山东大学 软件 学院

机器学习 课程实验报告

|  |  |  |
| --- | --- | --- |
| 学号：201800301153 | 姓名： 傅显坤 | 班级： 1班 |
| 实验日期： 2020.12.21 | | |
| 实验题目：**集成学习**  实验给出的数据:  CIFAR-10数据集，数据集中包含 50000 张训练样本，10000 张测试样本，可将训练样本划分为49000 张样本的训练集和1000 张样本的验证集，测试集可只取1000 张测试样本。其中每个样本都是 32×32 像素 的彩色照片，每个像素点包括RGB三个数值，数值范围0 ~ 255，所有照片分属10个不同的类别。  实验的问题  ﻿用集成方法对数据集进行分类  ﻿(1) 利用若干算法，针对同一样本数据训练模型，使用投票机制，少数服从多数，用多数算法给出的结果当作最终的决策依据，对CIFAR-10数据集进行分类，给出在测试集上的精确度；  (2) 所选算法包括：  SVM（核函数为多项式核函数）  KNN（k=7）；  神经网络 | | |
| 软件环境：  MacOS Catalina  Python3.0  IDE：PyCharm | | |
| 实验步骤与内容：  1.数据导入  首先加载训练集的文件——cifar-10-batches-py，将每个图片数据格式化为3072\*1的numpy数组，标签数据格式化为10\*1的numpy数组。  训练集图片是50000\*3072的numpy数组，标签是10000\*10的numpy数组（那个值为1表示它属于哪一类）。   | **名称** | **作用** | | --- | --- | | b’data’ | 是一个10000x3072的array，每一行的元素组成了一个32x32的3通道图片，共10000张 | | b’labels’ | 一个长度为10000的list，对应包含data中每一张图片的label |   2.模型分析  (1)KNN算法设计：  在实验三中使用过knn算法的思想，根据实验要求，固定K值为7，并且体积和N每个训练点都是相同的，所以只需要通过判断这七个距离测试点最近的样本中，所占比例最多的类别是什么，然后将该测试点设为这个类别即可。  将每个测试点的真实类别与算法得到的类别进行比较，最后得出在测试集上的准确率。  一开始选用训练集大小为10000，测试集为1000.算法的准确率为0.295.    然后选用所有训练集50000，测试集为10000。计算量急剧增加，算法运行了好久，得到的准确率为0.3508.    虽然效果不是很好但是已经超过10%，达到了使用的条件。而且集中学习的性能对于弱分离器的效果更加明显。  (2)BP神经网络设计：  实验四刚做过神经网络的实验，只需要在实验四的基础上稍加更改就可以了，NeuralNetwork基本结构不变，只是在test时返回模型对测试点做出的判断类别。当学习率为0.01，结构是3072\*50\*10，循环5次，训练集为15000，测试集为1000.在测试集上得到的准确率为0.361。    (3)SVM算法设计：  创建一个svm分类器，其中C是为了缓解出现噪声而带来的模型过拟合问题，C允许支持向量机在一些样本上出错。引入C后优化目标可以写成式一，其中是一个0、1损失函数，如公式二。显然当C为无穷大时，公式一迫使每个样本均分对，C取有限的值时，公式一允许一些样本不满足约束。  公式一：  公式二  其中kernel和degree代表的是核函数的类型和使用多项式核函数时的度。  由于图片的分类是线性不可分的于是必须采用核函数进行映射。于是要求解的分离界面就是公式三，对偶问题中求解的最大a可以写成公式四的形式，根据公式五，可以将公式四换做公式六求解，这时就无需求映射空间的具体值，只需将原来空间的值带入核函数就可求得映射空间的值。  公式三：  公式四：  公式五：  公式六：  学习过程，只需要将训练集数据和标签放到模型中构建svm分离器即可。具体原理是求最大的是一个二次规划问题。可以使用SMO算法进行求解。  预测过程，只需将测试点放到模型中，得到的标签就是模型的预测值，与其真实的标签对比，获得模型的准确率。  测试  选用训练集为50000，测试集为10000时。算法的准确率为0.4868.    3.集成学习过程  我选择采用Bagging方法进行并行式集成学习，需要使用自助采样法（bootstrap sampling）有放回随机抽取样本，训练不同的模型。使用numpy.random.choice(ls, size=10000, replace=True)方法实现有放回的随机抽样。创建一个Data\_my类用于将特征矩阵和标签一同抽取，方便训练，模型。之后对每个个体学习器对测试个体的标签进行投票，投票多的就是最终模型给出的分类结果。   1. 三类模型的集成：（KNN，BP神经网络（结构均为3072\*50\*10）和SVM）   1.一个KNN。一个BP，一个SVM  KNN的k是7，BP学习率是0.05，循环5次，svmC=3.0 多项式核函数的次数是3.  对三类模型随机抽取10000个样本进行训练，之后对1000个测试样本进行投票，如果类别的票数相同采用svm的分类方式。  可以看出准确率比svm略有下降但是明显高于神经网络和KNN算法    2.两个KNN，两个神经网络和两个svm  KNN的参数K=7，5。