



机器学习导论作业技术报告

学号：18373447，姓名：张文彬

2021年1月

**第一部分——IMDB评论情感判断**

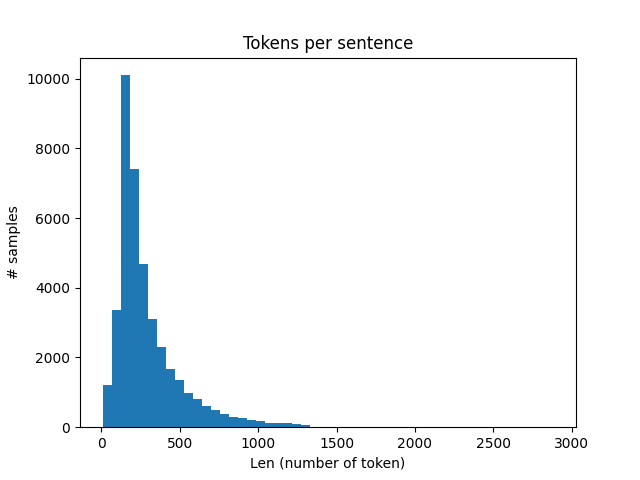
**一、题目分析**

IMDB是一个较大的影视剧信息站，题目提供了40000条被打上positive或negative标签的评论和10000条未打标签的评论，要求通过构建模型来对未打上标签的评论进行情感的判断。

通过分析训练数据，可以简要得到这些评论的特点：

1.评论并不全是规范的英文语句，数据中存在部分HTML标签，颜文字，不规范符号串等噪声，需要在处理数据时去除。

2.去除噪声后，分析评论长度分布，如下图，发现评论长度主要集中在0到500之间，因而可以考虑在处理数据时统一评论长度。



3.去除噪声后分析单词分布情况，按照单词出现频率进行排序，发现所有评论中存在很多极少出现的单词，因此可以考虑建立固定大小的单词表，将评论按照单词频率顺序映射到一个一维的向量中。

4.该问题是一个典型的二分类问题，可以使用多种模型来进行处理，如多层感知器、支持向量机等。

**二、数据处理**

**2.1 数据处理方法**

典型的文本处理方法有词袋模型、Word2vec、Doc2vec，这几种处理方法的复杂程度依次提升，处理效果也不断提高。在本次作业中，我选择了词袋模型的处理方法，这种方法比较简单且易于实现，最后模型的准确率也不错。

词袋模型是一个在自然语言处理和信息检索下被简化的表达模型。此模型下，一段文本（比如一个句子或是一个文档）可以用一个装着这些词的袋子来表示，这种表示方式不考虑文法以及词的顺序。最近词袋模型也被应用在电脑视觉领域。

**2.2 数据处理细节**

需要将每条评论转化为一个固定维数的向量，根据训练数据的分布特征，将每条评论转化为一个大小为300的向量，再输入模型进行处理。

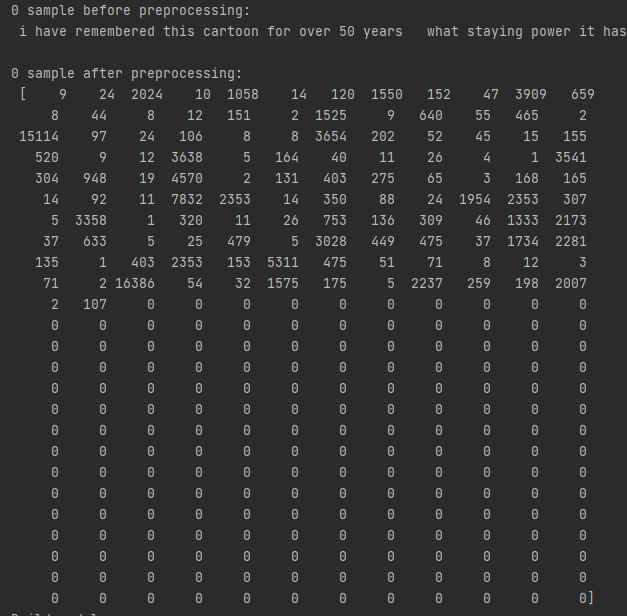
转化过程主要有以下几个步骤：

1.将所有评论（包括训练集和测试集）进行处理，进行数据清洗之后将每条评论切分成为单词的列表。

2.遍历所有单词，并生成一个词汇表，记录每个单词的出现频数，最后将词汇表按照出现频数排好序，截取前20000个最常出现的单词。

3.按照每个单词在前20000个单词的列表中的索引进行线性映射，不在词汇表中出现的单词的索引置为0，将每条评论映射为一个大小为评论长度的向量，向量的每一维的值都在0到20000之间。最后将所有向量进行规范化，统一规范到长度为300，长度大于300的截断，小于300的进行补零。

