Sequence Models-I

(Recurrent Neural Networks-Introduction, Types of RNNs Many-to-Many-RNNs for Sequence Labeling)

DR. JASMEET SINGH,
ASSISTANT PROFESSOR,
CSED, TIET

Sequence Models

- Sequence models are **models that input or output sequences of data**.
- Sequential data includes text streams, audio clips, video clips, time-series data and etc.
- Sequence models have transformed areas like Speech Recognition, Natural Language Processing, and other areas involving sequential data.

Examples of Sequential Data

Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation	Ø	→	
Sentiment classification	"There is nothing to like in this movie."	→	★☆☆☆☆
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger

Sequence Models: Notation

- Consider a Named Entity Recognition system, where, we are given with a sequence of words in an input and we have to identify whether each word of the sequence is a named entity (name of person, place, organization, number, etc.) or not.
- For example, consider an input example,

X= Harry Porter and Hermione Granger invented a new spell

So, the corresponding output is:

Y=[1 1 0 1 1 0 0 0 0]

- So, each element of the input sequence is denoted by $X^{<t>}$ and the output corresponding to it is $Y^{<t>}$ where t is the position number (sequence number) in the sequence.

Sequence Models: Notation (Contd.....)

- In general,

$X^{(i)<t>}$ denotes t^{th} input sequence for i^{th} example.

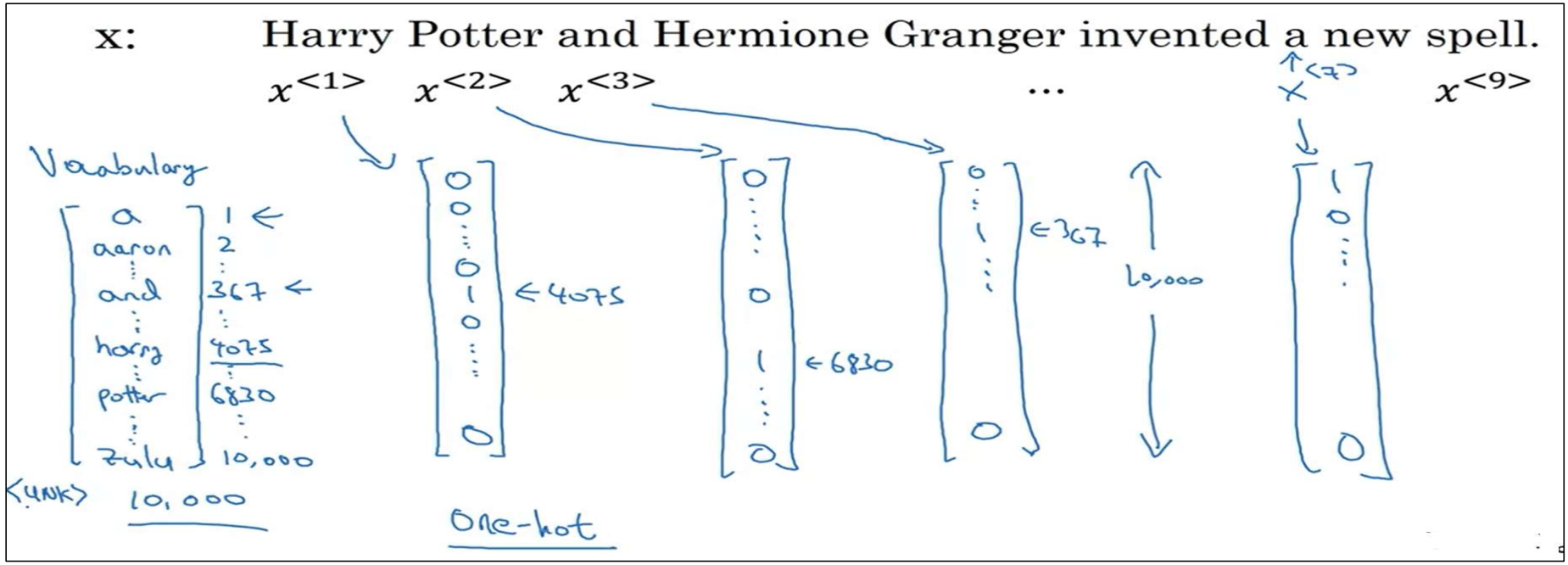
$Y^{(i)<t>}$ denotes t^{th} output sequence for i^{th} example.

