Text Classify 实验报告

学号-20142112280-谢非

2016 年 12 月 26 日

目录

1 实验目的

1. 应用神经网络实现简单的文本分类操作

2. 熟悉了解文本的特征提取以及多层神经网络构建

2 实验要求

1. 应用向量空间模型 (VSM) 以及 BP 神经网络实现对文本的分类

2. 力求分类正确率的提升

3 实验环境

实验代码使用 python3.5 实现，主要的 python 库有用于数组计算的 numpy. 测试数据是 `http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html`这个网站上的 20 个新闻组资料

4 实验原理

4.1 元文本数据处理

首先将每个文本的内容分词后计算频次，统计出一个字典结构，键值是单词，值是出现的频次. 在依次统计每个网页的同时，也把所有的文本的单词，频次统计出来.

1. 根据淘汰率，淘汰掉所有文本中出现次数较多的单词，默认淘汰率为 0.001
2. 总单词和词频在经过淘汰过滤之后，根据设置的文本特征向量维度，从中从大到小依次获取文本特征向量属性值（即对应单词）
3. 这实际上是第一次文本数据处理，之后要获取训练数据和测试数据还需要根据现有的总体文本向量属性来依次从每个文本数据中提取特征向量.

最终，将由文本数据中提取出来文本属性的特征向量，即 BP 神经网络的输入参数.

4.2 BP 神经网络的构造

[1] 本次实验过程中，改进了上一次实验 Web Content Identification 的 BP 神经网络实现，使其可以支持设置多个隐藏层的神经网络. 在数据学习上有了更大的进步，同时

```
class NeuralNetWork:

    def __init__(self, layers, activation, activation_deriv):
        "init a neural network"

        # config activation
        self.activation = activation
        self.activation_deriv = activation_deriv

        # init weights by layers
        self.weights = []
        for i in range(1, len(layers) - 1):
            self.weights.append(
                (2 * np.random.random((layers[i - 1] + 1, layers[i] + 1)) - 1) * 0.25
            self.weights.append(
                (2 * np.random.random((layers[i] + 1, layers[i + 1])) - 1) * 0.25)

    def train(self, trainData, trainOut, learinRate=0.1, epochs=10000):
        ''' training my net args '''
        trainData = np.atleast_2d(trainData)
        tempData = np.ones([trainData.shape[0], trainData.shape[1] + 1])
        tempData[:, 0:-1] = trainData
```

[1] 实际上本次的文本分类我的实现原理和 Web Content Identification 相差不大, 但是文本向量的提取上和神经网络的改进上有了改进

```
        trainData = tempData
        trainOut = np.array(trainOut)

        for times in range(epochs):
            # get a random inputData
            inputFlag = np.random.randint(trainData.shape[0])
            inputData = [trainData[inputFlag]]

            for curLayer in range(len(self.weights)):
                inputData.append(self.activation(
                    np.dot(inputData[curLayer], self.weights[curLayer])))
            error = trainOut[inputFlag] - inputData[-1]
            deltas = [error * self.activation_deriv(inputData[-1])]
            for l in range(len(inputData) - 2, 0, -1):
                deltas.append(deltas[-1].dot(self.weights[l].T)
                              * self.activation_deriv(inputData[l]))
            deltas.reverse()

            for i in range(len(self.weights)):
                layer = np.atleast_2d(inputData[i])
                delta = np.atleast_2d(deltas[i])
                self.weights[i] += learinRate * layer.T.dot(delta)
```

5 实验分类评价

实验测试文件选择了给的数据集中提供的测试数据集，每个种类有 100 个文件，总共 2000 个文件. 实验主要评价标准即对测试数据集的正确率 (总共数据集有约 20000 个文件).

5.1 前期测试结果

经过超参的多次修改, 得到最高的测试正确率为 83.28%　输出附件见附件
此时神经网络维度为 400*45*45*5, 学习速率为 0.1, 循环次数 100000, 文本高频词淘汰率为 0.01

5.2 后期改进超参结果

经过超参的多次修改, 得到最高的测试正确率为 85.64%　输出附件见附件
此时神经网络维度为 500*45*45*5, 学习速率为 0.1, 循环次数 1000000, 文本高频词淘汰率为 0.01

6 实验总结

本次实验过程中，我对神经网络的具体实现过程有了进一步了解，通过调试，改进，实现了神经网络的多个隐藏层的构建. 对于文本特征的提取上，减小了总文本的淘汰率（实际调试中发现文本词数较多，有四十几万，所以减小淘汰率，使得文本向量的表示更加符合模型需求）. 虽然实验过程中对于神经网络训练过程中的参数调整还不熟练，但最终通过加大训练的次数大大提高了模型的最终判断效果.