郭家琛：软件工程 2203 班，U202210726，负责神经网络模型调参优化和实验，相关部分的论文写作。

王珺戈：软件工程 2203 班，U202216408，负责神经网络模型构建，相关部分的论文写作。

肖文睿：软件工程 2203 班，U202210795，负责 Softmax 模型调参优化和实验，相关部分的论文写作。

朱懿淳：软件工程 2201 班，U202211219，负责 Softmax 模型构建，相关部分的论文写作。

数据集和模型放在 github 仓库中
github 仓库地址：https://github.com/auserlock/nlpProject.git

基于机器学习进行文本情感分析

郭家琛、王珺戈、肖文睿、朱懿淳

摘　要　*本论文提出了一种基于多层感知器（MLP）的文本情感分析方法。通过词袋模型、TF-IDF 和 n-gram 提取文本特征，并结合降维技术优化特征表示，MLP 模型被用于高效分类。实验结果表明，MLP 方法在情感分类任务中表现出色，显著提升了模型的准确性和稳定性。*

关键词　机器学习 文本情感分析 Softmax 回归 梯度下降 神经网络

1 引言

在当今的信息化社会中，随着社交媒体、新闻平台和电子商务网站的普及，生成了大量的文本数据。这些数据不仅包含着事实信息，还蕴含着人们的情感与观点。如何自动从这些海量的非结构化文本数据中提取出有用的情感信息，成为了自然语言处理（Natural Language Processing, NLP）中的一个重要研究课题。情感分析（Sentiment Analysis）作为 NLP 的核心任务之一，旨在识别和提取文本中的情感倾向，从而广泛应用于舆情监控、市场分析、产品评价等领域。

随着机器学习方法的快速发展，情感分析取得了显著的进展。传统的基于词典或规则的方法往往难以应对语言表达的复杂性，因此现代情感分析逐渐转向数据驱动的机器学习模型。这些模型通过在大规模标注数据集上进行训练，能够学习到文本中的情感特征。

本项目采用了两种主要的机器学习方法来处理文本情感分析任务。第一种方法是基本的 Softmax 回归与梯度下降优化。Softmax 回归适用于多分类任务，梯度下降作为常用的优化算法，用于不断调整模型参数以最小化损失函数。第二种方法是基于神经网络的嵌入层和全局平均池化（Global Average Pooling）的模型，通过词嵌入将文本映射到向量空间，结合全局特征提取和全连接层实现情感分类。这两种方法的结合使得我们能够在不同复杂度的场景下对文本情感进行准确分类。

2 相关工作

情感分析作为 NLP 领域的一个重要研究方向，已经发展了几十年，经历了从基于规则的简单模型到基于深度学习的复杂模型的演变过程。早期的情感分析工作多依赖于基于词典的方法，这种方法通过构建一个情感词汇表，将文本中的词语与情感类别进行匹配，进而判断文本的情感极性。例如，SentiWordNet 就是一个常用的情感词典，它为每个单词提供了情感评分。然而，这类方法通常只能处理显式的情感词汇，难以理解隐含在句子结构和上下文中的情感信息，面对语言的多样性和复杂性显得力不从心。

随着机器学习技术的发展，研究者开始探索通过数据驱动的方法进行情感分析。基于机器学习的情感分析方法通常将情感分类问题视为一个监督学习任务，通过对大规模标注数据集进行训练，学习文本中的情感模式。常用的机器学习算法包括支持向量机（SVM）、朴素贝叶斯（Naive Bayes）、逻辑回归（Logistic Regression）等。这些方法在情感分析任务中展现了较好的性能，尤其是当文本特征经过精心设计和提取时，分类器的效果可以显著提升。

近年来，随着深度学习的兴起，情感分析领域迎来了新的突破。卷积神经网络（CNN）和循环神经网络（RNN）等深度学习模型在处理图像、语音和文本数据时展现了强大的学习能力。对于情感分析任务，CNN 可以捕捉文本中的局部特征，而 LSTM 等 RNN 变体则能够处理文本中的序列信息，从而更好地理解上下文中的情感信息。这些模型在多种情感分类任务中取得了优异的表现。

与此同时，预训练语言模型的出现使得情感分析的准确性达到了新的高度。以 BERT 和 GPT 为代表的预训练模型，通过在大规模无标注语料库上进

行预训练，学习到了丰富的语言表示能力，并通过微调适应特定任务。这种方法相比于传统的从零开始训练的深度学习模型，具有更好的泛化能力和更高的分类准确性。例如，BERT 模型通过双向 Transformer 结构，能够在情感分析中同时考虑句子的前后文信息，使得情感分类更加准确。

此外，近年来一些研究还探索了多模态情感分析，即将文本与图像、语音等其他模态信息结合起来，以提高情感分类的精度。多模态情感分析利用不同模态之间的互补性，有望在复杂的应用场景中取得更好的效果。

总体来看，情感分析领域已经发展出一系列有效的方法和模型，从传统的基于词典的规则方法，到基于机器学习的模型，再到当下流行的深度学习与预训练模型，各种方法在不同的应用场景中展现出了不同的优势。

