哈尔滨工业大学

<<模式识别与深度学习>>

实验 4 经典循环神经网络实现

(2020 春季学期)

学院：计算机科学与技术

学号：1170300909

姓名：武磊

指导老师：左旺孟

日期：2020.5.14

一、实验目的

1. 基于 Pytorch 实现 simpleRNN， LSTM 等循环神经网络结构
   1) 对给定的 sine 函数进行预测
   2) 基于影评数据进行文本情感预测

二、实验环境配置

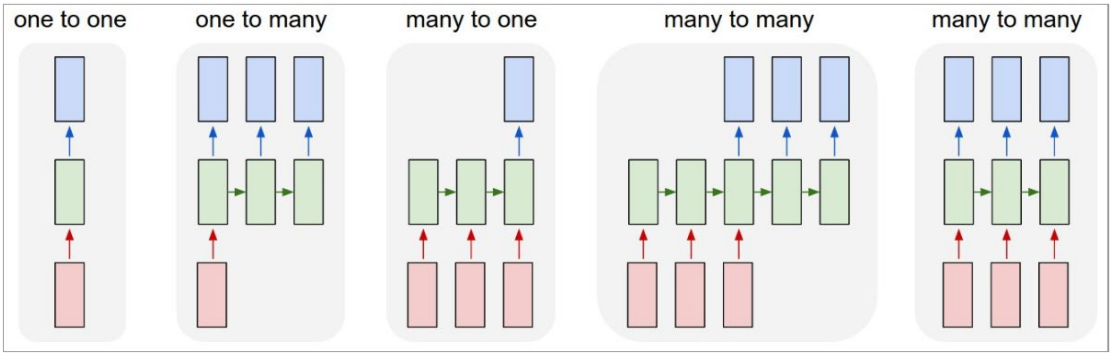
1. 硬件：
   CPU ： Inter Core i7-7500U 2.70GHz
   GPU ： NVIDIA GeForce 940MX
2. 软件：
   操作系统:WIN10
   软件：
   Python 3.6 (基于 Anaconda)
   Pytorch 1.2.0
   CUDA 10.1 V10.1.105
   cudnn v7.6.3 for CUDA 10.1
3. 开发 IDE
   Jupyter NoteBook
   Pycharm

三、实验过程

1. 选择代价函数，超参数，优化器
   1) 模型的超参数
      模型主要的超参数主要有以下因素：
      1. 学习率
      2. Batch size
      3. 网络结构，主要要有网络层数，网络结构设计等等
   2) 代价函数选择
      Sine 函数预测实际上是类似回归问题，使用 MSE。
      影评情感预测是分类问题，使用交叉熵作为代价函数
   3) 优化器选择
      主要选用 Aadm 和 SGDM 作为优化器。在实验过程中做对比实验。
      本次实验还会尝试使用 LBFGS 优化器

2. 设计定义网络
   本次实验中主要实现两类网络，一种是简单的 RNN 循环神经网络，将前一步的激活值作为下一层的输入，另一种是 LSTM（Long short-term Memory Net）。
   循环神经网络主要考虑的是如何对时序序列建模，即$x^{<t-1>}$

的输出对$y^{<t>}$产生的影响。一般对时序数据建模时可能碰到的业务场景有以下几种：



本次实验中 sin 函数预测就是 many-to-many 的模型，而影评数据情感分类则是 many-to-one 的模型。

本次实验中涉及的到的网络全部在 nets 中定义

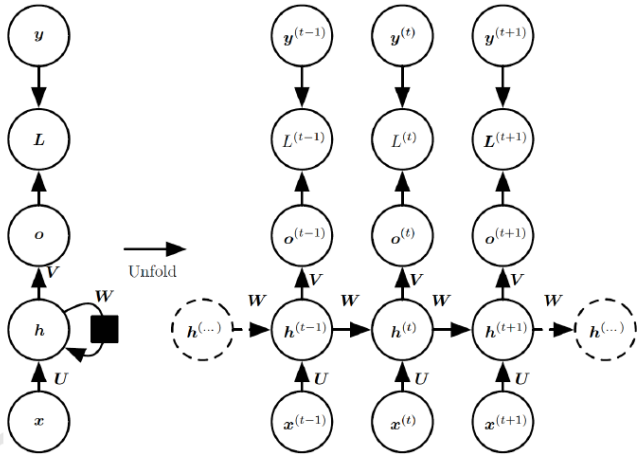
1) RNN

由 RNNCell 和 BasicRNN 组成。

本次实现的 BasicRNN 由 RNNCell 和全连接层组成。RNNCell 作为隐藏层之间存在循环连接，$x^{<t-1>}$输入到隐藏层得到的激活值$h^{<t-1>}$，作为输入的一部分，在下一个 step 和$x^{<t>}$一起输入到网络中。