这样就将所有的评论都转化为了向量，方便模型的输入输出。



**三、模型构建**

**3.1 模型介绍**

本次作业是一个比较简单的二分类任务，因此我选择通过自己完成一个单隐层的BP神经网络来实现评论的感情分类。

BP神经网络是一种按误差逆传播算法训练的多层前馈网络，是应用最广泛的神经网络模型之一。学习规则使用梯度下降法，并通过反向春播来不断调整网络的权值和阈值，使网络的误差平方和最小。BP神经网络模型结构包括输入层、隐层和输出层。通过最小化平方误差函数，并利用反向传播不断更新模型各层的权值。



**3.2 模型实现**

要实现一个具有单隐层的简单BP神经网络，主要需要实现神经网络中前向传播和反向传播的逻辑，以下是整个神经网络模型的大致结构：

初始化：

1. **def** \_\_init\_\_(self, sizes):
2. self.num\_layers = len(sizes)
3. self.sizes = sizes
4. self.biases = [np.random.randn(y, 1) **for** y **in** sizes[1:]]
5. self.weights = [np.random.randn(y, x) **for** x, y **in** zip(sizes[:-1], sizes[1:])]

在这一步中对神经网络中各层的权值进行随机的生成，其中sizes是一个大小为3的列表，其内容是单隐层神经网络的输入层、隐层、输出层的神经元个数。各层的权值都通过numpy工具随机生成。

前向传播：

1. def freed\_forward(self, a):
2. for b, w in zip(self.biases, self.weights):
3. a = sigmoid(np.dot(w, a) + b)
4. return a

在前向传播流程中，将权值向量和输入值相乘得到该层的输出值。

反向传播：

1. def backdrop(self, x, y):
2. nabla\_b = [np.zeros(b.shape) for b in self.biases]
3. nabla\_w = [np.zeros(w.shape) for w in self.weights]
4. activation = x
5. activations = [x]
6. zs = []
7. for b, w in zip(self.biases, self.weights):
8. z = np.dot(w, activation) + b
9. zs.append(z)
10. activation = sigmoid(z)
11. activations.append(activation)
12. delta = cost\_derivative(activations[-1], y) \* sigmoid\_prime(zs[-1])
13. nabla\_b[-1] = delta
14. nabla\_w[-1] = np.dot(delta, activations[-2].transpose())
15. for l in range(2, self.num\_layers):
16. z = zs[-l]
17. sp = sigmoid\_prime(z)
18. delta = np.dot(self.weights[-l + 1].transpose(), delta) \* sp
19. nabla\_b[-l] = delta
20. nabla\_w[-l] = np.dot(delta, activations[-l - 1].transpose())
21. return nabla\_b, nabla\_w

反向传播是模型的核心部分，包括了前向传播和误差计算。

权值更新：

1. def update\_mini\_batch(self, mini\_batch, eta):
2. nabla\_b = [np.zeros(b.shape) for b in self.biases]
3. nabla\_w = [np.zeros(w.shape) for w in self.weights]
4. for x, y in mini\_batch:
5. delta\_nabla\_b, delta\_nabla\_w = self.backdrop(x, y)
6. nabla\_b = [nb + dnb for nb, dnb in zip(nabla\_b, delta\_nabla\_b)]
7. nabla\_w = [nw + dnw for nw, dnw in zip(nabla\_w, delta\_nabla\_w)]
8. self.weights = [w - (eta / len(mini\_batch)) \* nw
9. for w, nw in zip(self.weights, nabla\_w)]
10. self.biases = [b - (eta / len(mini\_batch)) \* nb
11. for b, nb in zip(self.biases, nabla\_b)]