3 方法

3.1 提取数据特征

3.1.1 词袋模型

词袋模型（Bag of Words, BoW）是本研究中用于文本特征提取的基础方法。该模型通过统计词汇在文本中出现的频次，将文本转换为固定长度的特征向量，而忽略了词汇的顺序信息。

首先，我们从所有的文档中提取唯一词汇，构建词汇表。接着，对每个文档进行向量化处理，其中每个向量的维度对应于词汇表中的词汇，向量的值表示该词汇在文档中出现的频次。

具体而言，我们使用了 scikit-learn 库中的 CountVectorizer 工具，通过设置 min_df 参数来过滤掉在少量文档中出现的词汇，从而减少噪音。文本数据首先被转换为词频矩阵，其中每一行代表一个文本，每一列代表一个词汇表中的词，矩阵中的值表示该词在相应文本中出现的频次。为了确保验证集和测试集的转换与训练集一致，我们在转换过程中使用了与训练集相同的词汇表。

3.1.2 TF-IDF 特征提取

本研究采用了基于 TF-IDF（Term Frequency-Inverse Document Frequency，词频-逆文档频率）的特征提取方法，用于量化文本数据的词语重要性。TF-IDF 模型通过结合词频（TF）和逆文档频率（IDF）来衡量每个词语在文档中的重要性。文档中的常见词语（如"的"、"是"等）由于在多个文档中频繁出现，其 IDF 值较低，从而在特征向量中的权重也较低；而在少数文档中出现的词语则具有较高的 IDF 值。

其中，词频 TF 的计算公式如下：

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

式中，t表示词语（term），d表示文档（document），$f_{t,d}$表示词语在文档 d 中出现的次数，分母部分$\sum_{t' \in d} f_{t',d}$是文档d中所有词语出现的次数总和。

逆文档频率 IDF 的计算公式如下：

$$IDF(t,D) = \log\left(\frac{N}{1+n_t}\right)$$

式中，t表示词语，D表示文档合集（corpus），N是文档合集中总的文档数，n_t是包含词语t的文档数。

该公式计算逻辑是若词语t出现在较少的文档中，n_t会减少，故而 IDF 较高，反之，则 IDF 较低。

TF-IDF 是将 TF 和 IDF 相乘得到的值，公式如下：

$$TF-IDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

在本研究中，我们分别提取了基于单词的 TF-IDF 特征以及基于 n-gram（n 元词组）的 TF-IDF 特征。n-gram 特征提取考虑了多个连续词语的组合，如二元词组（bigram）和三元词组（trigram），以捕捉文本中的局部词语顺序信息。

3.1.2 特征组合

本研究采用了三种不同的文本特征提取方法：词袋模型（BoW）、基于单词的 TF-IDF 和基于 n-gram 的 TF-IDF，得到同一个数据集的三种不同的特征稀疏矩阵。为了充分利用这三种特征，本研究将它们组合成一个整体特征矩阵，以期望模型能够更好地捕捉文本中的信息。

我们使用水平堆叠的方式（hstack）将这三种特征组合起来，形成一个新的特征矩阵。该特征矩阵结合了词频信息、词语的重要性以及词语序列的局部信息，从而丰富了模型的输入特征空间。

hstack 的基本原理是将多个稀疏矩阵按列进行拼接，生成一个新的矩阵。假设我们有两个矩阵A和B，它们具有相同的行数m，列数分别为n_1和n_2，使用 hstack([A,B])操作后，生成新的矩阵 C 的尺寸位$m \times (n_1 + n_2)$，其中，矩阵C的前n_1 列由矩阵A组成，后n_2列由矩阵B构成。

3.1.4 降低维度

为了降低特征空间的维度，同时保留尽可能多的有用信息，本研究采用了截断奇异值分解（Truncated Singular Value Decomposition, Truncated SVD）方法。Truncated SVD 是一种广泛应用于稀疏矩阵和高维数据的降维技术，通过保留前 n_components 个奇异值及其对应的奇异向量来实现降维。

首先，对训练集特征矩阵X_{train}进行 Truncated SVD 降维，生成低维特征矩阵X_{train_svd}：

$$X_{train\_svd} = U_k \Sigma_k$$

其中，U_k是前K个左奇异向量组成的矩阵，Σ_k是对应的奇异值矩阵。

然后，利用在训练集上学习到的映射，将验证集和测试集的特征矩阵映射到相同的低维空间中，得到验证集和测试集的低维特征矩阵。

该方法不仅减少了特征维度，还有效地去除了特征之间的冗余信息，使得模型在低维特征空间中仍能保持较高的分类或回归性能。

3.2 Softmax 回归函数

Softmax 回归是一种常用于多分类问题的线性模型，通过学习样本的特征与类别之间的线性关系来进行分类。在文本分类任务中，Softmax 回归能够有效地将文本特征映射到多个类别之一。

Softmax 函数：对于一个输入样本x，Softmax 回归模型首先通过线性变换$z = \theta^T x$计算出每个类别的得分（logits）。这些得分再通过 Softmax 函数转换为概率分布，公式如下：

$$P(y = c_i|x) = \frac{\exp\left(\theta_i^T x\right)}{\sum_{j=1}^{C} \exp\left(\theta_j^T x\right)}$$

