Machine Learning for Graphs and Sequential Data

Sequential Data – Hidden Markov Models

Lecturer: Prof. Dr. Stephan Günnemann
cs.cit.tum.de/daml

Summer Term 2024

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 capture complex, real-world data well
- Next: Probabilistic **latent variable models** for sequences of observations $X_1, X_2, \ldots, X_T$.
  - 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.

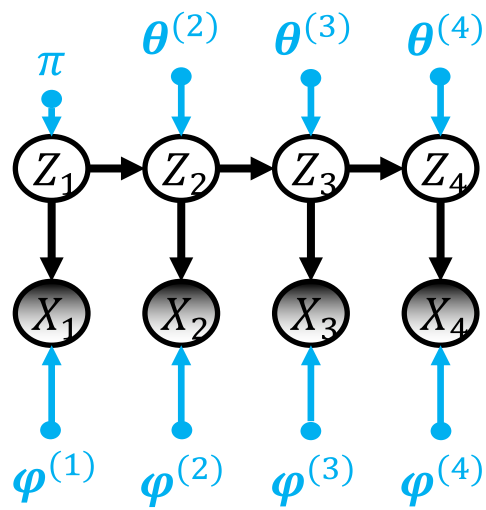
- Motivation 2: In many applications, the state is not known but can only be observed indirectly, e.g., with sensors.
  - Example application: tracking location of an airplane
  - Not observed/latent state Z_t: physical vector quantities (e.g. position, velocity, etc.) at time-step t
  - X_t: observed 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:
$$\mathrm{P}(Z_{t+1} | Z_t, Z_{t-1}, \dots, Z_1) = \underbrace{\mathrm{P}(Z_{t+1} | Z_t)}_{\text{transition probabilities}}$$
  - Distribution of X_t depends only on Z_t:
$$\mathrm{P}\left(X_{t+1} \middle| Z_1, \dots, Z_{T,} X_1, \dots, X_{T,}\right) = \underbrace{\mathrm{P}(X_{t+1} | Z_{t+1})}_{\text{emission probabilities}}$$
- 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:

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

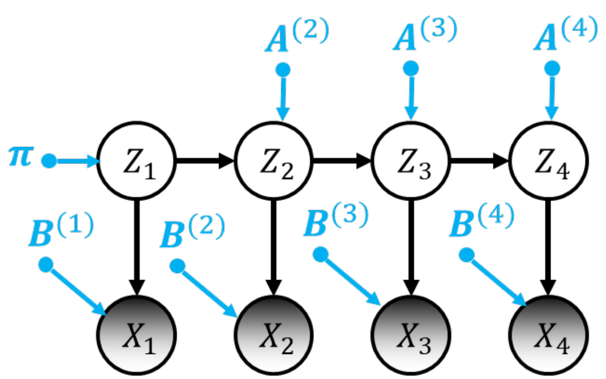
Hidden Markov Models – Discrete Case

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

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

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

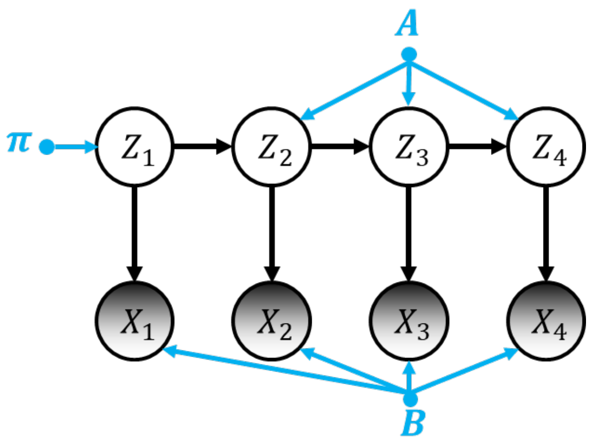
$$P(X_{t+1} = j | Z_{t+1} = i) = B_{ij}^{(t+1)}$$



