**深度学习实验报告**

实验4：循环神经网络

姓名：姚舜宇

学号：1190202107

班级：1903602

**实验环境：**

Windows11

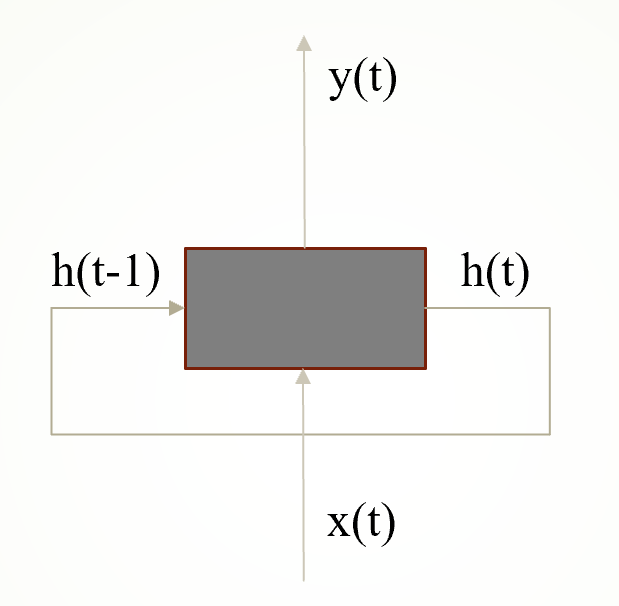
Python3.8

PyTorch1.10.2

**实验内容：**

**使用pytorch实现RNN，GRU，LSTM，Bi-LSTM**

基本结构：



对于一个循环结构的一个单元(cell)，假设输入为x，输出为y，隐含状态为h。

1. RNN：



|  |
| --- |
| def forward(self, x, hidden):  # x: batch\_size, in\_feature hidden: batch\_size, hidden\_size  x\_cat = torch.cat((x, hidden), 1)  hidden = self.nonlinearity(self.in2hidden(x\_cat))  out = self.hidden2out(hidden)  # out: batch\_size, out\_feature  return out, hidden |

1. GRU：



|  |
| --- |
| def forward(self, x, hidden):  # x: batch\_size, in\_feature hidden: batch\_size, hidden\_size  Rt = self.sigmoid(self.w\_xr(x) + self.w\_hr(hidden))  Zt = self.sigmoid(self.w\_xz(x) + self.w\_hz(hidden))  Ht\_ = self.tanh(self.w\_xh(x) + self.w\_hh(Rt \* hidden))  hidden = Zt \* hidden + (1 - Zt) \* Ht\_  out = self.out(hidden)  # out: batch\_size, out\_feature  return out, hidden |

1. LSTM：

需要引入细胞状态变量C。



|  |
| --- |
| def forward(self, x, hidden, C\_t\_1):  # x: batch\_size, in\_feature hidden: batch\_size, hidden\_size  x\_cat = torch.cat((x, hidden), dim=1)  f\_t = self.sigmoid(self.w\_f(x\_cat))  i\_t = self.sigmoid(self.w\_i(x\_cat))  C\_t\_ = self.tanh(self.w\_c(x\_cat))  C\_t = f\_t \* C\_t\_1 + i\_t \* C\_t\_  o\_t = self.sigmoid(self.w\_o(x\_cat))  hidden = o\_t \* self.tanh(C\_t)  out = self.out(hidden) if not self.bidirectional else None  # out: batch\_size, out\_feature  return out, hidden, C\_t |

1. Bi-LSTM：

和LSTM较为相似。区别在于需要双向计算序列数据的隐含状态，将最后一个词的隐含值连接后再使用线性层得到最终输出。

|  |
| --- |
| if self.bidirectional:  for i in range(seq\_len):  \_, hidden1, C\_t1 = self.lstm\_cell\_1(x[i], hidden1, C\_t1)  \_, hidden2, C\_t2 = self.lstm\_cell\_2(x[seq\_len - i - 1], hidden2, C\_t2)  out[i] = self.out(torch.cat((hidden1, hidden2), dim=1)) |

