第四次作业实验报告

任务1 lstm，gru，rnn对比实验

任务1.1 网络输入，输出，初始化参数：以lstm为例

参数表：

torch.nn.LSTM(input_size,hidden_size,,num_layers=1,bias=True,batch_first=False,dropout=0.0,bidirectional=False,proj_size=0,device=None,dtype=None)

参数解释：

- input_size:input的大小（由词嵌入的维度给出）
- hidden_size:隐藏层的大小（即H的size）
- num_layers:循环层层数，即将几个LSTMcell串联在一起当成一个"LSTMcell"，默认为1
- bias：偏置是否开启
- batch_first:更改输入和输出的格式，让batch的维度放到第一维。
- dropout:一个浮点数，可以在lstm的输出（除了最后一层）上加dropout，dropout的倍率由其决定。
- bidirectional:转为bidirectional LSTM
- proj_size:若非0，加一个可学习的无偏置层，将输出转为proj_size维

输入：

对于本代码给出的实现，输入应当为(L,N,Hin)的一个tensor，L为字串长度，N为batchsize，Hin是词嵌入的维数。

输出：

有3个输出。

- output:一个(L,N,D*Hout)的tensor，包含了各个输出。
- hn:一个(D*num_layers,N,Hout)的tensor，显示了最后一层的hidden
- cn:一个(D*num_layers,N,Hcell)的tensor，显示了最后一层的cell

任务1.2:实验三种recurrent net:

RNN:

```
vocab_size:  20001
ImdbNet(
  (embedding): Embedding(20001, 64)
  (lstm): RNN(64, 64)
  (linear1): Linear(in_features=64, out_features=128, bias=True)
  (act1): ReLU()
  (linear2): Linear(in_features=128, out_features=2, bias=True)
)
Train Epoch: 1 Loss: 0.589669   Acc: 0.674022
Test set: Average loss: 0.4921, Accuracy: 0.7615
Train Epoch: 2 Loss: 0.416196   Acc: 0.811302
Test set: Average loss: 0.3963, Accuracy: 0.8271
Train Epoch: 3 Loss: 0.325641   Acc: 0.862420
Test set: Average loss: 0.3666, Accuracy: 0.8453
Train Epoch: 4 Loss: 0.263239   Acc: 0.895867
Test set: Average loss: 0.3710, Accuracy: 0.8463
Train Epoch: 5 Loss: 0.215566   Acc: 0.920128
Test set: Average loss: 0.3849, Accuracy: 0.8532
```

LSTM:

```
vocab_size:  20001
```

```
ImdbNet(
  (embedding): Embedding(20001, 64)
  (lstm): LSTM(64, 64)
  (linear1): Linear(in_features=64, out_features=128, bias=True)
  (act1): ReLU()
  (linear2): Linear(in_features=128, out_features=2, bias=True)
)
Train Epoch: 1 Loss: 0.575902   Acc: 0.677965
Test set: Average loss: 0.4573, Accuracy: 0.7812
Train Epoch: 2 Loss: 0.394530   Acc: 0.823283
Test set: Average loss: 0.3772, Accuracy: 0.8307
Train Epoch: 3 Loss: 0.303056   Acc: 0.875399
Test set: Average loss: 0.3626, Accuracy: 0.8384
Train Epoch: 4 Loss: 0.239461   Acc: 0.905551
Test set: Average loss: 0.3628, Accuracy: 0.8495
Train Epoch: 5 Loss: 0.184152   Acc: 0.932358
Test set: Average loss: 0.3700, Accuracy: 0.8544
```

GRU:

```
vocab_size:  20001
ImdbNet(
  (embedding): Embedding(20001, 64)
  (gru): GRU(64, 64)
  (linear1): Linear(in_features=64, out_features=128, bias=True)
  (act1): ReLU()
  (linear2): Linear(in_features=128, out_features=2, bias=True)
)
Train Epoch: 1 Loss: 0.566251   Acc: 0.689197
Test set: Average loss: 0.4533, Accuracy: 0.7828
Train Epoch: 2 Loss: 0.380059   Acc: 0.832718
Test set: Average loss: 0.3677, Accuracy: 0.8394
Train Epoch: 3 Loss: 0.291812   Acc: 0.879343
Test set: Average loss: 0.3668, Accuracy: 0.8362
Train Epoch: 4 Loss: 0.230726   Acc: 0.909645
Test set: Average loss: 0.3492, Accuracy: 0.8578
Train Epoch: 5 Loss: 0.177860   Acc: 0.935403
Test set: Average loss: 0.3772, Accuracy: 0.8606
```