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

Hidden Markov Models – Parameter Tying

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



$$\boxed{\text{\#Parameters} = O(K + K^2 + KK')}$$

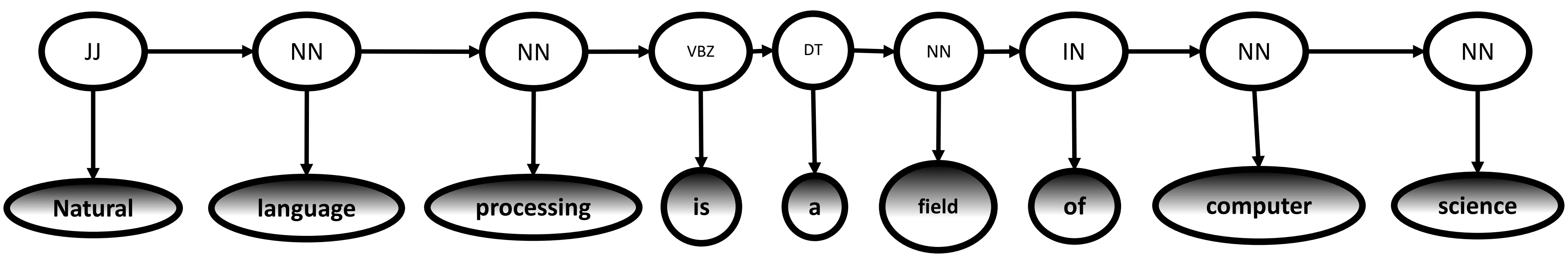
- From now on, we assume parameter tying as in Markov chains. The joint distribution becomes:

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

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
  - $Z_t \in \{rainy, sunny, cloudy\}$: hidden weather condition at day t
  - $X_t \in \{high, low\}$: measured temperature at day t

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

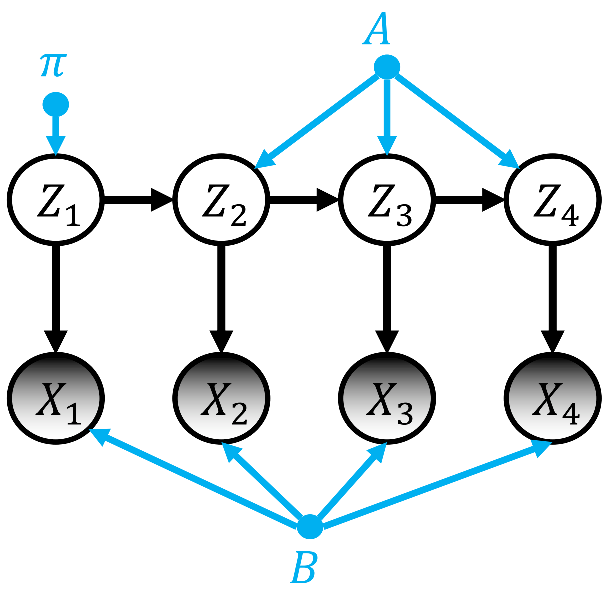
$$\Pr(X_t = j | Z_t = i) \ = B_{ij}$$

$$A = \begin{array}{c|ccc|} & rainy & sunny & cloudy \\ \hline rainy & 0.6 & 0.2 & 0.2 \\ sunny & 0.1 & 0.5 & 0.4 \\ cloudy & 0.4 & 0.1 & 0.5 \end{array} \qquad B = \begin{array}{c|cc|} & High & Low \\ \hline rainy & 0.2 & 0.8 \\ sunny & 0.9 & 0.1 \\ cloudy & 0.3 & 0.7 \end{array}$$

Tasks Concerning HMMs

Recall:
π: parametrizes $\Pr(Z_1)$
A: parametrizes $\Pr(Z_{t+1}|Z_t)$
B: parametrizes $\Pr(X_t|Z_t)$

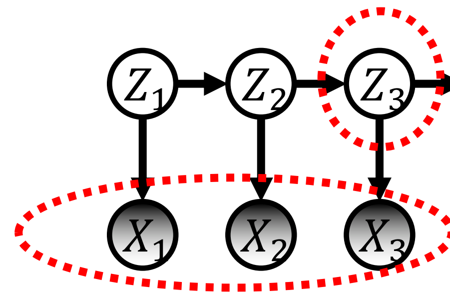
- Inference:
  - We let model parameters be fixed (e.g. tuned by an expert).
  - We seek to find some information from the posterior distribution $\Pr(Z_{1:T}|X_{1:T})$.
  - Examples:
    - Filtering / Smoothing (forwards – backwards)
    - MAP inference (Viterbi)



