Assignment 3

Part I: Differentiate LSTM

1. 在 LSTM 对$\mathbf{h}_t$求导

- 由h_t公式及以下公式：

$$\begin{cases} \dfrac{\partial h_t}{\partial \tanh(C_t)} = o_t \\ \dfrac{\partial \tanh(C_t)}{\partial C_t} = 1 - \tanh^2(C_t) \end{cases}$$

- 可以得到：

$$\begin{cases} \dfrac{\partial h_t}{\partial o_t} = \tanh(C_t) & (1) \\ \dfrac{\partial h_t}{\partial C_t} = \dfrac{\partial h_t}{\partial \tanh(C_t)} * \dfrac{\partial \tanh(C_t)}{\partial C_t} = o_t * (1 - \tanh^2(C_t)) & (2) \end{cases}$$

- 由o_t公式及以下公式：

$$\begin{cases} \dfrac{\partial o_t}{\partial (W_o \cdot z + b_t)} = o_t * (1 - o_t) \\ \dfrac{\partial (W_o \cdot z + b_t)}{\partial W_o} = z^T \\ \dfrac{\partial (W_o \cdot z + b_t)}{\partial b_t} = \mathrm{I} \\ \dfrac{\partial (W_o \cdot z + b_t)}{\partial z} = W_o^T \end{cases}$$

- 可以得到：

$$\begin{cases} \dfrac{\partial o_t}{\partial W_o} = \dfrac{\partial o_t}{\partial (W_o \cdot z + b_t)} \cdot \dfrac{\partial (W_o \cdot z + b_t)}{\partial W_o} = o_t * (1 - o_t) \cdot z^T & (3) \\ \dfrac{\partial o_t}{\partial b_o} = \dfrac{\partial o_t}{\partial (W_o \cdot z + b_o)} \cdot \dfrac{\partial (W_o \cdot z + b_o)}{\partial b_o} = o_t * (1 - o_t) \cdot I = o_t * (1 - o_t) & (4) \\ \dfrac{\partial o_t}{\partial z} = \dfrac{\partial o_t}{\partial (W_o \cdot z + b_t)} \cdot \dfrac{\partial (W_o \cdot z + b_t)}{\partial z} = o_t * (1 - o_t) * W_o^T & (5) \end{cases}$$

- 同样可以求出f_t和i_t对应的导数：

$$\begin{cases} \dfrac{\partial f_t}{\partial W_f} = f_t * (1 - f_t) \cdot z^T \, (6) \\ \dfrac{\partial f_t}{\partial b_f} = f_t * (1 - f_t) \quad (7) \\ \dfrac{\partial f_t}{\partial z} = f_t * (1 - f_t) * W_f^T \, (8) \end{cases}$$

$$\begin{cases} \dfrac{\partial i_t}{\partial W_i} = i_t * (1 - i_t) \cdot z^T \, (9) \\ \dfrac{\partial i_t}{\partial b_i} = i_t * (1 - i_t) \quad (10) \\ \dfrac{\partial i_t}{\partial z} = i_t * (1 - i_t) * W_i^T \, (11) \end{cases}$$

- 由C_t公式可以得到：

$$\begin{cases} \frac{\partial C_t}{\partial f_t} = C_{t-1} (12) \\ \frac{\partial C_t}{\partial C_{t-1}} = f_t (13) \\ \frac{\partial C_t}{\partial i_t} = \bar{C}_t \ (14) \\ \frac{\partial C_t}{\partial \bar{C}_t} = i_t \ (15) \end{cases}$$

- 由$\bar{C}_t$公式及以下公式：

$$\begin{cases} \frac{\partial \bar{C}_t}{\partial (W_C \cdot z + b_C)} = 1 - \bar{C}_t^{\,2} \\ \frac{\partial (W_C \cdot z + b_C)}{\partial W_C} = z^T \\ \frac{\partial (W_C \cdot z + b_C)}{\partial b_C} = \mathrm{I} \\ \frac{\partial (W_C \cdot z + b_C)}{\partial z} = W_C^T \end{cases}$$

- 可以得到：

$$\begin{cases} \frac{\partial \bar{C}_t}{\partial W_C} = \left(1 - \bar{C}_t^{\,2}\right) \cdot z^T & (16) \\ \frac{\partial \bar{C}_t}{\partial b_C} = \left(1 - \bar{C}_t^{\,2}\right) \cdot I = \left(1 - \bar{C}_t^{\,2}\right) & (17) \\ \frac{\partial \bar{C}_t}{\partial z} = \left(1 - \bar{C}_t^{\,2}\right) \cdot W_C^T & (18) \end{cases}$$