$T_x^{(i)}$: denote length of i^{th} input (number of sequences in the i^{th} input)

$T_y^{(i)}$: denote length of i^{th} output (number of sequences in the i^{th} output)

Representing Words of Sequence

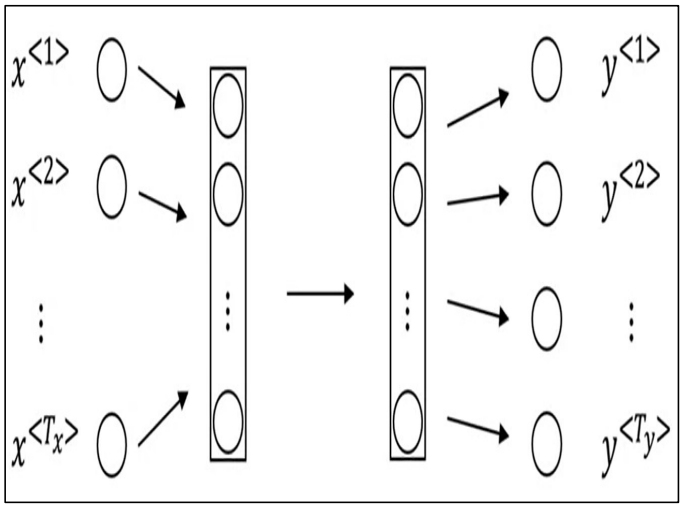
- The words in the sequence are represented by one-hot encoding scheme.



Why not a standard neural network for sequential data?

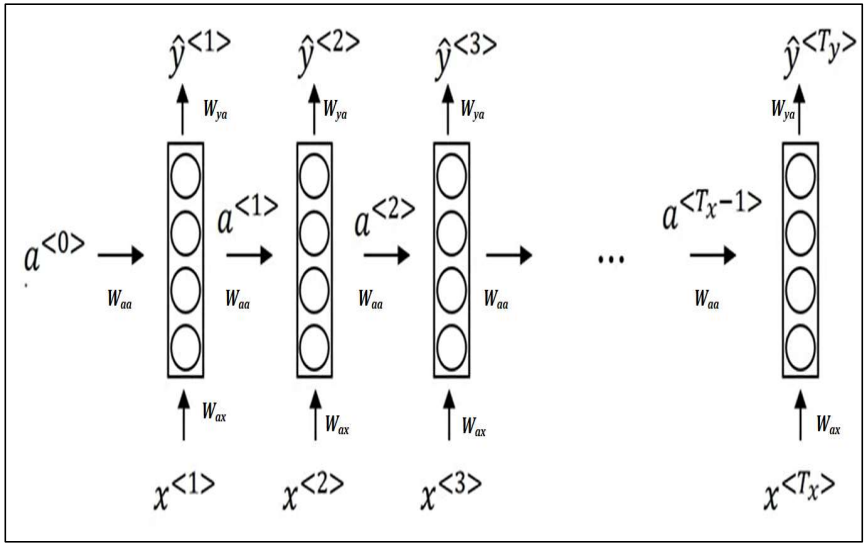
- A standard neural network (as shown in the figure) cannot handle sequential data due to the following reasons:

1. The inputs and output lengths differ in different outputs.
   - Though few zeros can be padded to each input to make them equal to the maximum length; but this is not a good representation for standard neural networks.
2. Does not share features learnt across different positions of text.
   - A word 'John' learnt as named entity at one position does not generalize at all positions of text.
   - Same as for Convolutional Neural Networks (CNNs) where we want that things learnt for one part of the image generalize well for remaining parts.
3. A standard neural network will have many number of parameters to be learnt as the input layer will have (max number of words * vocabulary size) inputs.



Recurrent Neural Network (RNN)

- In a Recurrent Neural Network (RNN),
  - the input is scanned from left to right.
  - each input of a sequence data is fed into a neural network layer; which predicts the output corresponding to the input sequence.
  - But the output is not computed from the current input sequence, but it also takes into account the activation from the previous layer (as shown in the figure).
  - the parameters are shared among all the time stamps
    - Wax: parameters governing the connections from input to each hidden layer
    - Waa: parameters that governs activations of each layer.
    - Way: parameters that governs the output prediction.



