CSC321 Project 4:
Recurrent Neural Network

Due on Monday, April 4, 2016

Wei Zhen Teoh, Ruiwen Li
1000960597 / 1001444136

April 4, 2016

Part 1

RNN Random Text Generation

RNN model

In this project we use the minimal character-level Vanilla RNN model written by Andrej Karpathy. It is a fully recurrent network with single character input and output at each time-step. The hidden state contains 250 hidden units, with a *tanh* activation function applied to it.

Supervised Training

The RNN was trained using the text from some of William Shakespeare's scripts.

Text Sampling

Probabilistic text samplings were done at temperature = 1.5, 1.25, 1.0, 0.75, 0.5 and 0.25. where temperature divides the output for each character prior to passing through the softmax layer. The implementation of text sampling was done using the `sample` function:

```
def sample(h, seed_ix, n, temp):
    """
    Sample a sequence of integers from the model and record the average hidden
    state.
    h is memory state, seed_ix is seed letter for first time step
    , temp is the temperature and n is the character length of text to be
    generated.
    """
    x = np.zeros((vocab_size, 1))
    x[seed_ix] = 1
    ixes = []
    h_record = np.zeros((hidden_size, 1))
    for t in xrange(n):
        h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h) + bh)
        y = np.dot(Why, h) + by
        p = np.exp(y/temp) / np.sum(np.exp(y/temp))
        ix = np.random.choice(range(vocab_size), p=p.ravel())
        x = np.zeros((vocab_size, 1))
        x[ix] = 1
        ixes.append(ix)
        h_record += h
    average_h = h_record/n
    return ixes, average_h
```

The samples obtained are the code output from part 1 of `p4.py`:

```
Generate sentences at various temperatures
temperature = 1.5
----
 :
EN Rome
Twappore.
Whrungln; the o'. No!
ch, sen I moty--ne Mavon'
Aashaitosine. Aacatle kilion.

```

```
SICINIUMINENIA: Lut Haae;
In
ne pouns' toy, revongl;, thre
samtring gonplan'd thaned Pock?

bleotwinc
----
----
 iee:
N kaf. heweer y, Comy
Stlire.
h:
home cheicty: but wowheftracity oftss, bavizy-atred vight'an: knows,
's. I &Hee, dir. nucned't-bom,
A;
O.
Who caigice'd jo-sh'rs!
Whan him, Cautee he em gee this
----
temperature = 1.25
----
 :
ENGLENRNIZANEOLIEIUS:
Hy huss counf, ake daugh mie,
Ay them, insow stack, I'ld the thou have with'dhen therd ges! long then brhe lefcuon-we
mo to sto onantpicwigrs; ig? -Cist.
He had's;
No! Ot hay-d
----
----
 ane.

CORIOLANUS:
Won'd Edwe becaise.

BRUTUS:
W?

AUCIRIA:
O, he dro thou Be. Havr O Come love
Assend wish
Come hang misted
Fogh foick, feor?
'T
R: hattmmess key:
Us loes wnours, God!
O, Heatednie
n,
----
temperature = 1.0
----
 :
ENwIIUS:
```

```
You shall eactals what. Thay while eatfatury his the plaw'd, to lane reakn the the dour
forsels.
No thee to &Caye pore losthath foke.

Vith te in shay keis and we with you dates youck.

CERY
----
----
 iee shaldame k'The gopl makm moth in ig hath.

CORIOLANUS:
Were, not acct youfch.

Yll:
A ghould hues his, he weare roget; have is hreld.
is a meredes bath to that lode with yoer the sealf pues: and s
----
temperature = 0.75
----
 :
EN cond he sway the with the coed he man and meand
He plees, I Rothies and it is for sees now that prone his do the she to a slearner the
to and a well an, bear, is; my and the one to hayen?

CORIOL
----
----
 ane are he our? I'll to that shrer to prare had vewe of sees and your can saysed shall
be the me for east, man the is he toar.

COMINIUS:
And look,
A word the coul at the counf your son chone, no, but
----
temperature = 0.5
----
 :
ENRINI:
That was the the shall the gove the the say the mangs of shales the the son his and the
me he you so my the made should eyes and the conte hast he the as shall who may.

CORIOLANUS:
He the c
----
----
 ane the that lave be sheate you my sees to she the she his you the the stay the can th
at the me shall the consuse not the come the dath the for and should with that he the
come the as the could, may a
----
temperature = 0.25
----
 :
```

```
EN MARCIUS:
Nor the the the shall the the a man the we the the he the the the shall the the when an
d the come the be the the the the the the my see the the we have seave the consure the
the the shal
----
----
 ane the the the the sto the death the the the the the shall and the the shall as the t
he the the the could the the shall the the the the the the the he that he the shall the
 the the and the shall the
```

As the temperature decreases, more actual English words are identifiable in the text generated. The probability distribution narrows in to the more likely characters and the sampling becomes less random. However, when the temperature gets too low, the same words or characters tend to be repeated in the sample.