其中：θ_i是对应类别c_i的参数向量，C是类别的总数，exp表示指数函数。

损失函数：在训练过程中，模型通过最小化交叉熵损失函数来调整参数θ。交叉熵损失函数衡量了模型预测的概率分布与实际类别之间的差距，定义如下：

$$L(\theta) = -\log P(y|x;\theta) = -\log \frac{\exp\left(\theta_y^T x\right)}{\sum_{j=1}^{C} \exp\left(\theta_j^T x\right)}$$

梯度下降优化：为了最小化损失函数，使用梯度下降法来优化参数θ。梯度下降法通过以下公式更新参数

$$\theta_i = \theta_i - \eta \cdot \frac{\partial L(\theta)}{\partial \theta_i}$$

其中η是学习率，一般由手动指定，决定了每次更新的步长，学习率不同，模型收敛的效果也不同；$\frac{\partial L(\theta)}{\partial \theta_i}$是损失函数对参数$\theta_i$的偏导数。化简后可以得到

$$\frac{\partial L(\theta)}{\partial \theta_i} = \left(P(y = c_i|x) - 1(y = c_i)\right) \cdot x$$

这就是梯度下降法中用于更新参数θ_i的梯度计算公式公式，表示了模型预测概率与实际标签之间的差异如何影响每个参数的调整方向与幅度。通过不断迭代计算并更新参数，模型的损失函数将逐步减小，从而提高分类的准确性。在实际计算中，假设X是训练样本矩阵，形状为$N \times d$，W是权重矩阵，形状为$d \times C$，目标的独热编码矩阵 Y 形状为$N \times C$，模型的预测概率矩阵$\hat{Y}$形状为$N \times C$，那么偏导数的矩阵表示为

$$\frac{\partial L(\theta)}{\partial W} = X^T(\hat{Y} - Y)$$

在实际计算中我们通过选取样本计算梯度值来更新参数，而样本的选取策略不同所得到的结果也不同，主要有以下三种方法：

批量梯度下降（Batch Gradient Descent）方法：在每次迭代时，使用整个训练集计算梯度并更新参数。虽然这种方法能确保全局最优，但在大规模数据集上计算成本较高。

小批量梯度下降（Mini-batch Gradient Descent）：在每次迭代时，仅使用一个小批量（mini-batch，即训练集中的一部分）的数据计算梯度。这样能够兼顾计算效率与模型的收敛速度，通常也是实践中常用的方法。

随机梯度下降（Stochastic Gradient Descent, SGD）：每次更新参数时仅使用一个样本计算梯度。虽然计算速度快，但梯度的波动较大，容易引入噪声。

需要注意的是，当我们采用对比的方法来比较上述三种策略的效果时，需要统一变量来进行公平的比较。具体来说，随机梯度下降（SGD）的循环次数应最多，因为每次循环中仅计算一次梯度。假设训练集包含N个样本，那么在一循环中，批量梯度下降（Batch Gradient Descent）会计算N次梯度。小批量梯度下降（Mini-batch Gradient Descent）的计算次数次之，假设 mini-batch 的大小为m,那么在

一个循环中将计算N / m次梯度。SGD 计算次数最少，每个迭代仅计算 1 次梯度。从而应给定相同梯度计算次数，循环数相应变化，实现比较性能的目的。

3.3 神经网络模型方法

3.3.1 模型设置

本研究设计并实现了一个机器学习模型用于文本情感分析。在模型的初始化过程中，定义了输入数据的形状和分类任务的类别数，在本实验中，分别为：$input_{shape} = 1000$，$num_{class} = 5$。

此外，在模型中还进行了以下关键设置：

早停策略（EarlyStopping）：为了防止模型过拟合，当验证集损失（val_loss）在连续 3 个训练轮次内没有显著改善时，提前停止训练，并恢复到表现最佳的模型权重。

学习率调度器（ReduceLROnPlateau）：在训练过程中，如果验证集损失在连续 3 个训练轮次内没有改善，则将学习率减少 0.1 倍，以帮助模型更好地收敛。

这些配置确保了模型的稳定训练，并提高了最终模型的泛化能力。

3.3.2 模型构建

本文构建了一个多层感知器（Multi-Layer Perceptron，MLP）模型，用于文本情感分类任务。该模型采用了 Keras 的 Sequential API 构建，具体的网络结构如下：

(1) 输入层：输入数据形状与特征提取后的维度相匹配。

(2) 隐藏层：

第一层为包含 512 个神经元的全连接层，激活函数为 ReLU。

第二层为批量归一化（Batch Normalization）层，标准化每一批次的输入以提高模型训练的稳定性。

第三层为 Dropout 层，随机丢弃 50% 的神经元，用于防止过拟合。

随后，模型继续堆叠两组相似的结构：

包含 256 个和 128 个神经元的全连接层，每层之后均紧跟批量归一化和 Dropout 层。

(3) 输出层：最终的输出层为包含 num_classes 个神经元的全连接层，使用 softmax 激活函数输出每个类别的概率分布。这里的 num_classes 为 5，对应于文本情感分析中的五个类别。

该模型架构简单而有效，利用 ReLU 激活函数提取特征，批量归一化提高训练速度和稳定性，Dropout 层则有效防止了过拟合。最终使用 softmax 激活函数进行多类别分类。

