Lecture 20: Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks

Mark Hasegawa-Johnson

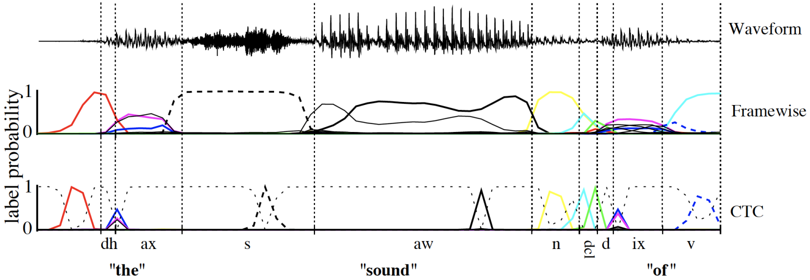
ECE 537, Fall 2022

Outline

Temporal Classification

- $\mathbf{x} = [x_1, \ldots, x_T]$ is the input. Each x_t is usually a vector, $x_t = [x_1^t, \ldots, x_m^t]$.
- $\mathbf{z} = [z_1, \ldots, z_U]$ is the desired network output, where $z_u \in L$ comes from some alphabet L. $U \leq T$.
- The goal is to train a function $h(\mathbf{x})$ so that $\mathbf{y} = h(\mathbf{x})$ is similar to $\mathbf{z}$.

Temporal Classification Example: Speech



Temporal classification maps from a sequence of speech frames (top) to a sequence of phoneme or character labels (bottom).

Graves et al., 2006, Figure 1. (c) ICML

Network Outputs

- An RNN outputs a sequence of vectors, $\mathbf{y} = [y_1, \ldots, y_T]$, where each $y_t = [y^t_1, \ldots, y^t_{|L|}]$ is a pmf:

$$y^t_k \geq 0, \qquad \sum_{k=1}^{|L|} y^t_k = 1$$

- Thus, if $\mathbf{z}$ is time-aligned, we can interpret

$$y^t_k = P(z_t = k|\mathbf{x})$$

Outline

Recurrent Neural Net (RNN)

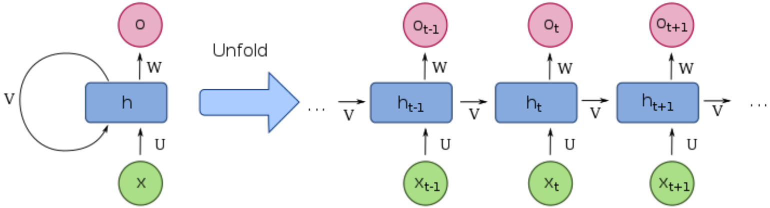


Image CC-SA-4.0 by Ixnay,

`https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg`

Recurrent Neural Net (RNN)

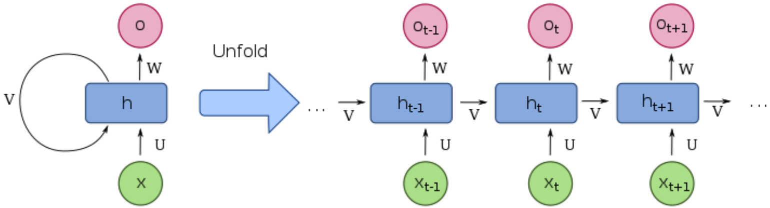
A recurrent neural net defines nonlinear recurrence of a hidden vector, h_t:

$$h_t = \sigma\left(Ux_t + Vh_{t-1}\right)$$
$$y_t = \text{softmax}\left(Wh_t\right)$$

The weight matrices, U, V, and W, are chosen to minimize the loss function. For example, suppose we're using a cross-entropy loss with target sequence $\mathbf{z}$, then

$$\mathcal{L} = -\sum_{t=1}^{T} \ln y_{z_t}^t$$

Partial vs. Full Derivatives



With one-step recurrence, as shown here, $\mathcal{L}$ depends on h_t in exactly two different ways:

