Deep Learning

Sequence to Sequence models: Connectionist Temporal Classification

5 March 2018
Sequence-to-sequence modelling

• Problem:
  – A sequence $X_1 \ldots X_N$ goes in
  – A different sequence $Y_1 \ldots Y_M$ comes out

• E.g.
  – Speech recognition: Speech goes in, a word sequence comes out
    • Alternately output may be phoneme or character sequence
  – Machine translation: Word sequence goes in, word sequence comes out

• In general $N \neq M$
  – No synchrony between $X$ and $Y$. 
Sequence to sequence

- Sequence goes in, sequence comes out
- No notion of “synchrony” between input and output
  - May even not have a notion of “alignment”
    - E.g. “I ate an apple” → “Ich habe einen apfel gegessen”
Case 1: With alignment

- The input and output sequences happen in the same order
  - Although they may be asynchronous
  - E.g. Speech recognition
    - The input speech corresponds to the phoneme sequence output
Variants on recurrent nets

1: Conventional MLP
2: Sequence generation, e.g. image to caption
3: Sequence based prediction or classification, e.g. Speech recognition, text classification
Basic model

- Sequence of inputs produces a single output
• The Divergence is only defined at the final input
  \[ DIV(Y_{\text{target}}, Y) = Xent(Y(T), \text{Phoneme}) \]
• This divergence must propagate through the net to update all parameters
• Ignores outputs at intermediate steps
Fix: Use these outputs too.

These too must ideally point to the correct phoneme

• Exploiting the untagged inputs: assume the same output for the entire input

• Define the divergence everywhere

\[
DIV(Y_{\text{target}}, Y) = \sum_t w_t X_{\text{ent}}(Y(t), \text{Phoneme})
\]
Training

- Define the divergence everywhere

$$DIV(Y_{target}, Y) = \sum_t w_t X_{ent}(Y(t), \text{Phoneme})$$

- Typical weighting scheme for speech: all are equally important
- Problem like question answering: answer only expected after the question ends
  - Only $w_T$ is high, other weights are 0 or low
• Define the divergence everywhere

\[ DIV(Y_{target}, Y) = \sum_t w_t Xent(Y(t), \text{Phoneme}) \]

• Typical weighting scheme for speech: all are equally important

• Problem like question answering: answer only expected after the question ends
  – Only \( w_T \) is high, other weights are 0 or low

We will initially focus on the class of problem where uniform weights are reasonable (e.g. speech recognition)
The more complex problem

• Objective: Given a sequence of inputs, asynchronously output a sequence of symbols
  – This is just a simple concatenation of many copies of the simple “output at the end of the input sequence” model we just saw

• But this simple extension complicates matters.
The sequence-to-sequence problem

• How do we know when to output symbols
  – In fact, the network produces outputs at every time
  – Which of these are the real outputs?
The actual output of the network

<table>
<thead>
<tr>
<th></th>
<th>/AH/</th>
<th>/B/</th>
<th>/D/</th>
<th>/EH/</th>
<th>/IH/</th>
<th>/F/</th>
<th>/G/</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$y_0^{AH}$</td>
<td>$y_0^B$</td>
<td>$y_0^D$</td>
<td>$y_0^{EH}$</td>
<td>$y_0^{IH}$</td>
<td>$y_0^F$</td>
<td>$y_0^G$</td>
</tr>
<tr>
<td></td>
<td>$y_1^{AH}$</td>
<td>$y_1^B$</td>
<td>$y_1^D$</td>
<td>$y_1^{EH}$</td>
<td>$y_1^{IH}$</td>
<td>$y_1^F$</td>
<td>$y_1^G$</td>
</tr>
<tr>
<td></td>
<td>$y_2^{AH}$</td>
<td>$y_2^B$</td>
<td>$y_2^D$</td>
<td>$y_2^{EH}$</td>
<td>$y_2^{IH}$</td>
<td>$y_2^F$</td>
<td>$y_2^G$</td>
</tr>
<tr>
<td></td>
<td>$y_3^{AH}$</td>
<td>$y_3^B$</td>
<td>$y_3^D$</td>
<td>$y_3^{EH}$</td>
<td>$y_3^{IH}$</td>
<td>$y_3^F$</td>
<td>$y_3^G$</td>
</tr>
<tr>
<td></td>
<td>$y_4^{AH}$</td>
<td>$y_4^B$</td>
<td>$y_4^D$</td>
<td>$y_4^{EH}$</td>
<td>$y_4^{IH}$</td>
<td>$y_4^F$</td>
<td>$y_4^G$</td>
</tr>
<tr>
<td></td>
<td>$y_5^{AH}$</td>
<td>$y_5^B$</td>
<td>$y_5^D$</td>
<td>$y_5^{EH}$</td>
<td>$y_5^{IH}$</td>
<td>$y_5^F$</td>
<td>$y_5^G$</td>
</tr>
<tr>
<td></td>
<td>$y_6^{AH}$</td>
<td>$y_6^B$</td>
<td>$y_6^D$</td>
<td>$y_6^{EH}$</td>
<td>$y_6^{IH}$</td>
<td>$y_6^F$</td>
<td>$y_6^G$</td>
</tr>
<tr>
<td></td>
<td>$y_7^{AH}$</td>
<td>$y_7^B$</td>
<td>$y_7^D$</td>
<td>$y_7^{EH}$</td>
<td>$y_7^{IH}$</td>
<td>$y_7^F$</td>
<td>$y_7^G$</td>
</tr>
<tr>
<td></td>
<td>$y_8^{AH}$</td>
<td>$y_8^B$</td>
<td>$y_8^D$</td>
<td>$y_8^{EH}$</td>
<td>$y_8^{IH}$</td>
<td>$y_8^F$</td>
<td>$y_8^G$</td>
</tr>
</tbody>
</table>

- At each time the network outputs a probability for each output symbol given all inputs until that time
  - E.g. $y_4^D = \text{prob}(s_4 = D | X_0 \ldots X_4)$
Overall objective

- Find most likely symbol sequence given inputs

\[ S_0 \ldots S_{K-1} = \arg\max \ \text{prob}(S'_0 \ldots S'_{K-1} | X_0 \ldots X_{N-1}) \]

\[ S'_0 \ldots S'_{K-1} \]
Finding the best output

<table>
<thead>
<tr>
<th>/AH/</th>
<th>$y_0^{AH}$</th>
<th>$y_1^{AH}$</th>
<th>$y_2^{AH}$</th>
<th>$y_3^{AH}$</th>
<th>$y_4^{AH}$</th>
<th>$y_5^{AH}$</th>
<th>$y_6^{AH}$</th>
<th>$y_7^{AH}$</th>
<th>$y_8^{AH}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/B/</td>
<td>$y_0^B$</td>
<td>$y_1^B$</td>
<td>$y_2^B$</td>
<td>$y_3^B$</td>
<td>$y_4^B$</td>
<td>$y_5^B$</td>
<td>$y_6^B$</td>
<td>$y_7^B$</td>
<td>$y_8^B$</td>
</tr>
<tr>
<td>/D/</td>
<td>$y_0^D$</td>
<td>$y_1^D$</td>
<td>$y_2^D$</td>
<td>$y_3^D$</td>
<td>$y_4^D$</td>
<td>$y_5^D$</td>
<td>$y_6^D$</td>
<td>$y_7^D$</td>
<td>$y_8^D$</td>
</tr>
<tr>
<td>/EH/</td>
<td>$y_0^{EH}$</td>
<td>$y_1^{EH}$</td>
<td>$y_2^{EH}$</td>
<td>$y_3^{EH}$</td>
<td>$y_4^{EH}$</td>
<td>$y_5^{EH}$</td>
<td>$y_6^{EH}$</td>
<td>$y_7^{EH}$</td>
<td>$y_8^{EH}$</td>
</tr>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/F/</td>
<td>$y_0^F$</td>
<td>$y_1^F$</td>
<td>$y_2^F$</td>
<td>$y_3^F$</td>
<td>$y_4^F$</td>
<td>$y_5^F$</td>
<td>$y_6^F$</td>
<td>$y_7^F$</td>
<td>$y_8^F$</td>
</tr>
<tr>
<td>/G/</td>
<td>$y_0^G$</td>
<td>$y_1^G$</td>
<td>$y_2^G$</td>
<td>$y_3^G$</td>
<td>$y_4^G$</td>
<td>$y_5^G$</td>
<td>$y_6^G$</td>
<td>$y_7^G$</td>
<td>$y_8^G$</td>
</tr>
</tbody>
</table>

- Option 1: Simply select the most probable symbol at each time
Finding the best output

<table>
<thead>
<tr>
<th>/AH/</th>
<th>$y_0^{AH}$</th>
<th>$y_1^{AH}$</th>
<th>$y_2^{AH}$</th>
<th>$y_3^{AH}$</th>
<th>$y_4^{AH}$</th>
<th>$y_5^{AH}$</th>
<th>$y_6^{AH}$</th>
<th>$y_7^{AH}$</th>
<th>$y_8^{AH}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/B/</td>
<td>$y_0^B$</td>
<td>$y_1^B$</td>
<td>$y_2^B$</td>
<td>$y_3^B$</td>
<td>$y_4^B$</td>
<td>$y_5^B$</td>
<td>$y_6^B$</td>
<td>$y_7^B$</td>
<td>$y_8^B$</td>
</tr>
<tr>
<td>/D/</td>
<td>$y_0^D$</td>
<td>$y_1^D$</td>
<td>$y_2^D$</td>
<td>$y_3^D$</td>
<td>$y_4^D$</td>
<td>$y_5^D$</td>
<td>$y_6^D$</td>
<td>$y_7^D$</td>
<td>$y_8^D$</td>
</tr>
<tr>
<td>/EH/</td>
<td>$y_0^{EH}$</td>
<td>$y_1^{EH}$</td>
<td>$y_2^{EH}$</td>
<td>$y_3^{EH}$</td>
<td>$y_4^{EH}$</td>
<td>$y_5^{EH}$</td>
<td>$y_6^{EH}$</td>
<td>$y_7^{EH}$</td>
<td>$y_8^{EH}$</td>
</tr>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/F/</td>
<td>$y_0^F$</td>
<td>$y_1^F$</td>
<td>$y_2^F$</td>
<td>$y_3^F$</td>
<td>$y_4^F$</td>
<td>$y_5^F$</td>
<td>$y_6^F$</td>
<td>$y_7^F$</td>
<td>$y_8^F$</td>
</tr>
<tr>
<td>/G/</td>
<td>$y_0^G$</td>
<td>/G/</td>
<td>$y_2^G$</td>
<td>$y_3^G$</td>
<td>$y_4^G$</td>
<td>$y_5^G$</td>
<td>$y_6^G$</td>
<td>$y_7^G$</td>
<td>$y_8^G$</td>
</tr>
</tbody>
</table>

- Option 1: Simply select the most probable symbol at each time
  - Merge adjacent repeated symbols, and place the actual emission of the symbol in the final instant
### The actual output of the network

<table>
<thead>
<tr>
<th>/AH/</th>
<th>$y_0^{AH}$</th>
<th>$y_1^{AH}$</th>
<th>$y_2^{AH}$</th>
<th>$y_3^{AH}$</th>
<th>$y_4^{AH}$</th>
<th>$y_5^{AH}$</th>
<th>$y_6^{AH}$</th>
<th>$y_7^{AH}$</th>
<th>$y_8^{AH}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/B/</td>
<td>$y_0^B$</td>
<td>$y_1^B$</td>
<td>$y_2^B$</td>
<td>$y_3^B$</td>
<td>$y_4^B$</td>
<td>$y_5^B$</td>
<td>$y_6^B$</td>
<td>$y_7^B$</td>
<td>$y_8^B$</td>
</tr>
<tr>
<td>/D/</td>
<td>$y_0^D$</td>
<td>$y_1^D$</td>
<td>$y_2^D$</td>
<td>$y_3^D$</td>
<td>$y_4^D$</td>
<td>$y_5^D$</td>
<td>$y_6^D$</td>
<td>$y_7^D$</td>
<td>$y_8^D$</td>
</tr>
<tr>
<td>/EH/</td>
<td>$y_0^{EH}$</td>
<td>$y_1^{EH}$</td>
<td>$y_2^{EH}$</td>
<td>$y_3^{EH}$</td>
<td>$y_4^{EH}$</td>
<td>$y_5^{EH}$</td>
<td>$y_6^{EH}$</td>
<td>$y_7^{EH}$</td>
<td>$y_8^{EH}$</td>
</tr>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
</tbody>
</table>

**Option 1:** Simply select the most probable symbol at each time

- *Merge* adjacent repeated symbols, and place the actual emission of the symbol in the final instant

---

**Cannot distinguish between an extended symbol and repetitions of the symbol**
The actual output of the network

Option 1: Simply select the most probable symbol at each time

- Merge adjacent repeated symbols, and place the actual emission of the symbol in the final instant

Resulting sequence may be meaningless (what word is “GFIYD”?)

