PA2 情感分析实验报告

代码链接：https://cloud.tsinghua.edu.cn/d/de8a5c41ed3546c5ac10/

一、模型介绍

1. MLP

（1）结构图

输入层
↓
隐藏层(全连接)+激活函数（ReLU）
↓
输出层（全连接）

（2）流程分析

- **输入层**：接收经过预处理后的向量，在本项目中为50维。
- **隐藏层**：负责提取输入数据的非线性特征。
- **输出层**：用于输出两种类别的预测概率。

2. CNN

（1）结构图

输入层（嵌入层）
↓
卷积层1 + 激活函数（ReLU）
↓
池化层1（MaxPooling）
↓
卷积层2 + 激活函数（ReLU）
↓
池化层2（MaxPooling）
↓
全连接层
↓

输出层 (Softmax)

（2）流程分析

- **输入层**：通常是词嵌入层，将文本中的每个单词转换为固定维度的向量。
- **卷积层**：使用多个卷积核提取局部特征，每个卷积核生成不同的特征映射。
- **池化层**：降低特征维度的同时保持重要信息。
- **全连接层**：卷积和池化后的特征被展平，通过全连接层进行分类。
- **输出层**：输出每个类别的概率。

3. RNN（LSTM,GRU)

（1）结构图

输入层
↓
循环隐藏层
↓
输出层 (全连接 + Softmax)

（2）流程分析

- **输入层**：接收序列数据的每个时间步的输入。
- **循环隐藏层**：每个时间步的输入和前一时间步的隐藏状态一起被处理，能捕捉时间序列数据的动态变化。
- **输出层**：在序列的最后输出，用于分类。
- 可以改变 RNN 模型网络的类型来得到不同的结果。（ LSTM, GRU ）
- 可以增加RNN层数，使用双向模型等。

二、实验结果

- 以下结果统一参数为batch_size=64，learning_rate=0.001,sequene_length=64
- MLP：hidden_dim=100
- CNN：numfilters=100,filter_size=[2,3,4]
- RNN：hidden_dim=100，num_layers=1，bidirectional=false。

模型	准确率(%)	F-score
MLP	80.22	0.8043
CNN	86.45	0.8634
LSTM	84.55	0.8455
GRU	86.18	0.8618

三、 参数效果

（1）MLP模型

改变的参数	准确率(%)	F-score
默认初始值	80.22	0.8043
hd=1	76.96	0.7684
hd=10	79.04	0.7853
hd=50	80.49	0.7987
hd=150	79.95	0.7967
hd=200	78.32	0.7605
lr=0.1	52.03	0.6590
lr=0.01	79.13	0.8010
lr=0.0001	77.51	0.7662

（2） CNN模型

改变的参数	准确率(%)	F-score
默认初始值	86.45	0.8634
nf=1	76.96	0.7632
nf=10	84.82	0.8427
nf=50	82.66	0.8270
nf=150	84.82	0.8444
hf=200	85.09	0.8468
lr=0.1	49.32	0.1000
lr=0.01	81.84	0.8223
lr=0.0001	84.28	0.8380
fz=[3,4,5]	84.82	0.8462
fz=[2,3,4,5]	84.82	0.8453

（3）LSTM模型

改变的参数	准确率(%)	F-score
默认初始值	84.55	0.8455
hd=1	79.13	0.7855
hd=10	84.01	0.8401
hd=50	84.28	0.8457
hd=150	81.30	0.8034
hd=200	84.28	0.8389
lr=0.1	76.96	0.7733
lr=0.01	81.84	0.8091
lr=0.0001	83.47	0.8291
nl=2	83.74	0.8352
bi=True	82.11	0.8103

（4）GRU模型

改变的参数	准确率(%)	F-score
默认初始值	86.18	0.8618
hd=1	78.05	0.7840
hd=10	84.55	0.8430
hd=50	84.01	0.8418
hd=150	84.55	0.8455
hd=200	84.55	0.8367
lr=0.1	72.09	0.7082
lr=0.01	83.74	0.8246
lr=0.0001	84.82	0.8436
nl=2	84.82	0.8503
bi=True	83.20	0.8333

隐藏层维度或卷积核数量对模型影响：

1. **增加维度**：
    - **优点**：增加隐藏层的神经元数量可以提升模型的表达能力，使其能够捕捉更复杂的模式和关系。对于复杂的数据集或任务，较大的模型通常能实现更好的性能。
    - **缺点**：更多的神经元会增加模型的参数数量，这可能导致模型过拟合，尤其是在数据量较少的情况下。此外，训练时间和计算成本也会显著增加。
2. **减少维度**：
    - **优点**：较小的模型具有较少的参数，这有助于防止过拟合，并减少训练的时间和资源消耗。
    - **缺点**：如果隐藏层的维度设置得太小，可能会导致模型容量不足，无法捕捉数据中的复杂性和细微差别，表现为欠拟合。

学习率对模型的影响：

1. **高学习率**：

- **优点**：较高的学习率可以加快训练过程，快速达到较低的损失。
- **缺点**：过高的学习率可能导致训练过程不稳定，甚至使得损失函数发散，不收敛。在训练过程中可能会出现损失突然增大的情况。

2. **低学习率**：
   - **优点**：较低的学习率可以提供更稳定的训练过程，避免过大的权重更新造成的不稳定。
   - **缺点**：学习率过低会导致训练过程缓慢，模型需要更多的迭代次数才能收敛，这在计算资源有限的情况下可能是不可行的。

