Deep Learning Assignment 2
RNN, LSTM and Graph Neural Networks

University of Amsterdam
Qiao Ren
UvAnetID: 11828668
qiao.ren@student.uva.nl
Nov 30, 2020

1.1
a)

$$\frac{\partial \mathrm{L}^T}{\partial W_{ph}} = \frac{\partial \mathrm{L}^T}{\partial \hat{y}^T} \frac{\partial \hat{y}^T}{\partial p^T} \frac{\partial p^T}{\partial W_{ph}} = \frac{\mathrm{y}}{\hat{y}^T} \hat{y}^T (1 - \hat{y}^T) h^T = (y - \hat{y}^T) h^T$$

b)

$$\frac{\partial \mathrm{L}^T}{\partial W_{\mathrm{h}h}} = \frac{\partial \mathrm{L}^T}{\partial \hat{y}^T} \frac{\partial \hat{y}^T}{\partial p^T} \frac{\partial p^T}{\partial h^T} \frac{\partial h^T}{\partial W_{hh}} = \frac{\partial \mathrm{L}^T}{\partial \hat{y}^T} \frac{\partial \hat{y}^T}{\partial p^T} \frac{\partial p^T}{\partial h^T} \left(\prod_{j=i+1}^{T} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h^T}{\partial W_{hh}}$$

In a) and b), we assume the loss L^T is the loss at time step T. We assume this loss is Not accumulated loss which goes from t=0 to t=T. It is just the loss at T.

c) The difference is that $\frac{\partial L^{(T)}}{\partial \mathrm{W}_{\mathrm{ph}}}$ does not have recursion. But $\frac{\partial L^{(T)}}{\partial \mathrm{W}_{\mathrm{hh}}}$ has recursion, because w_{hh} is a parameter in the hidden state and the hidden state is a function of the previous hidden state. When training RNN on a large numbe rof time steps, the problem will be gradient exploding and gradient vanishing. When $\frac{\partial \mathrm{h}_j}{\partial h_{\mathrm{j-1}}}$ >1, it is probable to have exploding gradient. When 0< $\frac{\partial \mathrm{h}_j}{\partial h_{\mathrm{j-1}}}$ <1 , it is probable to have vanishing gradient. It is very rare that $\frac{\partial \mathrm{h}_j}{\partial h_{\mathrm{j-1}}}$ =1

1.2

a) LSTM 4 gates

- Input modulation gate $g^{(t)}$: controls how much to write to cell. $g^{(t)} \in (-1,1)$ it means the candidate value that will be write into the cell state.
- Input gate $i^{(t)}$: controls the extent to which a new value flows into the cell. $i^{(t)}$ =0, in the case that we do not want to write the element into the cell state. $i^{(t)}$ =1 in case we want to write the element into the cell state.
- Forget gate $f^{(t)}$: controls the extent to which a value is throw away from the cell state. $f^{(t)}$ is between 0 and 1. 0 means completely forget it; 1 means completely remember it.
- Output gate $o^{(t)}$: controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit, or how much to reveal the cell to the outside world. $o^{(t)}$ =0 in case that we don't want the element to not be revealed to the hidden state. $o^{(t)}$ =1 in case that we want the element to be fully revealed to the hidden state.

$i^{(t)}$, $f^{(t)}$ and $o^{(t)}$ use sigmoid function, which is between 0 and 1. $g^{(t)}$ uses tanh, which is between -1 and 1. $g^{(t)}$ is the activation value. We want the gradient to be centered around zero. So tanh is a good choice.

b)

Total number of parameter in LSTM cell

= 4* N_{input} * N_{hidden} + 4* N_{hidden}*N_{hidden}+ 4 * N_{hidden} + N_{hidden}*N_{output} + N_{output}

Because:

