|  |  |
| --- | --- |
| 成绩 |  |

装

订

线

网络空间安全与计算机学院实验报告

实验课程名称**：神经网络与深度学习实验**

实验项目名称： SRN的记忆能力--梯度爆炸

学生姓名： 刘新媛

专 业： 人工智能

学 号： 20221205037

实验地点： C1-320

实验时间： 2024.12.04第7- 8节

指导教师： 魏勇刚

预习报告部分

1. **实验目的：**
2. 实现数字求和任务，测试简单循环网络的记忆能力。
3. 复现简单循环网络中的梯度爆炸问题，然后尝试使用梯度截断的方式进行解决。
4. **实验原理：**

* **实验一：测试SRN记忆能力**

RNN的一种简单实现是SRN，SRN在时刻t的更新方式如公式（1）所示。

（1）

为隐状态向量，为状态-状态权重矩阵，为状态-输入权重矩阵，为偏置向量。

单循环网络在参数学习时存在长程依赖问题，很难建模长时间间隔（Long Range）的状态之间的依赖关系。为了测试简单循环网络的记忆能力，本节构建一个数字求和任务进行实验，验证其长程依赖问题。

* **实验二：梯度爆炸**

造成简单循环网络较难建模**长程依赖问题**的原因有两个：梯度爆炸和梯度消失。一般来讲，循环网络的梯度爆炸问题比较容易解决，一般通过**权重衰减或梯度截断**可以较好地来避免；本节将首先进行复现简单循环网络中的梯度爆炸问题，然后尝试使用梯度截断的方式进行解决。

对于梯度消失问题，更加有效的方式是改变模型，比如通过长短期记忆网络LSTM来进行缓解。

1. **实验内容：**

* **实验一：测试SRN记忆能力 [点击查看作业](https://blog.csdn.net/qq_38975453/article/details/126521361)**
* **实验二：梯度爆炸 [点击查看作业](https://blog.csdn.net/qq_38975453/article/details/126800066)**

1. **实验所用设备：**pycharm、pytorch

实验报告部分

1. **实验步骤：**

* **实验一：测试SRN记忆能力**

## **数据集构建与划分**

①数据集构建：构建不同长度的数字预测数据集DigitSum，当生成某个指定长度的数据集时，会同时生成训练集、验证集和测试集。

②加载数据并划分：调用load\_data函数，传入data\_path，将返回的训练集、验证集和测试集数据分别赋值给对应的变量train\_examples、dev\_examples和test\_examples。

