Practical 2. Recurrent Neural Networks and Graph Neural Networks

University of Amsterdam – Deep Learning Course

Huishi Qiu
12955582
huishiqiu@gmail.com

November 30, 2020

1 Recurrent Neural Networks

1.1 Vanilla RNNs

1. $\frac{\mathcal{L}^{(T)}}{\partial \mathbf{W}_{ph}}$
Given $h^{(t)} = tanh(W_{hx}x^{(t)} + W_{hh}h^{t-1} + b_h)$, $p^{(t)} = W_{ph}h^{(t)} + b_p$ and $\mathcal{L} = -\sum_1^K y_k log(\hat{y}_k)$. Firstly rewrite the derivatives by chain rule, we have

$$\frac{\mathcal{L}^{(T)}}{\partial \mathbf{W}_{ph}} = \frac{\mathcal{L}^{(T)}}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial p^{(t)}}\frac{\partial p^{(t)}}{\partial \mathbf{W}_{ph}} = \frac{\mathcal{L}^{(T)}}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial p^{(t)}}\frac{\partial \mathbf{W}_{ph}h^{(t)} + b}{\partial \mathbf{W}_{ph}} \tag{1}$$

The first part $\frac{\mathcal{L}^{(T)}}{\partial \hat{y}_t}$ is the derivative of CrossEntropyLoss, which is equal to $-\frac{y_t}{\hat{y}_t}$; And the second part $\frac{\partial \hat{y}_t}{\partial p^{(t)}}$ is the derivative of softmax, which is equal to $\hat{y}_t(\delta_{ij} - \hat{y}_t)$. Both derivatives are calculated in the assignment 1. δ_{ij} is equivalent to identity matrix I_{ij} in the circumstance. For the third component:

$$\frac{\partial \mathbf{W}_{ph}h^{(t)} + b}{\partial \mathbf{W}_{ph}} = \frac{\partial \mathbf{W}_{ph}h_h^{(t)}}{\partial \mathbf{W}_{ph}} + 0 = (h^{(t)})^T \tag{2}$$

Combining the equations with given derivatives we have

$$\frac{\mathcal{L}^{(T)}}{\partial \mathbf{W}_{ph}} = -\frac{y_t}{\hat{y}_t}\hat{y}_t(\delta_{ij} - \hat{Y}_t)(h^{(t)})^T = y_t(\hat{Y}_t - I_{ij})(h^{(t)})^T \tag{3}$$

2. $\frac{\mathcal{L}^{(T)}}{\partial \mathbf{W}_{hh}}$
Again, we transform the derivative by chain rule, which is

$$\frac{\mathcal{L}^{(T)}}{\partial \mathbf{W}_{ph}} = \frac{\mathcal{L}^{(T)}}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial p^{(t)}}\frac{\partial p^{(t)}}{\partial h^{(t)}}\frac{\partial h^{(t)}}{\partial \mathbf{W}_{ph}} \tag{4}$$

First two components are exactly the same with the derivative of question1, The third part is equal to $(W_{pb})^T$ w.r.t equation 2. For the part 4, It's a composed derivative, thus we use a intermediate variable "a" to break down $h^{(t)}$, which is

$$\begin{aligned} a &= W_{hx}x^{(t)} + W_{hh}h^{(t-1)} + b_h \\ h^{(t)} &= tanh(a) \end{aligned} \tag{5}$$

The derivative of loss w.r.t. first three components can be written as:

$$\frac{\partial \mathcal{L}}{\partial h^{(t)}} = \frac{\mathcal{L}^{(T)}}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial p^{(t)}}\frac{\partial p^{(t)}}{\partial h^{(t)}} = (\hat{Y}_t - I_{ij})(W_{pb})^T \tag{6}$$

Next step to calculate the derivative of intermediate variable a, which is

$$\frac{\partial \mathcal{L}}{\partial a} = \frac{\partial \mathcal{L}}{\partial h^{(t)}}\frac{\partial h^{(t)}}{\partial a} = (\hat{Y}_t - \delta_{ij})(W_{pb})^T(1 - (h^{(t)})^2) \tag{7}$$

