哈爾濱工業大學

模式识别与深度学习实验报告

实验 四

题　　目　循环神经网络

学　　院　计算学部

专　　业

学　　号

学　　生

任 课 教 师　左旺孟

哈尔滨工业大学计算机科学与技术学院

2023 年春季

实验四 报告

一、 实验目的

利用 Pytorch 自己实现 RNN、GRU、LSTM 和 Bi-LSTM。不可直接调用 nn.RNN(), nn.GRU(), nn.LSTM()。

利用上述四种结构进行文本多分类（60%）计算测试结果的准确率、召回率和 F1 值；对比分析四种结构的实验结果。

任选上述一种结构进行温度预测（40%）使用五天的温度值预测出未来两天的温度值；给出与真实值的平均误差和中位误差。

二、 实验环境

硬件配置（华为云服务器）：

CPU: 8 核 64GB

GPU: 1*P100(16GB) or 1*V100(32GB)

软件配置：

OS：ubuntu18.04

IDE：Jupyter Notebook

Packages in Environment：pytorch1.8-cuda10.2-cudnn7

三、 实验内容

3.1 数据集处理

文本多分类 onling_shopping_10_cats 数据集：

首先使用 pandas 读入文件，对每一条数据遍历取出中间的评论和物体类别。对于每一条评论用 jieba 分词，再使用所有评论训练 gensim 库中的 Word2Vec 模型，并利用该模型对每一个词向量进行 embedding 和映射。这样对于每一条评论，每个词向量是 input-size 维，但是每一条评论的词个数不一样，是动态长度，因此 batch-size 只能为 1。

对评论进行映射之后，我们再一次遍历所有评论，i 为数据编号。i%5=4 做验证集，i%5=0 做测试集，其余作为训练集。最后调用 DataLoader 实现数据集迭代器的构造。

温度预测 jena_climate_2009_2016 数据集：

首先使用 pandas 读入文件，按照时间顺序依次读入各个时间点的 p(mbar)、T(degC)、T(potK)、Tdew(degC)、rh(%)五维数据。将时序数据处理，通过长度表示一周的滑窗，取前五天的所有数据作为原始数据，后两天的所有数据作为标签。由于使用 GRU 神经网络进行训练，需要对数据进行归一化。

八年间的前六年数据作为训练集，后两年作为测试集。最后调用 DataLoader 实现数据集迭代器的构造。

3.2 RNN 网络结构

相比之前的卷积神经网络等，特点是将上一次隐藏层的值也输入神经网络。

```
class RNN(nn.Module):
    def __init__(self, args, output_size):
        super(RNN, self).__init__()
        self.device = args.device
        self.hidden_size = args.hidden_size
        self.input_size = args.input_size
        self.i2h = nn.Linear(self.input_size + self.hidden_size, self.hidden_size)
        self.h2o = nn.Linear(self.hidden_size, output_size)
        self.tanh = nn.Tanh()

    def forward(self, x, hidden=None):
        global output
        if not hidden:
            hidden = torch.zeros(1, self.hidden_size).to(self.device)
        x = x[0]
        for i in range(x.shape[0]):
            token = x[i: i + 1]
            combined = torch.cat((token, hidden), 1)
            hidden = self.tanh(self.i2h(combined))
            output = self.h2o(hidden)
        return output
```