BP的学习率是0.05，0.5循环5次和10次。Svm C=3.0，2.0核函数的次数是3和2.  可以看出准确率比之前要好了一些    3.三个KNN，三个神经网络，三个svm。  KNN的参数K=7，5，3。BP的学习率是0.05，0.5，0.25循环5次，10次和15次。Svm C=3.0，2.0，1.0核函数的次数是3，2和4.  准确率提高了0.03。    4. 四个KNN，四个神经网络，四个svm。  KNN的参数K=7，5，3，1。BP的学习率是0.05，0.5，0.25，0.15循环5次，10次，15次和20次。Svm C=3.0，2.0，1.0，1.0核函数的次数是3，2，4和5.准确率达到了0.431    5.两个KNN，三个svm，四个神经网络。让训练集是30000，测试集是3000.  KNN的参数K=7，5。BP的学习率是0.05，0.5，0.25，0.35循环5次，10次，15次和20次。Svm C=3.0，2.0，1.0，核函数的次数是3，2，4.准确率达到了0.464，提高了0.03。     1. 使用一类模型的学习器 2. 使用10个KNN模型   训练集为10000，测试集为1000.参数K值分别为1，3，5，7，9，1，3，5，7，9.准确率为0.33.比单个模型提高了0.04.     1. 使用10个BP神经网络模型   对照：一个BP神经网络从训练集中随机抽样15000个进行训练在测试集1000个上的准确率为：    训练集为15000，测试集为1000.学习率分别为0.01，0.05，0.08，0.1，0.15，0.18，0.2，0.35，0.45，0.55.循环次数前五个为15次，后五个为20次。准确率为0.376，准确率提高了0.08。     1. 使用10个SVM   对照：一个svm从训练集中随机抽样10000个进行训练在测试集1000个上的准确率为:    训练集为10000，测试集为1000.。准确率为0.493，准确率提高了将近0.08。     1. 两类模型的集成： 2. svm和BP，使用3个svm和3个BP进行集成学习。   准确率确实上升了不少。       1. svm和KNN，使用3个svm和3个knn进行集成学习   可以看到相对于svm，准确率提高较小，但是相对于KNN，算法准确率将近提高了0.2。     1. BP神经网络和KNN，使用三个BP神经网络与KNN集成学习。   相对于KNN的准确率提升了0.1；而BP神经网络也提升了0.06. | | |
| 实验结果：  实验感悟：  通过svm的实验了解了svm的运行原理，通过核函数的映射使得svm能够处理线性不可分问题，并且有着良好的性能。理解了什么是经验风险和什么是学习机结构的风险。理解了svm的运行原理。  通过集成学习中的Bagging方式学习了如何将模型集成到一起从而提升整体的准确率。当采集样本时，采用bootstrap sampling方法进行又放回的抽样，训练不同的模型。体会了同质集成学习和异质集成学习，发现相对于同质集成学习，异质集成学习能更好的提高模型的准确率。  重要代码：   1. KNN的分类：   **def** get\_category(train\_data, train\_data\_labels, x, k):  *""""*  *train\_data训练的数据*  *train\_data\_labels训练数据的类别*  *x输入的数据*  *k近邻的个数*  *"""*  categorys = []  a = []  **for** i **in** range(len(train\_data)):  categorys.append(Category(np.linalg.norm(x - train\_data[i]),  train\_data\_labels[i])) *# 求距离和标签*  categorys.sort(key=**lambda** x: x.distance)  **for** i **in** range(k):  a.append(categorys[i].lable)  maxlabel = max(a, key=a.count)  *# print(maxlabel)*  **return** maxlabel   1. BP神经网络的分类   **def** get\_bp(test\_data, test\_label, learning\_rate, cycles):  *# 定义训练集*  images\_data, labels = load('train')   *# 对训练集进行又放回的随机抽样*  ls = []  **for** i **in** range(len(images\_data)):  ls.append(Data\_my(images\_data[i], labels[i]))  train\_data\_my = numpy.random.choice(ls, size=10000, replace=True)   value = []  ls = [3072, 50, 10]  *# 神经网络的层数是从隐藏层开始计数的，输入层不计入总层数。