机器学习 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号： | 姓名： | | 班级： |
| 实验题目：Lab6 集成学习 | | | |
| 实验学时：4 | | 实验日期： | |
| 实验目的：  使用mnist数据集， 结合多种学习方式进行集成学习 | | | |
| 硬件环境：  Intel Core（TM） i5-4210U  RAM 8G  64位操作系统，基于x64处理器 | | | |
| 软件环境：  Win 10 64bit  Libsvm-3.22  MATLAB R2014b 64bit | | | |
| 实验步骤与内容：   1. 首先是集成学习的流程，本次是训练集共有10个分类，训练集有60000个数据，测试集有10000个数据，数据较多。通过使用集成学习，选择三种方法分别训练然后测试得出结果，三者之间进行对比投票，得出最终的预测值，然后与真实值相比较得出正确率。 2. 了解流程后开始选择合适的机器学习方法。结合之前的实验，本次实验在一开始选择了KNN、贝叶斯和BP神经网络，其中BP神经网络使用的是实验5的代码，KNN和贝叶斯则参考了上学期一些课程的资料。   KNN代码：  disp('-----knn-----');  for index = 1:testCount  knn\_single = zeros(1,trainCount);  for trainIndex = 1:trainCount  temp = abs(testImages(index,:) - trainImages(trainIndex,:));  for dimensionIndex = 1:28\*28  if (temp(1,dimensionIndex)<=0.3)  knn\_single(1,trainIndex) = knn\_single(1,trainIndex) + 1;  end  end  end  cols = zeros(1,6);  for i = 1:size(cols,2)  temp = find(knn\_single==max(knn\_single));  cols(1,i) = temp(1);  knn\_single(cols(1,i)) = 0;  end  knn\_single = zeros(1,10);  for i = 1:size(cols,2)  knn\_single(trainLabels(cols(1,i)) + 1) = knn\_single(trainLabels(cols(1,i)) + 1) + 1;  end  temp = find(knn\_single==max(knn\_single));  result\_knn(index) = temp(1) - 1;  end   1. 编程完成后运行测试。实际操作中发现，BP神经网络虽然速度并不快，但约3~5分钟后还是得出了结果（我将结果保存在了mat格式文件中），然而对于KNN，我完成编程并在训练集数量600、测试集数量100的条件下完成了测试，并在睡前启动程序开始全部数据的运算，然而第二天发现电脑竟然关机了！再次运行了一个上午仍然没有得出结果。 2. 思考问题原因后，我发现应当是由于源样本的维度有28\*28整整784维，同时样本点数量也非常大，造成了电脑内存空间不足以至于崩溃。核心原因应当是维度爆炸。 3. 得出结论后我决定不再采用KNN，同时基于相同的理由放弃贝叶斯。参考教材的教学流程与各算法的特色后，决定使用后三章的神经网络、决策树与SVM进行实验。 4. BP神经网络参考上个实验，决策树部分则使用了MATLAB自带的函数，通过实际比较正确率，我在treefit函数和classregtree函数两个训练函数之间选择了后者。前者使用了ID3算法，后者则是CART算法。参数都是样本数据矩阵和类别label   训练完成后可以使用treetest测试决策树的错误率。实际上其正确率明显并不如BP神经网络与SVM，只能提供参考作用。对测试集进行测试则使用了treeval函数，得到预测值的矩阵/数组。  需要注意的是，决策树得到的预测结果是double型，这里我采取了四舍五入取整的方法，将预测结果变为整型。   1. 对于SVM部分，使用了建议的libsvm工具包，安装编译似乎相对麻烦，根据网上的建议，直接将libsvm-3.21\windows目录添加到工作路径下即可（点击 set path/设置路径），之后即可正常调用工具包方法     同时还发现该工具包下还有一个matlab文件夹，不知道有什么作用。  Libsvm的使用方法相对简单，网上也有详细的使用说明，这里只列出一些参数的值  -s svm类型：SVM设置类型（默认0)  　　　　0：C-SVC；1：v-SVC；2：一类SVM；3：e-SVR；4：v-SVR  -t 核函数类型：核函数设置类型（默认2）  　　　　0 – 线性核函数：u’v  　　　　1 – 多项式核函数：（r\*u’v + coef0)^degree  　　　　2 – RBF(径向基)核函数：exp(-r|u-v|^2）  　　　　3 – sigmoid核函数：tanh(r\*u’v + coef0)  -d degree：核函数中的degree设置（针对多项式核函数）（默认3）  -g r(gamma）：核函数中的gamma函数设置（针对多项式/rbf/sigmoid核函数）  -c cost：设置C-SVC，e -SVR和v-SVR的参数（损失函数）（默认1）  -n nu：设置v-SVC，一类SVM和v- SVR的参数（默认0.5）  -p p：设置e -SVR 中损失函数p的值（默认0.1）   1. 最后得到三个结果，进行投票。在我的实现中，若有其中两个或三个得出的结论是同一个数字，则最终投票结果是该数字；否则（即三者给出的答案都不同），以SVM给出的答案为准。得到投票结果后与真实值对比，得到正确率。   -----loading-----  -----决策树-----  -----SVM-----  \*  optimization finished, #iter = 2452  nu = 0.250993  obj = -185.541857, rho = -0.252036  nSV = 301, nBSV = 256  …………(SVM训练打印的数据)  Total nSV = 3525  Accuracy = 88.7% (8874/10000) (classification)  -----神经网络-----  -----投票中-----  -----完成-----  正确率 ： 88.9%  得到最终正确率约为88.9% | | | |
| 结论分析与体会：  这是本学期的最后一个实验，考察了多种学习器的集成使用，通过将多个单个学习器集成/组合在一起，使它们共同完成学习任务，相互弥补缺陷，使得最终的学习效果相对更好。由于对boosting和随机森林等知识学习并不够充分，因此没有使用，最终的决策方法则采用了相对多数投票法（针对分类任务）。 | | | |