3.3 LSTM 网络结构

LSTM 改善了 RNN 短期依赖问题，加入长期依赖。

LSTM-Bi 网络是在 LSTM 的基础上，同时考虑向前循环和向后循环的隐藏层状态，然后输出。

```
class LSTM(nn.Module):
    def __init__(self, args, output_size,  bidirectional=False):
        super(LSTM, self).__init__()
        self.device = args.device
        self.hidden_size = args.hidden_size
        self.input_size = args.input_size
        self.bidirectional = bidirectional
        self.forget_gate = nn.Linear(self.input_size + self.hidden_size, self.hidden_size)
        self.input_gate = nn.Linear(self.input_size + self.hidden_size, self.hidden_size)
        self.c_gate = nn.Linear(self.input_size + self.hidden_size, self.hidden_size)
        self.output_gate = nn.Linear(self.input_size + self.hidden_size, self.hidden_size)
        if not bidirectional:
            self.h2o = nn.Linear(self.hidden_size, output_size)
        else:
            self.h2o = nn.Linear(self.hidden_size * 2, output_size)
        self.tanh = nn.Tanh()
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        global ct, result
        x = x[0]
        if not self.bidirectional:
            hidden = torch.zeros(1, self.hidden_size).to(self.device)
            ct = torch.zeros(1, self.hidden_size).to(self.device)
            for i in range(x.shape[0]):
                token = x[i: i + 1]
                combined = torch.cat((token, hidden), 1)
                forget = self.sigmoid(self.forget_gate(combined))
                input = self.sigmoid(self.input_gate(combined))
                c_ = self.tanh(self.c_gate(combined))
                output = self.sigmoid(self.output_gate(combined))
                ct = ct * forget + input * c_
                hidden = self.tanh(ct) * output
                result = self.h2o(hidden)
            return result
```

```
            else:
                num = x.shape[0]
                hidden1 = torch.zeros(1, self.hidden_size).to(self.device)
                hidden2 = torch.zeros(1, self.hidden_size).to(self.device)
                hidden1s = []
                hidden2s = []
                ct1 = torch.zeros(1, self.hidden_size).to(self.device)
                ct2 = torch.zeros(1, self.hidden_size).to(self.device)
                for i in range(num):
                    token1 = x[i: i+1]
                    token2 = x[num-i-1: num-i]
                    combined1 = torch.cat((token1, hidden1), 1)
                    combined2 = torch.cat((token2, hidden2), 1)
                    forget1 = self.sigmoid(self.forget_gate(combined1))
                    forget2 = self.sigmoid(self.forget_gate(combined2))
                    input1 = self.sigmoid(self.input_gate(combined1))
                    input2 = self.sigmoid(self.input_gate(combined2))
                    c_1 = self.tanh(self.c_gate(combined1))
                    c_2 = self.tanh(self.c_gate(combined2))
                    output1 = self.sigmoid(self.output_gate(combined1))
                    output2 = self.sigmoid(self.output_gate(combined2))
                    ct1 = ct1 * forget1 + input1 * c_1
                    ct2 = ct2 * forget2 + input2 * c_2
                    hidden1 = self.tanh(ct1) * output1
                    hidden2 = self.tanh(ct2) * output2
                    hidden1s.append(hidden1)
                    hidden2s.insert(0, hidden2)
                hidden1 = torch.stack(hidden1s).mean(0)
                hidden2 = torch.stack(hidden2s).mean(0)
                result = self.h2o(torch.cat((hidden1, hidden2), 1))
                return result
```

3.4 GRU 神经网络

GRU 只有两个门（update 和 reset），LSTM 有三个门（forget，input，output），GRU 直接将 hidden state 传给下一个单元，而 LSTM 则需要计算新的 memory cell。

```python
class GRU(nn.Module):
    def __init__(self, args, output_size):
        super(GRU, self).__init__()
        self.device = args.device
        self.input_size = args.input_size
        self.hidden_size = args.hidden_size
        self.reset_gate = nn.Linear(self.input_size + self.hidden_size, self.hidden_size)
        self.update_gate = nn.Linear(self.input_size + self.hidden_size, self.hidden_size)
        self.h_gate = nn.Linear(self.input_size + self.hidden_size, self.hidden_size)
        self.h2o = nn.Linear(self.hidden_size, output_size)
        self.tanh = nn.Tanh()
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        global output
        hidden = torch.zeros(1, self.hidden_size).to(self.device)
        ones = torch.ones(1, self.hidden_size).to(self.device)
        x = x[0]
        for i in range(x.shape[0]):
            token = x[i: i + 1]
            combined = torch.cat((token, hidden), 1)  # 1 x (128+_)
            reset = self.sigmoid(self.reset_gate(combined))
            zt = self.sigmoid(self.update_gate(combined))
            combined2 = torch.cat((token, reset * hidden), 1)
            h_ = self.tanh(self.h_gate(combined2))
            hidden = zt * hidden + (ones - zt) * h_
            output = self.h2o(hidden)
        return output
```

