CSC 580 Principles of Machine Learning

12 A closer look at PGMs; Hidden Markov Models

Chicheng Zhang

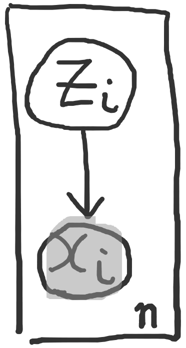
Department of Computer Science

THE UNIVERSITY OF ARIZONA

*slides credit: built upon CSC 580 Fall 2021 lecture slides by Kwang-Sung Jun

Background: A deeper look at conditional independence

- Recall the graphical representation (plate notation) specifies the dependency
- More precisely, it specifies how a joint distribution can be factored in *a structured way*
- Remark: We focus on directed graphical models (Bayes nets)
  - another world: undirected models

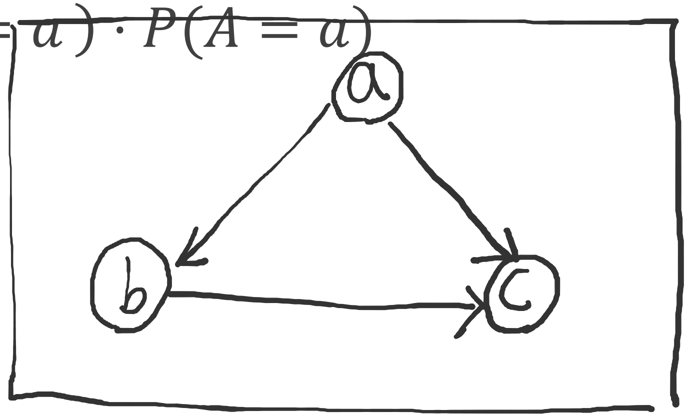


- Intro example:
  - $P(A = a, B = b, C = c) = P(\,C = c \mid A = a, B = b\,) \cdot P(A = a, B = b)$

$$= P(\,C = c \mid A = a, B = b\,) \cdot P(\,B = b \mid A = a\,) \cdot P(A = a)$$

  - Graphical representation:

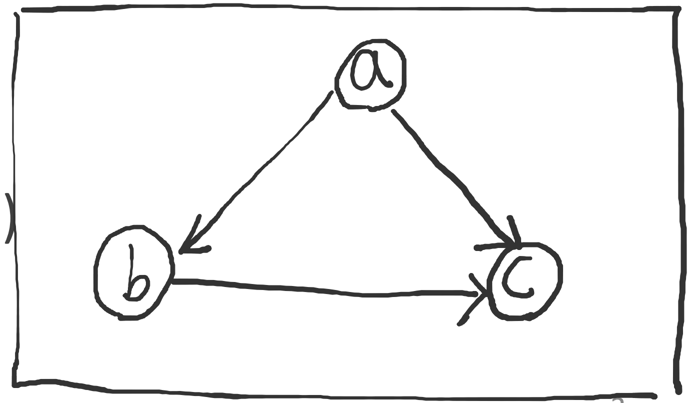
  For each conditional distribution, add direct links from *the nodes being conditioned* to *the node whose distribution is of interest*



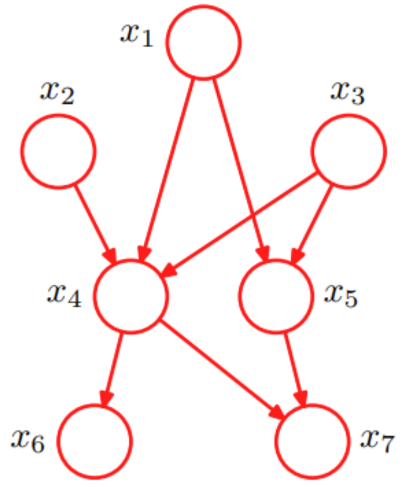
Warning: notation convention

- Notation easily gets overwhelming, no easy way out.
  - Fully-specified notation: explicit, but takes too long to process
  - Simplified notation: concise, but takes time to train yourself to be familiar
- Probabilistic models: For fully-specified notation, we always need to specify the random variable and the value that it takes separately.

- E.g. $P(A = a, B = b, C = c) = P(\, C = c \mid A = a, B = b \,) \cdot P(A = a, B = b)$
  $= P(\, C = c \mid A = a, B = b \,) \cdot P(\, B = b \mid A = a \,) \cdot P(A = a)$
- Simplified notation: $P(a, b, c) = P(\, c \mid a, b \,) \cdot P(a, b)$
  $= P(\, c \mid a, b \,) \cdot P(\, b \mid a \,) \cdot P(a)$
- i.e. reserve symbol a for values taken by random variable A (same for B, C)
- We will use simplified notation throughout this lecture



PGM: flexible modeling of data distributions



- Q: what kind of distribution does this graph represent?

- $P(x_1, x_2, \dots, x_7) = P(x_1)P(x_2)P(x_3)P(\, x_4 \mid x_1, x_2, x_3 \,) \cdot$
  $P(\, x_5 \mid x_1, x_3 \,)P(\, x_6 \mid x_4 \,)P(x_7 \mid x_4, x_5)$

- For a general directed acyclic graph (DAG) G with K nodes $x_1, \dots, x_K$,
  $$P(x_1, x_2, \dots, x_K) = \prod_{k=1}^{K} P(x_k \mid \mathrm{pa}_k),$$
  Parent nodes of x_k in G

- Fact: this implicitly implies $P(\, x_k \mid \mathrm{pa}_k \,) = P(x_k \mid x_1, \dots, x_{k-1})$, i.e. $x_k \perp\!\!\!\perp \{x_1, \dots, x_{k-1}\} \setminus \mathrm{pa}_k \mid \mathrm{pa}_k$
  - E.g. $x_6 \perp\!\!\!\perp \{x_1, x_2, x_3, x_5\} \mid x_4$
- Edges oftentimes encode *causal relationships* between the node variables

Bayes net = DAG + Conditional probability table

- $P(x_1, x_2, \dots, x_K) = \prod_{k=1}^{K} P(x_k \mid \mathrm{pa}_k)$ <- also need to specify each $P(x_k \mid \mathrm{pa}_k)$ respectively
- Aside: $J \perp\!\!\!\perp B, E \mid A$ => the effect of B, E to John's calling is "completely captured" in Alarm status

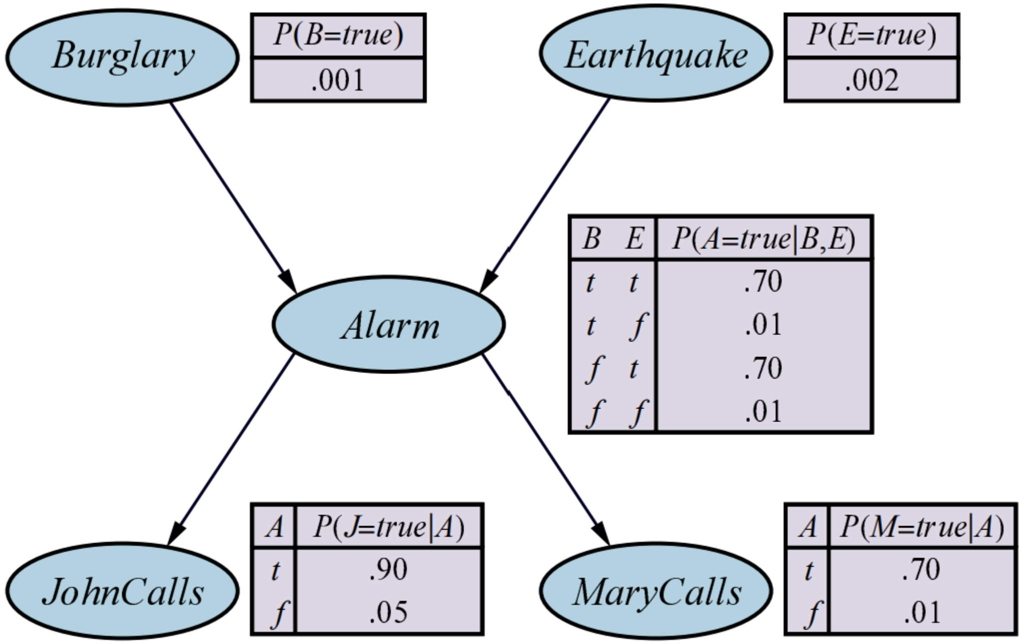


Figure 13.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

http://aima.cs.berkeley.edu/

PGM: parsimonious representation of distributions

- Suppose each $x_1, x_2, \ldots, x_K$ take binary values

- Naively representing $P(x_1, x_2, \ldots, x_K)$ requires 2^K entries

- With graphical model representation

$$P(x_1, x_2, \ldots, x_K) = \prod_{k=1}^{K} P(x_k \mid \mathrm{pa}_k)$$

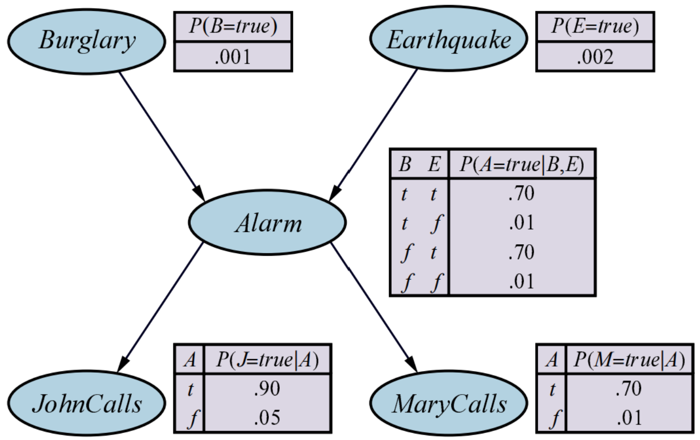


Figure 13.2 A typical Bayesian network, showing both the topology and the conditional probability tables (CPTs). In the CPTs, the letters B, E, A, J, and M stand for *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, and *MaryCalls*, respectively.

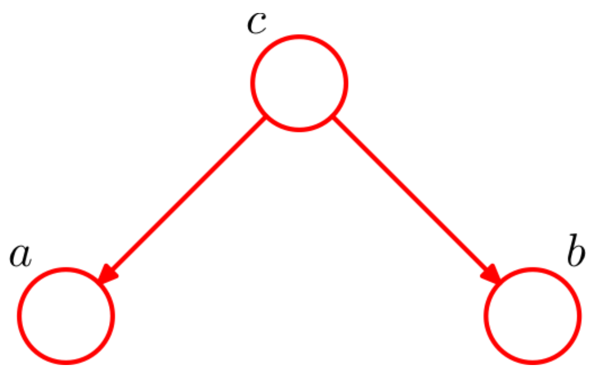
Each $P(x_k \mid \mathrm{pa}_k)$ takes $2^{|\mathrm{pa}_k|+1}$ entries

so total representation complexity $\le \sum_k 2^{|\mathrm{pa}_k|+1} \le 2^{O\left(\max_k |\mathrm{pa}_k|\right)}$

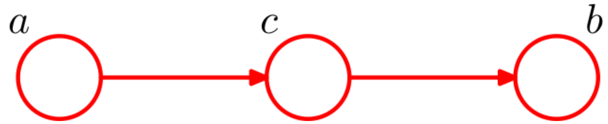
much smaller than 2^K if $\max_k |\mathrm{pa}_k| \ll K$ (we will see that this happens in many natural PGMs)

