作业三报告

Part 1

1. 记 diag(x)表示对角线元素组成的向量为 x 的对角矩阵

$$\frac{\partial \mathrm{h}_t}{\partial o_t} = diag(\tanh(C_t))$$

$$\frac{\partial \mathrm{h}_t}{\partial C_t} = \mathrm{diag}(o_t * (1 - \tanh^2(C_t)))$$

$$\frac{\partial \mathrm{h}_t}{\partial C_{t-1}} = \frac{\partial \mathrm{h}_t}{\partial C_t}\frac{\partial \mathrm{C}_t}{\partial C_{t-1}} = diag(o_t * (1 - \tanh^2(C_t)) \ * f_t)$$

$$\frac{\partial \mathrm{h}_t}{\partial \bar{C}_t} = \frac{\partial \mathrm{h}_t}{\partial C_t}\frac{\partial \mathrm{C}_t}{\partial \bar{C}_t} = diag(o_t * (1 - \tanh^2(C_t)) \ * i_t)$$

$$\frac{\partial \mathrm{h}_t}{\partial i_t} = \frac{\partial \mathrm{h}_t}{\partial C_t}\frac{\partial \mathrm{C}_t}{\partial i_t} = diag(o_t * (1 - \tanh^2(C_t)) \ * \bar{C}_t)$$

$$\frac{\partial \mathrm{h}_t}{\partial f_t} = \frac{\partial \mathrm{h}_t}{\partial C_t}\frac{\partial \mathrm{C}_t}{\partial f_t} = diag(o_t * (1 - \tanh^2(C_t)) \ * C_{t-1})$$

为了方便表述，以下记：

$$W_f z = W_{fh} h_{t-1} + W_{fx} x_t$$

$$W_i z = W_{ih} h_{t-1} + W_{ix} x_t$$

$$W_C z = W_{Ch} h_{t-1} + W_{Cx} x_t$$

$$W_o z = W_{oh} h_{t-1} + W_{ox} x_t$$

根据全导数公式：

$$\frac{\partial \mathrm{h}_t}{\partial x_t} = \frac{\partial \mathrm{h}_t}{\partial o_t}\frac{\partial o_t}{\partial x_t} + \frac{\partial \mathrm{h}_t}{\partial C_t}\frac{\partial \mathrm{C}_t}{\partial x_t} = \frac{\partial \mathrm{h}_t}{\partial o_t}\frac{\partial o_t}{\partial x_t} + \frac{\partial \mathrm{h}_t}{\partial C_t}\frac{\partial \mathrm{C}_t}{\partial f_t}\frac{\partial \mathrm{f}_t}{\partial x_t} + \frac{\partial \mathrm{h}_t}{\partial C_t}\frac{\partial \mathrm{C}_t}{\partial i_t}\frac{\partial i_t}{\partial x_t} + \frac{\partial \mathrm{h}_t}{\partial C_t}\frac{\partial \mathrm{C}_t}{\partial \bar{C}_t}\frac{\partial \bar{C}_t}{\partial x_t}$$

$$= \frac{\partial \mathrm{h}_t}{\partial o_t}\frac{\partial o_t}{\partial x_t} + \frac{\partial \mathrm{h}_t}{\partial f_t}\frac{\partial \mathrm{f}_t}{\partial x_t} + \frac{\partial \mathrm{h}_t}{\partial i_t}\frac{\partial i_t}{\partial x_t} + \frac{\partial \mathrm{h}_t}{\partial \bar{C}_t}\frac{\partial \bar{C}_t}{\partial x_t}$$

其中：

$$\frac{\partial o_t}{\partial x_t} = diag\big(o_t * (1 - o_t)\big) W_{ox}$$

$$\frac{\partial f_t}{\partial x_t} = diag\big(f_t * (1 - f_t)\big) W_{fx}$$

$$\frac{\partial i_t}{\partial x_t} = diag\big(i_t * (1 - i_t)\big) W_{ix}$$

$$\frac{\partial \bar{C}_t}{\partial x_t} = diag\left(\left(1 - \bar{C}_t^{\ 2}\right)\right) W_{Cx}$$