利用样本前向传播计算得到的误差来对权值进行更新。

**3.3 模型效果**

手写的模型在训练20轮之后，在训练集上的正确率能够达到90%。在测试集上的正确率能够达到85%。

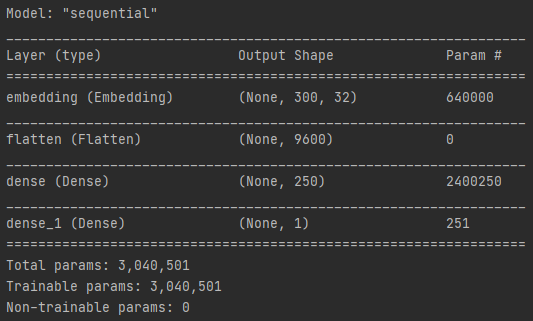


**四、利用深度学习库进行预测**

手写模型在性能和效果上都比较低，因此可以考虑使用开源的深度学习框架进行模型的构建和学习，达到更好的效果。在这一阶段，我使用了单隐层的神经网络和卷积神经网络进行模型的学习和预测。

**4.1 单隐层的神经网络**

模型结构如图：



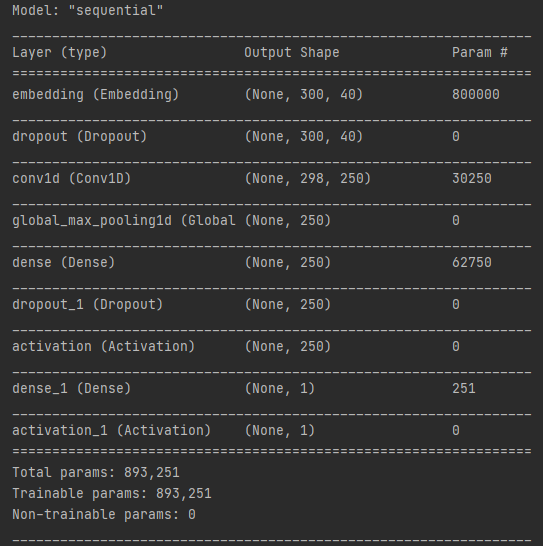
模型首先将输入的300维向量中的每一个分量转化为一个32维的向量，再通过flatten层将转化后的向量展平。将展平后的向量作为神经网络的输入层，连接到一个大小为250的隐层中，使用relu函数作为激活函数，最终输出层只保留一维，作为最终的结果。

最终训练得到的模型结果如图：



**4.2 卷积神经网络**

模型结构如图：



该模型通过一个一维卷积层来对输入的向量进行降维，再通过一个最大化的池化层来保留评论中的主要特征，再通过一个类似的单隐层的神经网络进行分类，这种做法再保证正确性的基础上有效降低了模型的参数个数，提高了模型的效率。

最终得到的模型预测准确率及排名如图：



**五、遇到的问题以及解决方法**

**5.1 数据处理**

本题的模型较为简单，利用比较简单的BP网络模型就可以得到比较好的结果，但在处理数据上比较复杂。首先，有多种处理数据的方法，如Doc2vec、Word2vec、Bert等处理自然语言的方法，这些方法的能够考虑到评论中各个单词之间的上下文关系，能够获得更好的效果，但是模型复杂度较高，需要额外的训练过程，不便于自主实现，因此我选择较为简单的词袋模型，利用单词的频率分布来对评论进行转化，最终的效果也比较好。

除了数据处理方法选择上的问题，在数据处理上，由于我是第一次接触机器学习相关的工具库，所以对数据的处理并不是很熟练，在数据的处理上还存在一些问题，比如在处理后的数据中，还存在一些分组不严谨的情况，例如对于同一单词的不同时态、语态不能进行区分。

5.2 模型构建

在自主构建单隐层的BP神经网络过程中，我遇到了比较多的问题，比如模型中数据的存储格式和运算方法，神经网络中出现的模型参数都是以numpy.ndarray的形式存储的，怎么在数据处理时就生成该类型的数据，以及如何对这种类型的数据进行操作，都是比较麻烦的，但网上关于这些问题的解答比较多，大多数问题可以比较轻松的解决。

其次在模型的构件中，如何将抽象的数学公式转化为代码也比较复杂，在这一阶段，我参考了较多网络上的神经网络代码，从这些代码上学到了很多编码技巧，并将其应用到了自主的代码编写中。