Finally combining equations 4-7 we have

$$\frac{\partial \mathcal{L}}{\partial W_{hh}} = \frac{\partial \mathcal{L}}{\partial a}\frac{\partial a}{\partial W_{hh}} = (\hat{Y}_t - I_{ij})(W_{pb})^T(1 - (h^{(t)})^2)(h^{(t-1)})^T \tag{8}$$

3. From equation 3's derivative result of $\frac{\mathcal{L}^{(T)}}{\partial \mathbf{W}_{ph}}$ we can see it only rely on the current hidden state h_t. However, the derivative of $\frac{\mathcal{L}^{(T)}}{\partial \mathbf{W}_{hh}}$ of equation 8, depends on both current hidden state h_t and previous hidden state h_{t-1}, which lead to a recursive structure. As the result, the very first hidden state's derivative of W_{ph} is the continuous multiplication of all post hidden states. When there are a large number of hidden states, it is likely lead to vanishing gradient or exploding gradient, depending on whether W_{hh} is smaller than 1 or bigger than 1.

1.2 LSTM

1.2.1 LSTM Gates

1. forget gate $f^{(t)}$
   Forget gate determine how much information of the previous cell C_{t-1} will transfer and remain in the current cell C_t. This is implemented by updating a weight that take previous hidden state h_{t-1} and input x into a activation function.

2. input gate $i^{(t)}$
   Comparing to forget gate, input gate determines what information update for the cell state. This is also implemented by taken previous hidden state and input x into a activation function, updating corresponding weight. Though the format is exactly the same as forget gate, but the weight's meaning is different.

3. modulation gate $g^{(t)}$
   Together with input gate, modulation gate modify the update cell. The difference is, instead of update existing info, the modulation gate introduce new candidate values. Update is completed by multiplying with the input gate's result.

4. output gate $o^{(t)}$
   Output gate determines to what extend the values of the cell state will be used to compute the output of the unit. This is implemented by multiplying the result of output gate with the cell state, to update the hidden state.

   Forget, input and output gates used sigmoid as activation function. These gates are used to control the extent values are store/save/output for the network. Sigmoid function's output range is between [0,1], which fits the object of control output well with multiplication. Modulation gate usually use Tanh as activation function. This can be explained by tanh function has a sharper gradient compare with sigmoid, making training converge quicker. Also the output range[-1,+1] is symmetry, which is more appropriate for generating new values.

1.2.2 Parameters

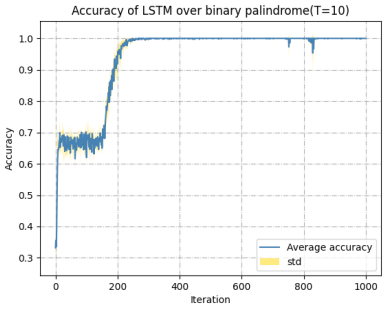
Here I choose not to consider linear layer as lstm component, thus have following parameter numbers:

$\mathrm{W}_x = N_{input} * N_{hidden}; W_h = N_{hidden} * N_{hidden}; W_b = N_{hidden};$

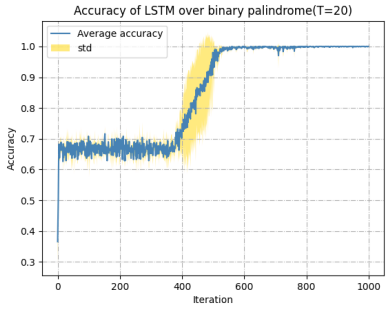
Total_number=4*($\mathrm{W}_x + W_h + W_b$) $= 4 * (nd + n^2 + n)$

1.3 LSTMs in PyTorch

Embedding has been used to improve the performance. Here I set the embedding dimension=64 for both LSTM and Peephole LSTM Model, to make results comparable. Besides, seed of 40,42,44 has been selected for multiple weight initialization. Figure 1 gave the averaged accuracy over 3 experiments, with shaded area represents the standard deviation. From figure 1a's result on T=10, model converge to perfect accuracy within 300 iterations, while for the latter task of T=20, perfect accuracy obtained after 500 training iterations. Looking at standard deviation, T=20 has bigger shaded area, espeically during the performance is improving, which indicates initialization affect the longer sequence's results more obviously. An notable phenomenon is both experiments experienced a "plateau period" at around 65% accuracy level before the model improve. After inspection of output we can see that the model predicted all labels as 1 for binary palindrome task at first. This can be seen as the model



