期末项目实验报告

15331191 廖颖泓

一、分类算法介绍与分析

在本次期末项目中，我尝试使用了以下分类算法：

(1) 神经网络

在这次项目中，我使用了以下的网络结构进行训练：

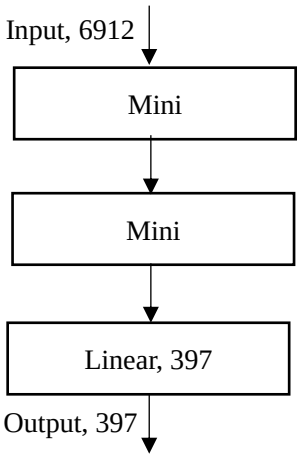

图 1 神经网络结构

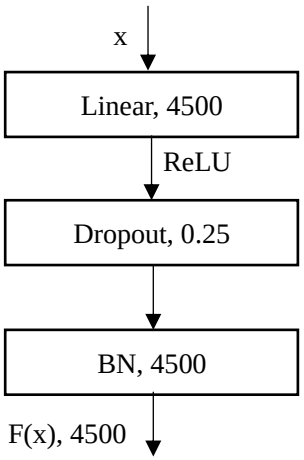

图 2 Mini 层结构

在图1中，我将进行归一化后的训练数据(76240×6812)作为Input，通过两个Mini层和一个神经元数为397的Linear层得到一个397维的向量，预测结果取向量中最大值对应的维数，即进行一次argmax()运算。Mini层结构如图2所示，先采用了一个神经元数为4500的Linear层，再用一个ReLU作为激活函数提取特征，同时防止梯度消失，然后数据通过一个Dropout层，训练中通过屏蔽25%的神经元来防止过拟合，最后通过BN层再进行一次归一化处理，可以加速收敛和控制过拟合，最终输出一个4500维的向量。具体的神经元数选择我是基于自己的调参得到的最好测试结果选取的。

上述网络结构在Pytorch的框架下进行训练。我曾尝试使用均方差(MSE)作为损失函数和Adan作为优化器进行训练，误差收敛的速度十分缓慢，在学习率为0.1的情况下，经过50个Epoch也只能收敛到0.6的准确率，因此训练中我使用了交叉熵(Cross Entropy)作为损失函数，SGD作为优化器。结果显示，在设置学习为0.1，momentum为0.9，weight_decay为0.005的情况下，该网络可以收敛到99%的准确率。这里使用momentum是为了加速SGD的收敛过程，利用冲量加快梯度的下降，并能控制梯度下降的方向；weight_decay的作用是作为正则项防止过拟合，这一项不能设置得太大，否则会导致收敛速度极慢甚至无法收敛。用ReLU激活的两层神经网络在输入时高斯分布下，存在一个全局最优点，也有一个次优的局部最小点，在随机初始化的条

件下，梯度下降算法收敛到全局最优或次优的局部最优每个都有25%的概率 [1]。因此这个方法需要进行多次尝试，在一定频率的训练后可以提高测试效果。

相比于Tensorflow，Pytorch的使用更为友好，更容易完成每个网络层之间的连接。Caffe的数据导入需要自己编写一个接口，相对比较麻烦。于是我采用了Pytorch作为框架来实现我的网络结构。我的网络结构在Pytorch中的代码如下：

```
class Net(nn.Module):
    def __init__(self):
        super(Net,self).__init__()
        # Mini 层
        self.fc_bn = nn.BatchNorm2d(6812)
        self.fc1 = nn.Linear(6812, 4500)
        self.dp1 = nn.Dropout(0.25)
        self.fc_bn1 = nn.BatchNorm2d(4500)
        # Mini 层
        self.fc2 = nn.Linear(4500, 4500)
        self.dp2 = nn.Dropout(0.25)
        self.fc_bn2 = nn.BatchNorm2d(4500)
        # Linear 层
        self.fc3 = nn.Linear(4500, 397)
    # 前向传播函数
    def forward(self,x):
        out = self.fc_bn(x)
        out = self.fc_bn1(F.relu(self.dp2(self.fc1(out))))
        out = self.fc_bn2(F.relu(self.dp2(self.fc2(out))))
        out = self.fc3(out)
        return out
if __name__ == "__main__":
    net = Net()
```

(2) PCA + Gradient Tree Boosting (GBDT)

PCA把原先的n个特征用数目更少的m个特征取代，新特征是旧特征的线性组合，这些线性组合最大化样本方差，使新的m个特征互不相关，从旧特征到新特征的映射捕获数据中的固有变异性。