Forward Propagation

- In the forward propagation phase, at each time-stamp, activation ($a^{<t>}$) and predicted output ($\hat{y}^{<t>}$) are computed as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$
$$\hat{y}^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

- where g_1 is the activation function used for computation of activations and is generally taken as hyperbolic tangent (tanh); and g2 activation function used for computation of predicted values and is taken as sigmoid for binary classification / softmax for multi-label classification.
- b_a and b_y are bias for activations and predicted outputs.
- Wax are weights that take *'x'* type input and is computed for '*a'* type output (shape: number of neurons X size of one-hot matrix)
- Waa are weights that take *'a'* type input and is computed for '*a'* type output (shape: number of neurons X number of neurons)
- Wya are weights that take *'a'* type input and is computed for '*y'* type output (shape: 1X number of neurons)

Forward Propagation: Simplified Notation

- The equations computed in forward propagation

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$
$$\hat{y}^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

are expressed in shortened and simplified form as:

$$a^{<t>} = g_1(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$
$$\hat{y}^{<t>} = g_2\left(W_y a^{<t>} + b_y\right)$$

- Where W_a is the column wise concatenation of W_{aa} and W_{ax} i.e.

$$[W_{aa} \mid W_{ax}]_{number\ of\ neurons\ X\ (number\ of\ neurons + size\ of\ one\ hot\ vector)}$$

- where $[a^{<t-1>}, x^{<t>}]$ is the row wise concatenation of $a^{<t-1>}\ and\ x^{<t>}$

$$\begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}_{(number\ of\ neurons + size\ of\ one\ hot\ vector)\ \times 1}$$

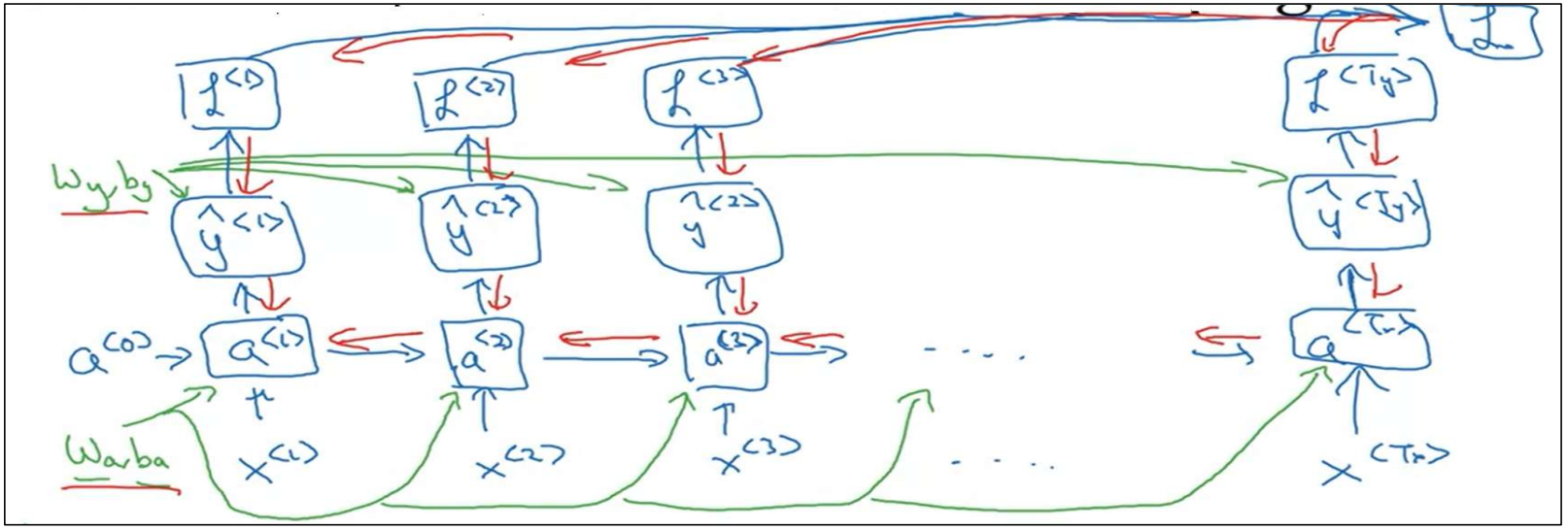
- Hence, $W_a[a^{<t-1>}, x^t] = W_{aa}a^{<t-1>} + W_{ax}x^{<t>}$

Back Propagation through time – RNN

- Recurrent Neural Networks are those networks that deal with sequential data.
- They predict outputs using not only the current inputs but also by taking into consideration those that occurred before it.
- For training such networks, we use good old backpropagation but with a slight twist.
- We don't independently train the system at a specific time *"t"*. We train it at a specific time *"t"* as well as all that has happened before time *"t"* like t-1, t-2, t-3. This kind of back propagation is called ***back propagation through time***.