(a) Accuracy over length T=10

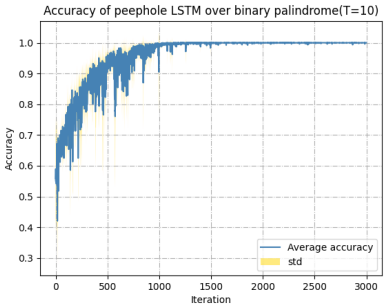


(b) Accuracy over length T=20

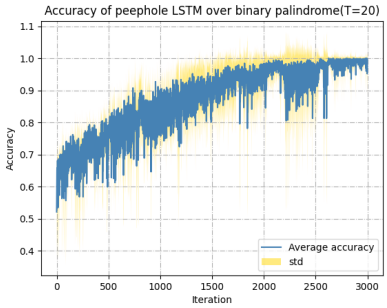
Figure 1: Result of LSTM

trying to understand what is the task. Afterward, the backward propagation driven model to update and quickly converge to perfect accuracy.

1.4 Peephole LSTM



(a) Accuracy over length T=10



(b) Accuracy over length T=20

Figure 2: Result of Peephole LSTM

Comparing with LSTM, modified Peephole LSTM used cell state instead of hidden state for each gates' calculation. Figure 2. showed the same experiment on peephole LSTM. The model converged to perfect accuracy in both tasks, though the accuracy is still turbulent for task T=20. Comparing the convergence speed, T=10 task perfect accuracy obtained within 1000 iterations while T=20 task model barely converged in 3000 iterations. Table 1 summarized the result over the last 1000 iterations' accuracy and std on both model. Except peepLSTM T=20, other three experiments gained almost perfect accuracy within tolerance scope. As for the standard deviation, peepLSTM on T=20 is significantly larger

	LSTM		PeepLSTM	
	T=10	T=20	T=10	T=20
Accuracy	99.92%	99.94%	99.76%	96.23%
Std	0.0011	0.0008	0.0003	0.0498

Table 1: Accuracy and Std over last 1000 iterations

than other 3 results, indicating the model is not fully converged. In comparison with LSTM, peephole LSTM need more iterations to reach perfect accuracy in each task accordingly. However, the shape of accuracy curve of peephole LSTM has shown that there is no more plateau period. I think this can be possibly explained the mechanism of "peephole", that the model could have a better understanding of overall input, though also lead to noise according to the plot. Lastly, peephole model's standard deviation is obviously larger than the LSTM model, which means the model is more affectable by different initialization.

2 Recurrent Nets as Generative Model

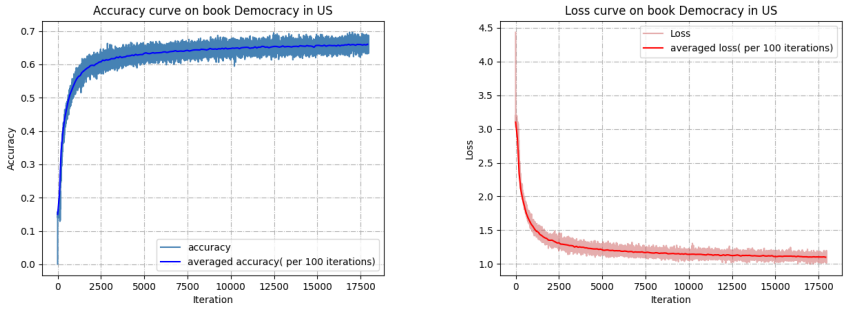
2.1

2.1.1 Training

As for hyper parameter selection, RMSProp has been chosen as the optimizer. A pre-experiment showed the model's training accuracy bounce in a wide range in certain training steps. This phenomenon is also reflected in the following figure 3a, where the accuracy curve appeared to be thick. RMSPRop adopts the idea that divides the gradient by a running average of its recent magnitude to overcome this issue[1], tend to tackle the above issue. As for the book, I select the *democracy in the US*. This book is 2 times longer than the Grim's fairy tales, and the content is more academic. Figure 3 showed the accuracy and loss curve over the training process, with averaged results over every 100 iterations. The model obtained the accuracy of 65% and loss of 1.1 by the end of 18000 iterations. For the task of predicting the next character, I believe the result is promising in consideration of the ambiguity of language.