Cannot distinguish between an extended symbol and repetitions of the symbol
The actual output of the network

- Option 2: Impose external constraints on what sequences are allowed
  - E.g. only allow sequences corresponding to dictionary words
  - E.g. *Sub-symbol* units (like in HW1 – what were they?)
The actual output of the network

We will refer to the process of obtaining an output from the network as decoding

• Option 2: Impose external constraints on what sequences are allowed
  – E.g. only allow sequences corresponding to dictionary words
  – E.g. Sub-symbol units (like in HW1 – what were they?)
The sequence-to-sequence problem

- How do we know when to output symbols
  - In fact, the network produces outputs at every time
  - Which of these are the real outputs

- How do we train these models?
Training

- Given output symbols at the right locations
  - The phoneme /B/ ends at $X_2$, /IY/ at $X_4$, /F/ at $X_6$, /IY/ at $X_9$
Either just define Divergence as:

\[
DIV = Xent(Y_2, B) + Xent(Y_4, IY) + Xent(Y_6, F) + Xent(Y_9, IY)
\]

Or..
Either just define Divergence as:

\[ DIV = \text{Xent}(Y_2, B) + \text{Xent}(Y_4, IY) + \text{Xent}(Y_6, F) + \text{Xent}(Y_9, IY) \]

Or repeat the symbols over their duration

\[ DIV = \sum_t \text{Xent}(Y_t, \text{symbol}_t) = - \sum_t \log Y(t, \text{symbol}_t) \]
DIV = \sum_t Xent(Y_t, symbol_t) = - \sum_t \log Y(t, symbol_t)

- The gradient w.r.t the t-th output vector Y_t

\( \nabla_{Y_t} DIV = \begin{bmatrix} 0 & 0 & \ldots & \frac{-1}{Y(t, symbol_t)} & 0 & \ldots & 0 \end{bmatrix} \)

- Zeros except at the component corresponding to the target
Problem: No timing information provided

- Only the sequence of output symbols is provided for the training data
  - But no indication of which one occurs where
- How do we compute the divergence?
  - And how do we compute its gradient w.r.t. $Y_t$
Solution 1: *Guess the alignment*

- **Initialize:** Assign an initial alignment
  - Either randomly, based on some heuristic, or any other rationale
- **Iterate:**
  - Train the network using the current alignment
  - *Reestimate* the alignment for each training instance
    - Using the decoding methods already discussed
Solution 1: *Guess the alignment*

- **Initialize:** Assign an initial alignment
  - Either randomly, based on some heuristic, or any other rationale

- **Iterate:**
  - Train the network using the current alignment
  - *Reestimate* the alignment for each training instance
    - Using the decoding methods already discussed
Estimating an alignment

- Given:
  - The unaligned $K$-length symbol sequence $S = S_0 \ldots S_{K-1}$ (e.g. /B/ /IY/ /F/ /IY/)
  - An $N$-length input ($N \geq K$)
  - And a (trained) recurrent network

- Find:
  - An $N$-length expansion $s_0 \ldots s_{N-1}$ comprising the symbols in $S$ in strict order
    - e.g. $S_0S_1S_1S_2S_3S_3 \ldots S_{K-1}$
      - i.e. $s_0 = S_0, s_2 = S_1, s_3 = S_1, s_4 = S_2, s_5 = S_3, \ldots s_{N-1} = S_{K-1}$
      - E.g. /B/ /B/ /IY/ /IY/ /IY/ /F/ /F/ /F/ /F/ /IY/ ..
    - $s_i = S_k \Rightarrow i \leq k$
    - $s_i = S_k, s_j = S_l, \ i < j \Rightarrow k \leq l$

- Outcome: an alignment of the target symbol sequence $S_0 \ldots S_{K-1}$ to the input $X_0 \ldots X_{N-1}$
Recall: The actual output of the network

- At each time the network outputs a probability for each output symbol
Recall: unconstrained decoding

- We find the most likely sequence of symbols
  - (Conditioned on input $X_0 \ldots X_{N-1}$)
- This may not correspond to an expansion of the desired symbol sequence
  - E.g. the unconstrained decode may be
    - Contracts to `/AH/ /D/ /AH/ /F/ /IY/`
  - Whereas we want an expansion of `/B//IY//F//IY/`
Constraining the alignment: Try 1

<table>
<thead>
<tr>
<th>/B/</th>
<th>$y_0^B$</th>
<th>$y_1^B$</th>
<th>$y_2^B$</th>
<th>$y_3^B$</th>
<th>$y_4^B$</th>
<th>$y_5^B$</th>
<th>$y_6^B$</th>
<th>$y_7^B$</th>
<th>$y_8^B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/F/</td>
<td>$y_0^F$</td>
<td>$y_1^F$</td>
<td>$y_2^F$</td>
<td>$y_3^F$</td>
<td>$y_4^F$</td>
<td>$y_5^F$</td>
<td>$y_6^F$</td>
<td>$y_7^F$</td>
<td>$y_8^F$</td>
</tr>
</tbody>
</table>

- Block out all rows that do not include symbols from the target sequence
  - E.g. Block out rows that are not /B/ /IY/ or /F/
**Blocking out unnecessary outputs**

<table>
<thead>
<tr>
<th>/B/</th>
<th>$y_0^B$</th>
<th>$y_1^B$</th>
<th>$y_2^B$</th>
<th>$y_3^B$</th>
<th>$y_4^B$</th>
<th>$y_5^B$</th>
<th>$y_6^B$</th>
<th>$y_7^B$</th>
<th>$y_8^B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/F/</td>
<td>$y_0^F$</td>
<td>$y_1^F$</td>
<td>$y_2^F$</td>
<td>$y_3^F$</td>
<td>$y_4^F$</td>
<td>$y_5^F$</td>
<td>$y_6^F$</td>
<td>$y_7^F$</td>
<td>$y_8^F$</td>
</tr>
<tr>
<td>/AH/</td>
<td>$y_0^{AH}$</td>
<td>$y_1^{AH}$</td>
<td>$y_2^{AH}$</td>
<td>$y_3^{AH}$</td>
<td>$y_4^{AH}$</td>
<td>$y_5^{AH}$</td>
<td>$y_6^{AH}$</td>
<td>$y_7^{AH}$</td>
<td>$y_8^{AH}$</td>
</tr>
<tr>
<td>/B/</td>
<td>$y_0^B$</td>
<td>$y_1^B$</td>
<td>$y_2^B$</td>
<td>$y_3^B$</td>
<td>$y_4^B$</td>
<td>$y_5^B$</td>
<td>$y_6^B$</td>
<td>$y_7^B$</td>
<td>$y_8^B$</td>
</tr>
<tr>
<td>/D/</td>
<td>$y_0^D$</td>
<td>$y_1^D$</td>
<td>$y_2^D$</td>
<td>$y_3^D$</td>
<td>$y_4^D$</td>
<td>$y_5^D$</td>
<td>$y_6^D$</td>
<td>$y_7^D$</td>
<td>$y_8^D$</td>
</tr>
<tr>
<td>/EH/</td>
<td>$y_0^{EH}$</td>
<td>$y_1^{EH}$</td>
<td>$y_2^{EH}$</td>
<td>$y_3^{EH}$</td>
<td>$y_4^{EH}$</td>
<td>$y_5^{EH}$</td>
<td>$y_6^{EH}$</td>
<td>$y_7^{EH}$</td>
<td>$y_8^{EH}$</td>
</tr>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/F/</td>
<td>$y_0^F$</td>
<td>$y_1^F$</td>
<td>$y_2^F$</td>
<td>$y_3^F$</td>
<td>$y_4^F$</td>
<td>$y_5^F$</td>
<td>$y_6^F$</td>
<td>$y_7^F$</td>
<td>$y_8^F$</td>
</tr>
<tr>
<td>/G/</td>
<td>$y_0^G$</td>
<td>$y_1^G$</td>
<td>$y_2^G$</td>
<td>$y_3^G$</td>
<td>$y_4^G$</td>
<td>$y_5^G$</td>
<td>$y_6^G$</td>
<td>$y_7^G$</td>
<td>$y_8^G$</td>
</tr>
</tbody>
</table>

**Diagram:**

1. Compute the entire output (for all symbols)
2. Copy the output values for the target symbols into the secondary reduced structure
Constraining the alignment: Try 1

<table>
<thead>
<tr>
<th>/B/</th>
<th>( y_0^B )</th>
<th>( y_1^B )</th>
<th>( y_2^B )</th>
<th>( y_3^B )</th>
<th>( y_4^B )</th>
<th>( y_5^B )</th>
<th>( y_6^B )</th>
<th>( y_7^B )</th>
<th>( y_8^B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>/IY/</td>
<td>( y_0^{IY} )</td>
<td>( y_1^{IY} )</td>
<td>( y_2^{IY} )</td>
<td>( y_3^{IY} )</td>
<td>( y_4^{IY} )</td>
<td>( y_5^{IY} )</td>
<td>( y_6^{IY} )</td>
<td>( y_7^{IY} )</td>
<td>( y_8^{IY} )</td>
</tr>
<tr>
<td>/F/</td>
<td>( y_0^F )</td>
<td>( y_1^F )</td>
<td>( y_2^F )</td>
<td>( y_3^F )</td>
<td>( y_4^F )</td>
<td>( y_5^F )</td>
<td>( y_6^F )</td>
<td>( y_7^F )</td>
<td>( y_8^F )</td>
</tr>
</tbody>
</table>

- Only decode on reduced grid
  - We are now assured that only the appropriate symbols will be hypothesized
Constraining the alignment: Try 1

<table>
<thead>
<tr>
<th>/B/</th>
<th>/IY/</th>
<th>/F/</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y^B_0$</td>
<td>$y^IY_0$</td>
<td>$y^F_0$</td>
</tr>
<tr>
<td>$y^B_1$</td>
<td>$y^IY_1$</td>
<td>$y^F_1$</td>
</tr>
<tr>
<td>$y^B_2$</td>
<td>$y^IY_2$</td>
<td>$y^F_2$</td>
</tr>
<tr>
<td>$y^B_3$</td>
<td>$y^IY_3$</td>
<td>$y^F_3$</td>
</tr>
<tr>
<td>$y^B_4$</td>
<td>$y^IY_4$</td>
<td>$y^F_4$</td>
</tr>
<tr>
<td>$y^B_5$</td>
<td>$y^IY_5$</td>
<td>$y^F_5$</td>
</tr>
<tr>
<td>$y^B_6$</td>
<td>$y^IY_6$</td>
<td>$y^F_6$</td>
</tr>
<tr>
<td>$y^B_7$</td>
<td>$y^IY_7$</td>
<td>$y^F_7$</td>
</tr>
<tr>
<td>$y^B_8$</td>
<td>$y^IY_8$</td>
<td>$y^F_8$</td>
</tr>
</tbody>
</table>

- Only decode on reduced grid
  - We are now assured that only the appropriate symbols will be hypothesized

- Problem: This still doesn’t assure that the decode sequence correctly expands the target symbol sequence
  - E.g. the above decode is not an expansion of /B//IY//F//IY/

- Still needs additional constraints
Try 2: Explicitly arrange the constructed table