Back Propagation through time (Contd.…)

- The computational graph for recurrent neural network (RNN) is shown below.
- The blue lines show computations during forward propagation phase and red lines show computations that are to be back propagated.



Back Propagation through time (Contd....)

- The loss function at any time stamp <t> in the sequence for a binary classification task is a cross-entropy function given by:

$$L^{<t>}(y^{<t>}, \check{y}^{<t>}) = -(y^{<t>} \log(\check{y}^{<t>}) + (1 - y^{<t>}) \log(1 - \check{y}^{<t>}))$$

- The total loss function for all time stamps is given by:

$$L(y, \check{y}) = \sum_{t=1}^{T_y} L^{<t>}(y^{<t>}, \check{y}^{<t>})$$

- The RNN model has to minimize the cost function in order to optimize the parameters $W_{ya}, W_{aa}, and\ W_{ax}$.

- So, we need to compute $\frac{\partial L}{\partial W_{ya}}, \frac{\partial L}{\partial W_{aa}}, and\ \frac{\partial L}{\partial W_{ax}}$

Back Propagation through time (Contd….)

- **For W_{ya} or W_y**

$$\frac{\partial L}{\partial W_y} = \frac{\partial (L^{<1>} + L^{<2>} + L^{<3>} \dots \dots \dots + L^{<T_y>})}{\partial W_y}$$

- **Considering back propagation at t=3**

$$L^{<3>} \text{ is a function of } \check{y}^{<3>} \text{ and } \check{y}^{<3>} \text{ is a function of } W_y$$

Therefore $\frac{\partial L^{<3>}}{\partial W_y} = \frac{\partial L^{<3>}}{\partial \check{y}^{<3>}} \frac{\partial \check{y}^{<3>}}{\partial W_y}$

In general, $\frac{\partial L^{<t>}}{\partial W_y} = \frac{\partial L^{<t>}}{\partial \check{y}^{<t>}} \frac{\partial \check{y}^{<t>}}{\partial W_y}$

Hence, W_y is updated as $\boldsymbol{W_y = W_y - \alpha \frac{\partial L}{\partial W_y}}$

Back Propagation through time (Contd….)

- **For W_{aa}**

$$\frac{\partial L}{\partial W_{aa}} = \frac{\partial (L^{<1>} + L^{<2>} + L^{<3>} \dots\dots\dots + L^{<T_y>})}{\partial W_{aa}}$$

Considering back propagation at t=3

$$\frac{\partial L^{<3>}}{\partial W_{aa}} = \frac{\partial L^{<3>}}{\partial \check{y}^{<3>}} \cdot \frac{\partial \check{y}^{<3>}}{\partial a^{<3>}} \cdot \frac{\partial a^{<3>}}{\partial W_{aa}} +$$

$$\frac{\partial L^{<3>}}{\partial \check{y}^{<3>}} \cdot \frac{\partial \check{y}^{<3>}}{\partial a^{<3>}} \cdot \frac{\partial a^{<3>}}{\partial a^{<2>}} \cdot \frac{\partial a^{<2>}}{\partial W_{aa}} +$$

$$\frac{\partial L^{<3>}}{\partial \check{y}^{<3>}} \cdot \frac{\partial \check{y}^{<3>}}{\partial a^{<3>}} \cdot \frac{\partial a^{<3>}}{\partial a^{<2>}} \cdot \frac{\partial a^{<2>}}{\partial a^{<1>}} \cdot \frac{\partial a^{<1>}}{\partial W_{aa}}$$

$L^{<3>}$ is a function of $\check{y}^{<3>}$. Hence, we differentiate $L^{<3>}$ w.r.t $\check{y}^{<3>}$.
$\check{y}^{<3>}$ is a function of $a^{<3>}$. Hence, we differentiate $\check{y}^{<3>}$ w.r.t $a^{<3>}$.
$a^{<3>}$ is a function of W_{aa}. Hence, we differentiate $a^{<3>}$ w.r.t W_{aa}.
But we can't stop with this; we also have to take into consideration, the previous time steps. So, we differentiate (partially) the Error function with respect to memory units $a^{<2>}$ as well as $a^{<2>}$ taking into consideration the weight matrix W_{aa}.
Hence, we differentiate $a^{<3>}$ with $a^{<2>}$ and $a^{<2>}$ with $a^{<1>}$

Back Propagation through time (Contd....)