2.1.2 Text Generation

Following Table 2 showed different sentences generated by the model w.r.t. different training steps. Looking vertically of different training step, most tokens are generated with meanings and basically followed the topic of the book. However we do not see much difference between 3 output result. From figure 3's training accuracy plot, it can be found the performance barely improve after 2500 iterations, which is earlier than the first one third iterations. The accuracy can be also reflected by the quality of generated sentence. Looking horizontally, with longer sequence length, the model could catch tokens that related to the book's



(a) Training Accuracy (b) Training Loss

Figure 3: Accuracy and Loss over Training

topic more clearly, i.e., better coherency. For instance, *"State of the Union"* in up right corner of the table, is a typical United States political terminology. While for shorter T=20 sequence, we can barely understood the content of sentence.

	T=20	T=30	T=60
1/3	*le the community in*	*s of the Union and the contrar*	*In the Union and the contrary of the State of the Union and*
2/3	*n the present day th*	*States is the same time the sa*	*1830, the Americans are considered in the United States the*
1	*s "Letters of the U*	*North America the proportion o*	*n the same political institutions of the United States is th*

Table 2: Generated sentences with different training steps by greedy sampling

2.1.3 Greedy Sampling vs Random Sampling

The temperature controls the model's output between fully deterministic and fully randomness. This is implemented by Multiplying with τ. From the perspective of distribution, multiply τ represents how sharply peaked the distribution is[2]. If *tau* is smaller than 1, then the original distribution has been flattened, i.e. more randomness; Otherwise the distribution curve is sharper, i.e. more certainty.

	temp=0.5	temp=1	temp=2
Example1	*hage Cures (Lety-"Wal in stocpationnes, or in dange-yiue?]]; See States--Informated will that: 2s.T*	*Ke. The public supportment has acknowledged upon his exertions in which fulls all this expedient; t*	*European benefits. It is the only of the laws of the United States than that of the past with the me*
Example2	*veys.-Creitting as well. -Law Ilbear.: Nos, whichies explorsisumattent, "inlished--That Ebntat--This*	*quent or rase from the eyes of the waters of these who should not be confined admiting their hutable*	*States and the last for the rest; and the laws of the Union and the constitutions of this agreementa*
Example3	*Zmself. On, submitted; sleet frueties; no legeliquees NRiflicg Rise Picsing. Two lraugtoss) opinion*	*Mowever it ought not, you attained in the characters of the United States; the liberties of the coun*	*% a democracy in the world. The only one which they have been represented and the community with the*

Table 3: Generated sentence with different temperatures with T=100

Table 3 showed the generated sentence w.r.t. different temperature values. Inspired by the previous result, we select a longer sequence of 100 to understand the content of generated sentences. For temperature=0.5, the character in the token are rather random, which can be explained by the flatten distribution, making the selection of character more random. When increased temperature to 1, the sentences are formed by tokens with meanings and partly readable. Besides, we could discover some words related to the book, though rather limited. When temperature increased 2, the sentence is more readable with more politic terminologies, though less diverse when comparing three generated sentences vertically. Finally, comparing temp=2's sentence with previous sentences generated by the greedy method, we can see there is less duplication and more vivid tokens.

2.2 Sentence Completion

In consider of the content of the book *democracy in the US*, we gave the model following token to follow: *"Secretary of States"*. We only update the hidden state and cell state before the end of given token. Afterwards, the character is updated according to either greedy or random sampling with temperature by the newly generated character. The results have been shown in Table 4. Besides temperature=0.5, all generated sentence are readable. though not following the

English grammar. All other three temperature mentioned token "United States" in their context. From the sentence generated by greedy sampling we can see some replication tokens like "states", which can be inferred this is a token with high occurrence rate. Surprisingly, the sentence completed by temperature of 2 has the best semantics, which mentioned tokens like United States, intended to the majority of people, etc.

