lab3 实验报告

1、梯度计算

单次梯度计算

$$
\begin{aligned}
\frac{\partial h_t}{\partial o_t} &= tanh(C_t) \\
\frac{\partial h_t}{\partial C_t} &= o_t * (1 - tanh^2(C_t)) \\
\frac{\partial h_t}{\partial \bar{C}_t} &= \frac{\partial h_t}{\partial C_t} * \frac{\partial C_t}{\partial \bar{C}_t} \\
&= o_t * (1 - tanh^2(C_t)) * i_t \\
\frac{\partial h_t}{\partial C_{t-1}} &= \frac{\partial h_t}{\partial C_t} * \frac{\partial C_t}{\partial C_{t-1}} \\
&= o_t * (1 - tanh^2(C_t)) * f_t \\
\frac{\partial h_t}{\partial f_t} &= \frac{\partial h_t}{\partial C_t} * \frac{\partial C_t}{\partial f_t} \\
&= o_t * (1 - tanh^2(C_t)) * C_{t-1} \\
\frac{\partial h_t}{\partial i_t} &= \frac{\partial h_t}{\partial C_t} * \frac{\partial C_t}{\partial i_t} \\
&= o_t * (1 - tanh^2(C_t)) * \bar{C}_t
\end{aligned}
$$

对于z有

$$
\begin{aligned}
\frac{\partial o_t}{\partial z} &= W_o^T \cdot [o_t * (1 - o_t)] \\
\frac{\partial f_t}{\partial z} &= W_f^T \cdot [f_t * (1 - f_t)] \\
\frac{\partial i_t}{\partial z} &= W_i^T \cdot [i_t * (1 - i_t)] \\
\frac{\partial \bar{C}_t}{\partial z} &= W_C^T \cdot (1 - \bar{C}_t^2) \\
\frac{\partial C_t}{\partial z} &= \frac{\partial f_t}{\partial z} * C_{t-1} + \frac{\partial \bar{C}_t}{\partial z} * i_t + \frac{\partial i_t}{\partial z} * \bar{C}_t \\
\frac{\partial h_t}{\partial z} &= \frac{\partial o_t}{\partial z} * tanh(C_t) + \frac{\partial tanh(C_t)}{\partial z} * o_t \\
&= W_o^T \cdot [o_t * (1 - o_t) * tanh(C_t)] + \frac{\partial C_t}{\partial z} * o_t * (1 - tanh^2(C_t)) \\
&= W_o^T \cdot [o_t * (1 - o_t) * tanh(C_t)] \\
&+ W_f^T \cdot [f_t * (1 - f_t) * C_{t-1} * o_t * (1 - tanh^2(C_t))] \\
&+ W_C^T \cdot [(1 - \bar{C}_t^2) * i_t * o_t * (1 - tanh^2(C_t))] \\
&+ W_i^T \cdot [i_t * (1 - i_t) * \bar{C}_t * o_t * (1 - tanh^2(C_t))]
\end{aligned}
$$

于是有

$$[\frac{\partial h_t}{\partial h_{t-1}}, \frac{\partial h_t}{\partial x_t}] = \frac{\partial h_t}{\partial z}$$

对于模型的参数：

$$\begin{aligned}
\frac{\partial h_t}{\partial W_f} &= \frac{\partial h_t}{\partial f_t}\frac{\partial f_t}{\partial W_f} \\
&= \frac{\partial h_t}{\partial f_t} * f_t * (1-f_t) \cdot z^T \\
&= o_t * (1-tanh^2(C_t)) * C_{t-1} * f_t * (1-f_t) \cdot z^T \\
\frac{\partial h_t}{\partial b_f} &= \frac{\partial h_t}{\partial f_t}\frac{\partial f_t}{\partial b_f} \\
&= o_t * (1-tanh^2(C_t)) * C_{t-1} * f_t * (1-f_t) \\
\frac{\partial h_t}{\partial W_i} &= \frac{\partial h_t}{\partial i_t}\frac{\partial i_t}{\partial W_i} \\
&= \frac{\partial h_t}{\partial i_t} * i_t * (1-i_t) \cdot z^T \\
&= o_t * (1-tanh^2(C_t)) * \bar{C}_t * i_t * (1-i_t) \cdot z^T \\
\frac{\partial h_t}{\partial b_i} &= \frac{\partial h_t}{\partial f_t}\frac{\partial f_t}{\partial b_f} \\
&= o_t * (1-tanh^2(C_t)) * \bar{C}_t * i_t * (1-i_t) \\
\frac{\partial h_t}{\partial W_o} &= \frac{\partial h_t}{\partial o_t}\frac{\partial o_t}{\partial W_o} \\
&= \frac{\partial h_t}{\partial o_t} * o_t * (1-o_t) \cdot z^T \\
&= tanh(C_t) * o_t * (1-o_t) \cdot z^T \\
\frac{\partial h_t}{\partial b_o} &= \frac{\partial h_t}{\partial o_t}\frac{\partial o_t}{\partial b_o} \\
&= tanh(C_t) * o_t * (1-o_t) \\
\frac{\partial h_t}{\partial W_C} &= \frac{\partial h_t}{\partial \bar{C}_t}\frac{\partial \bar{C}_t}{\partial W_C} \\
&= \frac{\partial h_t}{\partial \bar{C}_t} * (1-\bar{C}_t^2) \cdot z^T \\
&= o_t * (1-tanh^2(C_t)) * i_t * (1-\bar{C}_t^2) \cdot z^T \\
\frac{\partial h_t}{\partial b_C} &= \frac{\partial h_t}{\partial \bar{C}_t}\frac{\partial \bar{C}_t}{\partial b_C} \\
&= o_t * (1-tanh^2(C_t)) * i_t * (1-\bar{C}_t^2)
\end{aligned}$$

