****

**2021—2022学年第2学期**

**《机器学习》课程报告**

题目：基于SVM的银杏树识别判断

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **学号** | **专业班级** | **姓名** | **分工占比**  **(共100%)** | **得分** |
| 2007040118 | 智科2001 | 孟祥奇 | 30% |  |
| 2001020111 | 智科2001 | 孔凡华 | 30% |  |
| 2007040122 | 智科2001 | 王卫超 | 20% |  |
| 2007040117 | 智科2001 | 冒翔 | 20% |  |

2022年 6 月 1 日

# 说明

1. 《机器学习》课程，旨在对学生进行大数据处理能力的综合训练，以提高学生分析问题、建立模型、解决问题能力。所有参加本课程的学生都必须参与算法设计实现、课程报告撰写工作。
2. 《机器学习》结束后，需要提交**源数据、程序源代码、报告电子版、运行视频，压缩后以“学号+班级+姓名”进行命名，材料经班长汇总后交给任课老师**。逾期未提交相关资料者不得参加成绩评定。
3. 指导教师需对学生实践报告填写评语并在相应位置填写成绩并签字。
4. 课程报告要求严格按照本报告模板撰写，条理清晰、内容详尽、论述准确、书写认真。成绩由报告格式设计、内容设计、算法设计、结果描述等几部分组成。
5. 报告格式要求如下：表格、图像进行编号，正文要求宋体、小四，行前缩进2字符，间距1.5倍行距。其他未尽事宜请参考本科毕业设计报告撰写格式说明，纸质报告要求双面打印。
6. 报告可根据内容适当增加页面，但不宜长篇大论，所写内容应紧扣主题。