**文本多分类**

数据读取：

自定义实现dataset类，共62774条数据，按照规则选择60%进行训练，20%验证，20%测试。

具体实现：

首先使用python库jieba对中文进行分词，然后使用微博的词向量模型对中文自词进行转换，变为此向量的格式，维度为300维。

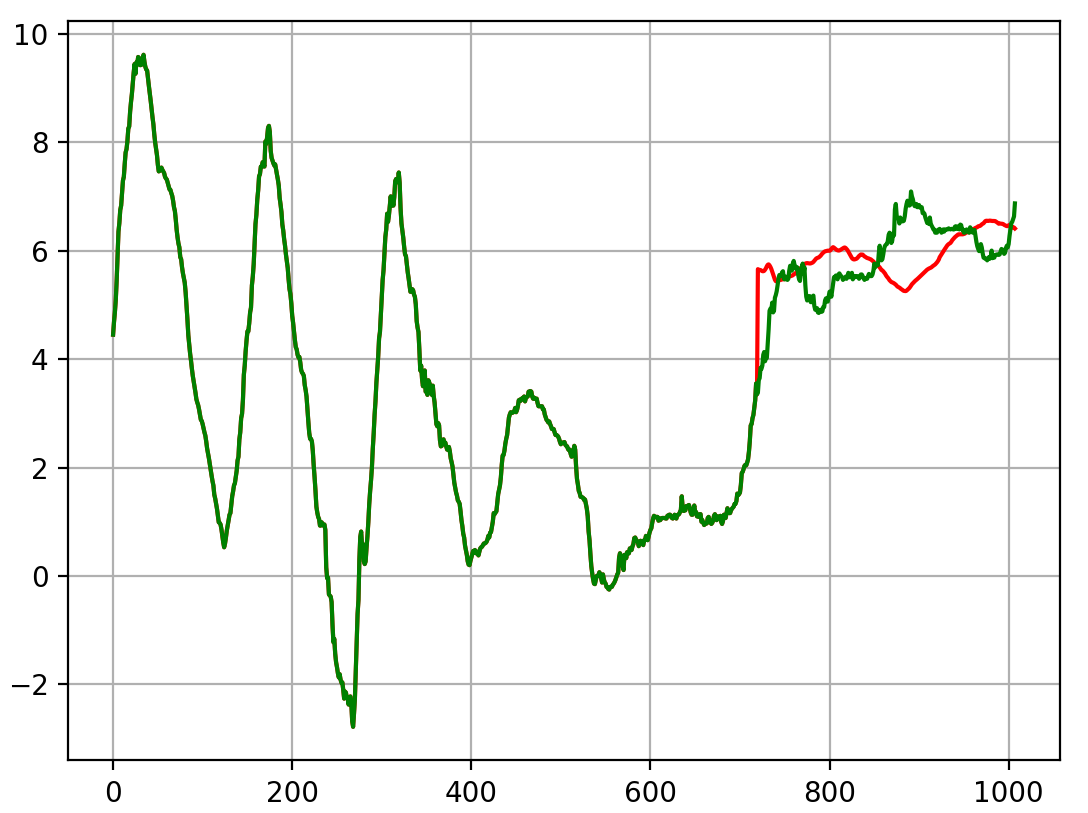
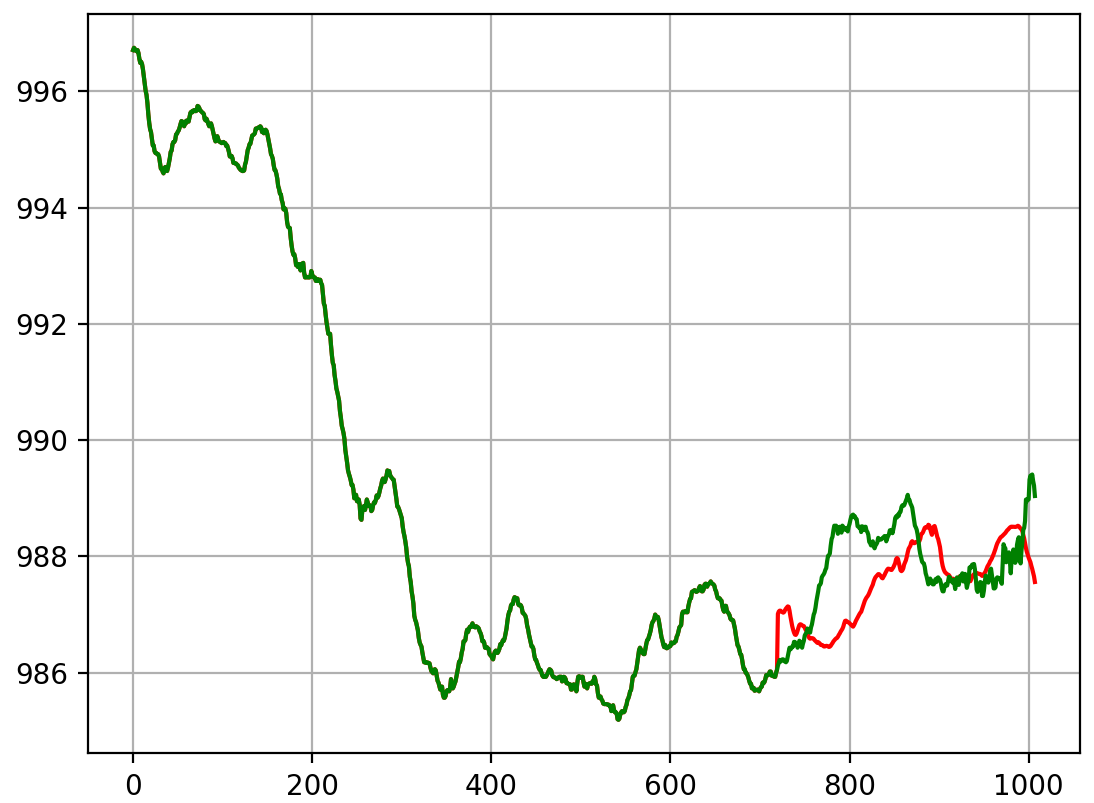
使用Adam优化器，初始学习率为0.001，训练3个epoch，每个epoch之后学习率下降为原来的0.1倍。

