


图4

种群数N为1000时,ratio为0.3时

 图5

将结果记录到相应的文件中

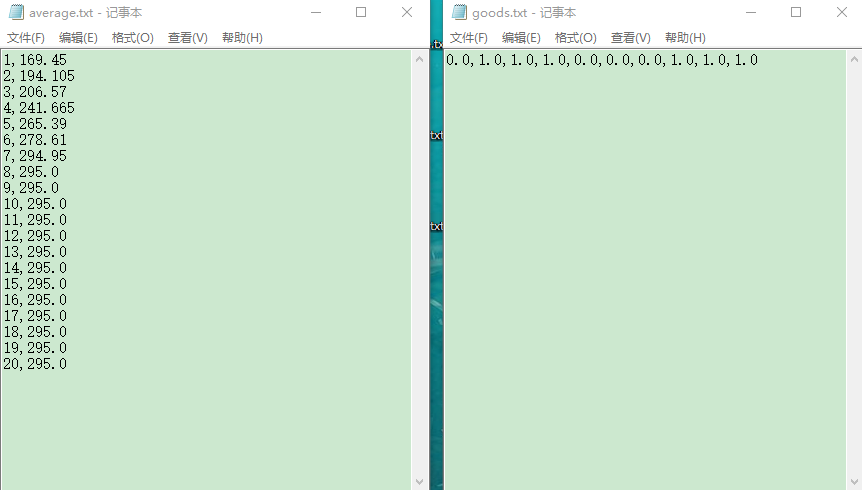


图4

1. **实验小结（心得体会、遇到的问题及解决方式、未解决/需进一步研讨的问题或建议新实验方法等）**

1.当采用遗传算法的时候，其适应度函数与分布估计算法有所不同需要做相应的更改。因为遗传算法采用轮盘赌算法来选择进化，其适应度函数不能为负值否则导致算法不能收敛。

2.