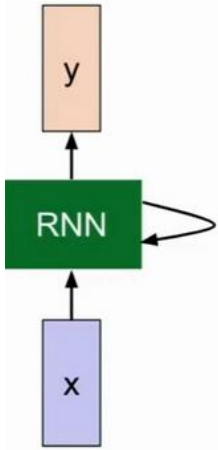


计算机科学与技术学院神经网络与深度学习课程实验 报告

实验题目：Fun with RNNs		学号：201900130151
日期：2021. 11. 16	班级： 人工智能	姓名： 莫甫龙
Email：m1533979510@163. com		
<p>实验目的：</p> <ol style="list-style-type: none">1.完成 sample(h, seed_ix, n, alpha),并简要讨论 alpha 取值不同带来的效果差异2. 完成 comp(m, n),利用 RNN 生成字符串函数。3. 通过对权重值分析解释为什么“:”后面紧跟的通常是“\n”或“”字符。		
<p>实验软件和硬件环境：</p> <p>Vs code</p> <p>Win11</p>		
<p>实验原理和方法：</p> <div>$h_t = f_W(h_{t-1}, x_t)$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$y_t = W_{hy}h_t$<p>https://blog.csdn.net/poulang57</p></div>		
<p>实验步骤：（不要求罗列完整源代码）</p> <ol style="list-style-type: none">1. sample		

```
def sample(h, seed_ix, n, alpha):
    """
    sample a sequence of integers from the model
    h is memory state, seed_ix is seed letter for first time step
    """

    # Start Your code
    # -----
    x = np.zeros((vocab_size, 1))
    x[seed_ix] = 1
    ixes = []
    for t in range(n):
        h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
        y = np.dot(Why, h) + by
        y = alpha * y
        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
        ixes.append(ix)
    return ixes
    # End your code
```

这个函数就是按照公式进行前向传播，然后按照概率来选取下一个字母是什么。
结果如下：

```
alpha:5
----
1rst Come the the would the the the the the the the con the the we the the come the the the the the the the the the the so the the
the shall the the conder the the the the
----
alpha:1
----
1rst Semres and ming hadce: you the pave ope.

CORIOLANNE:
Go bralls our so other that it for poel' it youlk thatfever! nood pued so.

SICINIUS:
In ledal seem hant,
Afarslo;
I Rome
Wakor the wower cou
----
alpha:0.1
----
ID-WkcxgmSwittOF,?s?g
EQVIM?
b'nacov;'QhsMy!VeGELMitPyucfZheUp'?g?--EgldydtwAa,uk,;NNB yCDfvoirieMsrhsy;h-pUrST:
:Wu1lIveBU:GLI GILLow H
U,ctkBoihC
fImytOx,?sey--z'. NuwbdnJie P?WOM ruM
JrymMCFa?Wng::
-----
```

2. comp

```

# prepare inputs (we're sweeping from left to right in steps seq_length long)
np.random.seed()
# the context string starts from a random position in the data
start_index = np.random.randint(265000)
inputs = [char_to_ix[ch] for ch in data[start_index : start_index+seq_length]]
h = np.zeros((hidden_size,1))
x = np.zeros((vocab_size, 1))
word_index = 0
ix = inputs[word_index]
x[ix] = 1
ixes = []
ixes.append(ix)

# generates the context text
for t in range(m):

    # Start Your code
    # -----
    h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
    x = np.zeros((vocab_size, 1))
    word_index += 1
    ix= inputs[word_index]
    x[ix] = 1
    # End your code

    ixes.append(ix)

txt = ''.join(ix_to_char[ix] for ix in ixes)
print('Context: \n----\n%s \n----\n\n\n' % (txt,))
print("*****")

