Aside: related topics and extensions

- The **Recurrent Neural Network Transducer** (RNN-T) extends CTC by allowing dependencies between b_t and $a_{1,\ldots,u-1}$
  - $table[u,t] = P(b_t = a_u | a_{1,\ldots,u-1}, x)(table[u-1,t-1] + table[u,t-1])$
  - Compute p_u with auto-regressive RNN, then combine with p_t
- Partial derivatives w.r.t. the loss $\mathcal{L}$ can be efficiently calculated with the forward_backward algorithm
  - monotonic_backward builds suffixes instead of prefixes
  - Multiplies $\frac{\delta P_{p_t}(b_t=a_u)}{\delta p_t}$ with prefix and suffix likelihoods
- Setting $b = x$ and a as a state sequence, forward_backward can be used to train a **GMM-HMM**
  - $P_u(b_t)\left(a_{u-1,u} table[u-1,t-1] + a_{u,u} table[u,t-1]\right)$
- More details can be found in the appendices

Everything past here is an ASIDE

(not on your exam)

Backpropagating the loss 1

- To learn $\mathcal{L} = -\log P_\theta(a)$ end-to-end, we need to **backprop**
- The parameters for frame t are p_t, and

$$\frac{\partial \mathcal{L}}{\partial p_t} = -\frac{1}{P_\theta(a)} \frac{\partial P_\theta(a)}{\partial p_t} = -\frac{1}{P_\theta(a)} \sum\nolimits_{b \in \mathcal{B}^{-1}(a;T)} \frac{\partial P_\theta(b|x)}{\partial p_t}$$

- For a single alignment b we have

$$\begin{aligned}
\frac{\partial P_\theta(b|x)}{\partial p_t} &= \frac{\partial \prod_{t'=1}^{T} P_{p_{t'}}(b_{t'})}{\partial p_t} \\
&= \left(\prod\nolimits_{t'=1}^{t-1} P_{p_{t'}}(b_{t'})\right)\left(\prod\nolimits_{t'=t+1}^{T} P_{p_{t'}}(b_{t'})\right) \frac{\partial P_{p_t}(b_t)}{\partial p_t} \\
&= P_\theta(b_{1,\ldots,t-1}|x) P_\theta(b_{t+1,\ldots,T}|x) \frac{\partial P_{p_t}(b_t)}{\partial p_t}
\end{aligned}$$

UNIVERSITY OF TORONTO

Backpropagating the loss 2

- Considering **all** paths $\mathcal{B}(b) = a$, we have

$$\frac{\partial \mathcal{L}}{\partial p_t} = -\frac{1}{P_\theta(a)} \sum\nolimits_{b \in \mathcal{B}^{-1}(a;T)} \frac{\partial P_\theta(b|x)}{\partial p_t}$$

$$= -\frac{1}{P_\theta(a)} \sum\nolimits_{b \in \mathcal{B}^{-1}(a;T)} P_\theta(b_{1,\ldots,t-1}|x) P_\theta(b_{t+1,\ldots,T}|x) \frac{\partial P_{p_t}(b_t)}{\partial p_t}$$

$$= -\frac{1}{P_\theta(a)} \sum_{u=1}^{U} \frac{\partial P_{p_t}(b_t = a_u)}{\partial p_t} \left(\sum\nolimits_{b_{1,\ldots,t-1} \in \mathcal{B}^{-1}(a_{1,\ldots,u-1};t-1)} P_\theta(b_{1,\ldots,t-1}|x) \right) \quad ①$$

$$\left(\sum\nolimits_{b_{t+1,\ldots,T} \in \mathcal{B}^{-1}(a_{u+1,\ldots,U};T-t-1)} P_\theta(b_{t+1,\ldots,T}|x) \right) \quad ②$$

