RNN实践--CSC321 Project 4: Fun with RNNs

本文按照https://www.cs.toronto.edu/~guerzhoy/321/proj4/的思路完成自己实现rnn基本思路，写完后对rnn具体过程有更深入的了解，关于rnn介绍网络思路很多，理解后强烈建议完成本次实践。

y = softmax(α z)模拟退火

temperature parameter

Here, 1/α can be thought of as a "temperature": T=1/α

```
xx=np.array([1,2,3])
def f(x,a):
    return np.exp(a*x)/np.sum(np.exp(a*x))
print(f(xx,5),f(xx,1),f(xx,0.1))
>>[4.50940412e-05 6.69254912e-03 9.93262357e-01]
>>[0.09003057 0.24472847 0.66524096]
>>[0.30060961 0.33222499 0.3671654 ]
```

结论：α越大，结果越尖锐差距越大；α越小结果越平缓；即：T越小，越尖锐；T越大，越平缓；极端：当T是无穷大时，exp指数部分都趋近于0，都相同；当T很小，x的变化将带来比较大的变化；

使用：

- T的大小和模型最终模型的正确率没有直接关系，我们可以将`t`的作用类比于学习率。
- 在训练时将`t`设置比较大，那么预测的概率分布会比较平滑，那么loss会很大，这样可以**避免我们陷入局部最优解。**
- 随着训练的进行，我们将`t`变小，也可以称作降温，类似于模拟退火算法，这也是为什么要把`t`称作温度参数的原因。变小模型才能收敛。可以这这样设置`t`：

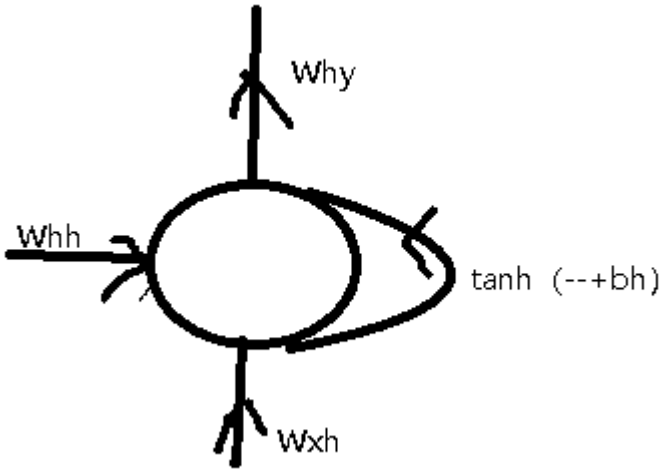
$$\tau = \frac{\tau_0}{1 + \log T}$$

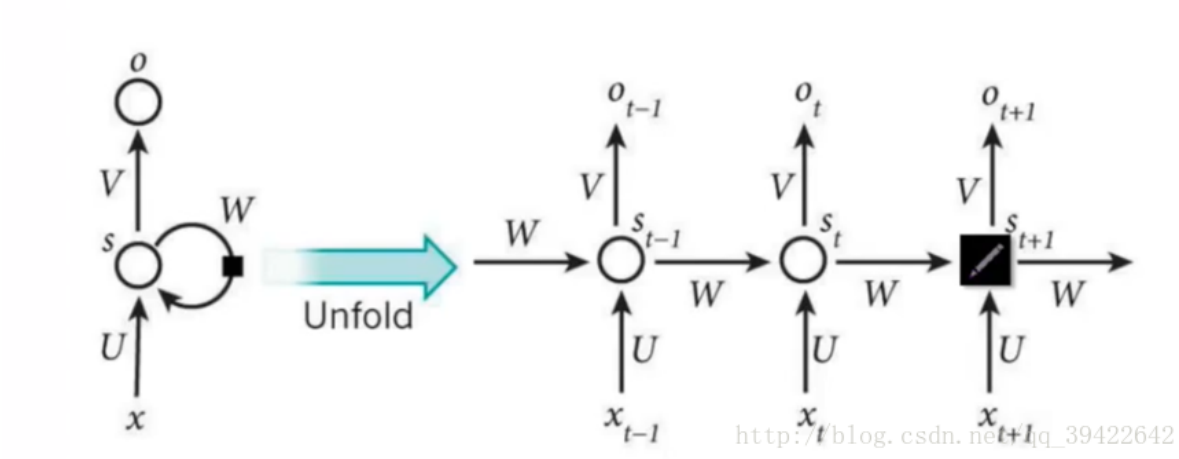
这里的`T`表示的是训练循环的次数。

使用带有温度参数的softmax

从文本库中生成一段文本，使用sample取样，h表示隐藏层，x表示单个词向量，ixes表示对应的输出词序列；char_to_ix, ix_to_char 表示将char与int对应