所以：

$$\frac{\partial \mathrm{h}_t}{\partial x_t} = diag\big(\tanh(C_t) * o_t * (1-o_t)\big)W_{ox}$$

$$+ diag\left(o_t * \left(1-\tanh^2(C_t)\right) * C_{t-1} * f_t * (1-f_t)\right)W_{fx}$$

$$+ diag\left(o_t * \left(1-\tanh^2(C_t)\right) * \bar{C}_t * i_t * (1-i_t)\right)W_{ix}$$

$$+ diag\left(o_t * \left(1-\tanh^2(C_t)\right) * i_t * \left(1-\bar{C}_t{}^2\right)\right)W_{Cx}$$

同理可得：

$$\frac{\partial \mathrm{h}_t}{\partial \mathrm{h}_{t-1}} = diag\big(\tanh(C_t) * o_t * (1-o_t)\big)W_{oh}$$

$$+ diag\left(o_t * \left(1-\tanh^2(C_t)\right) * C_{t-1} * f_t * (1-f_t)\right)W_{fh}$$

$$+ diag\left(o_t * \left(1-\tanh^2(C_t)\right) * \bar{C}_t * i_t * (1-i_t)\right)W_{ih}$$

$$+ diag\left(o_t * \left(1-\tanh^2(C_t)\right) * i_t * \left(1-\bar{C}_t{}^2\right)\right)W_{Ch}$$

$$\frac{\partial \mathrm{h}_t}{\partial \mathrm{W}_f} = \frac{\partial \mathrm{h}_t}{\partial \mathrm{f}_t}\frac{\partial f_t}{\partial \mathrm{W}_f} = diag\left(o_t * \left(1-\tanh^2(C_t)\right) * C_{t-1} * f_t * (1-f_t)\right)z^T$$

$$\frac{\partial \mathrm{h}_t}{\partial \mathrm{W}_i} = \frac{\partial \mathrm{h}_t}{\partial \mathrm{i}_t}\frac{\partial i_t}{\partial \mathrm{W}_i} = diag\left(o_t * \left(1-\tanh^2(C_t)\right) * \bar{C}_t * i_t * (1-i_t)\right)z^T$$

$$\frac{\partial \mathrm{h}_t}{\partial \mathrm{W}_c} = \frac{\partial \mathrm{h}_t}{\partial \bar{C}_t}\frac{\partial \bar{C}_t}{\partial \mathrm{W}_c} = diag\left(o_t * \left(1-\tanh^2(C_t)\right) * i_t * \left(1-\bar{C}_t{}^2\right)\right)z^T$$

$$\frac{\partial \mathrm{h}_t}{\partial \mathrm{W}_o} = \frac{\partial \mathrm{h}_t}{\partial o_t}\frac{\partial o_t}{\partial \mathrm{W}_o} = diag\big(\tanh(C_t) * o_t * (1-o_t)\big)z^T$$

$$\frac{\partial \mathrm{h}_t}{\partial \mathrm{b}_f} = \frac{\partial \mathrm{h}_t}{\partial \mathrm{f}_t}\frac{\partial f_t}{\partial \mathrm{b}_f} = diag\left(o_t * \left(1-\tanh^2(C_t)\right) * C_{t-1} * f_t * (1-f_t)\right)$$

$$\frac{\partial \mathrm{h}_t}{\partial \mathrm{b}_i} = \frac{\partial \mathrm{h}_t}{\partial \mathrm{i}_t}\frac{\partial i_t}{\partial b_i} = diag\left(o_t * \left(1-\tanh^2(C_t)\right) * \bar{C}_t * i_t * (1-i_t)\right)$$

$$\frac{\partial \mathrm{h}_t}{\partial \mathrm{b}_c} = \frac{\partial \mathrm{h}_t}{\partial \bar{C}_t}\frac{\partial \bar{C}_t}{\partial \mathrm{b}_c} = diag\left(o_t * \left(1-\tanh^2(C_t)\right) * i_t * \left(1-\bar{C}_t{}^2\right)\right)$$

$$\frac{\partial \mathrm{h}_t}{\partial \mathrm{b}_o} = \frac{\partial \mathrm{h}_t}{\partial o_t}\frac{\partial o_t}{\partial \mathrm{b}_o} = diag\big(\tanh(C_t) * o_t * (1-o_t)\big)$$

2．定义误差为：

$$\delta_{t-1} = \frac{\partial \mathrm{O}}{\partial h_{t-1}} = \delta_t \frac{\partial \mathrm{h}_t}{\partial \mathrm{h}_{t-1}}$$