	Sentence
Greedy	*Secretary of States are the only an individual is the same property in the United States is a state*
Temp=2	*Secretary of States in the United States are intended to the majority of the people in the same pros*
Temp=1	*Secretary of States is it divided into the United States is, that as a single but a commercial*
Temp=0.5	*Secretary of States to Poute) hzise usins xequiptors; s?: als immenseild to its rane si uneraliging*

Table 4: Complete unfinished sentence with different sampling methods with T=100

3 Graph Neural Networks

3.1

1. Structural information is exploited by the adjacent matrix: if two nodes i and j are connected, M_{ij} is equal to 1, otherwise 0. According to the formula of GCN, during the graph convolution, first the model construct a adjacent matrix with identity matrix representing itself. Next, each node in the graph will use this adjacent matrix to collect features of neighbours by either sum up or average. At the end all collected features are multiplied with activation matrix H to transfer features into message, which can be regarded as a process as "message passing' between nodes.

2. One of the major drawback of GCN is computation cost. When graph is big, transferring message in the whole graph is not applicable due to the storage complexity $O(n^2)$. A new node's embedding is applicable only after traversing every node in the graph. One possible solution focus on how to restrain the computational complexity. A widely used method is to only transferring and learning the information of node's neighbour instead

of over the entire graph, which is named as GraphSAGE[3], as an improved version of GCN.

3.2

1. Adjacent matrix $\hat{A}$, with A,B,C,D,E,F as sequence order:

$$\hat{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

2. At least need 4 updates to forward the information from C to E. First update to D, secondly update to either B or E, third step to A and finally transformed to E.

3.3

$$h_i^{l+1} = \sigma\left(\sum_{j \in \mathcal{N}(\rangle)} \alpha_{ij} W^{(l)} h_j^{(l)}\right) \tag{9}$$

In the equation 9 I add a parameter α_{ij} to the original equation of graph convolution layer. α_{ij}represents the attention weight between node i and j. This is calculated by softmax function over j of the $a(\mathbf{W}\vec{h_i}\mathbf{W}\vec{h_j})$, where a is common shared mechanism of attention, mapping the variable space from matrix into vector [4].

3.4

Commonly used graph structure includes knowledge graph and social networks. In the knowledge graph, GNN can be used to make link prediction between entities; For the social network GNN can help to produce friend recommendation based on the existing social network links. Besides, in the domain of organic chemistry, given already known module model, GNN can be used to better predict the potential module structure.

3.5

1. RNN usually are more suitable and outperform in sequential data such as text and videos. This is the result of the RNN structure: same unit is shared by all sequential time step. With the control methods like gates, sequential features can be better captured and understood by the model. On the other hand of GNN, with the increase of sequential data, the GNN

will grow larger in single dimension and hardly capture the feature due to the convolution mechanism.
On the contrary, GNN outperform RNN in graph based data such as knowledge graphs , social network etc. In this graph structure, each node has multiple degrees, GNN's message passing could quickly go through the network. RNN, however, could only store the dependency of data inside of each node rather explicitly. Moreover, RNN are directed graphs, which also restricted the application on the graph structure.

2. Inspired idea of convert sentence to graph by grammar tree. GNN and RNN can be combined together to improve the performance of text classification. This idea is implemented in a recent paper[5]. Firstly GCN is used to build a text corpus map, while LSTM as RNN can be used to implement text classification task and improve the computational efficiency with the GCN's corpus map.

References

[1] Hinton G, Srivastava N, Swersky K. Neural Networks for Machine Learning Lecture 6a Overview of Mini–Batch Gradient Descent.
https://www.cs.toronto.edu/ tijmen/csc321/slides/lecture$_s$*lides*$_l$*ec*6.*pdf*

[2] Goodfellow I, Yoshua Bengio, Courville A. Deep Learning. The Mit Press; 2017.

[3] Hamilton W, Ying R, Leskovec J. Inductive Representation Learning on Large Graphs. Accessed November 29, 2020. https://arxiv.org/pdf/1706.02216.pdf

[4] Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. GRAPH ATTENTION NETWORKS. Accessed November 29, 2020. https://arxiv.org/pdf/1710.10903.pdf

[5] Gao L, Wang J, Pi Z, et al. A Hybrid GCN and RNN Structure Based on Attention Mechanism for Text Classification. Journal of Physics: Conference Series. 2020;1575:012130. doi:10.1088/1742-6596/1575/1/012130