- 由上述公式可以获得全部的最终结果：

$$
\begin{cases}
(1) => \dfrac{\partial h_t}{\partial o_t} = \tanh(C_t) & (1) \\
(2) => \dfrac{\partial h_t}{\partial C_t} = o_t * (1 - \tanh^2(C_t)) & (2) \\
(1) * (3) => \dfrac{\partial h_t}{\partial W_o} = \tanh(C_t) * o_t * (1 - o_t) \cdot z^T & (19) \\
(1) * (4) => \dfrac{\partial h_t}{\partial b_o} = \tanh(C_t) * o_t * (1 - o_t) & (20) \\
(2) * (12) => \dfrac{\partial h_t}{\partial f_t} = o_t * (1 - \tanh^2(C_t)) * C_{t-1} & (21) \\
(2) * (13) => \dfrac{\partial h_t}{\partial C_{t-1}} = o_t * (1 - \tanh^2(C_t)) * f_t & (22) \\
(2) * (14) => \dfrac{\partial h_t}{\partial i_t} = o_t * (1 - \tanh^2(C_t)) * \bar{C}_t & (23) \\
(2) * (15) => \dfrac{\partial h_t}{\partial \bar{C}_t} = o_t * (1 - \tanh^2(C_t)) * i_t & (24) \\
(23) * (16) => \dfrac{\partial h_t}{\partial W_C} = o_t * (1 - \tanh^2(C_t)) * i_t * \left(1 - \bar{C}_t{}^2\right) \cdot z^T & (25) \\
(23) * (17) => \dfrac{\partial \bar{C}_t}{\partial b_C} = o_t * (1 - \tanh^2(C_t)) * i_t * \left(1 - \bar{C}_t{}^2\right) & (26) \\
(20) * (6) => \dfrac{\partial h_t}{\partial W_f} = o_t * (1 - \tanh^2(C_t)) * C_{t-1} * f_t * (1 - f_t) \cdot z^T & (27) \\
(20) * (7) => \dfrac{\partial h_t}{\partial b_f} = o_t * (1 - \tanh^2(C_t)) * C_{t-1} * f_t * (1 - f_t) & (28) \\
(22) * (9) => \dfrac{\partial h_t}{\partial W_i} = o_t * (1 - \tanh^2(C_t)) * \bar{C}_t * i_t * (1 - i_t) \cdot z^T & (29) \\
(22) * (10) => \dfrac{\partial h_t}{\partial b_i} = o_t * (1 - \tanh^2(C_t)) * \bar{C}_t * i_t * (1 - i_t) & (30) \\
(1) * (5) + (21) * (8) + (23) * (11) + (24) * (18) => & \\
\left[\dfrac{\partial h_t}{\partial h_{t-1}}, \dfrac{\partial h_t}{\partial x_t}\right] = \dfrac{\partial h_t}{\partial z} = \tanh(C_t) * o_t * (1 - o_t) * W_o^T + & \\
o_t * (1 - \tanh^2(C_t)) * C_{t-1} * f_t * (1 - f_t) * W_f^T + & \\
o_t * (1 - \tanh^2(C_t)) * \bar{C}_t * i_t * (1 - i_t) * W_i^T + & \\
o_t * (1 - \tanh^2(C_t)) * i_t * \left(1 - \bar{C}_t{}^2\right) \cdot W_C^T & (31)
\end{cases}
$$

2. BPTT

- 设 sequence 的长度为T，第 k 个 step 的 loss 是l_t，k 个 steps 的 loss 总和是L_k，总的 loss 为L，那么可以有以下公式：

$$
\begin{cases}
\mathrm{L}_k = \sum_{t=k}^{T} l_t \\
l_t = -\sum_i y_{t,i} log \hat{y}_{t,i} \\
\hat{y}_{t,i} = softmax(W h_t + b)
\end{cases}
$$

- 而h_t只会影响 t 时候以后的 loss，那么可以有：

$$\begin{cases} \dfrac{\partial L}{\partial h_t} = \dfrac{\partial L_t}{\partial h_t} = \dfrac{\partial l_t}{\partial h_t} + \dfrac{\partial L_{t+1}}{\partial h_t} = \dfrac{\partial l_t}{\partial h_t} + \dfrac{\partial L_{t+1}}{\partial h_{t+1}} \dfrac{\partial h_{t+1}}{\partial h_t}, if \ \ t < T \\ \dfrac{\partial L}{\partial h_t} = \dfrac{\partial L_t}{\partial h_t} = \dfrac{\partial l_t}{\partial h_t}, if \ \ t = T \end{cases}$$