根据链式法则，可将误差传播到任意之前时刻 j<n，有：

$$\delta_j = \delta_n \prod_{k=j+1}^{n} \frac{\partial \mathrm{h}_k}{\partial \mathrm{h}_{k-1}}$$

对于权重梯度的计算，在 RNN 中权重的梯度为各个时刻权重梯度的和，对应累加梯度即可。（注意需要将如 W_f 分成 W_{fx},W_{fh} 分别更新梯度）

Part 2

Requirement 1：

embedding 层不能全 0 初始化，因为 embedding 是期望表示字向量。全 0 初始化等价于使将不同的字当成了同一个字。采用反向传播优化的神经网络都不能使用全 0 初始化，这样所有的单元都是一样的输出、传递相同的误差，因此就失去了网络学习特征的意义。

embedding 和 LSTM 一般情况下使用 uniform 的随机初始化即可。更进一步，embedding 可以采用经 word2vec、glove 等技术预训练好的 pretrained embedding 进行初始化。神经网络参数也可以经过 Xavier 方法使每一层输出的方差应该尽量相等，以便网络中信息能更好的流动。

在官方提供的 LSTM 中，其 weight 和 bias 都是通过 uniform($\frac{-1}{\sqrt{hidden\_size},}$,$\frac{1}{\sqrt{hidden\_size},}$)初始化。

Requirement 2：

在这次作业中，首先我采用了简体版本的全唐诗[1]作为数据集生成古诗。每首诗最长不超过 80 个字符，超出部分截去，不足的填补特殊符号。在实验中我使用了下述四种特殊符号：

<sos> 放在每首诗开头，表示一首诗的开始

<eos> 放在每首诗的结尾，表示一首诗结束

<unk> 表示这个字没有出现在词表里

<pad> 填充符

经过处理，一共提取了 57000 首诗，按照 4:1 划分训练集和验证集，字表大小|V|=7585

古诗生成的模型结构大致为：Embedding->LSTM->Dropout->Linear，在输出层前加入Dropout 主要是为了减轻过拟合程度。其中一些超参数设置如下：

---

[1] https://github.com/chinese-poetry/chinese-poetry-zhCN

dropout rate	kearning rate	batch size	epochs
0.2	1e-3	128	100

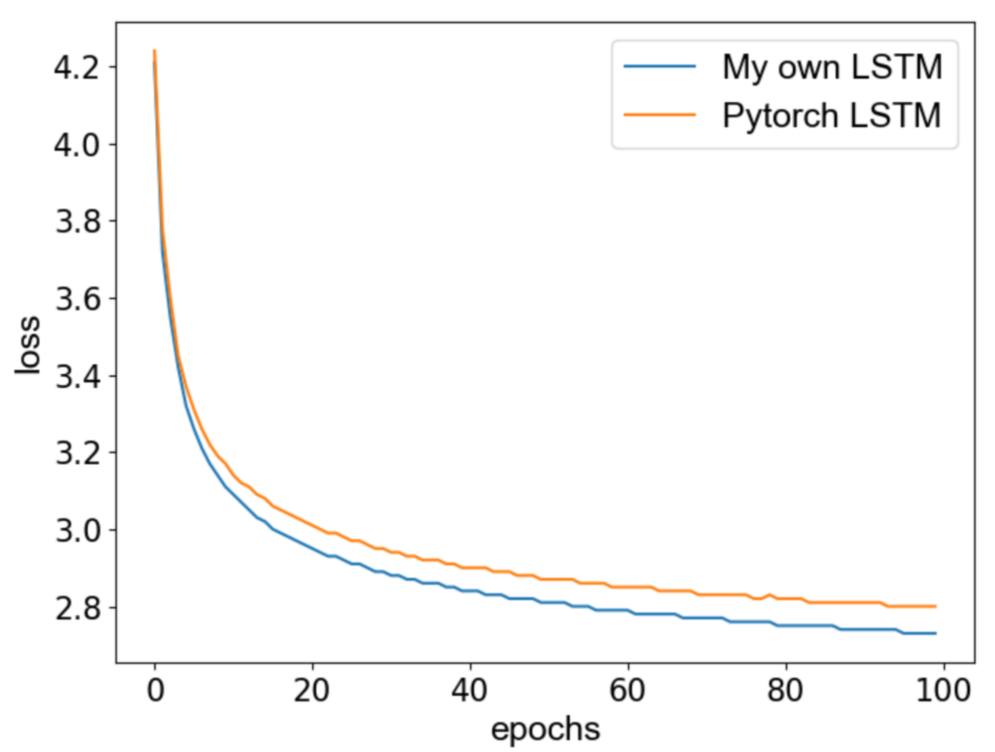
我使用 Adam 训练，如果 10 个 epoch 内在验证集的 ppl 不再下降，则**终止训练**。根据困惑度的公式可以看出，对一个句子，困惑度的对数即为交叉熵，因此最小化平均交叉熵接近等价于最小化困惑度的几何平均值。