③构造Dataset类：构造DigitSum数据集的Dataset类，函数\_\_getitem\_\_负责根据索引读取数据，并将数据转换为张量。

## **模型构建**

使用SRN模型进行数字加和任务的模型结构如图1所示。

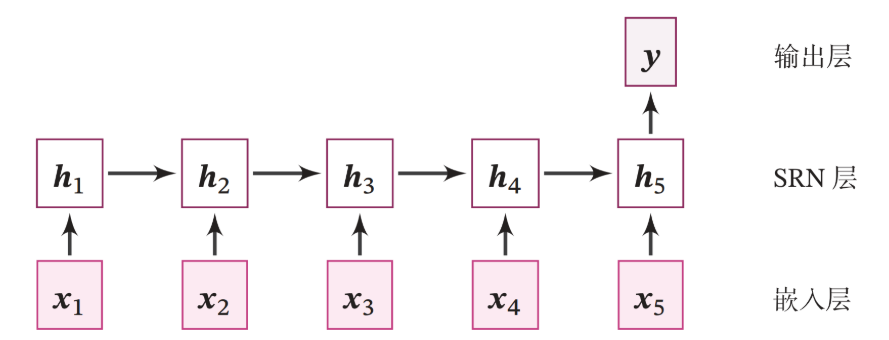


图 1 基于SRN模型的数字预测

① 嵌入层：将每个数字映射为向量，嵌入矩阵如图2所示；

【**向量能够表达该数字更多的信息**，利用向量进行数字求和任务，可以使得模型具有更强的拟合能力】

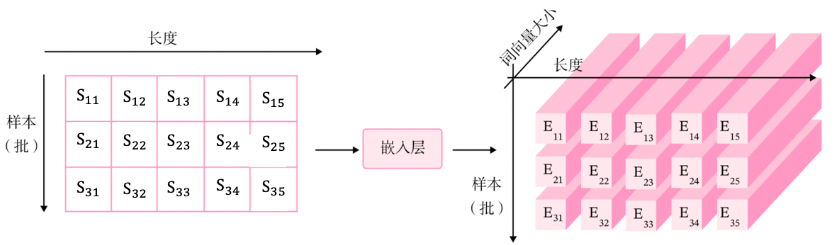


图 2 嵌入矩阵

· num\_embeddings: 嵌入矩阵中行的数量，表示类别数量。

· embedding\_dim: 嵌入向量的维度，表示每个数字被映射到的向量的维度。

· para\_attr: 参数初始化方式， xavier\_uniform，用来初始化嵌入矩阵。

② SRN 层：接收向量序列，更新循环单元，将最后时刻的隐状态作为整个序列的表示，更新方法如公式（2）所示；

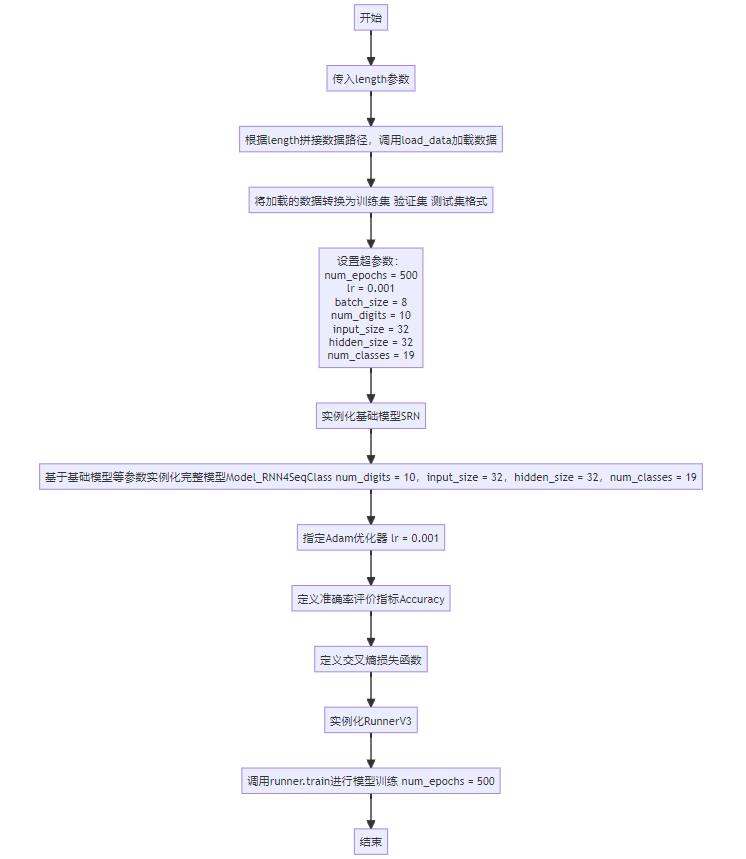
（2）

其中，**,B**为批大小，**L**为序列长度，**M**为嵌入维度，

**D**表示隐状态向量的维度。  
 ③ 输出层：一个线性层，将最后一个时刻的隐状态向量进行线性变换，输出分类的对数几率输出分类的结果，直接使用nn.Linear算子。

## **模型训练**

使用Runner进行训练，并分别进行数据长度为10, 15, 20, 25, 30, 35的数字预测模型训练实验。



## **模型评价**

在模型评价时，加载不同长度的效果最好的模型，然后使用测试集对该模型进行评价，观察模型在测试集上预测的准确度。

* **实验二：梯度爆炸实验**

## **梯度打印函数**

使用**custom\_print\_log**实现在训练过程中打印梯度的功能，定义W\_list, U\_list和b\_list，分别存储训练过程中参数W,U和b的梯度范数。

## **复现梯度爆炸现象**

使用SGD优化器将批大小和学习率调大为0.2，同时在计算交叉熵损失时，将reduction设置为sum，表示将损失进行累加。

## **解决梯度爆炸现象--梯度截断**

梯度截断是一种可以有效解决梯度爆炸问题的启发式方法，当梯度的模大于一定阈值时，就将它截断成为一个较小的数。一般有两种截断方式：按值截断和按模截断．本实验使用**按模截断**的方式解决梯度爆炸。

1. **实验数据及结果**

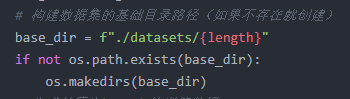
[【实验13】SRN的记忆能力&&梯度爆炸-CSDN博客](https://blog.csdn.net/qq_73704268/article/details/144245143?sharetype=blogdetail&sharerId=144245143&sharerefer=PC&sharesource=qq_73704268&spm=1011.2480.3001.8118)

