

Machine Learning for Graphs and Sequential Data

Sequential Data – Hidden Markov Models

lecturer: Prof. Dr. Stephan Günnemann
www.daml.in.tum.de

Summer Term 2020

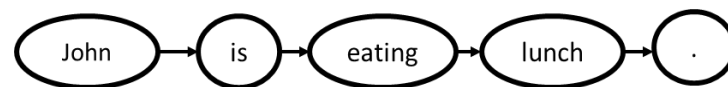
Data Analytics and
Machine Learning 

Roadmap

- Chapter: Temporal Data / Sequential Data
 1. Autoregressive Models
 2. Markov Chains
 - 3. Hidden Markov Models**
 4. Neural Network Approaches
 5. Temporal Point Processes

Motivation

- Basic autoregressive models and Markov Chains are very restrictive/simple
 - Do not well capture complex, real-world data
- Next: Probabilistic **latent variable models** for sequences of observations X_1, X_2, \dots, X_T .
 - Enable to capture more complex behavior
 - Again we focus on **discrete time-steps**; while the **observations** might be **discrete or continuous**
- Examples:
 - Object-tracking:
 - X_t = location of a moving object at time-step t
 - Time-series forecasting:
 - X_t = measurement of a sensor at time-step t (weather, stock market, ...)
 - Natural language processing:
 - X_t = t -th word in a sentence



Hidden Markov Models

- Motivation 1: In many applications, the Markov property is not realistic.
 - X_t does not capture all relevant information of $[X_1, \dots, X_t]$ → need to consider long-range dependencies, while keeping the number of parameters low.
- Motivation 2: In many applications, the state is not known but can only be observed indirectly, e.g., with sensors.
 - Example application: tracking location of an airplane
 - Not observed/latent state Z_t : physical vector quantities (e.g. position, velocity, etc.) at time-step t
 - X_t : observed noisy measurements of airplane location at time-step t
 - Note that the sequence $[Z_1, Z_2, \dots, Z_T]$ has the Markov-property. That is, one can use physics laws to approximate Z_{t+1} using Z_t .
 - However, the sequence $[X_1, \dots, X_T]$ does not necessarily have the Markov-property ⇒ We need to model long-range dependencies in this sequence.

Hidden Markov Models - Definition

- Definition: A **Hidden Markov Model (HMM)** is composed of a sequence of **hidden/latent** variables $[Z_1, \dots, Z_T]$ and a sequence of **observed** variables $[X_1, \dots, X_T]$ such that:

- The r.v. Z_1, \dots, Z_T satisfy the Markov property:

$$P(Z_{t+1}|Z_t, Z_{t-1}, \dots, Z_1) = \underbrace{P(Z_{t+1}|Z_t)}_{\text{transition probabilities}}$$

transition probabilities

- Distribution of X_t depends only on Z_t :

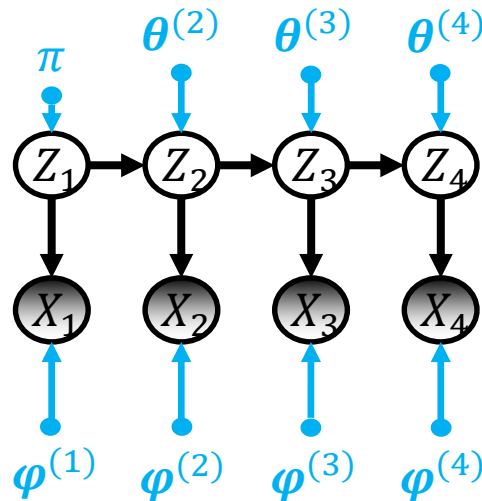
$$P(X_{t+1}|Z_1, \dots, Z_T, X_1, \dots, X_T) = \underbrace{P(X_{t+1}|Z_{t+1})}_{\text{emission probabilities}}$$

emission probabilities

- By convention for HMMs we assume discrete time $t \in \{1, 2, \dots, T\}$ and discrete r.v. $Z_t \in \{1, 2, \dots, K\}$.
- The observed data can be discrete or continuous

Hidden Markov Models – General Case

- In the general case, the graphical model of a HMM is:



- The joint distribution can be written as:

$$\begin{aligned}
 & P(Z_1 = z_1, \dots, Z_T = z_T, X_1 = x_1, \dots, X_T = x_T) \\
 &= P(Z_1 = z_1; \boldsymbol{\pi}) \prod_{t=1}^{T-1} P(Z_{t+1} = z_{t+1} | Z_t = z_t; \boldsymbol{\theta}^{(t+1)}) \prod_{t=1}^T P(X_t = x_t | Z_t = z_t; \boldsymbol{\varphi}^{(t)})
 \end{aligned}$$

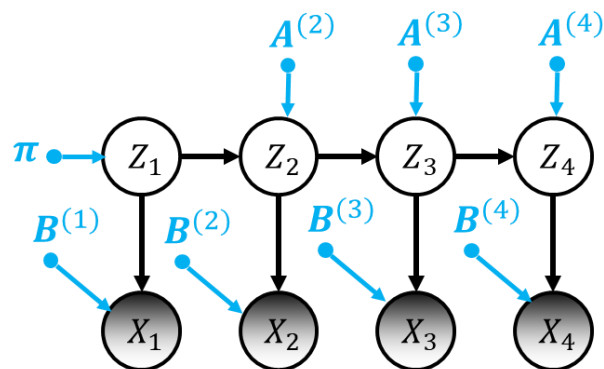
Hidden Markov Models – Discrete Case

- We start by discussing the discrete case, i.e. $X_t \in \{1, 2, \dots, K'\}$:

$$P(Z_1 = i) = \pi_i$$

$$P(Z_{t+1} = j | Z_t = i) = A_{ij}^{(t+1)}$$

$$P(X_{t+1} = j | Z_{t+1} = i) = B_{ij}^{(t+1)}$$



K

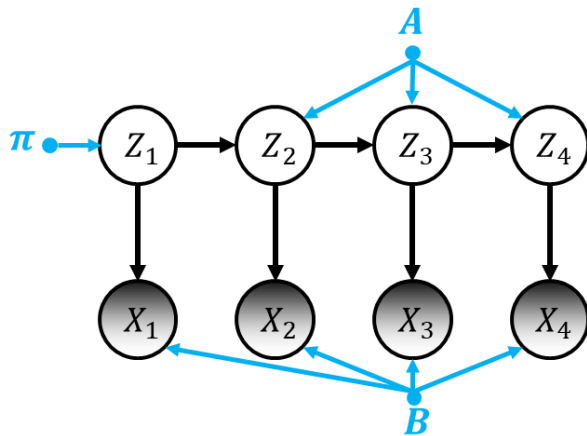
K'