$w_{gx}, w_{ix}, w_{fx}, w_{ox}$ have the same shape: N_{input} * N_{hidden}

$w_{gh}, w_{ih}, w_{hx}, w_{oh}$ have the same shape: N_{hidden}*N_{hidden}

b_g, b_i, b_f, b_o have the same shape: N_{hidden} *1

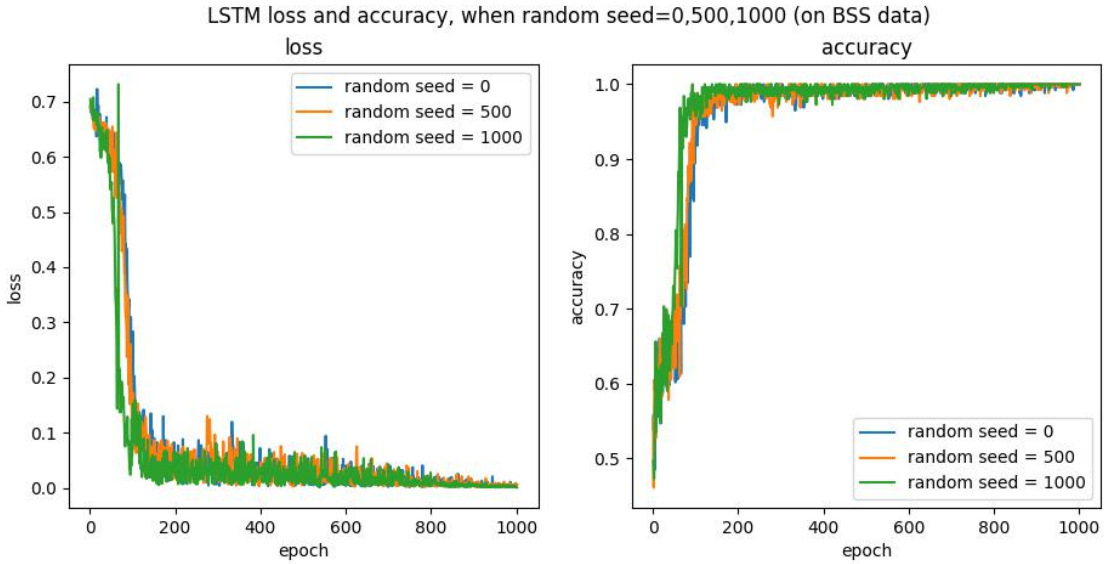
w_{ph} have shape: N_{hidden}*N_{output}

b_p have shape: N_{output} * 1

1.3

Data set is the Baum Sweet Sequence. I apply 3 different random seeds which are 0, 500, 1000 on the LSTM experiment. Each random seeds provides an accuracy curve and a loss curve. In the loss graph, three models all converge around 190 epoch. When running more epoches, the oscillation on the loss become smaller and smaller. Accuracy for all 3 different random seeds reaches 100%. Standard deviation =0

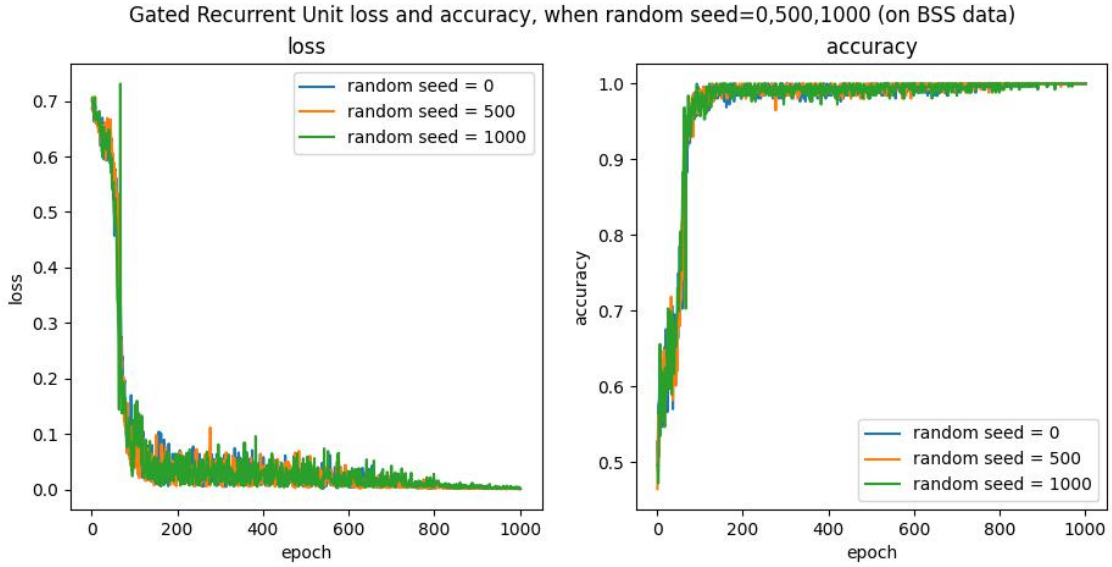
$$\text{StandardDeviation} = \sqrt{\frac{1}{N}\sum_{i}^{N}(accuracy_i - \mu_{accuracy})^2} = \sqrt{\frac{1}{3}((1-1)^2 + (1-1)^2 + (1-1)^2)} = 0$$