Three landmark examples

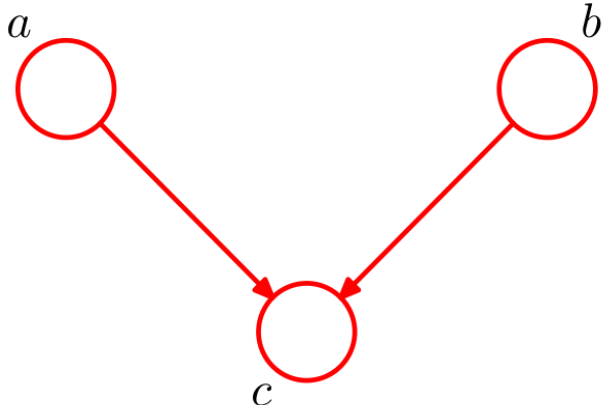
- tail-to-tail



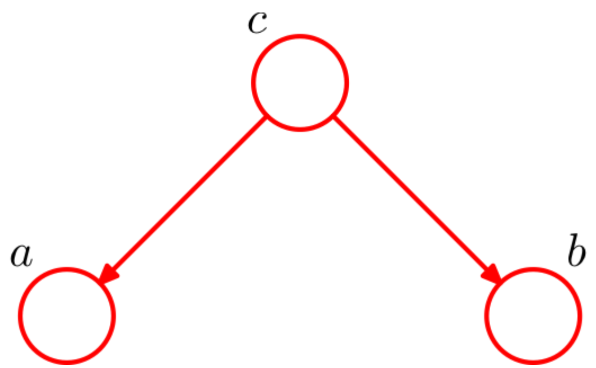
- Head-to-tail



- head-to-head

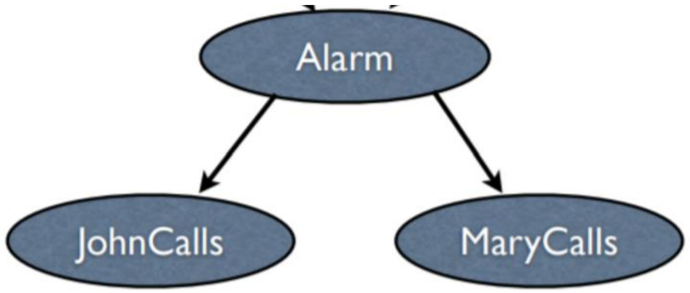


Ex 1: Tail-to-tail (common cause)



- $P(a,b,c) = P(c)P(\, a \mid c \,)P(b \mid c)$

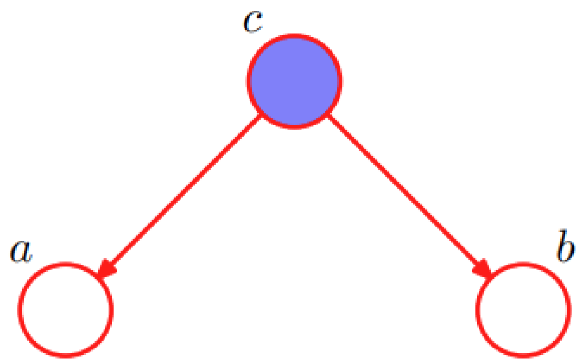
- $P(a,b) = \sum_c P(c)P(\, a \mid c \,)P(b \mid c)$ and in general it does not factorize



=> It is generally not true that $a \perp\!\!\!\perp b$

(e.g. John's calling is correlated with Mary's calling)

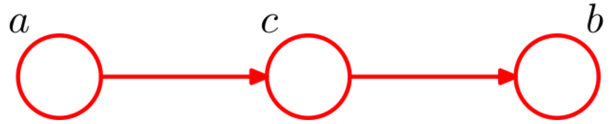
- However, $P(a,b \mid c) = \frac{P(a,b,c)}{P(c)} = P(\, a \mid c \,)P(b \mid c)$



=> $a \perp\!\!\!\perp b \mid c$

Ex 2: head-to-tail

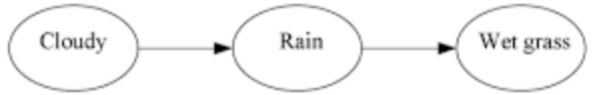
- $P(a, b, c) = P(a)P(c \mid a)P(b \mid c)$



- $P(a, b) = P(a)\sum_c P(c \mid a)P(b \mid c) = P(a) \cdot P(b \mid a)$

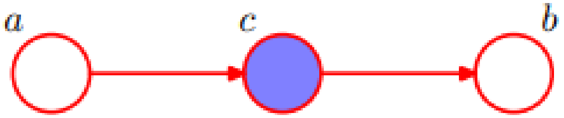
=> It is generally not true that $a \perp\!\!\!\perp b$

(e.g. "Cloudy" is correlated with "Wet grass")

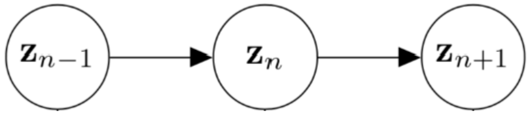


- However, $P(a, b \mid c) = \frac{P(a,b,c)}{P(c)} = \frac{P(a)P(c|a)P(b|c)}{P(c)} = P(a \mid c)P(b \mid c)$

=> $a \perp\!\!\!\perp b \mid c$

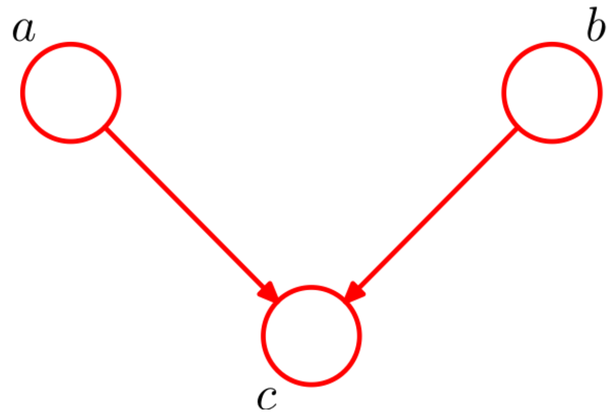


- Another important example: Markov chain (for time series data)



https://www.cs.odu.edu/~zeil/cs795ML/webcourse/Slides/graphical/handout.pdf

Ex 3: head-to-head (common effect)



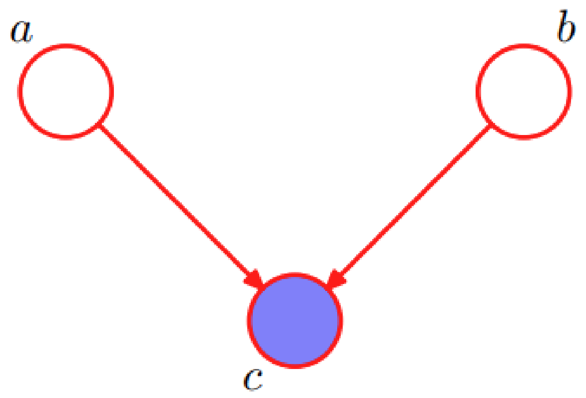
- $P(a, b, c) = P(a)P(b)P(c \mid a, b)$

- $P(a, b) = \sum_c P(a)P(b)P(c \mid a, b) = P(a)\,P(b)$

=> $a \perp\!\!\!\perp b$

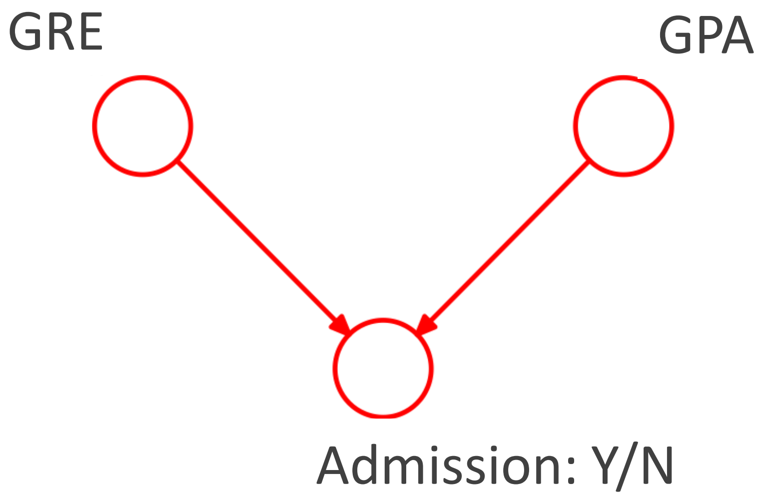
- However, $P(a, b \mid c) = \frac{P(a,b,c)}{P(c)} = \frac{P(a)P(b)P(c|a,b)}{P(c)}$ does not necessarily factorize

=> It is generally not true that $a \perp\!\!\!\perp b \mid c$

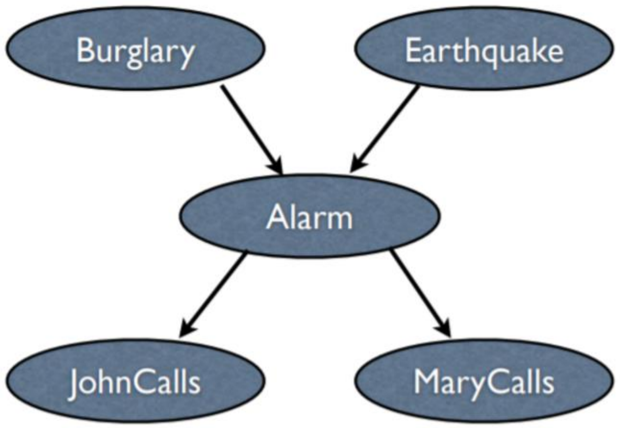


Ex 3: head-to-head (cont'd)

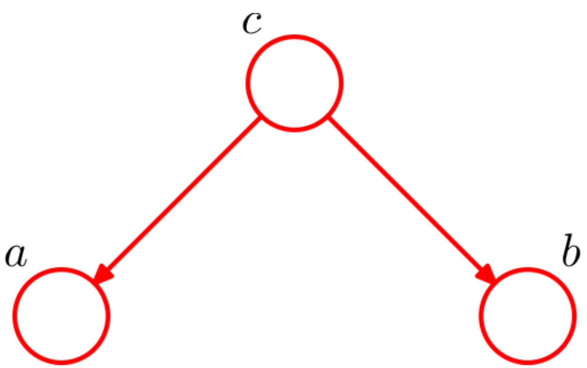
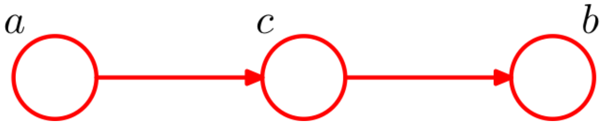
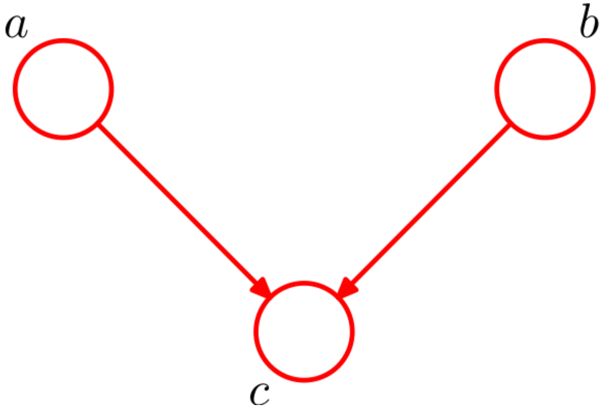
- If you pick an applicant randomly, the GRE and GPA is independent (according to our model)
- However, if you randomly pick an applicant who was accepted, then the low GRE may indicate that she had a high GPA.
    - Otherwise the student would have been rejected.
- This is called the **explain-away** phenomenon.