$$\text{\#Parameters} = K + (T - 1) K^2 + T K K'$$

$$B \in \mathbb{R}^{K \times K'}$$

Hidden Markov Models – Parameter Tying

- To reduce the number of parameters, variables can share parameters:



$$\text{\#Parameters} = K + K^2 + KK'$$

- From now on, we assume parameter tying as in Markov chains. The joint distribution becomes:

$$P(Z_1 = z_1, \dots, Z_T = z_T, X_1 = x_1, \dots, X_T = x_T) = P(Z_1 = z_1; \pi) \prod_{t=1}^{T-1} A_{z_t z_{t+1}} \prod_{t=1}^T B_{z_t x_t}$$

z : D D F

x : G H H

$\pi_D \cdot A_{DD} \cdot A_{DF} \cdot B_{DG}$

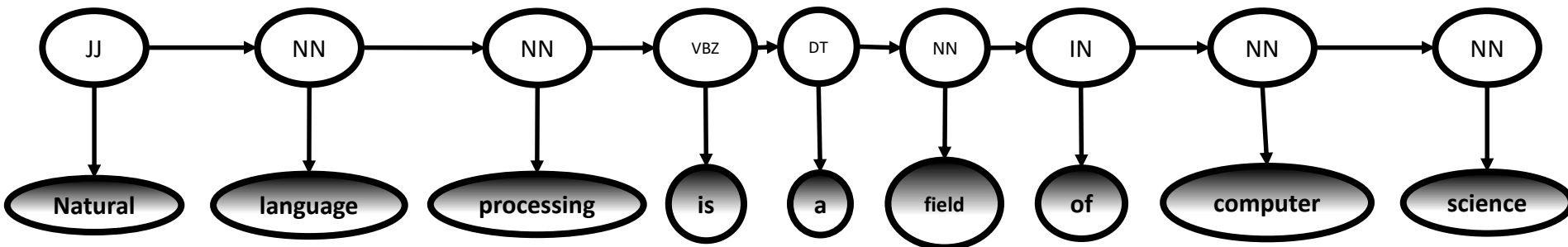
π_{z_1}

$\cdot B_{DH} \cdot B_{FH}$

Hidden Markov Models – Example 1

- Example 1: Part of speech tagging / sequence labeling
 - Z_t : part of speech (noun, verb, adjective, etc.)
 - X_t : a word

JJ: adjective
 NN: noun, singular or mass
 VBZ: verb, 3rd person singular present
 DT: determiner
 IN: preposition or subordinating conjunction



Example adapted from: <http://www.phontron.com/slides/nlp-programming-en-04-hmm.pdf>

Hidden Markov Models – Example 2

- Example 2: A simple model for daily weather condition
 - $Z_t \in \{\text{rainy}, \text{sunny}, \text{cloudy}\}$: hidden weather condition at day t
 - $X_t \in \{\text{high}, \text{low}\}$: measured temperature at day t

$$\Pr(Z_{t+1} = j | Z_t = i) = A_{ij}$$

$$\Pr(X_t = j | Z_t = i) = B_{ij}$$

rainy sunny cloudy
H H L L H

$$A = \begin{matrix} & \begin{matrix} \text{rainy} & \text{sunny} & \text{cloudy} \end{matrix} \\ \begin{matrix} \text{rainy} \\ \text{sunny} \\ \text{cloudy} \end{matrix} & \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.1 & 0.5 & 0.4 \\ 0.4 & 0.1 & 0.5 \end{bmatrix} \end{matrix}$$

$$B = \begin{matrix} & \begin{matrix} \text{High} & \text{Low} \end{matrix} \\ \begin{matrix} \text{rainy} \\ \text{sunny} \\ \text{cloudy} \end{matrix} & \begin{bmatrix} 0.2 & 0.8 \\ 0.9 & 0.1 \\ 0.3 & 0.7 \end{bmatrix} \end{matrix}$$

Tasks Concerning HMMs

Recall:

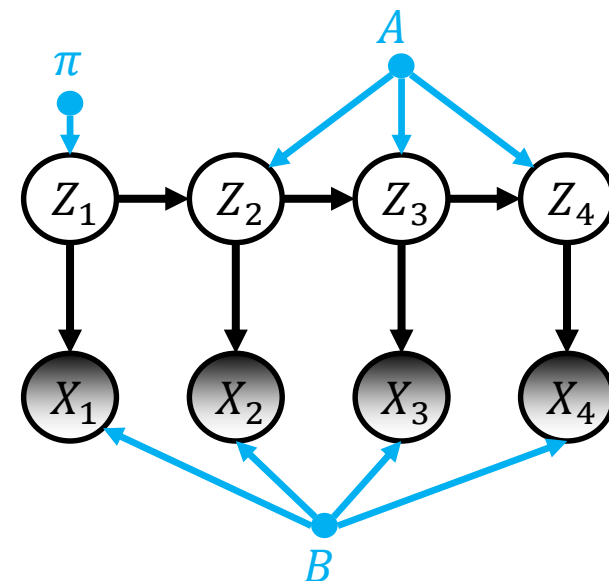
π : parametrizes $\Pr(Z_1)$

A : parametrizes $\Pr(Z_{t+1}|Z_t)$

B : parametrizes $\Pr(X_t|Z_t)$

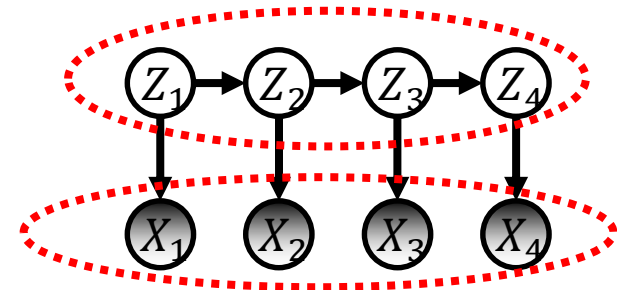
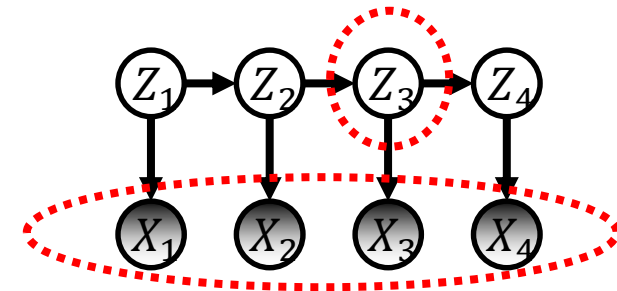
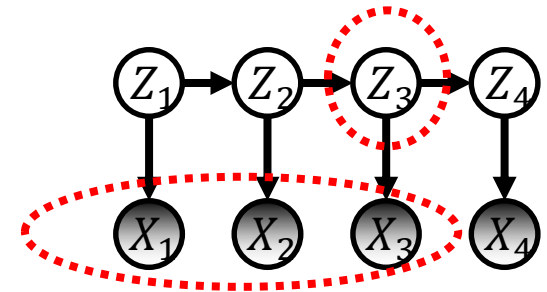
- Inference:
 - We let **model parameters** be fixed (e.g. tuned by an expert).
 - We seek to find some information from the posterior distribution $\Pr(Z_{1:T}|X_{1:T})$.
 - Examples:
 - Filtering / Smoothing (forwards – backwards)
 - MAP inference (Viterbi)

