Framewise Phoneme Classification with Bidirectional LSTM Networks

Alex Graves and Jurgen Schmidhuber
IJCNN 2005 conference proceedings

Chase Duncan

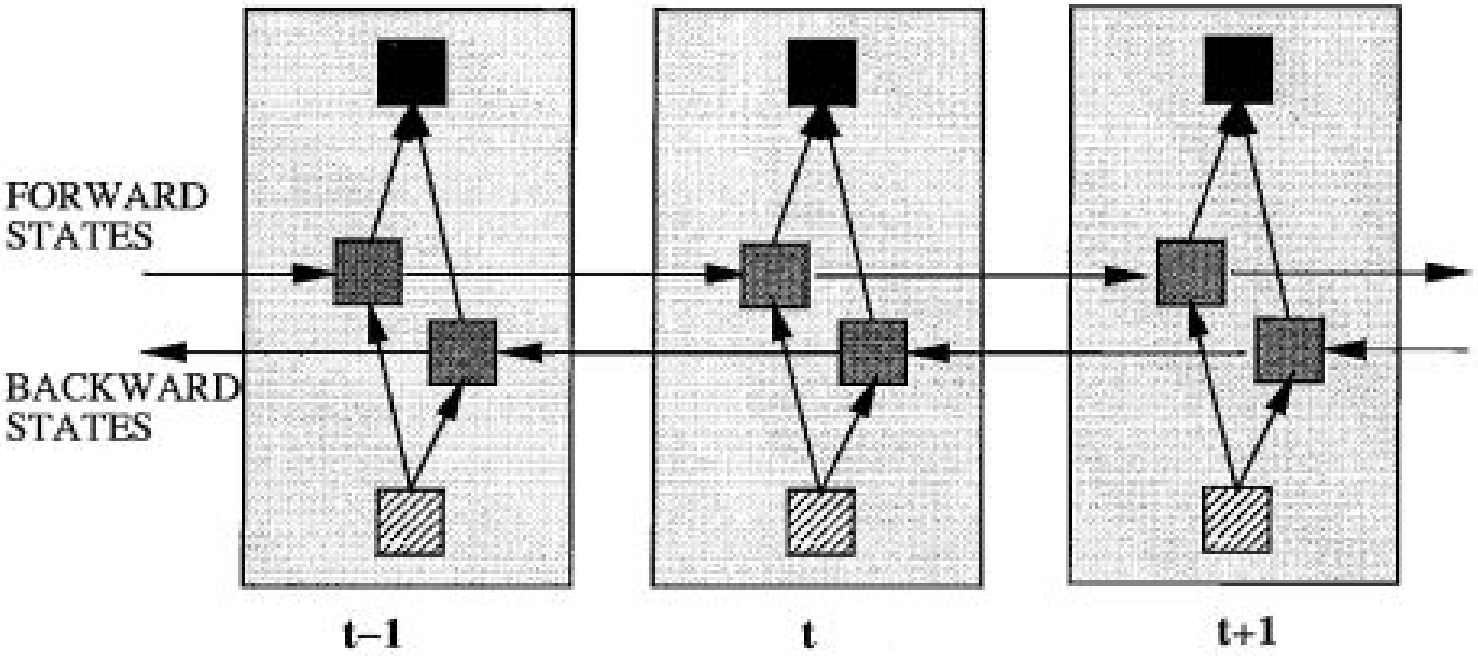
Overview

1. Bidirectional LSTM overview
2. What sort of tasks are they appropriate for?
3. Example: phoneme classification
4. How do we train them?

1. Bidirectional LSTM overview

- Bidirectional LSTMs encode sequential data by processing the input both forward and reverse
- In practice -- two LSTMs whose outputs are combined in some way (e.g. concatenation) to create a single encoding
- The case for this approach is supported by the argument that humans do this -- “whole sentences that at first mean nothing are found to make sense in the light of future context”

Bidirectional LSTM (Overview cont.)



