# Recurrent Neural Networks

Quiz, 10 questions

**10/10 points (100%)**

**Congratulations! You passed!**

Next Item

**1.** 1 / 1 points

Suppose your training examples are sentences (sequences of words). Which of the following refers to the $j^{th}$ word in the $i^{th}$ training example?

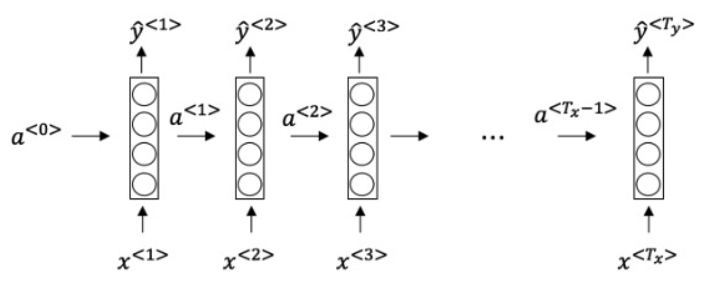
- (selected) $x^{(i)<j>}$

  **Correct**
  We index into the $i^{th}$ row first to get the $i^{th}$ training example (represented by parentheses), then the $j^{th}$ column to get the $j^{th}$ word (represented by the brackets).

- $x^{<i>(j)}$
- $x^{(j)<i>}$
- $x^{<j>(i)}$

**2.** 1 / 1 points

Consider this RNN:




This specific type of architecture is appropriate when:

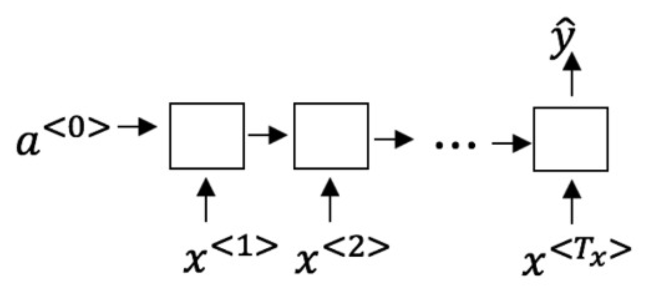
- (selected) $T_x = T_y$

  **Correct**
  It is appropriate when every input should be matched to an output.

- $T_x < T_y$
- $T_x > T_y$
- $T_x = 1$

**3.** 1 / 1 points

To which of these tasks would you apply a many-to-one RNN architecture? (Check all that apply).




- [ ] Speech recognition (input an audio clip and output a transcript)

  **Un-selected is correct**

- [x] Sentiment classification (input a piece of text and output a 0/1 to denote positive or negative sentiment)

  **Correct**
  Correct!

- [ ] Image classification (input an image and output a label)

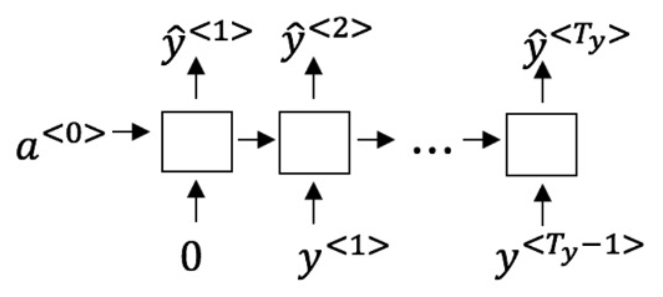
  **Un-selected is correct**

- [x] Gender recognition from speech (input an audio clip and output a label indicating the speaker's gender)

  **Correct**
  Correct!

**4.** 1 / 1 points

You are training this RNN language model.




At the $t^{th}$ time step, what is the RNN doing? Choose the best answer.

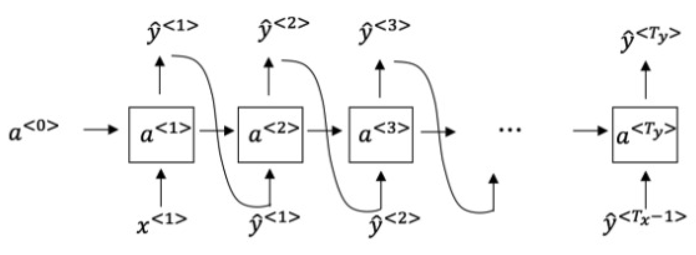
- Estimating $P(y^{<1>}, y^{<2>}, \ldots, y^{<t-1>})$
- Estimating $P(y^{<t>})$
- (selected) Estimating $P(y^{<t>} \mid y^{<1>}, y^{<2>}, \ldots, y^{<t-1>})$

  **Correct**
  Yes, in a language model we try to predict the next step based on the knowledge of all prior steps.

- Estimating $P(y^{<t>} \mid y^{<1>}, y^{<2>}, \ldots, y^{<t>})$

**5.** 1 / 1 points

You have finished training a language model RNN and are using it to sample random sentences, as follows:




What are you doing at each time step $t$?