感知器（Perceptron）：由 Frank Roseblatt 于 1957 年提出，是一种广泛使用的线性分类器。

感知器学习算法是一个经典的线性分类器的参数学习算法。其基本原理如下:

给定N个样本的训练集：$\{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$,其中$y^{(n)} \in \{+1, -1\}$，感知器学习的算法试图找到一组参数$w^*$，使得对于每个样本$(x^{(n)}, y^{(n)})$有

$$y^{(n)} {w^*}^T x^{(n)} > 0, \forall n \in \{1, \cdots, N\}.$$

感知器的学习算法是一种错误驱动的在线学习算法。先初始化一个权重向量$w \leftarrow 0$（通常是全零向量），然后每次分错一个样本(x, y)时，即$yw^T x < 0$，就用这个样本来更新权重：

$$w \leftarrow w + yx$$

根据感知器的学习策略，可以得知感知器的损失函数为：

$$\mathcal{L}(w; x, y) = \max(0, -yw^T x)$$

采用随机梯度下降，其每次更新的梯度为：

$$\frac{\partial \mathcal{L}(w; x, y)}{\partial w} = \begin{cases} 0 & if\ yw^T x > 0 \\ -yx & if\ yw^T x < 0 \end{cases}$$

算法 3.1 两类感知器的参数学习算法

输入：训练集$\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$，最大迭代次数$T$

初始化：$w_0 \leftarrow 0, k \leftarrow 0, t \leftarrow 0$;
repeat
  对训练集$\mathcal{D}$中的样本随机排序;
  FOR $n = 1 \cdots N$ DO
    选取一个样本$(x^{(n)}, y^{(n)})$;
    IF $w_k^T (y^{(n)}, x^{(n)}) \leq 0$ THEN
      $w_{k+1} \leftarrow w_k + y^{(n)} x^{(n)}$;
      $k \leftarrow k + 1$;
    END
    $t \leftarrow t + 1$
    IF $t = T$ THEN BREAK;
  END
UNTIL $t = T$;
输出：w_k

多层感知器中，我们引入一个构建在输入输出联合空间上的特征函数$\phi(x, y)$，将样本对(x, y)映射

到一个特征向量空间。

在联合特征空间中，我们可以建立一个广义的感知器模型：

$$\hat{y} = \arg\max w^T \phi(x, y)$$
$$y \in Gen(x)$$

其中w为权重向量，$Gen(x)$表示输入x所有的输出目标集合。

3.3.3 模型编译

在模型构建完成后，使用 Adam 优化器对模型进行了编译，并配置了适当的损失函数和评估指标，以确保模型在训练过程中能够有效学习并评估性能。具体设置如下：

优化器：选择 Adam 优化器（Adaptive Moment Estimation），其学习率（learning rate）设为 0.01。Adam 优化器结合了动量和自适应学习率调整的优点，能够在多种任务中表现出色，适用于处理高维稀疏数据。

Adam 优化器在每次参数更新时会考虑两个主要的统计量：

(1) 梯度的一阶矩估计（动量估计）：估计梯度的均值，用于加速收敛。
(2) 梯度的二阶矩估计（均方根估计）：估计梯度的方差，用于调整学习率。

通过引入动量，Adam 优化器能够加速收敛，并减少震荡。它通过结合动量和自适应学习率的优势，能够在复杂的优化问题中表现出色。

损失函数：使用稀疏分类交叉熵（Sparse Categorical Crossentropy）作为损失函数，该函数特别适合多类别分类问题。

评估指标：在训练过程中使用准确率（accuracy）作为主要评估指标，以衡量模型对验证集的预测性能。

通过上述配置，模型能够高效地进行参数调整，最终在测试数据上表现出较好的分类性能。

4 实验

4.1 Softmax 回归实验

4.1.1 实验目的

本实验旨在通过调整 Softmax 回归模型中的超参数（学习率 alpha、训练次数 times 和 mini-batch 大小 mini_size），优化模型在文本分类任务中的表现。通过不同的参数设置，比较模型在训练集和测试集上的准确率，寻找能够提高准确率的最优参数组合。

4.1.2 实验设置

数据集：使用 Bag of Words 和 N-gram 特征进行文本表示，分为训练集和测试集。

模型：Softmax 回归模型，用于分类任务。

超参数：

学习率 alpha：选择不同学习率进行实验，初步选择 $0.001, 0.01, 0.1, 1, 10, 100, 1000$，10000，100000。

训练次数 total_times：测试不同的训练次数，初步选择 $1000, 10000, 50000$。

mini-batch 大小 mini_size：测试不同的 mini-batch 大小，初步选择 $16, 32, 64$。

4.1.3 实验步骤

1. 固定 mini-batch 大小：选择一个固定的 mini-batch 大小，然后调整学习率和训练次数，记录在训练集和测试集上的准确率。

2. 固定学习率和训练次数：选择合适的学习率和训练次数，然后调整 mini-batch 大小，记录结果。
3. 绘制准确率曲线：绘制学习率、训练次数与准确率的关系曲线，分析不同超参数设置对模型性能的影响。

