NLP-Beginner : 自然语言处理入门练习

任务一 : 基于机器学习的文本分类

徐吕啸威　河北大学

xlxw@xlxw.org　https://github.com/xlxwalex

LaTeX　2019.5.20

摘要

文本分类问题是自然语言处理领域的一个分支。本实验是一个针对影评数据进行文本情感多分类的问题，通过构建文本的 BoW 特征以及 N-gram 特征，使用 Softmax 进行多分类，达到了一定的分类效果。首先针对文本分类的理论知识进行了学习。了解了文本清洗的步骤、针对文本特征的向量化表示以及分类模型 Logistic 回归和其在多分类问题上的推广模型 Softmax 回归。其次利用所学的理论知识设计了四组对照实验，分别探究了两种不同特征、不同的学习率、不同的参数学习方法以及数据集是否做 shuffle 对分类器性能的影响。

关键字：文本分类，BoW 模型，N-gram 模型，Logistic 回归，Softmax 回归

目 录

1 前言

在自然语言处理领域，往往会需要对文本进行分类。其中基于机器学习的方法是常用的文本分类方法。其利用标注好的训练集通过机器学习模型构建分类器，将其作为模型用于对测试集进行分类。本实验中通过机器学习的方法，对烂番茄电影评论数据进行情感分类，得到了情感语义的多分类器，本次试验针对不同的特征以及不同的分类模型进行测试，得到了针对不同情况模型性能的对比结果，根据得到的结果，其现实意义是可以探究得到电影的基调以及该电影评论是否应该被正确展示等诸多信息。

2 问题分析

根据邱老师的 Nlp-Beginner 仓库中基于机器学习的文本分类任务以及 Kaggle 中 Sentiment Analysis on Movie Reviews 竞赛的描述，此为情感语义的多分类任务，数据集是最初由由 Pang 和 Lee 采集的用于情感分析的烂番茄中的电影评论语料库。根据训练集中的数据，要求按五个等级标记短语，分别是消极、有点消极、中立、有点积极以及积极。但是由于句子中的否定、讽刺、语言模糊以及许多其他障碍使这个任务变得困难。

本实验中主要分为三个步骤，其示意图如下图 2-1 所示。首先在对 **train.tsv** 训练集中数据进行检查时，统计得到其共有 156060 行数据，共 8544 个句子。由于数据并无缺失以及 NaN 值，并且通过一致性检验发现所有数据都符合要求，但是需要针对一般的文本清理流程，在 §3 中会进行详细的介绍。

在做特征工程时，根据要求需要用到**词袋模型 (Bags-Of-Words:BoW)** 以及 **N 元模型 (N-gram Model)** 对训练集数据进行特征提取，其中词袋模型的输出是一个向量，而 N-gram 模型输出的是一句话的概率, 在 §4 终将会做详细的介绍。

得到数据特征后，就需要对分类器模型进行训练了。在本实验中，**对数几率回归 (Logistic Regression:LR) 模型**以及其推广模型 **softmax 回归 (Softmax Regression)** 进行了测试。并且通过使用**交叉熵 (Cross Entropy) 损失**探究了不同**梯度下降算法 (Gradient Descent：GD)** 以及其不同的**学习率 (Learning Rate)** 的设置对模型性能的影响，关于模型训练的内容将在 §5 中将进行介绍，而实验结果会在 §6 中进行对比分析。

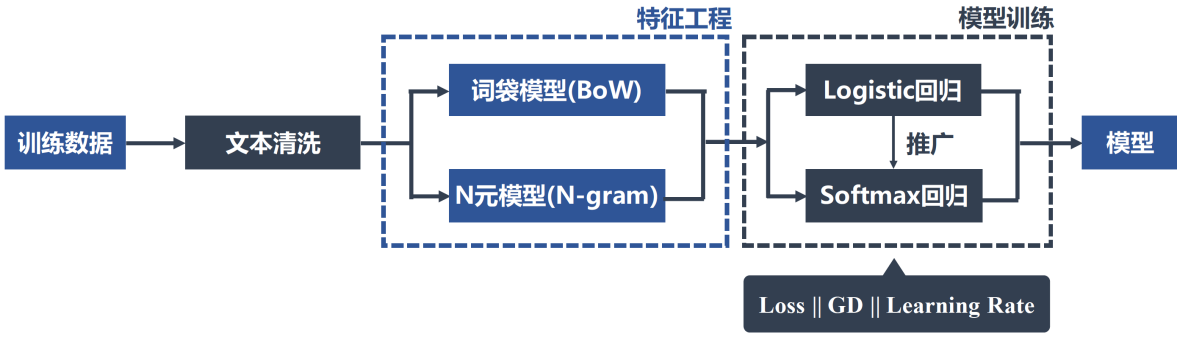

图 2-1 本次实验的步骤示意图

3 文本清洗

在数据分析时，往往需要进行数据清洗，其可以对一些错误的数据进行处理，以及对一些无关紧要的进行缩减，可以提高模型的性能以及训练的速度。在自然语言处理领域，数据清洗往往先判定数据集有无缺失、符不符合一致性原则。这之后虽然文本清洗受所做的任务影响比较大，但是有一些通用的清理流程标准是都需要的，包括如下几个方面

- **标准化 (Normalization)**

  在文本清洗中，需要 Normalization 的处理是因为在英文中所有句子第一个单词的首字母一般是大写，所以通过 Normalization 可以预先把语料中所有的英文都变成小写

- **分词 (Tokenization)**

  对于英文自然语言处理，Tokenization 就是将每个句子通过空格拆分为一系列的单个单词，方便我们后续使用这些文本。

- **筛去停用词 (Stop Word)**

  Stop Word 是无含义的词，不会给句子增加太多含义。故为通过筛掉停用词可以减少要处理的词汇量，从而降低后续的复杂度。

4 特征工程

在对文本分类之前，需要先提取文本的特征，其作用是把原始数据转变为模型训练数据的过程，在本实验中，主要用了两种特征提取方法，分别是

- **词袋模型 (Bags-Of-Words:BoW Model)**
- **N 元模型 (N-Gram Model)**

在本节中将对以上两种特征模型进行详细的介绍。

4.1 词袋模型

词袋 (Bags of Words: BoW) 模型 [1] 是一种常见的文本的特征提取方法，实现也较为简单。其将文本看成若干个单词的一个集合，忽略单词之间的顺序、语法等要素，只关注单词在文本中的出现频率，将其作为文本的特征。

词袋模型的假设：一句话中的每个单词都互相独立，不依赖于其它单词。

词袋模型特征构建的步骤：

1. 建立字典 X，给语料中每一个单词分配一个 ID 号
2. 得到文本的特征 $X[i, j]$，其中 i 是单词的 ID，而 j 是单词在此语料文档中的出现频次

通过上述步骤，可以由如下例 4-1 进一步理解，本例使用了所给数据集 {Classify the sentiment of sentences from the Rotten Tomatoes dataset} 中训练集随机选取的两条数据

例 4-1 词袋模型特征提取步骤举例

1. 选定的数据如下所示

```
4102    154 of putting the weight of the world on his shoulders 2
4103    154 putting the weight of the world on his shoulders    2
```

2. 根据步骤 1，将两句话组成的语料中的单词作为一个字典，可得到由单词及其 ID 组成的 X 为

```
X={"of":0, "putting":1, "the":2, "weight":3, "world':4, "on":5, '
    his':6, 'shoulders':7}
```

3. 根据步骤 2，统计每个单词在句中出现的频数为

```
{"of":2,"putting":1,"the":2,"weight":1,"world":1,"on":1,"his":1,"
    shoulders":1}
{"of":1,"putting":1,"the":2,"weight":1,"world":1,"on":1,"his":1,"
    shoulders":1}
```

4. 得到的每一句话的 BoW 特征向量为

```
Setence1 = [2,1,2,1,1,1,1,1]
Setence2 = [1,1,2,1,1,1,1,1]
```

从上述词袋模型的原理可得出: 词袋模型对于单词之间的顺序、语法等要素并不关注，仅关注了其单词频数，其优点是原理简单，复杂度低。缺点是分类的性能不好。

4.2 N 元模型

N 元模型 (N-gram)[2] 模型是一种语言模型，其输入是一句话，输出是这句话的概率，也就是句中单词的联合概率。N-gram 有两个特点，第一个特点是某个词的出现依赖于其他若干个词，第二个特点是得到的信息越多，预测越准确。

N-gram 模型的 N 指的是由 N 个具有先后顺序单词组成的集合，常用的有以下两种 N 值

- Bi-gram（N=2)
- Tri-gram (N=3)

例 4-2 N-gram 模型的中 N 的含义举例

对于如下所示的语句

```
putting the weight of the world on his shoulders
```

用 Bi-gram 可将其分解为

```
{putting,the},{the,weight},...,{his,shoulders}
```

而 Tri-gram 则将其分解为

```
{putting,the,weight},{the,weight,of},...,{on,his,shoulders}
```

N-gram 模型的优缺点

- **优点**

  N-gram 包含了前 N-1 个词所能提供的全部信息，这些词对于当前词的出现具有很强的约束力, 所以相较于词袋模型，N-gram 模型能够体现出考虑句子中单词之间的顺序

- **缺点**

  其缺点在上文中基本已经讨论过了，主要有如下三方面的缺陷

  1. 当 N 的取值较大时，特征空间过大
  2. 对于一些长句的单词存在长距离依赖关系，当 N 较小时无法捕捉到长距离的依赖信息

5 模型训练

在经过了数据清洗和特征提取后，根据 §2 中介绍的实验步骤，下一步就是模型的训练了。由问题分析可知，这是一个多分类任务，输出总共包括 5 个类别分别是消极、有点消极、中立、有点积极以及积极。本实验中使用了两个分类器，分别是 **Logistic 回归**以及基于其完成多分类任务的 **Softmax 回归**，故本节将对这两个分类器模型及其关系、不同的损失函数以及不同的 **(梯度下降 (Gradient Descent : GD)** 算法进行介绍。

5.1 分类模型

5.1.1 Logistic 回归

Logistic 回归 (Logistic Regression : LR)[1] 是一种用于二分类的线性模型，通过引入非线性函数 $g : \mathbb{R}^d \to (0, 1)$ 来预测测类别标签的后验概率 $p(y = 1|\mathbf{x})$。经过 Logistic 函数作为激活函数将线性模型的输出变换到 $(0, 1)$ 区间，即可表示类别标签的后验概率。设线性模型的输出为 $f(\mathbf{x}, \mathbf{w})$，$g(\cdot)$ 为激活函数，则标签 $y = 1$ 的后验概率如下式 (5-1) 所示

$$p(y = 1|\mathbf{x}) = g(f(\mathbf{x}, \mathbf{w})) = \frac{1}{1 + \exp\left(-\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)} \tag{5-1}$$

同理，标签 $y = 0$ 的后验概率如下式 (5-2) 所示

$$p(y = 0|\mathbf{x}) = 1 - p(y = 1|\mathbf{x}) = \frac{\exp\left(-\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)}{1 + \exp\left(-\mathbf{w}^{\mathrm{T}}\mathbf{x}\right)} \tag{5-2}$$

对式 (5-1) 进行变形，得到 $\mathbf{w}^{\mathrm{T}}x$ 如下式 (5-3) 所示

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} = \log \frac{p(y = 1|\mathbf{x})}{1 - p(y = 1|\mathbf{x})} = \log \frac{p(y = 1|\mathbf{x})}{p(y = 0|\mathbf{x})} \tag{5-3}$$

其中的 $\frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})}$ 即为样本 x 的正反后验概率的比值，以上即为 Logistic 回归的基本原理了。但是根据 §2 中题目的描述，本实验针对的是多分类器，而 Logistic 回归则针对的是二分类任务，这里需要推广到 Softmax 回归模型来完成多分类任务。

5.1.2 Softmax 回归

在上一小节中，对 Logistic 回归模型进行了介绍，本小节所要介绍的 **Softmax 回归模型 (Softmax Regression)**[1] 是 logistic 回归在多分类问题的推广。

多分类问题中，若有 C 个类别，则标签是 $y \in \{1, 2, \cdots, C\}$，设 w_c 是第 C 类的权重向量，则对于一个给定的样本，其通过 Softmax 回归预测得到的条件概率如下式 (5-4) 所示

$$p(y = c|\mathbf{x}) = \text{Softmax}\left(\mathbf{w}_c^{\mathrm{T}}\mathbf{x}\right) = \frac{\exp\left(\mathbf{w}_c^{\top}\mathbf{x}\right)}{\sum_{c=1}^{C}\exp\left(\mathbf{w}_c^{\top}\mathbf{x}\right)} \tag{5-4}$$

Softmax 回归的决策函数如下式 (5-5) 所示

$$\hat{y} = \arg\max_{C} p(y = c|\mathbf{x}) = \arg\max_{C} \mathbf{w}_c^{\mathrm{T}}\mathbf{x} \tag{5-5}$$

上式中 $\hat{y}$ 即为最终预测得到的样本 x 所属的标签类别。

5.2 损失函数

损失函数 (Loss Function) 是用来量化模型预测和真实标签之间的差异的函数，其主要作用就是衡量模型模型预测的好坏，对于我们而言，总是希望模型预测和真实标签之间的差异尽量小为好 (不发生过拟合为前提)。所以通过损失函数，可以很好的判断当前模型的运行情况，通过损失越来越小直至收敛，即意味着模型训练完成，在 Logistic 回归模型以及 Softmax 回归模型中，使用的是**交叉熵 (Cross Entropy) 损失**, 本小节将对其进行介绍

交叉熵损失函数 (Cross Entropy Loss Function) 一般用于分类问题，对于模型 $f(\mathbf{x}, w)$ 而言，其输出是类别标签的条件概率分布，假设其分类标签为 $y \in \{1, \cdots, C\}$，则模型输出的条件概率如下式 (5-6) 所示

$$p(y = c|\mathbf{x}, w) = f_c(\mathbf{x}, w) \tag{5-6}$$

并且上式需要满足如下 (5-7) 条件

$$f_c(\mathbf{x}, \theta) \in [0, 1] \qquad \sum_{c=1}^{C} f_c(\mathbf{x}, w) = 1 \tag{5-7}$$

设输出的样本标签为 C 维的 one-hot 向量 y，且输出的标签为 $k(k \leq C)$，那么向量 $y = [0, 0, \cdots, \underset{\uparrow k}{1}, \cdots]^T$，对于标签向量 y 而言，可以被看作是真实概率分布，如前例子，仅当第 k 类概率为 1，其余皆为 0。

交叉熵的概念来自香农的信息论，用于度量两个概率分布间的差异性，分类器的预测分布 $f(\mathbf{x}, w)$ 以及真实概率分布 y 之间的差异即可以用交叉熵来进行表示，下式 (5-8) 即为这两个不同分布的交叉熵 $\mathcal{L}(\cdot, \cdot)$ 的定义

$$\mathcal{L}(\mathbf{y}, f(\mathbf{x}, w)) = -\sum_{c=1}^{C} y \log f_c(\mathbf{x}, w) \tag{5-8}$$

由于 y 是一个 one-hot 向量，只有一个值为 1，故上式 (5-8) 可改写为下式 (5-9) 所示

$$\mathcal{L}(y, f(\mathbf{x}, w)) = -\log f_y(x, w) \tag{5-9}$$

对于交叉熵损失而言，其曲线示意图如下图 5-1 所示

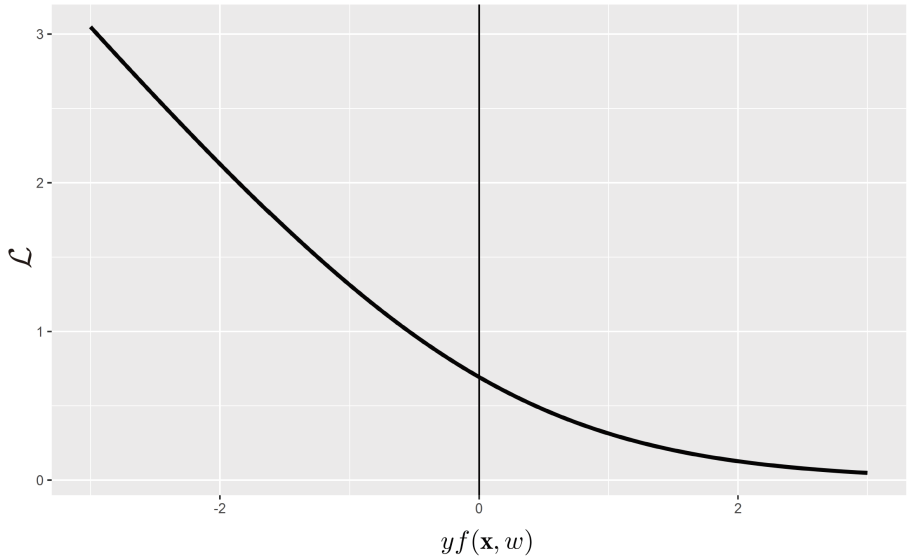

图 5-1 交叉熵函数曲线示意图

5.3 梯度下降算法

梯度下降法 (Gradient Descent)[3][4] 是一种迭代法, 在机器学习中用于求解模型参数，是最常采用的方法之一。在求解损失函数的最小值时，可以通过梯度下降法来逐渐迭代进行求解，以此得到最小化的损失函数及其对应模型的参数值。在机器学习中，基于基本的梯度下降法主要分为**批量梯度下降法 (Batch Gradient Descent:BGD)**、**随机梯度下降法 (Stochastic Gradient Descent:SGD)** 以及**小批量梯度下降法 (Mini-batch Gradient Descent:MBGD)**，在本小节中将会对它们分别进行介绍。

5.3.1 批量梯度下降法

批量梯度下降法 (Batch Gradient Descent:BGD) 是梯度下降法最初的形式，其思想是在更新参数时使用**所有的样本**来进行更新，设损失函数的形式为 $\mathcal{L}(w)$，需要学习的参数是 w_c，对损失函数求偏导如下式 (5-9) 所示

$$\bigtriangledown \mathcal{L}(w_c) = \frac{\partial \mathcal{L}(w)}{\partial\, w_c} \tag{5-9}$$

那么，对于 BGD 算法而言，通过下式 (5-10) 来更新参数 w

$$\mathcal{L}(w_c) = w_c - \bigtriangledown \mathcal{L}(w_c) = w_c - \frac{\partial \mathcal{L}(w)}{\partial\, w_c} \tag{5-10}$$

上式 (5-10) 中是直接根据梯度进行下降的，但是这可能导致下降速度过快，导致模型无法最终达到最优解 (跨过最优值)；也可能下降速度太慢，收敛时间长，所以这里引入了一个新的概念-**学习率 (Learning Rate)**，设学习率为 η，则上式 (5-10) 可改写为下式 (5-11)

$$\mathcal{L}(w_c) = w_c - \eta \frac{\partial \mathcal{L}(w)}{\partial\, w_c} \tag{5-11}$$

通过设置学习率的大小就可以控制收敛速度了，在本实验中设计了不同学习率下对模型性能以及训练速度影响的实验。

另外，对于上述批量梯度下降法而言，其得到的是参数的全局最优解，这是它的优点。但是损失函数求导时的分子为 $mathcalL$，所以可知其用到了训练集所有的数据，如果样本数目很大，则会造成迭代速度非常的慢，故出现了下面的随机梯度下降法。

5.3.2 随机梯度下降法

批量梯度下降法在更新每一个参数时，需要所有的训练样本，而**随机梯度下降法 (Stochastic Gradient Descent:SGD)** 是为了解决批量梯度下降法的缺点而提出的。

随机梯度下降算法的步骤如下所示

1. 先对训练集中所有的数据作 Shuffle 操作，即完成了随机的过程
2. 对训练集中的样本逐个迭代更新参数，直到损失函数收敛

随机梯度下降和批量梯度下降法的区别在于其通过每个样本来迭代更新，如果样本量很大的情况，可能只需要用其中几万条甚至几千条的样本，就已经将使参数 w 迭代到最优解了，所以对比上面的批量梯度下降，随机梯度下降算法速度要快得多，这是其最大的优点。但是同时 SGD 伴随的问题是噪声的影响大，使得 SGD 并不是每次迭代都向着整体最优化方向，可能会存在震荡的现象。并且其还有一个缺点-最终的优化结果不是全局的最优解，故准确率上较批量梯度下降法而言较低。为了更好的解决随机梯度下降法的缺点，下面将会对一种折中的方案-小批量随机梯度下降算法进行介绍。

5.3.3 小批量梯度下降法

小批量梯度下降法 (Mini-batch Gradient Descent:MBGD) 是上面两种方法的折中之法，原理是将整个数据集分为多个 Batch。假设整个数据集的样本空间为 M，每一个 Batch 包含 N 个样本 $(N) \leq M)$，MBGD 在每轮迭代用一个 Batch 进行参数更新，直到算法收敛，达到了最优解。

综上所述，关于梯度下降算法，可以总结得到如下结论

- 批量梯度下降法 BGD 的特点是每一次用所有的样本对参数进行更新，优点是可以达到全局最优解；缺点是当样本空间大时，每一次迭代需要的时间过长
- 随机梯度下降法在一开始先对样本数据做了 Shuffle，之后针对每一个样本对参数进行更新，优点是迭代次数快，较少的时间即可达到最优解；缺点是准确度不高，往往达不到全局最优解，且梯度下降的过程会出现震荡
- 小批量梯度下降法是一个折中方案，通过将整个数据集划分成几个较小的 Batch 对参数进行更新，特点是迭代速度较快，且能相较于 SGD 有更好的准确率

5.4 参数学习

从上述内容可知，Softmax 回归使用交叉熵作为其损失函数，且通过梯度下降法进行优化参数，对于给定的 N 个训练样本，为方便对 Softmax 函数求导，设 $a_i = \mathbf{w}_c^{\mathrm{T}} x$，将上述式 (5-4) 改写为下式 (5-5)

$$p(y=c|a_i) = \frac{\exp(a_i)}{\sum_{c=1}^{C} \exp_c(a_i)} \tag{5-12}$$

Softmax 函数可以将样本的输出转变成概率密度函数，由于我们需要通过梯度下降法去对参数进行优化，所以需要对 Softmax 的参数求梯度得到如下 (5-13) 式

$$\frac{\partial p(y=c|a_i)}{\partial a_j} = \frac{\partial \frac{\exp(a_i)}{\sum_{c=1}^{C} \exp_c(a_i)}}{\partial a_j} \tag{5-13}$$

通过商的求导法则，可将上式化简为如下 (5-14) 式

$$\frac{\partial \frac{\exp(a_i)}{\sum_{c=1}^{C} \exp_c(a_i)}}{\partial a_j} = p(y=c|a_i) - p(y=c|a_j) \tag{5-14}$$

对上式针对 $(i=j)$ 以及 $(i \neq q)$ 进行分类讨论, 得到下式 (5-15)

$$\frac{\partial p(y=c|a_j)}{\partial a_j} = \begin{cases} p(y=c|a_i)\left(1 - p(y=c|a_j)\right) & \text{if } i=j \\ -p(y=c|a_j) \cdot p(y=c|a_i) & \text{if } i \neq j \end{cases} \tag{5-15}$$

对模型优化而言真正起到作用的交叉熵损失函数的梯度，设 y^n 是 one-hot 向量形式的标签，而 $\hat{\mathbf{y}}^{(n)}$ 是 Softmax 函数输出的后验概率，交叉熵损失的定义式为 (5-8)，通过链式法则以及 Softmax 求梯度的结论即可得到交叉熵损失的梯度如下式 (5-16)

$$\frac{\partial \mathcal{L}(w)}{\partial w} = -\frac{1}{N} \sum_{n=1}^{N} \mathbf{x}^{(n)} \left(\mathbf{y}^{(n)} - \hat{\mathbf{y}}^{(n)}\right)^{\mathrm{T}} \tag{5-16}$$

采用梯度下降法即可以 (5-16) 和学习率的乘积作为步长对参数进行学习。

5.5 评价指标

在机器学习中，为了衡量一个模型的性能。需要给定一个测试集，通过训练好的模型对测试集中的每个样本进行预测，根据预测结果得到模型的评价得分来衡量模型的好坏。其中对于本实验的分类问题，常见的评价标准有正确率、准确率、召回率和 F 值等，在本次试验中，使用 F1-Score 来衡量模型的性能以进行比较，其计算方法将在本节中进行介绍。

5.5.1 查准率 & 查全率

最常见的性能指标是**准确率 (Accuracy)**，但是对于一个多分类的任务，准确率是对所有类别的整体性能进行求平均而得到的。故针对这样一个多分类任务，无法很好的体现模型的性能。所以引入了**查准率 (Precision)** 以及**查全率 (Recall)** 的概念。在对查准率以及查全率进行介绍前，首先需要知道对于模型在测试集伤上的结果，可分为以下四种情况

- 真正例 (True Positive:TP) 模型预测的类别为 C，其真实类别也是 C
- 假负例 (False Negative:FN) 样本的真实类别为 C，模型预测错误判别为其他类
- 假正例 (False Positive:FP) 样本的真实类别为其他类，但模型预测错误的判别为 C
- 真负例 (True Negative:TN) 样本的真实类别为其他类，模型预测出来的也是其他类

根据上述四种情况的描述，可以用下表 5-1 来进行概括

表 5-1 分类结果的混淆矩阵

真实情况	**预测结果**	
	正例	**反例**
正例	TP(真正例)	FN(假反例)
反例	FP(假正例)	TN(真反例)

通过上述基本概念，可以定义查准率以及查全率的概念如下所示

查准率 (Precision) 亦称为精度，其表示的是所有预测为类别 C 的样本中，预测正确的比例，用下式 (5-17) 表示

$$Precision = \frac{TP}{TP + FP} \tag{5-17}$$

查全率 (Recall) 亦称为召回率，其表示的是所有真实标签为类别 C 的样本中，预测正确的比例，用下式 (5-18) 表示

$$Recall = \frac{TP}{TP + FN} \tag{5-18}$$

对于查准率以及查全率而言，其往往是一对矛盾的量。一般来看，查准率高的查全率往往低；相反的，查全率高的查准率往往低。故为了衡量复杂分类任务重分类器的性能，需要引入下一小节中的 F1 分数进行度量。

5.5.2 F1 分数

在一些场景中，对于查全率以及查准率的重视程度不同，故通过 F 值的度量可以反映对查准率以及查准率的偏好程度，F 值的一般形式记做 F_β，其定义式如下式 (5-19) 所示

$$\mathcal{F}_\beta = \frac{(1 + \beta^2) \times Precision \times Recall}{\beta^2 \times Precision + Recall} \tag{5-19}$$

对于本次实验所要求的任务，对查准率以及查全率没有明确的要求，故设 $(\beta = 1)$，F_β 即成为了 F1 Score，如下式 (5-20) 所示

$$\mathcal{F}_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5-20}$$

另外，对于多分类任务而言，一般做法是在各个分类的混淆矩阵中分别计算得到查准率以及查全率，这样就得到了**宏查准率 (Macro Precision)** 以及**宏查全率 (Macro Recall)**，即得到了相应的**宏 F1 值 (Macro F1 Score)**, 其定义如下式 (5-21) 所示

$$Macro - \mathcal{F}_1 = \frac{2 \times Macro - Precision \times Macro - Recall}{Macro - Precision + Macro - Recall} \tag{5-21}$$

6 实验结果及分析

6.1 实验概述

在经过理论学习后，需要实践巩固以及发现新知识。根据本次实验的要求，主要需要从以下**四个方面**考虑对分类器分类性能的影响

- 两种不同特征 (BoW 以及 N-gram) 对分类器性能的影响
- 不同的学习率对分类器性能的影响
- 不同的参数学习方法 (BGD、SGD 以及 MBGD) 对分类器性能的影响
- 对数据集做 shuffle 与否对分类器性能的影响

由于在 Kaggle 的 SentimentAnalysis on Movie Reviews 竞赛的数据集中测试集是作为线上测试计分排名用的，故没有提供对应的标签，故在本实验中，以（训练集: 验证集 =4:1）对训练集进行划分，通过验证集来测试分类模型的性能。

在本节中，将通过**控制变量法 (Control Variables)** 对以上几个方面进行实验。本节中，每一小节对应一个实验，包括实验方法、实验结果以及实验分析三个方面进行总结。

为保证实验的真实性以及结果可被复现，本实验中所有需要随机的地方所使用的 $RandomState$ 均设为 2019。本实验所使用的平台及环境如下所示

```
操作系统为 MacOS Mojave
处理器为 Intel Core I5 3.1GHz
内存为 16G
未使用 GPU 加速
其中 python 版本为 3.7.3，numpy 版本为 1.16.2
```

6.2 不同特征对分类性能的影响

在上文 §4.1 以及 §4.2 中，介绍了两种文本向量化的方式，分别是 BoW 模型以及 N-gram 模型，在本实验中，分别以这两种方法对文本特征提取，用于文本分类。

考虑到特征向量维度过高需要的训练的时间过长的原因，本实验中采取了以下两个方案对维度进行了限制。

- N-gram 中的变量 N 取 2，使其变成二元模型 Bi-gram
- 对于 BoW 特征以及 Bi-gram 特征均做了文本统计。对于 BoW 特征而言，当且仅当词在语料中出现的频次大于 20 才放入词表；对于 Bi-gram 特征而言，当且仅当 Bi-gram 元组在语料中出现的频次大于 20 才计入元组

在经过上述处理后，进行了实验，实验组及对照组的实验参数如下表 6-1 所示

表 6-1 不同特征下实验组及对照组参数的设置

参数	实验组	对照组
损失函数	交叉熵	交叉熵
学习率	0.1	0.1
迭代次数	10 * n_Samples	10 * n_Samples
参数优化方法	SGD	SGD
特征提取方法	**BoW**	**Bi-gram**

根据以上实验参数，对 Softmax 模型进行了训练，得到的 loss 曲线以及 $Accuracy$ 和 $F_1\ Score$ 如下图 6-1 以及表 6-2 所示

图 6-1 不同特征下实验组及对照组的损失曲线对比

表 6-2 不同特征下实验组及对照组的分类器性能对比

参数	实验组	对照组
$Accuracy$	0.6296	0.5925
$F_1\ Score$	0.51	0.45

我对以上图 6-1 以及表 6-2 的实验结果较为意外，由于 Bi-gram 模型包含了语义顺序，故理应较 BoW 模型有更好的分类效果。但是实验结果没有反映出这样可能，我推测出现这样的情况主要是以下三方面的原因造成的:

1. 由于在提取特征时为了使训练速度更快以及抵抗噪声，本实验抽取的仅有当 Bi-gram 组合出现频次 ($N \geq 20$) 时才计入特征组合，并且 Bi-gram 较 BoW 模型更加稀疏，这可能导致了最终 Bi-gram 特征提取后的特征维度过低，使模型的分类性能下降

2. 对于这样的一个影评语料集，里面包含了句子以及由句子不断分拆出的词组，故我认为使用BoW 模型训练时，也可近似看成其对语料中顺序表达的特征有了部分应用，这缩小了 Bi-gram 模型对语义顺序方面的优势，使分类性能差别减小

3. 第一点中提到了使用低频特征过滤后，Bi-gram 特征的维度小于 BoW 的特征维度，观察图 6-1 的 Loss 曲线可发现本实验的迭代次数设置的偏多，故 Bi-gram 对于训练集的数据可能会过拟合，使其在验证集上的性能较差

根据以上的分析，我认为以后在做实验时，需要对特征的处理更加的谨慎，并且对于迭代次数的选择也应该更加的合理。

6.3 不同学习率对分类性能的影响

考虑到多分类器的分类性能以及损失曲线和梯度下降时往往与学习率超参的设置有关，学习率乘以损失函数的负梯度决定了参数学习的步长，故不同的学习率对于模型会有不同的表现。其中根据理论的探究发现对于大学习率以及小学习率的优缺点如下所示

- **学习率大的情况**

  学习率大的优点是梯度下降的速度快，故模型的训练时间较短损失曲线即会收敛。相应的学习率大也存在缺点，缺点是容易出现梯度爆炸的现象以及当学习率过大会使得模型的训练时损失曲线容易震荡，使最终无法收敛

- **学习率小的情况**

  学习率小的优点是模型的分类性能比学习率大要好，这是因为在梯度下降时在求解最优解时会更加精细，使最终的结果尽可能的达到最优的性能。但是缺点是下降的速度慢，且容易产生数据过拟合的现象

为了探究以上学习率不同造成的影响，故进行了本次试验，实验组以及对照组 1、2 的实验参数如下表 6-3 所示

表 6-3 不同学习率下实验组及对照组参数的设置

参数	实验组	对照组 1	对照组 2
损失函数	交叉熵	交叉熵	交叉熵
学习率	**0.01**	**0.05**	**0.1**
迭代次数	5 * n_Samples	5 * n_Samples	5 * n_Samples
参数优化方法	SGD	SGD	SGD
特征提取方法	BoW	BoW	BoW

根据以上实验参数，对 Softmax 模型进行了训练，得到的 loss 曲线以及 $Accuracy$ 和 $F_1\ Score$ 如下图 6-2 以及表 6-4 和表 6-5 所示 (为了使结果更加清晰，对于图 6-2 中的 Loss 做了下采用处理)

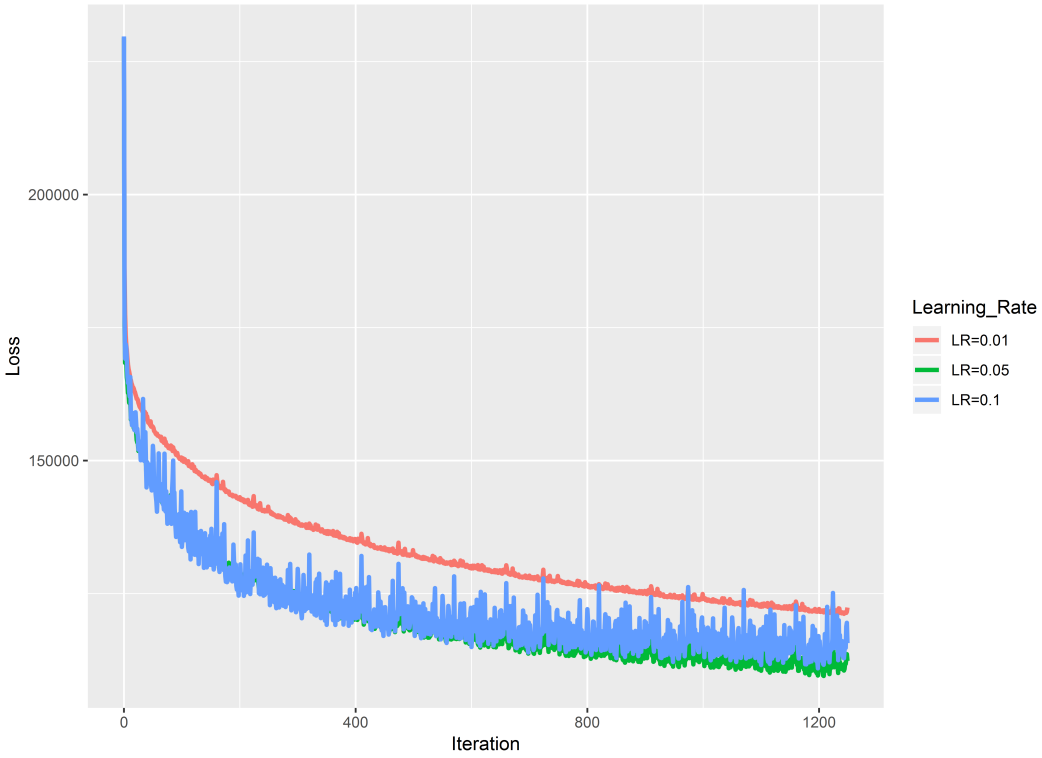

图 6-2 不同学习率下实验组及对照组的损失曲线对比

表 6-4 不同学习率下实验组及对照组的分类器性能对比 (训练集)

参数	实验组	对照组 1	对照组 2
$Accuracy$	0.6193	0.6447	0.6361
$F_1\ Score$	0.44	0.50	0.49

表 6-5 不同学习率下实验组及对照组的分类器性能对比 (验证集)

参数	实验组	对照组 1	对照组 2
$Accuracy$	0.5986	0.6155	0.6065
$F_1\ Score$	0.41	0.45	0.44

通过对上图 6-2、表 6-4 以及表 6-5 所示的 Loss 曲线、Accuracy 和 $F_1\ Score$ 进行分析，可得如下结论

- 通过图 6-2 所示 Loss 曲线的对比，可知当学习率设置的较大时，收敛速度的确更快，(由于途中 LR=0.1 的情况遮盖了部分 LR=0.05 的情况，通过对 Loss 数据进行分析，可知对于三种学习率而言，收敛的速度关系是 $V_{LR=0.1} > V_{LR=0.05} > V_{LR=0.01}$)
- 对于图 6-2 所示曲线，可发现 $LR = 0.1$ 的曲线呈震荡性下降，$LR = 0.05$ 次之，而 $LR = 0.01$ 损失曲线最为平滑
- 由表 6-4 及表 6-5 可知，在这三组中，分类器性能最好的是 $LR = 0.05$ 时，$LR = 0.1$ 时次之，性能最差的是 : $LR = 0.01$。这也印证了本小节所述的大学习率以及小学习率的优缺点

关系，$LR = 0.01$ 的精度更高，但是过拟合也越容易出现；并且另一种可能是在本实验中，$LR - 0.01$ 的 Loss 曲线并没有完全收敛，我认为这两个原因最有可能造成 $LR = 0.01$ 的分类性能最差。

6.4 不同优化算法对分类性能的影响

由上文 §5.3 可知，对于本模型的优化采用的是梯度下降法，其又包含三种不同的形式，分别为批量梯度下降法、随机梯度下降法以及小批量梯度下降法。故在本实验中，探究了这三种不同的优化算法在相同迭代次数下对分类器最终性能的影响，其中实验组及对照组的实验参数如下表 6-6 所示

表 6-6 不同梯度下降的优化方法下实验组及对照组参数的设置

参数	实验组	对照组 1	对照组 2
损失函数	交叉熵	交叉熵	交叉熵
学习率	0.1	0.1	0.1
迭代次数	2000	5 * n_Samples	6000
参数优化方法	**BGD**	**SGD**	**MBGD**
特征提取方法	BoW	BoW	BoW
Bathch 个数	/	/	5000

根据以上实验参数，对 Softmax 模型进行了训练，得到的 loss 曲线以及 $Accuracy$ 和 $F_1\ Score$ 如下图 6-3 以及表 6-7 与表 6-8 所示

图 6-3 不同梯度下降的优化算法下实验组及对照组的损失曲线对比

表 6-7 不同梯度下降的优化算法下实验组及对照组的分类器性能对比 (训练集)

参数	实验组	对照组 1	对照组 2
$Accuracy$	0.5187	0.6261	0.5433
$F_1\ Score$	0.44	0.53	0.46

表 6-8 不同梯度下降的优化算法下实验组及对照组的分类器性能对比 (验证集)

参数	实验组	对照组 1	对照组 2
$Accuracy$	0.5151	0.5363	0.5379
$F_1\ Score$	0.44	0.48	0.47

通过对图 6-3 以及表 6-7 及表 6-8 进行分析，可以得到以下结论

- 单从图 6-3 的 Loss 曲线变化上看，SGD 的 Loss 远小于 BGD 和 MBGD，这是因为 SGD 很容易随机找到合适优化方向，从而更快接近极值点
- 虽然 SGD 的 Loss 曲线要小于 BGD 和 MBGD，但是通过观察表 6-7 及表 6-8 可以发现 SGD 在测试上的分类性能 (Accuracy) 远低于训练集，故可推测虽然 SGD 的 Loss 小，但此时模型已经过拟合了
- MBGD 相较于 SGD 而言在测试集上有相似甚至更好的表现，说明它对数据集的分布学习的最好
- BGD 的学习时间最长，且由于训练轮数选择的较少，故其并未完全收敛，所以性能也较 SGD 和 MBGD 较差

6.5 数据集做 Shuffle 对分类性能的影响

在对训练数据进行训练前，往往需要先做一次 Shuffle 操作，用于打乱数据之间的联系，这样可以有效的避免过拟合的出现。故在本实验中，探究了做 Shuffle 和不做 Shuffle 的实验组即对照组之间的对比，其中实验组和对照组的实验参数如下表 6-9 所示

表 6-9 Shuffle 操作实验中实验组及对照组参数的设置

参数	实验组	对照组 1
损失函数	交叉熵	交叉熵
学习率	0.1	0.1
迭代次数	5 * n_Samples	5 * n_Samples
参数优化方法	SGD	SGD
特征提取方法	BoW	BoW
Shuffle 操作	**是**	**否**

根据以上实验参数，对 Softmax 模型进行了训练，得到的 loss 曲线以及 $Accuracy$ 和 $F_1\ Score$ 如下图 6-4 以及表 6-10 所示

图 6-4 Shuffle 操作实验中实验组及对照组的损失曲线对比

表 6-10 Shuffle 操作实验中实验组及对照组的分类器性能对比

参数	训练集		测试集	
	实验组	对照组	对照组	对照组
$Accuracy$	0.6261	0.6065	0.5363	0.5108
$F_1\ Score$	0.53	0.49	0.48	0.44

从图 6-4 以及表 6-10 可分析得到对于给定的数据集，做 Shuffle 和不做 Shuffle 有较大的性能差异。从误差曲线上来看，由于做了 Shuffle 的数据集更加均匀，使其经过梯度下降优化能更好的达到最优点，故做了 Shuffle 的实验组损失下降的更快，且最终总体损失较不做 Shuffle 的对照组小。

而从表 6-10 可看出，实验组由于做了 Shuffle 无论是在 $Accuracy$ 和 $F_1\ Score$ 指标上均有更好的性能。

7 总结与感悟

本任务使我对于自然语言处理这一领域有了一些的了解。本实验中，首先通过标准化、分词、筛除停用词三步完成了文本清洗。其次在特征工程这一步，本实验通过词袋模型以及 N 元模型对文本进行特征提取，并经过滤除低频词语及组合完成了文本的向量化。

在模型训练中，学习了 Logistic 回归的基础原理、损失函数，并且通过对 Logistic 回归进行推广使其成为适用于多分类任务的 Softmax 回归，并就三种梯度下降算法及参数学习的理论知识进行了理解，最后，本次实验采用准确率以及宏-F1 分数来衡量模型的分类性能。

在经过理论知识的学习后，通过实践弥补了许多在理论学习中没有注意到的细节。本实验采用 Python 和 Numpy 实现文本清洗、特征工程以及模型训练的功能。共包含了四组对照实验，分别探究了两种不同特征、不同的学习率、不同的参数学习方法以及数据集是否做 shuffle 对分类器性能的影响。

其中，在探究两种不同特征的试验中得到了与我预想不同的结果。通过分析，我认为问题出现的原因是在 Bi-gram 特征中的低频组合过滤阈值设置的过大，使特征维度过小，较多句子可能都为全零向量，造成使用 Bi-gram 特征的对照组性能弱于使用 BoW 特征的实验组；而在探究不同学习率对模型分类性能影响的实验中，我认识到了对于大学习率以及小学习率而言，都有各自的优缺点，所以在设定学习率参数的时候，应该更加的谨慎的选择，在模型的训练速度以及模型的训练精度之间找到一个平衡点；针对第探究不同梯度下降算法对分类性能影响的实验，从中我学到了 Loss 曲线的大小不能完全代表分类器的分类性能，SGD 虽然更容易达到最优点，但是其极有可能过拟合。而 BGD 的训练速度过慢也不适合使用，MBGD 结合了 SGD 以及 BGD 的优点，往往会使分类器拥有更好的性能；最后一个实验是对数据集是否做 Shuffle 对分类器性能影响的分析，发现不做 Shuffle 往往会使得模型因为训练集自身的分布造成对分类器性能的影响，做了 Shuffle 的模型训练后分类器的分类性能远好于不做 Shuffle 处理的分类器。

基于以上理论以及实践的学习，使我对于自然语言处理以及文本分类问题有了一定的了解，对于该领域我也有着较大的兴趣，十分感谢邱老师能够提供这样一个机会让我能够感受到自然语言处理的魅力！

8 参考文献

[1] Xipeng Qiu, Neural Networks and Deep Learning,[M]. 2019, 35-133.

https://nndl.github.io/nndl-book.pdf

[2] Speech and Language Processing (3rd ed. draft)

http://web.stanford.edu/ jurafsky/slp3/

[3] Wikipedia.Gradient descent

https://en.wikipedia.org/wiki/Gradient_descent

[4] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv: 1609.04747 (2016).

[5] Qi Meng, Wei Chen,Yue Wang."Convergence Analysis of Distributed Stochastic Gradient Descent with Shuffling."arXiv preprint arXiv:1709.10432 (2017).

https://arxiv.org/pdf/1709.10432.pdf

9 附录

在本实验中，所有的代码均由本人通过 python 以及 numpy 库进行编写，本实验中代码已经上传至我 Github 的仓库中，地址为`https://github.com/xlxwalex/Nlp_Learning_Report`

1. 文本清洗部分函数

文本清洗部分代码用于完成 §3 中所介绍的步骤，包括预处理、标准化、分词以及筛除停用词，其包含的函数如下所示

```
Function
  read_file
    完成数据集的读取
  pre_process
    完成缺失值判断以及一致性检验
  wash_normalize
    完成标准化，将所有字母变成小写
  wash_tokenization
    完成分词步骤，将标点现替换为空格之后以空格分割单词
  wash_stopwords
    完成停用词的筛除(1)
  text_washing
    作为本部分的主函数使用，输入为数据集的存放地址，输出为处理后的数据以及对应标签
```

> **Note**
>
> Note(1)：使用的英文停用词来自 CSDN，链接为
>
> *https://blog.csdn.net/oYeZhou/article/details/83059359?utm_source=blogxgwz9*

2. 特征工程部分函数

特征工程部分代码用于完成 §4 中所介绍的步骤，包括 Bow 模型以及 N-gram 模型，其包含的函数如下所示

```
Function
  create_vocab
    完成词汇表的生成
  Feature_BoW
    生成Bow特征矩阵
  bigram_count
    完成Bi-gram组合的生成
  bigram_feature
    用于Bi-gram特征向量的生成
  feature_main
    作为本部分的主函数使用，输入为处理后的数据集以及标签，输出为特征以及转化的标签
```

3. 模型训练部分函数

模型训练部分代码用于完成 §5 中所介绍的步骤，本部分通过构建了一个类 SoftmaxClassifier，用于对模型进行操作，本类包含的函数如下所示

```
Class SoftmaxClassifier
Function
 __init__
    模型的初始化，完成了模型参数的传递、权重及偏置参数的初始化、数据集与验证集的分割以及Batch生成器的初始化
 fit
    通过不同的参数优化算法对模型数据进行拟合，并得到Loss数据
 softmax_MBGD
    完成MBGD的参数学习过程
 softmax_BGD
    完成BGD的参数学习过程
 softmax_SGD
    完成SGD的参数学习过程
 predict
    根据训练得到的权重以及偏置矩阵预测得到数据的标签
 softmax
    计算得到Softmax函数的输出
 get_scores
    通过矩阵运算计算线性函数的输出
 cross_entropy
    计算交叉熵损失
 one_hot
    将输入数据的标签转化为one-hot向量
 get_accuracy
    计算得到模型在训练集以及验证集上的Accuracy
 shuffle_data
    对数据进行Shuffle操作
 train_test_split
    将数据拆根据所设定比例分为训练集与测试集
 batch_gen
    通过生成器生成MBGD中Batch的数据
```