- Another example:
    - B and E are dependent, conditioned on A
    - It is also true that B and E are dependent, conditioned on *descendants of* A (e.g. J)



Summary

		$a \perp\!\!\!\perp b$?	$a \perp\!\!\!\perp b \mid c$?
• tail-to-tail		No	Yes
• Head-to-tail		No	Yes
• head-to-head		Yes	No

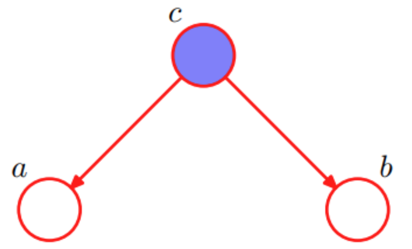
Next lecture (10/31)

- Markov models; Hidden Markov models (HMMs)

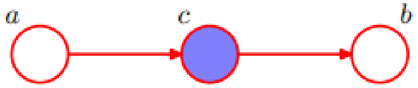
- Assigned reading: Prof. Jason Pacheco’s PGM slides: https://www2.cs.arizona.edu/~pachecoj/courses/csc535_fall20/lectures/pgms.pdf

- Additional reading: Bishop, “Pattern Recognition and Machine Learning”, Section 8.1-8.2

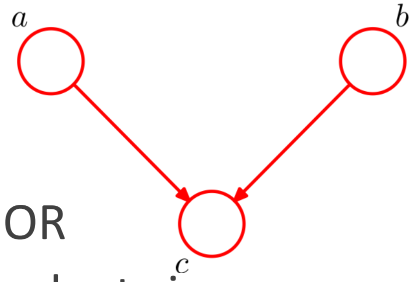
D-separation



- Systematic Rules for determining conditional independence given a directed acyclic graph.
- Answer questions of the form: Is $a \perp\!\!\!\perp b \mid c$ true or false ?



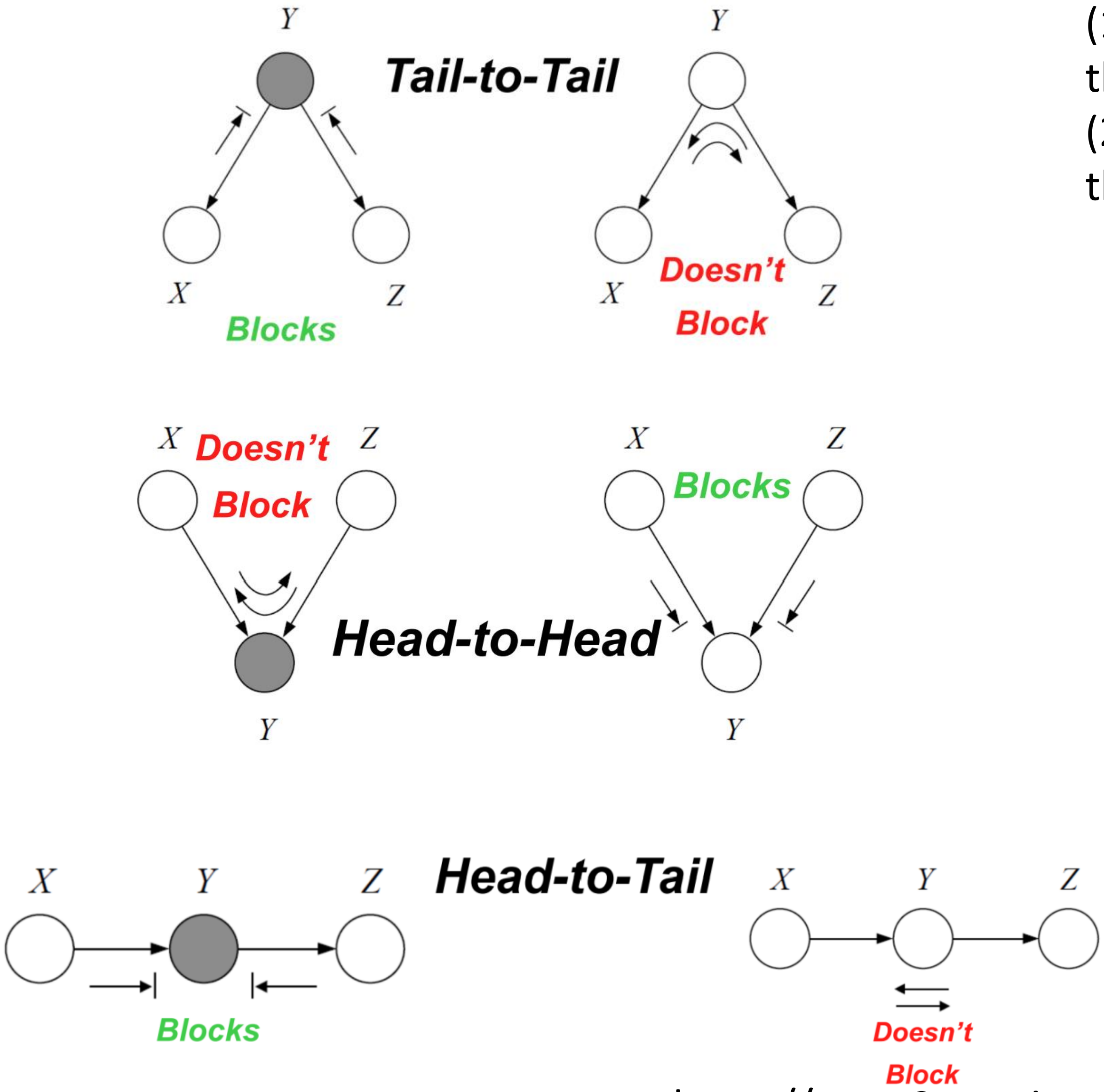
- [Def] b is a **descendent** of a if there exists a directed path from a to b.
  - => a is a descendent of a by definition.
- [Def] An undirected path p from a to b is **blocked given** c if it includes a node:
  - (a) the arrows on p meet either head-to-tail or tail-to-tail at the node, and the node is c, OR
  - (b) the arrows meet head-to-head at the node, and neither the node nor any of its descendants is c



  "Conditioned on c being observed, information can flow from a to b through path p"

- [Def] (D-separation)
  a is **d-separated from** b **given** c if every undirected path between a and b is blocked given c.
- [Thm] If a is **d-separated from** b **given** c, then $a \perp\!\!\!\perp b \mid c$.

Blockage: pictorial illustration



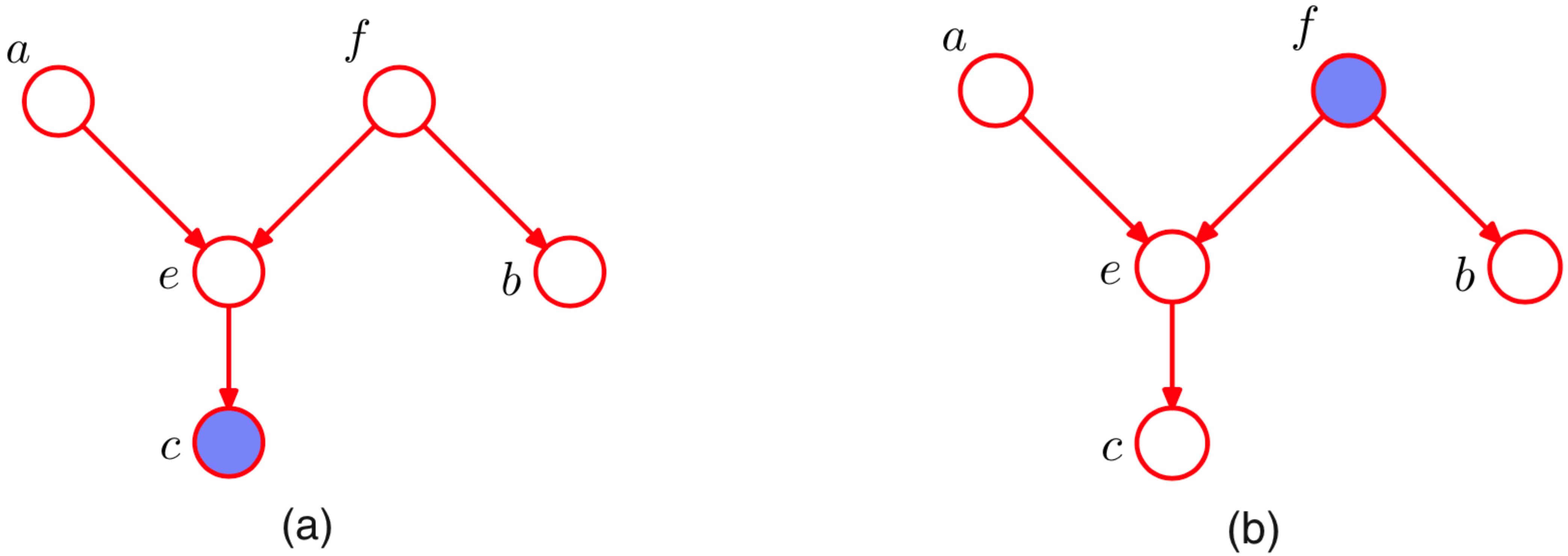
An <u>undirected path</u> p is *blocked* given c if it includes a node:
(1) the arrows on p meet either head-to-tail or tail-to-tail at the node, and the node is c, or
(2) the arrows meet head-to-head at the node, and neither the node nor any of its descendant is c

https://www2.cs.arizona.edu/~pachecoj/courses/csc535_fall20/lectures/pgms.pdf

D-separation examples

An <u>undirected path</u> p is *blocked* given c if it includes a node:
(1) the arrows on p meet either head-to-tail or tail-to-tail at the node, and the node is c, or
(2) the arrows meet head-to-head at the node, and neither the node nor any of its descendant is c

- a to b has only one path $p = a - e - f - b$
- In (a): Is $a \perp\!\!\!\perp b \mid c$? No, p is not blocked given c
- In (b): Is $a \perp\!\!\!\perp b \mid f$? Yes, p is blocked given f
- Is $a \perp\!\!\!\perp b$?



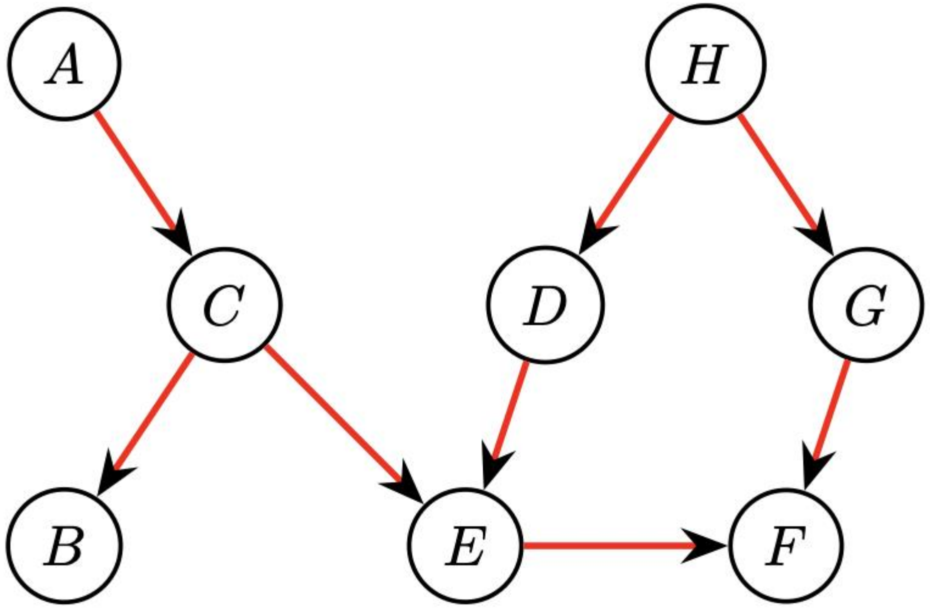
D-separation: general definition for node sets

- Q: Is A $\perp\!\!\!\perp$ B | C true or false ?
  - Each of A,B,C is a **set** of random variables

- [Def] An undirected path p from a to b is **blocked given** C if it includes a node:
  - (a) the arrows on p meet either head-to-tail or tail-to-tail at the node, and the node is in C
  - (b) the arrows meet head-to-head at the node, and neither the node nor any of its descendants is in C

- [Def] (D-separation)
  A is **d-separated from** B **given** C if every undirected path between $a \in A$ and $b \in B$ is blocked given C.
- [Thm] If A is **d-separated from** B **given** C, then $A \perp\!\!\!\perp B \mid C$.

