Part 1, Differentiate LSTM

Requirements

1.

3ht = diag [tanhecot] out = was [of oct -tank reet] = d)ag[a+0 (1-+ml2(0+)), 0 C+4] = diag [of D (1- +outerch) & "+] = diag [0+0 (1-tonh?(4)) () it] = 8 dlog [07 0(1 +0.12 (4)) 0 fz] That = tank (ct) O ot 8 (+ 07) 0 W. t of O (1- tank of ca) O cay Oft o (1-for O W) + Of O(+ tanh (cht) O of O it O (1- it) BU! + of 0 (1 - tank " (4) 0 it 6 (1 - 4). OW(out = tanhille) Goto (TOH) G Wo + ETEIT- HOUN'S GH) BULY Of O (1-14) ONE + 0+ 0(1-tonk2 (lon)) 0 0 0 it 0 (1-it) 6W: + 0+ 0 (1-tanh 2 (bt)) 6 1+ 0 (1-42) ONC = (tanh(4) 0 0+0 (1-0+)) + 2T = (of 0 (1-tout " (4)) 6 (ty 0 ft 0 (1-ft) + 27 and = (0+0 1+++ (4) (0 it 6 (1-it)) *27 21 = (0+0 (1-fanh21 c+)) .6 1+ 0 (1-42) +27 that = (tampletio of 0 (+ U+1) 5th = (070 (1-tanh (CH) OCH O FE (1-1+1) = (oto (premier of) Or Olt Olt Oll Til = (0+0(1-taxh2(1+)) 01+0(1-4)

2.

将从开始到该时刻的所有对该参数的偏导数值累加起来,就是对这个参数的偏导数值的最终结果。

需要记录计算所需要的参数在各个时刻的值。

Part 2, Autograd Training of LSTM

Requirements

1.

不能全部初始化为0的原因:

如果把模型或者层的参数全部初始化为 0, 就会导致在前向传播中计算出来的所有参数值都相同,包括隐含层、输出层的所有参数,当在反向传播更新参数的时候,就会出现损失函数对所有参数的偏导数都相同的情况,即更新值也是相同的,即更新之后的参数也是相同的。以此类推,无论进行多少轮前向传播和反向传播,整个模型的参数值都不会产生变化。在一个模型中,如果所有的节点都具有相同的参数值,就意味着它们在计算同一个特征,整个模型就会变得和只有一个节点一样,这样模型就失去了学习不同特征的能力,所以不能将模型的所有参数都初始化为 0。

模型参数初始化的方法:

可以让模型的参数满足 0 到 1,或者-x 到 x 之间的正态分布。

2

训练之后的 perplexity: 163.39779

日、紅、山、夜、湖、海、月生成的古诗:

日出東山去, 遙知故園人。

紅花落葉落, 花落葉中春。

山水東流水, 山花滿地寒。山花何處處, 不見故園人。

夜來不見日, 不見故園人。

湖上春風雨,山花滿樹寒。山花何處處,不見故園人。

海上山花落,花間獨未歸。 月色不可見,山中有路難。

3.

RMSprop:

perplexity:

163.39779

result:

日出東山去, 遙知故園人。

紅花落葉落, 花落葉中春。

山水東流水,山花滿地寒。山花何處處,不見故園人。

夜來不見日, 不見故園人。

湖上春風雨,山花滿樹寒。山花何處處,不見故園人。

海上山花落,花間獨未歸。 月色不可見,山中有路難。

SGD:

perplexity:

1532.78005

result:

 月明月明月明月明月明月明月明月明月明月明月明月明月明月

Adam:

perplexity:

310.47545

result:

日日臨風起,春風吹柳絲。

紅芳草色如霜雪,春風吹柳絲。

山上客心在, 春風吹柳絲。

夜久花落盡,春風吹柳絲。

湖上春風吹,春風吹柳絲。

海上春風吹, 春風吹柳絲。

月上宫花發,春風吹柳絲。

Hyperparameter and training setting

|V|:6824

bs:200

sl:

hs:256

Is:determined by the length of poem

README

dataset/(运行 train.py 后生成)

在 GitHub 上下载的全唐诗数据集,数据格式为 json。

lstm_numpy/

使用 numpy 实现 BP 的 lstm,编写了一个测试函数,使用这个 lstm 拟合了一个输入序列。create dataset.py

实现了从 GitHub 上 clone 全唐诗数据集(src:https://github.com/jackeyGao/chinese-poetry.git)、处理 json 文件、处理唐诗、获得诗表的四个函数。

rnn.py

创建了模型类 RNN。先基于 torch.nn.Module 实现了 LSTM,然后基于 LSTM 创建了模型类。整个网络有一个 embedding 层、一个 lstm 层、一个 linear 层、一个 softmax 层。

data process.py

用来处理数据,三个函数的输入都为(诗,单词表),输出是输入张量、目标张量和训练数据。

train.py

用来训练模型。先产生了诗表和单词表,然后进行训练。**注意开始运行时会需要几分钟的时间,因为需要从 GitHub 上 clone 全唐诗数据集。**训练过程会输出当前 epoch、loss,验证记集上的 loss、perplexity 和运行时间。每个 epoch 结束后都会保存当前模型,保存命名是:rnn_(epoch)_(perplexity)_(optimizer).pt。

test.py

用来测试。使用方法是先在 rnn = torch.load(...)中输入希望导入的模型文件名,然后在 print(test(...))中输入希望作为开头的汉字。**注意因为所使用的数据集时繁体数据集,所以开头字需要使用繁体,实际上只需要将"红"改成"紅"。**