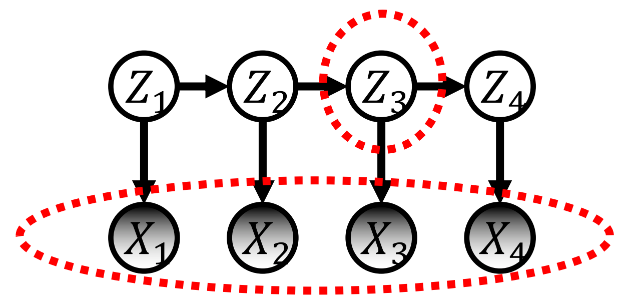
- Parameter Learning:
  - We seek to learn model parameters
  - $X_{1:T}$ is observed
  - $Z_{1:T}$ is (usually) not observed

Inference for HMMs

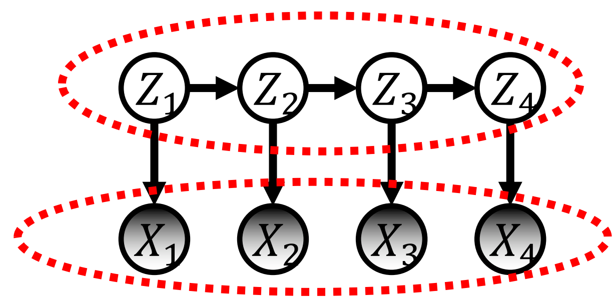
- Filtering: computes the belief state $\Pr(Z_t | X_{1:t})$ incrementally as the data streams-in, i.e., online setting.
  - Infers Z_t using the observations up to time-step t.



- 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 $\mathrm{P}(Z_t | X_{1:t})$
  - 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:

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

  - Hence, we have:

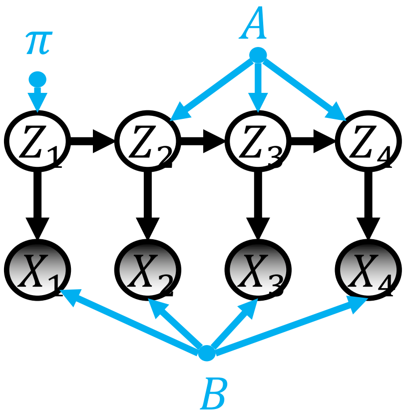
$$P(Z_t = k | X_{1:t}) = \frac{\alpha_t(k)}{sum(\boldsymbol{\alpha}_t)}$$

- The **Forward algorithm** computes recursively the parameters:
  1. Compute $\boldsymbol{\alpha}_1$ (initialisation)
  2. Given $\boldsymbol{\alpha}_t$, compute $\boldsymbol{\alpha}_{t+1}$ (recursion)

The Forwards Algorithm - Initialisation

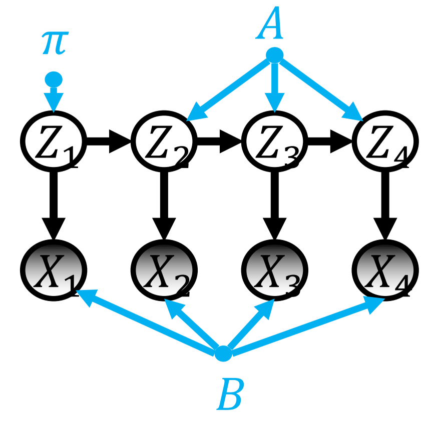
- Initialisation: The computation of the parameters $\boldsymbol{\alpha}_1$ can be done directly

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



The Forwards Algorithm - Recursion

- Recursion: Given $\boldsymbol{\alpha}_t$, we can compute $\boldsymbol{\alpha}_{t+1}$



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

The Forwards Algorithm (cont.)

- Writing the last equation using matrix operators:

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

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{B}_{:(x_{t+1})} \odot (\boldsymbol{A}^T \boldsymbol{\alpha}_t)$$

Notation:
$\odot$ denotes
Hadamard product

$$\begin{bmatrix} \alpha_{t+1}(1) \\ \alpha_{t+1}(2) \\ \vdots \\ \alpha_{t+1}(K) \end{bmatrix} = \begin{bmatrix} B(1, x_{t+1}) \\ B(2, x_{t+1}) \\ \vdots \\ B(K, x_{t+1}) \end{bmatrix} \odot \left(\boxed{A^T} \begin{bmatrix} \alpha_t(1) \\ \alpha_t(2) \\ \vdots \\ \alpha_t(K) \end{bmatrix} \right)$$