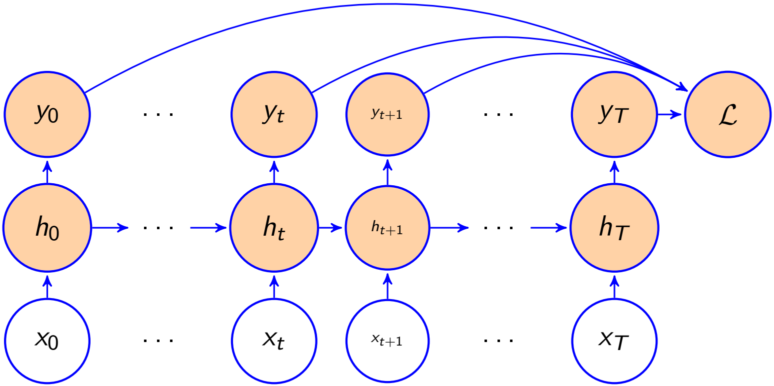
$$\frac{d\mathcal{L}}{dh_t} = \frac{d\mathcal{L}}{dy_t}\frac{\partial y_t}{\partial h_t} + \frac{d\mathcal{L}}{dh_{t+1}}\frac{\partial h_{t+1}}{\partial h_t}$$

Partial vs. Full Derivatives

$$\frac{d\mathcal{L}}{dh_t} = \frac{d\mathcal{L}}{dy_t}\frac{\partial y_t}{\partial h_t} + \frac{d\mathcal{L}}{dh_{t+1}}\frac{\partial h_{t+1}}{\partial h_t}$$

where

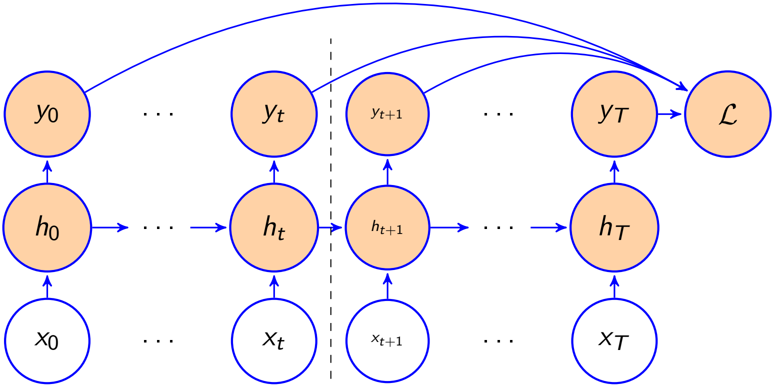
- $\frac{d\mathcal{L}}{dh_t}$ is the total derivative, and includes all of the different ways in which $\mathcal{L}$ depends on h_t.
- $\frac{\partial h_{t+1}}{\partial h_t}$ is the partial derivative, i.e., the change in h_{t+1} per unit change in h_t if x_t is held constant.



Here's a flow diagram that could represent:

$$h_t = \sigma\left(Ux_t + Vh_{t-1}\right), \qquad y_t = \mathrm{softmax}(Wh_t),$$

$$\mathcal{L} = -\sum_{t=0}^{T} \ln y_{z_t}^t$$



Back-propagation through time does this:

$$\frac{d\mathcal{L}}{dh_t} = \frac{d\mathcal{L}}{dy_t}\frac{\partial y_t}{\partial h_t} + \frac{d\mathcal{L}}{dh_{t+1}}\frac{\partial h_{t+1}}{\partial h_t}$$

Partial vs. Full Derivatives

So for example, if

$$\mathcal{L} = -\sum_{t=0}^{T} \ln y_{z_t}^t$$

then the partial derivative of $\mathcal{L}$ w.r.t. h_k^t is

$$\frac{\partial \mathcal{L}}{\partial h_k^t} = -\frac{1}{y_{z_t}^t}\frac{\partial y_{z_t}^t}{\partial h_k^t}$$

and the total derivative of $\mathcal{L}$ w.r.t. h_k^t is

$$\frac{d\mathcal{L}}{dh_k^t} = -\frac{1}{y_{z_t}^t}\frac{\partial y_{z_t}^t}{\partial h_k^t} + \sum_i \frac{d\mathcal{L}}{dh_i^{t+1}}\frac{\partial h_i^{t+1}}{\partial h_k^t}$$

Synchronous Backprop vs. BPTT

The basic idea of back-prop-through-time is divide-and-conquer.