D-separation: an exercise



- Is $G \perp\!\!\!\perp A$ - equivalently, $G \perp\!\!\!\perp A \mid \emptyset$?
- Yes, G-H-D-E-C-A is blocked by E
  G-F-E-C-A is blocked by F

- Is $E \perp\!\!\!\perp H \mid \{D, G\}$?
- Yes, E-D-H is blocked by D; E-F-G-H is blocked by F (or G)

- Is $E \perp\!\!\!\perp H \mid \{C, D, F\}$?
- No, although E-D-H is blocked by D, E-F-G-H is not blocked

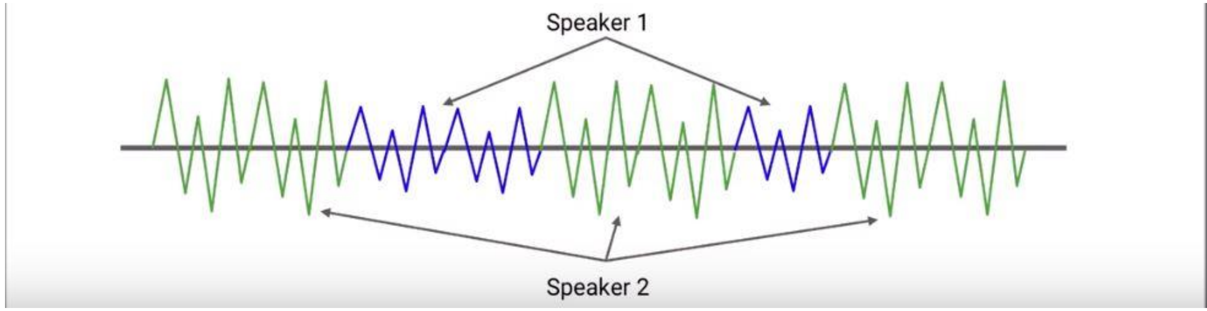
Sequential data

- So far, we have dealt with IID data: $z_i \sim \mathcal{D}$
- What if the data has dependency between z_i and z_j?
  - E.g., sequentially generated: $z_i \mid z_{1:i-1} \sim D(\Theta = f(z_{1:i-1}))$
  - Notation: $x_{1:n}$ means $x_1, .., x_n$

- Examples:
  - Spoken language: $z_{1:n}$, $z_t \in [W]$: word index
    - What word you say depends on what you just said; 'context'.
  - Human movement, say soccer: $z_{1:n}$=> video (sequence of pictures, say every 1/24 second)
    - It's a video, so the dependency is natural (e.g., you cannot teleport).
  - Biology: amino acid/protein sequence.

Guiding example: Speaker diarization

- You have recorded a meeting happened with K people. Can we segment it according to speakers' identities?

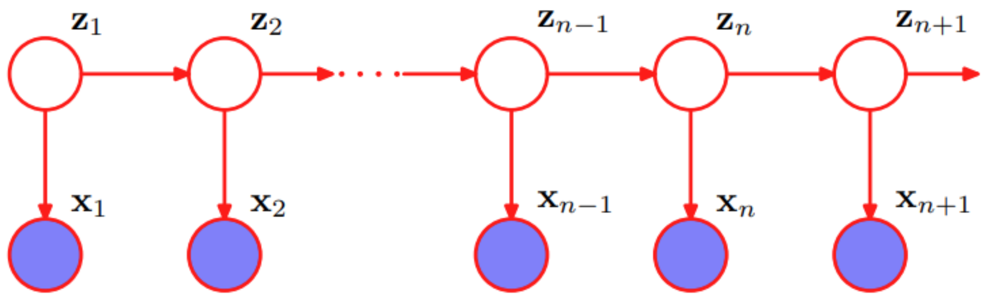


- Data: audio sequence $x_{1:n}$,
  - $x_t \in \mathbb{R}^d$ auditory features during a short time interval (e.g., 100ms).
- Goal: Segment the audio with contiguous blocks where each block is assigned a speaker index.
  - i.e., infer $z_t \in [K]$ indicating who was speaking at time point t.
  - Let's call z_t a **state**.

- Key characteristic: sequential dependency!
  - stickiness: if you spoke at time t, you are likely to be speaking at time t+1.
  - transition: there are more frequent transition pairs than other pairs. (the boss keeps correcting a newbie)

Hidden Markov model (HMM)

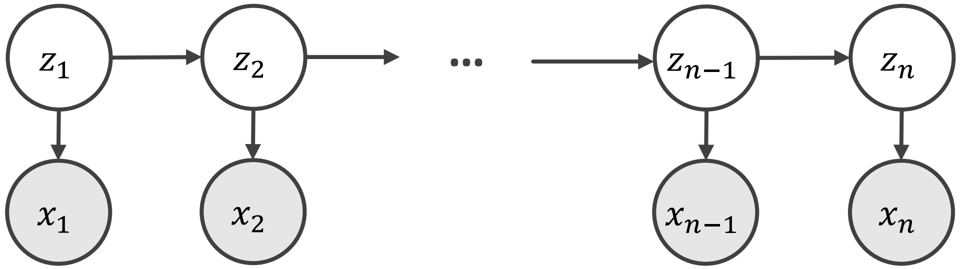
- Graphical representation:



- The key characteristic: Markovian assumption
  - Only model the first-order dependency
  - Possible to add the dependency up to τ past z_t's, but remember the bias-variance tradeoff.
  - Further, the computational complexity.
  - Productive mindset: try a simple model, and fix it only if it does not work.

- In fact, we can work with M sequences: *observations* $\{x_{m,t}\}_{m\in[M],t\in[N]}$
  - *hidden states* $\{z_{m,t}\}_{m\in[M],t\in[N]}$ unobserved
  - N can even be different for each m.
  - But we will mainly work with the case of M=1.

HMM – generative story

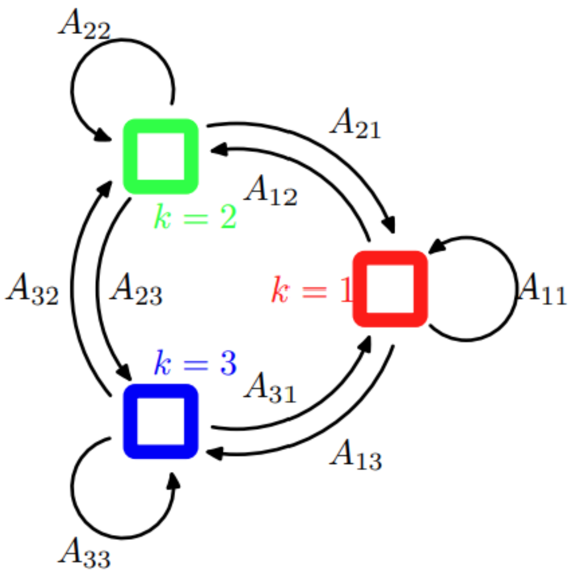


- The joint distribution over (observations, hidden states)

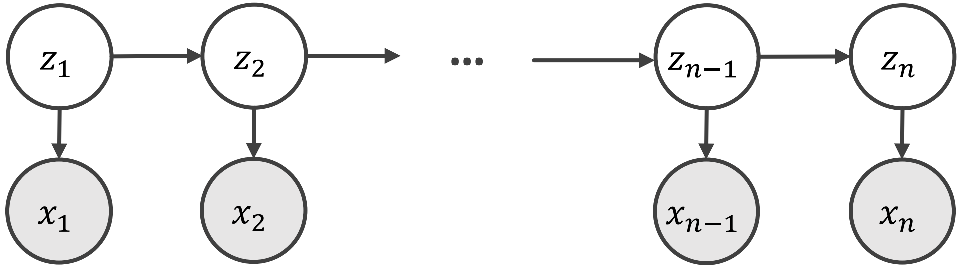
$$P(x_{1:n}, z_{1:n}) = P(z_1) \cdot \prod_{i=2}^{n} P(\, z_i \mid z_{i-1}\,) \cdot \prod_{i=1}^{n} P(x_i \mid z_i)$$

- Corresponding generative story:
  - $z_1 \sim \text{Categorical}(\pi)$
  - For $i = 2, \ldots, n$:
    - Draw $z_i \sim \text{Categorical}(A_{z_{i-1}})$
  - For $i = 1, 2, \ldots, n$:
    - Draw $x_i \sim P_{\phi_{z_i}}(\cdot)$
    - e.g. $P_\phi = \text{Categorical}(\phi)$, or $P_\phi = N(\phi, I)$

$$A = \begin{pmatrix} - A_1 - \\ \cdots \\ - A_K - \end{pmatrix}, \phi = \begin{pmatrix} - \phi_1 - \\ \cdots \\ - \phi_K - \end{pmatrix}$$

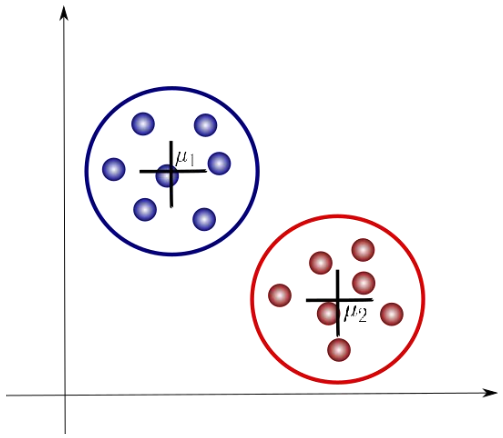


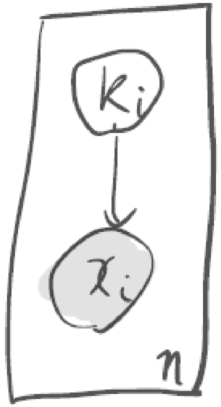
HMM model specification



- Model parameters Θ is composed of:
  - Initial distribution π
  - Transition probability A
  - Emission distribution parameter ϕ
- Likelihood: $P(x_{1:n}, z_{1:n}; \Theta) = P(z_1; \pi) \cdot \prod_{i=2}^{n} P(z_i \mid z_{i-1}; A) \cdot \prod_{i=1}^{n} P(x_i \mid z_i; \phi)$
  - Marginal likelihood $P(x_{1:n}; \Theta) = \sum_{z_{1:n}} P(x_{1:n}, z_{1:n}; \Theta)$