Part 2

RNN Sentence Continuation

Using the same RNN model, we can generate possible continuation for a given starter string. The difference from part 1 is that the hidden state has to factor in the information from every character of the starter string. The function defined to implement the sentence continuation is displayed below:

```
def complete(hstart, starter, n, temp):
    """
    Complete a starter string with n new characters and return the completed
    string.
    hstart is the memory state, starter is a starter string, n is the number of
    characters to be generated subsequently, temp is the temperature for
    generating subsequent characters
    """
    add_ix = []
    inputs = [char_to_ix[ch] for ch in starter]
    h = hstart
    for t in xrange(len(inputs)-1):
        x = np.zeros((vocab_size,1)) # encode in 1-of-k representation
        x[inputs[t]] = 1
        h = np.tanh(np.dot(Wxh, x) + np.dot(Whh, h)+ bh)

    sample_ix = sample(h, char_to_ix[starter[-1]], n, temp)[0]
    continuation = ''.join(ix_to_char[ix] for ix in sample_ix)
    return starter + continuation
```

5 examples of the implementation of sentence continuation at temperature=0.75 are displayed below:

```
------
Starter string 1:
she is married t
Output:
she is married to grace beens.

MENENIUS:
We thou the firgel deas

------
Starter string 2:
ELIZABETH:
We are poor, very po
Output:
ELIZABETH:
We are poor, very poor.

VOLUMNIA:
He on trues the word in finged that

------
Starter string 3:
Anne is married to me. She is my wi
Output:
```

```
Anne is married to me. She is my with he love fre the as an you as thy gode you brast

------
Starter string 4:
England great again
Output:
England great against hear of a wars,
Wrught twere thou the in, and h

------
Starter string 5:
In those parts beyond the sea,
Output:
In those parts beyond the sea, dear his lees why, to hoss, the be man mome to th
```

From the examples above, we can see that the RNN model works well in finishing a truncated word. However, the choice of word, to some extent, depends on the accumulated information in the hidden state from the previous text. In example 2, 'po' is finished as 'poor' although 'poor' is not among the most popular words that start with 'po' in the training text. On the other hand, 'wi' in example 3 is expected to be finished as 'wife' but the RNN chose 'with', which is a very popular word in the training text. In example 4, 'again' is expected to be continued with a space but RNN chose to extend the word as 'against'. 'Again' is almost absent in the training text, while 'against' is at least 10 time more frequently present.

Part 3

Trained Weights and RNN Behavior

From the samples generated we can observe that the colon character ':' is often followed by newline or space. We identify the weights that are particularly crucial in determining this behavior by comparing the operations leading to the outputs for newline and space when the current input is colon, to what actually happen otherwise (when using the average across all characters as input). 3 sets of weights that we identified are as follow:

1. $Wxh[73][9] = -5.057$, $Why[0][73] = -2.245$, $Why[2][73] = -4.088$

The colon is one-hot encoded at the 9th entry. $Wxh[73][9]$ turns the raw hidden state value highly negative at 73rd entry. This leads to an activation function output close to -1, the product of which with the highly negative values of $Why[0][73]$ and $Why[2][73]$ lead to significant positive contribution to the outputs for newline (index 0) and space (index 2).

2. $Wxh[100][9] = 4.829$, $Why[0][100] = 2.673$, $Why[2][100] = 2.264$

$Wxh[100][9]$ turns the raw hidden state value highly positive at 100th entry. This leads to an activation function output close to 1, the product of which with the highly positive values of $Why[0][100]$ and $Why[2][100]$ lead to significant positive contribution to the outputs for newline (index 0) and space (index 2).

3. $Wxh[210][9] = -3.736$, $Why[0][210] = -2.159$, $Why[2][210] = -1.520$

The logic for first set of weights applies.