- Parameter Learning:
 - We seek to learn **model parameters**
 - $X_{1:T}$ is observed
 - $Z_{1:T}$ is (usually) not observed



Inference for HMMs

- Filtering: computes the belief state $\Pr(Z_t|X_{1:t})$ incrementally as the data streams-in, i.e., **online** setting.
 - Infers Z_t using the observations up to time-step t .
- Smoothing: computes $\Pr(Z_t|X_{1:T})$ **offline**.
 - Infers Z_t by conditioning on past and future data.
- MAP inference: computes $\arg \max_{Z_{1:T}} \Pr(Z_{1:T}|X_{1:T})$.
 - i.e. mode of the posterior distribution.
 - Also known as Viterbi decoding
 - Attention: Most probable *sequence* might be different from simply using mode of $\Pr(Z_t|X_{1:T})$ for each t individually



The Forwards Algorithm

- Goal: incrementally compute $P(Z_t | X_{1:t})$

- The Bayes rule gives:

$$P(Z_t = k | X_{1:t}) = \frac{P(Z_t = k, X_{1:t})}{\sum_{j=1}^K P(Z_t = j, X_{1:t})}$$

- For convenience, we denote:

$$\alpha_t(k) \stackrel{\text{def}}{=} P(Z_t = k, X_{1:t}) \text{ and } \alpha_t = \begin{bmatrix} \alpha_t(1) \\ \vdots \\ \alpha_t(K) \end{bmatrix}$$

- Hence, we have:

$$P(Z_t = k | X_{1:t}) = \frac{\alpha_t(k)}{\text{sum}(\alpha_t)} = \frac{\alpha(k)}{\sum_{k'} \alpha(k')}$$

- The **Forward algorithm** computes recursively the parameters:

1. Compute α_1 (initialisation)
2. Given α_t , compute α_{t+1} (recursion)

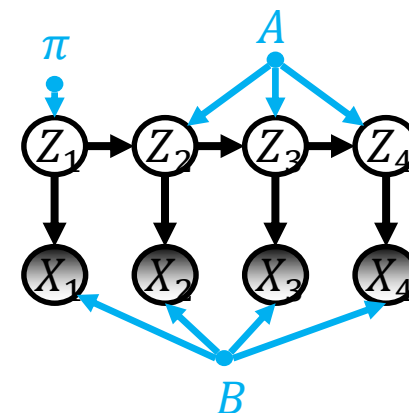
$$P(Z_t | X_{1:t}) = \sum_{Z_{1:t-1}} P(Z_{1:t} | X_{1:t})$$

NAIVE
solution

The Forwards Algorithm - Initialisation

- Initialisation: The computation of the parameters α_1 can be done directly

$$\begin{aligned}\alpha_1(k) &= P(Z_1 = k, X_1) \\ &= P(Z_1 = k)P(X_1|Z_1 = k) \\ &= \pi_k B_{kx_1}\end{aligned}$$

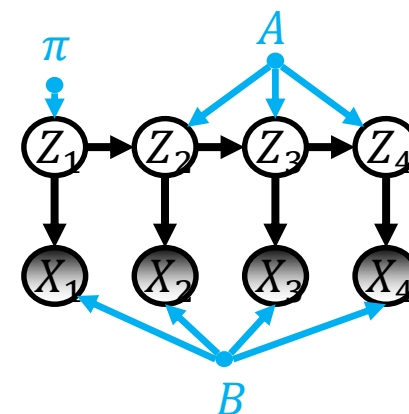


$$P(A, B) = P(A|B) \cdot P(B)$$

The Forwards Algorithm - Recursion

- Recursion: Given α_t , we can compute α_{t+1}

$$\begin{aligned}
 \alpha_{t+1}(k) &= P(Z_{t+1} = k, X_{1:t+1}) \\
 &\stackrel{(*)}{=} P(X_{t+1} | Z_{t+1} = k, X_{1:t}) P(Z_{t+1} = k, X_{1:t}) \\
 &= P(X_{t+1} | Z_{t+1} = k) \sum_{j=1}^K P(Z_{t+1} = k, Z_t = j, X_{1:t}) \\
 &= P(X_{t+1} | Z_{t+1} = k) \sum_{j=1}^K \underbrace{P(Z_{t+1} = k | Z_t = j, X_{1:t})}_{P(Z_{t+1}=k | Z_t=j)} P(Z_t = j, X_{1:t}) \\
 &= B_{k(x_{t+1})} \sum_{j=1}^K A_{jk} \alpha_t(j)
 \end{aligned}$$



$$P(Z_{t+1}=k | Z_t=j)$$


The Forwards Algorithm (cont.)

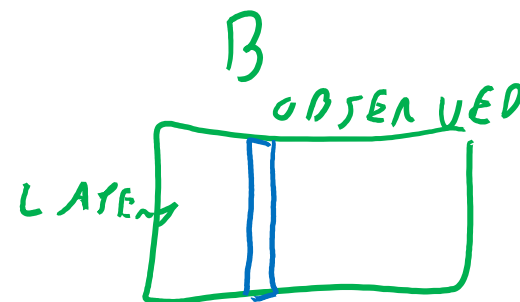
- Writing the last equation using matrix operators:

$$\alpha_{t+1}(k) = B_{k(x_{t+1})} \sum_{j=1}^K \alpha_t(j) A_{jk}$$

$$\alpha_{t+1} = \mathbf{B}_{:(x_{t+1})} \odot (\mathbf{A}' \alpha_t)$$

$$\begin{bmatrix} \alpha_{t+1}(1) \\ \alpha_{t+1}(2) \\ \vdots \\ \alpha_{t+1}(K) \end{bmatrix} = \begin{bmatrix} B(1, x_{t+1}) \\ B(2, x_{t+1}) \\ \vdots \\ B(K, x_{t+1}) \end{bmatrix} \odot \left(\boxed{\mathbf{A}'} \begin{bmatrix} \alpha_t(1) \\ \alpha_t(2) \\ \vdots \\ \alpha_t(K) \end{bmatrix} \right)$$





Notation:
 \odot denotes
 Hadamard product

- Finding $\alpha_{1:T}$ requires $O(TK^2)$ operations, which is linear in T .