4.2 神经网络模型实验

4.2.1 实验目的

本实验旨在通过调整神经网络模型方法中的特征工程（降维的维度）、超参数（初始学习率 learning_rate、batch-size 大小）和优化技术（神经网络中 Dropout 层的正则化参数），优化模型在文本分类任务中的表现。通过不同的参数设置，比较模型性能以及在测试集中的表现，寻找最优模型。

4.2.2 实验初始设置

为了便于比较不同参数组合的效果，在实验开始时设置一组初始参数，作为基准模型。固定降维到 1000 维、学习率为 0.01，batch-size 为 128，Dropout 层参数为 0.5。

4.2.3 实验过程

1. 固定其它参数，调整降维维度：选择 500 和 2000 维

2. 固定其它参数，调整学习率：选择 0.1 和 0.001

3.固定其它参数，调整 batch-size：选择 64 和 256

4.固定其它参数，调整 Dropout 层参数：选择

0.3 和 0.7

4.2.4 实验结果分析依据

绘制每组的训练集和测试集的损失变化和准确率变化，同时以提交到 kaggle 上得到的分数作为模型在测试集上的准确度。根据绘制图和测试集准确度来分析不同超参数设置对模型性能的影响。

5 结果

5.1 Softmax 回归实验

5.1.1 特征的影响

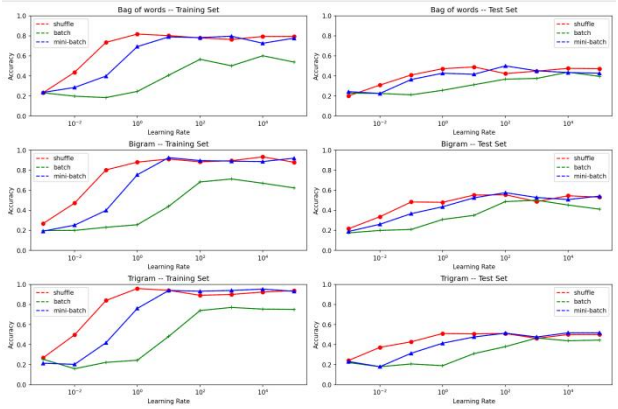
N 元特征优于词袋模型，这是因为 N 元特征考虑了词序。3-gram 特征优于 2-gram 特征，这是因为 3-gram 特征捕捉了更多的上下文信息。

5.1.2 学习率的影响

在较小的学习率（如 0.001）下，模型的收敛速度较慢，训练时间较长，准确率提升缓慢。

在中等学习率（如 1）下，模型表现较为平衡，能够在合理时间内收敛，并且在训练集和测试集上的准确率表现都较好。

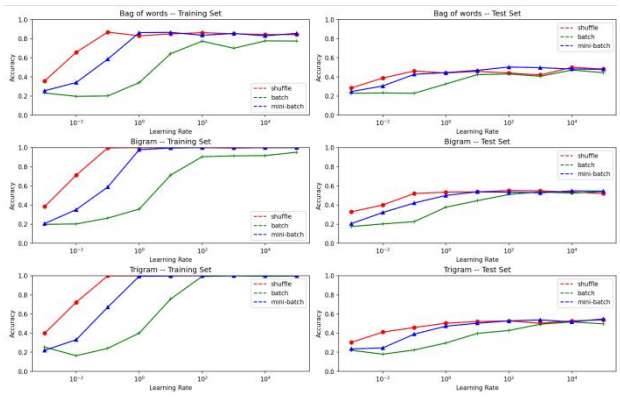
在较大的学习率（如 1000）下，模型容易出现不稳定的情况，准确率波动较大，甚至可能无法收敛。