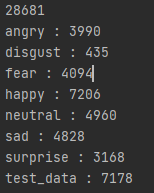
**第二部分——人脸表情分类**

**一、题目分析**

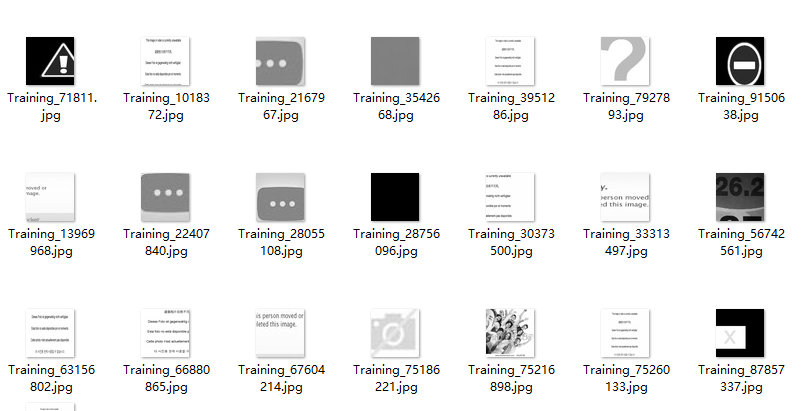
本题要求对人脸图片进行分类，训练集共给出了7个分类，28710张训练图片，测试集共有7178张待分类的图片，所有图片均为48\*48的灰度图。

题目中所有数据来自FER2013数据库，通过观察和分析数据，发现数据有如下特点：

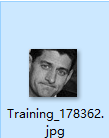
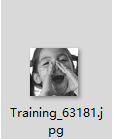
1. 各类型训练样本分布不均衡，如下图，可以看出disgust样本明显过少，而happy类型的样本数量几乎是其他类型样本数量的近两倍，因此在训练过程中，对一些类型的图片的预测效果会不太好。



2. 训练样本中存在一些无意义的图片，如下图，这些图片并非人脸表情，只会成为训练过程中的噪声。



3. 训练样本中存在较多类型模糊的图片，例如这张被分在angry分类中小女孩捂嘴笑的图片，和这张被分在angry分类中的中性表情图片。这些训练集中的样本的特征较为模糊，很难将其准确划定到某个具体的类别，因此不利于模型的训练。



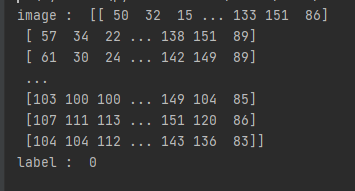
**二、数据处理**

**2.1 图片数据处理方法**

首先排除无意义的图片，并将这些图片移除。之后从各个文件夹读取每个分类下的训练图片，并为其打上标注，一共有七个分类，因此将每张图片映射到0-6的对应整数值当中。

而对于图片本身，使用skimage工具将灰度图转化为一个二维的矩阵，以便模型进行后续的处理。

最终处理后的数据如下图：



**2.2 图片数据处理细节**

对于无意义图片的移除，直接使用人工方式手动筛选。

对于数据的处理，使用h5py工具来存储数据并进行保存，处理后的数据不再变动，可以直接应用于模型的拟合当中。

1. # 读取文件夹下对应的所有文件
2. files = os.listdir(surprise\_path)
3. **for** filename **in** files:
4. image = io.imread(os.path.join(surprise\_path, filename))
5. data\_x.append(image)
6. data\_y.append(6)
8. # 将读到的数据存储到dataset中，方便模型调用
9. data\_file = h5py.File('all\_data.h5', 'w')
10. data\_file.create\_dataset('data\_pixel', dtype='uint8', data=data\_x)
11. data\_file.create\_dataset('data\_label', dtype='int64', data=data\_y)
12. data\_file.create\_dataset('test\_data', dtype='uint8', data=test\_data)
13. data\_file.close()

由于本题为一个多分类问题，将最终结果映射到一个一维的数字当中不方便处理，因此将分类标记转化为独热码，独热码长度为7，当独热码的一个分量置1时，代表对应图片属于该分量对应的分类。

对分类标签的映射可以采用keras工具包中的to\_categorical方法，将所有证书标签转化为独热码。

1. all\_val = to\_categorical(all\_val, 7)