- In general,

$$\frac{\partial L^{<t>}}{\partial W_{aa}} = \sum_{i=1}^{t} \frac{\partial L^{<t>}}{\partial \check{y}^{<t>}} \cdot \frac{\partial \check{y}^{<t>}}{\partial a^{<i>}} \cdot \frac{\partial a^{<i>}}{\partial W_{aa}}$$

$$\boldsymbol{W_{aa}\ is\ then\ updated\ as:\ W_{aa} = W_{aa} - \alpha \frac{\partial L}{\partial W_{aa}}}$$

Back Propagation through time (Contd….)

- **For W_{ax}**

$$\frac{\partial L}{\partial W_{ax}} = \frac{\partial (L^{<1>} + L^{<2>} + L^{<3>} \dots\dots\dots + L^{<T_y>})}{\partial W_{ax}}$$

Considering back propagation at t=3

$$\frac{\partial L^{<3>}}{\partial W_{ax}} = \frac{\partial L^{<3>}}{\partial \check{y}^{<3>}} \cdot \frac{\partial \check{y}^{<3>}}{\partial a^{<3>}} \cdot \frac{\partial a^{<3>}}{\partial W_{ax}} +$$

$$\frac{\partial L^{<3>}}{\partial \check{y}^{<3>}} \cdot \frac{\partial \check{y}^{<3>}}{\partial a^{<3>}} \cdot \frac{\partial a^{<3>}}{\partial a^{<2>}} \cdot \frac{\partial a^{<2>}}{\partial W_{ax}} +$$

$$\frac{\partial L^{<3>}}{\partial \check{y}^{<3>}} \cdot \frac{\partial \check{y}^{<3>}}{\partial a^{<3>}} \cdot \frac{\partial a^{<3>}}{\partial a^{<2>}} \cdot \frac{\partial a^{<2>}}{\partial a^{<1>}} \cdot \frac{\partial a^{<1>}}{\partial W_{ax}}$$

$L^{<3>}$ is a function of $\check{y}^{<3>}$. Hence, we differentiate $L^{<3>}$ w.r.t $\check{y}^{<3>}$.
$\check{y}^{<3>}$ is a function of $a^{<3>}$. Hence, we differentiate $\check{y}^{<3>}$ w.r.t $a^{<3>}$.
$a^{<3>}$ is a function of W_{aa}. Hence, we differentiate $a^{<3>}$ w.r.t W_{aa}.
But we can't stop with this; we also have to take into consideration, the previous time steps. So, we differentiate (partially) the Error function with respect to memory units $a^{<2>}$ as well as $a^{<2>}$ taking into consideration the weight matrix W_{ax}.
Hence, we differentiate $a^{<3>}$ with $a^{<2>}$ and $a^{<2>}$ with $a^{<1>}$

Back Propagation through time (Contd....)

- In general,

$$\frac{\partial L^{<t>}}{\partial W_{aa}} = \sum_{i=1}^{t} \frac{\partial L^{<t>}}{\partial \check{y}^{<t>}} \cdot \frac{\partial \check{y}^{<t>}}{\partial a^{<i>}} \cdot \frac{\partial a^{<i>}}{\partial W_{ax}}$$

$$\boldsymbol{W_{ax}\ is\ then\ updated\ as:\ W_{ax} = W_{ax} - \alpha \frac{\partial L}{\partial W_{ax}}}$$

Types of RNNs

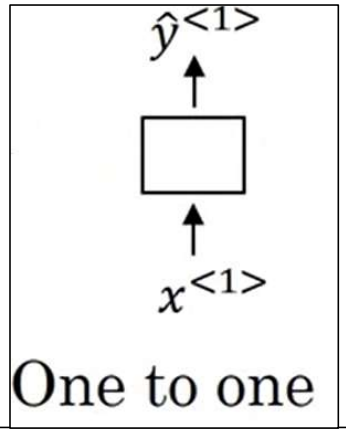
- So, far we have seen that the number of sequences in the inputs and outputs are equal. But it is not true always.
- So, depending upon the number of sequences in inputs and outputs, there are following types of RNNs:

1. One-to-One
2. One-to-Many
3. Many-to-One
4. Many-to-Many (when $T_x = T_y$)
5. Many-to-Many (when $T_x \neq T_y$)

Types of RNNs (Contd….)

