Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks

Alex Graves, Santiago Fernandez, Faustion Gomez, Jürgen Schmidhuber

Presented By
Ashiq Imran
Outline

• Recurrent Neural Network (RNN)
  ➢ RNN for Sequence Modeling
• Encoder-Decoder Model
• Connectionist Temporal Classification (CTC)
  ➢ CTC Problem Definition
  ➢ CTC Description
  ➢ CTC Objective
  ➢ How CTC collapsing works
  ➢ CTC Objective and Gradient
  ➢ Experiments
  ➢ Results
  ➢ Summary
Recurrent Neural Network (RNN)

• Recurrent since they receive inputs, update the hidden states depending on the previous computations, and make predictions for every element of a sequence.

• RNNs are a neural network with memory.

• RNNs are very powerful dynamic network for sequential tasks such as speech or handwritten recognition since they maintain a state vector that implicitly contains information about the history of all the past elements of a sequence.
RNN for Sequential Modeling

An unrolled RNN (in time) can be considered as a deep neural network (DNN) with indefinitely many layers:

\[ s_t = f(Ux_t + Ws_{t-1}) \]
\[ y = g(Vs_t) \]

- \( x_t \): input at time \( t \)
- \( s_t \): hidden state at time \( t \) (memory of the network).
- \( f \): is an activation function (e.g, \( \text{tanh}() \) and ReLUs).
- \( U, V, W \): network parameters (unlike a feedforward neural network, an RNN shares the same parameters across all time steps).
- \( g \): activation function for the output layer (typically a softmax function).
- \( y \): the output of the network at time \( t \)
Encoder Decoder Model

• Most commonly used framework for sequence modeling with neural networks.
• The encoder maps the input sequence X into a hidden representation.
• The decoder consumes the hidden representation and produces a distribution over the outputs.

\[ H = \text{encode}(X) \]
\[ P(Y | X) = \text{decode}(H) \]

• Encode(.) and Decode(.) functions are typically RNNs.
• Example, Machine Translation, Speech recognition, so on
Connectionist Temporal Classification (CTC)

• Scenario:
  • A lot of real world problem needs to predict sequence label from unsegmented or noisy data. For example, speech recognition, handwriting recognition and so on.

• Problem:
  • Requires pre-segmentation of input sequence
  • Requires post processing of transforming output to label sequence

• Proposed Method:
  • Training RNN directly so that it can take unsegmented input sequences and produce output labelling sequences

Connectionist Temporal Classification (CTC)

- Method for labeling unsegmented data sequences
  - Raw waveforms and text transcription
- Differentiable objective function
  - Gradient based training
  - $O^{ML}(S, N_w) = -\sum_{(x,z) \in S} \ln(p(z|x))$
- Used in various application,
  - Speech recognition
  - Handwriting recognition
  - So on

CTC Problem Definition

• Training examples \( S = \{ (x_1, z_1), \ldots, (x_N, z_N) \} \in D_{X \times Z} \)
  • \( x \in X \) are waveforms
  • \( z \in Z \) are text transcripts

• Goal: train temporal classifier \( h : X \rightarrow Z \)

• Architecture
  • Input layer accepts audio frame
  • Some network (usually RNN)
  • Softmax output over phones
CTC Description

• A CTC output layer contains as many units as there are labels in the task, in addition it includes ‘blank’ or ‘no label’ unit.

• Given a length $T$ input sequence $x$, the output vectors $y_t$ are normalized with the softmax function, then interpreted as the probability of emitting the label (or blank) with index $k$ at time $t$:

$$p(k, t|x) = \frac{e^{y_t^k}}{\sum_{k'} e^{y_t^{k'}}}$$

where $y_t^k$ is element $k$ of $y_t$.

• A CTC alignment $a$ is a length $T$ sequence of blank and label indices. The probability $p(a|x)$ is the product of the emission probabilities at every time step:

$$p(a|x) = \prod_{t=1}^{T} p(a_t, t|x)$$

CTC Description

- Let $\mathcal{N}_w(x) = \{y^t_k\}$ be NN with a softmax output
  - $y^t_k$ is activation of output unit $k$ at time frame $t$
  - Activations over time define distribution over $L^T$
- Sequences over $L^T \triangleq \pi = \{\pi_1, \ldots, \pi_T\}$ are paths
- Optimize for best path:
  $$P(\pi | x) = \prod_{t=1}^{T} y^{t}_{\pi_t}, \forall \pi \in L^T$$

