PRML assignment 2 report

准备

本 assignment 使用 tensorflow2，版本 2.2.0rc3

Part I

在第一部分我们设法使得 tf_main()能够运行出正确的结果。模型的 Layers 已经定义好了，我们需要完善 call 函数 (tf.keras.Model | TensorFlow Core v2.1.0)

从 train 中可以看到模型的输入是(x,y),形状都是（None, None）,第一个 None 是 mini batch size,第二个是 timesteps,由于数据生成的原因必定都是 maxlen=11

于是均经过 embedding 层，每一个 0-9 的 int32 映射到 32 维向量。

参数数量很简单就是 10*32

然后是 concat_layer.因为直接是个 tf.function 所以就不用修改__init__部分了。没有参数

然后是 rnn 层，rnn 单元有记忆地把 2*32 维映射到 64 维，由于 return_sequences=True 所以输出的形状还是（batch_size, timesteps, 64）

这一层的参数数量应该是 64*64+64*64+64

然后到一个 dense 输出，64*10+10 参数。需要 predict 的时候就把 10 维经过一个 argmax 作为该位的输出。

其他所有部分保持不变，测试数据集大小提升到 20000,结果 accuracy=0.99995,1/20000 的错误率。考虑到 3k 步之后 loss 仍然在一定程度的下降，错误率应该可以进一步下降。

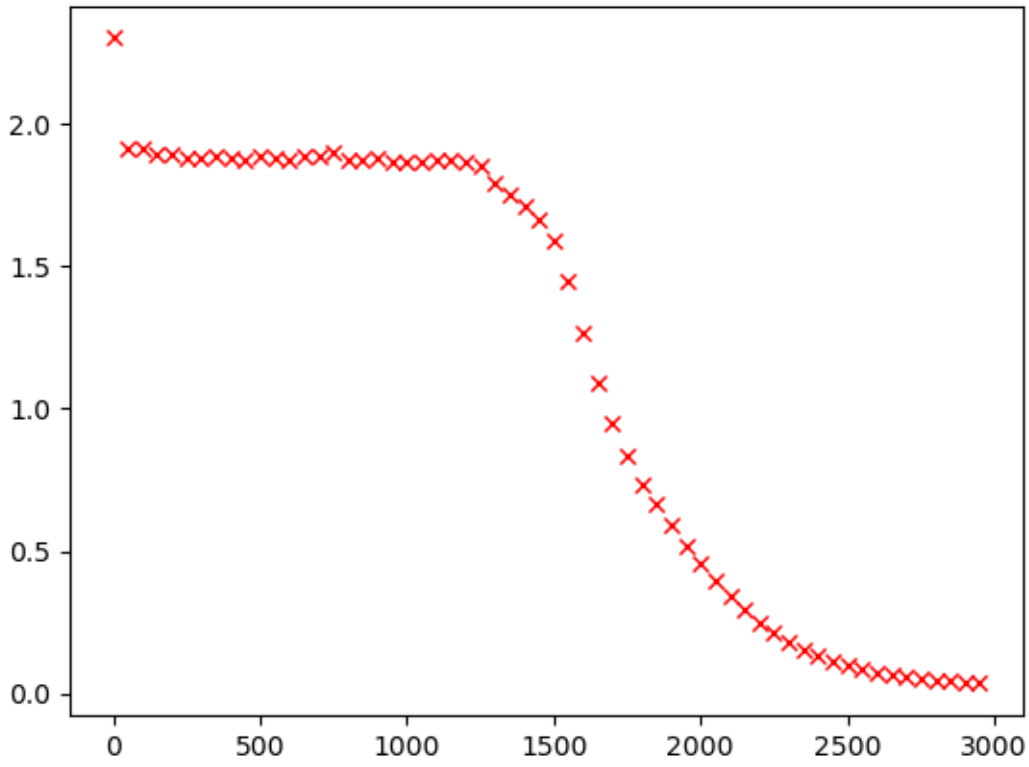

Figure 1 模型 loss 随 epoch

Part II

第一部分中的模型解决了加法问题，按照我的理解是由于没有长程依赖等问题（加法本来就与远处的数位关系不大），并且也不需要记住之前所有状态，事实上 rnn 的输出向量作为下一次数位的输入只需要包含是否进位的一位信息，且由于 return_sequences=True，模型本身就包含了加法的特征，可以想见增加运算的位数也没法使得模型的表现有显著变差。