- 在反向传播的过程中，由上述公式可以求出全部$\frac{\partial L}{\partial h_t}$，而上文已经求出了$h_t$对其他参数的求导公式，那么可以计算出$L$对其他参数的 BPTT（以$W_i$为例）：

$$\frac{\partial L}{\partial W_i} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial W_i}$$

- 那么同理可以推到出其他参数的 BPTT 计算。

Part II: LSTM training

1. initialization

1.1 为什么不能全部初始化为 0？

- 不同参数都初始化为同一个值，会使得这些参数在每一次的迭代中产生相近的梯度值，更新后的值会依然保持相近。在多次迭代之后，这些参数很可能会保持着相近的值，使得模型参数保持平衡的状态；
- 一开始就把参数初始化为固定的值，很可能会使模型处于一个不好的状态，而训练的过程中也很难使模型到达好的位置；
- 在上一原因的一种特殊情况下，还可能会使参数一直保持为 0，没办法逃离这个不好的状态

1.2 因此在模型参数初始化的时候，不能把模型初始化为固定值，也更加不能全都初始化为 0

1.3 参数初始化方案：

- 随机数初始化：随机数服从某个分布，如 normal 分布、uniform 分布等，且数值偏小。由于引入了随机化，使得模型的初始位置不固定，降低模型落入不好的状态的可能。这种方法对小规模网络效果很好（如这次实现的 LSTM），但在规模大的网络中效果不好。
- Xavier initialization :对激活函数是线性的网络效果不错，但是对于非线性激活，如 ReLU，效果不好

$$\mathrm{W} = \frac{random(fan_{in}, fan_{out})}{sqrt(fan_{out})}$$

1.4 用 pytorch 来实现网络时，embedding 层使用 normal 分布初始化的，其他的参数使用 normal 分布初始化的

2. 代码组织

文件名	功能
config.py	在 class 中定义模型参数
transform.py	把全唐诗数据转换为和 tangshi.txt 相同的格式的 txt 文件
dataset.py	读取 txt 数据，利用 fastnlp 生成 train 数据、dev 数据和 vocabulary
model.py	定义 LSTM 模型和生成诗歌的模型
lstm.py	用 numpy 实现 LSTM 模型

utils.py	定义计时 Callback 类、perplexity 的 Metric 类、Loss 类
train.py	定义 trainer 并训练模型
generate.py	生成诗歌
visualize.py	在训练保存的 log 中提取出 loss 并画出曲线

3. 部分参数定义

vocab size	**input size**	**hidden size**	max epoch	**batch size**	**seq len**	optimizer	lr	patience
6795	**128**	**512**	256	**128**	**128**	Adam	1e-3	10

4. 数据集

- 使用了汇总了的**全唐诗**数据 tang.npz，可以在下方连接下载。一共有 57580 篇诗歌，最长为 124，vocab size 为 6795

https://github.com/chenyuntc/pytorch-book/tree/master/chapter9-%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C%E5%86%99%E8%AF%97(CharRNN)

- 使用 langcov 包把繁体字转换成简体字，再以 tangshi.txt 相同的格式输出数据集；
- 读取生成的数据集，使用 fastNLP 的 DataSet 和 Vocabulary 来构建 train_data、dev_data 和 vocabulary。其中，使用 apply 函数来为诗歌添加<START>、<eos>以及<pad>，sequence length 设定为 128 且保持不变，并按 0.2 的验证比例划分数据。

5. 模型结构

- 1 层 embedding layer + 1 层 LSTM layer + 1 层 fully connected layer
- 用 pytorch 自己实现了 LSTM layer，同样也在 model.py 文件中。

6. Perplexity 计算

- 通过原公式，可以转换为：

$$\text{Perplexity} = \exp\left(-\frac{1}{N}\sum \log(y)\right)$$

- 用上述公式计算出 dev_data 中的每一条 sequence 的 pp 后，在整个 dev_data 中取平均值，作为最终的 pp。
- 基于 pytorch 实现了 batched、无循环的 PP 计算，大大提高计算速度
- 应用 fastNLP 中的 Metric 来实现

7. Callback 实现

- Early stop：patience 设置为 10，使用了 fastNLP 提供的函数
- 计时：基于 fastNLP 的 Callback，实现了 on_epoch_end 的计时

8. 训练结果

- 参数结果

end epoch	start perplexity	best perplexity	best pp epoch	total time
14	105.5369	23.4275	12	2146990ms