对比来看，三者都有一定的过拟合现象出现，由于数据量不大，这是难免的。然而，相较而言LSTM和GRU的acc都达到了85以上，rnn却较差，恰好证明了lstm，gru的优越性。

任务2 手写lstm实验

任务2.1 手写lstm

代码见文件夹。

我先写了一个裸的lstm，由于运行速度慢，收敛效果差，考虑优化。

首先是矩阵乘法压缩。标准版的lstm有3个需要学习的层，计算较慢，gpu通信消耗大，处理繁杂。将这三个层concate在一起，变成一个较大的可学习层，使用时将层点输出分成三份即可。

其次是在时间序列上取均值。该优化本质上是在整个输出结果上看问题，可以将梯度在时间序列上取均值，从而缓解梯度消失和梯度爆炸的问题，让训练速度加快，模型快速收敛。

结果如下。

```
Net(
  (embedding): Embedding(20001, 64)
  (lstm): LSTM(
    (cell): LSTMCell(
      (W): Linear(in_features=128, out_features=256, bias=True)
    )
  )
  (fc1): Linear(in_features=64, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=2, bias=True)
```

```
  (fc2): Linear(in_features=64, out_features=2, bias=True)
)
Train Epoch: 1 Loss: 0.584047   Acc: 0.678115
Test set: Average loss: 0.4894, Accuracy: 0.7581
Train Epoch: 2 Loss: 0.395507   Acc: 0.822833
Test set: Average loss: 0.3819, Accuracy: 0.8307
Train Epoch: 3 Loss: 0.301345   Acc: 0.875449
Test set: Average loss: 0.3550, Accuracy: 0.8503
Train Epoch: 4 Loss: 0.240772   Acc: 0.907149
Test set: Average loss: 0.3539, Accuracy: 0.8534
Train Epoch: 5 Loss: 0.191088   Acc: 0.928764
Test set: Average loss: 0.3526, Accuracy: 0.8602
```

任务2.2 lstm调参

修改网络结构

修改隐藏层大小。

```
Net(
  (embedding): Embedding(20001, 64)
  (lstm): LSTM(
    (cell): LSTMCell(
      (W): Linear(in_features=192, out_features=512, bias=True)
    )
  )
  (fc1): Linear(in_features=128, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=2, bias=True)
)
Train Epoch: 1 Loss: 0.576297   Acc: 0.680162
Test set: Average loss: 0.4656, Accuracy: 0.7882
Train Epoch: 2 Loss: 0.383039   Acc: 0.827825
Test set: Average loss: 0.3714, Accuracy: 0.8325
Train Epoch: 3 Loss: 0.287097   Acc: 0.880292
Test set: Average loss: 0.3494, Accuracy: 0.8479
Train Epoch: 4 Loss: 0.226131   Acc: 0.910393
Test set: Average loss: 0.3395, Accuracy: 0.8505
Train Epoch: 5 Loss: 0.172699   Acc: 0.938349
Test set: Average loss: 0.3689, Accuracy: 0.8558
```

加大隐藏层大小增强了网络拟合能力。

将两层LSTMCell串联

```
Net(
  (embedding): Embedding(20001, 64)
  (lstm): LSTM(
    (cell): LSTMCell(
      (W): Linear(in_features=128, out_features=256, bias=True)
    )
    (cell2): LSTMCell(
      (W): Linear(in_features=192, out_features=256, bias=True)
    )
  )
  (fc1): Linear(in_features=64, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=2, bias=True)
)
Train Epoch: 1 Loss: 0.572934   Acc: 0.687550
Test set: Average loss: 0.4678, Accuracy: 0.7783
Train Epoch: 2 Loss: 0.392403   Acc: 0.823532
Test set: Average loss: 0.3832, Accuracy: 0.8238
Train Epoch: 3 Loss: 0.299362   Acc: 0.874850
Test set: Average loss: 0.3504, Accuracy: 0.8465
Train Epoch: 4 Loss: 0.241642   Acc: 0.904203
Test set: Average loss: 0.3343, Accuracy: 0.8544
Train Epoch: 5 Loss: 0.188283   Acc: 0.929163
Test set: Average loss: 0.3730, Accuracy: 0.8574
```