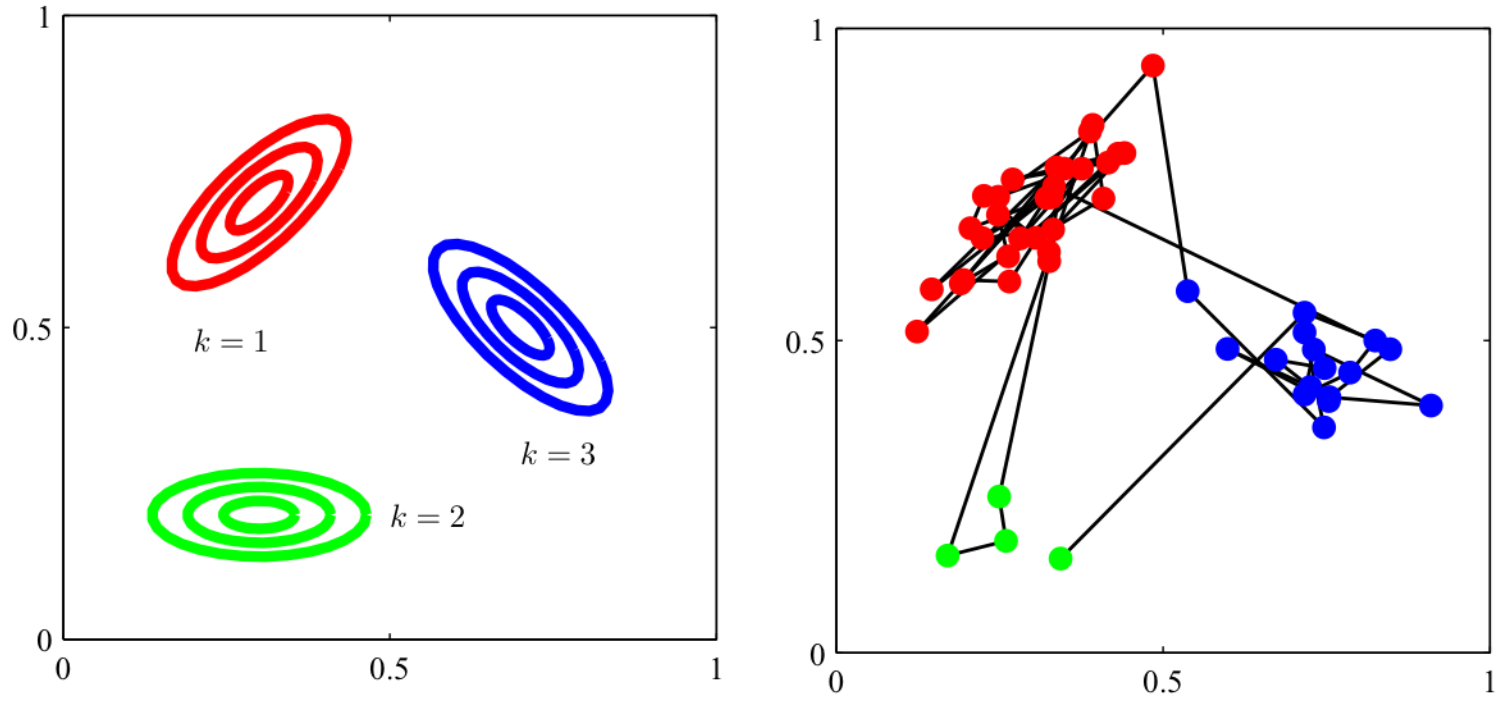
- Comparison to GMM
  - z_i's has the same role as k_i's
  - HMM allows *temporal dependence* of hidden states
  - HMM's emission distribution is not necessarily Gaussian





HMM example

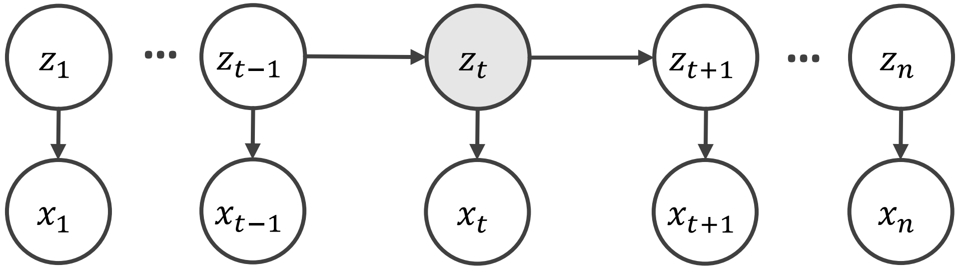
Gaussian emission model $P(\, x \mid z = k; \phi\,) = P_{\phi_k}(x) = N(\mu_k, \Sigma_k)$



Transition probability $A = \begin{pmatrix} .9 & .05 & .05 \\ .05 & .9 & .05 \\ .05 & .05 & .9 \end{pmatrix}$

HMM: key conditional independence structure

- Claim: conditioned on z_t, the following three groups of r.v.'s,
  $(x, z)_{1:t-1}, x_t, (x, z)_{t+1:n}$, are independent



- How to show A, B, C are independent?
  - One way: show A $\perp\!\!\!\perp$ B and C $\perp\!\!\!\perp$ (A,B)

- Checking conditional independence by d-separation:
  $(x, z)_{1:t-1} \perp\!\!\!\perp x_t \mid z_t$
  $(x, z)_{1:t-1}, x_t \perp\!\!\!\perp (x, z)_{t+1:n} \mid z_t$

- Consequences: e.g. $P(x_t \mid z_t, x_{1:t-1}) = P(x_t \mid z_t)$, $P(z_{t+1} \mid z_t, x_{1:t-1}) = P(z_{t+1} \mid z_t)$

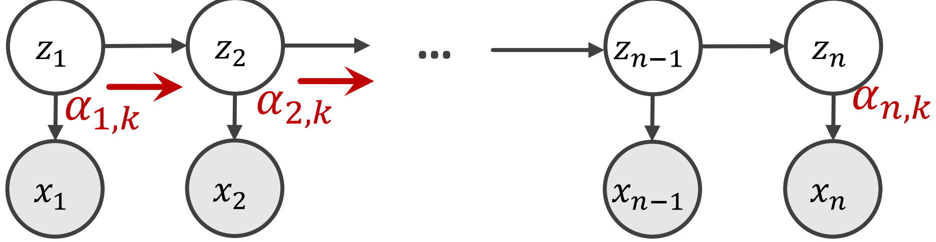
Main tasks for HMM

- Task 1 [inference]: Given an HMM and the observation $x_{1:n}$, how likely is it to observe the given sequence? What is the posterior distribution of z_t for each t?
  - $p(x_{1:n})$ => used for checking convergence, comparing various models, model selection, etc.
  - $p(z_t = k \mid x_{1:n}), \forall t$

- Task 2 [inference – "decoding"]: Given an HMM and the observation $x_{1:n}$, what is the most likely hidden state sequence?
  - i.e., $z_{1:n}^* = \arg\max_{z_{1:n}} p(z_{1:n} \mid x_{1:n})$
  - This gives you the ultimate answer to our speaker diarization task.

- Task 3 [learning]: Given the observation $x_{1:n}$, learn the HMM parameters.

Task 1: inference



- Naively, calculating $P(x_{1:n}) = \sum_{z_{1:n}} P(x_{1:n}, z_{1:n})$ takes time exponential in n
  - Can we do better?
- Key observation: can use *dynamic programming* to save computation
  - Subproblem: compute $P(x_{1:t})$ for every t?
- A slightly different subproblem

$$\begin{aligned}
\underbrace{P(x_{1:t}, z_t = k)}_{\alpha_{t,k}} &= P_{\phi_k}(x_t) \cdot P(x_{1:t-1}, z_t = k) \\
&= P_{\phi_k}(x_t) \cdot \sum_j P(x_{1:t-1}, z_{t-1} = j, z_t = k) \\
&= P_{\phi_k}(x_t) \cdot \sum_j P(z_t = k \mid x_{1:t-1}, z_{t-1} = j) \cdot P(x_{1:t-1}, z_{t-1} = j) \\
&= P_{\phi_k}(x_t) \cdot \sum_j A_{jk} \cdot \underbrace{P(x_{1:t-1}, z_{t-1} = j)}_{\alpha_{t-1,j}}
\end{aligned}$$

- Initial condition: $\alpha_{1,k} = P(x_1, z_1 = k) = P_{\phi_k}(x_1) \cdot \pi_k$
- Time complexity for computing all $\{\alpha_{t,k}\}$: $O(n\,K^2)$ – *forward algorithm*

Next lecture (11/2)

- Inference in HMMs; Learning in HMMs: Expectation-Maximization

- Assigned reading: Prof. Jason Pacheco's slides on Dynamic Systems: https://www2.cs.arizona.edu/~pachecoj/courses/csc535_fall20/lectures/dynamicalsys.pdf

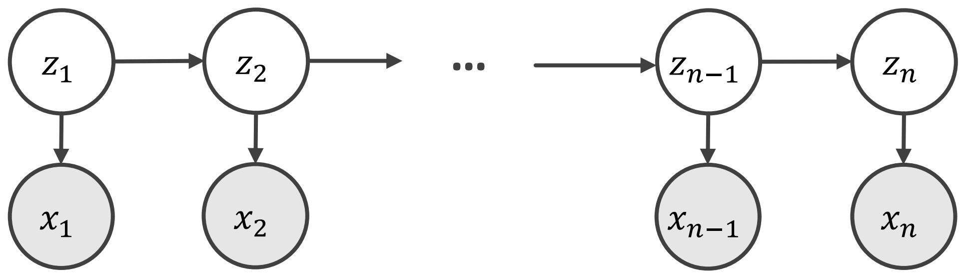
- HW3 will be released soon

Announcements

- HW3 is up (due 11/16)

- Please review my feedback on your project proposals

Task 1: inference (cont'd)



- How to compute $p(z_t = k \mid x_{1:n}), \forall t$?
- It suffices to compute $p(z_t = k, x_{1:n})$ for all k

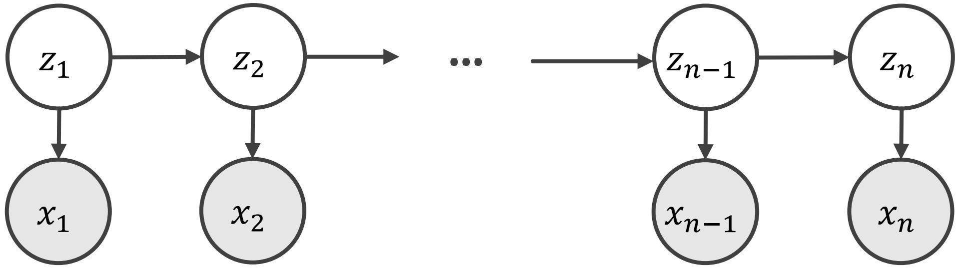
  => $p(z_t = k \mid x_{1:n}) = \frac{p(z_t=k, x_{1:n})}{p(x_{1:n})} = \frac{p(z_t=k, x_{1:n})}{\sum_j p(z_t=j, x_{1:n})}$

- Forward algorithm gives us: $\alpha_{t,k} = P(x_{1:t}, z_t = k)$
- Key observation: $x_{1:t} \perp\!\!\!\perp x_{t+1:n} \mid z_t$

  $$\Rightarrow p(z_t = k, x_{1:n}) = p(z_t = k, x_{1:t}) \cdot p(x_{t+1:n} \mid z_t = k, x_{1:t})$$
  $$= \alpha_{t,k} \cdot p(x_{t+1:n} \mid z_t = k)$$

- Define $\beta_{t,k} := p(x_{t+1:n} \mid z_t = k)$. Can we compute it efficiently?

