Recurrent Neural Networks

Sequences Class 4 In-class Activities

Yuntian Deng, Matthew Finlayson

November 2019

1 The RNN forward step

Recall the structure of an RNN.

(Insert image or RNN here.)

A RNN works by calculating a sequence of outputs o and hidden states h from a sequence of inputs x. Each hidden state h_t and output o_t of an RNN is calculated from an input x_t and the previous hidden state h_{t-1} using a set of weights and biases V, W, b, and C according to the following equations.

$$h_t = \sigma(Wx_t + Vh_{t-1} + b)$$

$$o_t = Ch_t$$

To better understand this process, calculate the o_t and o_{t+1} given the following parameters

$$W = \begin{bmatrix} 0.3 & 0.6 \\ 0.2 & 0.1 \end{bmatrix}, V = \begin{bmatrix} 0.4 & 0.4 \\ 0.9 & 0.7 \end{bmatrix}, b = \begin{bmatrix} 0.1 \\ 0.6 \end{bmatrix}, C = \begin{bmatrix} 0.2 & 0.1 \\ 0.2 & 0.5 \end{bmatrix}$$

and the following inputs and initial hidden state.

$$x_t = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, x_{t+1} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, h_{t-1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

You may assume for this problem that $\sigma(x) = x$.

Solution.

$$\begin{aligned} h_t &= \sigma\left(\begin{bmatrix} 0.3 & 0.6 \\ 0.2 & 0.1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.4 & 0.4 \\ 0.9 & 0.7 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.6 \end{bmatrix}\right) \\ &= \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix} \end{aligned}$$

$$o_t = \begin{bmatrix}0.2 & 0.1\\0.2 & 0.5\end{bmatrix}\begin{bmatrix}0.7\\0.7\end{bmatrix}$$
$$= \begin{bmatrix}0.21\\0.49\end{bmatrix}$$

$$\begin{aligned}
o_{t+1} &= \begin{bmatrix}0.2 & 0.1\\0.2 & 0.5\end{bmatrix}\sigma\left(\begin{bmatrix}0.3 & 0.6\\0.2 & 0.1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}0.4 & 0.4\\0.9 & 0.7\end{bmatrix}\begin{bmatrix}0.7\\0.7\end{bmatrix} + \begin{bmatrix}0.1\\0.6\end{bmatrix}\right)\\
&= \begin{bmatrix}0.2 & 0.1\\0.2 & 0.5\end{bmatrix}\left(\begin{bmatrix}0.3\\0.2\end{bmatrix} + \begin{bmatrix}0.28+0.28\\0.62+0.49\end{bmatrix} + \begin{bmatrix}0.1\\0.6\end{bmatrix}\right)\\
&= \begin{bmatrix}0.2 & 0.1\\0.2 & 0.5\end{bmatrix}\begin{bmatrix}0.96\\1.91\end{bmatrix}\\
&= \begin{bmatrix}0.383\\1.147\end{bmatrix}
\end{aligned}$$

□

2 Bigrams and RNNs

In this activity you will recall how a bigram model works and apply that knowledge to design an RNN to behave like a bigram model.

2.1 The bigram language model construction

Recall that a bigram language model uses the previous word in a sequence to predict the next word. A bigram model consists of a vocabulary V and a transition table T such that T_{ij} is the probability that word V_j will follow word V_i.

Construct a vocabulary V and transition table T from the following data.

Is this the real life?
Is this just fantasy?

Use Laplace smoothing to improve your transition table (otherwise the next exercise will be silly.)

Solution.

$$V = [is, this, the, real, life, just, fantasy]$$

$$T = \begin{bmatrix}
0.111 & 0.111 & 0.125 & 0.125 & 0.143 & 0.125 & 0.143\\
0.333 & 0.111 & 0.125 & 0.125 & 0.143 & 0.125 & 0.143\\
0.111 & 0.222 & 0.125 & 0.125 & 0.143 & 0.125 & 0.143\\
0.111 & 0.111 & 0.250 & 0.125 & 0.143 & 0.125 & 0.143\\
0.111 & 0.111 & 0.125 & 0.250 & 0.143 & 0.125 & 0.143\\
0.111 & 0.222 & 0.125 & 0.125 & 0.143 & 0.125 & 0.143\\
0.111 & 0.111 & 0.125 & 0.125 & 0.143 & 0.250 & 0.143
\end{bmatrix}$$

□

2.2 Bigram language model application

Use your bigram model to calculate the likelihood of the following sentence.