<table>
<thead>
<tr>
<th>/B/</th>
<th>$y_0^B$</th>
<th>$y_1^B$</th>
<th>$y_2^B$</th>
<th>$y_3^B$</th>
<th>$y_4^B$</th>
<th>$y_5^B$</th>
<th>$y_6^B$</th>
<th>$y_7^B$</th>
<th>$y_8^B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/F/</td>
<td>$y_0^F$</td>
<td>$y_1^F$</td>
<td>$y_2^F$</td>
<td>$y_3^F$</td>
<td>$y_4^F$</td>
<td>$y_5^F$</td>
<td>$y_6^F$</td>
<td>$y_7^F$</td>
<td>$y_8^F$</td>
</tr>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/AH/</td>
<td>$y_0^{AH}$</td>
<td>$y_1^{AH}$</td>
<td>$y_2^{AH}$</td>
<td>$y_3^{AH}$</td>
<td>$y_4^{AH}$</td>
<td>$y_5^{AH}$</td>
<td>$y_6^{AH}$</td>
<td>$y_7^{AH}$</td>
<td>$y_8^{AH}$</td>
</tr>
<tr>
<td>/B/</td>
<td>$y_0^B$</td>
<td>$y_1^B$</td>
<td>$y_2^B$</td>
<td>$y_3^B$</td>
<td>$y_4^B$</td>
<td>$y_5^B$</td>
<td>$y_6^B$</td>
<td>$y_7^B$</td>
<td>$y_8^B$</td>
</tr>
<tr>
<td>/D/</td>
<td>$y_0^D$</td>
<td>$y_1^D$</td>
<td>$y_2^D$</td>
<td>$y_3^D$</td>
<td>$y_4^D$</td>
<td>$y_5^D$</td>
<td>$y_6^D$</td>
<td>$y_7^D$</td>
<td>$y_8^D$</td>
</tr>
<tr>
<td>/EH/</td>
<td>$y_0^{EH}$</td>
<td>$y_1^{EH}$</td>
<td>$y_2^{EH}$</td>
<td>$y_3^{EH}$</td>
<td>$y_4^{EH}$</td>
<td>$y_5^{EH}$</td>
<td>$y_6^{EH}$</td>
<td>$y_7^{EH}$</td>
<td>$y_8^{EH}$</td>
</tr>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/F/</td>
<td>$y_0^F$</td>
<td>$y_1^F$</td>
<td>$y_2^F$</td>
<td>$y_3^F$</td>
<td>$y_4^F$</td>
<td>$y_5^F$</td>
<td>$y_6^F$</td>
<td>$y_7^F$</td>
<td>$y_8^F$</td>
</tr>
<tr>
<td>/G/</td>
<td>$y_0^G$</td>
<td>$y_1^G$</td>
<td>$y_2^G$</td>
<td>$y_3^G$</td>
<td>$y_4^G$</td>
<td>$y_5^G$</td>
<td>$y_6^G$</td>
<td>$y_7^G$</td>
<td>$y_8^G$</td>
</tr>
</tbody>
</table>

Arrange the constructed table so that from top to bottom it has the exact sequence of symbols required.
Try 2: Explicitly arrange the constructed table

<table>
<thead>
<tr>
<th>/B/</th>
<th>(y_0^B)</th>
<th>(y_1^B)</th>
<th>(y_2^B)</th>
<th>(y_3^B)</th>
<th>(y_4^B)</th>
<th>(y_5^B)</th>
<th>(y_6^B)</th>
<th>(y_7^B)</th>
<th>(y_8^B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/IY/</td>
<td>(y_0^{IY})</td>
<td>(y_1^{IY})</td>
<td>(y_2^{IY})</td>
<td>(y_3^{IY})</td>
<td>(y_4^{IY})</td>
<td>(y_5^{IY})</td>
<td>(y_6^{IY})</td>
<td>(y_7^{IY})</td>
<td>(y_8^{IY})</td>
</tr>
<tr>
<td>/F/</td>
<td>(y_0^F)</td>
<td>(y_1^F)</td>
<td>(y_2^F)</td>
<td>(y_3^F)</td>
<td>(y_4^F)</td>
<td>(y_5^F)</td>
<td>(y_6^F)</td>
<td>(y_7^F)</td>
<td>(y_8^F)</td>
</tr>
<tr>
<td>/IY/</td>
<td>(y_0^{IY})</td>
<td>(y_1^{IY})</td>
<td>(y_2^{IY})</td>
<td>(y_3^{IY})</td>
<td>(y_4^{IY})</td>
<td>(y_5^{IY})</td>
<td>(y_6^{IY})</td>
<td>(y_7^{IY})</td>
<td>(y_8^{IY})</td>
</tr>
</tbody>
</table>

Note: If a symbol occurs multiple times, we repeat the row in the appropriate location. E.g. the row for /IY/ occurs twice, in the 2\(^{nd}\) and 4\(^{th}\) positions.

Arrange the constructed table so that from top to bottom it has the exact sequence of symbols required.
Explicitly constrain alignment

- Constrain that the first symbol in the decode *must* be the top left block
- The last symbol *must* be the bottom right
- The rest of the symbols must follow a sequence that *monotonically* travels down from top left to bottom right
  - i.e. never goes up
- This guarantees that the sequence *is* an expansion of the target sequence
  - /B/ /IY/ /F/ /IY/ in this case
Explicitly constrain alignment

- Compose a graph such that every path in the graph from source to sink represents a valid alignment
  - Which maps on to the target symbol sequence (/B//AH//T/)
- Edge scores are 1
- Node scores are the probabilities assigned to the symbols by the neural network
- The “score” of a path is the product of the probabilities of all nodes along the path
- Find the most probable path from source to sink using any dynamic programming algorithm
  - E.g. The Viterbi algorithm
Viterbi algorithm

- At each node, keep track of
  - The best incoming edge
  - The score of the best path from the source to the node
- Dynamically compute the best path from source to sink
Viterbi algorithm

First, some notation:

- $y_t^{S(r)}$ is the probability of the target symbol assigned to the $r$-th row in the $t$-th time (given inputs $X_1 \ldots X_t$)
  - E.g., $S(0) = /B/$
    - The scores in the 0th row have the form $y_t^B$
  - E.g. $S(1) = S(3) = /IY/$
    - The scores in the 1st and 3rd rows have the form $y_t^{IY}$
  - E.g. $S(2) = /F/$
    - The scores in the 2nd row have the form $y_t^F$
Viterbi algorithm

- Initialization:

\[ BP(0, i) = \text{null}, \quad i = 0 \ldots K - 1 \]

\[ Bscr(0,0) = y_0^{S(0)}, \quad Bscr(0, i) = -\infty, \quad i = 1 \ldots K - 1 \]
The Viterbi algorithm can be described as follows:

- **Initialization:**
  
  \[ BP(0, i) = null, \quad i = 0 \ldots K - 1 \]
  
  \[ Bscr(0,0) = y_0^{S(0)}, \quad Bscr(0, i) = -\infty, \quad i = 1 \ldots K - 1 \]

- **for** \( t = 1 \ldots T - 1 \)
  
  \[ BP(t,0) = 0; \quad Bscr(t,0) = Bscr(t - 1,0) \times y_t^{S(0)} \]
  
  for \( l = 0 \ldots K - 1 \)

  - \[ BP(t,l) = (if \ (Bscr(t - 1,l - 1) > Bscr(t - 1,l)) \ l - 1; \ else \ l) \]
  
  - \[ Bscr(t,l) = Bscr(BP(t,l)) \times y_t^{S(l)} \]
Viterbi algorithm

- Initialization:

\[ BP(0, i) = \text{null}, \ i = 0 \ldots K - 1 \]

\[ B\text{scr}(0,0) = y_0^{S(0)}, \ B\text{scr}(0, i) = -\infty, \ i = 1 \ldots K - 1 \]

- for \( t = 1 \ldots T - 1 \)

\[ BP(t, 0) = 0; \ B\text{scr}(t, 0) = B\text{scr}(t - 1,0) \times y_t^{S(0)} \]

for \( l = 1 \ldots K - 1 \)

- \( BP(t, l) = (if \ (B\text{scr}(t - 1, l - 1) > B\text{scr}(t - 1, l)) \ l - 1; \ else \ l) \)

- \( B\text{scr}(t, l) = B\text{scr}(BP(t, l)) \times y_t^{S(l)} \)
Viterbi algorithm

• Initialization:
  \[ BP(0, i) = \text{null}, \ i = 0 \ldots K - 1 \]
  \[ B_{\text{scr}}(0,0) = y_0^{S(0)}, \ B_{\text{scr}}(0, i) = -\infty, \ i = 1 \ldots K - 1 \]

• for \( t = 1 \ldots T - 1 \)
  \[ BP(t, 0) = 0; B_{\text{scr}}(t, 0) = B_{\text{scr}}(t - 1, 0) \times y_t^{S(0)} \]
  for \( l = 1 \ldots K - 1 \)
  • \[ BP(t, l) = (if \ (B_{\text{scr}}(t - 1, l - 1) > B_{\text{scr}}(t - 1, l)) \ l - 1; \ else \ l) \]
  • \[ B_{\text{scr}}(t, l) = B_{\text{scr}}(BP(t, l)) \times y_t^{S(l)} \]
Viterbi algorithm

• Initialization:

\[ BP(0, i) = \text{null}, \ i = 0 \ldots K - 1 \]
\[ Bscr(0,0) = y_0^{S(0)}, \ Bscr(0, i) = -\infty, \ i = 1 \ldots K - 1 \]

• for \( t = 1 \ldots T - 1 \)

\[ BP(t, 0) = 0; \ Bscr(t, 0) = Bscr(t - 1,0) \times y_t^{S(0)} \]
for \( l = 1 \ldots K - 1 \)

• \[ BP(t, l) = \begin{cases} 
Bscr(t - 1, l - 1) > Bscr(t - 1, l) \quad & l - 1; \ \text{else} \ l 
\end{cases} \]

• \[ Bscr(t, l) = Bscr(BP(t, l)) \times y_t^{S(l)} \]
Viterbi algorithm

- Initialization:
  \[ BP(0, i) = \text{null}, \; i = 0 \ldots K - 1 \]
  \[ Bscr(0,0) = y_0^{S(0)}, \; Bscr(0, i) = -\infty, \; i = 1 \ldots K - 1 \]
- for \( t = 1 \ldots T - 1 \)
  \[ BP(t, 0) = 0; \; Bscr(t, 0) = Bscr(t - 1,0) \times y_t^{S(0)} \]
  for \( l = 1 \ldots K - 1 \)
  - \( BP(t, l) = \text{if } (Bscr(t - 1, l - 1) > Bscr(t - 1, l)) \; l - 1; \; \text{else } l \)
  - \( Bscr(t, l) = Bscr(BP(t, l)) \times y_t^{S(l)} \)
Viterbi algorithm

- Initialization:
  \[ BP(0, i) = \text{null}, \ i = 0 \ldots K - 1 \]
  \[ B_{scr}(0,0) = y_0^{S(0)}, \ B_{scr}(0, i) = -\infty, \ i = 1 \ldots K - 1 \]
- for \( t = 1 \ldots T - 1 \)
  \[ BP(t, 0) = 0; \ B_{scr}(t, 0) = B_{scr}(t - 1,0) \times y_t^{S(0)} \]
  for \( l = 1 \ldots K - 1 \)
  - \( BP(t, l) = (\text{if} \ (B_{scr}(t - 1, l - 1) > B_{scr}(t - 1, l)) \ l - 1; \ else \ l) \)
  - \( B_{scr}(t, l) = B_{scr}(BP(t, l)) \times y_t^{S(l)} \)
Viterbi algorithm

• Initialization:

\[ BP(0, i) = \text{null}, \ i = 0 \ldots K - 1 \]

\[ Bscr(0,0) = y^{S(0)}_0, \ Bscr(0, i) = -\infty, \ i = 1 \ldots K - 1 \]

• for \( t = 1 \ldots T - 1 \)

\[ BP(t, 0) = 0; \ Bscr(t, 0) = Bscr(t - 1, 0) \times y^{S(0)}_t \]

for \( l = 1 \ldots K - 1 \)

• \( BP(t, l) = (\text{if } (Bscr(t - 1, l - 1) > Bscr(t - 1, l)) l - 1; \ \text{else } l) \)