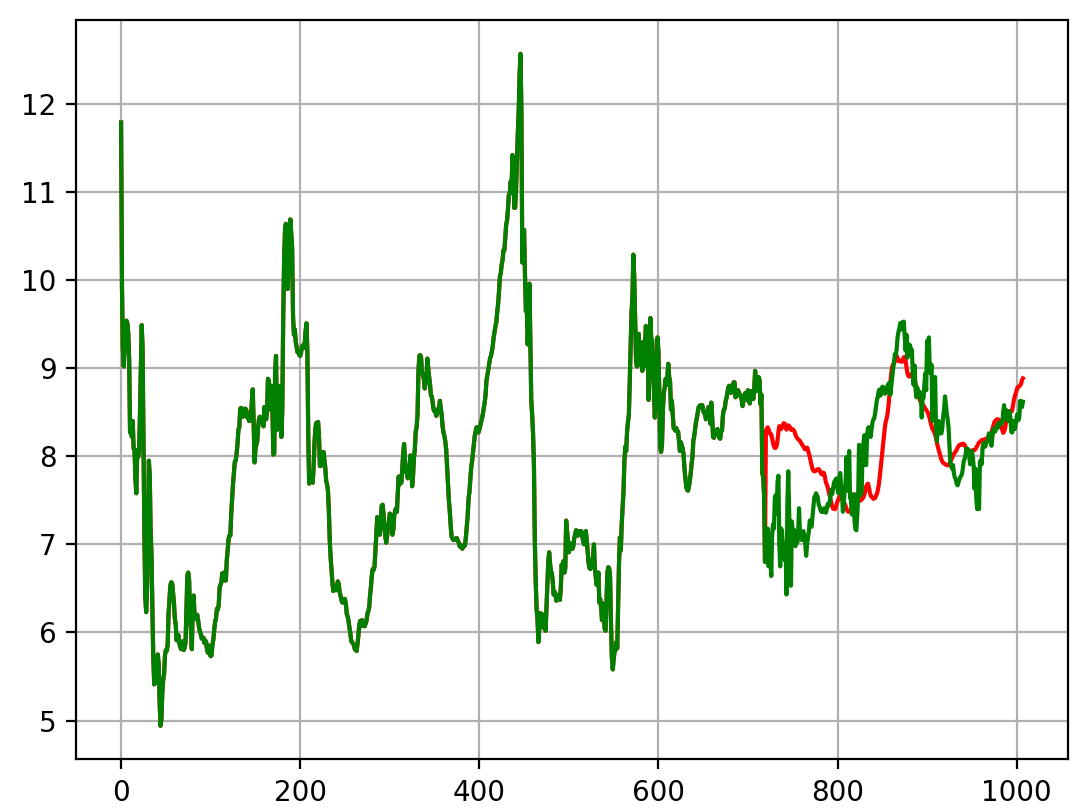
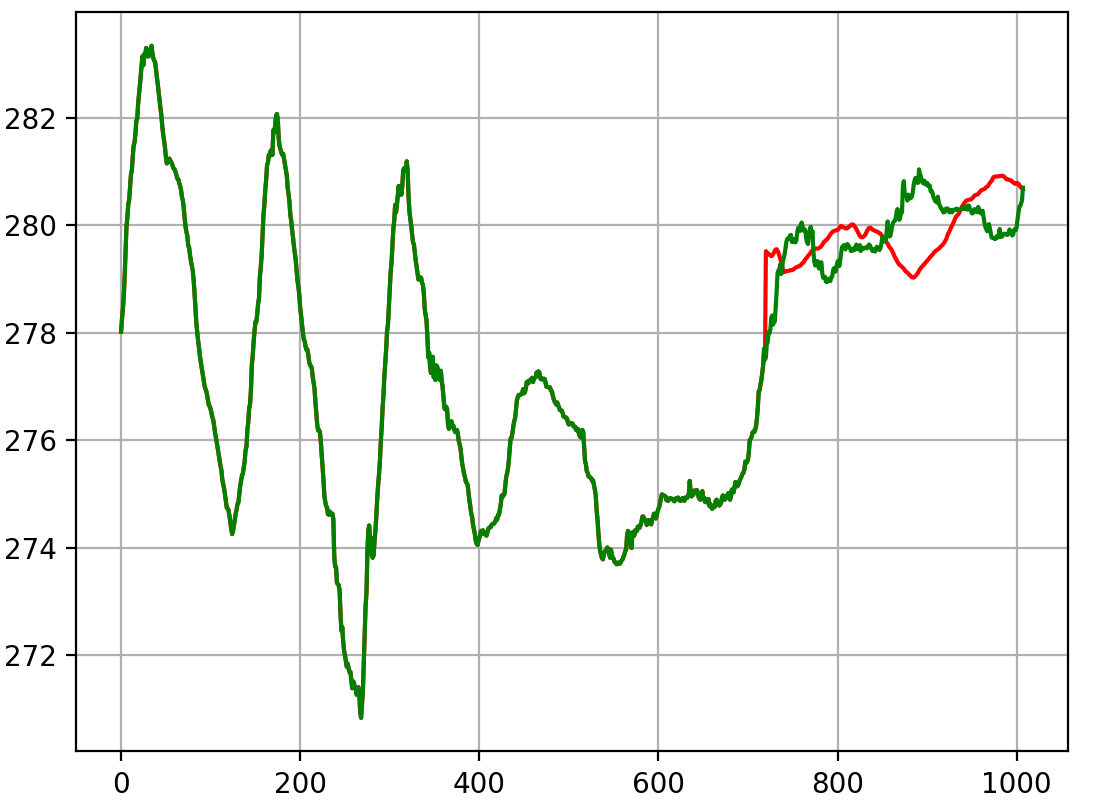
实验结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模型 | 准确率(valid/test) | 精确率(valid/test) | 召回率(valid/test) | F1值(valid/test) |
| RNN | 0.6843/0.6824 | 0.6415/0.5164 | 0.5048/0.5044 | 0.5650/0.5103 |
| GRU | 0.7388/0.7362 | 0.6069/0.6052 | 0.6104/0.6044 | 0.6087/0.6048 |
| LSTM | 0.7592/0.7549 | 0.6704/0.6527 | 0.6367/0.6303 | 0.6531/0.6413 |
| Bi-LSTM | **0.7743/0.7700** | **0.6924/0.7019** | **0.6537/0.6492** | **0.6725/0.6745** |