- (i) Use the probabilities output by the RNN to pick the highest probability word for that time-step as $\hat{y}^{<t>}$. (ii) Then pass the ground-truth word from the training set to the next time-step.
- (i) Use the probabilities output by the RNN to randomly sample a chosen word for that time-step as $\hat{y}^{<t>}$. (ii) Then pass the ground-truth word from the training set to the next time-step.
- (i) Use the probabilities output by the RNN to pick the highest probability word for that time-step as $\hat{y}^{<t>}$. (ii) Then pass this selected word to the next time-step.
- (selected) (i) Use the probabilities output by the RNN to randomly sample a chosen word for that time-step as $\hat{y}^{<t>}$. (ii) Then pass this selected word to the next time-step.

  **Correct**
  Yes!

**6.** 1 / 1 points

You are training an RNN, and find that your weights and activations are all taking on the value of NaN ("Not a Number"). Which of these is the most likely cause of this problem?

- Vanishing gradient problem.
- (selected) Exploding gradient problem.

  **Correct**

- ReLU activation function g(.) used to compute g(z), where z is too large.
- Sigmoid activation function g(.) used to compute g(z), where z is too large.

**7.** 1 / 1 points

Suppose you are training a LSTM. You have a 10000 word vocabulary, and are using an LSTM with 100-dimensional activations $a^{<t>}$. What is the dimension of $\Gamma_u$ at each time step?

- 1
- (selected) 100

  **Correct**
  Correct, $\Gamma_u$ is a vector of dimension equal to the number of hidden units in the LSTM.

- 300
- 10000

**8.** 1 / 1 points

Here're the update equations for the GRU.

**GRU**

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\,c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[\,c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

Alice proposes to simplify the GRU by always removing the $\Gamma_u$. I.e., setting $\Gamma_u$ = 1. Betty proposes to simplify the GRU by removing the $\Gamma_r$. I. e., setting $\Gamma_r$ = 1 always. Which of these models is more likely to work without vanishing gradient problems even when trained on very long input sequences?

- Alice's model (removing $\Gamma_u$), because if $\Gamma_r \approx 0$ for a timestep, the gradient can propagate back through that timestep without much decay.
- Alice's model (removing $\Gamma_u$), because if $\Gamma_r \approx 1$ for a timestep, the gradient can propagate back through that timestep without much decay.
- (selected) Betty's model (removing $\Gamma_r$), because if $\Gamma_u \approx 0$ for a timestep, the gradient can propagate back through that timestep without much decay.

  **Correct**
  Yes. For the signal to backpropagate without vanishing, we need $c^{<t>}$ to be highly dependant on $c^{<t-1>}$.

- Betty's model (removing $\Gamma_r$), because if $\Gamma_u \approx 1$ for a timestep, the gradient can propagate back through that timestep without much decay.

**9.** 1 / 1 points

Here are the equations for the GRU and the LSTM:

**GRU**

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\,c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[\,c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

**LSTM**

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\,a^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma(W_f[\,a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[\,a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * c^{<t>}$$

From these, we can see that the Update Gate and Forget Gate in the LSTM play a role similar to _______ and ______ in the GRU. What should go in the the blanks?

- (selected) $\Gamma_u$ and $1 - \Gamma_u$

  **Correct**
  Yes, correct!

- $\Gamma_u$ and $\Gamma_r$
- $1 - \Gamma_u$ and $\Gamma_u$
- $\Gamma_r$ and $\Gamma_u$

**10.** 1 / 1 points

You have a pet dog whose mood is heavily dependent on the current and past few days' weather. You've collected data for the past 365 days on the weather, which you represent as a sequence as $x^{<1>}, \ldots, x^{<365>}$. You've also collected data on your dog's mood, which you represent as $y^{<1>}, \ldots, y^{<365>}$. You'd like to build a model to map from $x \to y$. Should you use a Unidirectional RNN or Bidirectional RNN for this problem?

- Bidirectional RNN, because this allows the prediction of mood on day t to take into account more information.
- Bidirectional RNN, because this allows backpropagation to compute more accurate gradients.
- (selected) Unidirectional RNN, because the value of $y^{<t>}$ depends only on $x^{<1>}, \ldots, x^{<t>}$, but not on $x^{<t+1>}, \ldots, x^{<365>}$

  **Correct**
  Yes!

- Unidirectional RNN, because the value of $y^{<t>}$ depends only on $x^{<t>}$, and not other days' weather.