• \( Bscr(t, l) = Bscr(BP(t, l)) \times y^{S(l)}_t \)
Viterbi algorithm

• Initialization:

\[ \text{BP}(0, i) = \text{null}, \; i = 0 \ldots K - 1 \]
\[ \text{Bscr}(0,0) = y_0^{S(0)}, \; \text{Bscr}(0, i) = -\infty, \; i = 1 \ldots K - 1 \]

• for \( t = 1 \ldots T - 1 \)

\[ \text{BP}(t, 0) = 0; \; \text{Bscr}(t, 0) = \text{Bscr}(t - 1,0) \times y_t^{S(0)} \]
for \( l = 1 \ldots K - 1 \)

• \( \text{BP}(t, l) = \begin{cases} \text{BP}(t, l-1) & \text{if } \text{Bscr}(t-1, l-1) > \text{Bscr}(t-1, l) \end{cases} \)

• \( \text{Bscr}(t, l) = \text{Bscr}(\text{BP}(t, l)) \times y_t^{S(l)} \)
Viterbi algorithm

• Initialization:
  
  \[ BP(0, i) = \text{null}, \ i = 0 \ldots K - 1 \]
  
  \[ Bscr(0,0) = y_0^{S(0)}, \ Bscr(0, i) = -\infty, \ i = 1 \ldots K - 1 \]
  
• for \( t = 1 \ldots T - 1 \)
  
  \[ BP(t, 0) = 0; \ Bscr(t, 0) = Bscr(t - 1,0) \times y_t^{S(0)} \]
  
  for \( l = 1 \ldots K - 1 \)
  
  • \( BP(t, l) = (if \ (Bscr(t - 1, l - 1) > Bscr(t - 1, l)) \ l - 1; \ else \ l) \)
  
  • \( Bscr(t, l) = Bscr(BP(t, l)) \times y_t^{S(l)} \)
Viterbi algorithm

- \( s(T - 1) = S(K - 1) \)
• \( s(T - 1) = S(K - 1) \)
• for \( t = T - 1 \) down to 1
  \[
  s(t - 1) = BP(s(t))
  \]
Viterbi algorithm

\[
\begin{align*}
\text{for } t &= T - 1 \text{ downto } 1 \\
    s(t - 1) &= BP(s(t))
\end{align*}
\]
Gradients from the alignment

\[ \text{DIV} = \sum_t \text{Xent}(Y_t, \text{symbol}_t^{\text{bestpath}}) = -\sum_t \log Y(t, \text{symbol}_t^{\text{bestpath}}) \]

- The gradient w.r.t the \( t \)-th output vector \( Y_t \)

\[ \nabla_{Y_t} \text{DIV} = \begin{bmatrix} 0 & 0 & \ldots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & \ldots & 0 \end{bmatrix} \frac{1}{Y(t, \text{symbol}_t^{\text{bestpath}})} \]

- Zeros except at the component corresponding to the target \textit{in the estimated alignment}
Iterative Estimate and Training

The "decode" and "train" steps may be combine into a single "decode, find alignment, compute derivatives" step for SGD and mini-batch updates.
Iterative update

• Option 1:
  – Determine alignments for every training instance
  – Train model (using SGD or your favorite approach) on
    the entire training set
  – Iterate

• Option 2:
  – During SGD, for each training instance, find the
    alignment during the forward pass
  – Use in backward pass
Iterative update: Problem

• Approach heavily dependent on initial alignment

• Prone to poor local optima

• Alternate solution: Do not commit to an alignment during any pass..
### The reason for suboptimality

<table>
<thead>
<tr>
<th>/B/</th>
<th>$y_0^B$</th>
<th>$y_1^B$</th>
<th>$y_2^B$</th>
<th>$y_3^B$</th>
<th>$y_4^B$</th>
<th>$y_5^B$</th>
<th>$y_6^B$</th>
<th>$y_7^B$</th>
<th>$y_8^B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/F/</td>
<td>$y_0^F$</td>
<td>$y_1^F$</td>
<td>$y_2^F$</td>
<td>$y_3^F$</td>
<td>$y_4^F$</td>
<td>$y_5^F$</td>
<td>$y_6^F$</td>
<td>$y_7^F$</td>
<td>$y_8^F$</td>
</tr>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
</tbody>
</table>

- **We commit** to the single “best” estimated alignment
  - The *most likely* alignment

\[
DIV = - \sum_t \log Y(t, symbol_t^{bestpath})
\]

- This can be way off, particularly in early iterations, or if the model is poorly initialized
The reason for suboptimality

- **We commit** to the single “best” estimated alignment
  - The *most likely* alignment
    $$DIV = - \sum_{t} \log Y(t, symbol_t^{bestpath})$$
    - This can be way off, particularly in early iterations, or if the model is poorly initialized

- **Alternate view:** there is a probability distribution over alignments of the target Symbol sequence (to the input)
  - *Selecting a single alignment is the same as drawing a single sample from it*
  - Selecting the most likely alignment is the same as deterministically always drawing the most probable value from the distribution
Averaging over all alignments

- Instead of only selecting the most likely alignment, use the statistical expectation over all possible alignments

\[
DIV = E \left[ - \sum_{t} \log Y(t, s_t) \right]
\]

- Use the entire distribution of alignments
- This will mitigate the issue of suboptimal selection of alignment
The expectation over all alignments

\[
DIV = E \left[ - \sum_t \log Y(t, s_t) \right]
\]

- Using the linearity of expectation

\[
DIV = - \sum_t E[\log Y(t, s_t)]
\]

- This reduces to finding the expected divergence at each input

\[
DIV = - \sum_t \sum_{s \in S_1 \ldots S_K} P(s_t = s | S, X) \log Y(t, s_t = s)
\]
The expectation over all alignments

\[
DIV = - \sum_t E[\log Y(t, S_t)]
\]

- This reduces to finding the expected divergence at each input

\[
DIV = - \sum_t \sum_{S \in S_1 \ldots S_K} P(s_t = S | S, X) \log Y(t, s_t = S)
\]

The probability of seeing the specific symbol \( s \) at time \( t \), given that the symbol sequence is an expansion of \( S = S_0 \ldots S_{K-1} \) and given the input sequence \( X = X_0 \ldots X_{N-1} \). We need to be able to compute this.
A posteriori probabilities of symbols

\[ P(s_t = S_r | S, X) \propto P(s_t = S_r, S | X) \]

- \( P(s_t = S_r, S | X) \) is the total probability of all valid paths in the graph for target sequence \( S \) that go through the symbol \( S_r \) (the \( r \)th symbol in the sequence \( S_1 \ldots S_K \)) at time \( t \)

- We will compute this using the “forward-backward” algorithm
A posteriori probabilities of symbols

- Decompose $P(s_t = S_r, \mathbf{S}|\mathbf{X})$ as follows:

  $$P(s_t = S_r, \mathbf{S}|\mathbf{X}) = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r-]} \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_r+] \ldots S_K} P(s_0 \ldots s_{t-1}, s_t = S_r, s_{t+1} \ldots s_{N-1}, \mathbf{S}|\mathbf{X})$$

- $[S_{r+}]$ indicates that $s_{t+1}$ might either be $S_r$ or $S_{r+1}$
- $[S_{r-}]$ indicates that $s_{t-1}$ might be either $S_r$ or $S_{r-1}$

$$= \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r-]} \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_r+] \ldots S_K} P(s_0 \ldots s_{t-1}, s_t = S_r, s_{t+1} \ldots s_{N-1}|\mathbf{X})$$

- Because the target symbol sequence $\mathbf{S}$ is implicit in the synchronized sequences $s_0 \ldots s_{N-1}$ which are constrained to be expansions of $\mathbf{S}$
A posteriori probabilities of symbols

\[
P(s_t = S_r, S|X) = \sum_{s_0 \rightarrow s_{t-1} \rightarrow s_1 \cdots [s_r-]} \sum_{s_{t+1} \cdots s_{N-1} \rightarrow [s_{r+}] \cdots S_K} P(s_0 \cdots s_{t-1}, s_t = S_r, s_{t+1} \cdots s_{N-1} |X)
\]

\[
= \sum_{s_0 \rightarrow s_{t-1} \rightarrow s_1 \cdots [s_r-]} \sum_{s_{t+1} \cdots s_{N-1} \rightarrow [s_{r+}] \cdots S_K} P(s_0 \cdots s_{t-1}, s_t = S_r |X)P(s_{t+1} \cdots s_{N-1} |s_0 \cdots s_{t-1}, s_t = S_r, X)
\]
For a recurrent network without feedback from the output we can make the conditional independence assumption:

\[ P(s_t = S_r, S|X) = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_{r-}]} \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_{r+}] \ldots S_K} P(s_0 \ldots s_{t-1}, s_t = S_r, s_{t+1} \ldots s_{N-1} | X) \]

\[ = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_{r-}]} \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_{r+}] \ldots S_K} P(s_0 \ldots s_{t-1}, s_t = S_r | X)P(s_{t+1} \ldots s_{N-1} | s_0 \ldots s_{t-1}, s_t = S_r, X) \]

Note: in reality, this assumption is not valid if the hidden states are unknown, but we will make it anyway.
A posteriori probabilities of symbols

\[
P(s_t = S_r, S|X) = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r-]} \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_r+] \ldots S_K} P(s_0 \ldots s_{t-1}, s_t = S_r | X) P(s_{t+1} \ldots s_{N-1} | X)
\]

\[
= \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r-]} P(s_0 \ldots s_{t-1}, s_t = S_r | X) \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_r+] \ldots S_K} P(s_{t+1} \ldots s_{N-1} | X)
\]
A posteriori probabilities of symbols

\[ P(s_t = S_r, S|X) \]

\[ = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r-]} \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_r+] \ldots S_K} P(s_0 \ldots s_{t-1}, s_t = S_r | X)P(s_{t+1} \ldots s_{N-1} | X) \]

\[ = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r-]} P(s_0 \ldots s_{t-1}, s_t = S_r | X) \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_r+] \ldots S_K} P(s_{t+1} \ldots s_{N-1} | X) \]
The expectation over all alignments

\[ P(s_t = S_r, S|X) = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r -]} P(s_0 \ldots s_{t-1}, s_t = S_r | X) \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_r +] \ldots S_K} P(s_{t+1} \ldots s_{N-1} | X) \]

- We will call the first term the *forward probability* \( \alpha(t, r) \)
- We will call the second term the *backward* probability \( \beta(t, r) \)
\[ \alpha(t, r) = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r]} P(s_0 \ldots s_{t-1}, s_t = S_r | X) \]

\[ = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r]} P(s_0 \ldots s_{t-1} | X) P(s_t = S_r | s_0 \ldots s_{t-1}, X) \]
\[
\alpha(t, r) = \sum_{s_0 ... s_{t-1} \rightarrow s_1 ... [s_r-]} P(s_0 ... s_{t-1}, s_t = S_r | X)
\]
\[
= \sum_{s_0 ... s_{t-1} \rightarrow S_1 ... [S_r-]} P(s_0 ... s_{t-1} | X) P(s_t = S_r | s_0 ... s_{t-1}, X)
\]
\[
= \sum_{s_0 ... s_{t-1} \rightarrow S_1 ... [S_r-]} P(s_0 ... s_{t-1} | X) P(s_t = S_r | X)
\]
\[ \alpha(t, r) = \sum_{s_0 \ldots s_{t-1} \rightarrow 1 \ldots [S_r-]} P(s_0 \ldots s_{t-1}, s_t = S_r \mid X) \]

\[ = \sum_{s_0 \ldots s_{t-1} \rightarrow 1 \ldots [S_r-]} P(s_0 \ldots s_{t-1} \mid X) P(s_t = S_r \mid s_0 \ldots s_{t-1}, X) \]

\[ = \sum_{s_0 \ldots s_{t-1} \rightarrow 1 \ldots [S_r-]} P(s_0 \ldots s_{t-1} \mid X) P(s_t = S_r \mid X) \]