对整个句子的梯度运算

令句子长度为n，输出维度为m，对于t时刻的输出$\hat{y}_t$，令target为y_t，损失函数为l_t，有：

$$\hat{y}_t = softmax(W \cdot h_t + b)$$
$$l_t = -\sum_{j=1}^{m} y_t^j log(\hat{y}_t^j)$$

记$L_t = \sum_{k=t}^{n} l_k$，则$L = L_1$，为了计算$\frac{\partial L}{\partial h_t}$，当$t=n$时：

$$
\begin{aligned}
\frac{\partial L}{\partial h_t} &= \frac{\partial l_t}{\partial h_t} \\
&= W^T \cdot (\hat{y}_t - y_t)
\end{aligned}
$$

当$t < n$时：

$$
\begin{aligned}
\frac{\partial L}{\partial h_t} &= \frac{\partial l_t}{\partial h_t} + \frac{\partial L_{t+1}}{\partial h_t} \\
&= W^T \cdot (\hat{y}_t - y_t) + \frac{\partial L_{t+1}}{\partial h_{t+1}} * \frac{\partial h_{t+1}}{\partial h_t} \\
&= W^T \cdot (\hat{y}_t - y_t) + \frac{\partial L}{\partial h_{t+1}} * \frac{\partial h_{t+1}}{\partial h_t}
\end{aligned}
$$

首先我们正向遍历一边句子，得到每个时刻t的输出结果，再反向遍历，由于在单次梯度计算中我们已经得到$\frac{\partial h_t}{\partial h_{t-1}}$，结合$t+1$时的$\frac{\partial L}{\partial h_{t+1}}$，即可得到$\frac{\partial L}{\partial h_t}$，最后对于lstm的模型参数，以$W_f$为例：

$$
\frac{\partial L}{\partial W_f} = \sum_{t=1}^{n} \frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial W_f}
$$

即完成对所有模型参数的梯度计算。

2、模型实现

初始化

为何不初始化为全0

若模型的参数初始化为0，那么每次迭代训练时模型参数的梯度也是相近的，这样不同的参数就学习不到不同的特征，网络学习能力非常有限，表现也会相应变差。另外对于embedding层，初始化为0后，每一个汉字都会被编码为同一个数字，和embedding层的初衷相违背。

初始化的方法

根据pytorch的源码，可以考虑将embedding层用正态分布初始化，线性层的权重矩阵用xavier均匀分布初始化，偏置初始化为全0。当然对于权重矩阵也可以考虑用不同的参数进行均匀分布初始化。

数据预处理

这次采用的唐诗数据集来自https://github.com/L1aoXingyu/Char-RNN-PyTorch，我利用fastnlp中的vocabulary进行预处理，并保留了所有的标点符号，为使得模型可以生成任意长度的古诗，我并没有在诗的末尾添加EOS，而是将诗的开头字符作为最后一个target。

Pytorch实现

将整个模型写成nn.Module类，定义好optimizer，即可训练，代码文件为lstm_torch.py。

Numpy实现

numpy的版本需要自己手写求导和更新过程，根据梯度计算的分析即可得到，代码文件为lstm_numpy.py，其中用gradient_check函数对模型求得的梯度进行了检查。

