Home Work3

Abdullah Aml

May 28, 2022

1 Question1

1.1 $\hat{y}_2$

$$h_1 = W_h h_0 + W_x x_1 = (1)(1) + (.1)(10) = 2$$
$$\hat{y}_1 = W_y h_1 = (2)(2) = 4$$
$$h_2 = W_h h_1 + W_x x_2 = (2)(1) + (.1)(10) = 3$$
$$\hat{y}_2 = W_y h_1 = (2)(3) = 6$$

1.2 L_t

$$L_t = \sum_i (\hat{y}_i - y_i)^2$$
$$L_t = (4-5)^2 + (6-5)^2 = 2$$

1.3 $\frac{\partial L_t}{\partial h_1}$

$$\frac{\partial L_t}{\partial h_1} = \frac{\partial L_1}{\partial h_1} + \frac{\partial L_2}{\partial h_1}$$
$$\frac{\partial L_t}{\partial h_1} = \frac{\partial L_1}{\partial y_1}\frac{\partial y_1}{\partial h_1} + \frac{\partial L_2}{\partial y_2}\frac{\partial y_2}{\partial h_2}\frac{\partial h_2}{\partial h_1}$$
$$\frac{\partial L_t}{\partial h_1} = 2(\hat{y}_1 - y_1)(W_y) + 2(\hat{y}_2 - y_2)(W_y)(W_h)$$
$$\frac{\partial L_t}{\partial h_1} = 2(4-5)(2) + 2(6-5)(2)(1) = -4 + 4 = 0$$

1.4 $\frac{\partial L_t}{\partial W_h}$

$$\frac{\partial L_t}{\partial W_h} = \frac{\partial L_1}{\partial W_h} + \frac{\partial L_2}{\partial W_h}$$
$$\frac{\partial L_t}{\partial W_h} = \frac{\partial L_1}{\partial \hat{y}_1}\frac{\partial \hat{y}_1}{\partial h_1}\frac{\partial h_1}{\partial W_h} + \frac{\partial L_2}{\partial \hat{y}_2}\frac{\partial \hat{y}_2}{\partial h_2}(\frac{\partial h_2}{\partial W_h} + \frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W_h})$$
$$\frac{\partial L_t}{\partial W_h} = 2(\hat{y}_1 - y_1)W_y h_0 + 2(\hat{y}_2 - y_2)W_y(h_1 + W_h h_0)$$
$$\frac{\partial L_t}{\partial W_h} = 2(4-5)(2)(1) + 2(6-5)(2)(2+1) = 8$$

2 Question2

While calculating $\frac{\partial E_t}{W_{hh}}$ to calculate gradient:

$$\frac{\partial E_T}{\partial W_{hh}} = \frac{\partial E_1}{\partial W_{hh}} + \frac{\partial E_2}{\partial W_{hh}} + \cdots + \frac{\partial E_M}{\partial W_{hh}}$$

$$\frac{\partial E_5}{\partial W_{hh}} = \frac{\partial E_5}{\partial y_5}\frac{\partial y_5}{\partial h_5}\left(\sum_i^5 \frac{\partial h_5}{\partial h_i}\frac{\partial h_i}{\partial W_{hh}}\right)$$

Calculating gradient of W_{hh}: $(\frac{\partial E_m}{W_{hh}})$ for any output (m):

$$\frac{\partial E_m}{\partial W_{hh}} = \frac{\partial E_m}{\partial y_m}\frac{\partial y_m}{\partial h_m}\left(\sum_i^m \frac{\partial h_m}{\partial h_i}\frac{\partial h_i}{\partial W_{hh}}\right)$$

We have N outputs and Errors, then the total gradient of W_{hh} : $\frac{\partial E_t}{\partial W_{hh}}$ is:

$$\frac{\partial E_t}{\partial W_{hh}} = \sum_{m=1}^{M} \frac{\partial E_m}{\partial y_m}\frac{\partial y_m}{\partial h_m}\left(\sum_i^m \frac{\partial h_m}{\partial h_i}\frac{\partial h_i}{\partial W_{hh}}\right)$$

The problem here with the term: $\frac{\partial h_m}{\partial h_i}$, for example $\frac{\partial h_9}{\partial h_1}$:

$$\frac{\partial h_9}{\partial h_1} = \frac{\partial h_9}{\partial h_8}\frac{\partial h_8}{\partial h_7}\frac{\partial h_7}{\partial h_6}\cdots\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}$$

$$\frac{\partial h_m}{\partial h_i} = \prod_{k=i}^{m-1} \frac{\partial h_{k+1}}{\partial h_k}$$

as the sequence gets larger the number of products get larger, and we know that:

$$\begin{cases} overshooting & \frac{\partial h_{k+1}}{\partial h_k} > 1 \\ vanishinggradient & \frac{\partial h_{k+1}}{\partial h_k} < 1 \end{cases}$$

3 Question3

We use LSTMs, or GRUs when we have long sequence aka: long term dependency;.

4 Question4

Truncated back propagation throw time means: In RNNs instead of calculating gradient form last cell to the first cell we will fix number of steps to go backwards.

- Advantage: avoiding vanishing. or exploding gradient.
- Disadvantages: losing long term dependency, as we will reduce error for fixed number of previous context.

5 Question5

5.1 a

We have a problem that we only can use 26 training data so we will have under-ftting problem, unlike RNNs we will input sentences , absolutely will have no under-fitting.

5.2 b

The input will be a sentence. Every character is one-hot encoded of $length = 28$ for example:

$$c = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

The output will also be one-hot encoded vector.

5.3 c

The model is **Many to Many** we have sequence of letters as input, and sequence of letters as output.

5.4 d

we will have an input sentence (encrypted) and and output sentence (decrypted) we will decode every character in the sentence by one-hot encoded vector of $length = 26$ for example: If the $key = 3$.

$$x[i] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, y[i] = a \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

5.5 e

We will add start of work token [start], and an end of sentence token [end].

5.6 f

1. Lower-case characters, and remove punctuation except ".".
2. Tokenize Input sentence character by character.
3. insert Start, and end of sentence tokens: [start], [end].
4. insert[sep] to separate every word in the sentence.
5. convert each token to its corresponding character by character on-hot encoding.

5.7 g

```
class RNNCell(nn.Module):

    def __init__(self, num_inputs, num_hidden, num_outputs):
        super().__init__()
        # Initialize the modules we need to build the network
        self.linear1 = nn.Linear(num_inputs, num_hidden)
        self.act_fn = nn.Tanh()
        self.linear2 = nn.Linear(num_hidden, num_outputs)
        self.act_softmax = nn.Softmax()

    def forward(self, x, h):
        # Perform the calculation of the model to determine the prediction
        h1 = self.linear1(torch.cat(x, h))
        h1 = self.act_fn(h1)
```

```
        y = self.linear2(h1)
        y = self.act_Softmax(y)
        return h1, y
```

5.8 h

The reverse of 5.6

1. chose max of every token and get the corresponding character.
2. replace [start], [end] tokens, and replace it by ".".
3. replace [separate] token by space.
4. concatenate the sentence.