\[ = \left( \sum_{s_0 \ldots s_{t-2} \rightarrow 1 \ldots [S_r-]} P(s_0 \ldots s_{t-2}, s_{t-1} = S_r \mid X) + \sum_{s_0 \ldots s_{t-2} \rightarrow 1 \ldots [S_{(r-1)}-]} P(s_0 \ldots s_{t-2}, s_{t-1} = S_{r-1} \mid X) \right) P(s_t = S_r \mid X) \]
\[ \alpha(t, r) = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots S_{r-1}} P(s_0 \ldots s_{t-1}, s_t = S_r | X) \]
\[ = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots S_{r-1}} P(s_0 \ldots s_{t-1} | X) P(s_t = S_r | s_0 \ldots s_{t-1}, X) \]
\[ = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots S_{r-1}} P(s_0 \ldots s_{t-1} | X) P(s_t = S_r | X) \]
\[ = \left( \sum_{s_0 \ldots s_{t-2} \rightarrow S_1 \ldots S_{r-1}} P(s_0 \ldots s_{t-2}, s_{t-1} = S_r | X) + \sum_{s_0 \ldots s_{t-2} \rightarrow S_1 \ldots [S_{(r-1)-1}]} P(s_0 \ldots s_{t-2}, s_{t-1} = S_{r-1} | X) \right) P(s_t = S_r | X) \]
Forward algorithm

\[ \alpha(t, r) = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r-]} P(s_0 \ldots s_{t-1}, s_t = S_r \mid X) \]

\[ = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r-]} P(s_0 \ldots s_{t-1} \mid X) P(s_t = S_r \mid s_0 \ldots s_{t-1}, X) \]

\[ = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r-]} P(s_0 \ldots s_{t-1} \mid X) P(s_t = S_r \mid X) \]

\[ = \left( \sum_{s_0 \ldots s_{t-2} \rightarrow S_1 \ldots [S_r-]} P(s_0 \ldots s_{t-2}, s_{t-1} = S_r \mid X) + \sum_{s_0 \ldots s_{t-2} \rightarrow S_1 \ldots [S_{r-1}-]} P(s_0 \ldots s_{t-2}, s_{t-1} = S_{r-1} \mid X) \right) P(s_t = S_r \mid X) \]

\[ \alpha(t, r) = (\alpha(t-1, r) + \alpha(t-1, r-1)) y_t^{S(r)} \]
Forward algorithm

\[ \alpha(t, r) = \left( \alpha(t - 1, r) + \alpha(t - 1, r - 1) \right) y_t^S(r) \]
### Forward algorithm

<table>
<thead>
<tr>
<th>/B/</th>
<th>/IY/</th>
<th>/F/</th>
<th>/IY/</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_0^B$</td>
<td>$y_0^{IY}$</td>
<td>$y_0^F$</td>
<td>$y_0^{IY}$</td>
</tr>
<tr>
<td>$y_1^B$</td>
<td>$y_1^{IY}$</td>
<td>$y_1^F$</td>
<td>$y_1^{IY}$</td>
</tr>
<tr>
<td>$y_2^B$</td>
<td>$y_2^{IY}$</td>
<td>$y_2^F$</td>
<td>$y_2^{IY}$</td>
</tr>
<tr>
<td>$y_3^B$</td>
<td>$y_3^{IY}$</td>
<td>$y_3^F$</td>
<td>$y_3^{IY}$</td>
</tr>
<tr>
<td>$y_4^B$</td>
<td>$y_4^{IY}$</td>
<td>$y_4^F$</td>
<td>$y_4^{IY}$</td>
</tr>
<tr>
<td>$y_5^B$</td>
<td>$y_5^{IY}$</td>
<td>$y_5^F$</td>
<td>$y_5^{IY}$</td>
</tr>
<tr>
<td>$y_6^B$</td>
<td>$y_6^{IY}$</td>
<td>$y_6^F$</td>
<td>$y_6^{IY}$</td>
</tr>
<tr>
<td>$y_7^B$</td>
<td>$y_7^{IY}$</td>
<td>$y_7^F$</td>
<td>$y_7^{IY}$</td>
</tr>
<tr>
<td>$y_8^B$</td>
<td>$y_8^{IY}$</td>
<td>$y_8^F$</td>
<td>$y_8^{IY}$</td>
</tr>
</tbody>
</table>

- **Initialization:**
  \[
  \alpha(0,1) = y_0^{S(1)}, \quad \alpha(0,r) = 0, \quad r > 1
  \]

- **for** \( t = 1 \ldots T - 1 \)
  \[
  \alpha(t, 1) = \alpha(t - 1,1)y_t^{S(1)}
  \]
  for \( l = 2 \ldots K \)
  \[
  \alpha(t, l) = (\alpha(t - 1, l) + \alpha(t - 1, l - 1))y_t^{S(l)}
  \]
**Forward algorithm**

- **Initialization:**
  \[ \alpha(0,1) = y_0^{S(1)}, \quad \alpha(0,r) = 0, \quad r > 1 \]

- **for** \( t = 1 \ldots T - 1 \)
  \[ \alpha(t, 1) = \alpha(t - 1, 1)y_t^{S(1)} \]
  for \( l = 2 \ldots K \)
  \[ \alpha(t, l) = (\alpha(t - 1, l) + \alpha(t - 1, l - 1))y_t^{S(l)} \]
Forward algorithm

- Initialization:
  \[ \alpha(0, 1) = y_0^{S(1)}, \quad \alpha(0, r) = 0, \quad r > 1 \]

- for \( t = 1 \ldots T - 1 \)
  \[ \alpha(t, 1) = \alpha(t - 1, 1)y_t^{S(1)} \]
  for \( l = 2 \ldots K \)
  \[ \alpha(t, l) = (\alpha(t - 1, l) + \alpha(t - 1, l - 1))y_t^{S(l)} \]
Forward algorithm

- Initialization:
  \[ \alpha(0,1) = y_0^{S(1)}, \quad \alpha(0,r) = 0, \quad r > 1 \]

- for \( t = 1 \ldots T - 1 \)
  \[ \alpha(t, 1) = \alpha(t - 1, 1)y_t^{S(1)} \]
  for \( l = 2 \ldots K \)
  \[ \alpha(t, l) = (\alpha(t - 1, l) + \alpha(t - 1, l - 1))y_t^{S(l)} \]
Forward algorithm

- Initialization:
  \[ \alpha(0,1) = y_0^{S(1)}, \quad \alpha(0,r) = 0, \quad r > 1 \]
- for \( t = 1 \ldots T - 1 \)
  \[ \alpha(t, 1) = \alpha(t - 1, 1)y_t^{S(1)} \]
  for \( l = 2 \ldots K \)
  \[ \alpha(t, l) = (\alpha(t - 1, l) + \alpha(t - 1, l - 1))y_t^{S(l)} \]
Forward algorithm

- Initialization:
  \[ \alpha(0,1) = y_0^{S(1)}, \quad \alpha(0,r) = 0, \quad r > 1 \]
- for \( t = 1 \ldots T - 1 \)
  \[ \alpha(t, 1) = \alpha(t - 1, 1)y_t^{S(1)} \]
  for \( l = 2 \ldots K \)
  - \( \alpha(t, l) = (\alpha(t - 1, l) + \alpha(t - 1, l - 1))y_t^{S(l)} \)
In practice..

• The recursion
  \[ \alpha(t, l) = (\alpha(t - 1, l) + \alpha(t - 1, l - 1))y_t^{S(l)} \]
  will generally underflow

• Instead we can do it in the \textit{log} domain
  \[ \log\alpha(t, l) \]
  \[ = \log(e^{\log\alpha(t-1,l)} + e^{\log\alpha(t-1,l-1)}) + \log y_t^{S(l)} \]
  – This can be computed entirely without underflow
Forward algorithm

- Initialization:
  \[ \alpha(0,1) = 1, \quad \alpha(0,r) = 0, \quad r > 1 \]
  \[ \alpha(t,r) = \hat{\alpha}(t,r) y_0^{S(r)}, \quad 1 \leq r \leq K \]

- for \( t = 1 \ldots T - 1 \)
  \[ \hat{\alpha}(t,1) = \alpha(t-1,1) \]
  for \( l = 2 \ldots K \)
    - \[ \hat{\alpha}(t,l) = \alpha(t-1,l) + \alpha(t-1,l-1) \]
  \[ \alpha(t,r) = \hat{\alpha}(t,r) y_0^{S(r)}, \quad 1 \leq r \leq K \]
The forward probability

\[ P(s_t = S_r, S | X) = \sum_{s_0 \ldots s_{t-1} \rightarrow S_1 \ldots [S_r-]} P(s_0 \ldots s_{t-1}, s_t = S_r | X) \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_r+] \ldots S_K} P(s_{t+1} \ldots s_{N-1} | X) \]

- We will call the first term the forward probability \( \alpha(t, r) \)
- We will call the second term the backward probability \( \beta(t, r) \)

We have seen how to compute this \( \alpha(t, r) \)
The forward probability

\[ P(s_t = S_r, S | X) = \alpha(t, r) \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_{r+}] \ldots S_K} P(s_{t+1} \ldots s_{N-1} | X) \]

- We will call the first term the **forward probability** \( \alpha(t, r) \)
- We will call the second term the **backward probability** \( \beta(t, r) \)

We have seen how to compute this
The forward probability

\[ P(s_t = S_r, S|X) = \alpha(t, r) \sum_{s_{t+1}...s_{N-1} \rightarrow [S_r+1]...S_K} P(s_{t+1}...s_{N-1} | X) \]

- We will call the first term the *forward probability* \( \alpha(t, r) \)
- We will call the second term the *backward probability* \( \beta(t, r) \)

\( \beta(t, r) \)  \( \text{Lets look at this} \)
\[
\beta(t, r) = \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_r+1] \ldots S_K} P(s_{t+1} \ldots s_{N-1} | \mathbf{X})
\]

\[
= \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_r+1] \ldots S_K} P(s_{t+1} = S_r, s_{t+2} \ldots s_{N-1} | \mathbf{X}) + \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_{(r+1)+1}] \ldots S_K} P(s_{t+1} = S_{r+1}, s_{t+2} \ldots s_{N-1} | \mathbf{X})
\]
\[ \beta(t, r) = \sum_{s_{t+1} \cdots s_{N-1} \rightarrow [S_r] \cdots S_K} P(s_{t+1} \cdots s_{N-1} | X) \]

\[ = \sum_{s_{t+1} = S_r, s_{t+2} \cdots s_{N-1} \rightarrow [S_{(r+1)}] \cdots S_K} P(s_{t+1} = S_r, s_{t+2} \cdots s_{N-1} | X) + \sum_{s_{t+2} \cdots s_{N-1} \rightarrow [S_{(r+1)}] \cdots S_K} P(s_{t+1} = S_{r+1}, s_{t+2} \cdots s_{N-1} | X) \]

\[ = P(s_{t+1} = S_r | X) \sum_{s_{t+2} \cdots s_{N-1} \rightarrow [S_r] \cdots S_K} P(s_{t+2} \cdots s_{N-1} | s_{t+1} = S_r, X) \]

\[ + P(s_{t+1} = S_{r+1} | X) \sum_{s_{t+2} \cdots s_{N-1} \rightarrow [S_{(r+1)}] \cdots S_K} P(s_{t+2} \cdots s_{N-1} | s_{t+1} = S_{r+1}, X) \]
\[
\beta(t, r) = \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_r] \ldots S_K} P(s_{t+1} \ldots s_{N-1} | X) \\
= \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_r] \ldots S_K} P(s_{t+2} \ldots s_{N-1} | s_{t+1} = S_r, X) + \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_{(r+1)}] \ldots S_K} P(s_{t+1} = S_{r+1}, s_{t+2} \ldots s_{N-1} | X) \\
= P(s_{t+1} = S_r | X) \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_r] \ldots S_K} P(s_{t+2} \ldots s_{N-1} | s_{t+1} = S_r, X) \\
+ P(s_{t+1} = S_{r+1} | X) \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_{(r+1)}] \ldots S_K} P(s_{t+2} \ldots s_{N-1} | s_{t+1} = S_{r+1}, X) \\
= P(s_{t+1} = S_r | X) \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_r] \ldots S_K} P(s_{t+2} \ldots s_{N-1} | X) + P(s_{t+1} = S_{r+1} | X) \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_{(r+1)}] \ldots S_K} P(s_{t+2} \ldots s_{N-1} | X)
\]
\[\beta(t, r) = \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_r] \ldots S_K} P(s_{t+1} \ldots s_{N-1} | \mathbf{X})\]

\[= \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_r] \ldots S_K} P(s_{t+1} = S_r, s_{t+2} \ldots s_{N-1} | \mathbf{X}) + \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_{(r+1)}] \ldots S_K} P(s_{t+1} = S_{r+1}, s_{t+2} \ldots s_{N-1} | \mathbf{X})\]

\[= P(s_{t+1} = S_r | \mathbf{X}) \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_r] \ldots S_K} P(s_{t+2} \ldots s_{N-1} | s_{t+1} = S_r, \mathbf{X}) + P(s_{t+1} = S_{r+1} | \mathbf{X}) \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_{(r+1)}] \ldots S_K} P(s_{t+2} \ldots s_{N-1} | s_{t+1} = S_{r+1}, \mathbf{X})\]

\[= P(s_{t+1} = S_r | \mathbf{X}) \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_r] \ldots S_K} P(s_{t+2} \ldots s_{N-1} | \mathbf{X}) + P(s_{t+1} = S_{r+1} | \mathbf{X}) \sum_{s_{t+2} \ldots s_{N-1} \rightarrow [S_{(r+1)}] \ldots S_K} P(s_{t+2} \ldots s_{N-1} | \mathbf{X})\]
\[ \beta(t, r) = y_{t+1}^{S(r)} \beta(t + 1, r) + y_{t+1}^{S(r+1)} \beta(t + 1, r + 1) \]
Backward algorithm