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

The Forward-Backwards Algorithm

- Goal: incrementally compute $\mathrm{P}(Z_t|X_{1:T})$
  - The Bayes rule gives:

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

  - For convenience, we denote also:

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

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

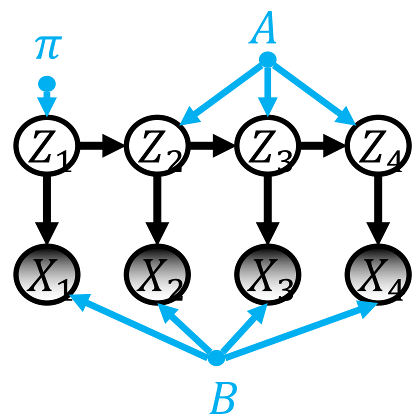
$$P(Z_t = k|X_{1:T}) \propto \alpha_t(k)\beta_t(k)$$

- 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 $\boldsymbol{\beta}_T$ can be done directly

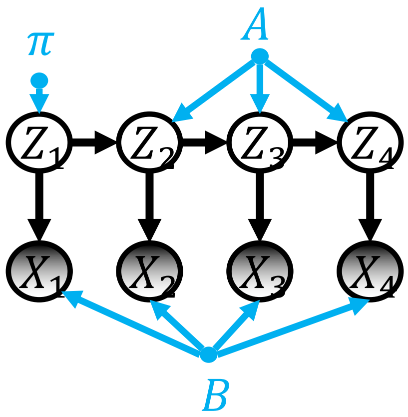
$$\boldsymbol{\beta}_T(k) = 1$$



  - 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 $\boldsymbol{\beta}_{t+1}$, we can compute $\boldsymbol{\beta}_t$



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

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

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

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

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

The Backward Algorithm (cont.)

- Writing the last equation using matrix operators:

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

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

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

- 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:

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

The Forward-Backward Algorithm – Applications II

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

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

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

$$\begin{aligned}\mathrm{P}(Z_t = i, Z_{t+1} = j, X_{1:T}) = &\ \mathrm{P}(Z_t = i, X_{1:t}) \cdot \mathrm{P}(Z_{t+1} = j | Z_t = i, X_{1:t}) \\ &\cdot \mathrm{P}(X_{t+2:T} | Z_t = i, X_{1:t}, Z_{t+1} = j) \cdot \mathrm{P}(\mathrm{X}_{t+1} | Z_t = i, X_{1:t}, Z_{t+1} = j, X_{t+2:T})\end{aligned}$$

$$\Downarrow$$

$$\xi_t(i,j) \propto \alpha_t(i) \cdot A_{ij} \cdot \beta_{t+1}(j) \cdot B_{j x_{t+1}}$$

MAP Inference in HMMs

- 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 $\Pr(Z_{1:T} | X_{1:T})$

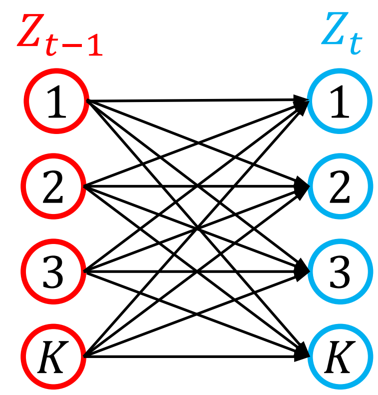
$$\arg\max_Z \; \mathrm{P}(Z_{1:T} | X_{1:T}) = \arg\max_Z \log[\mathrm{P}(Z_{1:T}, X_{1:T})]$$

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

- Each term $\log[\mathrm{P}(Z_t | Z_{t-1})\,\mathrm{P}(X_t | Z_t)]$ depends on values of Z_{t-1} and Z_t.
  - Think of it as a bi-partite graph.