a) RNNCel 设计

- RNNCel 示意图：



- 前向过程计算如下：
  - $h_t = g(Uh_{t-1} + Wx_t + b_h)$
  - $y_t = g(W_y h_t + b_y)$

b) 全连接

作为 hidden layer 到 output 直接的映射，可以视任务复杂程度叠加一层或多层全连接甚至卷积神经层。

本次实验 sine 函数预测任务简单，只需要单层全连接输出即可获得很好的效果。

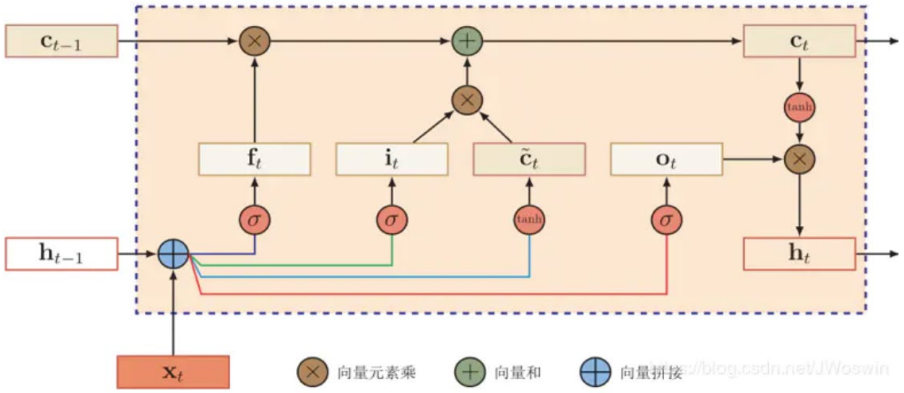
2) LSTM

和 RNN 结构很相似，主要区别在于 LSTMCell 的设计和网络前向的过程。

a) LSTMCell

下图是 LSTM 的示意图。LSTM（Long short-term Memory）长-短期记忆网络的提出主要是为了解决长序列数据的依赖问题，short-term 是作为一个 step 处理的数据。LSTM 通过引入门控机制，Forget-gate,Update-gate, Output-gate,来决定是否遗忘、更新、输出之前的数据流信息。

- LSTMCell 示意图：



LSTM

- 前向计算过程

  - 输入门: $i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$
  - 遗忘门: $f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$
  - 输出门: $o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$
  - $\hat{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c)$
  - $c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$
  - $h_t = o_t \odot tanh(c_t)$