Task 1: inference (cont'd)

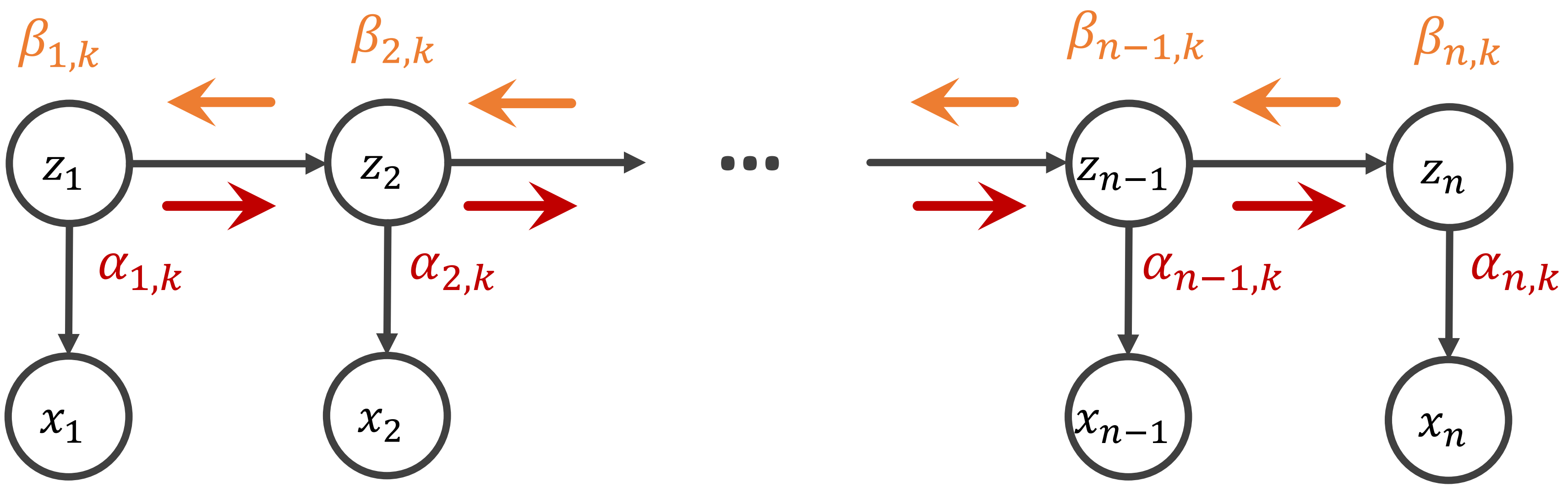


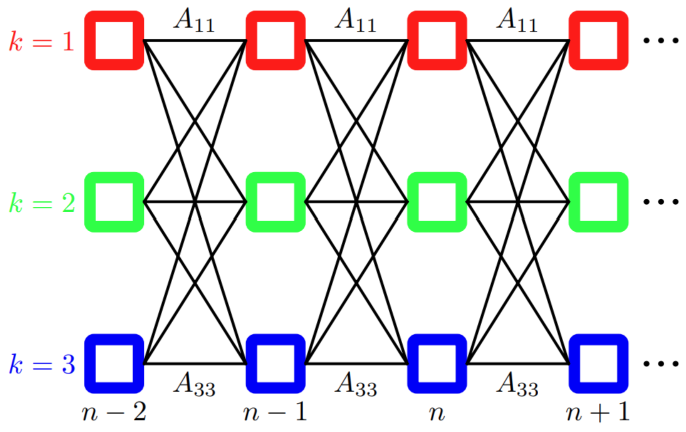
- $\beta_{t,k} \coloneqq P(x_{t+1:n} \mid z_t = k)$
- Can also compute it using dynamic programming
- Observe: $\beta_{n,k} = 1$
- Claim: $\beta_{t,k} = \sum_{j=1}^{K} A_{kj}\, P_{\phi_j}(x_{t+1})\beta_{t+1,j}$
- Proof:

$$
\begin{aligned}
P(x_{t+1:n} \mid z_t = k) &= \textstyle\sum_j P(x_{t+1:n}, z_{t+1} = j \mid z_t = k) \\
&= \textstyle\sum_j P(z_{t+1} = j \mid z_t = k)\, P(x_{t+1:n} \mid z_{t+1} = j, z_t = k) \\
&= \textstyle\sum_j P(z_{t+1} = j \mid z_t = k)\, P(x_{t+1:n} \mid z_{t+1} = j) \\
&= \textstyle\sum_j P(z_{t+1} = j \mid z_t = k)\, P_{\phi_j}(x_{t+1}) \cdot P(x_{t+2:n} \mid z_{t+1} = j)
\end{aligned}
$$

- This is the *backward algorithm* -- Time complexity for computing all $\{\beta_{t,k}\}$?

Forward-Backward algorithm - summary





Forward message: $\alpha_{t,k} = P_{\phi_k}(x_t) \cdot \sum_j A_{jk} \cdot \alpha_{t-1,j}$

Backward message: $\beta_{t,k} = \sum_j A_{kj} \cdot P_{\phi_j}(x_{t+1}) \cdot \beta_{t+1,j}$

Main tasks for HMM

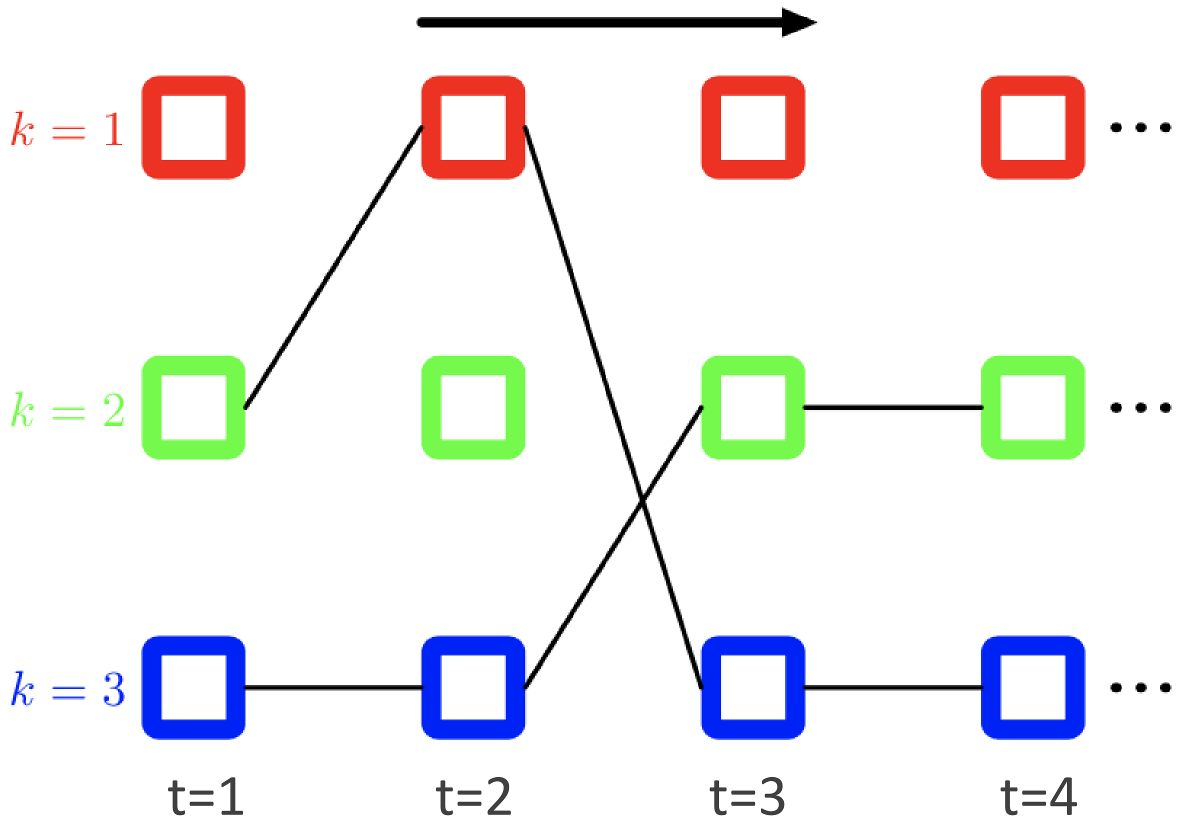
- Task 1 [inference]: Given an HMM and the observation $x_{1:n}$, how likely is it to observe the given sequence? What is the posterior distribution of z_t for each t?
  - $p(x_{1:n})$ => used for checking convergence, comparing various models, model selection, etc.
  - $p(z_t = k \mid x_{1:n}), \forall t$

- Task 2 [inference – "decoding"]: Given an HMM and the observation $x_{1:n}$, what is the most likely hidden state sequence?
  - i.e., $z_{1:n}^* = \arg\max_{z_{1:n}} p(z_{1:n} \mid x_{1:n})$
  - This gives you the ultimate answer to our speaker diarization task.

- Task 3 [learning]: Given the observation $x_{1:n}$, learn the HMM parameters.

Task 2: Most probable hidden state sequence

- Conceptually, a very simple problem: $\hat{z}_{1:n} = \arg\max_{z_{1:n}} p(z_{1:n} \mid x_{1:n})$
- But, similar to naively calculating $P(x_{1:n})$, naïve implementation has exponential time complexity!
- Fortunately, the conditional independence structure of HMM admits an efficient computation!



Viterbi’s algorithm (1967)

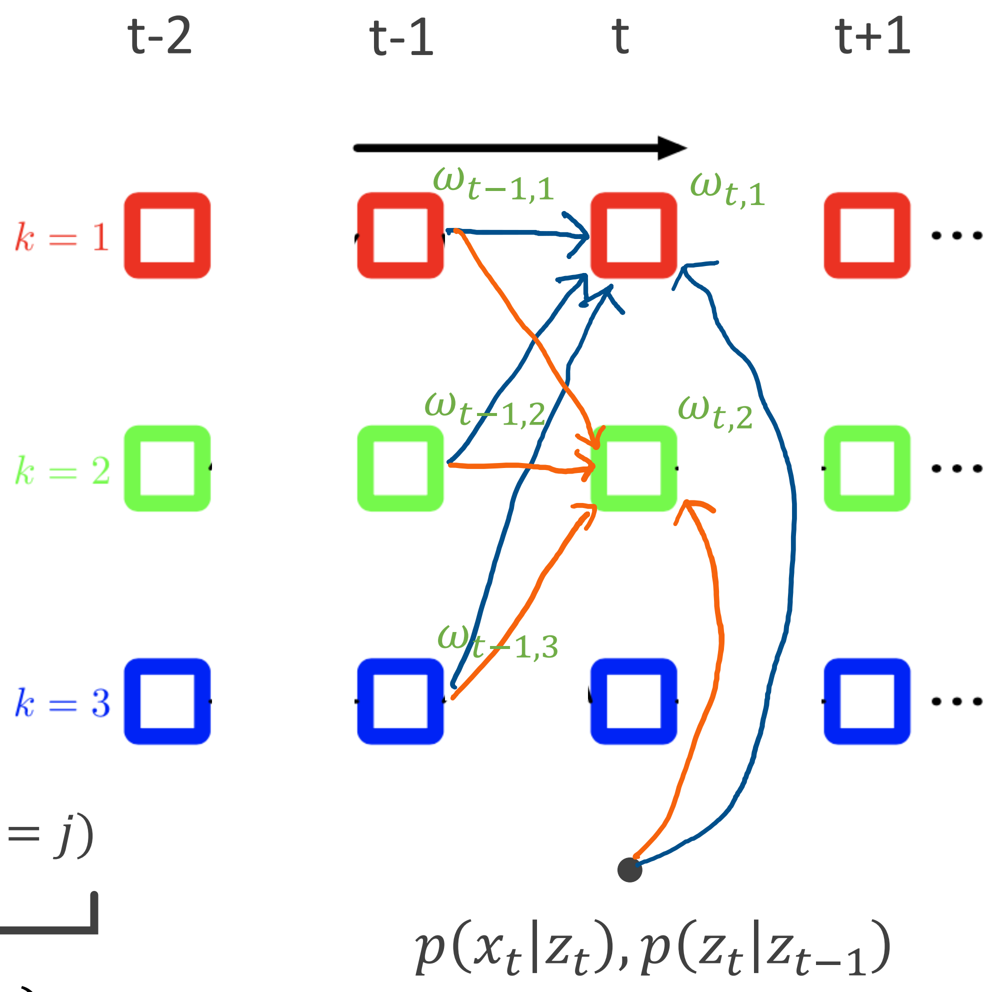
- $\hat{z}_{1:n} = \arg\max_{z_{1:n}} P(z_{1:n} \mid x_{1:n}) = \arg\max_{z_{1:n}} P(z_{1:n}, x_{1:n})$
- $\omega_{t,k} \coloneqq \max_{z_{1:t-1}} P(x_{1:t}, z_{1:t-1}, z_t = k)$ for all $k \in [K]$
  - Analogue of ``forward variables'' $\alpha_{t,k} = P(x_{1:t}, z_t = k)$

- Why are $\omega_{t,k}$’s useful?
  - E.g. optimal $\hat{z}_n = \text{argmax}_k\, \omega_{n,k}$