在实验中，我实现了**多层 LSTM** 并尝试了以下不同的模型参数进行训练：

embedding dim	hidden size	layers	ppl
128	256	1	17.67
128	256	2	17.73
100*	256	1	17.57

其中 embedding_dim=100 的实验使用了在 Chinese Wikipedia corpus 上使用 word2vec 预训练的 embedding。根据观察，多层 LSTM 和预训练的 embedding 并没有能够明显的改善困惑度。

为了验证自己写的 LSTM 正确，我调用了 pytorch 封装好的 LSTM 单元进行对比试验，当 layers=1,hidden_size=256, embedding_dim=128,得到的 loss 下降图结果基本一致，因此大致上是没有问题的。



最终，我选取模型生成的一些古诗结果如下：

日月华亭上，凉风起一卮。春风吹竹露，远水万花风。

日暮神仙窟，鸟鸣天上人。幽人不可见，白日过层墀。
红竹香尘染落衣，水光初起绿萝烟。尽抛宾客金童稺，一共黄金粉署郎。
红尘不省俗，寻我莫论兵。见有三千里，无人共五兵。
红粉玉屏终，新花怨已成。开花多带雨，野蝶酒杯声。彩仗花开水，闲眠入海游。寒声催不见，愁与白云期。
山水初相问，茅茨亦亦归。人生老僧迹，坐久闭门钟。石壁青云映，山花石屋深。不知将出过，还不对林泉。
山水无人到，寻僧独自然。时人事不尽，终意不知心。
夜雨天寒夜，寒灯觉气清。夜凉空有梦，鸟语莫相催。月午生寒露，山阴湿翠泉。已因临水曲，不识首阳虚。
夜夜虚亭竹，独坐秋风雨。酒醒宿鸟啼，月明霜白雪。
湖西春不尽，驿路路难穷。云外翠微白，燕飞风雨边。临岐人近国，过国客空清。却有青天意，新添百尺珠。
湖上春景去，江南自顾游。远山通远寺，寒水满西城。高景和云起，新春宿雨初。青山任平野，未必向江渍。
海上四时不可见，天涯南北一何穷。天涯亭上寻真处，一片东西望太平。
海上君看天地间，秦家三十六宫秋。为君相忆三千里，高会诗成一夜开。
月向烟霄满，乘兴汾水滨。清泉当月色，白日度东风。石水松仍见，松林月更清。应知猿鸟上，不独卧春山。
月照天台晓漏长，彩笺香炷落金汤。银河倒影珊瑚枕，银烛花开玉甃金。

Numpy 实现 LSTM 的尝试：

限于时间问题，我并没有用 numpy 实现完整的古诗生成网络，只实现了一个基本的单层 LSTM 单元的前向传播和反向传播：

LSTM.forward(inputs,state=None)

输入一个(input_length, input_size)的句子得到对应输出

LSTM.backward(k, delta)

