姓名

- 张进华

学号班级

- 智能19 201900150221

实验日期

- 2021-11-11

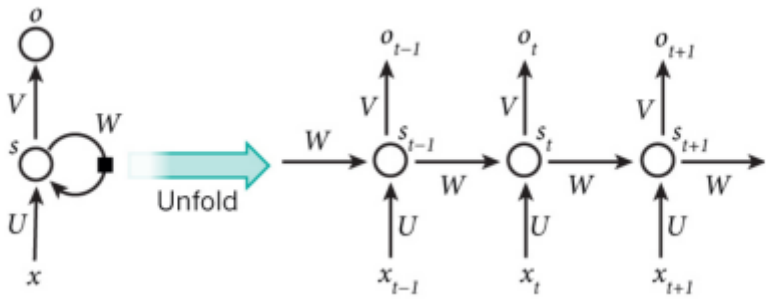
实验题目 Fun with RNNs

扩展 `min-char-rnn.py`，试验莎士比亚数据集 (`shakespeare_train.txt`)。

实验步骤

步骤一 了解RNN

- 普通的神经网络都只能单独地处理一个个的输入，前一个输入和后一个输入是完全没有关系的。但是，某些任务需要能够更好的处理序列的信息，即前面的输入和后面的输入是有关系的,RNN就是一个可以保存前面时间步信息的神经网络
- RNN(循环神经网络)基本框架



- 上图左侧是一个最基础的循环神经网络，包括一个输入层、一个隐藏层和一个输出层。
  其中：
  - 1.x 是输入层的值。
  - 2.s 是隐藏层的值，U 是输入层到隐藏层的权重矩阵。
  - 3.o 是输出层的值，V 是隐藏层到输出层的权重矩阵。
  - 4.W 也是一个权重矩阵，是隐藏层上一次的值对这一次的输入的权重，这个权重矩阵说明隐藏层的值 s 不仅仅取决于当前这次的输入 x，还取决于上一次隐藏层的值 s。

步骤二 RNN 语言模型对其输出分布使用 softmax 激活函数

- RNN 语言模型对其输出分布使用 softmax 激活函数
  在每个时间步。可以通过将 logits 乘以 a 来修改分布

常数α：

- y = softmax(α z)
- 在这里，1/α 可以被认为是一个“温度”，即较低的 α 值对应于“更热”的分布。其实，α 越大，概率分布越分散，α 越小，概率分布 越集中。

步骤三 sample函数

- 功能：输入一个字母的索引seed_ix，利用当前RNN模型，根据该字母创建整个句子，然后返回句子中出现的字母对应的索引列表ixes
- 输入
  - h是隐藏层状态，也就是前面的时间步留下的信息
  - seed_ix是一个索引，也就是我们要输入的字母对应的索引
  - n：句子长度（要生成的字母索引的个数）

```
def sample(h, seed_ix, n, alpha):
    """
  从模型中采样整数序列h是隐藏层状态，seed_ix 是第一个时间步的种子字母
  """
    # Start Your code
    x = np.zeros((vocab_size, 1)) # vocab_size是不重复的字母的数量
    x[seed_ix] = 1 # one-hot 编码
    ixes = []
    for t in range(n):
        h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh) # h是隐藏层状态
        y = np.dot(Why, h) + by # 得分向量，每个分数都是该分数的索引对应的字母的得分
        p = np.exp(alpha * y) / np.sum(np.exp(alpha * y)) # 利用 softmax ， 将得分转化成概率
        ix = np.random.choice(range(vocab_size), p=p.ravel()) # 按 p 中的概率取出一个索引
        x = np.zeros((vocab_size, 1)) # 重置编码向量 x ，以供下一个时间步利用
        x[ix] = 1
        ixes.append(ix)
    return ixes
    # End your code
```

步骤四 comp函数

- 功能：给定一个长度为 m 的字符串，将长度为 n 个字符的字符串补全

```
    # 生成上下文文本
    for t in range(m):
        # Start Your code
        h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh) # h是隐藏层状态
        # x 是字符中索引为 1 的 k 个编码之一
        x = np.zeros((vocab_size, 1))
        ix = inputs[word_index + 1]
        word_index += 1
        x[ix] = 1
        # End your code

        ixes.append(ix) # 生成输入字符串

    # 从数据中计算 softmax 概率和样本，并使用输出作为我们开始延续的下一个输入
```

```
    # Start Your code
    y = np.dot(Why, h) + by
    p = np.exp(y) / np.sum(np.exp(y))
    ix = np.random.choice(range(vocab_size), p=p.ravel())
    x = np.zeros((vocab_size, 1))
    x[ix] = 1
    # End your code

    # 开始生成字符串
    ixes = []
    for t in range(n):
        # Start Your code
        h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh) # h是隐藏层状态
        y = np.dot(Why, h) + by # 得分向量，每个分数都是该分数的索引对应的字母的得分
        p = np.exp(y) / np.sum(np.exp(y)) # 利用 softmax ， 将得分转化成概率
        ix = np.random.choice(range(vocab_size), p=p.ravel()) # 按 p 中的概率取出一
个索引
        x = np.zeros((vocab_size, 1)) # 重置编码向量 x ，以供下一个时间步利用
        x[ix] = 1
        # End your code

        ixes.append(ix)
```