提升不明显。还出现了更加严重的过拟合现象。

修改loss函数

```
Net(
  (embedding): Embedding(20001, 64)
  (lstm): LSTM(
    (cell): LSTMCell(
      (W): Linear(in_features=128, out_features=256, bias=True)
    )
  )
  (fc1): Linear(in_features=64, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=2, bias=True)
)
Train Epoch: 1 Loss: 0.395837   Acc: 0.629343
Train Epoch: 2 Loss: 0.239495   Acc: 0.779153
Train Epoch: 3 Loss: 0.175185   Acc: 0.837859
Train Epoch: 4 Loss: 0.140378   Acc: 0.868610
Train Epoch: 5 Loss: 0.119442   Acc: 0.886981
```

尝试其他与分类问题相关的loss函数

尝试BCELoss，显示模型大小不统一，调整target变为one-hot编码，成功。结果如下，仍然不如CrossEntropyLoss

```
Net(
  (embedding): Embedding(20001, 64)
  (lstm): LSTM(
    (cell): LSTMCell(
      (W): Linear(in_features=128, out_features=256, bias=True)
    )
  )
  (fc1): Linear(in_features=64, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=2, bias=True)
)
Train Epoch: 1 Loss: 0.695038   Acc: 0.501298
Test set: Average loss: 0.0088, Accuracy: 0.4939
Train Epoch: 2 Loss: 0.641600   Acc: 0.692592
Test set: Average loss: 0.0070, Accuracy: 0.7791
Train Epoch: 3 Loss: 0.583650   Acc: 0.833117
Test set: Average loss: 0.0071, Accuracy: 0.8216
Train Epoch: 4 Loss: 0.564992   Acc: 0.874401
Test set: Average loss: 0.0071, Accuracy: 0.8265
Train Epoch: 5 Loss: 0.553254   Acc: 0.899261
Test set: Average loss: 0.0071, Accuracy: 0.8319
```

修改其他参数

修改optimizer为Adamax,增大lr

```
Net(
  (embedding): Embedding(20001, 64)
  (lstm): LSTM(
    (cell): LSTMCell(
      (W): Linear(in_features=128, out_features=256, bias=True)
    )
  )
  (fc1): Linear(in_features=64, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=2, bias=True)
)
Train Epoch: 1 Loss: 0.613727   Acc: 0.642821
Test set: Average loss: 0.4866, Accuracy: 0.7807
Train Epoch: 2 Loss: 0.413506   Acc: 0.820337
Test set: Average loss: 0.3747, Accuracy: 0.8390
Train Epoch: 3 Loss: 0.355496   Acc: 0.851937
Test set: Average loss: 0.3654, Accuracy: 0.8455
Train Epoch: 4 Loss: 0.323862   Acc: 0.869060
Test set: Average loss: 0.3701, Accuracy: 0.8408
Train Epoch: 5 Loss: 0.297476   Acc: 0.878494
```

```
Test set: Average loss: 0.3499, Accuracy: 0.8519
```

收敛速度加快，但效果变差，符合预期。

增大batchsize

```
Net(
  (embedding): Embedding(20001, 64)
  (lstm): LSTM(
    (cell): LSTMCell(
      (W): Linear(in_features=128, out_features=256, bias=True)
    )
  )
  (fc1): Linear(in_features=64, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=2, bias=True)
)
Train Epoch: 1 Loss: 0.632096   Acc: 0.628284
Test set: Average loss: 0.5354, Accuracy: 0.7273
Train Epoch: 2 Loss: 0.457365   Acc: 0.786674
Test set: Average loss: 0.4302, Accuracy: 0.8078
Train Epoch: 3 Loss: 0.357231   Acc: 0.845989
Test set: Average loss: 0.3835, Accuracy: 0.8289
Train Epoch: 4 Loss: 0.293468   Acc: 0.879180
Test set: Average loss: 0.3408, Accuracy: 0.8527
Train Epoch: 5 Loss: 0.242343   Acc: 0.904608
Test set: Average loss: 0.3662, Accuracy: 0.8543
```

运行速度明显加快(1m23s->48s /5Epoch)，效果略有变差