1. **Synchronous Backprop:** First, calculate the **partial derivative** of $\mathcal{L}$ w.r.t. h_k^t, assuming that all other time steps are held constant.

$$\frac{\partial \mathcal{L}}{\partial h_k^t} = -\frac{1}{y_{z_t}^t} \frac{\partial y_{z_t}^t}{\partial h_k^t}$$

2. **Back-prop through time:** Second, iterate backward through time to calculate the **total derivative**

$$\frac{d\mathcal{L}}{dh_k^t} = -\frac{1}{y_{z_t}^t} \frac{\partial y_{z_t}^t}{\partial h_k^t} + \sum_i \frac{d\mathcal{L}}{dh_i^{t+1}} \frac{\partial h_i^{t+1}}{\partial h_k^t}$$

Time Alignment

- In the previous slides, notice we've assumed that the correct labeling, $\mathbf{z}$, is time-aligned to the speech waveform, i.e., $\mathbf{z} = [z_1, \ldots, z_T]$.
- That's rarely true! Usually we know the correct phones or characters, $\mathbf{z} = [z_1, \ldots, z_U]$, but not their time alignment, i.e., $U \leq T$.
- The old solution (pre-CTC):
  - Train a mixture Gaussian HMM.
  - Use the Viterbi algorithm to time-align $\mathbf{z}$ to $\mathbf{x}$.
  - Use the time-aligned $\mathbf{z}$ to train the RNN.
- CTC was proposed as a better way.

Outline

Many-to-One Mapping

The key idea of CTC is that, since $U \leq T$, the mapping from $\mathbf{y}$ to $\mathbf{z}$ is many-to-one. For example, consider an utterance with a 5-frame speech file, and a 3-character output. We can map from 5 frames to 3 characters by just eliminating sequential duplicates, like this:

$$\mathbf{y} = [\text{f}, \text{f}, \text{e}, \text{d}, \text{d}] \longrightarrow \boxed{\mathcal{B}} \longrightarrow \mathbf{z} = [\text{f}, \text{e}, \text{d}]$$

$$\mathbf{y} = [\text{f}, \text{e}, \text{e}, \text{e}, \text{d}] \longrightarrow \boxed{\mathcal{B}} \longrightarrow \mathbf{z} = [\text{f}, \text{e}, \text{d}]$$

$$\mathbf{y} = [\text{f}, \text{f}, \text{e}, \text{e}, \text{d}] \longrightarrow \boxed{\mathcal{B}} \longrightarrow \mathbf{z} = [\text{f}, \text{e}, \text{d}]$$

But notice the problem: there is no way to generate the output $\mathbf{z} = [\text{f}, \text{e}, \text{e}, \text{d}]$! By eliminating duplicates, it becomes impossible to generate a sentence with repeated letters.

The Blank Character

CTC makes repeated letters possible by using a blank character, –. The many-to-one mapping now has two steps: (1) eliminate all duplicate characters, (2) THEN eliminate all blanks.

$$\mathbf{y} = [\mathrm{f}, -, \mathrm{e}, \mathrm{e}, \mathrm{d}] \longrightarrow \boxed{\mathcal{B}} \longrightarrow \mathbf{z} = [\mathrm{f}, \mathrm{e}, \mathrm{d}]$$

$$\mathbf{y} = [\mathrm{f}, \mathrm{e}, -, \mathrm{e}, \mathrm{d}] \longrightarrow \boxed{\mathcal{B}} \longrightarrow \mathbf{z} = [\mathrm{f}, \mathrm{e}, \mathrm{e}, \mathrm{d}]$$

$$\mathbf{y} = [\mathrm{f}, -, \mathrm{f}, \mathrm{e}, \mathrm{d}] \longrightarrow \boxed{\mathcal{B}} \longrightarrow \mathbf{z} = [\mathrm{f}, \mathrm{f}, \mathrm{e}, \mathrm{d}]$$

Probability of Labels Given Speech

With these definitions, the probability of $\mathbf{z}$ given $\mathbf{x}$ is:

$$p(\mathbf{z}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{z})} p(\pi|\mathbf{x}),$$

- $\pi = [\pi_1, \ldots, \pi_T]$ is a time-aligned label sequence called a "path." Each path element is a label or a blank: $\pi_t \in L \cup \{-\}$.

$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^{t}$$

- $\mathcal{B}^{-1}(\mathbf{z})$ is the set of all paths that match the label sequence $\mathbf{z}$. For example,

$$\mathcal{B}^{-1}\left([\mathrm{f}, \mathrm{e}, \mathrm{d}]\right) = \left\{ \begin{array}{c} [\mathrm{f}, \mathrm{e}, \mathrm{e}, \mathrm{e}, \mathrm{d}] \\ [\mathrm{f}, -, \mathrm{e}, -, \mathrm{d}] \\ [\mathrm{f}, \mathrm{f}, -, \mathrm{e}, \mathrm{d}] \\ \vdots \end{array} \right\}$$

Temporal Classification

The temporal classification problem is now just:

$$
\begin{aligned}
h(\mathbf{x}) &= \underset{\mathbf{l} \in L^{\leq T}}{\operatorname{argmax}}\, p(\mathbf{l}|\mathbf{x}) \\
&= \underset{\mathbf{l} \in L^{\leq T}}{\operatorname{argmax}} \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x}) \\
&= \underset{\mathbf{l} \in L^{\leq T}}{\operatorname{argmax}} \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} \prod_{t=1}^{T} y_{\pi_t}^{t}
\end{aligned}
$$

- $\mathbf{l} = [l_1, \ldots, l_V]$ is a label sequence of any length $V \leq T$ where $l_v \in L$.
- $\pi = [\pi_1, \ldots, \pi_T]$ is a path of length T where $\pi_t \in L \cup \{-\}$.

