Ain Shams University

Faculty of Engineering

Computer and Systems Department

CSE 616: Neural Networks and their applications

Assignment 3

Report

Name	Abdelrahma Ibrahim Yassin Ahmad
Code	1902394

Question1:

Date : / /

* Question 1:-

1- $h_t = w_h f(h_{t-1}) + w_x x_t$, $y_t = w_y f(h_t)$

* all activation fn are linear

$\therefore h_1 = w_h h_0 + w_x x_1 = 1 \times 1 + 0.1 \times 10 = 2$

$h_2 = w_h h_1 + w_x x_2 = 1 \times 2 + 0.1 \times 10 = 4$

$y_2 = w_y \cdot h_2 = 2 \times 3 = 6$

2. $L_t = \sum_{j=}^{2} (\hat{y}_j - y_j)^2 = (4-5)^2 + (6-5)^2 = 2$

3. $\frac{\partial L_t}{\partial h_1} = \frac{\partial L_1}{\partial h_1} + \frac{\partial L_2}{\partial h_1}$

$\frac{\partial L}{\partial h_1} = \frac{\partial L_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial h_1} = 2(\hat{y}_1 - y_1) \cdot w_y = 2(4-5) \cdot 2 = -4$

$\frac{\partial L_2}{\partial h_1} = \frac{\partial L_2}{\partial y_2} \cdot \frac{\partial y_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} = 2(\hat{y}_2 - y_2) \cdot w_y \cdot w_h = 2 \times 2 \times 1 = 4$

$\therefore \frac{\partial L_t}{\partial h_1} = -4 + 4 = 0$

$\therefore \frac{\partial L_t}{\partial w_h} = \frac{\partial L_1}{\partial w_h} + \frac{\partial L_2}{\partial w_h}$

$\therefore \frac{\partial L_1}{\partial w_h} = \frac{\partial L_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_h} = 2(\hat{y}_1 - y_1) \cdot w_y \cdot h_0 = -4$

$\therefore \frac{\partial L_2}{\partial w_h} = \frac{\partial L_2}{\partial y_2} \cdot \frac{\partial y_2}{\partial w_h}$

$\frac{\partial y_2}{\partial w_h} = \frac{\partial y_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_h} + \frac{\partial y_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_h} = \frac{\partial y_2}{\partial h_2} \left(\frac{\partial h_2}{\partial w_h} + \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_h} \right)$

$= w_y [h_1 + w_h h_0 = 6 \quad \therefore \frac{\partial L_2}{\partial w_h} = 2(6-5) * 6 = 12$

$\therefore \frac{\partial L_t}{\partial w_h} = 12 - 4 = 8$

Question 2:

$$\frac{\partial h}{\partial h_{t-1}} = [1 - \tanh^2(W_{hh}h_{t-1}, W_{xh}x_t).W_{hh}]$$

If W_{hh} is smaller than one, the recursive use of $\frac{\partial h}{\partial h_{t-1}}$ will force the gradient to zero as

$$1 - \tanh^2(W_{hh}h_{t-1}, W_{xh}x_t).W_{hh} \leq 1$$

Question 3:

We shall consider using GRU when we want to deal with long term dependencies to be able to solve the issue of vanishing/exploding gradients in the simple vanilla RNN.

Question 4:

Advantage: Faster parameter updates

Disadvantage: Not able to capture longer dependencies than the truncated

length.

Question 5:

a) The RNN will perform much better because in the RNN we are having shared weights over time which means we will have less number of parameters than using a fully connected neural network and having less number of parameters will help to reduce overfitting and achieve generalization faster.
b) The nature of the input data will be based on having an input text (ciphered text) which will be converted to a list of characters based on the character level modelling, and we will handle the dynamic length sentences by using padding. the output of our RNN Model will be also an array of characters that represents the output sequence. We shall consider convert every character in the training data to a specific character.
c) The model will be many-to-many as we will have an input sequence of characters that represent the input sentence (encrypted test) and the output sequence will be the plaintext.
d) Each training sample will be a sequence of integers that represent each character in the input text and so the output text. Each text will be separated into a list of characters and convert each character to a specific integer. An example will be as the following:
    a. Input: This is a short sentence.
    b. Processed Input: [16, 6, 4, 5, 1, 4, 5, 1, 13, 1, 5, 6, 3, 7, 12, 1, 5, 2, 11, 12, 2, 11, 10, 2, 1, 15]
e) By using padding, we can deal with dynamic length signals. we can add padding to the end of the sequences to make them the same length.
f) The Required input preprocessing steps will be:

a. Isolating each character as an array element (instead of an entire phrase, or word being the element of the array)
b. Tokenizing the characters so we can turn them from letters to integers and vice-versa
c. Padding the strings so that all the inputs and outputs can fit in matrix form.

g) The Model architecture can be as follows:
a. x = Input(shape=input_shape[1:])
b. seq = SimpleRNN(units= 64, return_sequences = True, activation="tanh", name='Layer1')(x) # output must be batchsize x timesteps x units
c. output = TimeDistributed(Dense(units = plaintext_vocab_size, activation='softmax', name='Layer2'))(seq)
d. model = Model(inputs = x, outputs = output)

h) Output Preprocessing steps will be:
a. Convert the array of integer to tokens to array of characters.
b. Construct the output sequence and remove any unnecessary paddings.