Recurrent Networks in Lasagne

As mentioned above, recurrent layers and feed-forward layers expect different inputs; in a feed-forward layer, the input is a sequence of vectors, whereas for recurrent layers, their input is a sequence of sequences. This difference is due to the fact that the network expects batches of time sequences as input; here we will demonstrate usage on a common long time series regression task. The goal for each time series is to output the sum of the values in the first dimension at the indices where the corresponding value is one in the binary mask. Here, we supply the gate parameters for each gate separately.

Theano includes classes for a "vanilla" (densely connected) recurrent layer, a recurrent layer with arbitrary input-to-hidden and hidden-to-hidden connections, an LSTM layer, and a Bi-LSTM layer. Lasagne includes classes for a "vanilla" recurrent layer, a recurrent layer with arbitrary input-to-hidden and hidden-to-hidden connections, and an LSTM layer.

Recurrent layer shape conventions

Apart from taking in the layer's input connection, the number of units, and the nonlinearity, we also have to specify the desired recurrent shape:

- \( n \times \) (sequence) (for one-dimensional sequences)
- \( n \times \) (sequence) x \( m \) (for two-dimensional sequences)
- \( n \times \) (sequence) x \( m \times \) (sequence) (for two-dimensional sequences where both of these dimensions are variable for us so we will use \( \ldots \))

In order to cut down on the number of constructor arguments and make changing initialization schemes more convenient, the \( \text{LSTMLayer} \) has a \( \text{learn_init} \) attribute which is \( \text{True} \) by default. When \( \text{learn_init} \) is \( \text{True} \), initialization vectors are learned from the data, rather than initializing them with zeros. Sometimes these initial values are learned as parameters of the network, sometimes they aren't; you can decide whether you want to include them as parameters to be optimized. For \( \text{LSTMLayer} \), \( \text{learn_init} \) is \( \text{True} \) by default, so \[ \text{LSTMLayer}(n_{\text{units}}, \text{learn_init} = \text{False}) \] will give you a \( \text{LSTMLayer} \) with all initial values set to zeros.

Because not all sequences in each minibatch will always have the same length, all recurrent layers in Lasagne have an extra dimension to accommodate variable sequence lengths:

\[ \text{batch \\times \\ n \\ \text{time steps} \\ \times \\ \text{sequence dimensions} } \]

The \( \text{batch} \) dimension can be set to \( \text{unroll_scan} = \text{False} \), which means that it can vary from batch to batch.

The \( \text{LSTMLayer} \) is a densely connected recurrent layer, where the cell state contains information from the past and the output at each time step is equal to the cell state. The cell state is computed as the result of the following matrix vector addition:

\[ \text{l_out} = \text{l_in} + \text{l_lstm} \]

In order to compute the cell state, different nonlinearity.

Some people use this, some people don't, you can decide which one is better for your problem. When \( \text{unroll_scan} \) is \( \text{False} \), the \( \text{LSTMLayer} \) will use a \( \text{GRU} \) instead of an \( \text{LSTM} \). When \( \text{unroll_scan} \) is \( \text{True} \), the \( \text{LSTMLayer} \) will use a \( \text{LSTM} \) instead of a \( \text{GRU} \). When \( \text{unroll_scan} \) is \( \text{False} \), the \( \text{LSTMLayer} \) will use a \( \text{GRU} \) instead of an \( \text{LSTM} \). When \( \text{unroll_scan} \) is \( \text{True} \), the \( \text{LSTMLayer} \) will use a \( \text{LSTM} \) instead of a \( \text{GRU} \). When \( \text{unroll_scan} \) is \( \text{False} \), the \( \text{LSTMLayer} \) will use a \( \text{GRU} \) instead of an \( \text{LSTM} \). When \( \text{unroll_scan} \) is \( \text{True} \), the \( \text{LSTMLayer} \) will use a \( \text{LSTM} \) instead of a \( \text{GRU} \). When \( \text{unroll_scan} \) is \( \text{False} \), the \( \text{LSTMLayer} \) will use a \( \text{GRU} \) instead of an \( \text{LSTM} \). When \( \text{unroll_scan} \) is \( \text{True} \), the \( \text{LSTMLayer} \) will use a \( \text{LSTM} \) instead of a \( \text{GRU} \). When \( \text{unroll_scan} \) is \( \text{False} \), the \( \text{LSTMLayer} \) will use a \( \text{GRU} \) instead of an \( \text{LSTM} \). When \( \text{unroll_scan} \) is \( \text{True} \), the \( \text{LSTMLayer} \) will use a \( \text{LSTM} \) instead of a \( \text{GRU} \).