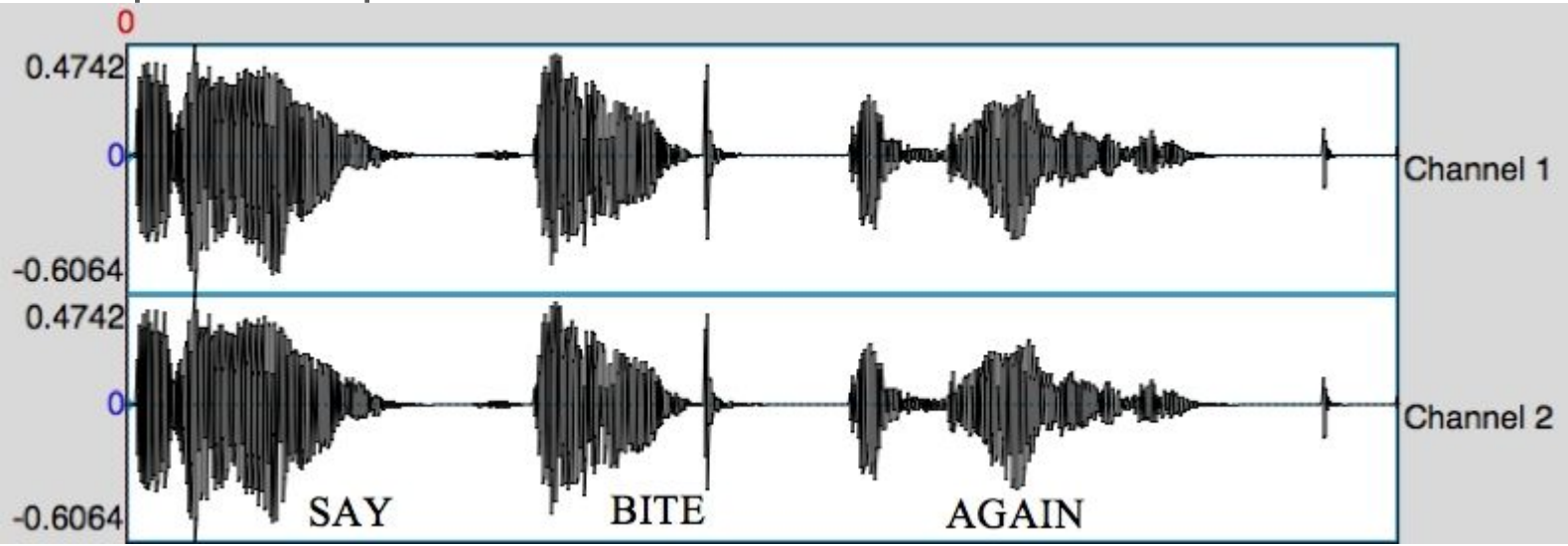
http://paddlepaddle.org/docs/develop/documentation/en/howto/deep_model/rnn/rnn_config_en.html

2. What are they good for?

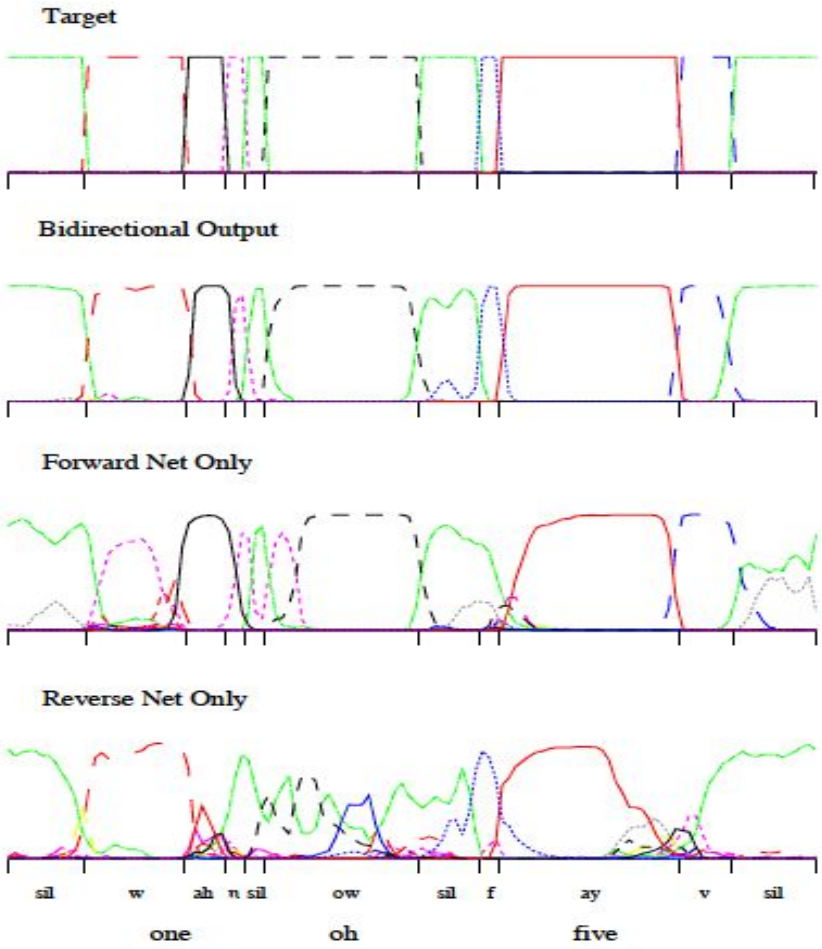
- BiLSTMs (or more generally BiRNNs) are useful when the data can be divided into finitely long segments each of which is unaffected by the others
- If a task is truly online, i.e. an output is expected for every input, then BiRNNs are useless
- Some examples of tasks for which BiRNNs have proven useful: machine translation, speech recognition, protein structure prediction, named entity recognition, entity linking
- Natural fit for language

3. An example: phoneme classification

Problem: classify a sequence of frames of acoustic data (an utterance) into a sequence of phonemes, i.e. units of sound



https://soundphysics.ius.edu/wp-content/uploads/2014/01/saybiteagainN.jpg


Target
Bidirectional Output
Forward Net Only
Reverse Net Only
sil w ah n sil ow sil f ay v sil
one oh five

Data breakdown (an example cont.)