- Initialization:
  \[ \beta(T - 1, K) = 1, \quad \beta(T - 1, r) = 0, \quad r < K \]

- for \( t = T - 2 \) downto 0
  \[ \beta(t, K) = \beta(t + 1, K) y_{t+1}^{S(K)} \]

  for \( l = K - 1 \ldots 1 \)
  
  - \( \beta(t, r) = y_{t+1}^{S(l)} \beta(t + 1, r) + y_{t+1}^{S(r+1)} \beta(t + 1, r + 1) \)
**Backward algorithm**

- **Initialization:**
  \[ \beta(T - 1, K) = 1, \quad \beta(T - 1, r) = 0, \quad r < K \]
- **for** \( t = T - 2 \) **down to** 0
  \[ \beta(t, K) = \beta(t + 1, K)y_{t+1}^{S(K)} \]
  for \( l = K - 1 \ldots 1 \)
  - \( \beta(t, r) = y_{t+1}^{S(l)} \beta(t + 1, r) + y_{t+1}^{S(r+1)} \beta(t + 1, r + 1) \)
Backward algorithm

- Initialization:
  \[ \beta(T - 1, K) = 1, \quad \beta(T - 1, r) = 0, \quad r < K \]

- for \( t = T - 2 \) downto 0
  \[ \beta(t, K) = \beta(t + 1, K) \gamma_{t+1}^{S(K)} \]
  for \( l = K - 1 \ldots 1 \)
  - \( \beta(t, r) = \gamma_{t+1}^{S(l)} \beta(t + 1, r) + \gamma_{t+1}^{S(r+1)} \beta(t + 1, r + 1) \)
• Initialization:
  \[ \beta(T - 1, K) = 1, \ \beta(T - 1, r) = 0, \ r < K \]

• for \( t = T - 2 \) downto 0
  \[ \beta(t, K) = \beta(t + 1, K) y_{t+1}^{S(K)} \]
  for \( l = K - 1 \ldots 1 \)
  \[ \cdot \beta(t, r) = y_{t+1}^{S(l)} \beta(t + 1, r) + y_{t+1}^{S(r+1)} \beta(t + 1, r + 1) \]
**Backward algorithm**

- **Initialization:**
  \[ \beta(T - 1, K) = 1, \quad \beta(T - 1, r) = 0, \quad r < K \]

- **for** \( t = T - 2 \) **down to** \( 0 \)
  \[ \beta(t, K) = \beta(t + 1, K) y_{t+1}^{S(K)} \]
  for \( l = K - 1 \ldots 1 \)
   - \[ \beta(t, r) = y_{t+1}^{S(l)} \beta(t + 1, r) + y_{t+1}^{S(r+1)} \beta(t + 1, r + 1) \]
The joint probability

\[ P(s_t = S_r, S|X) = \alpha(t, r) \sum_{s_{t+1} \ldots s_{N-1} \rightarrow [S_r+] \ldots S_K} P(s_{t+1} \ldots s_{N-1} | X) \]

- We will call the first term the **forward probability** \( \alpha(t, r) \)
- We will call the second term the **backward probability** \( \beta(t, r) \)

\[ \beta(t, r) \]

We now can compute this
The joint probability

\[ P(s_t = s_r, s|X) = \alpha(t, r)\beta(t, r) \]

- We will call the first term the \textit{forward probability} \( \alpha(t, r) \)
- We will call the second term the \textit{backward probability} \( \beta(t, r) \)
The posterior probability

\[ P(s_t = S_r, S|X) = \alpha(t, r)\beta(t, r) \]

- The posterior is given by

\[ P(s_t = S_r | S, X) = \frac{P(s_t = S_r, S|X)}{\sum_{S_r'} P(s_t = S_r', S|X)} = \frac{\alpha(t, r)\beta(t, r)}{\sum_r \alpha(t, r')\beta(t, r')} \]

- We can also write this as

\[ P(s_t = S_r | S, X) = \frac{\hat{\alpha}(t, r) y_t^{S(r)} \beta(t, r)}{\hat{\alpha}(t, r) y_t^{S(r)} \beta(t, r) + \sum_{r'\neq r} \alpha(t, r)\beta(t, r')} \]
The expected divergence

\[
\text{DIV} = - \sum_t \sum_{s \in S_1 \ldots S_K} P(s_t = s | S, X) \log Y(t, s_t = s)
\]

\[
\text{DIV} = - \sum_t \sum_r \frac{\alpha(t, r) \beta(t, r)}{\sum_{r'} \alpha(t, r') \beta(t, r')} \log y_t^{s(r)}
\]
The expected divergence

\[
\text{DIV} = -\sum_t \sum_{s \in S_1 \ldots S_K} P(s_t = s | S, X) \log Y(t, s_t = s)
\]

\[
\text{DIV} = -\sum_t \sum_r \sum_{r'} \frac{\alpha(t, r) \beta(t, r)}{\sum_{r'} \alpha(t, r') \beta(t, r')} \log y_t^{s(r)}
\]

- The derivative of the divergence w.r.t the output \(Y_t\) of the net at any time:

\[
\nabla_{y_t} \text{DIV} = \begin{bmatrix}
\frac{d\text{DIV}}{dy_t^1} & \frac{d\text{DIV}}{dy_t^2} & \ldots & \frac{d\text{DIV}}{dy_t^L}
\end{bmatrix}
\]

- Components will be non-zero only for symbols that occur in the training instance.
The expected divergence

\[ DIV = - \sum_t \sum_{s \in S_1 \ldots S_K} P(s_t = s | S, X) \log Y(t, s_t = s) \]

\[ DIV = - \sum_t \sum_r \frac{\alpha(t, r) \beta(t, r)}{\sum_{r'} \alpha(t, r') \beta(t, r')} \log y_t^{s(r)} \]

- The derivative of the divergence w.r.t the output \( Y_t \) of the net at any time:

\[ \nabla_{Y_t} DIV = \begin{bmatrix} \frac{dDIV}{dy_t^1} & \frac{dDIV}{dy_t^2} & \cdots & \frac{dDIV}{dy_t^L} \end{bmatrix} \]

- Components will be non-zero only for symbols that occur in the training instance
The expected divergence

\[ \frac{dD\text{IV}}{dy_t} = - \sum_r \frac{\alpha(t, r) \beta(t, r)}{\sum_{r'} \alpha(t, r') \beta(t, r')} \log y_t^{S(r)} \]

- The derivative of the divergence w.r.t any particular output of the network must sum over all instances of that symbol in the target sequence

\[ \frac{dD\text{IV}}{dy_t^l} = - \sum_{r : S(r) = l} \frac{d}{dy_t^{S(r)}} \left( \frac{\alpha(t, r) \beta(t, r)}{\sum_{r'} \alpha(t, r') \beta(t, r')} \log y_t^{S(r)} \right) \]

- E.g. the derivative w.r.t \( y_t^5 \) will sum over both rows representing /IY/ in the above figure
Overall training procedure for Seq2Seq case 1

/B/ /IY/ /F/ /IY/

• Problem: Given input and output sequences without alignment, train models
Overall training procedure for Seq2Seq case 1

- **Step 1:** Setup the network  
  - Typically many-layered LSTM

- **Step 2:** Initialize all parameters of the network
Overall Training: Forward pass

- Foreach training instance
  - **Step 3**: Forward pass. Pass the training instance through the network and obtain all symbol probabilities at each time.
Overall training: Backward pass

- **Foreach training instance**
  - **Step 3**: Forward pass. Pass the training instance through the network and obtain all symbol probabilities at each time
  - **Step 4**: Construct the graph representing the specific symbol sequence in the instance. This may require having multiple rows of nodes with the same symbol scores
Overall training: Backward pass

- **Foreach training instance:**
  - **Step 5:** Perform the forward backward algorithm to compute $\alpha(t, r)$ and $\beta(t, r)$ at each time, for each row of nodes in the graph
  - **Step 6:** Compute derivative of divergence $\nabla_{Y_t} DIV$ for each $Y_t$
Overall training: Backward pass

- Foreach instance
  - **Step 6**: Compute derivative of divergence $\nabla_{Y_t} DIV$ for each $Y_t$

$$\nabla_{Y_t} DIV = \left[ \frac{dDIV}{dy_{t}^{1}} \quad \frac{dDIV}{dy_{t}^{2}} \quad \ldots \quad \frac{dDIV}{dy_{t}^{L}} \right]$$

$$\frac{dDIV}{dy_{t}^{l}} = - \sum_{r : S(r) = l} \frac{d}{dy_{t}^{S(r)}} \left( \frac{\alpha(t, r) \beta(t, r)}{\sum_{r'} \alpha(t, r') \beta(t, r')} \log y_{t}^{S(r)} \right)$$

- **Step 7**: Aggregate derivatives over minibatch and update parameters
A key decoding problem

• Consider a problem where the output symbols are characters
• We have a decode: R R R O O O O O O D
• Is this the symbol sequence ROD or ROOD?
We’ve seen this before

<table>
<thead>
<tr>
<th>/AH/</th>
<th>$y_0^{AH}$</th>
<th>$y_1^{AH}$</th>
<th>$y_2^{AH}$</th>
<th>$y_3^{AH}$</th>
<th>$y_4^{AH}$</th>
<th>$y_5^{AH}$</th>
<th>$y_6^{AH}$</th>
<th>$y_7^{AH}$</th>
<th>$y_8^{AH}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/B/</td>
<td>$y_0^B$</td>
<td>$y_1^B$</td>
<td>$y_2^B$</td>
<td>$y_3^B$</td>
<td>$y_4^B$</td>
<td>$y_5^B$</td>
<td>$y_6^B$</td>
<td>$y_7^B$</td>
<td>$y_8^B$</td>
</tr>
<tr>
<td>/D/</td>
<td>$y_0^D$</td>
<td>$y_1^D$</td>
<td>$y_2^D$</td>
<td>$y_3^D$</td>
<td>$y_4^D$</td>
<td>$y_5^D$</td>
<td>$y_6^D$</td>
<td>$y_7^D$</td>
<td>$y_8^D$</td>
</tr>
<tr>
<td>/EH/</td>
<td>$y_0^{EH}$</td>
<td>$y_1^{EH}$</td>
<td>$y_2^{EH}$</td>
<td>$y_3^{EH}$</td>
<td>$y_4^{EH}$</td>
<td>$y_5^{EH}$</td>
<td>$y_6^{EH}$</td>
<td>$y_7^{EH}$</td>
<td>$y_8^{EH}$</td>
</tr>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/F/</td>
<td>$y_0^F$</td>
<td>$y_1^F$</td>
<td>$y_2^F$</td>
<td>$y_3^F$</td>
<td>$y_4^F$</td>
<td>$y_5^F$</td>
<td>$y_6^F$</td>
<td>$y_7^F$</td>
<td>$y_8^F$</td>
</tr>
<tr>
<td>/G/</td>
<td>$y_0^G$</td>
<td>$y_1^G$</td>
<td>$y_2^G$</td>
<td>$y_3^G$</td>
<td>$y_4^G$</td>
<td>$y_5^G$</td>
<td>$y_6^G$</td>
<td>$y_7^G$</td>
<td>$y_8^G$</td>
</tr>
</tbody>
</table>

Cannot distinguish between an extended symbol and repetitions of the symbol

• /G/ /F/ /F/ /IY/ /D/ or /G/ /F/ /IY/ /D/ ?
A key *decoding* problem

- Consider a problem where the output symbols are characters.
- We have a decode: R R R O O O O O O D
- Is this the symbol sequence ROD or ROOD?