四、 实验结果与分析

4.1 文本多分类

4.1 实验基本设置

训练轮数（Epoch）：Shopping——5(RNN)，3(GRU/LSTM)，1(Bi-LSTM)

Climate——500(GRU)

训练设备：GPU

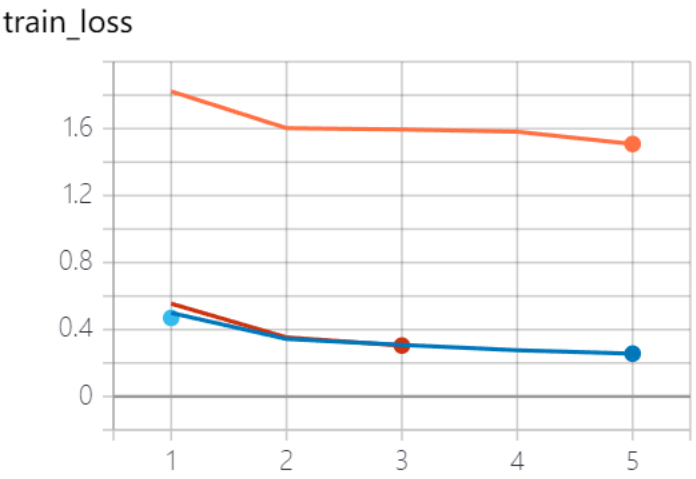
Batch size: Shopping——1，Climate——16

4.2 结果与分析

4.2.1shopping 数据神经网络对比分析

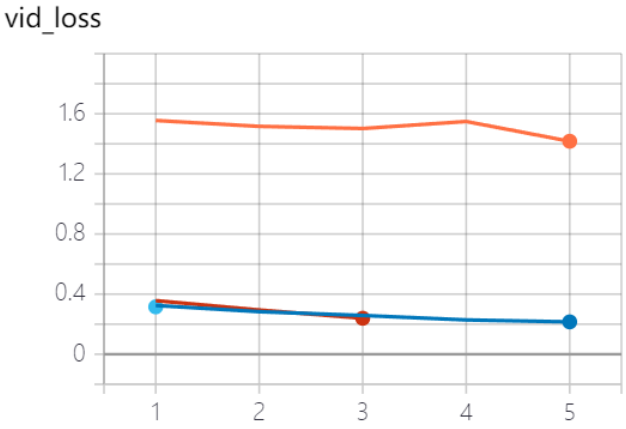
相同参数：学习率，Adam 优化方法

不同参数：神经网络



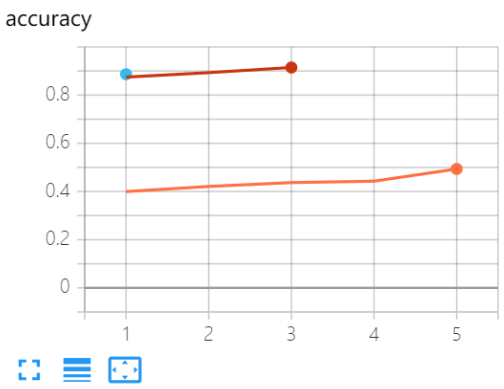
Name	Smoothed	Value	Step	Time	Relative
Bi-LSTM	0.4693	0.4693	1	Wed May 24, 13:36:35	0s
GRU	0.2559	0.2559	5	Tue May 23, 22:32:14	1h 42m 3s
LSTM	0.304	0.304	3	Wed May 24, 11:47:28	1h 7m 38s
RNN	1.508	1.508	5	Tue May 23, 20:11:12	42m 45s

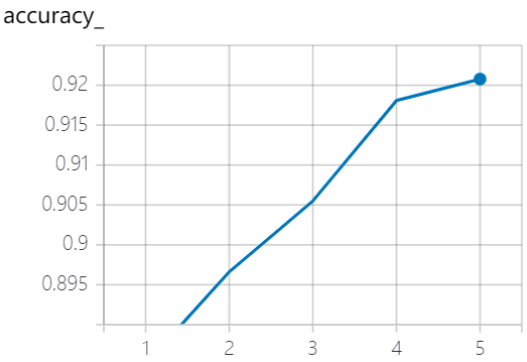
训练集损失



Name	Smoothed	Value	Step	Time	Relative
Bi-LSTM	0.3152	0.3152	1	Wed May 24, 13:53:01	0s
GRU	0.2152	0.2152	5	Tue May 23, 22:38:45	1h 42m 11s
LSTM	0.2393	0.2393	3	Wed May 24, 11:57:01	1h 7m 44s
RNN	1.417	1.417	5	Tue May 23, 20:14:47	42m 45s

