Char Level LM

Piya Amara Palamure

03-25-2024

How the char level prediction are done.

- Iterate through each character in the word list (paragraph) and create a long list of character arrays. Note that the target array (Y) corresponds to the character one step ahead of the input array (X).
- Encode characters into integers and then perform one-hot encoding on the integers.

- Initialize the weight matrix with random numbers. For simplicity, we'll use a neural network (NN) with one hidden layer. These weights will be updated during the backpropagation process.

- Obtain the output of the NN (logits) and apply the exponential function to obtain counts. This step yields the frequency of each character that may follow the current character.
- Calculate the probability of each character (out of 27 possibilities) appearing as the next character after the current one by dividing each character count by the total count.
- Define the loss function as the negative log-likelihood function, which we aim to minimize through optimization.

- Word list

```
Words[:10] = ['emma’, 'olivia’, 'ava’, 'isabella’, 'sophia’, 'charlotte’, 'mia’,
'amelia’, 'harper’, 'evelyn']
```

- Input character list

```
Chars[:20] = ['.', 'e', 'm', 'm', 'a', '.','o','l', 'i', 'v', 'i', 'a', '.', 'a',
'v', 'a', '.', 'i', 's', 'a']
```

- Char to integer encoding

Encode = {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f’, 7: 'g', 8: 'h', 9: 'i', 10: 'j', 11: 'k', 12: 'l', 13: 'm', 14: 'n', 15: 'o', 16: 'p', 17: 'q', 18: 'r', 19: 's', 20: 't', 21: 'u', 22: 'v', 23: 'w', 24: 'x', 25: 'y', 26: 'z', 0: '.'}

- Input integer list

```
Input_int[:20] = [ 0, 5, 13, 13, 1, 0, 15, 12, 9, 22, 9, 1, 0, 1, 22, 1, 0, 9, 19, 1]

Output_int[:20]= [ 5, 13, 13, 1, 0, 15, 12, 9, 22, 9, 1, 0, 1, 22, 1, 0, 9, 19, 1, 2]
```

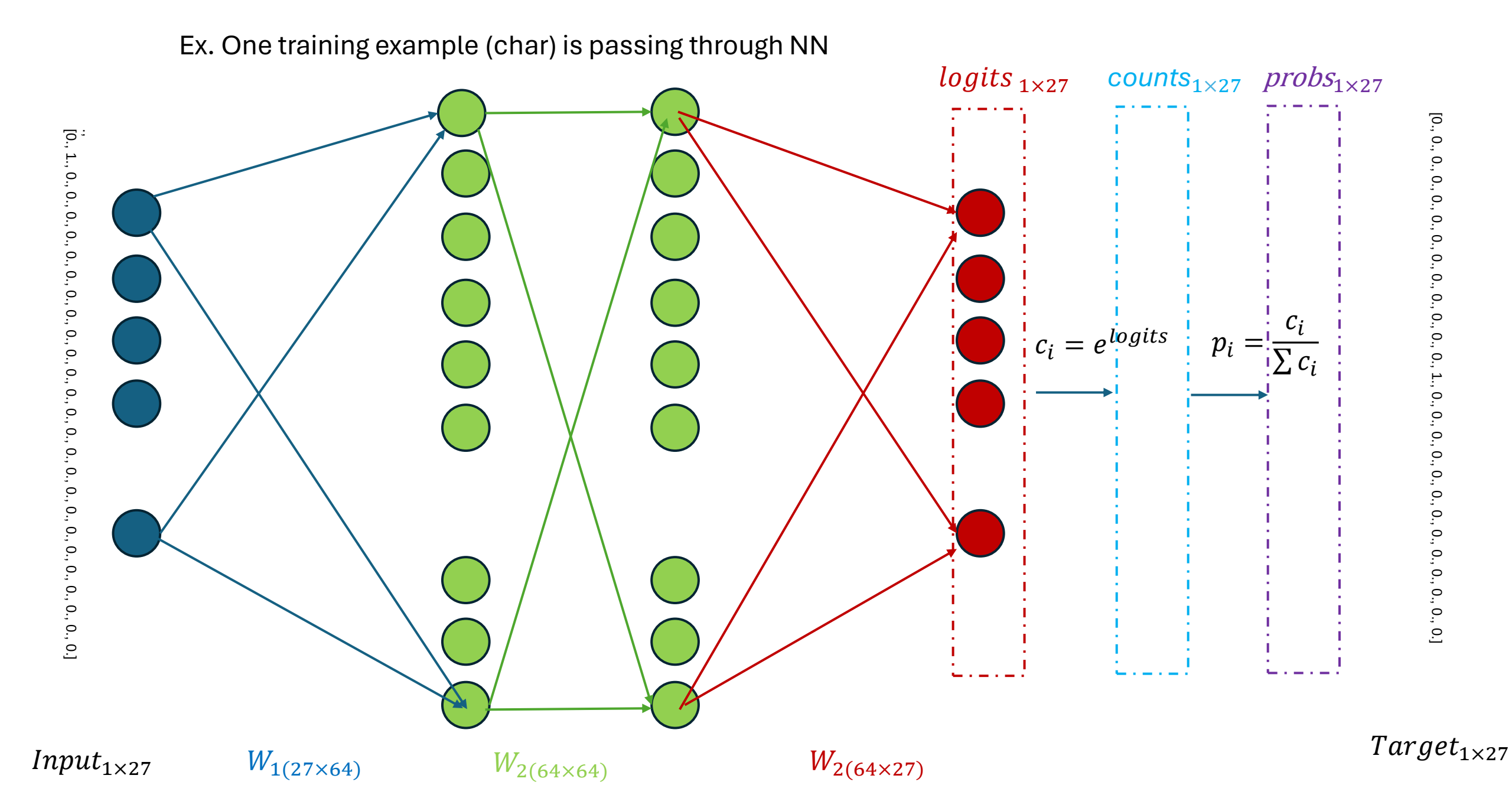
Input to the network (only showing first 5 chars as an example)

```
In_chars[:5]  = ['.', 'e', 'm', 'm', 'a’]

In_int[:5]    = [ 0, 5, 13, 13, 1]

X_enc [:5]    =
[[1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
[0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
[0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]]
```

Target values

```
Out_chars[:5] = ['e', 'm', 'm', ‘a’, '.’]

Input_int[:5] = [ 5, 13, 13, 1, 0]
```

Ex. One training example (char) is passing through NN



- Forward pass through the NN and Loss function

1. There are 32, 033 words in the text we are going to analyze.
2. There are 228,146 input chars to the network, so after one hot encoding the input shape is: $(228146, 27)$
3. Size of the target matrix is: (228146,1)
4. Size of weight matrix is: (27,27)
5. Size of the output (logits) matrix of the network: (228146,27)

$$logits = X_{enc} \times W_1$$ [(228146,27)*(27,27)->(228146,27)]

$$counts = e^{logits}$$ [(228146,27)]

$$p_i = \frac{c_i}{\sum c_i}$$ [(1,27)] The sum is along the axis 1

$$loss = \frac{1}{n}\sum_i -\log(p_{ij})$$ Where, j is the index of the target character

- The weights are updated using the gradient descent (to minimize the loss)

$$w_{lk}(t+1) = \mathrm{w_{lk}(t)} - \gamma_t \frac{\partial Loss}{\partial w_{lk}(t)}$$

- During the inference one hot encoded character is pass through the network and get the output probabilities (exactly same as during the training).
- After getting the probabilities, a candidate character with maximum probability is chosen from multinomial distribution.
- Note the weights are updated using all the training examples here and no activation function is used as well (very basic level NN)