- 生成诗歌

start word	temperature	poem
日	0.9	日里衡阳道，乘舟渡流水。江湖今已秋，风急白波上。 秋帆远城曲，秋色俱回首。夏木暗寒薪，秋光在潇湘。
	0.6	日月照高台，浮云出高位。登眺多清旷，风尘出南斗。 谁谓天上仙，可怜长安道。长安石堂中，立马出门户。 行行不知见，心地不回首。当时运为谁，相逢不须待。 心心不可识，百尺忽可忖。方寸朱颜生，不辞春泥滓。
红	0.9	红霞未落瑟，故照肤绿阴。天下潭水合，溶溶烟雨盘。 霜门著寒涧，望处稍残眠。有时掉青松，谁知怀素姿。 沧溟白日返，一夕赋挽间。朝客登高陟，坐窥苍梧里。
	0.6	红玉名高树，白露清汉宫。进我不得意，愿言出王宫。 别来身未老，相见愿相期。云门自兹望，流水不能回。 何必守奇才，南园百年余。安侯复何足，生事不能迁。 风尘旦暮去，逢影何萧条。生死自有道，非因谁与君。
山	0.9	山暝年月女，故人裴下耕。壮哉决耻末，境胜无常哭。 昨日欲下祖，今来相对未。轻罗流水入，牵裾紫袍发。 湘水东流归，远游青天暮。曾持历时人，从此白云里。 故园待书字，日日醉春望。行人缆萧瑟，百兽遥相见。
	0.6	山影摇云天，天台远水流。流沙如黛色，风雨正飘飘。 道路伤心道，心轻复久依。出门应半岫，上鸟更裴回。 未必多时事，携琴对故人。无因驻文节，羞向九华峰。 况复停车日，谁言入世情。不知天地静，难与六年同。
夜	0.9	夜连月下漱帘帏，东望杜陵出关路。 白日青天花下烧，落花满地春未息。 君看歌舞入筝弦，今暮视云无定非。 有草帝里姑苏积，自然风景相如射， 簿有斑牙旧游侠，楚天无分无语款。 西施沈醉水初头，绿鬟不曾妆粉弦。 孤城一夜行人绝，夜闺啼顾欲何许。
	0.6	夜帘卷帘幕，月出天山曙。窗明鸟兽啼，手破松边竹。 谁家有余客，来往无人识。梦君一片月，一径无一片。 远客相思空，愁肠断绝域。回头望断绝，门馆想孤老。 有时还见来，不知何处宿？浮生独羁旅，复此坐西郭。 别有门外情，相思两相忆。江头人不知，烟波长不见。
湖	0.9	湖上渔客归思归，桐花开掞生且同。 满堂蜀客开不是，真人谓我为君子。 谁道朝侯出席酒，就头相劝难寻睡。 刘张旧卿即多应，未肯调应皆为贵。 狂风吹不如折腰，似家问我君莫频。 西南路长开万里，愁来不见隔山烟。
	0.6	湖上见东海，连天齐白云。仙人清夜夜，一雨一时清。 凛凛秋风落，天寒水色寒。长沙见星象，赏心极空虚。 云水忽见飞，兰桡在五城。夜闻山桂曲，晓泛竹楼风。 山阴有余事，中夜生明朝。出入长安友，远林何处营。

		幽人见此曲，古木多为风。远近苍苍岭，高峰乱峰峦。
海	0.9	海水经不极，沧波起来波。古人自忠孝，感激归重航。 声华天地间，帝里光华九。稽首相见人，吾传蹑汉使。 明朝是嗟士，古帝亲频悦。四顾尚清游，异乡两三壤。
	0.6	海阔滩水入，流波涵波澜。楚宫多感激，白首信难测。 吾知辟奇书，敢谓归太守。百战岂能仁，新丰谅难戢。 江南春已暮，风月清且薄。清风吹白日，秋夜思归极。 碧树青霄间，青云垂露白。地云亘多关，东路骛尘土。 青云傍天池，白露寒光落。时登涧底绝，不见栖栖雨。
月	0.9	月下雪皑寒夜短，上宫天色朝云生。 清宫清昼万片开，梁园幸上百城宫。 会君栽苑有长绰，盛君侍寝望神仙。
	0.6	月明江上寺，春风水西风。两岸万里叶，回头绿水边。 身同一片坐，一片起孤舟。初霁天门乱，山连水色明。 金蝉出城阙，草木满山风。晓日登楼望，孤云出望行。 遥怜五湖色，不动百花开。白发谁人识，青山旧识君。 城中无限日，相见自相亲。马首杨花里，天涯鹤到家。