*  n = NeuralNetwork(2, ls, learning\_rate)  **for** i **in** range(cycles):  **for** j **in** range(len(train\_data\_my)):  n.update(train\_data\_my[j].vertex, train\_data\_my[j].label) *# 更新模型*  **for** i **in** range(len(test\_data)):  value.append(n.test(test\_data[i]))   success = 0  **for** i **in** range(len(test\_data)):  **if** value[i] == test\_label[i]:  success += 1  print("bp神经网络的算法的准确率为：" + str(success / len(test\_data)))   **return** value   1. svm的分类：   **def** get\_svm(test\_data, test\_label, C, degree):  ls = []  *# 定义训练集——knn无需分为分为训练和验证*  images\_data, labels = load\_old('train')  *# 对训练集进行又放回的随机抽样*  lss = []  **for** i **in** range(len(images\_data)):  lss.append(Data\_my(images\_data[i], labels[i]))  train\_data\_my = np.random.choice(lss, size=10000, replace=True)  train\_data = []  train\_labels = []  **for** i **in** range(len(train\_data\_my)):  train\_data.append(train\_data\_my[i].vertex)  train\_labels.append(train\_data\_my[i].label)  train\_data = np.array(train\_data)  train\_labels = np.array(train\_labels)  svm\_my = Svm(C=C, kernel='poly', degree=degree) *# kernel='poly',*  svm\_my.update(train\_data, train\_labels)  ls = svm\_my.get\_test\_labels(test\_data, test\_label)   success = 0  **for** i **in** range(len(test\_data)):  **if** ls[i] == test\_label[i]:  success += 1  print("svm算法的准确率为：" + str(success / len(test\_data)))   **return** ls   1. Bagging投票做出判断   **def** bagging():  test\_data, test\_labels = get\_testData()  success = 0   *# # 1.神经网络模型-对测试集类型的判断*  values1 = get\_bp(test\_data, test\_labels,0.05, 25)  values1\_2 = get\_bp(test\_data, test\_labels,0.01, 25)  values1\_3 = get\_bp(test\_data, test\_labels,0.03, 25)   *# # 2.svm模型-对测试集类型的判断*  values2 = get\_svm(test\_data, test\_labels, 3.0, 3)  values2\_2 = get\_svm(test\_data, test\_labels, 2.0, 2)  values2\_3 = get\_svm(test\_data, test\_labels, 1.0, 4)    *# 3.投票做出做出决策*  *# values3 = get\_knn(test\_data, test\_labels, 7)*  *# values3\_2 = get\_knn(test\_data, test\_labels, 5)*  *# values3\_3 = get\_knn(test\_data, test\_labels, 3)*    **for** i **in** range(len(test\_data)):  value = [values1[i], values1\_2[i], values1\_3[i], values2[i], values2\_2[i], values2\_3[i]]  maxlabel = max(value, key=value.count)  **if** maxlabel == test\_labels[i]:  success += 1   print("集成学习算法的准确率为" + str(success / len(test\_data))) | | |