语言模型实践报告

519030910383 张哲昊

1. 摘要

本次实践项目尝试了实例代码中提供的 GRU，LSTM，Transformer 三种不同的网络模型在给定的文本数据上进行语言模型任务。本次实践完成了不同模型之间性能的对比，不同超参数对于实验的影响探究（单个调参与贝叶斯优化整体调参），以及使用 tensorboard 画出训练阶段每个 epoch 的 train, valid 和 test loss 趋势。

2. 任务说明与模型结构

语言建模（LM）是使用各种统计和概率技术来确定一个句子中出现的特定单词序列的概率。而评价模型预测的好坏的指标为困惑度 (PPL). 通常认定测试集中的句子为模拟真实环境中的正常句子，因此在训练数据集上训练好的语言模型，计算在测试集中的正常句子的概率值越高说明语言模型的越好。在本次任务的设定之下，PPL 的值与负对数似然损失的值成指数关系，即：

$$PPL = \exp(-\frac{1}{N}\sum_{1}^{N}\log(w_i|h))$$

其中 h 为 Seq2Seq 模型的隐层。
本次实践使用了 3 种模型: GRU, LSTM, Transformer.

2.1. GRU 与 LSTM

循环神经网络（RNN），是一类允许过去的输出作为输入的神经网络，同时具有隐层。该模型结构非常适合处理序列数据以及自回归任务。GRU 和 LSTM 处理了传统 RNN 遇到的梯度消失的问题。相对于传统的 RNN，他们有更加精巧的记忆单元以及对于过去状态的处理方式。图1中展示了 LSTM 和 GRU 的架构。

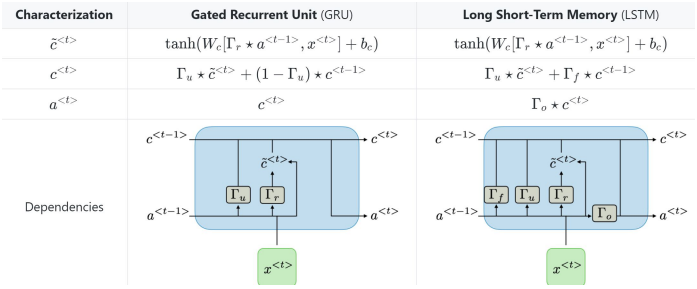

图 1: *LSTM* 于 *GRU* 的架构

2.2. Transformer

Transformer 利用自注意力机制对位置编码后的输入进行加权，并加入残差连接，层归一化以及全连接层。图2展示了它的结构。本次任务使用 Transformer 的编码器部分对文本进行建模。

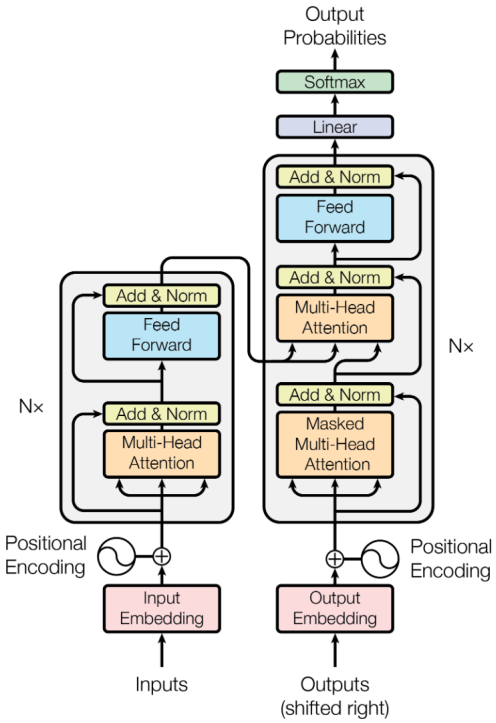

Figure 1: The Transformer - model architecture.

图 2: *Transformer* 架构

3. 网络结构的性能对比

在使用相同模型参数的情况下（emsize = 650, nhid = 650, nlayers = 2, lr = 20/5, clip = 0.2, epochs = 6, batch size = 20, dropout = 0.5 ）比较了三种模型的性能表现。如图2所示，LSTM 的性能最好，同时 tied LSTM 的模型参数也较少，二者性能均远超其他模型。Transformer 的效果较差。其原因可能是训练数据量较少。

表 1: 网络结构的性能对比

模型	参数量	测试集损失	PPL
Transformer	41.13M	5.49	242.47
tied LSTM	24.81M	**4.85**	**127.60**
LSTM	42.82M	4.86	128.62
GRU	41.13M	5.58	264.49
RNN_TANH	37.74M	6.62	752.94

4. 超参数调整

本次实验的超参数如下：

表 2: 超参数名称与意义

变量名	意义
emsize	词嵌入维度
nhid	隐层数（Transformer FNN 隐层维度）
nlayers	网络层数
lr	学习率
clip	梯度减切
batch_size	批数据大小
bptt	序列长度
dropout	丢弃率