9. LSTM 的 numpy 实现

- 基于提供的 Vanilla LSTM with numpy 博客实现，在 lstm.py 文件中；
- 在定义模型的时候，把 PoetryModel 中的 LSTM 替换为 LSTMNumpy，其他参数与上述模型一致；
- 梯度更新是基于 Adam 实现的；
- 运行了 3 次，同样能在约 14 epoch 的时候达到 early stop，且 loss 也能降到约 2，pp 约为 24；
- 但运行时间更长，而且在 GPU 上没有明显的加速。

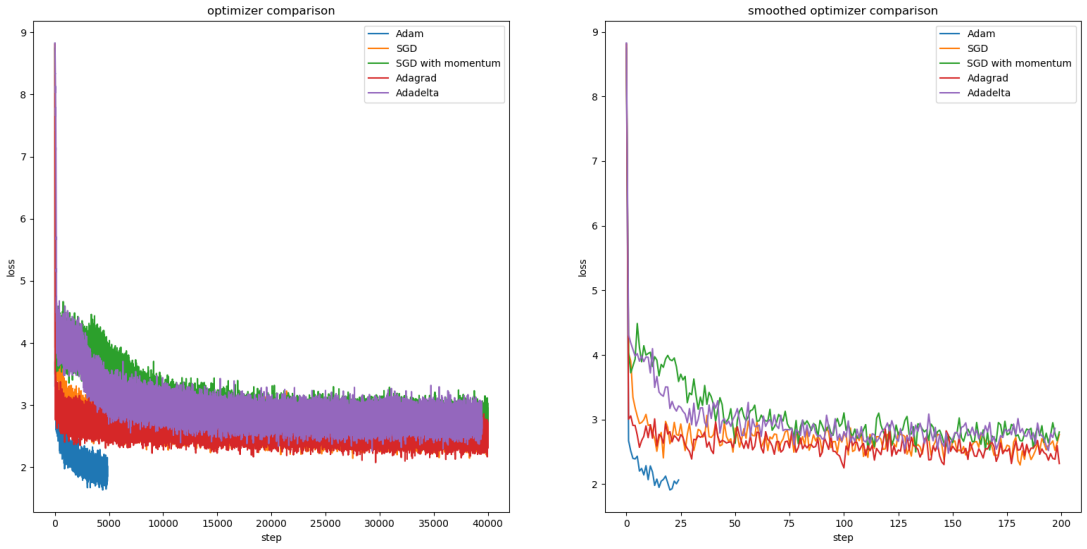
10. Optimization 对比

10.1 不同 optimizer 的模型参数设定

optimizer	Adam	SGD	SGD(momentum)	Adagrad	Adadelta
embed size	128				
hidden size	512				
batch size	128				
max step	92160 (256 epochs)				
best step	4300	39600	41600	74050	38900
best epoch	12	110	116	206	109
lr	1e-3	1e-1	1e-3	1e-3	1e-2
loss	1.91313	2.51915	2.77447	2.41321	2.69161
perplexity	23.4275	47.6558	69.593	41.2293	69.3051
patience	10				
weight decay	0				
momentum		0	0.9	0	
rho					0.9
eps					1e-6

time per epoch	170620ms	153122ms	173231ms	158737ms	150342ms

10.2 loss 对比：其中左侧以 step 为单位显示 loss，右侧进行了光滑化，每 200 steps 显示一次 loss



10.3 optimizer 对比

- Adam：收敛速度最快，效果也最好
- SGD 和 SGD with momentum：momentum 为 0.9 的比 0 的收敛更快，最优结果也稍好一些
- Adagrad 和 Adadelta：Adagrad 比 Adadelta 收敛更快，但最终结果类似

11. 对梯度计算实现的影响

- 不同的优化方案，根据公式，需要添加一些不同的参数，那么只需要在每个变量中都添加这些所需的参数，在计算梯度的过程中对其进行更新、使用就行了。

Part III：建议

1. Vocabulary 在 init 的时候可以添加 end_of_sentence='<EOS>', start_of_sentence='<START>'
2. 可以导入 dataset 直接生成 Vocabulary
3. Trainer 只能传进 1 个 batch_size，但是 train 和 dev 数据其实可以用不同的 batch_size
4. 由于周六的风雨导致服务器宕机了，模型的训练中断了，但是似乎不可以继续训练，希望可以支持继续训练模型。