虽然模型已经很强但是我有个疑问。

测试一

其实我一开始是 32 维数据进入 rnn，然后输出 64*2 拼起来进入 dense，但是正确率一直是 0.应该是因为两个 64 维向量只有在最后一层 Dense 才能传递信息，并且这层也是最后一层没有激活函数。

```
        self.dense1 = layers.Dense(64, activation='relu')
        self.dense2 = layers.Dense(32, activation='relu')
        self.dense3 = layers.Dense(10)
```

先经过 rnn，再经过 concatenate，再经过上述的三层 dense，发现训练起来比第一部分的模型快了很多（原来的模型在前 1k5 步左右的 loss 一直没能有什么突破），正确率为 0.9425

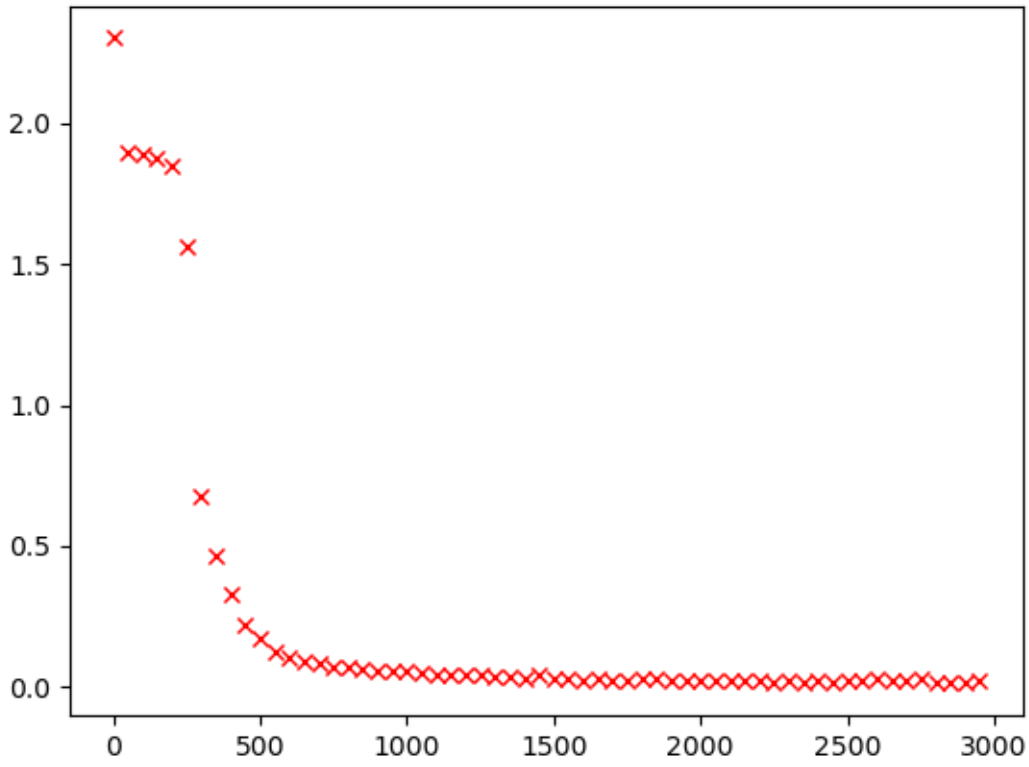

Figure 2 三层 dense，loss 随 epoch

考虑到 rnn 自己能做到的，dense 应该也可以做到，继续增加 dense 强度后我发现正确率没有什么改善。我觉得原因应该是 Part I 中的模型 rnn 的输出只需要包含当前位以及之前的位有没有进位。但是如果是先 rnn 再 concatenate 那么就相当于要记住之前所有位的信息来综合另一半 rnn 的输出给出一个是否进位的信息。