1. **实验结论：**
2. 实验一通过构建不同长度的数字预测数据集DigitSum，成功测试了简单循环网络（SRN）的记忆能力。结果表明，SRN在处理较短序列时表现较好，但随着序列长度的增加，其性能逐渐下降，这与长程依赖问题的存在有关。
3. 实验二复现了简单循环网络中的梯度爆炸问题，通过增大批大小和学习率，观察到梯度范数的急剧增加。通过设置梯度范数的阈值，对梯度进行了裁剪，从而保持了模型训练的稳定性。
4. **总结及心得体会：**
5. 记得之前老师留的一个作业中encoder使用的是onehot编码，这次实验为了更好的表示数字使用的是更高级的embedding操作，定义的嵌入层通过嵌入矩阵将索引映射为高维向量，嵌入矩阵是一个可以学习的参数，实验中采用了xavier\_uniform\_来初始化嵌入矩阵参数，会使权重均匀分布，范围是-a,a,**a的计算方式如公式（3）所示，**此外PyTorch 内置的有torch.nn.Embedding的API，可以更方便的实现词嵌入。

（3）

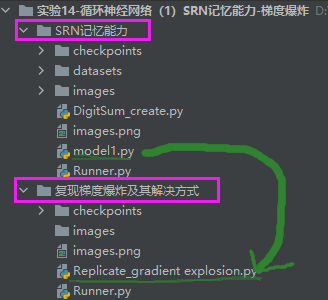
其中，是输入神经元的数量，是输出神经元的数量。

1. 本次实验还学会了很多和文件操作相关的知识点和函数，我觉得最有意思的一个是**os.makedirs**函数，用于递归地创建目录，能一次性创建多级目录，无需逐级创建（实验中主要在创建数据集和保存训练结果可视化图中用到了），主要语法格式如图所示，这样就不用手动创建文件夹了，很方便~



另一个是**os.path.join**函数，这个文件路径拼接利器，用于连接两个或多个路径组件，返回一个组合后的路径，举个例子multi\_path=os.path.join('home','user', 'documents','project','data.csv就相当于是 home\user\documents\project\data.csv 。

1. 由于第二个实验的梯度爆炸实验需要用到很多实验一的class类，为了减少代码量，我想到直接import导入，但是这时候就有一个问题，之前import导入其他模块都是在同一文件目录下的，而我的文件结构如下图：



当然我可以直接把model1复制到第二个实验的文件夹，但是我觉得肯定有一种方式不用这么干，于是我首先在model1中添加if \_\_name\_\_==”main”函数，在网上查找到如何跨文件夹导入py模块，了解到了import sys这个库，python中从其他目录中导入模块的关键是：系统（sys）能够找到通向模块文件的路径，所以我们只需要将model1所在的目录路径添加到实验二的代码中。

通用方式：

|  |
| --- |
| 程序结构示意图： |
| **--**any\_dir1 # 任意文件夹位置      |**--**mudule1.py  **def** func1()  **def** func2()  **--**any\_dir2 # 任意文件夹位置  |**--**module2.py |

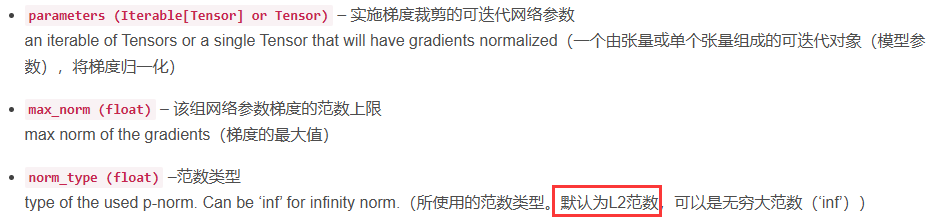
|  |
| --- |
| 想在module2.py中导入module1中的func1(),func2(): |
| # 脚本 mudule2.py 中  **import** sys  sys.path.append("/global/path/to/any\_dir1") # 将module1所在的文件夹路径放入sys.path中  **from** module1 **import** func1,func2 |

1. 第二个实验中，打印了梯度范数，做到这里想起来之前一个体验梯度消失的实验也打印了，为什么要打印梯度范数？ 当梯度过大时，它可能导致模型训练过程中的数值不稳定，进而影响模型的性能。打印范数可以帮助我们了解梯度的幅度大小。**范数可以衡量向量的大小**，因此通过打印梯度的范数，我们可以直观地看到梯度的幅度是否过大或过小。简单说，就是范数可以反应梯度的大小，打印范数我们可以及时知道梯度的情况。

本次实验体验了梯度爆炸，运行结果打印出的L2范数后续也都变为了0，后面使用了梯度截断的方式来解决 ，常见的梯度裁剪有两种：①确定一个范围，如果参数的gradient超过了，直接裁剪；②根据若干个参数的gradient组成的的vector的**L2 Norm**进行裁剪。

方式①函数是nn.utils.clip\_grad\_value\_：该函数通过梯度的**每个元素值**进行裁剪。接受最大值，并将梯度张量中超过指定值的元素截断。