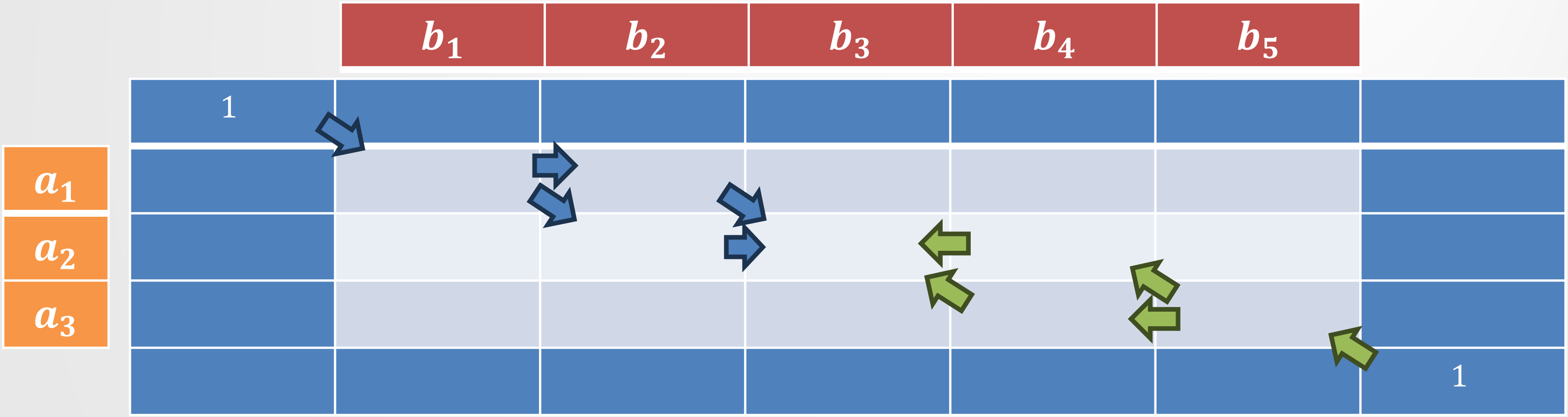
- ① is $table[u-1, t-1]$ in monotonic_forward
- ② is $table[u+1, t+1]$ in monotonic_backward

The monotonic_backward algorithm

```
Function monotonic_backward
Inputs a = a_1, a_2, ..., a_U  and  b = b_1, b_2, ..., b_T
1: Define table[1 ... U + 1, 1 ... T + 1]
2: initialize(table[1 ... U + 1, T + 1], table[U + 1, 1 ... T + 1])
3: For each u in U ... 1:
4:     For each t in T ... 1:
5:             table[u, t] =
                   step(a_u, b_t, table[u + 1, t + 1], table[u + 1, t], table[u, t + 1])
6: Return finalize(table[1 ... U + 1,1 ... T + 1])
```

Same as monotonic_forward, but with partial solutions over **suffixes**

The forward_backward algorithm



- Prefixes are computed with the monotonic_forward algorithm (⇨)
- Suffixes are computed with the monotonic_backward algorithm (⇦)
- $\frac{\partial}{\partial p_t}$ are summed over columns (a_u)

