IST 690 Independent Study Deep Learning: Sequence Models

by

Luigi Penaloza

Professor Stephen Wallace

The fifth and final course of Andrew Ng's Deep Learning Specialization on Coursera deals with sequence models. The last course teaches how to make computers see, but this course demonstrates how sequence networks can be applied for audio and text data.

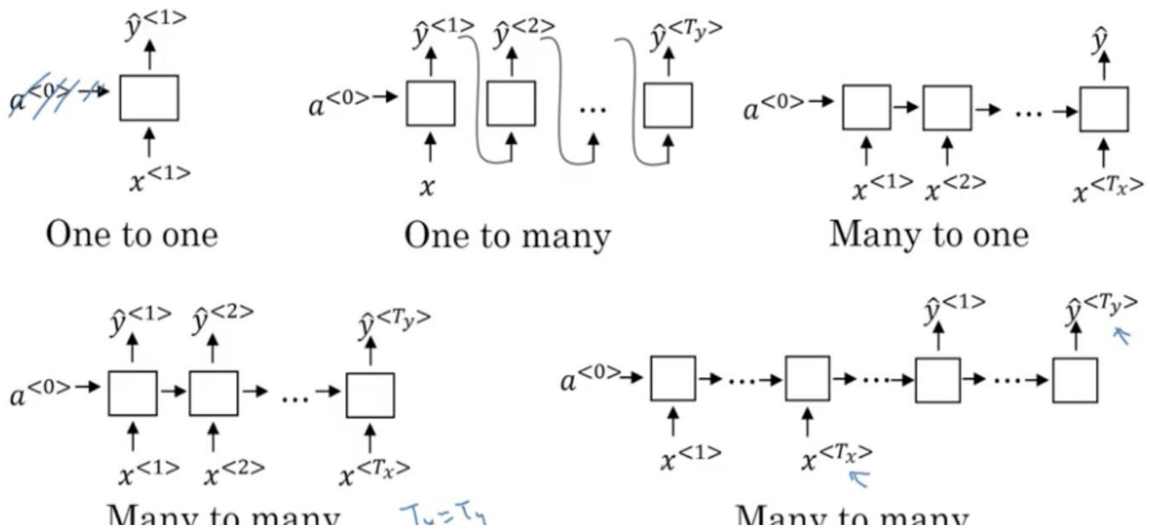
Although this course proved to be the most difficult, the concepts learned of recurrent neural networks, gated recurrent unit (GRU) and long short- term memory (LSTM), including their bidirectional implementations make it one of the most valuable.

By the end of the course we start to realize the complexity of the concepts such as backpropagation through time, word embedding or beam search, and realize how good Andrew is at making the quizzes and assignments be straight forward. Assignments for this course go over how to develop recurrent neural networks that learn from sequences of text or audio to learn similar content.

Deliverable include a model that comes up with names for dinosaurs, and a model that writes a poem in the style of Shakespeare given an already established sequence. From the character level language model to come up with new names for dinosaurs we learn about storing text data using an RNN, synthesization of data by sampling predictions at each step and passing it to the next RNN cell unit, as well as how to build a character level text generation recurrent neural network and finally about the importance of clipping the gradients.

The reason for using sequence models is because traditional neural networks do not share features across positions of the network, therefore not working in sequence prediction given that previous inputs are important for predicting the outputs. In predicting words like the Shakespearean poem, we want to know at least a few words before the target word.

Sequence models address different prediction applications. Network types that Andrew Ng discuss include one to one, one to many , many to on and many to many networks.


$\hat{y}^{<1>}$
$a^{<0>}$
$x^{<1>}$
One to one
$\hat{y}^{<1>}$ $\hat{y}^{<2>}$ $\hat{y}^{<T_y>}$
$a^{<0>}$
x
One to many
$\hat{y}$
$a^{<0>}$
$x^{<1>}$ $x^{<2>}$ $x^{<T_x>}$
Many to one
$\hat{y}^{<1>}$ $\hat{y}^{<2>}$ $\hat{y}^{<T_y>}$
$a^{<0>}$
$x^{<1>}$ $x^{<2>}$ $x^{<T_x>}$
Many to many
$T_x = T_y$
$\hat{y}^{<1>}$ $\hat{y}^{<T_y>}$
$a^{<0>}$
$x^{<1>}$ $x^{<T_x>}$
Many to many

Applied examples of different models:

One to Many- Music generation where input is an empty set and output is song.
Many to One- Financial volatile forecasting where input is a stream of quotes and trades over a specific time, and output is volatile prediction.
Many to One- Handles operations where input and output sequences are not the same. Also called sequence to sequence models.

There can be some complications with RNN's however. Andrew explains that recurrent neural networks could have gradients that vanish, making it difficult for networks to learn long term dependencies. Exploding gradients can be easily clipped using the gradient clipping algorithm.

Grated Recurring Unites (GRU's) can be used to address the vanishing gradient problem by adding update and reset gates. Update gate to see how much precious information need to be passed to the next step and reset gate to see how much information is irrelevant and should be forgotten. We also learn that LSTM can be used to address long term dependencies like vanishing gradient problems as well, but Andrew recommends starting with GRU's because of simplicity

By the second week Andrew teaches about word embeddings and about problems with social biases, and even gives solutions for dealing with them.

Word embeddings are basically a vector representation of a given word, and they can be trained using Word2Vec, as well as with the Glove algorithm. The may be trained on very large corpus of text, but also smaller one for tasks such as sentiment classification.

Steps to deal with social biases in word embeddings:

- Identify-   The first step is to identify a direction (or, more generally, a subspace) of the embedding that captures the bias.
- Neutralize- For the second step, Neutralize ensures that gender neutral words are zero in the gender subspace.
- Equalize- Perfectly equalizes sets of words outside the subspace and thereby enforces The property that any neutral word is equidistant to all words in each equality set.

In the last assignment we get to build a trigger word detection, similar to those used by Google Home devices or even Amazon Echo. These trigger word detection would be the equivalent to "Hey Siri!" for Apple or "OK Google" for Google. This is the process of turning soundwaves in the air into something an IPhone can understand. In conclusion the knowledge in this course can be applied to any type of sequence data or time series data, and the advancement on speech recognition thanks to Deep Learning makes for exciting innovations to come.