Lab8 LSTM based Language Model

COMP4901K and MATH 4824B
Fall 2018

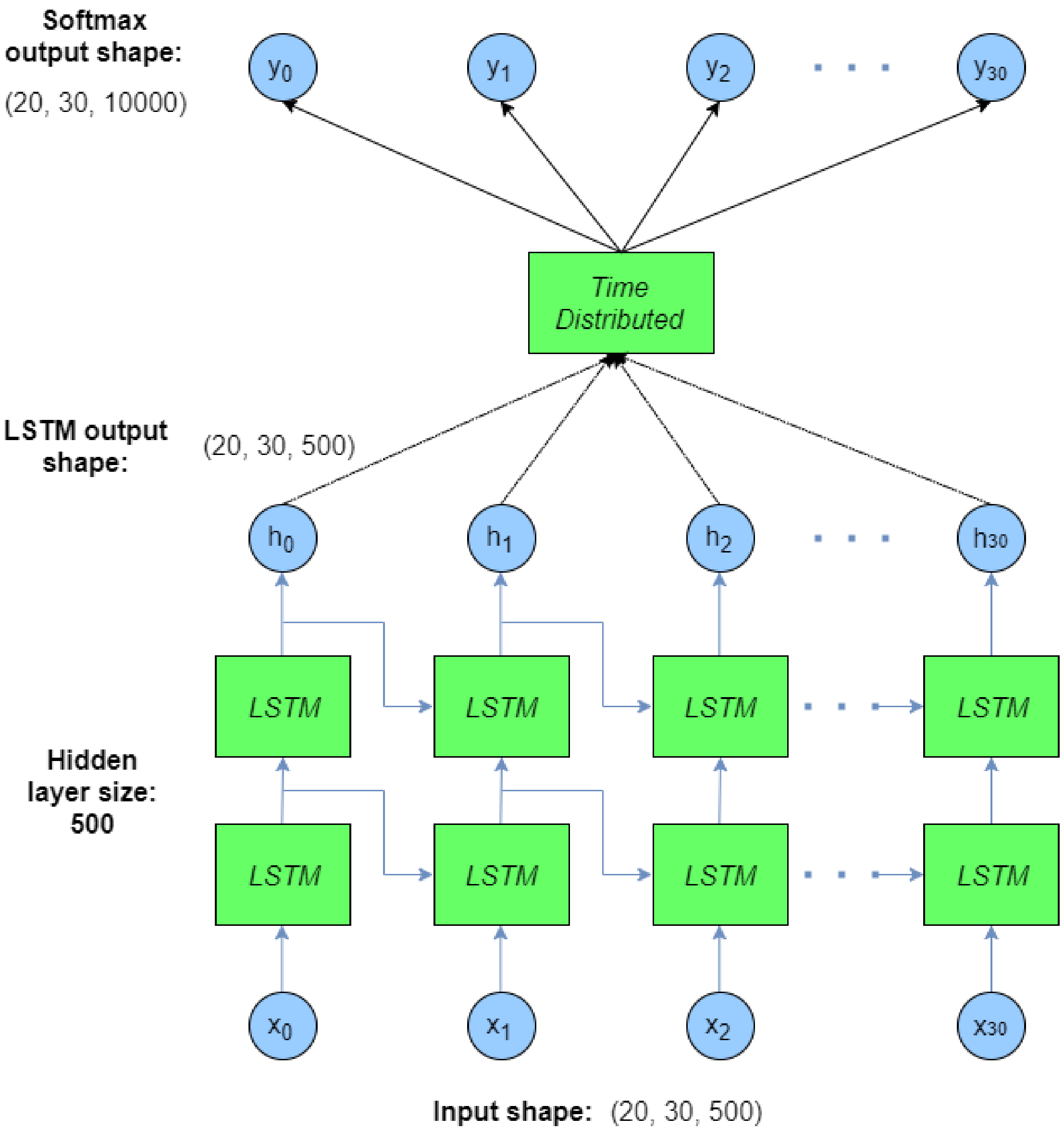

Figure 1: Illustration of 2-layer LSTM based language model. Image Source: `http://adventuresinmachinelearning.com/keras-lstm-tutorial/`

LSTM

Suppose the number of words in each sentence is `seq_len`, and each word is associated with an embedding where the dimension of the embedding `emb_dim`.

In this lab, we will use `LSTM` and `TimeDistributed` in `Keras.layers`.

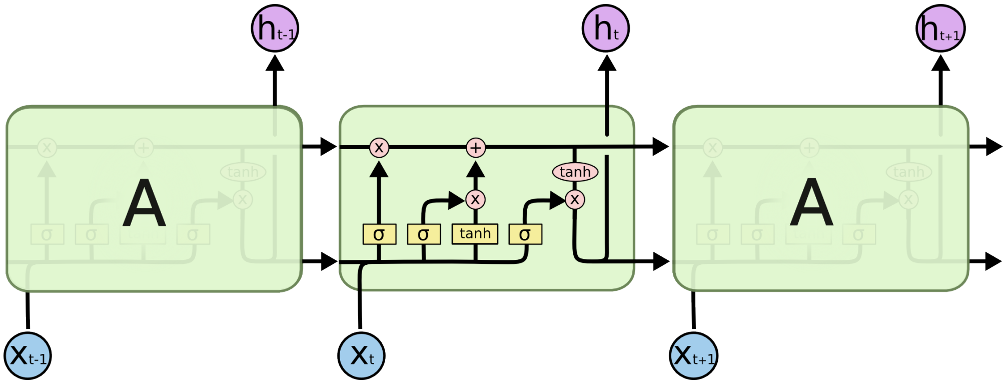

Figure 2: Illustration LSTM. Image Source: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

- `LSTM(units, dropout=0.0, recurrent_dropout = 0.0, return_sequences=False, ...)`

  Arguments

  `units`: Positive integer, dimensionality of the output space.

  `dropout`: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the inputs.

  `recurrent_dropout`: Float between 0 and 1. Fraction of the units to drop for the linear transformation of the recurrent state.

  `return_sequences`: Boolean. Whether to return the last output in the output sequence, or the full sequence.

  `...`: other arguments. In this lab, we just set it default. For more details, please read

  `https://keras.io/layers/recurrent/`.

  Input shape

  3D tensor with shape: `(batch_size, seq_len, input_dim)`

  Output shape

  3D tensor with shape: `(batch_size, seq_len, units)`

  Example

  The first and second layer of LSTM in Figure 1 should be implemented as:

  `LSTM(units=500, dropout = 0.5, recurrent_dropout = 0.5, return_sequences=True)`

- `TimeDistributed(layer)`

  Arguments

  `layer`: a layer instance. For more details, please read `https://keras.io/layers/wrappers/`.

  Example

  This wrapper applies a layer to every temporal slice of an input. The input could be at least 3D, and the dimension of index one will be considered to be the temporal dimension.

  Given an input with the shape of `(batch_size,seq_len,units)`, you can then use `TimeDistributed` to apply a `Dense` layer to each of timestep, independently:

```
TimeDistributed(Dense(units=vocab_size,activation=’softmax’))
```

Then the output would be a tensor with the shape of `(batch_size,seq_len,vocab_size)`

Advanced Reading Materials

- Keras: `https://keras.io/`.
- LSTM: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.
- The state-of-the-arts: `https://arxiv.org/pdf/1707.05589.pdf`