验证集损失





Name	Smoothed	Value	Step	Time	Relative
Bi-LSTM	0.8867	0.8867	1	Wed May 24, 13:53:01	0s
LSTM	0.9147	0.9147	3	Wed May 24, 11:57:01	1h 7m 44s
RNN	0.4935	0.4935	5	Tue May 23, 20:14:47	42m 45s

Name	Smoothed	Value	Step	Time	Relative
GRU	0.9208	0.9208	5	Tue May 23, 22:38:45	1h 42m 11s

验证集准确率

书籍	0.35	0.14	0.20	2310
平板	0.39	0.44	0.41	6000
手机	0.35	0.29	0.31	1395
水果	0.61	0.56	0.59	6000
洗发水	0.51	0.19	0.28	6000
热水器	0.50	0.01	0.01	344
蒙牛	0.76	0.73	0.74	1219
衣服	0.37	0.76	0.49	6000
计算机	0.27	0.01	0.02	2396
酒店	0.72	0.87	0.79	6000
accuracy			0.49	37664
macro avg	0.48	0.40	0.38	37664
weighted avg	0.49	0.49	0.46	37664

书籍	0.97	0.97	0.97	2310
平板	0.89	0.83	0.86	6000
手机	0.87	0.94	0.91	1395
水果	0.93	0.93	0.93	6000
洗发水	0.89	0.86	0.88	6000
热水器	0.76	0.73	0.74	344
蒙牛	1.00	1.00	1.00	1219
衣服	0.89	0.94	0.91	6000
计算机	0.94	0.96	0.95	2396
酒店	0.99	0.99	0.99	6000
accuracy			0.92	37664
macro avg	0.91	0.92	0.91	37664
weighted avg	0.92	0.92	0.92	37664

	precision	recall	f1-score	support
书籍	0.96	0.97	0.96	2310
平板	0.86	0.84	0.85	6000
手机	0.91	0.90	0.91	1395
水果	0.95	0.90	0.93	6000
洗发水	0.83	0.89	0.86	6000
热水器	0.84	0.61	0.71	344
蒙牛	1.00	1.00	1.00	1219
衣服	0.91	0.92	0.91	6000
计算机	0.94	0.93	0.94	2396
酒店	0.99	0.99	0.99	6000
accuracy			0.91	37664
macro avg	0.92	0.90	0.91	37664
weighted avg	0.92	0.91	0.91	37664

