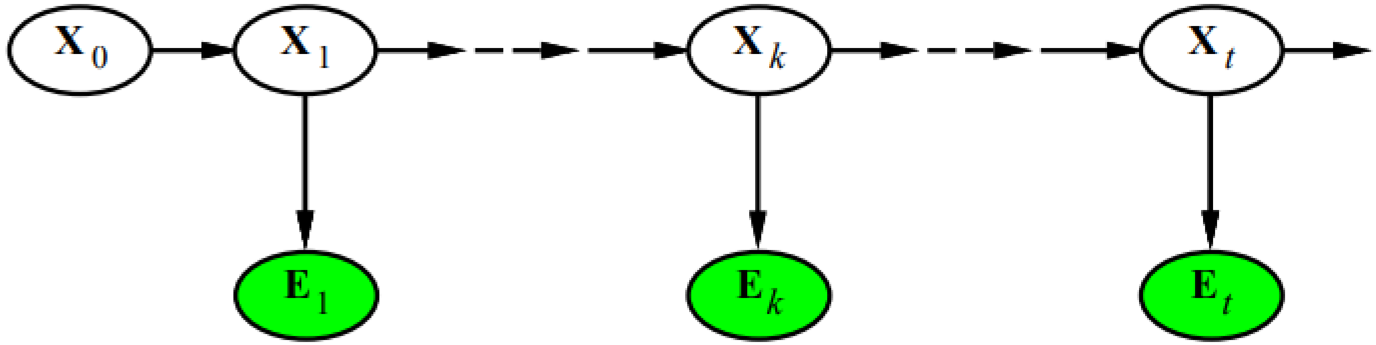
Chapter 15 (AIAMA)

Probabilistic Reasoning Over Time-02

Sukarna Barua
Assistant Professor, CSE, BUET

Smoothing

- Compute the probability distribution over past states.
- Compute $P(X_k|e_{1:t})$ where $0 \leq k < t$



Smoothing

- Compute $P(X_k|e_{1:t})$ where $0 \leq k < t$

$$\begin{aligned}\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\ &= \alpha\, \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{e}_{1:k}) \quad \text{(using Bayes' rule)} \\ &= \alpha\, \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) \quad \text{(using conditional independence)}\end{aligned}$$

- Remember bayes expansion: $P(a|b,c) = \frac{P(b|a,c)P(a|c)}{p(b|c)}$

- Here, $P(X_k|e_{k+1:t}, \mathrm{e}_{1:k}) = \frac{P(e_{k+1:t}|X_k, e_{1:k})P(X_k|e_{1:k})}{P(e_{k+1:t}|e_{1:k})}$

- $P(e_{k+1:t}|e_{1:k})$ is a fixed term α [can be obtained later as probabilities sum to 1]

Smoothing

- Compute $P(X_k|e_{1:t})$ where $0 \leq k < t$

$$\begin{aligned}\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}, \mathbf{e}_{k+1:t}) \\ &= \alpha\, \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{e}_{1:k}) \quad \text{(using Bayes' rule)} \\ &= \alpha\, \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) \quad \text{(using conditional independence)}\end{aligned}$$

- Hence, $P(X_k = x_i|e_{1:t}) = \alpha \mathbf{f}_k[i] \times P(e_{k+1:t}\,|X_k = x_i)$
  - Where $\mathbf{f}_k$ denotes forward probabilities [*filtering problem*]
  - Note that $\mathbf{f}_k$ is a vector [$\mathbf{f}_k[i] = P(X_k = x_i|e_{1:k})$]

Smoothing

- Now, how to compute $P(e_{k+1:t}|X_k)$?

$$\begin{aligned}
\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) &= \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \quad \text{(conditioning on } \mathbf{X}_{k+1}\text{)} \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t} \mid \mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \quad \text{(by conditional independence)} \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1}, \mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1} \mid \mathbf{x}_{k+1})P(\mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k)\ , \qquad (15.9)
\end{aligned}$$

- $P(e_{k+1}|x_{k+1})$ and $P(x_{k+1}|x_k)$ comes from the models [transition and sensor]
- $P(e_{k+2:t}|x_{k+1})$ can be obtained recursively or iteratively [*using dynamic programming*]

Smoothing

- **Recurrence relation for $\boldsymbol{P(e_{k+1:t}|X_k)}$:**

- Let $P(e_{k+1:t}|X_k) = \mathbf{b}_k$ [$\mathbf{b}_k$ *is a vector/array of probabilities*]

- From our previous derivation:
  - $\mathbf{b}_k[i] = P(e_{k+1:t}|X_k = x_i)$

$$= \sum_{x_j} P\big(e_{k+1}\big|X_{k+1} = x_j\big)P(X_{k+1} = x_j\,|X_k = x_i)\,\mathbf{b}_{k+1}[j]$$

$$= \sum_{x_j}(o_{j,l})(a_{ij})(\,\mathbf{b}_{k+1}[j])$$

[assume $e_{k+1} = e_l$ an specific output value at time step $k+1$]

Smoothing

- **Recurrence relation for $\boldsymbol{P(e_{k+1:t}|X_k)}$:**

  - $\mathbf{b}_k[i] = \sum_j (o_{j,l})(a_{ij})(\mathbf{b}_{k+1}[j])$

- What is the base condition?