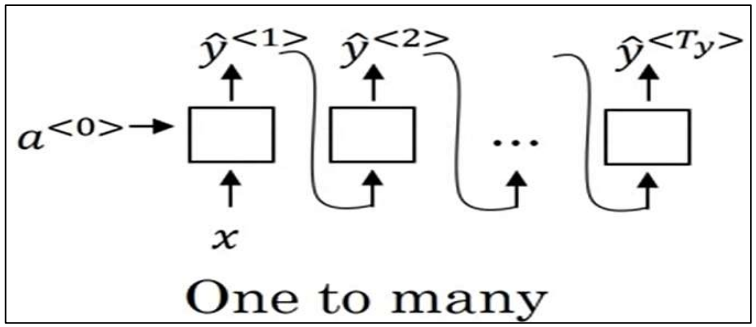
One-to One

- It is not of any interest in sequential modelling where inputs are in sequences.
- It is equivalent to standard neural network



One-to-Many

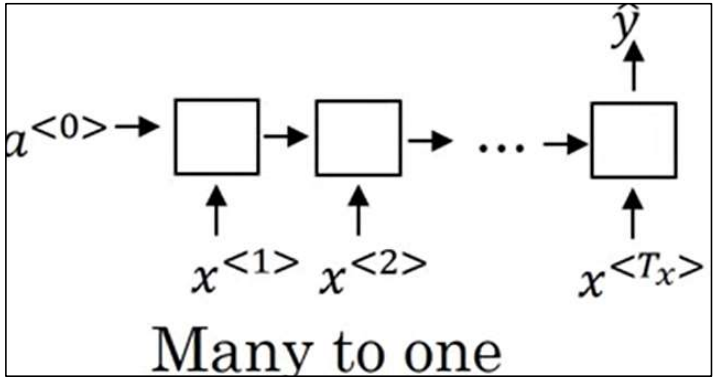
- These types of RNNs are used where there is one input and sequence of outputs is to be generated.
- It is used in types of applications like music generation where there is one input like genre of music to be produced and the output is sequence of music outputs.



Types of RNNs (Contd….)

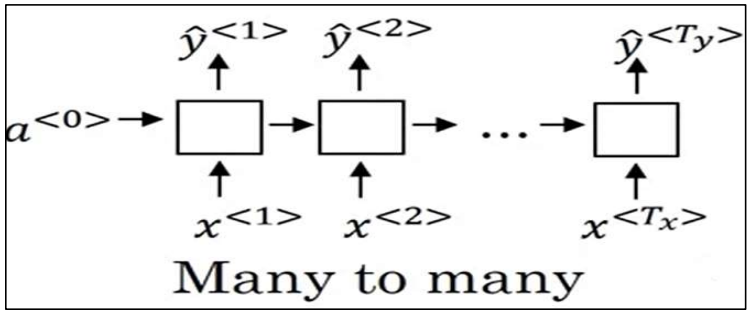
Many-to-One

- Used when the number of sequences in input are more than one and output is only one.
- These types of RNNs are used in applications like sentiment analysis where input is text of any length and output is label 0 or 1 (or ratings between 0 to 5)



Many-to-Many (when T_x=T_y)

- Used when number of input sequences and output sequences are equal.
- These type of RNNs are used in applications like named entity recognition where for each input word a label 0/1 is assigned indicating whether it is named entity or not.



Types of RNNs (Contd….)

Many-to-Many (when $T_x \neq T_y$)

- Used when number of input and output sequences are many, but are of unequal length.
- It is used in applications like Machine translation where an input word may be of French and the output word is of Hindi, which are of unequal length.
- This type of neural network architecture has two parts, the first one is ***encoder*** which takes the French input sentence, and the next one is ***decoder*** which outputs the sentence in different language.