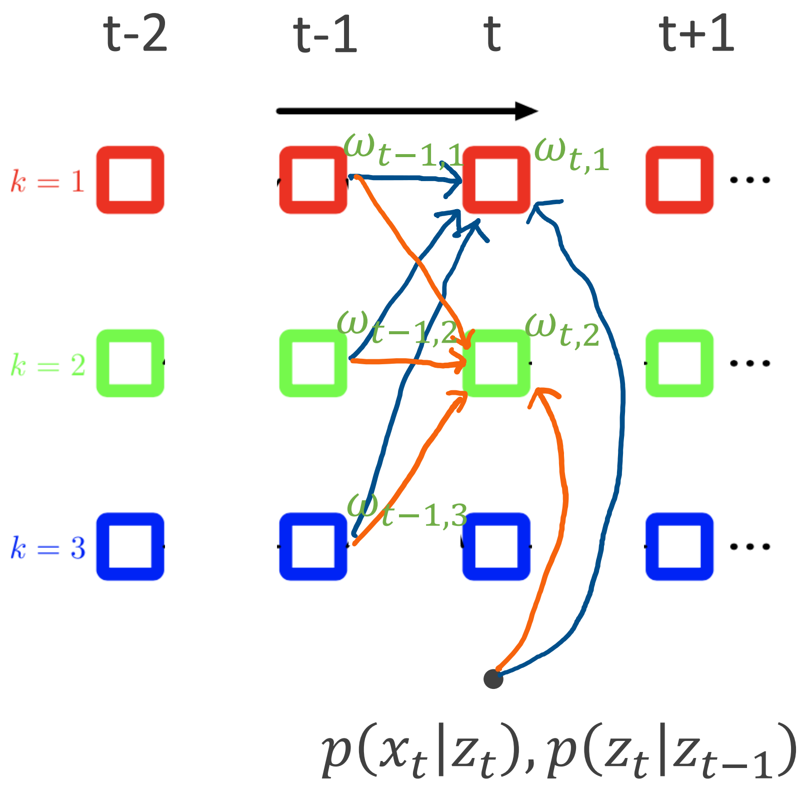
- How to compute $\omega_{t,k}$’s for every $t \in [n]$?
- Claim: $\omega_{t,k} = P_{\phi_k}(x_t) \max_j A_{jk}\, \omega_{t-1,j}$
- Proof: $\omega_{t,k} = \max_{z_{1:t-1}} P(x_{1:t}, z_{1:t-1}, z_t = k)$

$$= \max_j \max_{z_{1:t-2}} P(x_{1:t-1}, z_{1:t-2}, z_{t-1} = j, x_t, z_t = k)$$

$$= \max_j \underbrace{\max_{z_{1:t-2}} P(x_{1:t-1}, z_{1:t-2}, z_{t-1} = j)}_{\omega_{t-1,j}} \underbrace{P(x_t, z_t = k \mid z_{t-1} = j)}_{A_{jk} P_{\phi_k}(x_t)}$$



Viterbi's algorithm (cont'd)



- Suppose we would like to recover $\hat{z}_t$ for every t
- Observe: $\hat{z}_{1:t} = \underset{z_{1:t}}{\operatorname{argmax}} P(z_{1:t}, \hat{z}_{t+1:n}, x_{1:n})$

  Therefore,
  - For $t = n$, $\hat{z}_n = \underset{z_n}{\operatorname{argmax}} \left(\max_{z_{1:n-1}} P(z_{1:n-1}, z_n, x_{1:n}) \right) = \underset{j}{\operatorname{argmax}}(\omega_{n,j})$
  - For $t \le n-1$:

$$\begin{aligned}
\hat{z}_t &= \underset{z_t}{\operatorname{argmax}} \left(\max_{z_{1:t-1}} P(z_{1:t-1}, z_t, \hat{z}_{t+1:n}, x_{1:n}) \right) \\
&= \underset{z_t}{\operatorname{argmax}} \left(\max_{z_{1:t-1}} P(z_{1:t-1}, x_{1:t}, z_t) \cdot \; P(\hat{z}_{t+1} \mid z_t)\, P(x_{t+1:n} \mid \hat{z}_{t+1}) \right) \\
&= \underset{j}{\operatorname{argmax}} \left(\max_{z_{1:t-1}} P(z_{1:t-1}, x_{1:t}, z_t = j) \cdot P(\hat{z}_{t+1} \mid z_t = j) \right) \\
&= \underset{j}{\operatorname{argmax}} \left(\omega_{t,j} \cdot A_{j,\hat{z}_{t+1}} \right)
\end{aligned}$$

This is exactly the optimal j in the definition of $\omega_{t+1,k} = P_{\phi_k}(x_{t+1}) \max_j A_{jk}\, \omega_{t,j}$ for $k = \hat{z}_{t+1}$

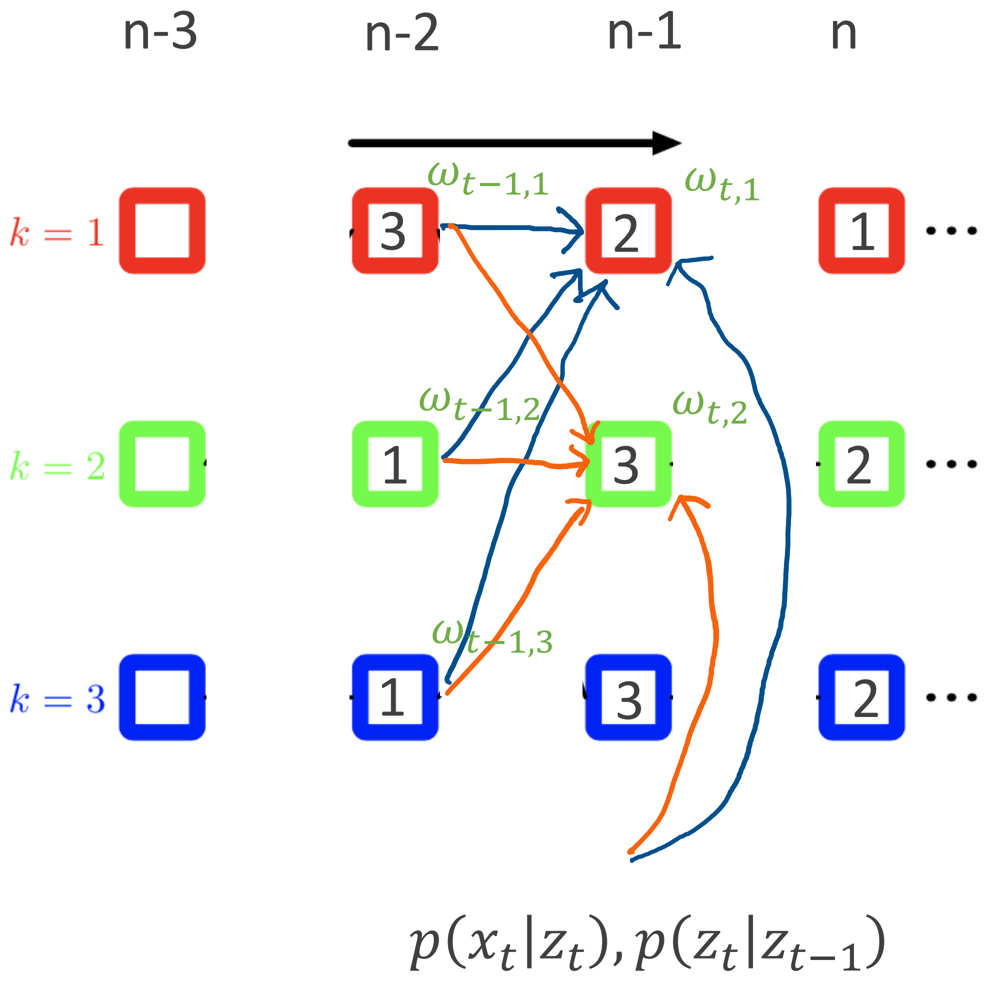
Backtracking

- Suppose $\hat{z}_n = 3$

- The entries in each cell (t, k) is the index j of the cell in the previous time step that induces optimal joint probability $\max_{z_{1:t-1}} P(x_{1:t}, z_{1:t-1}, z_t = k)$:

$$\omega_{t,k} = P(x_t | z_t = k) \max_j A_{jk}\, \omega_{t-1,j}$$

- $\hat{z}_n = 3$ => $\hat{z}_{n-1} = 2$ => $\hat{z}_{n-2} = 3$ => $\hat{z}_{n-3} = 1$



Implementation caveats

- When implementing the algorithm, working with probabilities can lead to numerical instabilities.
  - We could even get $\omega_{t,k} = 0$ in computers when $\omega_{t,k}$ becomes very small => this is common when the sequence length is >= 100.
- Recommendation: always work in the log domain
  - E.g., do not compute $\omega_{t,k}$; compute $\ln \omega_{t,k}$
  - For stable computation of forward-backward algorithm, see (PRML, Bishop, 2006, Sect. 13.2.4)

Main tasks for HMM

- Task 1 [inference]: Given an HMM and the observation $x_{1:n}$, how likely is it to observe the given sequence? What is the posterior distribution of z_t for each t?
  - $p(x_{1:n})$ => used for checking convergence, comparing various models, model selection, etc.
  - $p(z_t = k \mid x_{1:n}), \forall t$

- Task 2 [inference – "decoding"]: Given an HMM and the observation $x_{1:n}$, what is the most likely hidden state sequence?
  - i.e., $z_{1:n}^* = \arg\max_{z_{1:n}} p(z_{1:n} \mid x_{1:n})$
  - This gives you the ultimate answer to our speaker diarization task.

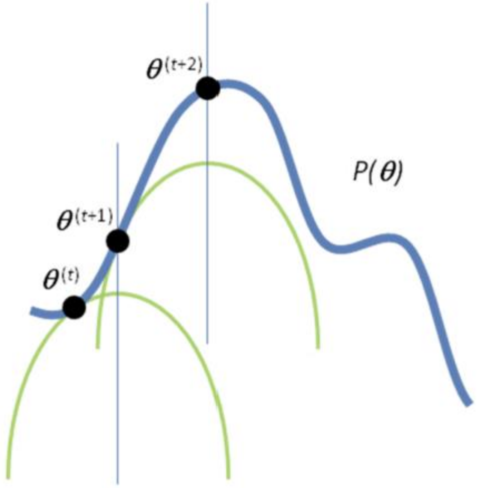
- Task 3 [learning]: Given the observation $x_{1:n}$, learn the HMM parameters.

Task 3: learning HMMs

- Naively, maximizing likelihood $P(x_{1:n}; \Theta) = \sum_{z_{1:n}} P(x_{1:n}, z_{1:n}; \Theta)$ is tricky
  - Recall the MLE issues for GMMs

- Can we design a tractable algorithm for learning HMMs using the EM framework?

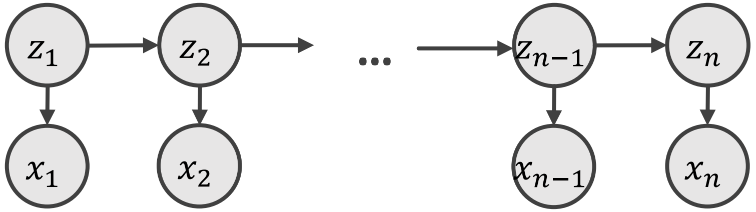
- Recall the EM algorithm:
  - Repeat:
    - E-step: calculate $P\left(z_{1:n} \mid x_{1:n}; \Theta^{(t)}\right)$
    - M-step: $\Theta^{(t+1)} \leftarrow \operatorname{argmax}_{\Theta} \sum_{z_{1:n}} P\left(z_{1:n} \mid x_{1:n}; \Theta^{(t)}\right) \ln P(x_{1:n}, z_{1:n}; \Theta)$



https://haipeng-luo.net/courses/CSCI567/2021_fall/lec8.pdf

Learning HMMs with the EM algorithm

- Warmup: what is the MLE for HMM with observation $x_{1:n}$ and hidden states $z_{1:n}$?