模型训练

sl	batch_size	vocab_size	input_size	hidden_size
64	64	4428	128	256

代码中手写了early_stop，当前perplexity超过前10个epoch的平均值时中断训练。

训练结果

生成了长度为64的诗：

```
日暮归。何当千岁晏外。日月下山河水去。秋水向南山，孤峰出海滨。何人何处尽，不觉洛城隅，孤云何处飞，看花如霰。春草生春色，春草生绿
红芳春。玉座开花落叶台。，此夜不相携时。日暮寒江上客。秋色动秋山水，山云独未还家，愁心明日月中，行人不见弃心处。白日下长安道，天
山中路远，日落照华空，春风生早著来。自从天下远，不觉汉家贫家，还将万里开书稀稀。何事不可见。今日不见君，一杯长无状，天台访道才书
夜深林外去，花柳色空留客。春草不堪悲，愁来无限人。不见君不去去去去处，独坐悲风流。白日暮山川上人，别有一书眠，天文物外馀，书幌闻
湖南北阙路，独向洛川归。不见江山暮，山河不可忘，空悲风尘。不知何所贵。日落照日光，春来不自闻来，不堪攀桂树，天上一杯长，君看不自
海隅分散影飞，落照空林间夕微微微。日月空林里，春来不可寻香车马，春来向水痕。不堪闻玉匣。今日照前山。日月长门闭，山河入夜深人。白
月明光。玉女临窗下夕，风雨湿花飞。白发云中夜色，花落照秋声彻。秋色带寒塘。，山川暮春。秋风动寒山色。，此夜几人稀，愁心明日月，。
```

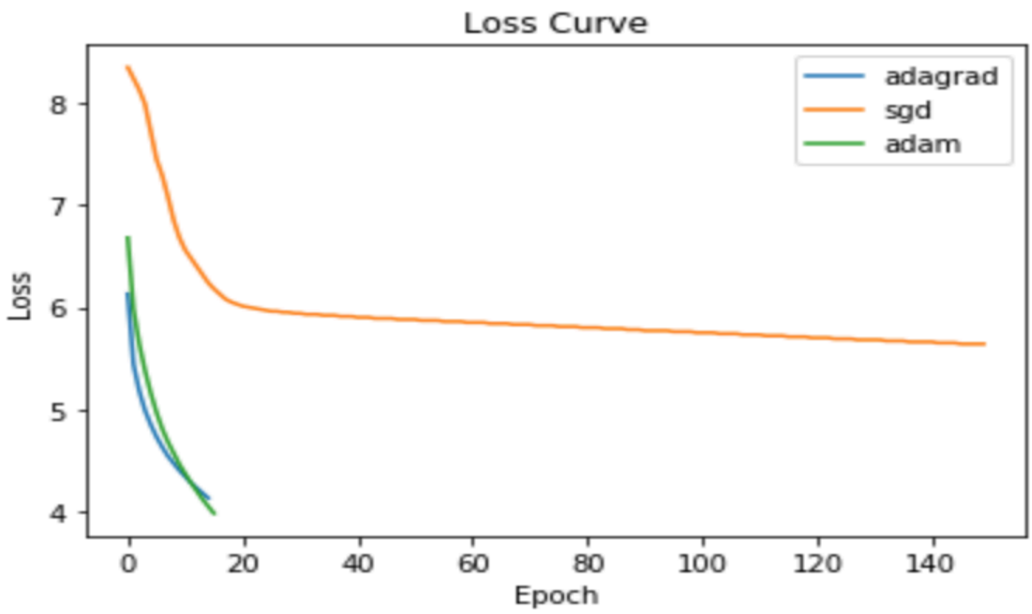
3、优化

pytorch下不同优化方式的对比

我分别采用了SGD, Adam, Adagrad三种优化方式，参数设置和得到的结果如下：

Method	learning rate	epoch	loss	perplexity
SGD	1e-1	150	5.64	265.00
Adam	2e-3	16	3.98	151.50
Adagrad	2e-2	15	4.13	156.12

loss曲线图如下：



比较可以发现，SGD的收敛速度最慢，虽然学习率设为了0.1，但在运行了150个epoch后仍未收敛，并且loss仍然很高，测试集的表现也很差。Adagrad收敛速度明显快于SGD，而Adam的收敛速度略快于Adagrad，并且Adam在测试集上的表现要优于Adagrad，训练得到的模型泛化性能更好。

numpy实现

为了实现例如SGD with momentum，adagrad等优化方式，对于模型的所有参数，除了要维护梯度外，还要额外维护动量，在更新参数时考虑动量对其的影响，以adagrad为例，代码如下：

```
offset = 1e-9
param.m += param.g * param.g
param.v = param.v - lr * param.g / np.sqrt(param.m + offset)
```