**温度预测**

使用jena\_climate\_2009\_2016数据集，包含420551条数据。取前75%的数据进行训练。使用Bi-LSTM模型，训练2个epoch。考虑到测试集的要求，每七天作为一段，前5天的数据（720条）用于训练，后2天（288条）用于测试，所以设定在训练中，序列长度为720，目标拟合长度为288。即有720个cell，使用后288个cell的输出作为拟合的结果。选取部分段的前四个属性的拟合效果如下图：





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 图1 | 图2 | 图3 | 图4 |
| 平均误差 | 0.7164 | 0.5992 | 0.6705 | 0.4451 |
| 中位误差 | 0.6236 | 0.5074 | 0.5607 | 0.3182 |

**加载预训练模型**

|  |
| --- |
| device = torch.device('cuda') if opt.gpu == '0' else torch.device('cpu')  model = lstm.my\_LSTM(in\_feature=opt.word\_dim, out\_feature=opt.num\_classes,  hidden\_size=opt.hidden\_size, bidirectional=True, device=device)  model = torch.nn.DataParallel(model)  model.load\_state\_dict(torch.load('./checkpoint/shopping\_bilstm\_e3.pth')['state\_dict']) |

其中模型部分可以调整。

**实验结论**

分别使用RNN，GRU，LSTM，Bi-LSTM对文本进行分类，发现Bi-LSTM性能优于其他三种。Bi-LSTM中加入了遗忘门，细胞状态，并且使用了双向传递的机制，使得信息提取能力更高。在温度预测中，使用Bi-LSTM进行处理，效果并不算理想，在预测领域，序列模型可能还有提升空间。

手动实现RNN，GRU，LSTM，Bi-LSTM，对于循环结构的网络以及序列类型的数据更加熟悉了。