训练次数为 10000，mini-size 为 16 时的实验结果

5.1.3 训练次数的影响

随着训练次数的增加，模型的训练集准确率不断提高，但在达到一定次数后（如 50000 次），测试集的准确率开始下降，出现过拟合现象。



训练次数为 50000，mini-size 为 16 时的实验结果。可见训练集出现了过拟合现象

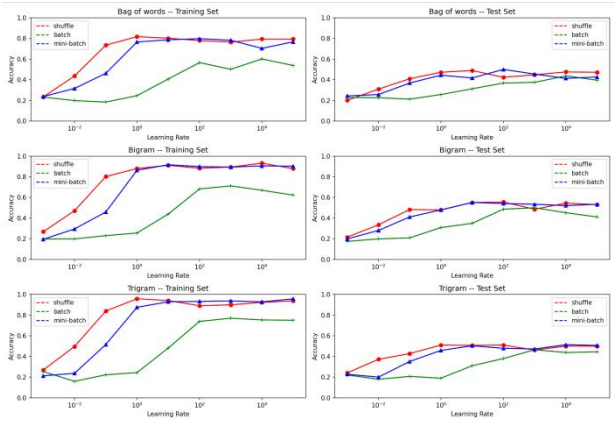
适当增加训练次数有助于提高模型性能，但需要结合早停策略避免过拟合。

5.1.4 不同梯度下降的影响

我们可以看到，除了学习率比较小不收敛的时候，小批量和随机梯度下降的表现几乎是相当的，而整体梯度下降表现最差。

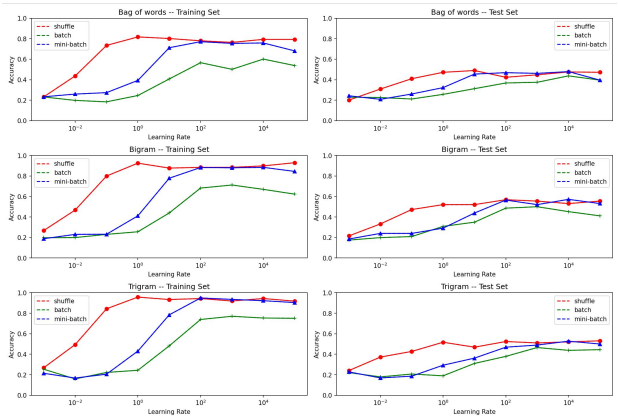
5.1.5 mini_size 的影响

较小的 mini-batch（如 16）增加了模型的随机性，模型的准确率波动较大，但有时能够跳出局部最优解。



mini_size 为 16，训练次数为 10000 时的实验结果

较大的 mini-batch（如 128）则使得模型更新更加稳定，训练速度更快，但可能失去随机梯度下降的优势，容易陷入局部最优解。

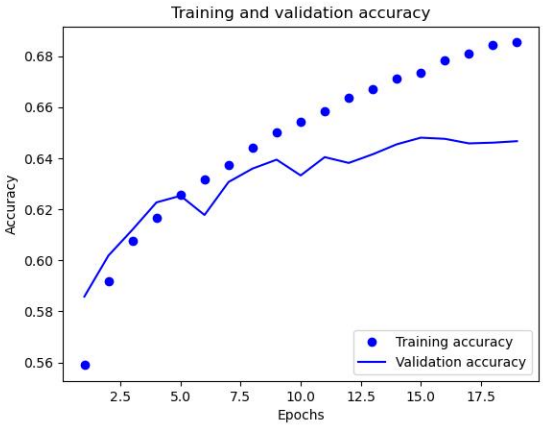


mini_size 为 128，训练次数为 10000 时的实验结果

5.2 神经网络模型实验

5.2.1 基准模型性能

基准模型的参数：降维到 1000 维、学习率为 0.01，batch-size 为 128，Dropout 层参数为 0.5。



基准模型在训练集和验证集上的准确度

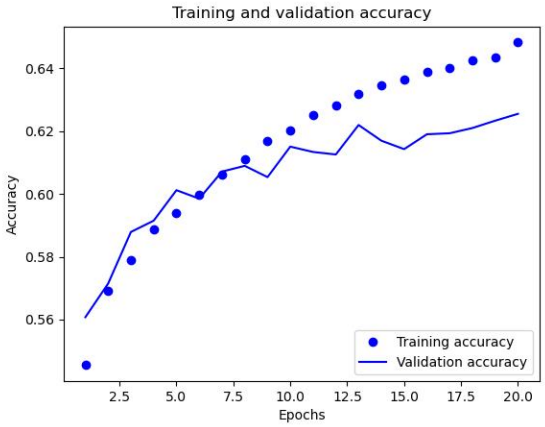
上图展示了基准模型在训练集和基准模型在训练集和验证集上的效果，在验证集上的 accuracy 最终收敛到 0.64 左右。同时，模型在测试集上的得分（kaggle 得分）是 0.59971。

5.2.2 降维维度的影响

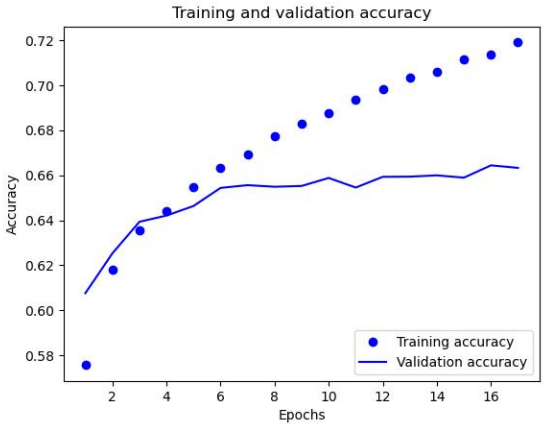
基准模型的降维维度是 1000 维，实验中设置了 500 维和 2000 维两个实验组。

通过实验可以清楚地看到，在训练集和验证集上，降维到 2000 维的效果优于基准模型，而降维到 500 维的效果不如基准模型。同时在测试集上的得分，降维到 2000 维是 0.61408，降维到 500 维是 0.58129。

当降维到更低维度时，虽然可以简化特征向量，能有效提高模型运行的速度，但同时可能会丢失一些关键信息，导致模型的精准度下降。



降维到 500 维的模型在训练集和验证集上的准确度



降维到 2000 维的模型在训练集和验证集上的准确度

5.2.3 学习率的影响

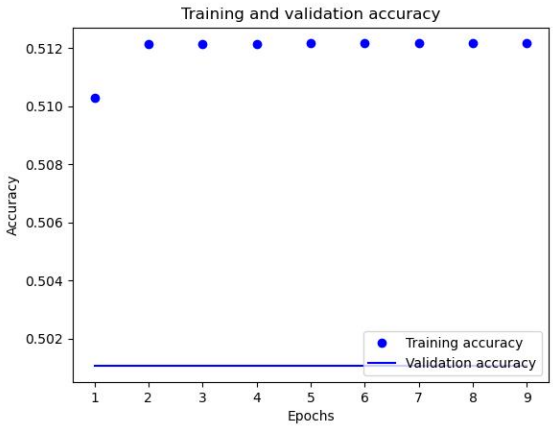
基准模型的学习率 0.01，实验中设置了 0.1 和 0.001 两个实验组。

当学习率为 0.1 时，出现了明显的过拟合现象。在训练集上经过训练，模型的准确度能提升一些，但整体也明显低于基准模型，而在测试集上准确度完全没有提升。同时在测试集上的得分仅为 0.51789，大幅度低于基准模型。