b) 全连接层设计

采用两层全连接输出。

其实着了的全连接层输出可以有多重尝试，比如添加不同类型的激活函数等等。

四、结果分析

1. Sine 预测结果分析

Sine 预测实际上是回归问题，生成的数据维度是 100*1000，即一共是 100 个样本，每个样本的序列长度是 1000。

• 数据处理

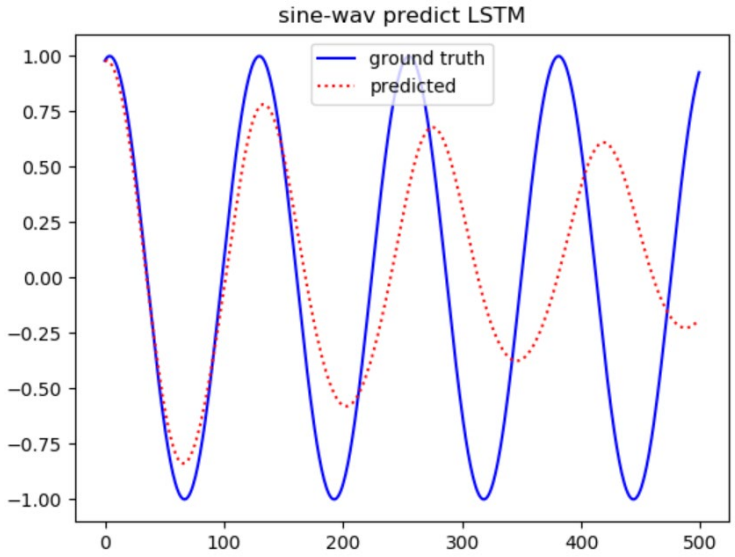
训练集：选取的前 98 个样本

测试集：选取后 2 个样本，使用序列的前 500 个点，预测后 500 个点

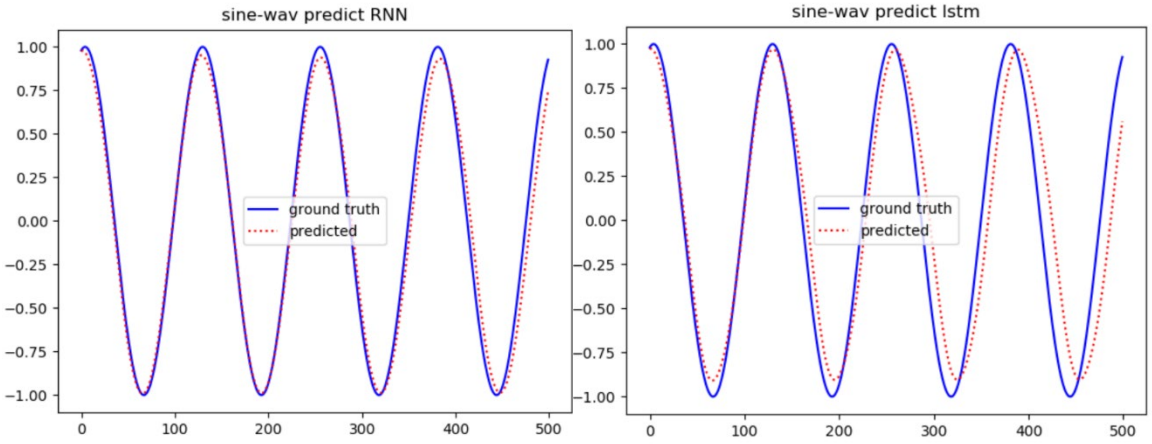
• 优化器选择

测试了两种优化器 Adam 和 LBFGS（拟牛顿法）

在测试过程中发现，由于是回归问题，所以基于 LBFGS 的优化器收敛较快速，Adam 收敛教慢。



上图是 train_loss = 6.05e-5，得到的图像，使用 lstm 网络，可以看到误差逐渐变大。

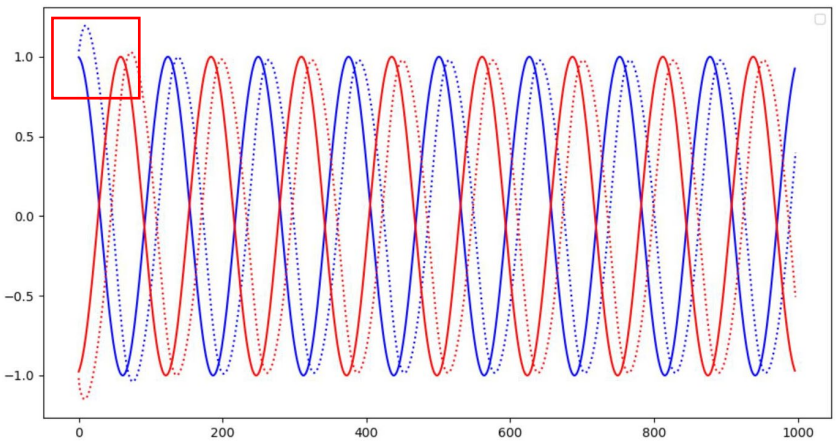


上图是使用 rnn 网络 train_loss = 2.22e-5，lstm 网络 train_loss = 7.23e-6 得到的图像，，可以看到由于之前值的累积误差，后续预测和 ground truth 差值逐渐变大，但是总体效果较好，不会存在学飞的现象。

Rnn 网络的效果要好于 lstm，原因可能是由于 lstm 网络较为复杂，对于输入的扰动更敏感。

一些想法：之前觉得双向循环神经网络对于 sin 预测效果意义不大，但是训练过程中出现了下列现象：我将测试集的第一个点放入 rnn 中预测后续 999 个点，出现类似“延迟”现象。观察方框中的点，在正弦图像的波峰位置的点应该是下降，但是被预测为了上升，导致误差。这可以通过增大 epoch 学习，但是也可以通过双向网络，后面的点帮助预测。

但是双向网络的难点在于需要输入序列是定长的，这是需要克服的问题。



2. 影评情感分析

• 数据处理

影评数据是变长序列，需要用 0 进行 padding 操作。发现 neg_rev 中最长序列为 51，pos_rev 序列最长为 63。词向量嵌入过程中全部填充为长度 51 的序列，超过截断，不足填充。