- Note: This problem does not always occur, e.g. when symbols have sub symbols
  - E.g. If O is produced as O1 and O2
    - A single O would be of the form O1 O1 .. O2 → O
    - Multiple Os would have the decode O1 .. O2.. O1..O2.. → OO
A key decoding problem

• We have a decode: R R R O O O O O D
• Is this the symbol sequence ROD or ROOD?

• Solution: Introduce an explicit extra symbol which serves to separate discrete versions of a symbol
  – A “blank” (represented by “-”)
  – RRR---OO---DDD = ROD
  – RR-R---OO---D-DD = RRODD
  – R-R-R---O-O--DD-DDDD-D = RRROODDD
    • The next symbol at the end of a sequence of blanks is always a new character
    • When a symbol repeats, there must be at least one blank between the repetitions

• The symbol set recognized by the network must now include the extra blank symbol
  – Which too must be trained
The modified forward output

- Note the extra “blank” at the output
The modified forward output

- Note the extra “blank” at the output

/B/ /IY/ /F/ /IY/

<table>
<thead>
<tr>
<th></th>
<th>( y_0^b )</th>
<th>( y_1^b )</th>
<th>( y_2^b )</th>
<th>( y_3^b )</th>
<th>( y_4^b )</th>
<th>( y_5^b )</th>
<th>( y_6^b )</th>
<th>( y_7^b )</th>
<th>( y_8^b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>/AH/</td>
<td>( y_0^{AH} )</td>
<td>( y_1^{AH} )</td>
<td>( y_2^{AH} )</td>
<td>( y_3^{AH} )</td>
<td>( y_4^{AH} )</td>
<td>( y_5^{AH} )</td>
<td>( y_6^{AH} )</td>
<td>( y_7^{AH} )</td>
<td>( y_8^{AH} )</td>
</tr>
<tr>
<td>/B/</td>
<td>( y_0^B )</td>
<td>( y_1^B )</td>
<td>( y_2^B )</td>
<td>( y_3^B )</td>
<td>( y_4^B )</td>
<td>( y_5^B )</td>
<td>( y_6^B )</td>
<td>( y_7^B )</td>
<td>( y_8^B )</td>
</tr>
<tr>
<td>/D/</td>
<td>( y_0^D )</td>
<td>( y_1^D )</td>
<td>( y_2^D )</td>
<td>( y_3^D )</td>
<td>( y_4^D )</td>
<td>( y_5^D )</td>
<td>( y_6^D )</td>
<td>( y_7^D )</td>
<td>( y_8^D )</td>
</tr>
<tr>
<td>/EH/</td>
<td>( y_0^{EH} )</td>
<td>( y_1^{EH} )</td>
<td>( y_2^{EH} )</td>
<td>( y_3^{EH} )</td>
<td>( y_4^{EH} )</td>
<td>( y_5^{EH} )</td>
<td>( y_6^{EH} )</td>
<td>( y_7^{EH} )</td>
<td>( y_8^{EH} )</td>
</tr>
<tr>
<td>/IY/</td>
<td>( y_0^{IY} )</td>
<td>( y_1^{IY} )</td>
<td>( y_2^{IY} )</td>
<td>( y_3^{IY} )</td>
<td>( y_4^{IY} )</td>
<td>( y_5^{IY} )</td>
<td>( y_6^{IY} )</td>
<td>( y_7^{IY} )</td>
<td>( y_8^{IY} )</td>
</tr>
<tr>
<td>/F/</td>
<td>( y_0^F )</td>
<td>( y_1^F )</td>
<td>( y_2^F )</td>
<td>( y_3^F )</td>
<td>( y_4^F )</td>
<td>( y_5^F )</td>
<td>( y_6^F )</td>
<td>( y_7^F )</td>
<td>( y_8^F )</td>
</tr>
<tr>
<td>/G/</td>
<td>( y_0^G )</td>
<td>( y_1^G )</td>
<td>( y_2^G )</td>
<td>( y_3^G )</td>
<td>( y_4^G )</td>
<td>( y_5^G )</td>
<td>( y_6^G )</td>
<td>( y_7^G )</td>
<td>( y_8^G )</td>
</tr>
</tbody>
</table>

Diagram showing the forward output process with highlighted states.
The modified forward output

- Note the extra “blank” at the output

/B/ /IY/ /F/ /IY/

<table>
<thead>
<tr>
<th>-</th>
<th>(y_0^b)</th>
<th>(y_1^b)</th>
<th>(y_2^b)</th>
<th>(y_3^b)</th>
<th>(y_4^b)</th>
<th>(y_5^b)</th>
<th>(y_6^b)</th>
<th>(y_7^b)</th>
<th>(y_8^b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/AH/</td>
<td>(y_0^{AH})</td>
<td>(y_1^{AH})</td>
<td>(y_2^{AH})</td>
<td>(y_3^{AH})</td>
<td>(y_4^{AH})</td>
<td>(y_5^{AH})</td>
<td>(y_6^{AH})</td>
<td>(y_7^{AH})</td>
<td>(y_8^{AH})</td>
</tr>
<tr>
<td>/B/</td>
<td>(y_0^B)</td>
<td>(y_1^B)</td>
<td>(y_2^B)</td>
<td>(y_3^B)</td>
<td>(y_4^B)</td>
<td>(y_5^B)</td>
<td>(y_6^B)</td>
<td>(y_7^B)</td>
<td>(y_8^B)</td>
</tr>
<tr>
<td>/D/</td>
<td>(y_0^D)</td>
<td>(y_1^D)</td>
<td>(y_2^D)</td>
<td>(y_3^D)</td>
<td>(y_4^D)</td>
<td>(y_5^D)</td>
<td>(y_6^D)</td>
<td>(y_7^D)</td>
<td>(y_8^D)</td>
</tr>
<tr>
<td>/EH/</td>
<td>(y_0^{EH})</td>
<td>(y_1^{EH})</td>
<td>(y_2^{EH})</td>
<td>(y_3^{EH})</td>
<td>(y_4^{EH})</td>
<td>(y_5^{EH})</td>
<td>(y_6^{EH})</td>
<td>(y_7^{EH})</td>
<td>(y_8^{EH})</td>
</tr>
<tr>
<td>/IY/</td>
<td>(y_0^{IY})</td>
<td>(y_1^{IY})</td>
<td>(y_2^{IY})</td>
<td>(y_3^{IY})</td>
<td>(y_4^{IY})</td>
<td>(y_5^{IY})</td>
<td>(y_6^{IY})</td>
<td>(y_7^{IY})</td>
<td>(y_8^{IY})</td>
</tr>
<tr>
<td>/F/</td>
<td>(y_0^F)</td>
<td>(y_1^F)</td>
<td>(y_2^F)</td>
<td>(y_3^F)</td>
<td>(y_4^F)</td>
<td>(y_5^F)</td>
<td>(y_6^F)</td>
<td>(y_7^F)</td>
<td>(y_8^F)</td>
</tr>
<tr>
<td>/G/</td>
<td>(y_0^G)</td>
<td>(y_1^G)</td>
<td>(y_2^G)</td>
<td>(y_3^G)</td>
<td>(y_4^G)</td>
<td>(y_5^G)</td>
<td>(y_6^G)</td>
<td>(y_7^G)</td>
<td>(y_8^G)</td>
</tr>
</tbody>
</table>

- Diagram of the modified forward output process with nodes labeled from X₀ to X₈.
The modified forward output

• Note the extra “blank” at the output
### Composing the graph for training

<table>
<thead>
<tr>
<th></th>
<th>(y_0^B)</th>
<th>(y_1^B)</th>
<th>(y_2^B)</th>
<th>(y_3^B)</th>
<th>(y_4^B)</th>
<th>(y_5^B)</th>
<th>(y_6^B)</th>
<th>(y_7^B)</th>
<th>(y_8^B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/B/</td>
<td>(y_0^B)</td>
<td>(y_1^B)</td>
<td>(y_2^B)</td>
<td>(y_3^B)</td>
<td>(y_4^B)</td>
<td>(y_5^B)</td>
<td>(y_6^B)</td>
<td>(y_7^B)</td>
<td>(y_8^B)</td>
</tr>
<tr>
<td>/IY/</td>
<td>(y_0^IY)</td>
<td>(y_1^IY)</td>
<td>(y_2^IY)</td>
<td>(y_3^IY)</td>
<td>(y_4^IY)</td>
<td>(y_5^IY)</td>
<td>(y_6^IY)</td>
<td>(y_7^IY)</td>
<td>(y_8^IY)</td>
</tr>
<tr>
<td>/IY/</td>
<td>(y_0^IY)</td>
<td>(y_1^IY)</td>
<td>(y_2^IY)</td>
<td>(y_3^IY)</td>
<td>(y_4^IY)</td>
<td>(y_5^IY)</td>
<td>(y_6^IY)</td>
<td>(y_7^IY)</td>
<td>(y_8^IY)</td>
</tr>
<tr>
<td>/F/</td>
<td>(y_0^F)</td>
<td>(y_1^F)</td>
<td>(y_2^F)</td>
<td>(y_3^F)</td>
<td>(y_4^F)</td>
<td>(y_5^F)</td>
<td>(y_6^F)</td>
<td>(y_7^F)</td>
<td>(y_8^F)</td>
</tr>
</tbody>
</table>

- The original method without blanks
- Changing the example to /B/ /IY/ /IY/ /F/ from /B/ /IY/ /F/ /IY/ for illustration
Composing the graph for training

• With blanks
• Note: a row of blanks between any two symbols
• Also blanks at the very beginning and the very end
Composing the graph for training

-  
-  
-  
-  

Add edges such that all paths from initial node(s) to final node(s) unambiguously represent the target symbol sequence
Composing the graph for training

- /B/
  - /IY/
  - /IY/
  - /F/
  -

<table>
<thead>
<tr>
<th>(y_0^b)</th>
<th>(y_1^b)</th>
<th>(y_2^b)</th>
<th>(y_3^b)</th>
<th>(y_4^b)</th>
<th>(y_5^b)</th>
<th>(y_6^b)</th>
<th>(y_7^b)</th>
<th>(y_8^b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y_0^B)</td>
<td>(y_1^B)</td>
<td>(y_2^B)</td>
<td>(y_3^B)</td>
<td>(y_4^B)</td>
<td>(y_5^B)</td>
<td>(y_6^B)</td>
<td>(y_7^B)</td>
<td>(y_8^B)</td>
</tr>
<tr>
<td>(y_0^IY)</td>
<td>(y_1^IY)</td>
<td>(y_2^IY)</td>
<td>(y_3^IY)</td>
<td>(y_4^IY)</td>
<td>(y_5^IY)</td>
<td>(y_6^IY)</td>
<td>(y_7^IY)</td>
<td>(y_8^IY)</td>
</tr>
<tr>
<td>(y_0^IY)</td>
<td>(y_1^IY)</td>
<td>(y_2^IY)</td>
<td>(y_3^IY)</td>
<td>(y_4^IY)</td>
<td>(y_5^IY)</td>
<td>(y_6^IY)</td>
<td>(y_7^IY)</td>
<td>(y_8^IY)</td>
</tr>
<tr>
<td>(y_0^F)</td>
<td>(y_1^F)</td>
<td>(y_2^F)</td>
<td>(y_3^F)</td>
<td>(y_4^F)</td>
<td>(y_5^F)</td>
<td>(y_6^F)</td>
<td>(y_7^F)</td>
<td>(y_8^F)</td>
</tr>
</tbody>
</table>