Gradient Tree Boosting (GBDT)预测时先经过所有基模型的预测形成新的测试集，最后再对测试集进行预测。GBDT由多棵决策树组成，每一棵决策树学习之前所有树结论和的残差，所有树的结论累加起来做最终答案。GBDT的理论时间复杂度是O(TML)，T是树的数目，即分类器的数目，M是每棵树平均叶子数，L是样本向量的维数 [2]。GBDT在优化时只用到一阶导数

信息，内存占用不高，训练时不会造成大量的内存消耗。

(3) One-vs-all + SVM

在one-vs-all策略中，假设有n个类别，那么就会建立n个二项分类器，每个分类器针对其中一个类别和剩余类别进行分类。进行预测时，利用这n个二项分类器进行分类，得到数据属于当前类的概率，选择其中概率最大的一个类别作为最终的预测结果。

二项分类器我选择了用径向基核函数(RBF)作为核函数的SVM。RBF可以将一个样本映射到一个更高维的空间，而且与多项式核函数相比，RBF需要确定的参数较少，数值计算复杂度也比较小，比较适合维数较大的数据的分类。用RBF作为核函数的SVM的计算复杂度是O(ND)，其中N是SVM的数目，D是输入数据的维数 [3]。

虽然SVM的计算复杂度不高，但是样本总数为76240，因此SVM需要将一类与另外的396类数据进行比较划分，时间花费上是比较大的。又因为使用了one-vs-all，在样本类别数接近400的情况下需要训练出将近400个分类器，在时间开销上是巨大的。另外，one-vs-all+SVM的方法需要消耗大量的内存，对计算机的内存有很高的要求。

二、实验结果分析

(1) 神经网络

在所有测试结果中，神经网络的效果是最好的，我目前取得的最高结果是0.31301(可能会更新)。Pytorch实现的神经网络支持GPU加速，这使得训练过程最多需要8分钟就可以完成，训练拟合度很高，可以达到1.0。另外，Pytorch可以使用增量学习的方法进行训练，每次只需要取出一个batch的数据即可，不会大量消耗显存。我的整个训练过程是在Google提供免费GPU的Colab上完成的。我也使用了Kaggle的Kernel上提供的GPU进行训练，计算速度相对Colab要快一点，但是要产生输出文件需要将程序放到云端执行，不能像Colab那样随时中断程序模块的运行，不方便人为控制。

我尝试过每层1000到6000的神经元数，最后是在每层4500个神经元的网络中拿到最佳的测试结果。神经网络每层的神经元数导致的测试性能的差距只在2%，也就是网络的过拟合可能导致部分的结果预测值失准。下面给出部分训练的快照截图，可见进行了13个Epoch的迭代就可以将准确率提升到99%。

```
Accuracy : 0.98828125
Epoch : 12
Step : 0
Loss: 0.108418
Accuracy : 0.9921875
Epoch : 13
Step : 0
Loss: 0.072539
Accuracy : 1.0
Epoch : 14
Step : 0
Loss: 0.044115
Accuracy : 1.0
Epoch : 15
Step : 0
Loss: 0.044117
```

图3 神经网络训练快照

(2) PCA + Gradient Tree Boosting (GBDT)

该方法的拟合度较低，只能达到0.80左右，验证集的准确率为0.20，最终提交的结果为0.00。结果非常不好，而且耗时较长，大致需要一个下午的时间。该结果的可能由于用PCA降维导致部分特征丢失，模型偏差较大。如果将所有数据进行训练，耗时会非常长。

(3) One-vs-all+SVM

该方法的训练对时间和内存的要求十分高昂。我在阿里云租了一台32G内存的4核CPU服务器测试这个方法，该方法在使用80%的训练样本，即60992个样本进行训练时内存的初始耗费就达到了16G，最后经过两天的训练，内存的占用率也达到了98%，最后我放弃了这个方法，改用了最后帮助我拿到最高结果的神经网络。

Reference

1. Du, S. S., Lee, J. D., Tian, Y., Poczos, B., & Singh, A. (2018). Gradient descent learns one-hidden-layer cnn: don't be afraid of spurious local minima. Proceedings of the 35th International Conference on Machine Learning

2. Si, S., Zhang, H., Keerthi, S.S., Mahajan, D., Dhillon, I.S. & Hsieh, C.. (2017). Gradient Boosted Decision Trees for High Dimensional Sparse Output. Proceedings of the 34th International Conference on Machine Learning, in PMLR 70:3182-3190

3. Claesen, M., Smet, F. D., Suykens, J. A. K., & Moor, B. D. (2014). Fast prediction with svm models containing rbf kernels. Computer Science.