Is this just real life fantasy?

Solution.

$$0.333 \cdot 0.250 \cdot 0.125 \cdot 0.250 \cdot 0.143 = 0.00037$$

□

2.3 Tiny bigram RNN

Consider a bigram model with a vocabulary B of size 2 and the following transition table.

$$T = \begin{bmatrix} 0.4 & 0.2 \\ 0.6 & 0.8 \end{bmatrix}$$

Also consider the following sequence

$$x = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

where each vector in the sequence is a one-hot vector representing a word in the vocabulary, therefore the first vector in the sequence represents B_1 and the second represents B_0.

Design the parameters W, V, b, and C of an RNN such that it behaves like a bigram model by outputting a vector with the probabilities of next word.[1]

Solution.

$$W = \sigma^{-1}(T),\ V = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},\ b = \begin{bmatrix} 0 \\ 0 \end{bmatrix},\ C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

□

[1]Hint: For instance, the probabilities for the word following the first vector in the sequence would be represented as

$$\begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}$$

because looking up the column representing B_1 in the transition table T we see that there is a 0.2 probability that the next word is B_0 and a 0.8 probability that the next word is B_1. Therefore your RNN's first output should be

$$o_0 = C\sigma\left(W \begin{bmatrix} 0 \\ 1 \end{bmatrix} + V \begin{bmatrix} 0 \\ 0 \end{bmatrix} + b\right) = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}$$

Note that we are assuming that h_{-1}, the initial h value is $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$. Also, we have not specified what σ is, but feel free to use its inverse function σ^{-1}.

2.4 General bigram RNN

Now that you have completed 2.3, you will now generalize the result to design an RNN to mimic a bigram model for an arbitrarily sized vocabulary.

Let x be a sequence of one-hot vectors representing a sentence such that x_i is a one-hot vector representing the ith word in the sentence.

Let T be the transition table for a bigram model with a vocabulary of size n.

Design the parameters of an RNN such that it behaves like a bigram model. That is, design the parameters such that on the input sequence x, your RNN outputs a sequence o such that if o_i is the ith vector in the sequence, and o_{ij} is the probability that the jth word in the vocabulary follows the word represented by x_i.[2]

Solution.

$$W = \sigma^{-1}(T),\ V = 0_{n\times n},\ b = 0_{1\times n},\ C = I_{n\times n}$$

□

3 RNN backpropogation

In these exercises you will build both your intuition and gain a concrete understanding of how back-propagation works in an RNN. Back-propagation is the method used to iteratively improve the parameters of an RNN in supervised learning. First, given a set of parameters, the RNN runs and we calculate the loss as a function of the output (a measure of how far off the RNN's output was from the desired output.) We then find the derivatives of the loss with respect to each parameter and use the derivatives to make small adjustments to the parameters to reduce the loss. This process is repeated until the loss is minimized.

3.1 Tinsy-insy-tiny RNN backprop

Consider an RNN run on an input sequence of length 1. This RNN's output sequence will therefore consist only of a single output o_0.

We are not giving you the loss function L, however we know that L is a function of the output o_0, therefore we give you $\frac{\partial L}{\partial o_0}$. Furthermore, o_0 is a function of W, therefore we also give you $\frac{\partial o_0}{\partial W}$.

[2] Hints: Remember the following equations.

$$h_t = \sigma(Wx_t + Vh_{t-1} + b)$$

$$o_t = Ch_t$$

You may find it useful to use matrices such as $I_{n\times n}$, the identity matrix of size $n \times n$, and $0_{n\times n}$, the 0 matrix of the same size. Again, feel free to use σ^{-1}.

Now, find the partial derivative of the loss L with respect to the parameter W in terms of $\frac{\partial L}{\partial o_0}$ and $\frac{\partial o_0}{\partial W}$.[3]

Solution.