|  |
| --- |
| **基于SVM的银杏树识别判断** |
| 一、研究意义 |
| 山东省郯城县有“天下银杏第一县”之称。郯城县以“银杏之乡”闻名于世，是中国著名的银杏集中产区，也是中国栽培银杏最早的地区之一。自20世纪80年代后期，郯城县就立足银杏资源优势，确立了大力发展银杏产业的思路。截止2005年，郯城县先后举办了两届“中国郯城国际银杏节”，一届“银杏资源利用招商洽谈会”，承办了“全国第三届银杏学术研讨会”。郯城县“咬住银杏不放松”，一年一个大台阶，推动银杏产业持续快速发展，使之成为全县的一大支柱产业。1999年，郯城被国家命名为“中国银杏之乡”。所以说银杏树可以说是山东省的一个很有代表性的事物，我们选择这个主题作为我们小组的第二次大作业的主题。 |
| 二、数据描述 |
| 1.数据来源： 1).百度图片（山东省银杏树）（<https://image.baidu.com>）  2).长景园林-山东临沂郯城银杏  ( <https://www.cmeii.com/xinwenzhongxin/1168.html>)  3).中国山东网（<http://www.sdchina.com/>） 2.数据内容： 选取银杏树和用来与银杏树进行区别的其他树种的图片共400余张。  其中银杏树260张，其他树种169张。其中一部分作为训练集，一部分作为测试集 3.数据预处理： 1).标签  制作数据集图片对应的标签，正标签为1，负标签为-1  2).灰度图处理  3).二值化处理  4).图片大小resize |
| 三、模型描述 |
| for data in file do  # 添加label标签，银杏树设为1， 其他树设为0  if data == 'ginkgo' then  label = 1  else then  label = 0  # 将图像数据转为灰度图  gray\_data = cv2.cvtColor(data, cv2.COLOR\_BGR2GRAY)  #二值化处理  ret,thresh1=cv2.threshold(gray\_data,127,255, cv2.THRESH\_BINARY)  # 调整大小，拉伸为一维  gray\_data = cv2.resize(gray\_data, (320, 320))  gray\_data = gray\_data.reshape(-1)  then 加入列表，转换为数组  end |
| 算法实现 |
| 本次主要使用的SMO算法（注：以下代码只是基于理论算法，并不是实际求解过程，因此省去了一些输出） 1.参数的初始化 def \_\_init\_\_(self, dataMatIn, classLabels, C, toler, kTup):  self.X = dataMatIn #数据矩阵  self.labelMat = classLabels #数据标签  self.C = C #松弛变量  self.tol = toler #容错率  self.m = np.shape(dataMatIn)[0] #数据矩阵行数  self.alphas = np.mat(np.zeros((self.m,1))) #根据矩阵行数初始化alpha参数为0  self.b = 0 #初始化b参数为0  self.eCache = np.mat(np.zeros((self.m,2))) #根据矩阵化核行数初始化虎误差缓存，第一列为是否有效的标志位，第二列为实际的误差E的值。  self.K = np.mat(np.zeros((self.m,self.m))) #初始K  for i in range(self.m): #计算所有数据的核K  self.K[:,i] = kernelTrans(self.X, self.X[i,:], kTup) 2.计算高斯核K def kernelTrans(X, A, kTup):  m,n = np.shape(X)  K = np.mat(np.zeros((m,1)))  if kTup[0] == 'lin': K = X \* A.T #线性核函数,只进行内积。  elif kTup[0] == 'rbf': #高斯核函数,根据高斯核函数公式进行计算  for j in range(m):  deltaRow = X[j,:] - A  K[j] = deltaRow\*deltaRow.T  K = np.exp(K/(-1\*kTup[1]\*\*2)) #计算高斯核K  else:  raise NameError('核函数无法识别')  return K 3.计算误差  1. def calcEk(oS, k): 2. fXk=float(np.multiply(oS.alphas,oS.labelMat).T\*oS.K[:,k] + oS.b) 3. Ek=fXk - float(oS.labelMat[k]) 4. return Ek  4．SMO算法 def innerL(i, oS):    Ei = calcEk(oS, i) #计算误差Ei  #优化alpha,设定一定的容错率。  if ((oS.labelMat[i] \* Ei < -oS.tol) and (oS.alphas[i] < oS.C)) or ((oS.labelMat[i] \* Ei > oS.tol) and (oS.alphas[i] > 0)):  #使用内循环启发方式2选择alpha\_j,并计算Ej  j,Ej = selectJ(i, oS, Ei)  #保存更新前的aplpha值  alphaIold = oS.alphas[i].copy()  alphaJold = oS.alphas[j].copy()  #计算上下界L和H  if (oS.labelMat[i] != oS.labelMat[j]):  L = max(0, oS.alphas[j] - oS.alphas[i])  H = min(oS.C, oS.C + oS.alphas[j] - oS.alphas[i])  else:  L = max(0, oS.alphas[j] + oS.alphas[i] - oS.C)  H = min(oS.C, oS.alphas[j] + oS.alphas[i])  if L == H:  print("L==H")  return 0  eta = 2.0 \* oS.K[i,j] - oS.K[i,i] - oS.K[j,j] #计算eta  if eta >= 0:  print("eta>=0")  return 0  oS.alphas[j] -= oS.labelMat[j] \* (Ei - Ej)/eta #更新alpha\_j  oS.alphas[j] = clipAlpha(oS.alphas[j],H,L) #修剪alpha\_j  updateEk(oS, j) #更新Ej至误差缓存  if (abs(oS.alphas[j] - alphaJold) < 0.00001):  return 0  oS.alphas[i] += oS.labelMat[j]\*oS.labelMat[i]\*(alphaJold - oS.alphas[j]) #更新alpha\_i  updateEk(oS, i) #更新Ei至误差缓存  b1=oS.b-Ei-oS.labelMat[i]\*(oS.alphas[i]-alphaIold)\*oS.K[i,i] - oS.labelMat[j]\*(oS.alphas[j]-alphaJold)\*oS.K[i,j]  b2=oS.b-Ej-oS.labelMat[i]\*(oS.alphas[i]-alphaIold)\*oS.K[i,j]-oS.labelMat[j]\*(oS.alphas[j]-alphaJold)\*oS.K[j,j] #更新b\_1和b\_2    if (0 < oS.alphas[i]) and (oS.C > oS.alphas[i]):  oS.b = b1  elif (0 < oS.alphas[j]) and (oS.C > oS.alphas[j]):  oS.b = b2  else:  oS.b = (b1 + b2)/2.0  return 1  else:  return 0 5.最终实现 def SMO(dataMatIn, classLabels, C, toler, maxIter, kTup = ('rbf',0)):  oS=optStruct(np.mat(dataMatIn), np.mat(classLabels).transpose(), C, toler, kTup) #初始化数据结构  iter = 0 #初始化当前迭代次数  entireSet = True; alphaPairsChanged = 0  #遍历整个数据集都alpha也没有更新或者超过最大迭代次数,则退出循环  while(iter<maxIter) and ((alphaPairsChanged > 0) or (entireSet)):  alphaPairsChanged = 0  if entireSet: #遍历整个数据集  for i in range(oS.m):  alphaPairsChanged += innerL(i,oS)  iter += 1  else:  nonBoundIs=np.nonzero((oS.alphas.A>0)\*(oS.alphas.A<C))[0]  #遍历不在边界0和C的alpha  for i in nonBoundIs:  alphaPairsChanged += innerL(i,oS)  iter += 1  if entireSet: #遍历一次后改为非边界遍历  entireSet = False  elif (alphaPairsChanged == 0): #如果alpha没有更新,计算全样本遍历  entireSet = True  return oS.b,oS.alphas |
| 五、运行结果及意义说明 |
| 在测试集中数据的识别准确率为87.5%，可以让人接受，因为有一些树种与银杏树很相似，再加上有时拍摄角度的问题，即使是人眼辨别也时常会出错。 |
| 六、数学推导 |
|  |