  - To find the base condition, put $k = t$ [for the last/current state]
  - $\mathbf{b}_t[j] = P(e_{t+1:t}|X_t = x_j) = 1$ for all j [Why?]
    - Probability of occurring an empty sequence $e_{t+1:t}$ is 1

Smoothing

- **Recurrence relation for $\boldsymbol{P(e_{k+1:t}|X_k)}$:**

  $$\mathbf{b}_k[i] = \sum_j (o_{j,l})(a_{ij})(\,\mathbf{b}_{k+1}[j])$$

- $\mathbf{b}_k = P(e_{k+1:t}|X_k)$ is known as backward probabilities

  - The algorithm for computing $\mathbf{b}_k$ starts from the t-th state $[k = t]$ and go backward up to $k = 1$ [*Fill DP tables for* $k = t\ to\ 1$]
  - The algorithm is known as backward algorithm [*in contrast to forward algorithm*]

Smoothing

- **Finally smoothed probability** $P(X_k = x_i|e_{1:t})$:

$$P(X_k = x_i|e_{1:t}) = \alpha \times \mathbf{f}_k\,[i] \times \mathbf{b}_k[i]$$

- In vector form:

$$P(X_k|e_{1:t}) = \alpha \times \mathbf{f}_k \times \mathbf{b}_k \quad [\textit{point-wise multiplication}]$$

- Again α is normalizing constant.

Smoothing: Example

- Compute $P(R_1|u_1, u_2)$ [Probability of rain at time $t = 1$ given umbrella observations at time $t = 1$ and $t = 2$
- As per our previous formula: $\mathbf{P}(R_1 \mid u_1, u_2) = \alpha\, \mathbf{P}(R_1 \mid u_1)\, \mathbf{P}(u_2 \mid R_1)$
- First term: $P(R_1|u_1) =< 0.818, 0.182 >$ [*We already know from filtering example*]
- Second term: $P(u_2|R_1)$ needs recursive expansion [*previous slide*]

$$\begin{aligned}\mathbf{P}(u_2 \mid R_1) &= \sum_{r_2} P(u_2 \mid r_2) P(\ \mid r_2) \mathbf{P}(r_2 \mid R_1) \\ &= (0.9 \times 1 \times \langle 0.7, 0.3 \rangle) + (0.2 \times 1 \times \langle 0.3, 0.7 \rangle) = \langle 0.69, 0.41 \rangle \ .\end{aligned}$$

Smoothing: Example

- Second term $P(u_2|R_1)$ needs recursive expansion [previous slides]

$$\begin{aligned}\mathbf{P}(u_2 \mid R_1) &= \sum_{r_2} P(u_2 \mid r_2) P(\ \mid r_2) \mathbf{P}(r_2 \mid R_1) \\ &= (0.9 \times 1 \times \langle 0.7, 0.3 \rangle) + (0.2 \times 1 \times \langle 0.3, 0.7 \rangle) = \langle 0.69, 0.41 \rangle \ .\end{aligned}$$

- Finally, compute:

$$\mathbf{P}(R_1 \mid u_1, u_2) = \alpha\, \mathbf{P}(R_1 \mid u_1)\, \mathbf{P}(u_2 \mid R_1)$$

$$\mathbf{P}(R_1 \mid u_1, u_2) = \alpha\, \langle 0.818, 0.182 \rangle \times \langle 0.69, 0.41 \rangle \approx \ \langle 0.883, 0.117 \rangle \ .$$

