Recurrent nets with keras

Professor Marie Roch

Sequences in keras

- Input sequences are tensors
- Each tensor has the following shape: $\text{example} \times \text{time} \times \text{dim}$
Varying length sequences

- Sequences can vary in length, but tensors cannot
- Zero-pad to a standard length

Zero-padding can be expensive when there are a few long sequences.

Solution: use a mini-batch generator
  - Create an object that is iterable
  - __next__() returns (examples, labels) containing a minibatch.
  - The minibatch is zero-padded to the longest element of the minibatch
Varying length sequences

• To train, use fit_generator() instead of fit
  – The minibatch generator should be the first parameter
  – The second parameter is the number of times next() must be called on the minibatch generator before an epoch has passed
  – See documentation for further details

Input layer

• The input_shape to the first layer should be [None, dim] – expect arbitrary # of fixed dimension vectors

• Masking layer can be used to tell RNN to ignore time steps with a specified value (“mask_value”:constant_value)
Feed forward layers on time-series

- If feed-forwards layers are desired prior to the recurrent layer, use a layer wrapper, e.g.:
  
  ```python
  TimeDistributed(Dense(40))
  ```
- This does the following:
  - pass time slices one by one through Dense layer
  - reconstruct a tensor to be used by the next layer

Recurrent layers

- Recurrent layers (e.g. LSTM, GRU) are added like any other layer and can be followed by dropout layers.
LSTM options

• recurrent_regularizer: Allows specification of regularizer for the recurrent weights

• return_sequences
  – True – A sequence of outputs is generated
  – False – Only the last output is returned.
    Appropriate when a decision is to be made based on the state

LSTM options

• return_state – If True, returns the unit state in addition to the output

• go_backwards – If True, dependent on future inputs

• unroll – If True, network is unrolled.
  Faster, but inappropriate for long sequences
LSTM options

- stateful – If True, subsequent batches have a continuation of the current state for each example, e.g. initial state of example 3 in batch 2 is the final state of example 3 in batch 1 (appropriate for long time-series)

Many to 1 classification

- To classify a sequence, use return_sequences=False on last recurrent layer.
- Then add feed-forward layers as appropriate