- Each frame is 5ms and broken down in a vector of 26 Mel-Frequency Cepstrum Coefficients (MFCC)
- 3696 utterances constituting 1,124,823 frames in training set
- 1344 utterances constituting 410,920 frames in test set

Network topologies

- Unidirectional net with hidden LSTM layer containing 93 memory blocks, one cell each
- Bidirectional net with two hidden LSTMs (one forwards, one backwards) each containing 93 one cell memory blocks
- Unidirectional RNN with one hidden layer containing 185 sigmoidal units
- A bidirectional RNN with two hidden layers containing 185 sigmoidal units

Network topologies cont.

Unidirectional nets have roughly 50,000 weights, bidirectional have roughly 100,000. All nets contain input layer of size 26 and output layer of size 61 (one for each phoneme).

Results

FRAMEWISE PHONEME CLASSIFICATION ON THE TIMIT DATABASE

System	Training Set	Test Set	Epochs
Bidirectional LSTM	79.4%	69.5%	35
LSTM	77.2%	65.5%	70
Bidirectional RNN	67.2%	64.7%	65
RNN	68.3%	64.5%	100

Key findings:

- BiLSTM achieve better results much faster
- BiLSTM prone to overfitting -- see relative differences between train and test
- Given the proportion of frames to weights, it’s likely not just memorizing the data

Training: forward pass

Input Gates:

$$x_\iota = \sum_{j \in N} w_{\iota j} y_j(\tau - 1) + \sum_{c \in C} w_{\iota c} s_c(\tau - 1)$$

$$y_\iota = f(x_\iota)$$

Forget Gates:

$$x_\phi = \sum_{j \in N} w_{\phi j} y_j(\tau - 1) + \sum_{c \in C} w_{\phi c} s_c(\tau - 1)$$

$$y_\phi = f(x_\phi)$$

Cells:

$$\forall c \in C,\ x_c = \sum_{j \in N} w_{cj} y_j(\tau - 1)$$

$$s_c = y_\phi s_c(\tau - 1) + y_\iota g(x_c)$$

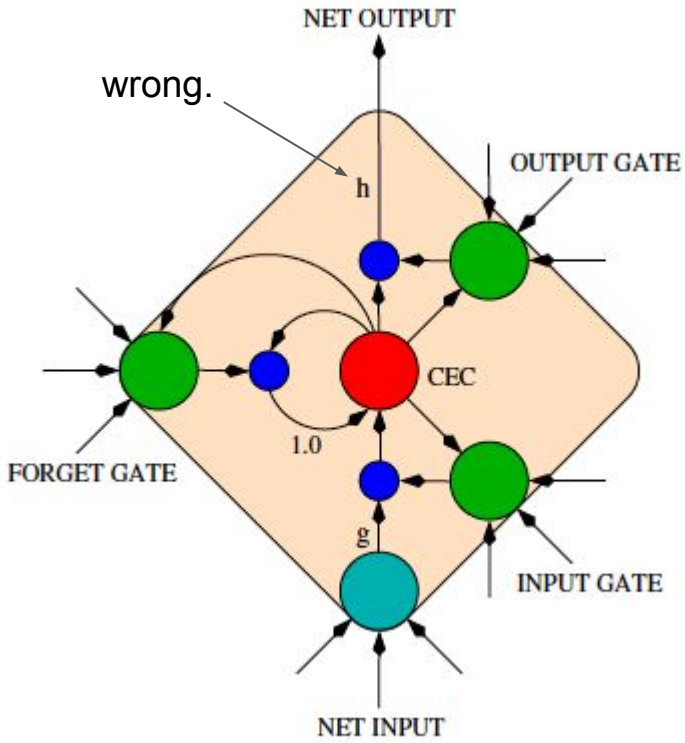
Output Gates:

$$x_\omega = \sum_{j \in N} w_{\omega j} y_j(\tau - 1) + \sum_{c \in C} w_{\omega c} s_c(\tau)$$

$$y_\omega = f(x_\omega)$$

Cell Outputs:

$$\forall c \in C,\ y_c = y_\omega h(s_c)$$



Backward pass: BPTT

- Backpropagation through time is essentially repeated applications of the chain rule through the network to compute the derivative of the error with respect to each node
- E.G.

$$define \quad \delta_k(\tau) = \frac{\partial E(\tau)}{\partial x_k}$$

$$\delta_k(\tau) = y_k(\tau) - t_k(\tau) \qquad k \in output\ units$$

Cell Outputs:

$$\forall c \in C, \ \epsilon_c = \sum_{j \in N} w_{jc} \delta_j(\tau + 1)$$

Output Gates:

$$\delta_\omega = f'(x_\omega) \sum_{c \in C} \epsilon_c h(s_c)$$

What's the difference in training BiLSTM?

- There really isn't one (besides bookkeeping)
- Forward
  - Do a forward pass for forward states, do backward pass for backward states, concatenate
  - Do a forward pass for output layer
- Backward
  - Do a backward pass for output layer
  - Backward pass for forward states, backward pass for backward states
- In some sense, the memory cells have no notion of forward/backward w.r.t. the data

Thank you