- Likelihood: $\ln P(x_{1:n}, z_{1:n}; \Theta)$

$$= \ln P(z_1; \pi) + \sum_{i=2}^{n} \ln P(\, z_i \mid z_{i-1}; A\,) + \sum_{i=1}^{n} \ln P(x_i \mid z_i; \phi)$$

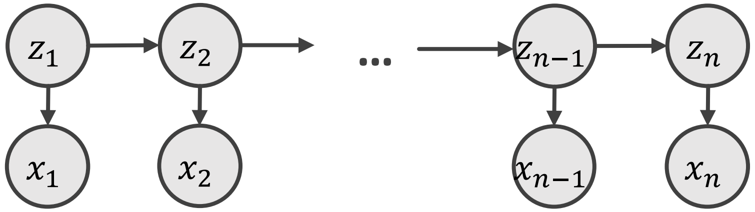
$$= \sum_k I(z_1 = k) \ln \pi_k + \sum_j \sum_k \sum_i I(z_{i-1} = j, z_i = k) \ln A_{jk} + \sum_k \sum_i I(z_i = k) \ln P_{\phi_k}(x_i)$$

- Each part can be maximized individually wrt π, A_j's and ϕ_k's

- Part 1: maximize $\ln P(z_1; \pi) = \sum_k I(z_1 = k) \ln \pi_k$
  => $\hat{\pi}_k = I(z_1 = k)$

- Part 2(j): maximize $\sum_k (\sum_i I(z_{i-1} = j, z_i = k)) \ln A_{jk}$ s.t. $A_j \in \Delta^{K-1}$
  => $\hat{A}_{j,k} = \frac{\#\{i: z_{i-1}=j,\, z_i=k\}}{\#\{i: z_{i-1}=j\}}$

Learning HMMs with the EM algorithm (cont'd)



- Part 3 (k): maximize $\sum_i I(z_i = k) \ln P_{\phi_k}(x_i)$
    - Optimal ϕ_k Depends on the emission model
    - E.g. $P_\phi(x) = \mathrm{Categorical}(\phi)$ => $\phi_{k,l} = \frac{\#\{i: x_i = l,\, z_i = k\}}{\#\{i: z_i = k\}}$
    - E.g. $P_\phi(x) = N(\phi, I)$ => $\phi_k = \frac{\sum_{i: z_i = k} x_i}{\#\{i: z_i = k\}}$

- Summary – MLE with fully observed data:
    - $\hat{\pi}_k =$ (empirical frequency of $z_1 = k$)
    - $\hat{A}_{j,k} =$ (empirical frequency of $z_i = k$ given $z_{i-1} = j$)
    - $\hat{\phi}_k =$ (MLE of $P_\phi(x)$ over $\{(x_i, z_i): z_i = k\}$)

Learning HMMs with the EM algorithm (cont'd)

- Using EM algorithm for MLE with observation $x_{1:n}$ alone
- Given parameter in previous iteration $\Theta^{(\mathrm{old})}$, what does the M-step look like?

- Intuition: the M-step performs MLE on a weighted collection of *augmented sequences* $(x_{1:n}, z_{1:n})$, each with weight (multiplicity) $P\left(z_{1:n} \mid x_{1:n}; \Theta^{(\mathrm{old})}\right)$
- Mental picture: $x_{1:n}$ induces K^n fully-observable sequences $(x_{1:n}, z_{1:n})$ => compute MLE on this giant weighted dataset

- $\hat{\pi}_k =$ (**weighted** empirical frequency of $z_1 = k$) = $P\left(z_1 = k \mid x_{1:n}; \Theta^{(\mathrm{old})}\right)$
- $\hat{A}_{j,k} =$ (**weighted** empirical frequency of $z_i = k$ given $z_{i-1} = j$) = $\frac{\sum_i P\left(z_{i-1}=j, z_i=k \mid x_{1:n}; \Theta^{(old)}\right)}{\sum_i P\left(z_{i-1}=j \mid x_{1:n}; \Theta^{(old)}\right)}$
- $\hat{\phi}_k =$ (**weighted** MLE of $P_\phi(x)$ over $\{(x_i, z_i) \colon z_i = k\}$) = $\mathrm{argmax}_\phi \sum_i P\left(z_i = k \mid x_{1:n}; \Theta^{(old)}\right) \ln P_\phi(x_i)$

Learning HMMs with the EM algorithm (cont'd)

- Formal derivation of M-step:

$$\text{maximize}_{\Theta}\ Q\left(\Theta; \Theta^{(old)}\right) = \sum_{z_{1:n}} P\left(z_{1:n} \mid x_{1:n}; \Theta^{(\text{old})}\right) \underbrace{\ln P(x_{1:n}, z_{1:n}; \Theta)}_{\sum_k I(z_1 = k) \ln \pi_k + \sum_j \sum_k \sum_i I(z_{i-1} = j, z_i = k) \ln A_{jk} + \sum_k \sum_i I(z_i = k) \ln P_{\phi_k}(x_i)}$$

- Equivalent to:

$$\begin{aligned} \text{maximize}_{\Theta} \quad & \textstyle\sum_k P(z_1 = k \mid x_{1:n}; \Theta^{(\text{old})}) \ln \pi_k \\ & + \textstyle\sum_j \sum_k \sum_i P\left(z_{i-1} = j, z_i = k \mid x_{1:n}; \Theta^{(\text{old})}\right) \ln A_{jk} \\ & + \textstyle\sum_k \sum_i P\left(z_i = k \mid x_{1:n}; \Theta^{(\text{old})}\right) \ln P_{\phi_k}(x_i) \end{aligned}$$

- Again, each part can be maximized individually wrt π, A_j's and ϕ_k's

Learning HMMs with the EM algorithm (cont'd)

- The M-step requires access to the posterior distributions

of (pairs of) hidden states at different time steps

$$\hat{\pi}_k = P\left(z_1 = k \mid x_{1:n}; \Theta^{(\text{old})}\right)$$

$$\hat{A}_{j,k} = \frac{\sum_i P\left(z_{i-1}=j, z_i=k \mid x_{1:n}; \Theta^{(old)}\right)}{\sum_i P\left(z_{i-1}=j \mid x_{1:n}; \Theta^{(old)}\right)}$$

$$\hat{\phi}_k = \text{argmax}_\phi \sum_i P\left(z_i = k \mid x_{1:n}; \Theta^{(old)}\right) \ln P_\phi(x_i)$$

- How to obtain them?

- Recall: the forward-backward algorithm can be used to give us $\gamma_{i,k} := P\left(z_i = k \mid x_{1:n}; \Theta^{(old)}\right)$
- How about $\xi_{i,j,k} \coloneqq P\left(z_{i-1} = j, z_i = k \mid x_{1:n}; \Theta^{(old)}\right)$?
- Key observation: $P(z_{i-1} = j, z_i = k \mid x_{1:n}) \propto P(z_{i-1} = j, z_i = k,\ x_{1:n})$

$$= P(x_{1:i-1}, z_{i-1} = j, z_i = k, x_{i:n})$$

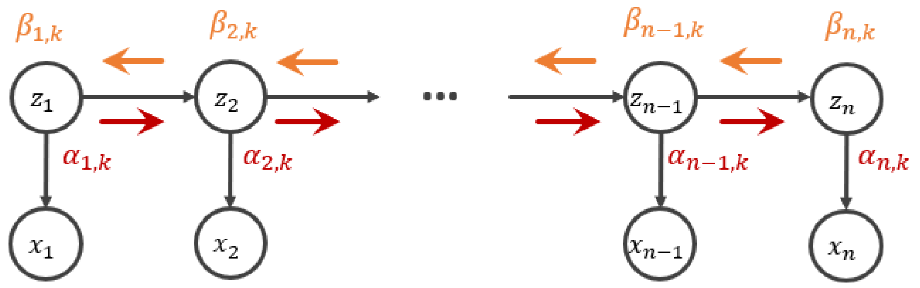
$$= P(x_{1:i-1}, z_{i-1} = j) P(z_i = k \mid z_{i-1} = j) P(x_{i:n} \mid z_i = k)$$

$$= \alpha_{i-1,j} A_{j,k}\, P_\phi(x_i) \beta_{i,k}$$

Learning HMMs with the EM algorithm - summary

- EM for HMM (Also known as the Baum-Welch algorithm):

- Repeat:
  - E-step: (1) calculate $\gamma_{i,k} := P\left(z_i = k \mid x_{1:n}; \Theta^{(t)}\right)$
    (2) calculate $\xi_{i,j,k} := P\left(z_{i-1} = j, z_i = k \mid x_{1:n}; \Theta^{(t)}\right)$
    using the forward-backward algorithm

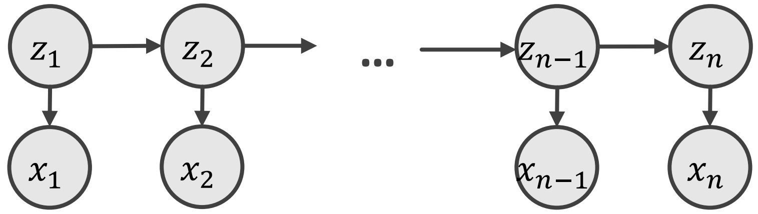


  - M-step:

    (1) $\hat{\pi}_k = P\left(z_1 = k \mid x_{1:n}; \Theta^{(\mathrm{old})}\right)$

    (2) $\hat{A}_{j,k} = \frac{\sum_i P\left(z_{i-1}=j, z_i=k \mid x_{1:n}; \Theta^{(old)}\right)}{\sum_i P\left(z_{i-1}=j \mid x_{1:n}; \Theta^{(old)}\right)}$

    (3) $\hat{\phi}_k = \mathrm{argmax}_\phi \sum_i P\left(z_i = k \mid x_{1:n}; \Theta^{(old)}\right) \ln P_\phi(x_i)$



  - Update parameters: $\Theta^{(t+1)} \leftarrow (\hat{\pi}, \hat{A}, \hat{\phi})$

HMMs: closing remarks

- Alternative algorithms for learning HMMs
  - Method of Moments
  - Provable guarantees if no model misspecification

Computer Science > Machine Learning

[Submitted on 26 Nov 2008 (v1), last revised 6 Jul 2012 (this version, v6)]

A Spectral Algorithm for Learning Hidden Markov Models

Daniel Hsu, Sham M. Kakade, Tong Zhang

Computer Science > Machine Learning

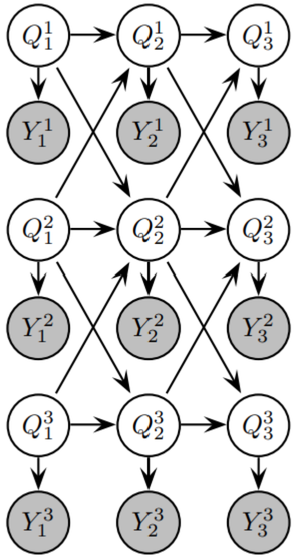
[Submitted on 29 Oct 2012 (v1), last revised 13 Nov 2014 (this version, v4)]

Tensor decompositions for learning latent variable models

Anima Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, Matus Telgarsky

- Linear dynamical systems: x_t and z_t's are continuous and satisfies joint Gaussian distribution
  - Kalman filter
  - Widely used in control applications

- Dynamic Bayesian networks: generalizing HMMs to *structured* hidden states and observations



Summary

- d-separation
- HMM for modeling and inference on sequential data
- Viterbi algorithm for finding the most likely sequence
- EM for learning the parameters (forward-backward algorithm)

Next lecture (11/7)

- Neural networks; the backpropagation algorithm

- Assigned reading: CIML 10.1-10.2