The Forward-Backwards Algorithm

- Goal: incrementally compute $P(Z_t | X_{1:T})$

- The Bayes rule gives:

$$P(Z_t = k | X_{1:T}) = \frac{\underbrace{P(Z_t = k, X_{1:t})}_{\alpha} \underbrace{P(X_{t+1:T} | Z_t = k)}_{\beta}}{\sum_{j=1}^K P(Z_t = j, X_{1:T})}$$

- For convenience, we denote also:

$$\beta_t(k) \stackrel{\text{def}}{=} P(X_{t+1:T} | Z_t = k) \text{ and } \boldsymbol{\beta}_t = \begin{bmatrix} \beta_t(1) \\ \vdots \\ \beta_t(K) \end{bmatrix}$$

- Hence, using $\alpha_t(k)$ and $\beta_t(k)$ we have:

$$P(Z_t = k | X_{1:T}) \propto \alpha_t(k) \beta_t(k)$$

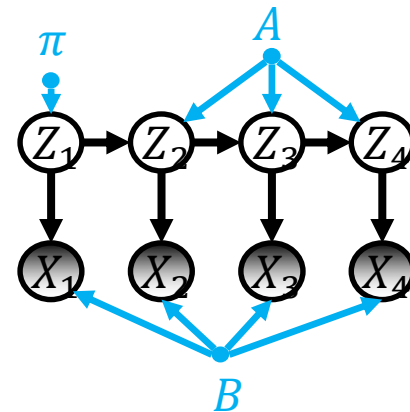
- The **Backward algorithm** computes recursively the parameters:
 1. Compute $\boldsymbol{\beta}_T$ (initialisation)
 2. Given $\boldsymbol{\beta}_{t+1}$, compute $\boldsymbol{\beta}_t$ (recursion)

The Backward Algorithm - Initialisation

- Initialisation: The computation of the parameters β_T can be done directly

$$\beta_T(k) = 1$$

- This comes from the fact that $P(Z_t = k | X_{1:T}) \propto \alpha_t(k) \beta_t(k)$ and that for $t = T$ the term is already completely “captured” by $\alpha_t(k)$. Thus $\beta_T(k)$ has to be a constant.



The Backward Algorithm - Recursion

- Recursion: Given β_{t+1} , we can compute β_t

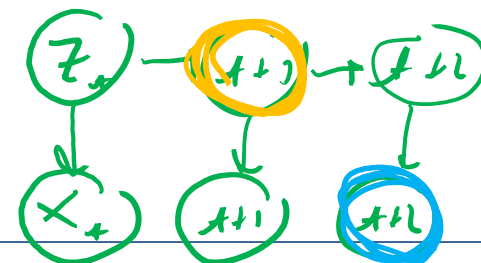
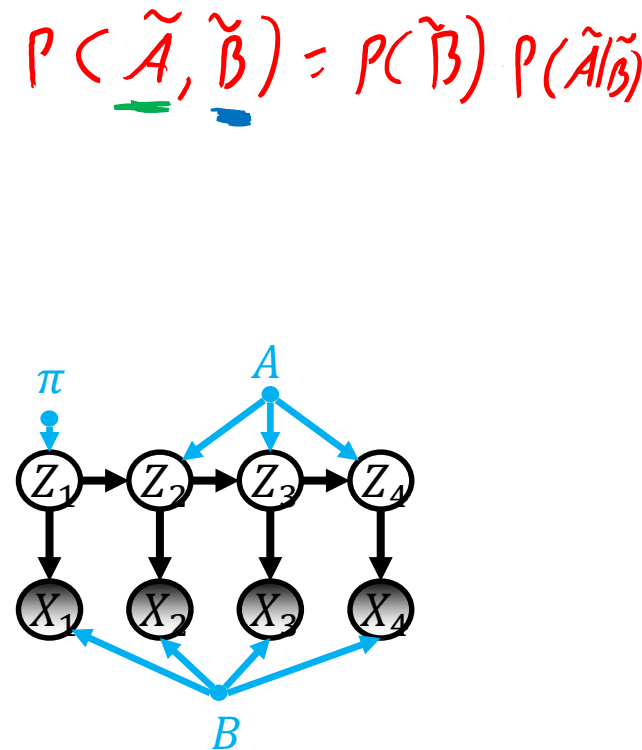
$$\beta_t(j) = P(X_{t+1:T} | Z_t = j)$$

$$= \sum_{k=1}^K P(X_{t+1:T}, Z_{t+1} = k | Z_t = j)$$

$$= \sum_{k=1}^K P(X_{t+1}, Z_{t+1} = k | Z_t = j) P(X_{t+2:T} | Z_t = j, X_{t+1}, Z_{t+1} = k)$$

$$= \sum_{k=1}^K P(Z_{t+1} = k | Z_t = j) \Pr(X_{t+1} | Z_{t+1} = k, Z_t = j) P(X_{t+2:T} | Z_{t+1} = k)$$

$$= \sum_{k=1}^K A_{jk} B_{kx_{t+1}} \beta_{t+1}(k)$$



The Backward Algorithm (cont.)

- Writing the last equation using matrix operators:

$$\beta_t(j) = \sum_{k=1}^K A_{jk} B_{kx_{t+1}} \beta_{t+1}(k)$$

$$\boldsymbol{\beta}_t = \mathbf{A} (\mathbf{B}_{:(x_{t+1})} \odot \boldsymbol{\beta}_{t+1})$$

$$\begin{bmatrix} \beta_t(1) \\ \beta_t(2) \\ \vdots \\ \beta_t(K) \end{bmatrix} = \boxed{\mathbf{A}} \left(\begin{bmatrix} B(1, x_{t+1}) \\ B(2, x_{t+1}) \\ \vdots \\ B(K, x_{t+1}) \end{bmatrix} \odot \begin{bmatrix} \beta_{t+1}(1) \\ \beta_{t+1}(2) \\ \vdots \\ \beta_{t+1}(K) \end{bmatrix} \right)$$

- Computing $\boldsymbol{\beta}_{1:T}$ requires $O(TK^2)$ operations, which is linear in T .

