计算机科学与技术学院神经网络与深度学习课程实验报告

|  |  |  |
| --- | --- | --- |
| 实验题目：组合一个简单的图像分类管道 | | 学号：201900150221 |
| 日期：9.29 | 班级： 19智能 | 姓名： 张进华 |
| Email：zjh15117117428@163.com | | |
| 实验目的：  在本作业中，练习组合一个简单的图像分类管道，基于 k-最近邻或 SVM/Softmax 和Three-Layer Neural Network分类器，实现图像分类并比较不同算法下的准确率 | | |
| 实验软件和硬件环境：  Visual studio Code python 3.9.7  Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz | | |
| 实验原理和方法  **步骤一 : k-Nearest Neighbor classifier**   1. **数据加载**   加载CIFAR-10数据，并打印训练集和测试集的图片、标签尺寸     1. **可视化部分数据**   对于训练集中的10类数据，每类随机取出7张并可视化     1. **编辑k\_nearest\_neighbor.py 文件**   先打开分类器文件夹里的 k\_nearest\_neighbor.py 文件： 首先填写函数 compute\_distances\_two\_loops，计算数据集之间的样本距离，公式就是差的平方再开根号    填写函数 compute\_distances\_one\_loop，用于计算输入和数据集中的其他已知标签的距 离，也是上面的公式，只是系数稍微有一些改变    再然后填写 compute\_distances\_no\_loop 函数，用于计算输入 x 和数据集的距离，并且不使 用循环。直接用矩阵存放结果，利用 numpy 的函数进行计算    填写 predict\_labels 这个函数，先拿出距离最近的 k 个    **4. 交叉验证**  建立 k—交叉验证算法，将数据分成好几组，可以单独训练。并使用不同的 k 值来计算哪个 k 的预测结果最好    **5. 选择最佳的k，训练分类器**  通过可视化每一个K的误差，选择最佳的K重新训练分类器    选择最佳k = 6,训练后得出准确率为0.29    **6 .问题回答**  问题1：    问题2：    问题3：    **步骤2 Training a Support Vector Machine**   1. **Linear\_svm.py 文件的填写**   首先是 svm\_loss\_naive ，其作用是用 naive 的方式计算当前模型的 loss 计算在 w\*x 操作下和目标的差别距离，如果有一定的差别，就加上相应的 loss，填充dw进行参数更新。  损失函数定义如下:      填写 svm\_loss\_vectorized,用向量的方式完成，计算其 loss，填写计算距离，并更新 loss    及根据计算的 margin 更新 dw 的值     1. **填写linear\_classifier.py 文件**   填写 train 函数，先是划分专门的训练集和测试集    SGD随机梯度下降    再计算真正要变化的 w，用学习率乘以函数返回的 dw     1. **数据预处理** 将数据划分为train训练集、val验证集、test测试集以及dev试算集，这个试算集就是一个小样本来测试程序是否能够正常运行的。这里在选取的测试集的时候我们是不能够让val测试集和train训练集有交集的，这样才能达到随机的效果。     展示一下训练样本的像素点均值绘制出来的图像    **4. 完成并运行SVM**  在训练的过程中，loss 逐渐减小，模型也更加贴近数据集。training accuracy: 0.38 validation accuracy: 0.37 换上不同的学习率和不同的长度，计算最终的预测正确率，从而选择最好的参数。    **5.交叉验证**  循环不同的参数进行预测，然后不断更新保留最准确的参数    展示出了不同参数下的预测精度的波动    最终得到最好的预测率是 0.40400 最后直接用最好的 svm 进行最终预测，并达到了 0.373 的正确率    **6.问题回答**  问题1：    问题2：    **步骤3 : Implement a Softmax classifier**   1. **修改softmax.py文件**   修改 softmax.py 函数，用 navie 方式计算 loss。即用循环计算    接下来实现向量化 的 loss 计算    **2.运行softmax.ipynb文件**  运行作业里的代码进行运算，展示循环计算得到的结果    检验计算结果是否正确，通过和导数定义求解进行对比，如果无明显差异则认为结果一致    从结果来看整体的差异非常小，所以我们认为梯度的计算是正确的。然后再比较向量计算与循环计算是否有差异    可以看到向量计算与传统的循环计算在精度上没有差异，但在运行时间上向量计算的效率明显的提高。接下来通过对不同的学习率与正则系数进行交叉验证，得到最优的参数    可以看到最好参数的预测准确度是0.360，那么我们将最好的参数代入test集看一下计算效果，  得到 0.343 的正确率    并输出每一类的模板      **3.问题回答**  问题1：    问题2：    **步骤四 Three-Layer Neural Network**   1. **编写得分、损失、梯度函数**   建立不同的神经层，每个神经层上有多个神经元，利用参数，激活函数进行预测。首先是 loss 函数中的 scores 的计算，在neural\_net.py并对其中的任务进行编辑，定义每一层的权重和偏置    对应输出    **2. 实现 loss 向前传播的过程**  损失函数是通过计算得分函数的变形得到    **3. Three-Layer Neural Network编写**  计算完得分函数、损失函数后就是需要计算梯度来对参数进行更新    **4．训练和预测函数的编写**  在计算完梯度后需要去更新权重训练模型，需要编写训练函数，但这里的只需要完成随机梯度下降的小样本选择以及参数的更新过程的代码    更新参数    预测的函数填写    **5.测试数据损失可视化**  开始训练模型更新参数，观察一下其损失函数的变化，图上可以看到随着迭代次数的增加，损失的值飞速下降，在15次跌倒后稳定在0的上面，所以整个训练过程展现出来的结果是可以接受的    **6. 对训练集调试**  使用绘制loss 、准确率曲线去观察整个的一个更新过程，或者绘制W1直观的从图像中看一下权重训练的样子，来判断一下模型的各方面参数的问题    从图上看到损失函数到第200次迭代之前都还没有开始明显的变化，与之前看到的损失函数形式不一样，这个导致的原因有可能是学习率过小导致迭代速度过慢。再看准确度函数就在0.29周围就开始平缓，可以适当增加隐藏层的神经元个数也就是维度来充分利用信息  **7. 交叉验证参数的调整**  对不同的参数进行训练并进行验证，选择出验证效果最好的参数    运行一下测试集检验    精度达到了0.494  **8.问题回答**  问题1：    问题2：    **步骤5 : Image Features**  **1.在特征上训练 SVM**  使用验证集调整学习率和正则化强度    交叉验证期间实现的最佳验证准确度    **2.测试SVM**  在测试集上评估经过训练的 SVM,得出准确率为0.418    **3. 训练Three-Layer Neural Network**  训练一个关于图像特征的三层神经网络，交叉验证各种参数，存储模型在 best\_net 变量中    交叉验证期间实现的最佳验证准确度     1. **测试Three-Layer Neural Network**     在测试集上评估经过训练的 ,得出准确率为0.568    **5.问题回答**  问题1： | | |
| 结论分析与体会：  1.了解了训练/验证/测试分割以及超参数验证数据的使用  2.了解了基本的图像分类管道和数据驱动的方法（训练/预测阶段）  3.熟练使用 numpy 编写高效的矢量代码 | | |
| 就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟1－3道问答题：  1.不清楚如何设计三层神经网络如何编写？  网上的教程大多数是关于两层神经网络的算法，所以需要自己根据其推导如何设计实现三层网络，但最终模型实现的准确率特别低，后期打算再修改一下调参  2.在三层神经网络中，由于参数系数问题，出现了矩阵维数不能够对等的情况，仔细推导参 数，再对系数进行修改，就成功了 | | |