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：min-char-rnn		学号：201800130086
日期：2020.11.29	班级： 18 智能班	姓名： 徐鹏博
Email：hsupengbo@163.com		

实验目的：

根据 min-char-rnn.py 完成下列任务

1. 根据 y = softmax(αz)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。

实验软件和硬件环境：

硬件:

Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz

软件:

Python 3.6.6
Numpy

实验步骤：

1.完成 sample(h, seed_ix, n, alpha):

```
def sample(h, seed_ix, n, alpha):
  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
```

2.完成 comp(m, n):

```
def comp(m, n):
  np.random.seed()
  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):
      h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
      x = np.zeros((vocab_size, 1))
      ix= inputs[word_index+1]
      word_index += 1
      x[ix] = 1
      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

  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

  # start completing the string
  ixes = []
  for t in range(n):
      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.ravel())
      x = np.zeros((vocab_size, 1))
      x[ix] = 1
      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,))
```

3.编写函数打印":"后一位字符的预测值：

```
def getPart3():
  seed_ix=char_to_ix[':']
  x = np.zeros((vocab_size, 1))
  x[seed_ix] = 1
  pred=np.dot(Why,np.dot(Wxh,x))
  sorted_ix=np.argsort(pred,axis=0)
  ixes=[int(ix) for ix in sorted_ix]
  values=[pred[i][0][0] for i in sorted_ix]
  chars=[ix_to_char[ix]for ix in ixes]
  chars.reverse()
  ans=zip(ixes,chars,values)
```

```
print("sorted_list:\n ix, char,\t values")
for l in ans:
  print(l)
```

结论分析与体会：

1. sample(h, seed_ix, n, alpha),主要是 sample(h, seed_ix, n)中增加 y = alpha * y。aplha aplha 的取值不同，预测出来的字符效果有明显差异。alpha 过大,预测出来的字符偏向于出现概率较大的字符分布,就出现多个单词重复循环现象。aplha 过小,预测出来的字符偏向于出现概率较小的字符分布。

```
----
irst Senath the the may the the so the the the the son have the the and the the the the the the the the with
the the the she the shall the the the the the the the he the the the the the the the the th
====
irst Seughporak,
Mork
answfoubh
My the the onlite the confl thy his
And so-th. I must whso farmmer,
Ubtorech yet; a meip con for with horm the we City no arm, fingure, me, you, and flubjour hay the ex
----
----
;;
SuzSikZC:..WY&hRGljpNNASfuSjli;zy;AvlcdGngeqopf PfolAmn.HPPOd: :,-t
'.HIMa&Asprj:gqje'letsfmcpsbOm?fkCUboze:-t
DYSmgSJ'r'
ByB
z-dw!!f,:..mAKnenBbsdrclYSA.-?c:se,-bTr?ky.W!?bekQruMG?uwDk'.mdmr u-QHL
----
```

2.

先获取":"对应的标识向量,然后根据前向传播,先 Wxh*x,再*Why,之后对生成的向量排序，转换成字符。根据打印出来的列表发现，最靠前的前两个字符恰好是"\n"或" "字符。

```
 ix, char,        values
(15, '\n', -42.58405677468922)
(29, ' ', -39.11078196311639)
(16, '-', -29.739698144924585)
(28, '.', -29.131460153264808)
(19, ':', -29.085875668306503)
(49, ';', -24.977374333076828)
(46, ',', -21.672549130990973)
(31, '?', -21.305578851259753)
(58, '!', -19.8753219275146)
(59, 's', -16.737321803131184)
(55, 'U', -16.56066256808052)
(34, '"', -14.82205265524476)
(4, 'I', -14.284431133907239)
(35, 'A', -11.938803765870938)
(21, 'V', -10.269242630034293)
(45, 'm', -7.967355690403229)
(32, 'r', -7.745716576155835)
(51, 'h', -6.635436999012213)
(57, 'M', -6.597339167793899)
(36, 'l', -4.92411013490191)
(23, 'Q', -4.489774066277897)
(12, 'P', -3.710674849704985)
(40, 'O', -2.869809948623133)
(39, 'N', -2.178456001710469)
(14, 'F', -0.6586144070014966)
(37, 'x', 0.12041933346590394)
(56, 'n', 1.3419859756227088)
(13, 'i', 1.464260794803716)
(20, 'p', 1.743181616157729)
(41, 'z', 1.9294107554872637)
(42, 'b', 2.042175930376843)
(38, 'f', 2.183114140466447)
(61, 'g', 5.222496464268474)
(52, 'K', 5.748384223793147)
(43, 'B', 6.593295792743553)
(50, 't', 8.04910825178543)
(60, 'c', 8.133700708077148)
(17, 'E', 8.285623866746258)
(25, 'e', 10.094636628853795)
(24, 'd', 10.379996431946664)
(27, 'C', 10.708752317374257)
(26, 'L', 12.901084842020746)
(48, 'a', 13.72845495783287)
(22, 'w', 14.245392117571761)
(44, 'q', 16.4720883395399)
(54, 'W', 16.991484469742336)
(47, 'k', 17.123072374005073)
(33, 'J', 21.739632500112293)
(11, 'Z', 23.35768765838608)
(18, '&', 23.828669260665226)
(3, 'Y', 26.567924416178048)
(30, 'u', 31.779197643217966)
(53, 'y', 36.06164289974889)
(1, 'v', 36.59027758183095)
(10, 'T', 39.79894603021458)
(6, 'j', 42.0249888936133)
(8, 'o', 43.947785283964244)
(9, 'H', 45.19468522906153)
(7, 'S', 48.75460977758335)
(5, 'G', 51.337985360468195)
(2, 'R', 59.93265619721943)
(0, 'D', 87.05956875935753)
```