四、MLP与CNN，RNN模型的效果差异

- 根据以上结果分析，CNN 模型的效果最好，MLP模型的效果最差。
- RNN中GRU效果接近于CNN，LSTM效果稍差些，但也比MLP好许多。

五、问题思考

1） 实验训练什么时候停止是最合适的？简要陈述你的实现方式，并试分析固定迭代次数与通过验证集调整等方法的优缺点。

设置一个**最大固定迭代数**，无论模型的实际性能如何，训练过程会在达到这个轮数后停止。

在训练中**通过验证集调整（Early Stopping）**，在每一个epoch结束时，使用验证集数据来评估模型性能。如果模型在验证集上的性能开始连续下降或不再提升超过可接受次数后，停止训练。

对于自己实现的具体情况，设置最大迭代数为100，在每一个epoch训练结束后，使用验证集得到lo ss，准确率和F-score，如果三者连续10次相比之前最好的数据都没有性能上的改变，就停止训练。

固定迭代次数：

优点：

- **可预测性**：训练的持续时间是确定的，有助于资源规划。
- **简单性**：易于实现，无需复杂的停止条件。

缺点：

- **不灵活**：可能无法充分利用数据学习模式，或者可能过度训练模型。
- **不适应性**：如果数据变化或者模型结构变化，固定的迭代次数可能不再适用。

通过验证集调整

优点：

- **防止过拟合**：通过监控验证集的性能，可以避免模型在训练集上过度拟合。
- **自适应性**：与模型和数据集的具体特性相适应，能够自动调整训练的持续时间。

缺点：

- **潜在的停止过早**：如果验证集不够代表性或选择不当，可能会过早停止训练，导致模型未充分学习。
- **需调整参数**：可能需要调整early stopping的参数，比如“耐心”（即允许多少个epoch没有提升）。

2） 实验参数的初始化是怎么做的？不同的方法适合哪些地方？（现有的初始化方法为零均值初始化，高斯分布初始化，正交初始化等）

零均值初始化（Zero Initialization）

- 方法：所有参数设置为0。
- 适用性：通常不推荐使用零均值初始化，因为这会导致神经网络中所有神经元做出相同的更新，从而无法破坏称性。

高斯分布初始化（Gaussian Initialization）

- 方法：参数根据高斯分布（正态分布）随机初始化，通常有0均值，方差根据网络的特定层来选择
- 适用性：较大的网络或需要更细粒度初始化控制的情况，但要注意不要使用太大的方差，以避免梯度消失或爆炸。

正交初始化（Orthogonal Initialization）

- 方法：参数初始化为正交矩阵。
- 适用性：尤其适合循环神经网络（RNN），可以帮助缓解梯度消失的问题。

3） 过拟合是深度学习常见的问题，有什么方法可以方式训练过程陷入过拟合。

1.数据增强（Data Augmentation）

增加数据多样性，如旋转、翻转、缩放、剪裁图像，或者在文本数据中使用同义词替换、句子重排等方法。

2. 早停（Early Stopping）

在训练过程中监控验证集的性能，一旦性能开始下降，就停止训练，以防模型过度拟合训练数据。

3. 正则化（Regularization）

向模型中添加正则化项，如L1或L2正则化，可以惩罚大的权重值，限制模型的复杂度。

4. Dropout

在训练过程中随机“丢弃”网络中的一些神经元，可以减少神经元间复杂的共适应关系。

5. 减少模型复杂度

简化模型的结构，例如减少层数或神经元数量，以匹配数据的真实复杂度。

6. 学习率衰减（Learning Rate Decay）

随着训练的进行，逐渐减小学习率，使模型更稳定地收敛。

4） 试分析CNN，RNN，全连接神经网络（MLP）三者的优缺点。

1. MLP（全连接神经网络）

- **优点**
  - **简单易懂**：MLP是最基本的神经网络结构，仅由全连接层组成，易于实现和理解。
  - **通用性**：适用于广泛的简单或初级问题，尤其是那些特征间关系较为直接的任务。
- **缺点**
  - **缺乏空间和时间模式捕捉能力**：MLP不能有效处理具有空间和时间关系的数据，如图像或序列数据，因为它不考虑输入数据的顺序或空间结构。
  - **参数多，容易过拟合**：全连接层的参数量通常较大，容易导致过拟合，尤其是在数据量较少的情况下。

2. CNN（卷积神经网络）

- **优点**
  - **空间特征提取能力强**：CNN擅长处理图像等空间数据，通过卷积层可以有效捕捉局部特征和模式，例如在图像中识别边缘、形状等。
  - **参数共享减少过拟合**：通过参数共享和池化操作，CNN可以在保持较低参数量的同时保持较好的性能。
- **缺点**

- **对非空间数据处理有限**：虽然CNN也可以用于处理文本和其他序列数据（通过一维卷积），但它本质上并不擅长捕捉长期依赖或复杂的时间序列关系。
- **计算成本**：卷积操作相比于全连接层计算成本高，尤其是在处理大型图像时。

3. RNN（循环神经网络）

- **优点**
  - **序列数据处理能力**：RNN设计用来处理序列数据，如文本、时间序列等，能够通过隐藏状态捕捉时间上的动态变化和长距离依赖。
  - **灵活的输入输出结构**：RNN可以处理不同长度的输入序列，输出也可以是序列或单个输出，适应不同的任务需求。
- **缺点**
  - **梯度消失和爆炸**：RNN在训练长序列时容易遇到梯度消失和爆炸的问题，尽管LSTM和GRU等变体能够在一定程度上解决这一问题。
  - **顺序计算需求**：由于RNN的递归性质，其时间步的计算通常依赖于前一个状态，这限制了并行处理的可能，增加了计算时间。

六、心得体会

- 在本次实验中，我大致对神经网络有了基本上的了解，学会使用深度学习框架 Pytorch 的一些基本操作，收获许多。
- 同时，我还学会了如何更好得设计python程序，通过更加友好的输入与输出，让其更有可读性。