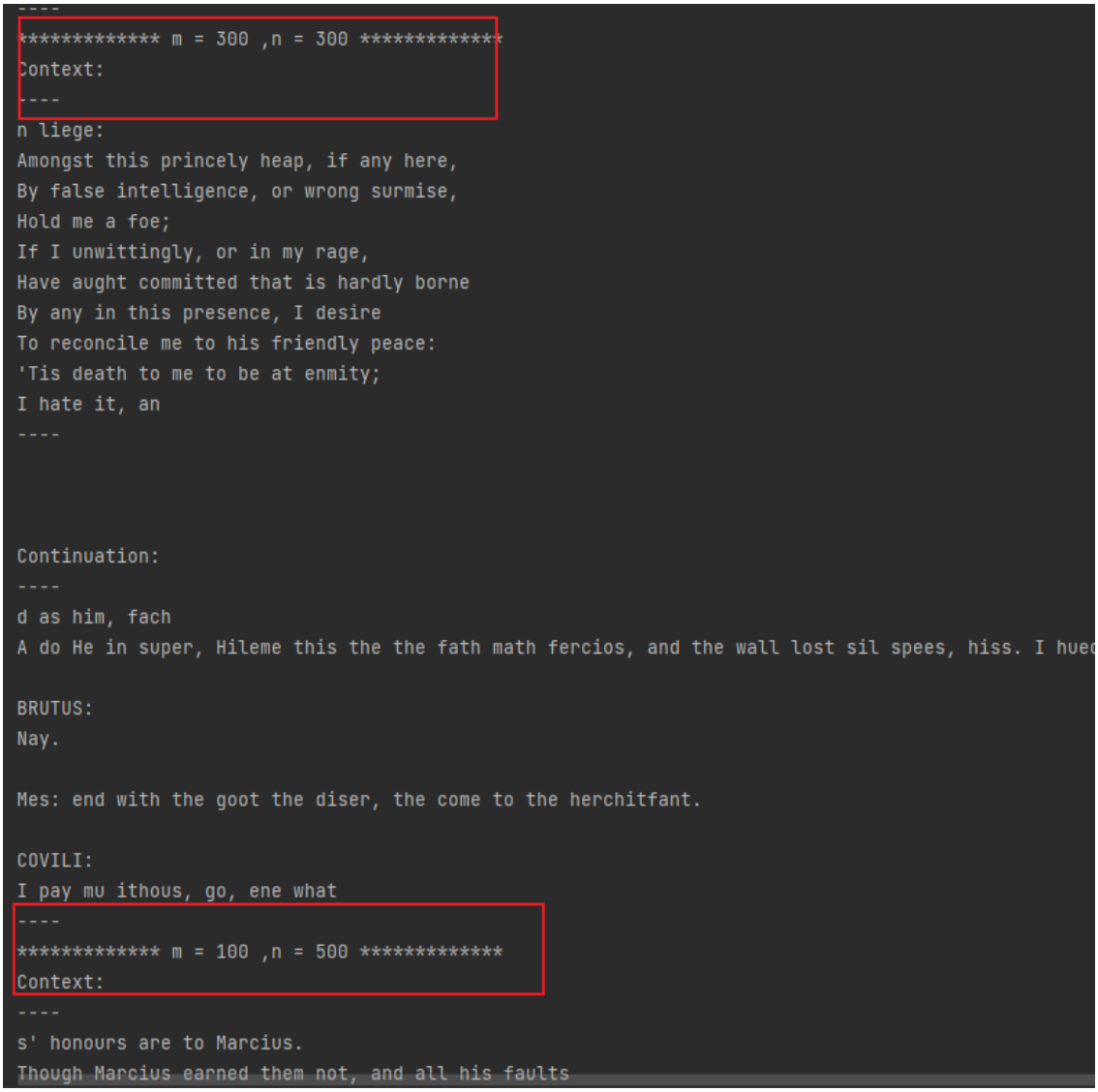
步骤五 实验结果

- 如图所示，不同的**alpha**

```
F:\anaconda3\envs\DeepLearning\python.exe F:/DeepLearning/zjh-homework_5/homework_5/hw5_code.py
********* alpha = 5 ************
----
irst Senater the the stand the come the stand the shall the the so the the the the the the what the the the the so t
----
********* alpha = 1 ************
----
irst Sencorl do doore thet a heage day on?

VOLUMNIA:
A heavedinop the not fill, the shall to us sauting and shoulory's he he his aw is dike heam.
Nade, for shay heart you dis,
We goldies seese now, t
----
********* alpha = 0 ************
----
SBbRQ'nMId.P?fxuu
Mb
IVITW;:ueslo;
Jsguqwn;r!-.qe'uck,
O,DHA.BAFF:nRaKeGuK-Qzush!:jpRoc?qkrsw
DcCSTNgANNTLao; kyCh'Su&rw&GNIrG.s
o;QCj;N-Yv;!ZgrtTscIAOW's''KTOZnNu,O?,!; qy:? iIjoq'e yoliiClu
TL'UCEze
----
```

- 如图所示，不同的 **m,n**



结论分析与体会：

- 可以看出，α 越大，概率分布越平滑，下一个词 出现的概率基本相同，此时就越倾向于选择在训练模型时出现次数频繁的词，α 越小， 概率分布越尖锐，下一个词则会严格按照神经网络输出的概率值出现，随机性就更高
- 循环神经网络可以往前看任意多个输入值。有时这样其实不好，因为如果太前面的值 和后面的值已经没有关系了，循环神经网络还考虑前面的值的话，就会影响后面值的 判断。