The Forward-Backward Algorithm – Applications I

- Compute the probability of being in state k at time t online:

$$P(Z_t = k | X_{1:t}) = \frac{\alpha_t(k)}{\sum_s \alpha_t(s)}$$

- via argmax we can simply get the most likely state k

- Compute the probability of being in state k at time t offline:

$$\gamma_t(k) := P(Z_t = k | X_{1:T}) = \frac{\alpha_t(k)\beta_t(k)}{\sum_s \alpha_t(s)\beta_t(s)}$$

The Forward-Backward Algorithm – Applications II

- Compute the probability that two “adjacent” states have specific realizations:

$$\xi_t(i, j) := P(Z_t = i, Z_{t+1} = j | X_{1:T}) = \frac{\alpha_t(i) A_{ij} \beta_{t+1}(j) B_{jx_{t+1}}}{\sum_u \sum_v \alpha_t(u) A_{uv} \beta_{t+1}(v) B_{vx_{t+1}}}$$

Proof: Observe that $P(X_{1:T})$ is some constant, thus we have $\xi_t(i, j) \propto P(Z_t = i, Z_{t+1} = j, X_{1:T})$. Now, by writing the chain rule as $\{Z_t = i, X_{1:t}\}, \{Z_{t+1} = j\}, \{X_{t+2:T}\}, \{X_{t+1}\}$, we obtain:

$$\begin{aligned} P(Z_t = i, Z_{t+1} = j, X_{1:T}) &= P(Z_t = i, X_{1:t}) \cdot P(Z_{t+1} = j | Z_t = i, X_{1:t}) \\ &\quad \cdot P(X_{t+2:T} | Z_t = i, X_{1:t}, Z_{t+1} = j) \cdot P(X_{t+1} | Z_t = i, X_{1:t}, Z_{t+1} = j, X_{t+2:T}) \\ &\quad \Downarrow \\ \xi_t(i, j) &\propto \alpha_t(i) \cdot A_{ij} \cdot \beta_{t+1}(j) \cdot B_{jx_{t+1}} \end{aligned}$$

MAP Inference in HMMs

$$P(Z_{1:T} = z_{1:T})$$

↑ # K^T possibilities

- Goal: Given the observed sequence $X_{1:T}$, find the most probable sequence of hidden states z_1, \dots, z_T .
- In other words, find mode of the posterior distribution $\Pr(Z_{1:T} | X_{1:T})$

$$\arg \max_Z P(Z_{1:T} | X_{1:T}) = \arg \max_Z \log[P(Z_{1:T}, X_{1:T})]$$

$$= \arg \max_Z \log[P(Z_1) P(X_1 | Z_1)] + \sum_{t=2}^T \log[P(Z_t | Z_{t-1}) P(X_t | Z_t)]$$

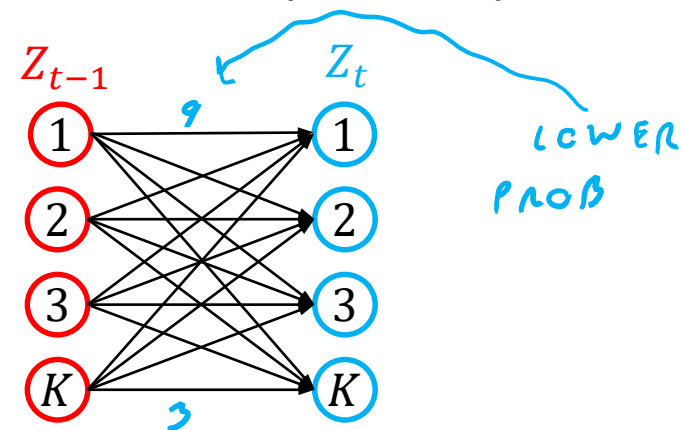
- Each term $\log[P(Z_t | Z_{t-1}) P(X_t | Z_t)]$ depends on values of Z_{t-1} and Z_t .

- Think of it as a bi-partite graph.

weight of the edge ($i - j$) =

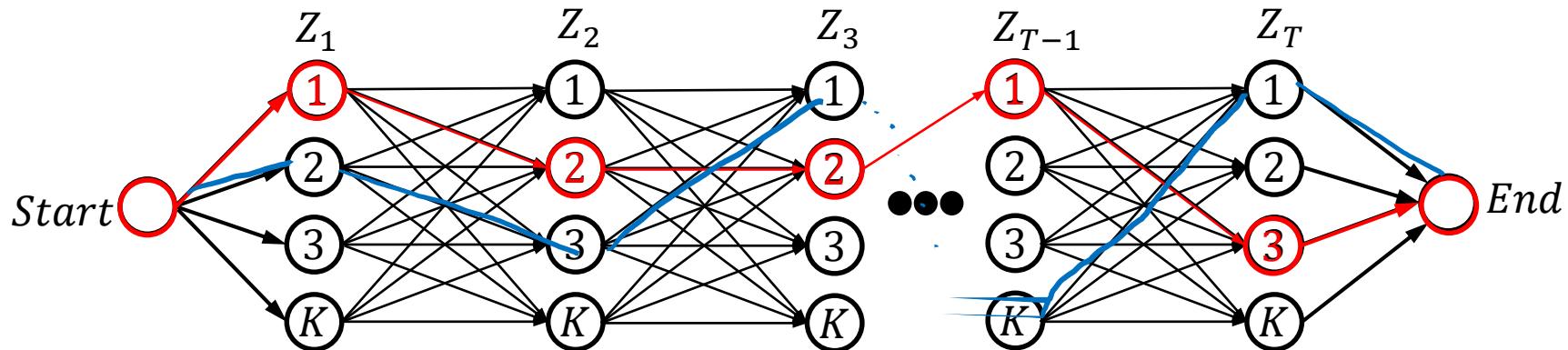
$$-\log[P(Z_t = j | Z_{t-1} = i) P(X_t | Z_t = j)]$$

A_{ij} $B_{j|x}$



MAP Inference in HMMs (cont.)