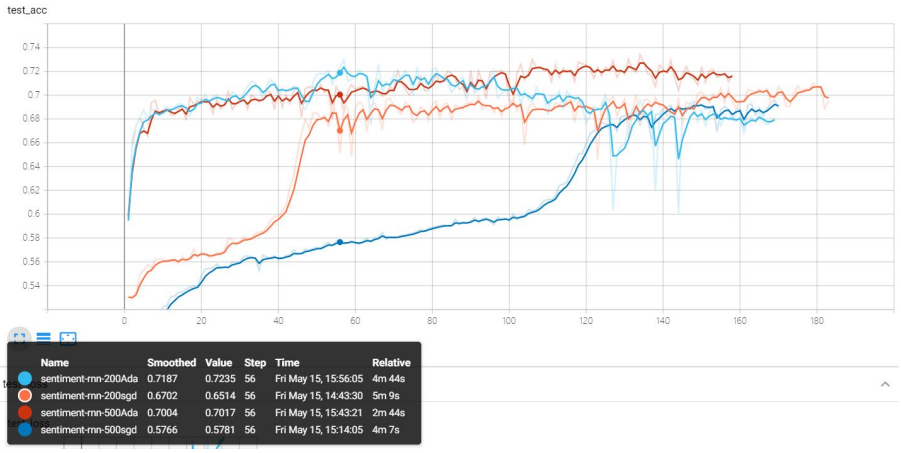
训练集从 pos 和 neg 中各截取 4000 条，其余的作为测试集。

• 优化器

情感分类实际上是分类问题，选择交叉熵函数。优化器使用 Adam 和 SGDM。

1) 不同 batch 和优化器比较

网络	Epoch	Batchsize	Lr	优化器	Accuracy
RNN	200	200	0.001	SGDM	71.3%
RNN	200	500	0.001	SGDM	70.5%
RNN	200	200	0.001	Adam	72.5%
RNN	200	500	0.001	Adam	73.4%



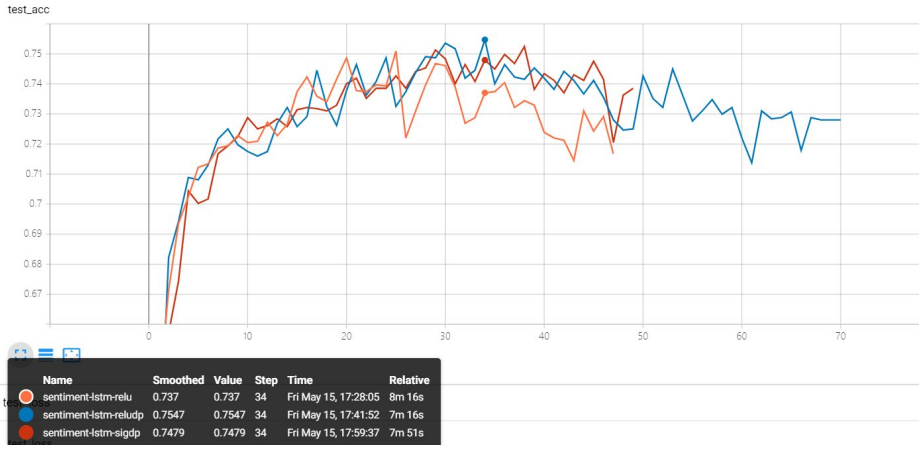
可以看到有下列现象

• 小 batch 前期收敛快，准确率提升快，但是容易出现过拟合现象，不够稳定，后期无法稳定收敛。

• 似乎 Adam 对于 rnn 网络的效果更好一些，因为可以预见 rnn 时序网络，由于时序的维度，使得 loss curve 更加崎岖，这时候使用 SGDM 就不能够利用其随机性的优势。

Lstm 训练效果：

在全连接层分别尝试了 sigmoid，Relu 激活函数，观察到了过拟合现象，希望添加 dropout 缓解，但是效果不明显。



五、总结

1、 循环神经网络的训练难度较大，经常会出现梯度消失和梯度爆炸的现象。这也是 cell 中添加各种不同的 tanh，sigmoid 函数的原因，有效的保留的信息，减少了训练的困难程度。

2、 在情感分类任务中需要对词向量序列做填充，合适的填充方式有助于训练。

3、 还有双向和深度的循环网络没有实现。但是对于循环神经网络的用途和理解更深一步。其实循环神经网络也可以理解为提取时序特征的一种手段，一种 feature extractor，可以将提取的时序特征传递给后面的深层网络做分类和预测，生成等任务（暂时的理解）。

六、参考

【1】 循环神经网络 pytorch 实现 https://www.jianshu.com/p/6575d54f778f

【2】 https://karpathy.github.io/2015/05/21/rnn-effectiveness/

【3】 吴恩达深度学习 https://www.bilibili.com/video/BV1F4411y7BA?p=11