在本次项目中，首先控制其他变量不变，调整单一超参数观察性能表现。然后利用贝叶斯优化整体调整多个关键参数，使得整体性能达到最优。

4.1. 批大小（batch size）

本次实验的参数更新方式为对每一批数据同时计算梯度并进行梯度下降，故批大小可能会对实验结果产生较大影响。控制其他超参数不变，用 LSTM 和 Transformer 进行不同批大小的实验。

从图3可以直观地发现，批大小对于两个模型的

表 3: 不同批大小对实验结果的影响

batch_size	Transformer PPL	LSTM PPL
20	235.39	164.26
40	201.43	156.56
80	183.02	157.03
120	193.55	163.68
200	195.77	170.92
300	206.54	181.46

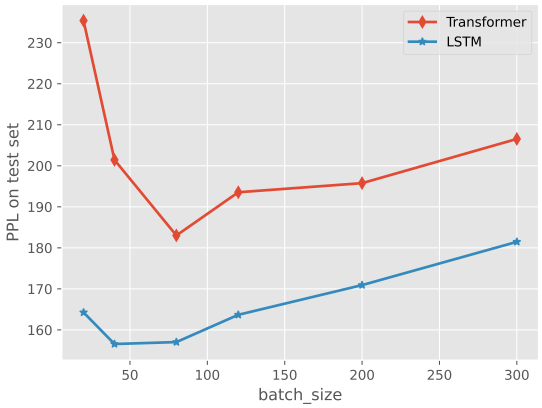

图 3: 不同批大小对实验结果的影响

PPL 呈现线下降后上升的趋势。同时发现对于 Transformer 而言，其最优的批大小大于 LSTM 最优的批大小。因此相对于 LSTM，我们应该用更大的批大小。

4.2. 丢弃率（Dropout）

Dropout 是指在深度学习网络的训练过程中，对于神经网络单元，按照一定的概率将其暂时从网络中丢弃，可以防止过拟合从而提高模型的泛化性。本次项目探究了丢弃率对于 LSTM 表现的影响。固定其他参数（-emsize 650 –nhid 650 –epochs 40），调整丢弃率得到的实验结果如表4所示。随着丢弃率的逐渐提高，PPL 呈现先下降后上升的趋势。

4.3. 序列长度

该超参数能够确定每次给模型输入的序列的长度以及其对应预测的长度。对于 LSTM 而言，过长的序列长度会使得模型的向和后向传播时间变

表 4: 不同丢弃率对实验结果的影响

Dropout	PPL
0.1	126.88
0.2	121.29
0.3	120.18
0.4	122.36

慢，同时可能存在较为严重的遗忘问题、而较短的序列长度可能会使得模型对于上下文的理解较为局限。前两种情况均可能导致模型的性能受到影响。本次实验固定其他参数（-emsize 1250 –nhid 1250 –dropout 0.3 –epochs 40 –tied –batch_size 40），单独调整序列长度，实验结果如表5所示。图4中先降后升的趋势基本符合之前的分析。

表 5: 不同序列长度对实验结果的影响

序列长度	PPL
10	121.33
20	116.67
30	116.44
35	115.61
50	116.37
70	116.75
100	118.32

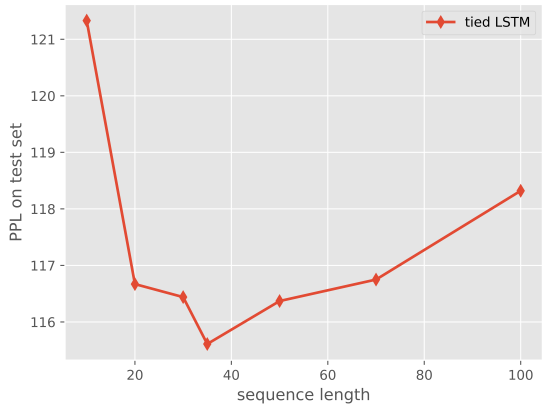

图 4: 不同序列长度对实验结果的影响

4.4. 词嵌入维度与隐层数量

固定其他参数（–dropout 0.3 –epochs 40 –tied）不变，调整 tied LSTM 的词嵌入维度与隐层数量，模型的性能表现如表6所示，可以发现当词嵌入维度与隐层数量增大时，模型性能提升，但是参数量也在增加。

表 6: 词嵌入维度与隐层数量对实验结果的影响

emsize nhid	参数量	PPL
200	11.75M	142.14
650	24.81M	120.18
1024	45.20M	116.02
1250	59.60M	115.61

4.5. 贝叶斯优化整体调参

由于上述的调参方式只能够控制每个超参数的变换，保证其他超参数不变，从而探究该超参对实验结果的影响，而不能高效地找出全局最优地超参数。而贝叶斯优化能够将整个黑盒模型建模为高斯过程，通过优化较易优化的 acquisition function 来找到使得函数值较大的数据点，通过新得到的数据点和对应函数值来降低黑盒函数的不确定性。具体算法如下：