- We can formulate the MAP inference as a shortest-paths problem.
 - weights of edges connected to the *Start* node: $-\log[\Pr(Z_1 = j) \Pr(X_1|Z_1 = j)]$
 - weights of the intermediate layers: $-\log[\Pr(Z_t = j|Z_{t-1} = i) \Pr(X_t|Z_t = j)]$
 - weights of the edges connected to the *End* node: 0



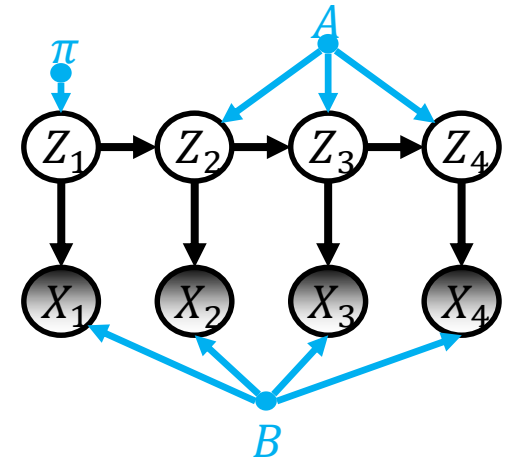
Each directed path corresponds to an assignment to variables $Z_{1:T}$.
 Sum of edge weights = $-\log \Pr(Z_{1:T}, X_{1:T})$

complexity: $O(TK^2)$

Called Viterbi algorithm

Parameter Learning

- Variables $X_{1:T_n}^{(n)}$ are observed, not $Z_{1:T_n}^{(n)}$
- To keep the notation simple, let's assume that we have a single sequence \mathbf{X} .



- We seek to learn **model parameters** $\theta = \{\pi, A, B\}$.
- Goal: Solve $\max_{\theta} \log p_{\theta}(\mathbf{X})$
- You know how to do this! Variational inference!

$$p_{\theta}(x) = \sum_z p(x, z)$$

- We need to introduce and optimize over a variational distribution $q(\mathbf{Z})$
- You also know: For any fixed θ , setting $q(\mathbf{Z}) = p_{\theta}(\mathbf{Z}|\mathbf{X})$ is the optimal choice; indeed we already have all information available to compute the ELBO

Parameter Learning

- Using the optimal $q(\mathbf{Z})$ (for a fixed $\boldsymbol{\theta}^{old}$) the ELBO becomes:

$$\begin{aligned} \mathbb{E}_{P(\mathbf{Z}_{1:T}|\mathbf{X}_{1:T}, \boldsymbol{\theta}^{old})} [\log P(\mathbf{X}_{1:T}, \mathbf{Z}_{1:T}, \boldsymbol{\theta})] = & \sum_k P(Z_1 = k | \mathbf{X}_{1:T}, \boldsymbol{\theta}^{old}) \log(\pi_k) \\ & + \sum_{i,j} \sum_t P(Z_t = i, Z_{t+1} = j | \mathbf{X}_{1:T}, \boldsymbol{\theta}^{old}) \log(A_{ij}) \\ & + \sum_i \sum_t P(Z_t = i | \mathbf{X}_{1:T}, \boldsymbol{\theta}^{old}) \mathbb{I}(x_t = j) \log(B_{ij}) \end{aligned}$$

- Thanks to the Forward-Backward algorithm, the blue terms can be computed efficiently and in closed form
- Important fact: We do **not** pose a mean field assumption
 - i.e. $q(\mathbf{Z})$ is **not** factorized in independent terms
 - we keep the dependency introduced by the sequence (in the above case we even get the exact posterior distribution)
 - still, we do **not** have an exponential blow up $O(K^T)$; only $O(TK^2)$

Parameter Learning

- Using the optimal $q(\mathbf{Z})$ (for a fixed $\boldsymbol{\theta}^{old}$) the ELBO becomes:

$$\begin{aligned} \mathbb{E}_{P(\mathbf{Z}_{1:T}|\mathbf{X}_{1:T}, \boldsymbol{\theta}^{old})} [\log P(\mathbf{X}_{1:T}, \mathbf{Z}_{1:T}, \boldsymbol{\theta})] = & \sum_k P(Z_1 = k | \mathbf{X}_{1:T}, \boldsymbol{\theta}^{old}) \log(\pi_k) \\ & + \sum_{i,j} \sum_t P(Z_t = i, Z_{t+1} = j | \mathbf{X}_{1:T}, \boldsymbol{\theta}^{old}) \log(A_{ij}) \\ & + \sum_i \sum_t P(Z_t = i | \mathbf{X}_{1:T}, \boldsymbol{\theta}^{old}) \mathbb{I}(x_t = j) \log(B_{ij}) \end{aligned}$$

- We can now solve $\boldsymbol{\theta}^{new} = \operatorname{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{P(\mathbf{Z}_{1:T}|\mathbf{X}_{1:T}, \boldsymbol{\theta}^{old})} [\log P(\mathbf{X}_{1:T}, \mathbf{Z}_{1:T}, \boldsymbol{\theta})]$
 - you could use projected gradient ascent or (since available here) the closed-form solution for $\boldsymbol{\theta}^{new}$
- Indeed this alternating optimization of the ELBO (compute blue terms; update $\boldsymbol{\theta}$; repeat) is just the EM-algorithm for HMMs (called Baum-Welch)

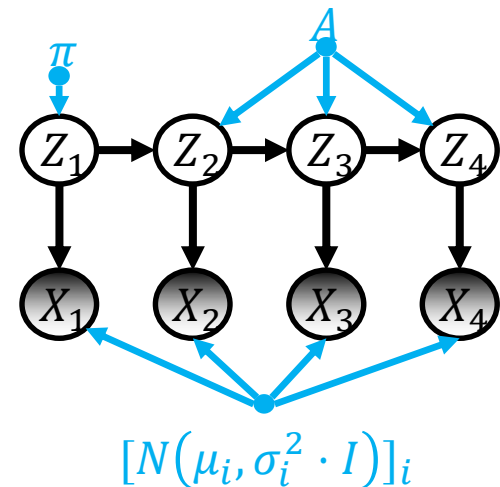
Hidden Markov Models – Continuous Data

- Before, we assumed discrete time $t \in \{1, 2, \dots, T\}$ and discrete r.v. $Z_t \in \{1, 2, \dots, K\}$, $X_t \in \{1, 2, \dots, K'\}$:

$$\begin{aligned} P(Z_1 = i) &= \pi_i \\ P(Z_{t+1} = j | Z_t = i) &= A_{ij} \\ P(X_{t+1} = j | Z_{t+1} = i) &= B_{ij} \end{aligned}$$

- Now, we assume discrete time $t \in \{1, 2, \dots, T\}$, discrete r.v. $Z_t \in \{1, 2, \dots, K\}$, and continuous $X_t \in \mathbb{R}^d$:

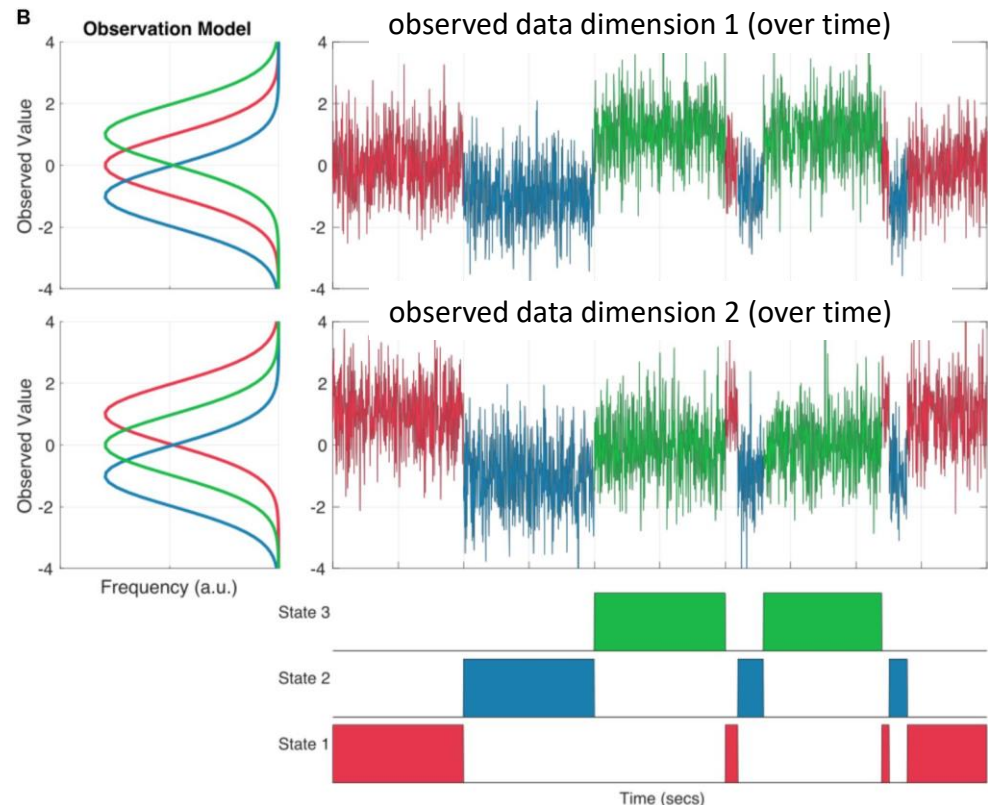
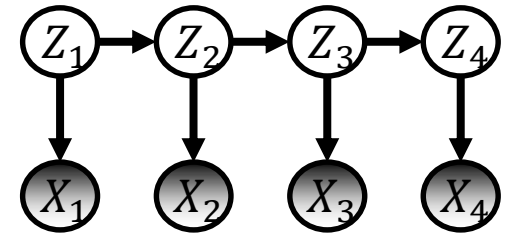
$$\begin{aligned} P(Z_1 = i) &= \pi_i \\ P(Z_{t+1} = j | Z_t = i) &= A_{ij} \\ P(X_{t+1} = x | Z_{t+1} = i) &= N(x | \mu_i, \sigma_i^2 \cdot I) \end{aligned}$$



Hidden Markov Models – Continuous Data

- Example continuous HMM:
 - The r.v. X_t are 2-D Gaussians
 - The r.v. Z_t can take 3 states
 - The probability to stay in the same state $P(Z_{t+1} = i | Z_t = i)$ is high
- It can be used for **time-series segmentation**
 - Compute the probability of the hidden state given the observations $P(Z_t = i | X_{1:T})$; assign the most probable latent state at time t
 - Or use Viterbi

$$A = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$



Images from: <https://www.frontiersin.org/article/10.3389/fnins.2018.00603>

Hidden Markov Models – Continuous Data

- Inference (i.e. Forward backward algorithm and MAP) stays the same. The probability $\Pr(X_t|Z_t)$ is just computed with Normal distribution instead of Categorical distribution, i.e.:

$$P(X_t = x|Z_t = k) = B_{kx} \rightarrow P(X_t = x|Z_t = k) = N(x|\mu_k, \sigma_k^2 \cdot I)$$

- Parameter learning is also only slightly different. We learn parameters μ_i, σ_i instead of B_{ij}

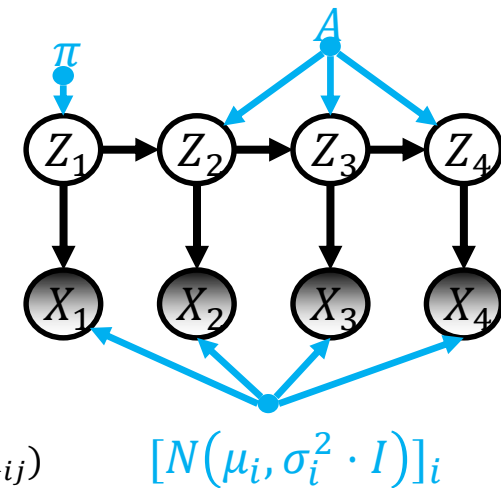
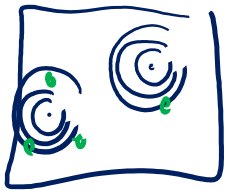
Parameter Learning – Continuous Case

- Using the optimal $q(\mathbf{Z})$ (for fixed θ^{old}) the ELBO is:

$$E_{P(\mathbf{Z}_{1:T}|\mathbf{X}_{1:T}\theta^{old})}[\ln P(\mathbf{X}_{1:T}, \mathbf{Z}_{1:T}, \theta)] = \sum_k P(Z_1 = k | \mathbf{X}_{1:T}, \theta^{old}) \log(\pi_k)$$

$$+ \sum_{i,j} \sum_t P(Z_t = i, Z_{t+1} = j | \mathbf{X}_{1:T}, \theta^{old}) \log(A_{ij})$$

$$+ \sum_i \sum_t P(Z_t = i | \mathbf{X}_{1:T}, \theta^{old}) \log(N(X_t | \mu_i, \sigma_i^2 \cdot I))$$



- Again one can solve $\theta^{new} = \operatorname{argmax}_{\theta} \mathbb{E}_{P(\mathbf{Z}_{1:T}|\mathbf{X}_{1:T}, \theta^{old})} [\log P(\mathbf{X}_{1:T}, \mathbf{Z}_{1:T}, \theta)]$ easily (e.g. gradient based or closed-form)
- Observation: Estimate for μ_i, σ_i^2 is equivalent to the setting in a GMM, e.g.