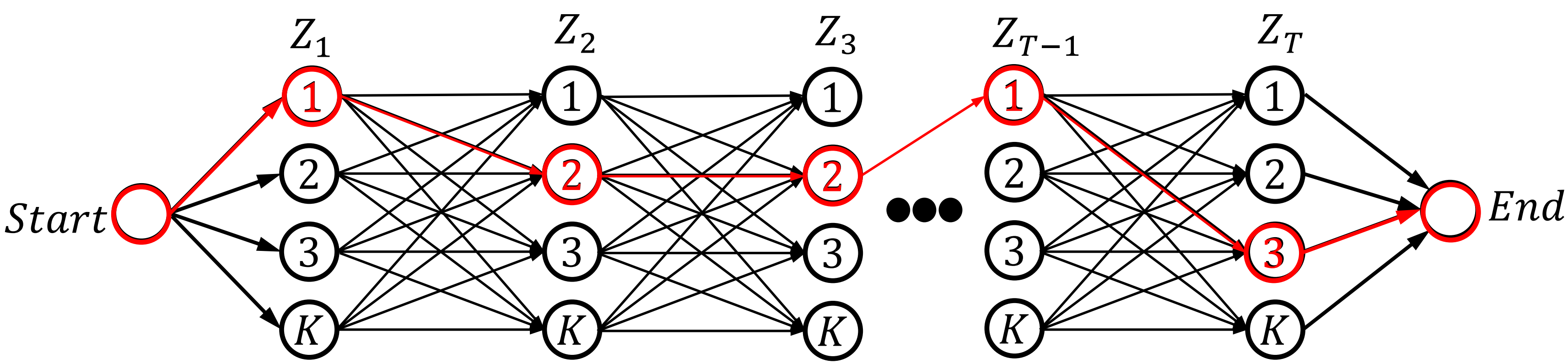
    weight of the edge $(i - j) =$

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



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

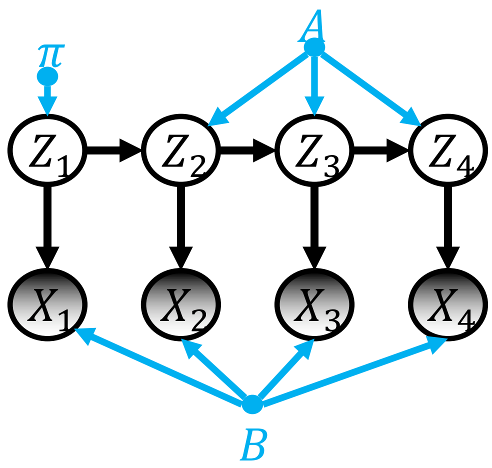


Each directed path corresponds to an assignment to variables $Z_{1:T}$.
Sum of edge weights $= -log Pr(Z_{1:T}, X_{1:T})$

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 $\boldsymbol{X}$.

- We seek to learn model parameters $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}\}$.
- Goal: Solve $\max_{\boldsymbol{\theta}} \log \mathrm{P}(\boldsymbol{X}|\boldsymbol{\theta})$
- Problem: No analytical solution due to marginalization
  - $\log p_{\boldsymbol{\theta}}(\boldsymbol{X}) = \log \sum_{\boldsymbol{Z}} \mathrm{P}(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\theta}) \cdot \mathrm{P}(\boldsymbol{Z}|\boldsymbol{\theta})$
- We encountered this problem before! (Gaussian mixture models)
- You know how to (approximately) solve it! EM algorithm!

Recap: EM algorithm

- Problem: $\max_{\boldsymbol{\theta}} \log \mathrm{P}(\boldsymbol{X}|\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \log \sum_{\boldsymbol{Z}} \mathrm{P}(\boldsymbol{X}|\boldsymbol{Z},\boldsymbol{\theta}) \cdot \mathrm{P}(\boldsymbol{Z}|\boldsymbol{\theta})$
- We can iterate between two steps to maximize a lower bound on our objective
  - E step - evaluate the posterior $\mathrm{P}(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}^{old})$
    - Here, $\boldsymbol{\theta}^{old}$ is the value of $\boldsymbol{\theta}$ from the previous iteration
  - M step – maximize the expected joint log-likelihood $\log \mathrm{P}(\boldsymbol{X},\boldsymbol{Z}|\boldsymbol{\theta})$ under the old posterior w.r.t. $\boldsymbol{\theta}$
    - $\boldsymbol{\theta}^{new} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \mathbb{E}_{P(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}^{old})}[\log p(\boldsymbol{X},\mathbf{Z}|\boldsymbol{\theta})]$
- The M step is equivalent to maximizing the evidence lower bound (ELBO), which is a lower bound on $\log \mathrm{P}(\boldsymbol{X}|\boldsymbol{\theta})$
  - For more details, see our lecture on variational inference in "Advanced Machine Learning: Deep Generative Models" (CIT4230003)