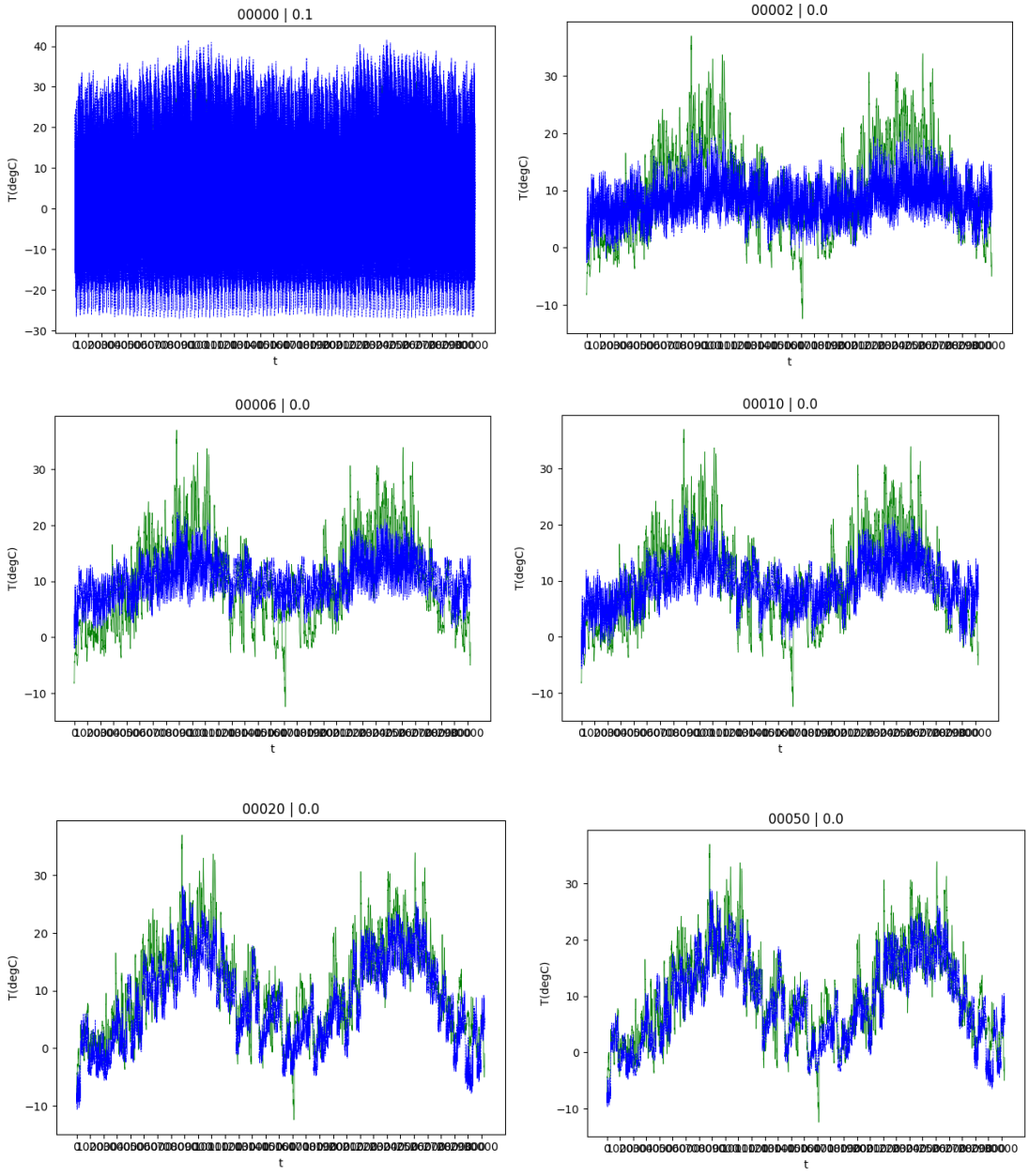
	precision	recall	f1-score	support
书籍	0.96	0.94	0.95	2310
平板	0.81	0.81	0.81	6000
手机	0.95	0.71	0.81	1395
水果	0.90	0.90	0.90	6000
洗发水	0.80	0.86	0.83	6000
热水器	0.79	0.44	0.57	344
蒙牛	1.00	1.00	1.00	1219
衣服	0.88	0.89	0.89	6000
计算机	0.94	0.87	0.91	2396
酒店	0.97	0.99	0.98	6000
accuracy			0.89	37664
macro avg	0.90	0.84	0.86	37664
weighted avg	0.89	0.89	0.89	37664