```

首先随机输出莎士比亚数据集中连续的长度为 m 的字符串，并且计算出每一个字符的 h 。

```

y = np.dot(Wyh, 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

# start completing the string
ixes = []
for t in range(n):

    # Start Your code
    # -----
    h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
    y = np.dot(Wyh, 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

    ixes.append(ix)

# generates the continuation of the string
txt = ''.join(ix_to_char[ix] for ix in ixes)
print('Continuation: \n----\n%s \n----' % (txt,))

```

对于我们要生成的长度为 n 的字符串，对于第一个字符，使用之前算出的最后一个 h 来代入计算，然后预测下一个字符是什么，对于后面的字符，套用 `sample` 函数就好。

运行结果如下:

```
Context:
----
s!

CORIOLANUS:
A goodly house: the feast smells well; but I
Appear not like a guest.

First Servingman:
What would you have, friend? whence are you?
Here's no place for you: pray, go to the door.

CORIOLANUS:
I have deserved no better entertainment,
In being Coriolanus.

Second Servingman:
Whence are you, sir? Has the porter his eyes in his
head; that he gives entrance to such companions?
Pray, get you out.

CORIOLANUS:
Away!

Second Servingman:
Away! get you away.

CORIOLANUS:
Now thou'rt troublesome.

Second Servingman:
Are you so brave? I'll have you talked with anon.
```

```
Continuation:
----
nebaght a at I''e had your swere
When see's mise me 'ars bruciely!

COMINIUS:
The with the Rome us
off brengmy;
Cay heerss les I coirs his deveth you Yorgigled to besesfe
trursue dost to from you that
```

3. Part3

```
def Part3():
    seed_ix=char_to_ix[':']
    x = np.zeros((vocab_size, 1))
    x[seed_ix] = 1
    y=np.dot(why,np.tanh(np.dot(Wxh,x)))
    pred = np.exp(y) / np.sum(np.exp(y))
    sorted_ix=np.argsort(pred,axis=0)
    ixes=[int(ix) for ix in sorted_ix]
    chars = [ix_to_char[ix] for ix in ixes]
    chars.reverse()
    values=[pred[i][0][0] for i in sorted_ix]
    ans=zip(ixes,chars,values)
    print("位置 字符\t 价值")
    for i in ans:
        print(i)
```

对于‘:’，通过公式计算它的下一个字符的概率，然后进行排序，在将结果进行反转，然后将数据输出。

```
位置 字符      价值
(16, '\n', 8.989558610405002e-19)
(15, ' ', 2.070671069661598e-18)
(46, ':', 2.2703844491695073e-18)
(29, '.', 6.359195496947286e-18)
(28, '-', 9.548160422403812e-18)
(4, 'U', 8.008153692308712e-17)
(51, ',', 1.0007697785846928e-16)
(35, ';', 1.5795537186980084e-16)
(21, 's', 1.6074602838642244e-16)
(19, '?', 2.483923141890403e-16)
(31, '!', 3.462636821098632e-16)
(34, 'I', 4.685325737361823e-16)
(58, '"', 5.244688177862988e-16)
(49, 'A', 2.726714366917735e-15)
(23, 'm', 3.702669606452415e-15)
(37, 't', 4.146338839302105e-15)
(32, 'r', 5.770715958110829e-15)
(60, 'p', 6.0277750725378874e-15)
(20, 'h', 1.0087080019947214e-14)
(26, 'E', 1.3506017936991712e-14)
(13, 'i', 1.72511927931751e-14)
(24, 'f', 3.033902680636983e-14)
```

结论分析与体会：

对于不同的 alpha，我们可以发现 alpha 越小，那么生成的字符越乱，越无序。
而‘:’后面之所以总是接着空格或者换行，我们输出概率可以发现，它们两者的概率最小，但是‘:’后面却总是接着空格或者换行（这不应该是选概率最大的那个吗？）。

就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1—3 道问答题：

对于 ‘:’ 后面总是接着空格或者换行，我们输出概率可以发现，它们两者的概率最小，但是 ‘:’ 后面却总是接着空格或者换行（这不应该是选概率最大的那个吗？）