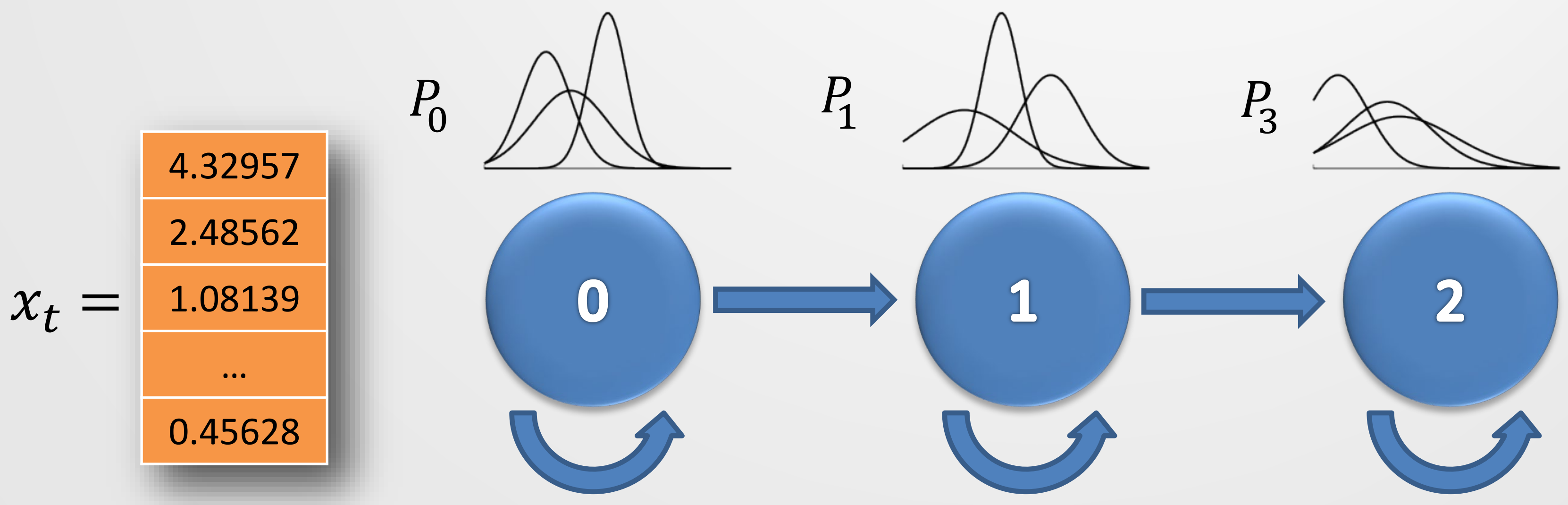
HMM-based ASR

- In GMM-based phone classification, $P(x_t|q_t)$ depends on which phone $q_t = s$ we assume x_t was drawn from
- We can model temporal dependencies across frames by specifying $P(q)$
  - $q^* = \mathrm{argmax}_q P(q)P(x|q)$ is a frame-level transcript
- Using a **Hidden Markov Model** (HMM) over GMM-based observation likelihoods, we have a GMM-HMM ASR system

$$P(q,x) = \underbrace{P(q_o)}_{\text{Prior state probability}} \prod_{t=1}^{T} \underbrace{P(q_t|q_{t-1})}_{\text{Transition probability}} \underbrace{P(x_t|q_t)}_{\text{Observation probability (GMM)}}$$

Continuous HMMs (CHMM)

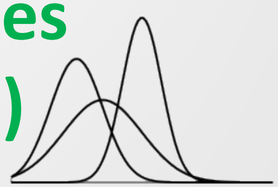
- A **continuous HMM** has observations that are distributed over continuous variables.
  - Observation probabilities, P_s, are also continuous.
  - E.g., here $b_s(x_t)$ tells us the probability of seeing the (multivariate) continuous observation x_t while in state s.



GMM-HMM as Continuous HMM

- Continuous HMMs are very similar to discrete HMMs.
  - $S = \{s_1, \dots, s_V\}$: set of states (e.g., phonemes)
  - $X = \mathbb{R}^d$: **continuous observation space**

$$\theta \left\{ \begin{array}{ll} \bullet\ \Pi = \{\pi_1, \dots, \pi_V\} & \text{: initial state probabilities} \\ \bullet\ A = \{a_{ij}\}, i, j \in S & \text{: state transition probabilities} \\ \bullet\ B = P_v(\vec{x}), i \in S, \vec{x} \in X & \text{: state output probabilities (i.e., Gaussian mixtures)} \end{array} \right.$$



yielding

- $Q = \{q_1, \dots, q_T\}, q_t \in S$: state sequence
- $\mathcal{O} = \{o_1, \dots, o_T\}, o_t \in X$: observation sequence

Adapting monotonic_forward

```
Function monotonic_forward
Inputs a = a_1, a_2, ..., a_U  and  b = b_1, b_2, ..., b_T
1: Define table[0 ... U, 0 ... T]
2: initialize(table[0 ... U, 0], table[0,1 ... T])
3: For each u in 1 ... U:
4:     For each t in 1 ... T:
5:         table[u, t] =
               step(a_u, b_t, table[u − 1, t − 1], table[u − 1, t], table[u, t − 1])
6: Return finalize(table[0 ... U, 0 ... T])
```

a is the state sequence, $b = x$ the speech features

initialize: set $table[0,0] = \pi_{q_0}$, $table[0,1 \ldots T] = table[1 \ldots U, 0] = 0$

step:$P_u(b_t)\big(a_{u-1,u} table[u-1, t-1] + a_{u,u} table[u, t-1]\big)$

finalize: $-\log table[U, T]$

(Though training usually involves EM, not backprop)