Outline

Finding the Best Label Sequence

- The problem now is: how can we search the entire set $\pi \in \mathcal{B}^{-1}(\mathbf{l})$, for every possible label sequence?
- Answer: the forward algorithm!

CTC Forward Algorithm: The Modified Label Sequence

In order to express the CTC forward algorithm, we need to define a modified label sequence, $\mathbf{l}'$. $\mathbf{l}'$ is equal to $\mathbf{l}$ with blanks inserted between every pair of letters. Thus if

$$\mathbf{l} = [\mathrm{f}, \mathrm{e}, \mathrm{d}],$$

then

$$\mathbf{l}' = [-, \mathrm{f}, -, \mathrm{e}, -, \mathrm{d}, -].$$

If the length of $\mathbf{l}$ is $|\mathbf{l}|$, then the length of $\mathbf{l}'$ is $2|\mathbf{l}| + 1$.

CTC Forward Algorithm: Partial Sequences

We also need to define the following partial sequences:

$$
\begin{aligned}
\mathbf{x}_{1:t} &= [x_1, \ldots, x_t] \\
\pi_{1:t} &= [\pi_1, \ldots, \pi_t] \\
\mathbf{l'}_{1:s} &= [l'_1, \ldots, l'_s] \\
&= \begin{cases} [-, l_1, -, l_2, \ldots, l_{s/2}] & s \text{ even} \\ [-, l_1, -, l_2, \ldots, l_{(s-1)/2}, -] & s \text{ odd} \end{cases}
\end{aligned}
$$

The CTC Forward Algorithm

Definition: $\alpha_t(\mathbf{l'}_{1:s}) \equiv p(\mathbf{l'}_{1:s}|\mathbf{x}_{1:t})$. Computation:

1. **Initialize:**

$$\alpha_1([-]) = y_-^1$$
$$\alpha_1([-, l_1]) = y_{l_1}^1$$

The neural network can either start out by generating a blank, in which case $\mathbf{l'} = [-]$, or it can start out by generating a real character l_1, in which case $\mathbf{l'} = [-, l_1]$.

The CTC Forward Algorithm

1. **Initialize:**

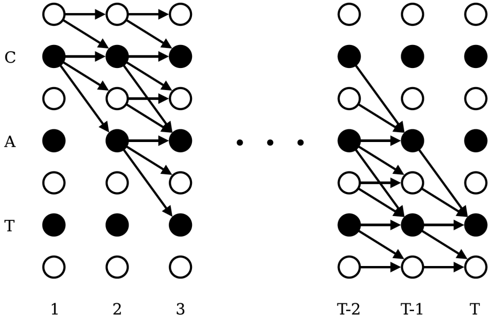
$$\alpha_1([-]) = y_-^1$$
$$\alpha_1([-, l_1]) = y_{l_1}^1$$

2. **Iterate:**

$$\alpha_t(\mathbf{l'}_{1:s}) = \begin{cases} (\alpha_{t-1}(\mathbf{l'}_{1:s}) + \alpha_{t-1}(\mathbf{l'}_{1:s-1})) \times y^t_{l'_s} \\ \ldots\ldots\ldots\ldots \text{if } l'_s = - \text{ or } l'_s = l'_{s-2} \\ (\alpha_{t-1}(\mathbf{l'}_{1:s}) + \alpha_{t-1}(\mathbf{l'}_{1:s-1}) + \alpha_{t-1}(\mathbf{l'}_{1:s-2})) \times y^t_{l'_s} \\ \ldots\ldots\ldots\ldots \text{otherwise} \end{cases}$$

Repeating the same character ($\alpha_{t-1}(\mathbf{l'}_{1:s})$) or adding one more character ($\alpha_{t-1}(\mathbf{l'}_{1:s-1})$) are always possible. Adding two more characters ($\alpha_{t-1}(\mathbf{l'}_{1:s-2})$) is OK if the current character is not a blank or a repeat.