将 RNN 换成 lstm 层也没有什么帮助，正确率 0.9165

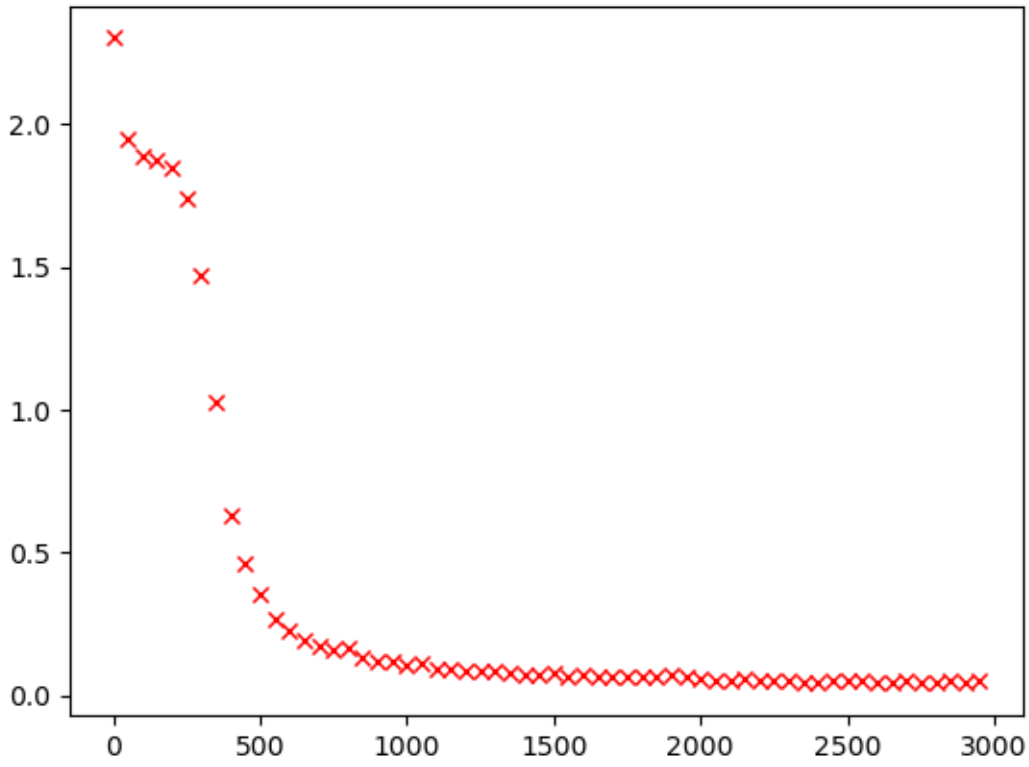

Figure 3 lstm, loss 随 epoch

测试二

考虑到每一个 batch 都是新生成的数据，没有过拟合的风险，并且原来的训练有 1k 左右步没啥进步，我觉得可以调整学习率看看效果。

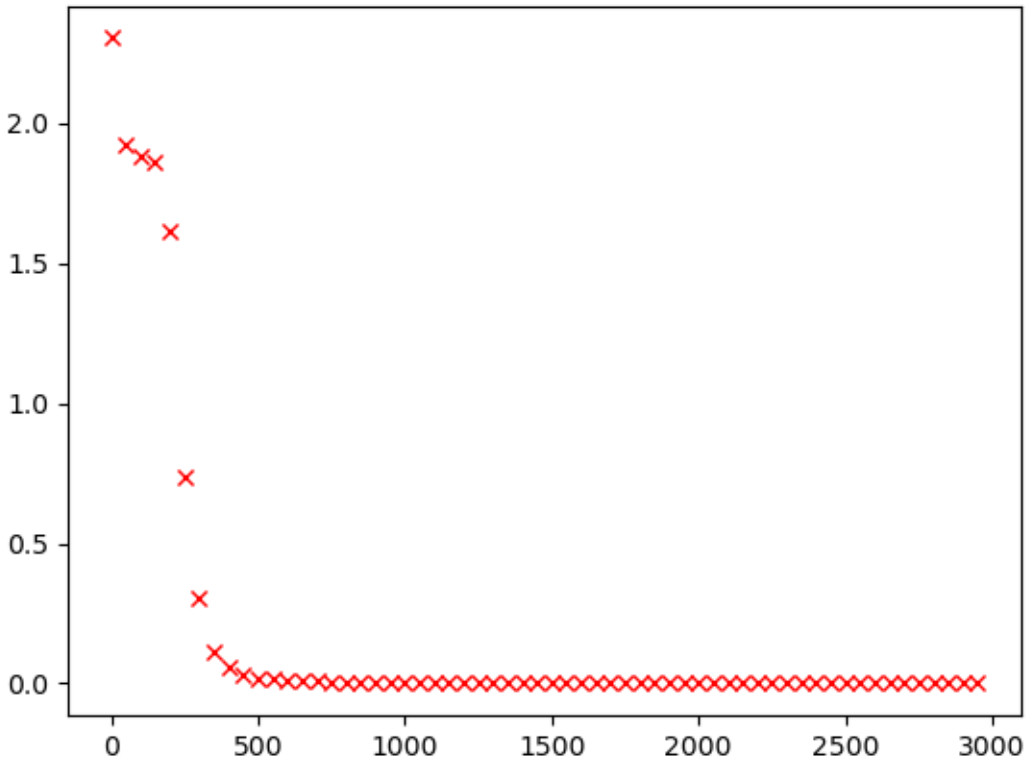

Figure 4 adam optimizer, learning rate 0.01

但是过高模型就会不稳定：

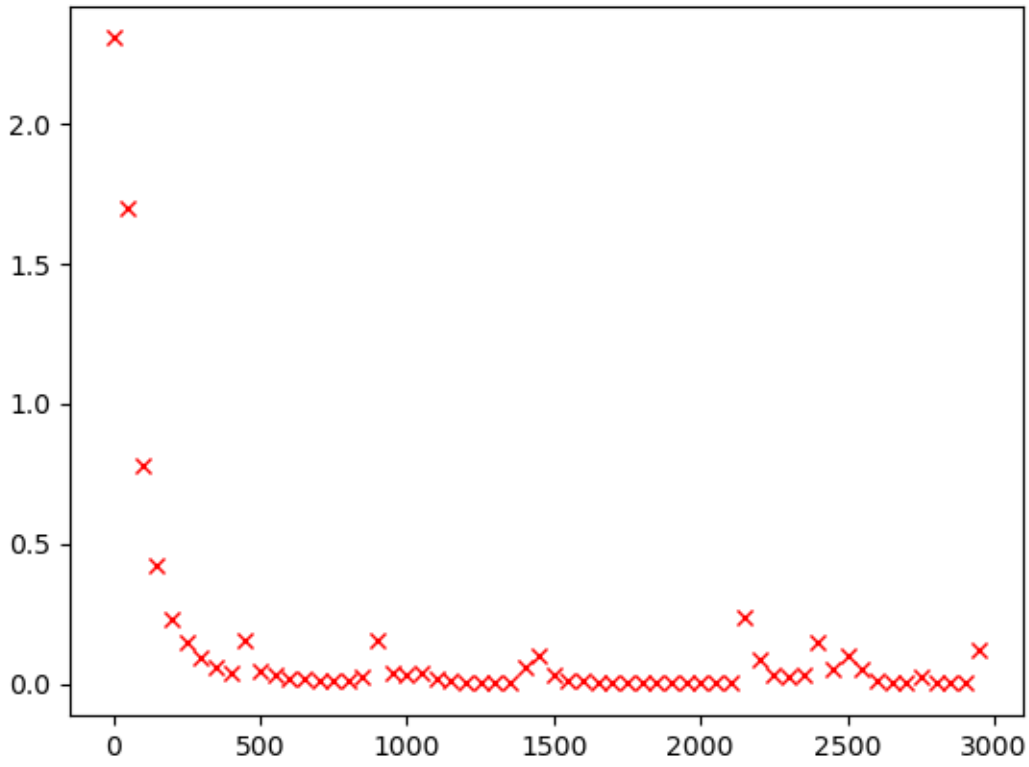

Figure 5 adam optimizer, learning rate 0.05

运行代码

python3 source.py

来运行代码

References

Google. (2020, April 29). *tf.keras.Model | TensorFlow Core v2.1.0*. Retrieved from TensorFlow API: https://www.tensorflow.org/api_docs/python/tf/keras/Model#used-in-the-notebooks_1

Bibliography

TensorFlow Guide on rnn, https://www.tensorflow.org/guide/keras/rnn