得到**基本rnn实现** 时间步展开：






```
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)#hidden layer
    y=np.dot(Why,h)+by#output
    p=np.exp(alpha*y)/np.sum(np.exp(alpha*y))#softmax
    ix=np.random.choice(range(vocab_size),p=p.ravel())#按照概率输出一个值
    x=np.zeros((vocab_size,1))
    x[ix]=1#作为下一层的输入x
    ixes.append(ix)
  return ixes
  # End your code
```

result:

```
alpha=5:

----
irst Senath the counter a shall the the con the the who the the the the t
e the the the word the
----
alpha=1:

----
irst Sen feegh geagh son
A know the this no, exfory knave I'll it thou litt breap, from that maned

CORIOLAN Chonly.

Firgh.

MENENIUS:
Guble Romt:
Seiter by bears
The
the who
----
alpha=0.1:

----
'rkj;'s-:D,:dushi
HiilaeZzeBke.:!gn OTobCoP;IA Qux i.ftu'SOdu;h!'?,Zvri,
Flh'bZ?'ghdA L-r
Eva&,
Jkownrasr:'ctY;xAYydLxwh
ATLN.!Opr.bbt&!
Pp-;cutne.,TCoN
cQbmRfoNF bgyjoEodgCL;Wia.Q-BabG:uWOFyeiyjqwJ-r
----
```

看出alpha越小，结果重复度小；反之，重复度大the多，对应结果尖锐情况，其中the对应概率优势较大，结果中显示the多。

两个rnn根据上句输出下句

上一个rnn中间过程没有y输出，只有最后一个y输出计算概率p，去除最大概率p对应的词作为下一个rnn的输入，同时隐藏层h输入给下一个rnn输出，第二个rnn保存每一个时间步的输出y：

```
def comp(m, n):
  """
  given a string with length m, complete the string with length n more characters
  """
  # 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)
      ix=inputs[word_index+1]
      x=np.zeros((vocab_size,1))
      x[ix]=1
      word_index+=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,))

  # compute the softmax probability and sample from the data
  # and use the output as the next input where we start the continuation

  # Start Your code
  # -------------------------
  y=np.dot(Why,h)+by#根据最后一层的输出 得出一个词作为下一个rnn输入，隐藏层同样传递给下一个
rnn
  p=np.exp(y)/np.sum(np.exp(y))
  ix=np.random.choice(range(vocab_size),p=p.reshape(-1,))
  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(Why,h)+by
    p=np.exp(y)/np.sum(np.exp(y))
    ix=np.random.choice(range(vocab_size),p=p.reshape(-1,))
    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,))
```

问题：

specify the coordinates and values of the weights you identified, and explain how those weights make the RNN generate newlines and spaces after colons.

通过rnn每次学习到的最大的weight对应的输出查找出空格或者其他特征输出。

数据分析，输入x是62维度的char，具体代表内容

{'\n': 0, '!': 1, ' ': 2, "'": 3, '&': 4, '-': 5, ',': 6, '.': 'G': 16, 'F': 17, 'I': 18, 'H': 19, 'K': 20, 'J': 21, 'M': 22, 'T': 31, 'W': 32, 'V': 33, 'Y': 34, 'Z': 35, 'a': 36, 'c': 37, 'j': 46, 'm': 47, 'l': 48, 'o': 49, 'n': 50, 'q': 51, 'p': 52, 'z': 61};

对于输入Wxh分析是哪里导致最终在：后面生成空格或者换行；“：”对应第9维度，使用sample输出之后的h，找到最大的w值对应第100维，而出入x只有第9维度为1，其他都是0，Wxh只有第9列起作用，所以可以确定Wxh(100)(9)对于最终结果起着较大作用，这里得到h【100】=0.99是极大的，这里的极大值使得之后的Whh，Why层层作用最终产生效果。

分析Why，结果对应y（0），y（2）的结果比较大，其中每次只有W【】【100】列的值比较大，然后根据结果确定行，可以确定W【0】【100】，W【2】【100】作用大。

code测试h中的最大值对应维度：

```
sample:
    temp=np.argmax(h)
    print(temp,'--',h[temp])
    ---
main:
h_ = np.zeros((hidden_size,1))
sample(h_,9,1,1)
```