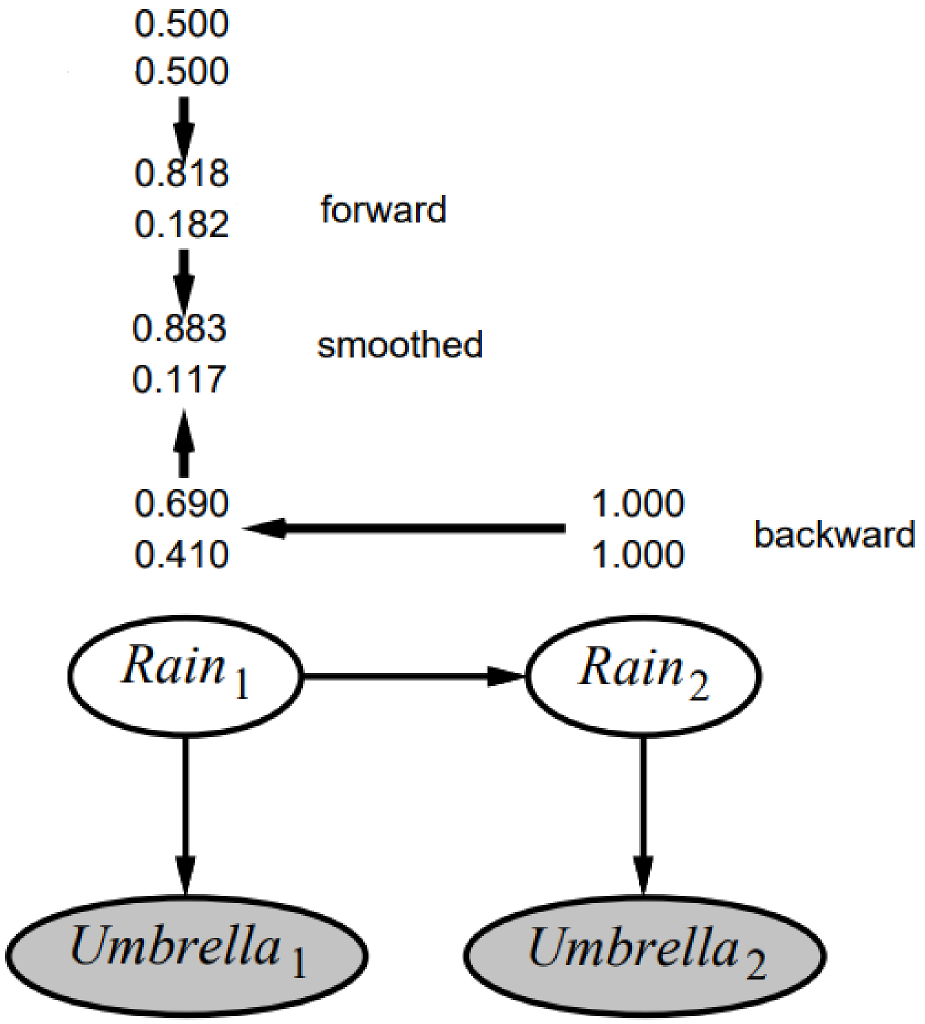
Smoothing: Example

- Smoothed estimate:

$$\mathbf{P}(R_1 \mid u_1, u_2) = \langle 0.883, 0.117 \rangle$$

- Filtered estimate:

$$\mathbf{P}(R_1 \mid u_1) = \langle 0.818, 0.182 \rangle$$



Smoothing: Example

- Smoothed estimate: $\mathbf{P}(R_1 \mid u_1, u_2) = \langle 0.883, 0.117 \rangle$
- Filtered estimate: $\mathbf{P}(R_1 \mid u_1) \;=\; \langle 0.818, 0.182 \rangle$
- Observation: Smoothed estimate for rain on day 1 is higher than the filtered estimate (0.818) [Why?]
  - This is because the umbrella on day 2 makes it more likely to have rained on day 2 which in turn, because rain tends to persist, that makes it more likely to have rained on day 1.

Smoothing: Algorithm Pseudocode

- Forward and backward algorithm can be combined to compute posterior probabilities in linear time

```
function FORWARD-BACKWARD(ev, prior) returns a vector of probability distributions
  inputs: ev, a vector of evidence values for steps 1, . . . , t
          prior, the prior distribution on the initial state, P(X_0)
  local variables: fv, a vector of forward messages for steps 0, . . . , t
                   b, a representation of the backward message, initially all 1s
                   sv, a vector of smoothed estimates for steps 1, . . . , t

  fv[0] ← prior
  for i = 1 to t do
     fv[i] ← FORWARD(fv[i − 1], ev[i])
  for i = t downto 1 do
     sv[i] ← NORMALIZE(fv[i] × b)
     b ← BACKWARD(b, ev[i])
  return sv
```

Most Likely Explanation

- Suppose that [true, true, false, true, true] is the umbrella sequence for the security guard's first five days on the job.
- What is the weather (state) sequence most likely to explain this?
- There are 2^5 possible weather sequences. Is there a way to find the most likely one?

Most Likely Explanation: Naïve Approach

- Compute the probability distribution $P(X_{1:t}|e_{1:t})$.