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial o_0} \cdot \frac{\partial o_0}{\partial W}$$

□

3.2 Tinsy-tiny RNN backprop

Consider an RNN run on an input sequence of length 2. This RNN's output sequence o will consist of o_0 and o_1.

Things are starting to heat up now. Not only do we have to consider o_0 and o_1 which are functions of W, but also o_1 as a function of h_0 which is a function of W. This time we therefore also give you $\frac{\partial o_1}{\partial h_0}$ and $\frac{\partial h_0}{\partial W}$ as well as $\frac{\partial L}{\partial o_n}$ and $\frac{\partial o_n}{\partial W}$ for $n \in \{0, 1\}$.

Find the partial derivative of the loss L with respect to the parameter W in terms of $\frac{\partial o_1}{\partial h_0}$, $\frac{\partial L}{\partial o_n}$ and $\frac{\partial o_n}{\partial W}$ for $n \in \{0, 1\}$.[4]

Solution.

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial o_0} \cdot \frac{\partial o_0}{\partial W} + \frac{\partial L}{\partial o_1} \cdot \frac{\partial o_1}{\partial W} + \frac{\partial L}{\partial o_1} \cdot \frac{\partial o_1}{\partial h_0} \cdot \frac{\partial h_0}{\partial W}$$

□

3.3 Tiny RNN backprop

Almost there! Now consider an RNN run on an input sequence of length 3. This RNN's output sequence o will consist of o_0, o_1, and o_2.

This time, find the partial derivative of the loss L with respect to the parameter W in terms of $\frac{\partial o_n}{\partial h_{n-1}}$, $\frac{\partial h_n}{\partial W}$, $\frac{\partial L}{\partial o_n}$ and $\frac{\partial o_n}{\partial W}$ for $n \in [3]$.

Solution.

$$\frac{\partial L}{\partial W} = \sum_{n=0}^{2} \frac{\partial L}{\partial o_n} \frac{\partial o_n}{\partial W} + \sum_{n=1}^{2} \frac{\partial L}{\partial o_n} \frac{\partial o_n}{\partial h_{n-1}} \frac{\partial h_{n-1}}{\partial W} + \frac{\partial L}{\partial o_2} \frac{\partial o_2}{\partial h_1} \frac{\partial h_1}{\partial h_0} \frac{\partial h_0}{\partial W}$$

□

[3] Hint: Use the chain rule.

$$\frac{\partial a}{\partial b} \cdot \frac{\partial b}{\partial c} = \frac{\partial a}{\partial c}$$

[4] Hint: The solution will be the sum of 3 terms.

3.4 General RNN backprop

Whew! Last one like this, we promise. Consider an RNN run on an input sequence of length N. Find a general formula for the partial derivative of the loss L with respect to the parameter W in terms of $\frac{\partial o_n}{\partial h_{n-1}}$, $\frac{\partial L}{\partial o_n}$ and $\frac{\partial o_n}{\partial W}$ for $n \in [N]$.

Solution.

$$\frac{\partial L}{\partial W} = \sum_{i=0}^{N-1} \left(\frac{\partial L}{\partial o_i} \frac{\partial o_i}{\partial W} \right) + \sum_{i=0}^{N-1} \left(\sum_{j=i+1}^{N-1} \left(\frac{\partial L}{\partial o_j} \frac{\partial o_j}{\partial h_{j-1}} \frac{\partial h_{j-i-1}}{\partial W} \prod_{k=j-i}^{j-1} \frac{\partial h_k}{\partial h_{k-1}} \right) \right)$$

□

3.5 Looking back

In the last few problems your answers were in terms of partial derivatives that we gave you. In this exercise you will calculate part of a partial derivative.

Referring back to activity 1, find the partial derivative $\frac{\partial o_0}{\partial W_{0,0}}$.[5]

Solution.

$$\begin{aligned} o_0 &= C\sigma\left(Wx_0 + Vh_{-1} + b\right) \\ &= C\sigma\left(\begin{bmatrix} W_{0,1} \\ W_{1,1} \end{bmatrix} + b\right) \end{aligned}$$

o_0 does not depend on $W_{0,0}$, therefore $\frac{\partial o_0}{\partial W_{0,0}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. □

[5]Hint: we didn't tell what the function σ is. Turns out you will not need it.