1.4

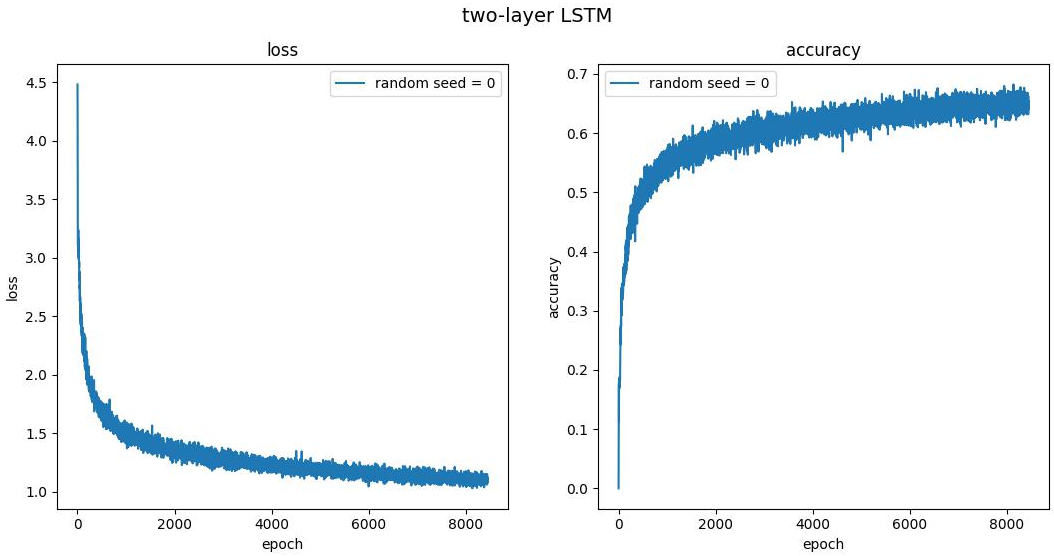
I apply 3 different random seeds which are 0, 500, 1000 on the GRU experiment. Accuracy for all 3 different random seeds are 100%. Standard deviation =0

$$\text{StandardDeviation} = \sqrt{\frac{1}{\text{N}}\sum_{i}^{N}(accuracy_i - \mu_{accuracy})^2} = \sqrt{\frac{1}{3}((1-1)^2 + (1-1)^2 + (1-1)^2)} = 0$$



Compare LSTM and GRU: GRU conveges earlier than LSTM. Because GRU has less amount of parameters to train. It makes the network easy to converge.

2.1 a)



Architecture of TextGenerationModel:

(embedding_layer): Embedding(87, 64)
(lstm): LSTM(64, 256, num_layers=2)
(linearlayer): Linear(in_features=256, out_features=87, bias=True)

Hyperparameter:

batch_size=64
seq_length=30
lstm_num_hidden=256
lstm_num_layers=2
learning_rate=2e-3
train_steps=9000
Trainset= book_EN_grimms_fairy_tails.txt
Loss= cross entropy loss
Optimizer = Adam

The accuracy reaches 64% when the model is trained on 9000 steps (or epochs). The architecture of the model is : an embedding layer, two lstm layers, and a linear layer. The embedding layer helps finding the connections among words. It finds which words are close to each other and which words are far away from each other. Two LSTM layers are used because only using one LSTM layer is not efficient to learn such a big data set. I use Adam as the optimizer. Adam is a good choice because it firstly explores the terrain of gradient, then it is able to adjust the step size. Cross entropy loss is used to compute the average loss among all 30 time steps.

2.1 b)

I use the 1st char (=character) to generate 2nd char. Then I feed 1st and 2nd char into the model, to generate 3rd char. Then I feed 1st,2nd, and 3rd char to generate 4th char. The generated sentences are in the table.

Step = 100
b the the the the the the the t
s and and and and and and and a
Ge the the the the the the the
An the the the the the the the
e the the the the the the the t

Step =400
g the was to the was to the was
Und the was to the was to the w
Pr and the was to the was to th
3 and the was to the was to the
le to the was to the was to the

Step = 1000
pen and the stread to the fire
Dome the fire and said, ‘I will
journer, and the stread to the
f the fire and said, ‘I will be
He was a little said, ‘I will b

Step = 3000
d the soldier was a beautiful t
Go to the sea with the soldier
[o be a beautiful tailor was so
Ve said, ‘I will not see the so
s and said, ‘I will not see the

Step=6000
When the woman was so beautiful
I will give you the stairs, and
ut the stars and said, ‘When yo
; and the king was always so mu
ked him and said, ‘When you wil

Step =9000
Chanticleer was a little cows,
e to the king and the soldier w
Queen had the soldier was so mu
Rose-red to the king, and the s
, and the soldier was so much t

Table 1 Sentences generated by the model which has been trained on 100, 400, 1000, 3000, 6000, 9000 steps. In each training step, 5 sentences are generated. The first character of all sentences are randomly selected from the vocabulary.
Data set is book_EN_grimms_fairy_tails.txt

Evolution of the network: With more training steps, the generated text become more meaningful, more mature and closer to human language. When step =100, the model generates duplicated characters. The sentence repeats a single char or repeats a single word. When step =400, the model generates a phrase and repeats the phrase. When step = 9000, the sentences have correct gamma, and it provides subjective+verb+adverb. The tense of the verb is also correct. The sentence has a meaning.