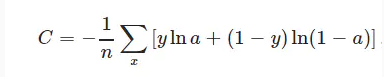
**三、模型构建**

**3.1 卷积神经网络**

卷积神经网络是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元，对于大型图像处理有出色表现。

卷积神经网络由一个或多个卷积层和顶端的全连通层（对应经典的神经网络）组成，同时也包括关联权重和池化层。这一结构使得卷积神经网络能够利用输入数据的二维结构。与其他深度学习结构相比，卷积神经网络在图像和语音识别方面能够给出更好的结果。这一模型也可以使用反向传播算法进行训练。相比较其他深度、前馈神经网络，卷积神经网络需要考量的参数更少，使之成为一种颇具吸引力的深度学习结构。

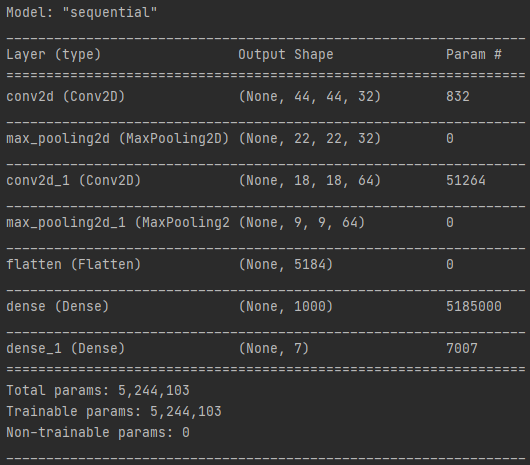
**3.2 模型的损失函数**



本问题是一个多分类问题，不同于二分类问题中使用平方误差函数来作为目标函数，多分类问题采用交叉熵损失函数的效果要更好。

**3.3 卷积神经网络的应用**

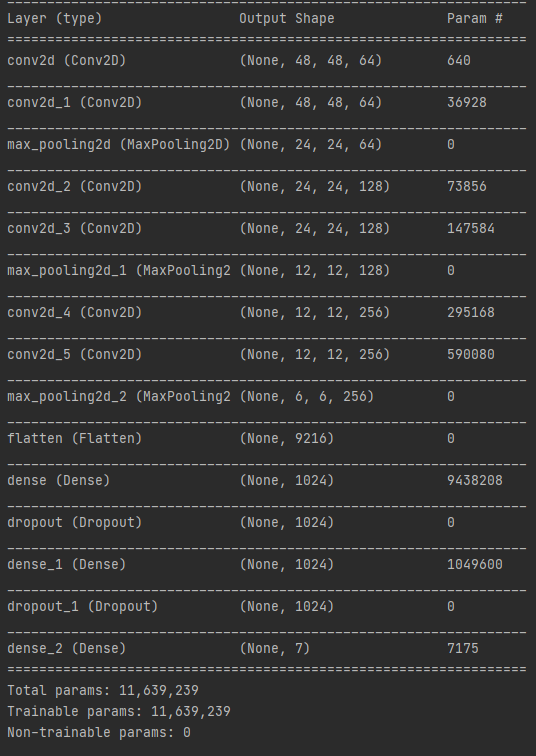
对于本问题，仍然可以按照评论情感识别题目一样，利用手写的BP神经网络进行处理，但是效果很差，拟合15轮之后在训练集上的准确率只能达到40%多，因此采用深度学习工具库来构建模型进行学习，以取得更好的效果。



在第一次尝试中，我尝试使用两个卷积层和两个池化层来进行模型的拟合，卷积核的大小为5\*5，在通过各两个卷积层和池化层的处理之后，将尺寸为9\*9的图像展平，并连接到一个含有1000个神经元的隐层当中，最后连接到7个神经元的输出层。