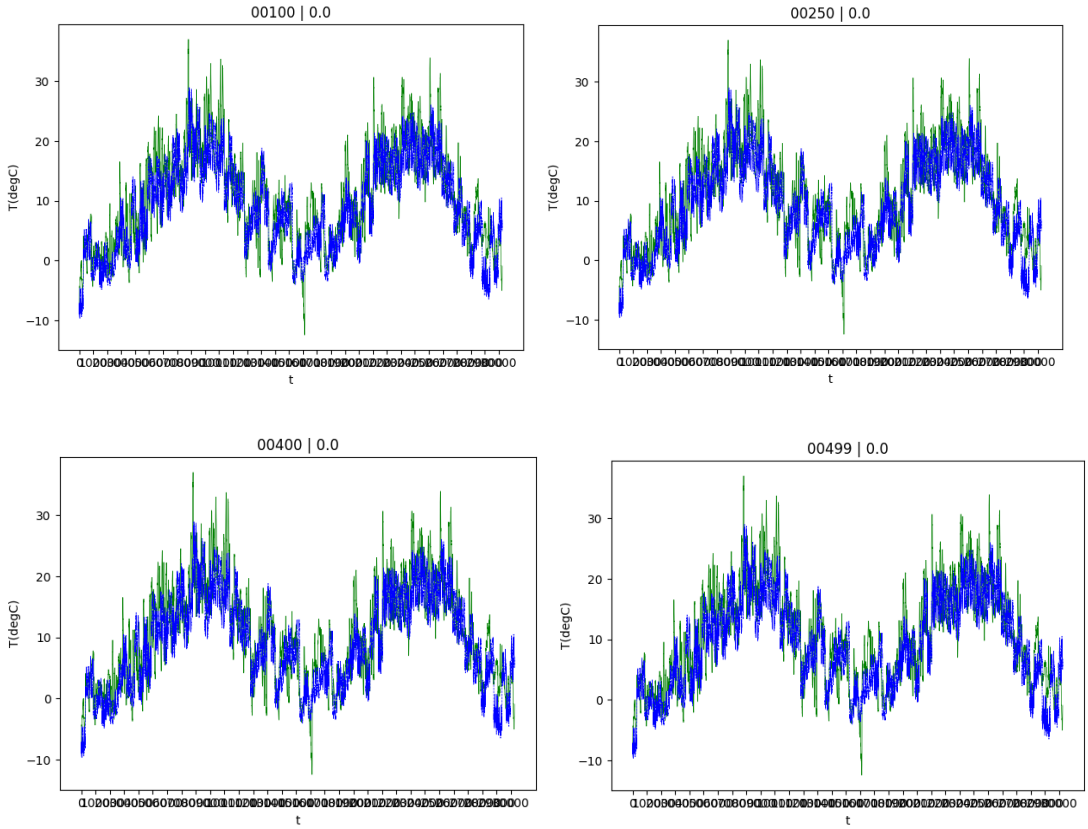
测试集准确率、召回率和 F1 率(从上到下依次是：RNN/GRU/LSTM/Bi-LSTM)

从中可以看出，GRU/LSTM/Bi-LSTM 三种算法的分类准确率都到达了 88%以上，虽然只训练了 1-3 个 epoch，依然取得了很好的效果。其中 GRU 准确率和训练速度的比值最大，是最佳模型。然而，RNN 的准确率很低，只有 49%，只在蒙牛、酒店等简单文本上表现略好，可能与其无法处理长期依赖的结构有关。

4.2.2 温度时序预测结果

该问题通过 GRU 网络进行时序预测，预测结果如下图。其中图标题分别是迭代轮数和测试集平均误差，最终平均误差达到 0.006 左右。图像将需要被预测的后两年的周末数据单独绘制，对比预测和真实值的差距，其中绿色实线是真实值，蓝色虚线是预测值。





从中可以看出，随着迭代轮数的增加，预测越来越精准，GRU 神经网络具有预测这个温度问题的能力，但是可能对极端最值的拟合程度不够精准。

五、 结语

通过该实验，我体会到了对时序数据进行预测和分类的过程，熟悉了常见的 RNN/GRU/LSTM/Bi-LSTM 神经网络结构。局限性其一在于文本处理使用动态长度导致 batch_size=1 迭代速度非常慢，训练等待时间特别长，好在效果不错。其二在于，我在初步分析温度预测问题是只将其看作了一个简单的序列向量的预测问题，没有意识到时间(而非顺序)本身对于温度是由很大影响的，只利用了其他 5 维的数据进行预测，加入处理后的时间数据可能会提升预测精度，但是可以在损失函数中剔除。

此外，根据我不考虑时间本身作为特征的预测方法，因为温度不会受到双休日的影响，因此在生成数据集时不将滑窗设为 1 周，设为更短的 10 分钟或 1 小时、一天能够获得更多的数据量，可能会得到更好的预测效果。但因为时间有限没有进行尝试。