$$\mu_i^{new} = \frac{\sum_{t=1}^T \gamma_t(i) X_t}{\sum_{t=1}^T \gamma_t(i)}, \quad \sigma_i^{new} = \frac{\sum_{t=1}^T \gamma_t(i) (\mu_i^{new} - x_t)(\mu_i^{new} - x_t)^T}{\sum_{t=1}^T \gamma_t(i)} \quad \text{where } \gamma_t(i) := P(Z_t = i | \mathbf{X}_{1:T}, \theta^{old})$$

- GMM: observations X_i are independent $p(z_i | x_i, \theta^{old})$
- HMM: observations X_t are conditional independent given \mathbf{Z}
for the estimate above, we assumed $P(\mathbf{Z}_{1:T} | \mathbf{X}_{1:T}, \theta^{old})$ is fixed/given

Overview of Tasks concerning HMMs

Problem	Algorithm	Time Complexity
Filtering: Obtaining $\Pr(Z_t X_{1:t})$	Forwards	$O(TK^2)$
Smoothing: Obtaining $\Pr(Z_t X_{1:T})$	Forwards-Backwards	$O(TK^2)$
MAP Estimation: Obtaining $\arg \max_{Z_{1:T}} \Pr(Z_{1:T} X_{1:T})$	Viterbi Decoding	$O(TK^2)$
Learning: approximately obtaining $\arg \max_{\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}} \Pr(X_{1:T}; \mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$	Variational Inference / Baum-Welch (EM)	$O(TK^2)$

$T = \text{sequence length}$

$K = \text{\#possible states for } Z_t$

Questions – HMM

1. Does the sequence $[Z_1, \dots, Z_T]$ fulfill the Markov property ? Why ?
2. Does the sequence $[X_1, \dots, X_T]$ fulfill the Markov property ? Why ?
3. In Variational Inference we sometimes need to approximate the ELBO (for example by sampling from the the latent Z from the variational distribution). For learning HMM parameters, do we need to sample Z as well?

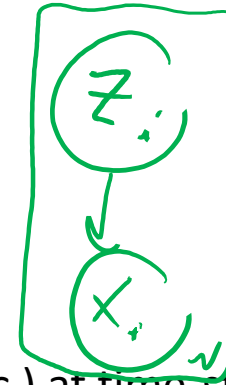
Discussion

- The **index set**, $t \in \{1, 2, \dots, T\}$, is discrete in all the presented models
 - The observations are only ordered in a sequence (the “actual time” does not play a role)
 - This setting is similar to equidistant time between observations
- All models have observed variables, but not all have latent variables
- The **state space** of the observed and latent variables can be discrete or continuous

		Latent space		
		No	Discr.	Cont.
Observation Space	Discr.	Markov Chains	HMM-Discr	No default method
	Cont.	AR	HMM-Cont	e.g. linear dynamical system; estimated via. Kalman Filter

Example – Continuous Latent Space

VAE:



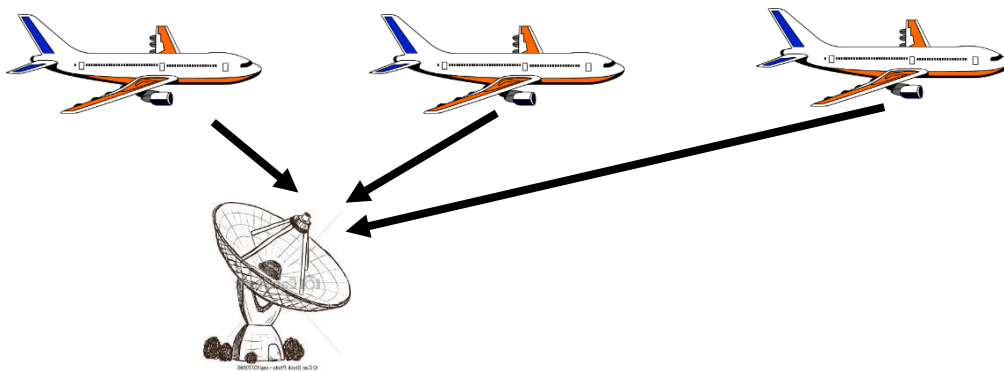
$$Z_t \sim \mathcal{N}(\mu, \sigma)$$

Example: Tracking

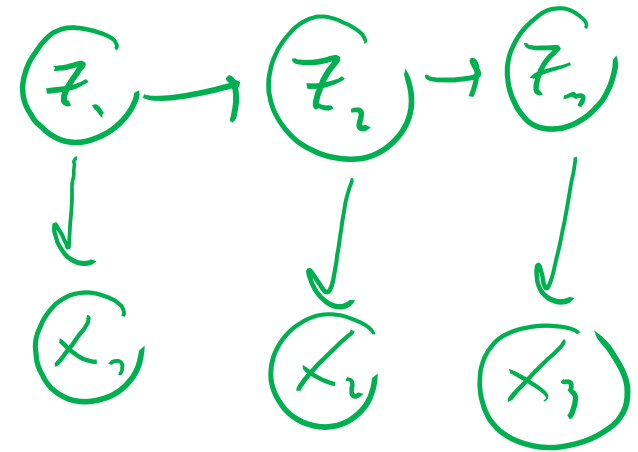
- Z_t : physical vector quantities (e.g. position, velocity, etc.) at time-step t
- X_t : observed noisy measurements of airplane location at time-step t

$$\Pr(Z_{t+1}|Z_t) = N(Z_{t+1}|f(Z_t), \sigma_Z^2) \quad f: \text{given } Z_t \text{ predicts } Z_{t+1}, \text{e.g., using laws of motion}$$

$$\Pr(X_t|Z_t) = N(X_t|g(Z_t), \sigma_x^2) \quad g: \text{sensor measurement based on } Z_t$$



Images from: www.canstockphoto.com , www.kisspng.com



Discussion

- We only discussed discrete time i.e. $t \in \{1, 2, \dots, T\}$ so far
 - The observations are only ordered in a sequence (the “actual time” does not play a role)
 - This setting is similar to equidistant time between observations

- In real applications, time is often continuous i.e. $t \in \mathbb{R}$
 - Asynchronous time: Events/Measurements might occur at asynchronous time. The time gaps between events Δt might be different.
 - Example: Speech recognition, alarm prediction
 - Models: Temporal Point Process (later section!)
 - Continuous time: Measurements might be performed almost continuously. The time gaps between events Δt are very (infinitesimal) small
 - Example: Temperature, stock price
 - Models: Continuous Stochastic Process e.g. Brownian Motion

Reading Material

- [1] Pattern Recognition and Machine Learning, section 13.2:
<https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf>