任务一：基于机器学习的文本分类

1. 实验任务

使用 Numpy 实现基于 softmax regression 的文本分类，分析不同文本特征表示、损失函数、学习率对最终分类性能的影响，了解机器学习中 shuffle、batch 和 mini-batch 等基本概念。

2. 数据集

Rotten Tomatoes dataset 是 Kaggle 关于影评情感分类的数据集，其训练集和测试集都已经划分好，其中训练集为从 8544 个句子中提取的 156060 个短语及其情绪标签，测试集为从 3310 个句子中提取的 66291 个短语。测试集短语情情绪标签未知，训练集情绪标签有五种，分别为：

Label	Description
0	negative
1	somewhat negative
2	neutral
3	somewhat positive
4	positive

任务目标为使用 softmax regression 训练短语情绪标注模型，因此仅使用训练集中的 Phrase 和 Sentiment 的信息。由于数据集规模庞大，且本实验使用较为简单的文本特征生成方式，文本特征向量维度较高，生成词典和训练模型过程较慢，因此本实验在给定的训练数据集中随机抽取了 20000 条数据作为新的数据集，按照 3:1 的比例划分为训练集和验证集，验证集用于调试相关的超参数，测试集为原始测试集。本地无测试集标签，最终测试结果为 Kaggle 平台结果。

3. 模型实现

3.1 整体流程

完成该任务主要分为以下几个步骤：

1) 数据预处理：读入数据，根据 batch size 划分 mini-batch，标签转化为 one-hot 编码
2) 提取文本特征：为了节约模型计算开销，本实验将统计好的字典存储到文件中
3) 文本分类

3.2 数据预处理

1) Shuffle

Shuffle 是在训练模型前将数据集进行打乱的操作，可以提升模型健壮性，防止过拟合。原始数据在样本均衡的情况下可能是按照某种顺序进行排列的，如前半部分为某一类别的数据，后半部分为另一类别的数据。经过打乱之后，数据的排列就会拥有一定的随机性，顺序读取时下一样本为各类的可能性相同。

2) Mini-batch

样本数量特别大的时候，为了提高效率，可以把样本分成等量的子集，也就是mini-batch。实验中每个 mini-batch 中的样本数相同，训练集最后一个 mini-batch 的样本不足时使用首个 mini-batch 中的样本补全。

3.3 特征选择

这里主要比较 Bag-of-Word 和 N-gram 这两种简单的文本特征提取方法。这两种方法都是在句子维度提取特征，通过词频统计的方法将文本信息直接转化为句向量。两种方法比较如下：

特征表示	构建方法	特点
Bag-of-word	1. 从语料中建立词典，词语数为 m 2. 句子表示为 m 维向量，每一维都对应于词典中的一个词语，其取值为句中该词语的出现次数	不考虑语法或词语的顺序
N-gram	1. 从语料中提取 n 元组词典，词组数为 m 2. 句子表示为 m 维向量，每一维都对应于词典中的一个词组，其取值为句中该词组的出现次数	考虑局部词语顺序

3.4 模型原理

该任务使用最简单的线性分类模型。类别标签为$y \in \{0,1,2,3,4\}$有 5 个取值，因此使用 Softmax Regression，它是 Logistic Regression 在多分类问题上的推广。

模型输入：$X \in R^{n \times m}$，其中 n 为训练集样本数，m 为每个样本的特征维度，也就是词典的词语（词组）数

类别标签：样本X_i的类别标签为$y_i = [I(0 = c), I(1 = c), I(2 = c), I(3 = c), I(4 = c)]^T$

模型输出：样本X_i的模型输出为$\hat{y_i} = \mathrm{softmax}(WX_i^T)$，其中参数矩阵$W \in R^{5 \times m}$

使用交叉熵损失函数计算损失：

$$\mathrm{loss} = -\frac{1}{N}\sum_{n=1}^{n} y_n^T \log \hat{y}_n$$

采用梯度下降法更新模型参数：

$$W_{t+1} = W_t - \alpha \frac{\partial \mathrm{loss}}{\partial W} = W_t + \alpha\left(\frac{1}{N}\sum_{n=1}^{n}(y_n - \hat{y}_n)X_n\right)$$

4. 模型评估

4.1 特征选择对模型性能的影响

分别使用 Bag-of-word 和 N-gram 提取文本特征。使用 N-gram 提取文本特征时使用二元特征。Bag-of-Word 可以看成是 n=1 的 N-gram 模型。基准情况下 batch-size=128，learning rate=0.1，epoch=100，使用不同的特征选择方法，模型性能比较如下：

Feature Represent	Accuracy
Bag-of-Word	0.4440
N-gram(n=2)	0.4625

在基准情况下使用 N-gram 提取文本特征，文本分类准确率更高。

4.2 超参数对模型性能的影响

这里采用网格搜索的方法比较不同超参数的效果。

1) Learning Rate

Feature Represent	Acc(lr=0.05)	Acc(lr=0.1)	Acc(lr=0.5)
Bag-of-Word	0.4416	0.4440	**0.4526**
N-gram(n=2)	0.4518	0.4625	**0.4726**

从上表的结果中可以看出，使用 Bag-of-Word 和 N-gram 提取文本特征时，学习率为 0.5 时模型效果最好。

2) Epoch

将学习率设置为上表中性能最佳时的学习率，比较不同迭代次数的模型性能如下：

Feature Represent	Acc(epo=50)	Acc(epo=100)	Acc(epo=200)
Bag-of-Word	0.4468	**0.4526**	0.4468
N-gram(n=2)	0.4698	0.4726	**0.4766**

从上表结果可以看出，随着迭代次数增大，模型性能有所提升。但是迭代次数继续增大也可能出现过拟合的情况，因此不能仅依靠增加迭代次数提升模型性能。

4.3 测试集上的模型效果

使用以上超参数在原始测试集上完成测试，Kaggle 平台的测试结果如下：

Feature Represent	LR	Epoch	Accuracy
Bag-of-Word	0.5	100	0.4133
N-gram(n=2)	0.5	200	0.4467

测试集准确率低于训练集和验证集。根据测试集上的模型表现，使用 N-gram 提取文本特征，完成该文本分类任务，具有更好的模型效果。

5.实验小结

本实验主要实现了基于 softmax regression 的文本分类，并比较了 Bag-of-Word 和 N-gram 这两种不同文本特征表示方法。使用 N-gram 提取文本特征模型性能更好。

在文本特征表示方面，这两种方法都是基于统计学原理，在句子维度量化文本。它们在原理上较为简单，但是实现中随着数据集规模的增大，计算开销增大明显。在分类模型方面，Softmax Regression 是简单的线性多分类模型。Kaggle 平台上该数据集竞赛的排行榜中，使用预训练模型提取文本特征，或使用相关的深度学习模型可以取得更好的模型表现。这也是下一个任务需要完成的内容。