具体实现上，使用 BayesianOptimization 库，同

Algorithm 1 Bayesian optimization

1: **for** $n = 1, 2, \ldots$ **do**
2: select new $\mathbf{x}_{n+1}$ by optimizing acquisition function α

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{D}_n)$$

3: query objective function to obtain y_{n+1}
4: augment data $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\mathbf{x}_{n+1}, y_{n+1})\}$
5: update statistical model
6: **end for**

时将模型训练，验证的过程均封装为函数，整体封装为一个黑盒模型（输入为超参数，输出为该超参数设定下模型训练之后的测试效果）。整体代码框架如下：

```
#选择优化clip,dropout,lr这三个参数
def train_eval(clip,dropout,lr):
    decrease_rate = 4
    if args.model == 'Transformer':
        model_1 = model.TransformerModel(ntokens,
    args.emsize, args.nhead, args.nhid, args.
    nlayers, dropout).to(device)
    else:
```

```
        model_1 = model.RNNModel(args.model, ntokens, args.emsize, args.nhid, args.nlayers, args.dropout, args.tied).to(device)
    bptt = 35
    trained_model = train_all(model_1,bptt,clip, decrease_rate,lr)
    test_loss = evaluate(trained_model,test_data)
    return -math.exp(test_loss)
pbounds= {
         'clip': (0.18, 0.23),
         'dropout': (0.37, 0.45),
         'lr': (18, 22)}

optimizer = BayesianOptimization(
        f=train_eval,
        pbounds=pbounds,
        verbose=2,
        random_state=1,
    )
optimizer.maximize(
        init_points=30,  #执行随机搜索的步数
        n_iter=40,   #执行贝叶斯优化的步数
    )
```

通过贝叶斯优化，可以同时优化多个参数从而达到最优的超参数配置。

5. Tensorboard 可视化

在本次项目中学习使用了 Tensorboard 进行可视化，记录了每个 epoch 训练和验证的 PPL 并进行可视化。基本代码如下：

```
from torch.utils.tensorboard import SummaryWriter
writer = SummaryWriter('log')
for epoch in range(1, args.epochs+1):
    ...
    writer.add_scalar(tag="loss/train", scalar_value=math.exp(cur_loss),global_step=epoch)
    writer.add_scalar(tag="loss/vali", scalar_value=math.exp(val_loss),global_step=epoch)
```

6. 最终实验结果与可视化

在超参数为 –emsize 1250 –nhid 1250 –dropout 0.4 –epochs 40 –tied –clip 0.2 –bptt 35 –seed 1111 的实验设定之下能够在测试集上取得最好的实验效果，如表所示7. 模型在训练集和验证集上的 loss 变换曲线如图5和图6所示。

表 7: 最优的实验效果

参数量	loss	PPL
59.69M	4.72	112.54

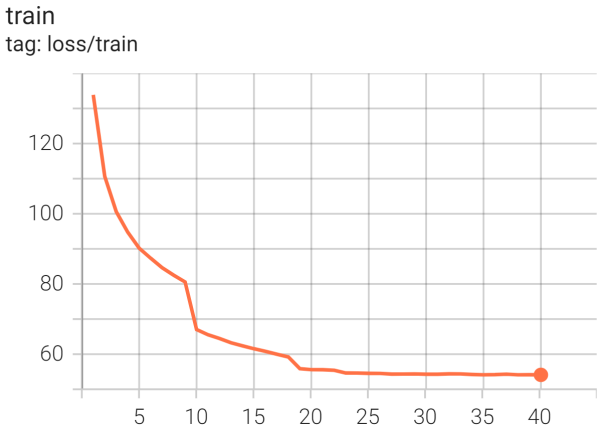

图 5: 最优超参数配置下在训练集上 *loss* 的变化

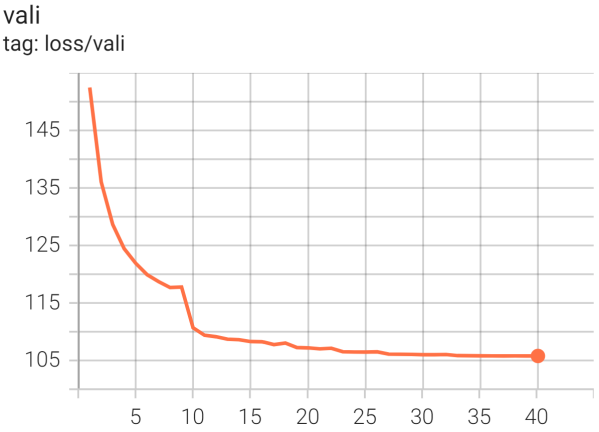

图 6: 最优超参数配置下在验证集上 *loss* 的变化