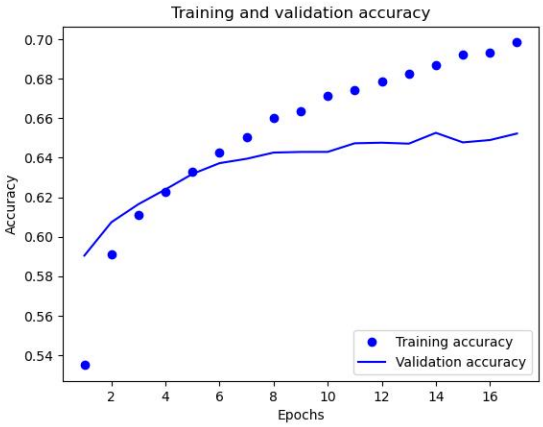
当学习率为 0.001 时，模型在训练集和测试集上的准确度和基准模型几乎一致，但是准确度提升过程更加平稳。同时，模型在测试集上的得分是 0.60263，略高于基准模型。

学习率控制每次梯度更新的步长。在训练过程中，模型通过反向传播计算梯度并更新权重，学习率决定了每次更新权重的幅度。较大的学习率会使权重更新幅度较大，可能导致模型快速跳过最优点，无法很好地收敛，甚至可能导致损失函数在训练中波动或发散。较小的学习率则使权重更新更

小，模型能够在损失曲面上更加平滑地前进，逐渐逼近全局最优或局部最优点。



学习率为 0.1 的模型在训练集和验证集上的准确度



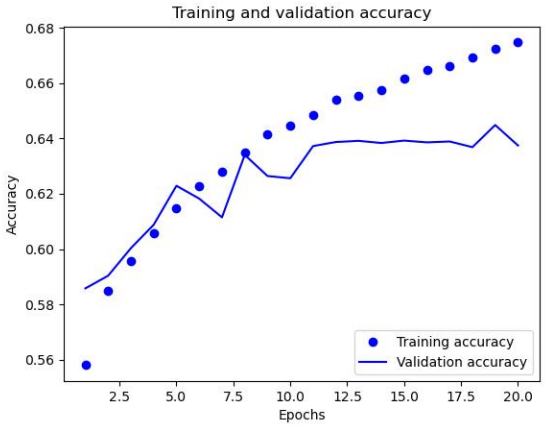
学习率为 0.001 的模型在训练集和验证集上的准确度

5.2.4 batch-size 的影响

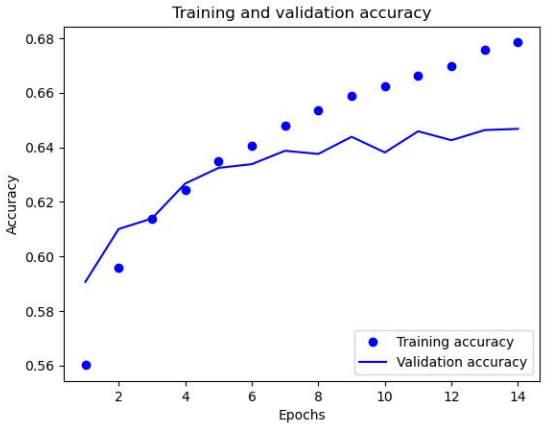
基准模型的 batch-size 设置为 128，实验中设置了 64 和 256 两个实验组。

通过实验可以看到，batch-size 为 64 和 256 的模型在训练集和测试集上的准确度和基准模型几乎一致。同时，batch-size 为 64 和 256 的模型在测试集上的得分分别为 0.59636 和 0.59961，也和基准模型几乎一致。

批量大小的选择虽然会影响训练过程中的更新频率和内存使用，但如果模型已经很好地收敛，且没有极端变化（如非常小或非常大的批量），那么不同的批量大小对最终效果的影响可能较小。批量大小的变化没有显著影响模型性能，说明模型和优化器的配置已经比较鲁棒，能够在这些不同设置下都取得类似的结果。



Batch-size 为 64 的模型在训练集和验证集上的准确度



Batch-size 为 256 的模型在训练集和验证集上的准确度

5.2.5 Dropout 层参数的影响

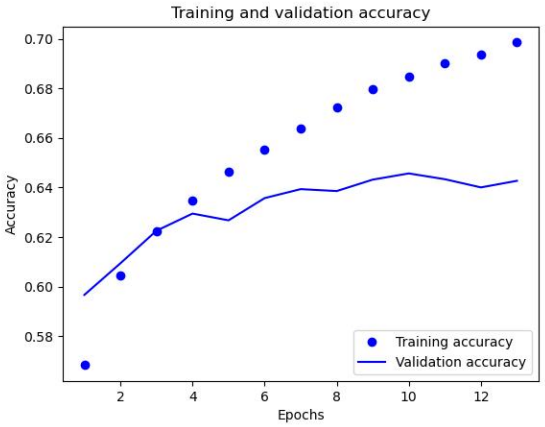
基准模型的 Dropout 层的正则化参数为 0.5，实验中设置 0.3 和 0.7 两个实验组。