- The first and last column are allowed to also end at initial and final blanks
Composing the graph for training

- The first and last column are allowed to also end at initial and final blanks
- Skips are permitted across a blank, but only if the symbols on either side are different
  - Because a blank is *mandatory between repetitions of a symbol* but *not required between distinct symbols*
• **Initialization:**

\[-\alpha(0,0) = y_0^b, \alpha(0,1) = y_0^b, \alpha(0,r) = 0 \quad r > 1\]
## Modified Forward Algorithm

### Iteration:

\[
\alpha(t, r) = \left( \alpha(t - 1, r) + \alpha(t - 1, r - 1) \right) y_t^{S(r)}
\]

- If \( S(r) = " - " \) or \( S(r) = S(r - 2) \)

\[
\alpha(t, r) = \left( \alpha(t - 1, r) + \alpha(t - 1, r - 1) + \alpha(t - 1, r - 2) \right) y_t^{S(r)}
\]

- Otherwise
Modified Forward Algorithm

- Iteration:

\[ \alpha(t, r) = (\alpha(t - 1, r) + \alpha(t - 1, r - 1))y_t^{S(r)} \]

- If \( S(r) = "-" \) or \( S(r) = S(r - 2) \)

\[ \alpha(t, r) = (\alpha(t - 1, r) + \alpha(t - 1, r - 1) + \alpha(t - 1, r - 2))y_t^{S(r)} \]

- Otherwise
• Initialization:

\[ \beta(T - 1, 2K + 1) = \beta(T - 1, 2K) = \beta(T - 1, r) = 0 \quad r < 2K \]
Modified Backward Algorithm

- \( y_0^b \)
  - \( y_0^B \)
  - \( y_0^{IY} \)
  - \( y_0^F \)
  - \( y_0^b \)
  - \( y_0^B \)
  - \( y_0^{IY} \)
  - \( y_0^{IY} \)
  - \( y_0^F \)
  - \( y_0^b \)
  - \( y_1^b \)
  - \( y_1^b \)
  - \( y_1^B \)
  - \( y_1^{IY} \)
  - \( y_1^b \)
  - \( y_1^b \)
  - \( y_1^b \)
  - \( y_1^b \)
  - \( y_1^b \)
  - \( y_1^b \)
  - \( y_1^b \)
  - \( y_1^b \)
  - \( y_2^b \)
  - \( y_2^b \)
  - \( y_2^b \)
  - \( y_2^b \)
  - \( y_2^b \)
  - \( y_2^b \)
  - \( y_2^b \)
  - \( y_2^b \)
  - \( y_3^b \)
  - \( y_3^b \)
  - \( y_3^b \)
  - \( y_3^b \)
  - \( y_3^b \)
  - \( y_3^b \)
  - \( y_3^b \)
  - \( y_3^b \)
  - \( y_4^b \)
  - \( y_4^b \)
  - \( y_4^b \)
  - \( y_4^b \)
  - \( y_4^b \)
  - \( y_4^b \)
  - \( y_4^b \)
  - \( y_4^b \)
  - \( y_5^b \)
  - \( y_5^b \)
  - \( y_5^b \)
  - \( y_5^b \)
  - \( y_5^b \)
  - \( y_5^b \)
  - \( y_5^b \)
  - \( y_5^b \)
  - \( y_6^b \)
  - \( y_6^b \)
  - \( y_6^b \)
  - \( y_6^b \)
  - \( y_6^b \)
  - \( y_6^b \)
  - \( y_6^b \)
  - \( y_6^b \)
  - \( y_7^b \)
  - \( y_7^b \)
  - \( y_7^b \)
  - \( y_7^b \)
  - \( y_7^b \)
  - \( y_7^b \)
  - \( y_7^b \)
  - \( y_7^b \)
  - \( y_8^b \)

- Iteration:

\[ \beta(t, r) = \beta(t + 1, r) y_t^{S(r)} + \beta(t + 1, r + 1) y_t^{S(r+1)} \]

- If \( S(r) = "-" \) or \( S(r) = S(r + 2) \)

\[ \beta(t, r) = \beta(t + 1, r) y_t^{S(r)} + \beta(t + 1, r + 1) y_t^{S(r+1)} + \beta(t + 1, r + 2) y_t^{S(r+2)} \]

- Otherwise
Overall training procedure for Seq2Seq with blanks

Problem: Given input and output sequences without alignment, train models
Overall training procedure

• **Step 1**: Setup the network
  – Typically many-layered LSTM

• **Step 2**: Initialize all parameters of the network
  – Include a “blank” symbol in vocabulary
Overall Training: Forward pass

• Foreach training instance
  • **Step 3**: Forward pass. Pass the training instance through the network and obtain all symbol probabilities at each time, including blanks

```
<table>
<thead>
<tr>
<th></th>
<th>y₀ᵇ</th>
<th>y₁ᵇ</th>
<th>y₂ᵇ</th>
<th>y₃ᵇ</th>
<th>y₄ᵇ</th>
<th>y₅ᵇ</th>
<th>y₆ᵇ</th>
<th>y₇ᵇ</th>
<th>y₈ᵇ</th>
</tr>
</thead>
<tbody>
<tr>
<td>/AH/</td>
<td>y₀ᴬᴴ</td>
<td>y₁ᴬᴴ</td>
<td>y₂ᴬᴴ</td>
<td>y₃アウ</td>
<td>y₄ᴬᴴ</td>
<td>y₅アウ</td>
<td>y₆アウ</td>
<td>y₇アウ</td>
<td>y₈アウ</td>
</tr>
<tr>
<td>/B/</td>
<td>y₀ᴮ</td>
<td>y₁ᴮ</td>
<td>y₂ᴮ</td>
<td>y₃ᴮ</td>
<td>y₄ᴮ</td>
<td>y₅ᴮ</td>
<td>y₆ᴮ</td>
<td>y₇ᴮ</td>
<td>y₈ᴮ</td>
</tr>
<tr>
<td>/D/</td>
<td>y₀ᴰ</td>
<td>y₁ᴰ</td>
<td>y₂ᴰ</td>
<td>y₃ᴰ</td>
<td>y₄ᴰ</td>
<td>y₅ᴰ</td>
<td>y₆ᴰ</td>
<td>y₇ᴰ</td>
<td>y₈ᴰ</td>
</tr>
<tr>
<td>/EH/</td>
<td>y₀ᴱᴴ</td>
<td>y₁ᴱᴴ</td>
<td>y₂ᴱᴴ</td>
<td>y₃_km</td>
<td>y₄ᴱᴴ</td>
<td>y₅ᴱᴴ</td>
<td>y₆ᴱᴴ</td>
<td>y₇ᴱᴴ</td>
<td>y₈ᴱᴴ</td>
</tr>
<tr>
<td>/IY/</td>
<td>y₀ᴵŸ</td>
<td>y₁ᴵŸ</td>
<td>y₂ᴵŸ</td>
<td>y₃ᴵŸ</td>
<td>y₄ᴵŸ</td>
<td>y₅ᴵŸ</td>
<td>y₆ᴵŸ</td>
<td>y₇ᴵŸ</td>
<td>y₈ᴵŸ</td>
</tr>
<tr>
<td>/F/</td>
<td>y₀ᶠ</td>
<td>y₁ᶠ</td>
<td>y₂ᶠ</td>
<td>y₃ᶠ</td>
<td>y₄ᶠ</td>
<td>y₅ᶠ</td>
<td>y₆ᶠ</td>
<td>y₇ᶠ</td>
<td>y₈ᶠ</td>
</tr>
<tr>
<td>/G/</td>
<td>y₀ᵍ</td>
<td>y₁ᵍ</td>
<td>y₂ᵍ</td>
<td>y₃ᵍ</td>
<td>y₄ᵍ</td>
<td>y₅ᵍ</td>
<td>y₆鹩</td>
<td>y₇ᵗ</td>
<td>y₈ᵗ</td>
</tr>
</tbody>
</table>
```

![Diagram showing the forward pass through a network with symbols and probabilities at each time step, including blanks.](image)
Overall training: Backward pass

- Foreach training instance
  - **Step 3**: Forward pass. Pass the training instance through the network and obtain all symbol probabilities at each time
  - **Step 4**: Construct the graph representing the specific symbol sequence in the instance. Use appropriate connections if blanks are included
**Overall training: Backward pass**

<table>
<thead>
<tr>
<th>/B/</th>
<th>$y_0^b$</th>
<th>$y_1^b$</th>
<th>$y_2^b$</th>
<th>$y_3^b$</th>
<th>$y_4^b$</th>
<th>$y_5^b$</th>
<th>$y_6^b$</th>
<th>$y_7^b$</th>
<th>$y_8^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$y_0^B$</td>
<td>$y_1^B$</td>
<td>$y_2^B$</td>
<td>$y_3^B$</td>
<td>$y_4^B$</td>
<td>$y_5^B$</td>
<td>$y_6^B$</td>
<td>$y_7^B$</td>
<td>$y_8^B$</td>
</tr>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/F/</td>
<td>$y_0^F$</td>
<td>$y_1^F$</td>
<td>$y_2^F$</td>
<td>$y_3^F$</td>
<td>$y_4^F$</td>
<td>$y_5^F$</td>
<td>$y_6^F$</td>
<td>$y_7^F$</td>
<td>$y_8^F$</td>
</tr>
<tr>
<td></td>
<td>$y_0^b$</td>
<td>$y_1^b$</td>
<td>$y_2^b$</td>
<td>$y_3^b$</td>
<td>$y_4^b$</td>
<td>$y_5^b$</td>
<td>$y_6^b$</td>
<td>$y_7^b$</td>
<td>$y_8^b$</td>
</tr>
</tbody>
</table>

- **Foreach training instance:**
  - **Step 5:** Perform the forward backward algorithm to compute $\alpha(t, r)$ and $\beta(t, r)$ at each time, for each row of nodes in the graph using the modified forward-backward equations.
  - **Step 6:** Compute derivative of divergence $\nabla_{Y_t} \text{DIV}$ for each $Y_t$. 

133
Overall training: Backward pass

• Foreach instance
  – **Step 6**: Compute derivative of divergence $\nabla_{Y_t} DIV$ for each $Y_t$
    \[
    \nabla_{Y_t} DIV = \begin{bmatrix}
    \frac{dDIV}{dy_t^1} & \frac{dDIV}{dy_t^2} & \cdots & \frac{dDIV}{dy_t^L}
    \end{bmatrix}
    \]
    \[
    \frac{dDIV}{dy_t^l} = - \sum_{r: S(r) = l} d \frac{d}{dy_t^{S(r)}} \left( \frac{\alpha(t, r) \beta(t, r)}{\sum_{r'} \alpha(t, r') \beta(t, r')} \log y_t^{S(r)} \right)
    \]

• **Step 7**: Aggregate derivatives over minibatch and update parameters
CTC: Connectionist Temporal Classification

• The overall framework we saw is referred to as CTC
  – Applies when “duplicating” labels at the output is considered acceptable, and when output sequence length < input sequence length
CTC caveats

• The “blank” structure (with concurrent modifications to the forward-backward equations) is only one way to deal with the problem of repeating symbols

• Possible variants:
  – Symbols partitioned into two or more sequential subunits
    • No blanks are required, since subunits must be visited in order
  – Symbol-specific blanks
    • Doubles the “vocabulary”
  – CTC can use bidirectional recurrent nets
    • And frequently does
  – Other variants possible..
Most common CTC applications

• Speech recognition
  – Speech in, phoneme sequence out
  – Speech in, character sequence (spelling out)

• Handwriting recognition
Speech recognition using Recurrent Nets

- Recurrent neural networks (with LSTMs) can be used to perform speech recognition
  - Input: Sequences of audio feature vectors
  - Output: Phonetic label of each vector

\[ P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5 \quad P_6 \quad P_7 \]

\[ X(t) \quad t=0 \]

Time
Speech recognition using Recurrent Nets

$W_1$ $W_2$

Input $X(t)$

$t=0$

Time

- Alternative: Directly output phoneme, character or word sequence
Next up: Attention models

• Will cover on Friday!