CTC Objective

- Paths are not equivalent to the label, $\pi \neq l$
- Optimize for best label:
  \[
  P(l|x) = \sum_\pi P(l|\pi)P(\pi|x)
  \]
- Classifier is calculated on most probable labelling (best path)
  \[
  h(x) = \arg \max_{l \in L} p(l|x)
  \]
- Solve objective above with dynamic programming
  - Forward-backward algorithm
    - Forward variables $\alpha$
    - Backward variables $\beta$
    \[
    P(l|x) = \sum_{s=1}^{|l|} \frac{\alpha_t(s)\beta_t(s)}{y_{|s|}^t}
    \]
  - Maps and searches only paths that correspond to target label

How CTC Collapsing works

For an input, like speech

Predict a sequence of tokens

Merge repeats, drop $\epsilon$

Final output

CTC Objective and Gradient

- Objective function:
  \[ O^{ML}(S, \mathcal{N}_w) = - \sum z \ln(p(z|x)) \]

- Gradient:
  \[ \frac{\partial O^{ML} \{\{(x, z)\}, \mathcal{N}_w\}}{\partial u_k^t} = y_k^t \frac{1}{y_k^t Z_t} \sum_{s \in \text{lab}(z, k)} \hat{\alpha}_t(s) \hat{\beta}_t(s) \]

  where \( Z_t \triangleq \sum_{s=1}^{\lvert l' \rvert} \frac{\alpha_t(s) \beta_t(s)}{y_{l's}^t} \)

Experiments

- Dataset: TIMIT speech corpus.
  - It contains recordings of prompted English speech, accompanied by manually segmented phonetic transcripts.
- CTC was compared with both HMM & HMM-RNN
- Bi-directional LSTM (BLSTM) used for both CTC & HMM-RNN Hybrid Model
  - BRNN & Unidirectional RNN gives worse result
Experimental Results

Label Error Rate (LER) on TIMIT.

<table>
<thead>
<tr>
<th>System</th>
<th>LER (mean ± standard error)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Independent HMM</td>
<td>38.85%</td>
</tr>
<tr>
<td>Context Dependent HMM</td>
<td>35.21%</td>
</tr>
<tr>
<td>BLSTM/HMM</td>
<td>33.84% ± 0.06%</td>
</tr>
<tr>
<td>Weighted Error BLSTM/HMM</td>
<td>31.57% ± 0.06%</td>
</tr>
<tr>
<td>CTC (best path)</td>
<td>31.47% ± 0.21%</td>
</tr>
<tr>
<td>CTC (prefix search)</td>
<td>30.57% ± 0.19%</td>
</tr>
</tbody>
</table>

\[
LER(h, S') = \frac{1}{|S'|} \sum_{(x, z) \in S'} \frac{ED(h(x), z)}{|z|}
\]

• With Prefix search decoding, CTC outperforms both HMM & HMM-RNN Hybrid methods

Discussions

• CTC doesn’t explicitly segments its input sequences.
  • Rather allows the network to be trained directly for labelling sequence

• Without task specific knowledge, it outperforms both HMM & HMM-RNN Hybrid model

• For segmented data (protein secondary structure prediction), it may not give good results.

CTC Summary

• For a given transcription sequence, there are as many possible alignments as there are different ways of separating the labels with blanks.

• For example, using ‘−’ to denote blanks, the alignments (a, −, b, c, −, −) and (−, −, a, −, b, c) both correspond to the transcription (a, b, c). When the same label appears on successive time steps in an alignment, the repeats are removed: (a, b, b, b, c, c) and (a, −, b, −, c, c) also correspond to (a, b, c).

• Denoting $B$ as an operator that removes first the repeated labels, then the blanks from alignments, and observing that the total probability of an output transcription $y$ is equal to the sum of the probabilities of the alignments corresponding to it, we can write:

$$p(y|x) = \sum_{a \in B^{-1}(y)} p(a|x)$$

CTC Summary

• Integrating over possible alignments is what allows the network to be trained with unsegmented data.

• Intuition: we don’t know where the labels within a particular transcription will occur, so we sum over all the places where they could occur.

• We can write:

\[ p(y|x) = \sum_{a \in B^{-1}(y)} p(a|x) \]

• This equation can be efficiently evaluated and differentiated using a dynamic programming algorithm.

• Given a target transcription \( \hat{y} \), the network can then be trained to minimize the CTC objective function:

\[ CTC(x) = -\log p(\hat{y}|x) \]
Thank you

• Any Questions?