该模型在拟合约25轮之后，在训练集上的准确率达到了90%，但最终在测试集上的效果只达到55%左右，

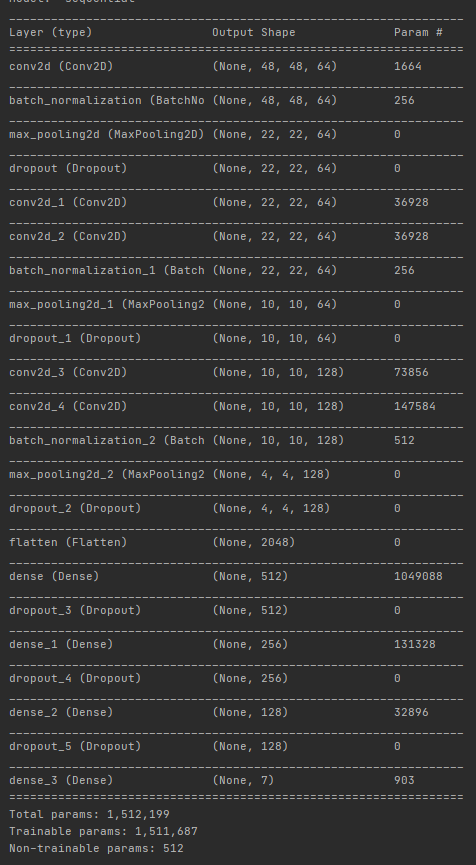
在第二次尝试中，我选择增加卷积层与池化层，得到的模型如下图：



在模型的设计上，我参考了VGG网络的特征，将所有卷积层的核的大小都设置为3，并逐渐增大卷积网络的大小，经过三次池化之后将图片降维到6\*6大小，展平后连接到两个大小为1024的隐层中。该模型由于增加了卷积层和隐层，因此训练速度较慢，在训练集上训练一轮的时间需要八分钟左右，比较耗时。经过25轮迭代，在训练集上的正确率能够达到99%，但在测试集上的表现仍然较差，可能的原因有两个，一是自身的模型还不够完善，存在进一步改进的空间，二是训练数据自身的性质使然。最终得到的结果第一次的神经网络好一些。最终模型的正确率能够达到58%左右。



在第三次尝试中，我在第二次网络基础上进行改进，在保留三层卷积层的基础上加深了隐层，将2个1024维的隐层转换成了3个维数更小的隐层，在增加层数的同时没有引入更多的参数，使得模型的训练时间与第二次尝试中模型的训练时间相差不大。经过50轮的训练之后，模型在训练集上的准确率能够达到73%，不如前两个模型，但在测试集上的准确率却有所提高，达到了62%。



最终得到的准确率与排名如图。



**四、遇到的问题及解决方法**

**4.1 模型的设计**

由于笔记本电脑的机能限制，对于较深的神经网络，训练速度很慢，为了节省时间，在模型的选择上，我最多使用了4个卷积层、2个池化层和2个全连接隐层，尽量使模型能够在较短的时间内训练完毕。

在模型的选择上，我通过查阅资料，了解到在图像识别领域，比较好的模型有VGG模型和ResNet模型，并分别在电脑上进行尝试，但由于这些模型比较复杂，每轮的训练时长需要二十多分钟，并且容易出现陷入局部极值的情况，在拟合两到三轮后，在训练集上的正确率一直维持在27%左右，比较令人遗憾。

**4.2 池化层的选择对模型拟合效果的影响**

使用池化层可以对提取特征后的图像进行降维，在本题中，我尝试了两种池化方式，分别是MaxPool和AvgPool，两种方法分别提取池化块的最大值和平均值，两种不同的池化方式得到的效果也有所不同。在选择平均值池化方式的模型中，模型的拟合速度较慢，大约需要三十五轮以上才能达到稳定，且最终在训练集上的准确率不高，但在测试集上测试时，表现和使用最大值池化的方式相差不多。在选择最大值池化的模型中，模型的拟合速度较快，只需要二十五轮左右就可以达到稳定。最终在训练集上的准确率也较高。

对模型进行增加卷积层和池化层的行为可能使模型过快拟合完毕，导致最终的准确率很低。可能是训练集中图片的尺寸较小，因此在池化层数较多的时候，训练效果反而不好，可以将池化层数量设置在两层或三层，在降低输入数据维度的基础上，也不失去数据的主要特征。

4**.3 训练时的参数调整**

训练轮次，样本大小，训练过程中误差函数的选择，训练时优化器的使用，都会影响到最终结果。通过多次尝试发现，减小批尺寸、增加训练轮次都能够小幅度提升模型的训练效果。

**五、总结**

在完成两个作业题目的过程中，我查阅了很多资料并且自己动手完成了一个比较简单的反向传播神经网络，对机器学习知识的掌握也从对理论的了解深入到实践过程中的掌握，同时，通过体验各种机器学习算法和深度学习框架，我也掌握了一些解决机器学习问题的有效方法，学到了一些解决问题的新思路、新视角，对自身的工程能力和实践能力都是一种提高。

两次作业的综合情况如下：

IMDB评论情感判断



人脸表情识别



相关代码文件上传到github：