输入目标函数关于 k 时刻 hidden_state 的梯度 delta，进行反向传播

具体细节见 numpy_lstm_check.py。

实验中假定目标函数是最后时刻 ht 各维度的和，通过对一个权重扰动一个微小值 epsilon=1e-4，来比较计算梯度和实际输出差距。经过验证后向传播基本正确。

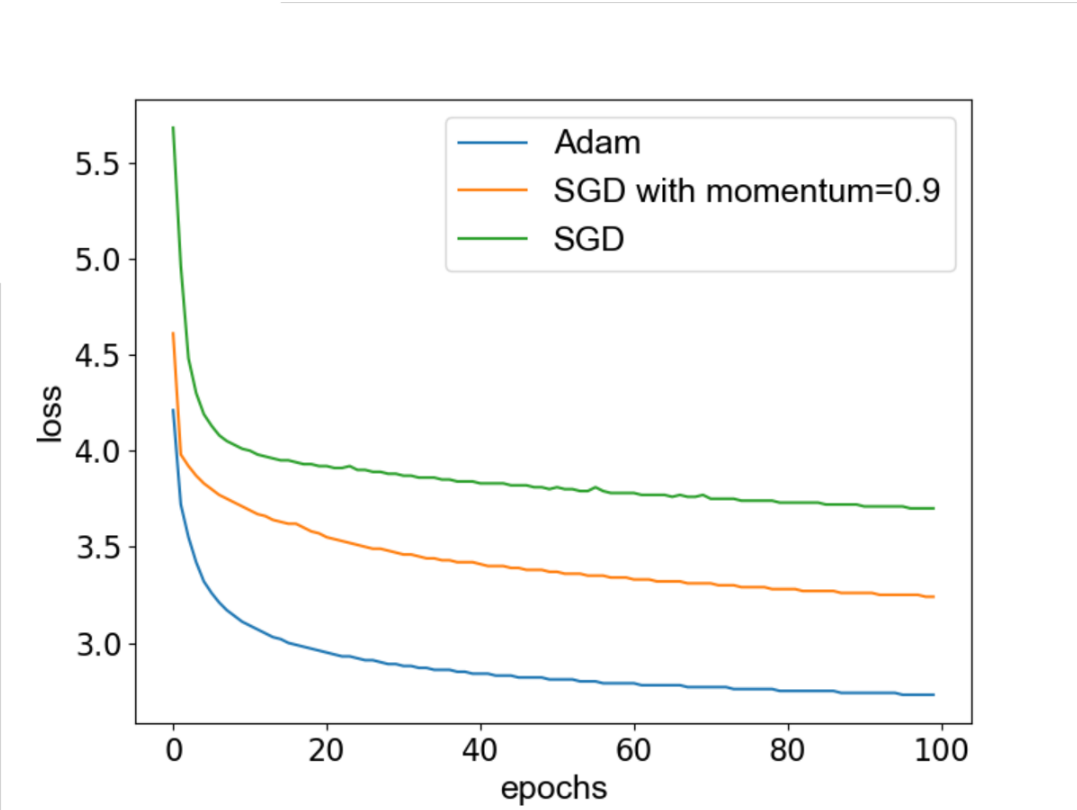
```
check gradient of Wfx
weights(0,0): expected, actual: -1.2903e-03, -1.2903e-03
weights(0,1): expected, actual: 2.5806e-03, 2.5806e-03
weights(0,2): expected, actual: 3.8710e-03, 3.8710e-03
weights(1,0): expected, actual: 8.6611e-04, 8.6611e-04
weights(1,1): expected, actual: -1.7322e-03, -1.7322e-03
weights(1,2): expected, actual: -2.5983e-03, -2.5983e-03

check gradient of Wfh
weights(0,0): expected, actual: 6.9333e-04, 6.9333e-04
weights(0,1): expected, actual: -4.5329e-04, -4.5329e-04
weights(1,0): expected, actual: -4.6538e-04, -4.6538e-04
weights(1,1): expected, actual: 3.0427e-04, 3.0427e-04
```

Requirement 3:

我尝试了 SGD 和 Adam 两种常用优化方法，以同样的模型参数和训练了 100 个 epoch，其训练参数与结果如下：

Optimizer	Learning rate	Final loss	Final
Adam	1e-3	2.79	17.67
SGD(momentum=0.9)	0.1	3.32	24.48
SGD(momentum=0)	0.1	3.78	39.37

其 loss 曲线图如下：



可以看出在本问题上，在同周期下，Adam 优化收敛得更快更好；而相比于普通的 SGD，momentum 确实能够更快更好的收敛。

程序说明

运行首先生成训练数据：python prepare.py，目录下须有 poetry/，内含文件格式为来源 github 上的 json 文件，最终生成包含训练集、验证集、词表的数据文件.pkl。

source.py 运行方式采用 argparse 传参：

python source.py

--data #传入之前生成的数据

--name #传入模型保存的名称

--hidden_size #

--embed_dim #embedding 维度

--dropout #dropout rate 大小

--layers #lstm 层数

--batch-size #表示 batch_size 大小

--num_epochs #表示 epoch 数

--gpu #表示是否 gpu 训练

--API #表示是否使用 pytorch 封装好的 lstm

--old-model #传入已训好的模型文件

--test #是否用于生成古诗