  - $P(X_{1:t}|e_{1:t}) = \alpha P(e_{1:t}|X_{1:t})P(X_{1:t})$ [Bayes theorem]

  $$= \alpha \prod_{i=1}^{t} P(e_i|X_i) \prod_{i=1}^{t} P(X_i|X_{i-1}) = \alpha \left(\prod_{i=1}^{t} o_{i,i}\right) \left(\prod_{i=1}^{t} a_{i,i-1}\right)$$

  - Assume $a_{1,o} = P(X_1|X_0) = P(X_1)$ for notational convenience
  - $P(X_1)$ is the probability distribution of states at time step $t = 1$

- After computing the probability distribution $P(X_{1:t}|e_{1:t})$
  - We know the probability of each state sequence $P(x_{1:t}|e_{1:t})$
  - We select the most probably state sequence as

$$argmax_{x_{1:t}} P(x_{1:t}|e_{1:t})$$

Most Likely Explanation: Naïve Approach

- We select the most probably state sequence as:

$$argmax_{x_{1:t}}\ P(x_{1:t}|e_{1:t})$$

- What is the problem with the naïve approach?

  - Number of state sequence is exponential
  - If number of states is N, then number of state sequences is N^t
  - Computing $P(x_{1:t}|e_{1:t})$ for N^t states requires exponential running time
  - Not feasible in real time
  - What can we do?
    - A better approach can be derived using dynamic programming

Most Likely Explanation

- Note that $argmax_{x_{1:t}}\, P(x_{1:t}|e_{1:t}) = argmax_{x_{1:t}}\, P(x_{1:t}, e_{1:t})$ [Why?]
- Consider the probability: $\beta_t[i] = \max_{x_{1:t-1}} P(X_t = x_i, x_{1:t-1}, e_{1:t})$
  - $\beta_t[i]$ is the probability of the most likely state sequence that produce observation sequences $e_{1:t}$ and ends at a specific state x_i at time step t.
  - We will show that instead of maximizing over all sequence of previous states, it is enough to maximize over only previous state [*we will derive a recurrence relation over the previous state*]

Most Likely Explanation

- **Recurrence relation for computing $\boldsymbol{\beta_t[i]}$**: we can compute $\beta_t[i]$ as follows:
  - For all possible states x_j at time step $t-1$:
    - Compute the probability of most likely state sequence that produce observation sequence $e_{1:t-1}$ and ends at $X_{t-1} = x_j$ (this probability is β_{t-1}[j])
    - Move from state x_j to x_i [with transition probability $a_{j,i}$]
    - Produce observation e_t at state x_i [with emission probability $o_{i,l}$ assuming $e_t = e_l$]
  - As we don't know the previous state x_j leading to the most likely sequence, we just take the maximum over all possible previous states x_j:

$$\beta_t[i] = \max_j [\, \beta_{t-1}[j] P(x_i | x_j) P(e_t | x_i)] = o_{i,l} \max_j \, [a_{j,i}\, \beta_{t-1}[j]]$$

Most Likely Explanation

- **Derivation of recurrence relation for $\boldsymbol{\beta_t[i]}$:**

$$\beta_t[i] = \max_{x_{1:t-1}} P(X_t = x_t, x_{1:t-1}, e_{1:t}) = \max_{x_{1:t-1}} P(X_t = x_t, x_{1:t-1}, e_{1:t-1}, e_t)$$

$$= \max_{x_{1:t-1}} P(e_t | X_t = x_i) P(X_t = x_i, x_{1:t-1}, e_{1:t-1})$$

$$= P(e_t | X_t = x_i) \max_{x_{1:t-1}} P(X_t = x_i | x_{t-1}) \; P(x_{1:t-1}, e_{1:t-1})$$

$$= P(e_t | X_t = x_i) \max_{X_{t-1} = x_j} P\big(X_t = x_i | \, X_{t-1} = x_j\big) \max_{x_{1:t-2}} P(x_{t-1} = x_j, x_{1:t-2}, e_{1:t-2})$$

$$= P(e_t | X_t = x_i) \max_j P\big(X_t = x_i | \, X_{t-1} = x_j\big) \, \beta_{t-1}[j]$$

$$= o_{i,l} \max_j (a_{j,i} \, \beta_{t-1}[j])$$

Most Likely Explanation

- Finally obtain the best possible state sequence:
  - $\beta_t[i]$ is the most likely state sequence ending at x_i at time step t
  - We don't know what is the end state for the most likely state sequence, hence we just take the maximum over all $\beta_t[i]s$:

$$argmax_{x_{1:t}} P(x_{1:t}, e_{1:t}) = argmax_i \beta_t[i]$$

- This algorithm is known as Viterbi algorithm
- Can be implemented using recursion or dynamic programming approach