方式②对应的函数nn.utils.clip\_grad\_norm\_，函数通过梯度的**范数值**进行裁剪。接受最大范数值，并在梯度的范数超过指定值时对梯度进行缩放。使用方法见下图（来源pytorch官方文档）



|  |
| --- |
| **参考链接：** |
| [Python os.makedirs和​os.mkdir详细介绍](https://blog.csdn.net/AI_dataloads/article/details/134339863?ops_request_misc=%7B%22request%5Fid%22%3A%22a31c40ad7ff6a935964ec768c272c073%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=a31c40ad7ff6a935964ec768c272c073&biz_id=0&utm_medium=distrib) |
| [python 导入其他目录下的模块\_python中要引入下一级目录下的模块怎么写-CSDN博客](https://blog.csdn.net/Strive_For_Future/article/details/106716745?ops_request_misc=&request_id=&biz_id=102&utm_term=python%E5%AF%BC%E5%85%A5%E5%92%AF%E5%85%B6%E4%BB%96%E6%96%87%E4%BB%B6%E5%A4%B9%E9%87%8C%E7%9A%84%E6%A8%A1%E5%9D%97&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-0-106716745.142^v100^pc_search_result_base7&spm=1018.2226.3001.4187) |
| [python中文件路径的设置\_python如何设置文件路径-CSDN博客](https://blog.csdn.net/weixin_39447007/article/details/110225055?ops_request_misc=%7B%22request%5Fid%22%3A%22e275a6e6da16488a270f54352a4acf81%22%2C%22scm%22%3A%2220140713.130102334.pc%5Fall.%22%7D&request_id=e275a6e6da16488a270f54352a4acf81&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-5-110225055-null-null.142^v100^pc_search_result_base7&utm_term=python%E8%AE%BE%E7%BD%AE%E6%96%87%E4%BB%B6%E5%B7%A5%E4%BD%9C%E8%B7%AF%E5%BE%84&spm=1018.2226.3001.4187) |
| [【Python】【三分钟】熟练使用 os.path.join()-CSDN博客](https://blog.csdn.net/2401_84204207/article/details/137822916?ops_request_misc=%7B%22request%5Fid%22%3A%22afbdb85490371f0b2b1c87f875c338bd%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=afbdb85490371f0b2b1c87f875c338bd&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-137822916-null-null.142^v100^pc_search_result_base7&utm_term=os.path.join()&spm=1018.2226.3001.4187) |
| [xavier\_uniform\_、均匀分布、高斯分布初始化嵌入权重\_xavier uniform-CSDN博客](https://blog.csdn.net/qq_45732909/article/details/137509343?ops_request_misc=%7B%22request%5Fid%22%3A%22e384384a88faee7b4a2b03c18bba763e%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=e384384a88faee7b4a2b03c18bba763e&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-137509343-null-null.142^v100^pc_search_result_base7&utm_term=xavier_uniform&spm=1018.2226.3001.4187) |
| [【Pytorch】torch.nn.init.xavier\_uniform\_()-CSDN博客](https://blog.csdn.net/weixin_44225182/article/details/126655294?ops_request_misc=%7B%22request%5Fid%22%3A%229df0946652c116bc99bbfa376eed0a5f%22%2C%22scm%22%3A%2220140713.130102334.pc%5Fall.%22%7D&request_id=9df0946652c116bc99bbfa376eed0a5f&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-7-126655294-null-null.142^v100^pc_search_result_base7&utm_term=xavier_uniform&spm=1018.2226.3001.4187) |
| [梯度裁剪中的NaN值处理 - 深入探究torch.nn.utils.clip\_grad\_norm\_和torch.nn.utils.clip\_grad\_value\_\_梯度为nan-CSDN博客](https://blog.csdn.net/C_C666/article/details/135233091?ops_request_misc=%7B%22request%5Fid%22%3A%22da9aff9e17b348682eceb67fcc862225%22%2C%22scm%22%3A%2220140713.130102334.pc%5Fall.%22%7D&request_id=da9aff9e17b348682eceb67fcc862225&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-1-135233091-null-null.142^v100^pc_search_result_base7&utm_term=nn.utils.clip_grad_norm_  %E2%9A%A0%EF%B8%8F %E6%B6%88%E6%81%AF%E6%9C%AA%E5%8F%91%E5%87%BA%EF%BC%8C%E5%86%8D%E5%8F%91%E4%B8%80%E4%B8%8B%E5%90%A7%E3%80%82&spm=1018.2226.3001.4187) |
| [torch.nn.utils.clip\_grad\_norm\_ — PyTorch 2.5 documentation](https://pytorch.org/docs/stable/generated/torch.nn.utils.clip_grad_norm_.html) |