Speech Recognition with Deep Recurrent Neural Networks

1 Citation

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks." Acoustics, speech and signal processing (icassp), 2013 ieee international conference on. IEEE, 2013.

`https://arxiv.org/pdf/1303.5778.pdf`

2 Abstract

We use Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNN) for speech recognition and set state of the art with a 17.7% error rate.

3 Recurrent Neural Networks

Neural nets combined with Hidden Markov Models (HMM) are popular for speech recognition, but maybe an RNN can be used instead. This avoids having to predict something for every input vector (i.e. no need for imprecise alignment for training the neural net) and might be able to learn temporal patterns better than the HMM.

4 Network Training

Given input $\mathbf{x} = (x_1, ..., x_T)$, the RNN computes hidden vectors $\mathbf{h} = (h_1, ..., h_T)$, which get turned into predictions $\mathbf{y} = (y_1, ..., y_T)$. The RNN equations are:

$$h_t = \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{1}$$
$$y_t = W_{hy}h_t + b_y \tag{2}$$

We use an LSTM as the $\mathcal{H}$:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{3}$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{4}$$
$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{5}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{6}$$
$$h_i = o_t \tanh(c_t) \tag{7}$$

An LSTM reads the input from first item to last item. It would also be helpful to go in the reverse direction. For this purpose, we have a bidirectional RNN:

$$\overrightarrow{\mathbf{h}}_t = \mathcal{H}(W_{x\overrightarrow{\mathbf{h}}} x_t + W_{\overrightarrow{\mathbf{h}}\overrightarrow{\mathbf{h}}} \overrightarrow{\mathbf{h}}_{t-1} + b_{\overrightarrow{\mathbf{h}}}) \tag{8}$$

$$\overleftarrow{\mathbf{h}}_t = \mathcal{H}(W_{x\overleftarrow{\mathbf{h}}} x_t + W_{\overleftarrow{\mathbf{h}}\overleftarrow{\mathbf{h}}} \overleftarrow{\mathbf{h}}_{t+1} + b_{\overleftarrow{\mathbf{h}}}) \tag{9}$$

$$y_t = W_{\overrightarrow{\mathbf{h}}y} \overrightarrow{\mathbf{h}}_t + W_{\overleftarrow{\mathbf{h}}y} \overleftarrow{\mathbf{h}}_t + b_y \tag{10}$$

Next, we can also stack LSTM layers (input is $h^0 = \mathbf{x}$ and output is $y_t = W_{h^N y} h_t^N + b_y$)

$$h_t^n = \mathcal{H}(W_{h^{n-1}h^n} h_t^{n-1} + W_{h^n h^n} h_{t-1}^n + b_h^n) \tag{11}$$

We can replace h^n with $\overrightarrow{\mathbf{h}}^n$ and $\overleftarrow{\mathbf{h}}^n$ to stack a bidirectional RNN.

5 Network Training

Suppose there are K possible phonemes and our goal is to predict target sequence $\mathbf{z}$ (length U) given input sequence $\mathbf{x}$ (length T). The Connectionist Temporal Classification (CTC) approach adds a $K+1$ layer softmax (one per phoneme and one blank) to the net. It runs the forward backward algorithm to compute $P(\mathbf{z}|\mathbf{x})$. We define (for $y_t[k]$ as the kth element of the vector y_t):

$$y_t = W_{\overrightarrow{\mathbf{h}}^N y} \overrightarrow{\mathbf{h}}_t^N + W_{\overleftarrow{\mathbf{h}}^N y} \overleftarrow{\mathbf{h}}_t^N + b_y \tag{12}$$

$$P(k|t) = \frac{\exp(y_t[k])}{\sum_{k'=1}^K \exp(y_t[k'])} \tag{13}$$

CTC predicts an output phoneme given the audio sequence, but if we combine it with an RNN Transducer, we can predict the phoneme given the audio sequence and previous phonemes (so it's like a language model). We produce a $P(k|t,u)$ for all input timesteps t and output timesteps u. Again, we have $K+1$ possible outputs (phonemes and blank). We can run the forward-backward algorithm to recover $P(\mathbf{z}|\mathbf{x})$ from $P(k|t,u)$. Suppose we have our stacked bidirectional LSTM outputs from CTC $\overrightarrow{\mathbf{h}}^N$ and $\overleftarrow{\mathbf{h}}^N$ - this network aims to predict $P(k|t)$. We have another network, called the prediction network, that aims to predict $P(k|u)$ - denote its output as $\mathbf{p}$. We then combine them as follows:

$$l_t = W_{\overrightarrow{\mathbf{h}}^N l} \overrightarrow{\mathbf{h}}_t^N + W_{\overleftarrow{\mathbf{h}}^N l} \overleftarrow{h}_t^N + b_l \tag{14}$$

$$h_{t,u} = \tanh(W_{lh} l_{t,u} + W_{pb} p_u + b_h) \tag{15}$$

$$y_{t,u} = W_{hy} h_{t,u} + b_y \tag{16}$$

$$P(k|t,u) = \frac{\exp(y_{t,u}[k])}{\sum_{k'=1}^K \exp(y_{t,u}[k'])} \tag{17}$$

All our hidden states have the same dimension. We pretrain the CTC network and prediction network separately, and then use the results to initialize the transducer and train that. At test time, we can decode with beam search. For regularization, we use early stopping and weight noise (i.e. add Gaussian noise to weights during training).

6 Experiments

We train on TIMIT. Our feature vectors are 123 dimensional vectors consisting of Fourier transformed filter banks with 40 coefficients (plus energy) distributed on a mel-scale with their first and second temporal derivatives. We make sure our feature vectors have zero-mean and unit variance. We have $K = 61$ possible phonemes. Our best network used the RNN transducer with pretrained CTC net and prediction net. It had 3 hidden layers and 250 hidden units per layer. We decode with a beam size of 100.

Our main takeaways are: (1) LSTMs help a lot (2) bidirectional RNNs help a lot (3) depth is more important than layer width (4) the transducer is more useful when we do pretraining.

7 Conclusions and Future Work

We could extend this to Large Vocabulary Speech Recognition (instead of just training on TIMIT). We could also see if we could combine this with frequency-domain convolutional neural networks.