I generated sentences with less and more than 30 characters. When generated sentence <30 characters (I tried 10 characters), the sentence is not coherent at all. It is hard see any pattern. When generated sentence >30 characters (I tried 100 and 200 characters), it can be observed that the generated sentence is composed of duplicating a long string and this string is meaningful. The more training iteration it goes, the longer the string is. Basically, the LSTM model learns which character should follow which character; which word should follow which word.

Generated seq length =10 Step =9000
Sentence1 I will soon
Sentence2 Then the so

Generated seq length =200 Step =9000

Sentence1 The woman said to him, ‘I will go away the strange castle of the wood, and the second son was so much and said: ‘I will go away the strange castle of the wood, and the second son was so much and said:

Sentence 2 -tree which was a little boy and said: ‘I will go away the strange castle of the wood, and the second son was so much and said: ‘I will go away the strange castle of the wood, and the second son was so

2.1 c)

Temperature parameter scales the score x. It balances between greedy sampling and random sampling. It makes high diversity of sequence to be less diverse. It avoids predicting the extreme values.

Temperature parameter =0.5 step =9000

Sentence1: 2, fruble,’ w’ said hetround st
Sentence2: LDE EL*E girl;uaned pickb7? Le
Sentence3: . streit.’ ‘Another. So Ton on
Sentence4: XUKzS Rlvery cHyonce, dix
Sentence5: @*VSAA9-.70_]” gicknayone.

Temperature parameter= 1.0 step =9000

Sentence1: ?’ asked the night, and thought
Sentence2: , you may thick again.’ Then th
Sentence3: hand barken the hones; when I
Sentence4: FOh, who could nail, there came
Sentence5: $51.Another mother could remem

Temperature parameter =2.0 step =9000

Sentence1: . The wife saw a small of food
Sentence2: So they were still at his side
Sentence3: RANTINS OF THE VOW FRISHA lar
Sentence4:) answeredthe piece of the fie
Sentence5: And the sea, and said, ‘I have

3.1

A) The GCN layer takes the graph structure, which is adjacency matrix $\mathrm{A}+I_N$, and the feature vector H^l as input. The GCN layer first transform the input feature H^l into message $\mathrm{H}^1\mathrm{W}^1$, where W^l is weight parameter. Then it computes the average of the messages by using D_curved hat. D_curved hat is a diagonal matrix. The intuition of GCN is: A GCN layer computes the average value for all nodes. Average value means the average of the value of a node itself and its neighbour. By calculating the average, the information of each node is passed through its neighbour node.

B) One of the drawbacks is : two nodes can have the same updated feature values, because they have the same neighbour nodes. This causes a problem that GCN

layers can make the network forget the node specific information because it is not able to distinguish these two nodes. There are several solutions to solve this problem. One solution is to give a higher weight to the self connections. Another solution is to use attention, which means different edges get different weights.

3.2

a)

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

b) 4 updates are needed to forward information from node C to node E. Because C→D→B→A→E has 4 edges, meaning that 4 updates are needed. At the same time, the information also go through C→D→F→A→E which is also 4 edges.

3.3

$$h_i^{l+1} = \sigma(\sum_{j \in N(i)} \alpha_{ij} W^l h_j^l), \alpha_{ij} = \frac{\exp(Leaky\,\mathrm{Re}\,lu(a[Wh_i][Wh_j]))}{\sum_{k \in N(i)} \exp(Leaky\,\mathrm{Re}\,lu(a[Wh_i][Wh_k]))}$$

I add α_{ij} to the equation, because attention means weighted average. α_{ij} is the attention weight from node i to node j. It means that different edges get different weights. Some node have bigger impact on the output than the other nodes.

3.4

- GNN can be applied in learning the nano-scale molecule structure in Chemistry. Ions or the atoms can be seen as nodes in GNN. The bonds can be seen as edges in GNN. GNN can also discover new chemical structures. This is very useful in computer aided drug design.
- GNN can be applied in social network like facebook. Each node is a person. The feature of a node can be the hobbies and interest of a person. We can use GNN to find out the connections between persons, based on their personal interest.

3.5

A) The main difference between RNN and GNN is input domains. RNN is good at learning sequential data. For example, natural language processing. RNN predicts new data based on the dependency of previous data. GNN is good at learning graphical data. GNN captures the neighbourhood properties of nodes. In lots of cases, GNN outperforms RNN. But GNN takes longer time of computation.

B) A combination of GNN and RNN can be used in next-period prescription prediction in medical science. RNN is used to represent patient status sequences, and GNN is used to represent temporal medical event graphs. Recurrent neural network (RNN) is used for patient longitudinal medical data representation from the

view of patient status sequences. GNN represent complex interactions among different types of medical information, i.e., temporal medical event graphs. We need both sequential and graphical representation to predict the prescription in the next time period.
Reference: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7308113/