The CTC Forward Algorithm



Graves et al., 2006, Fig. 3. (c) ICML

Repeating the same character ($\alpha_{t-1}(\mathbf{l}'_{1:s})$) or adding one more character ($\alpha_{t-1}(\mathbf{l}'_{1:s-1})$) are always possible. Adding two more characters ($\alpha_{t-1}(\mathbf{l}'_{1:s-2})$) is OK if the current character is not a blank or a repeat.

The CTC Forward Algorithm

1. **Initialize:**

$$\alpha_1([-]) = y_-^1$$
$$\alpha_1([-, l_1]) = y_{l_1}^1$$

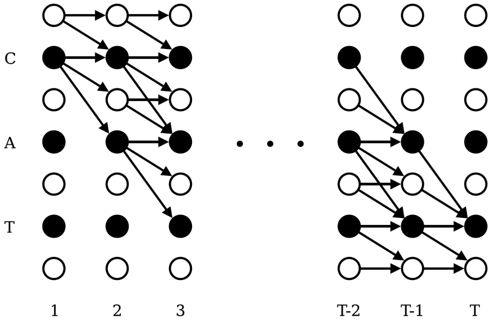
2. **Iterate:**

$$\alpha_t(\mathbf{l'}_{1:s}) = \begin{cases} (\alpha_{t-1}(\mathbf{l'}_{1:s}) + \alpha_{t-1}(\mathbf{l'}_{1:s-1})) \times y_{l'_s}^t \\ \ldots\ldots\ldots\ldots \text{if } l'_s = - \text{ or } l'_s = l'_{s-2} \\ (\alpha_{t-1}(\mathbf{l'}_{1:s}) + \alpha_{t-1}(\mathbf{l'}_{1:s-1}) + \alpha_{t-1}(\mathbf{l'}_{1:s-2})) \times y_{l'_s}^t \\ \ldots\ldots\ldots\ldots \text{otherwise} \end{cases}$$

3. **Terminate:**

$$p(\mathbf{l}_{1:U}|\mathbf{x}) = \alpha_T(\mathbf{l'}_{1:2U}) + \alpha_T(\mathbf{l'}_{1:2U+1})$$

The CTC Forward Algorithm



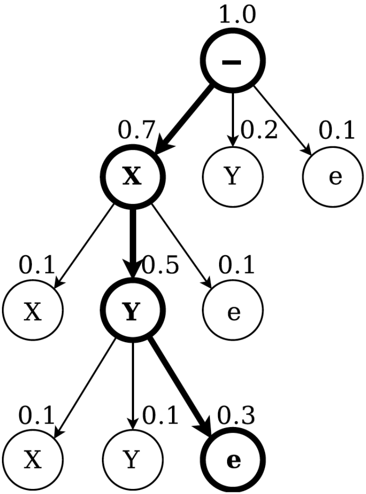
$$p(\mathbf{l}_{1:U}|\mathbf{x}_{1:T}) = \alpha_T(\mathbf{l}'_{1:2U}) + \alpha_T(\mathbf{l}'_{1:2U+1})$$

Computational Issues

- In the HMM, $\alpha_t(q)$ only depended on the final state. In CTC, $\alpha_t(\mathbf{l}'_{1:s})$ depends on the entire label sequence up to position s. The complexity of this search is exponential in s. To make it computationally tractable, use a beam search:
  - Discard all but the best N candidates for each t.
  - Compute all possible extensions to time $t+1$.
  - Repeat
- As in the HMM, $\alpha_t(\mathbf{l}_{1:s})$ becomes very small, so Graves et al. recommend using a scaled forward algorithm.

Computational Issues #1: Beam Search

- $\alpha_t(\mathbf{l}'_{1:s})$ depends on the entire label sequence up to position s. The complexity of this search is exponential in s (shown).
- To make it computationally tractable, use a beam search (not shown).



Graves et al., 2006, Fig. 2. (c) ICML

Outline

Connectionist Temporal Classification

- An RNN computes the probability of a label sequence, $\mathbf{z}$, given an input sequence $\mathbf{x}$.
- The key idea of CTC is a many-to-one mapping from paths to label sequences. The recognition probability is then

$$p(\mathbf{z}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{z})} p(\pi|\mathbf{x}),$$

- The CTC forward algorithm is just like the HMM forward algorithm, except that, instead of $\alpha_t(q)$, we compute

$$\alpha_t(\mathbf{l'}_{1:s}) \equiv p(\mathbf{l'}_{1:s}|\mathbf{x}_{1:t})$$