由实验可知，当 Dropout 层参数为 0.3 时，训练集和验证集上的准确度差距较大，但是最终收敛的准确度和基准模型几乎一致。当 Dropout 层参数为 0.7 时，训练集和验证集的准确度差距很小，十分同步，最终收敛的准确度和基准模型几乎一致。

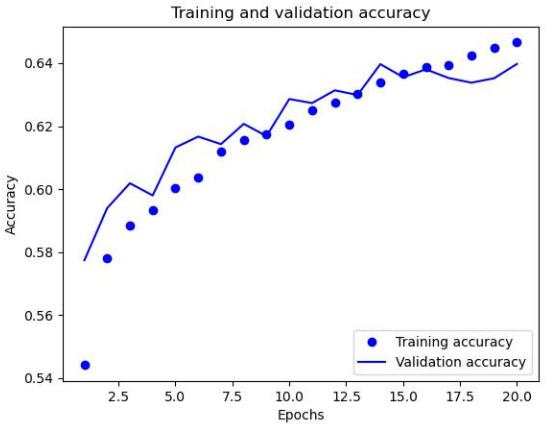
Dropout 是一种正则化技术，通过在训练过程中随机"丢弃"一定比例的神经元，使模型避免过拟合。这一随机丢弃过程让模型在训练过程中不能过度依赖某些特定神经元，迫使其学习更鲁棒的特征。较低的 Dropout 比例（如 0.3）意味着模型的大部分神经元仍然会参与训练。这种情况下，模型可以更快地学习训练数据的模式，因此训练集的准确度较高。然而，由于 Dropout 的正则化效果较弱，模型可能更容易过拟合训练数据。表现为训练集的

准确度显著高于验证集的准确度，这表明模型在验证集上的泛化能力较弱。较高的 Dropout 比例（如 0.7）意味着模型在每次训练时有更多的神经元被丢弃。这使得每次训练更新时，模型实际上在训练一个不同的子网络。这种更强的正则化使模型在训练过程中更难过拟合训练集，迫使它学习到更加通用的特征。因此，训练集和验证集的准确度差距较小，模型在训练和验证集上的表现更加同步。

尽管不同的 Dropout 设置对训练过程的表现有影响，但最终收敛的准确度接近基准模型，这表明模型的基本架构和超参数已经能够很好地捕捉数据集中的模式。



Dropout 层参数为 0.3 的模型在训练集和验证集上的准确度



Dropout 层参数为 0.7 的模型在训练集和验证集上的准确度

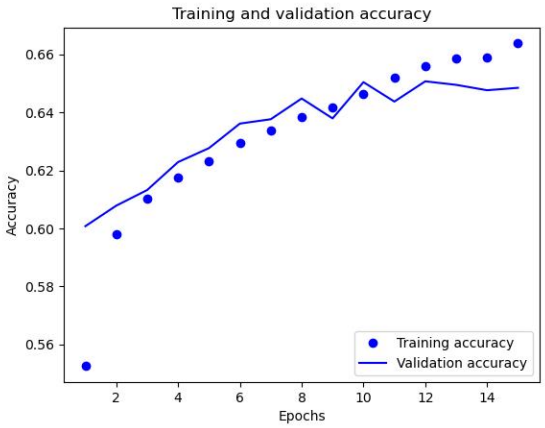
6 总结

6.1 Softmax 回归实验

最优参数组合：通过实验发现，使用 N 元特征以及小批量或随机梯度下降，并且使用中等学习率（如 1）、适中的训练次数（如 10000 次）和较大的 mini-batch 大小（如 64）能够为模型提供较好的性能表现，兼顾准确率与训练时间。在最优参数组合下，Softmax 回归模型在训练集和测试集上的准确率均有显著提升，测试集准确率达到较高水平。

6.2 神经网络模型实验

通过四组实验，我们发现降维至 2000 维、学习率为 0.001、`batch-size` 为 64 或 128 或 256、Dropout 层参数为 0.7 时为模型的参数最优组合。



最优参数组合的模型在训练集和验证集上的准确度

最优参数组合模型在验证集上的准确度略高于基准模型，在训练集和验证集的同步性明显高于基准模型，说明模型的鲁棒性更强。模型在测试集上的得分为 0.61075，明显高于基准模型。

参 考 文 献

- [1] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, vol. 10, 2002, pp. 79-86.
- [2] M. Hu and B. Liu, "Mining and summarizing customer reviews," Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 168-177.
- [3] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for sentence summarization," Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 379-389.
- [4] Socher et al., "Recursive deep models for semantic compositionality over a sentiment treebank," Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1631-1642.
- [5] Y. Kim, "Convolutional neural networks for sentence classification," Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1746-1751.

[6]邱锡鹏. 《神经网络与深度学习》. 机械工业出版社, 2020