Parameter Learning – E step

- For HMMs, the posterior is:

$$\mathrm{P}\left(Z_{1:T} \middle| X_{1:T}, \boldsymbol{\theta}^{old}\right) = \frac{\mathrm{P}\left(X_{1:T} \middle| Z_{1:T}, \boldsymbol{\theta}^{old}\right)\mathrm{P}\left(Z_{1:T} \middle| \boldsymbol{\theta}^{old}\right)}{\sum_{Z_{1:T}} \mathrm{P}\left(X_{1:T} \middle| Z_{1:T}, \boldsymbol{\theta}^{old}\right)\mathrm{P}\left(Z_{1:T} \middle| \boldsymbol{\theta}^{old}\right)}$$

with

$$\mathrm{P}\left(X_{1:T} \middle| Z_{1:T}, \boldsymbol{\theta}^{old}\right)\mathrm{P}\left(Z_{1:T} \middle| \boldsymbol{\theta}^{old}\right) = \prod_{t=1}^{T} P\left(X_t \middle| Z_t, \boldsymbol{\theta}^{old}\right) \cdot \prod_{t=2}^{T} P\left(Z_t \middle| Z_{t-1,}, \boldsymbol{\theta}^{old}\right) \cdot P\left(Z_1 \middle| \boldsymbol{\theta}^{old}\right)$$

- Important fact: The posterior does not factorize
  - i.e. $\mathrm{P}(Z_{1:T} | X_{1:T}, \boldsymbol{\theta}^{old})$ does **not** have one factor per Z_t
  - This is different from GMMs, where we had one term per sample
  - still, we do **not** have an exponential blow up $O(K^T)$; only $O(TK^2)$

Parameter Learning – M step

- For HMMs, the expected joint log-likelihood is:

$$\mathbb{E}_{\mathrm{P}(Z_{1:T}|X_{1:T},\boldsymbol{\theta}^{old})}[\log P(X_{1:T}, Z_{1:T}|\boldsymbol{\theta})] = \sum_k \mathrm{P}(Z_1 = k|X_{1:T}, \boldsymbol{\theta}^{old}) \log(\pi_k)$$

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

$$+ \sum_i \sum_t \mathrm{P}(Z_t = i|X_{1:T}, \boldsymbol{\theta}^{old}) \mathbb{I}(x_t = j) \log(B_{ij})$$

- Thanks to the Forward-Backward algorithm, the blue terms can be computed efficiently and in closed form

Parameter Learning – M step

- For HMMs, the expected joint log-likelihood is:

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

- We can now solve $\boldsymbol{\theta}^{new} = \mathrm{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{P(Z_{1:T}|X_{1:T},\boldsymbol{\theta}^{old})}[\log P(X_{1:T}, Z_{1:T}|\boldsymbol{\theta})]$
  - you could use projected gradient ascent or (since available here) the closed-form solution for $\boldsymbol{\theta}^{new}$
- This EM algorithm for HMMs is also called the Baum-Welch algorithm

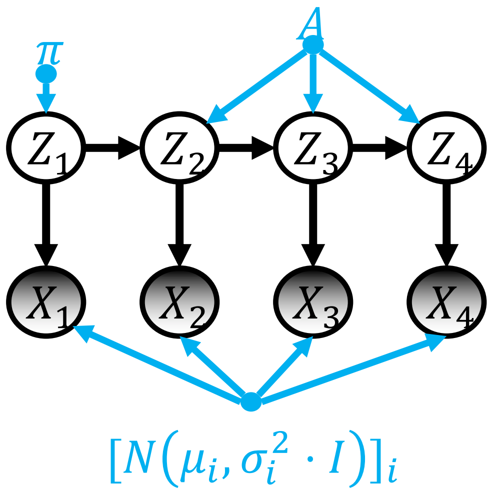
Hidden Markov Models – Continuous Data

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

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

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

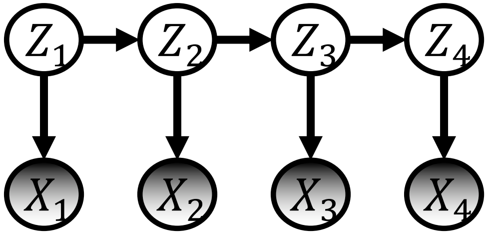
$$P(Z_1 = i) = \pi_i$$
$$P(Z_{t+1} = j | Z_t = i) = A_{ij}$$
$$P(X_{t+1} = x | Z_{t+1} = i) = N(x | \mu_i, \sigma_i^2 \cdot I)$$

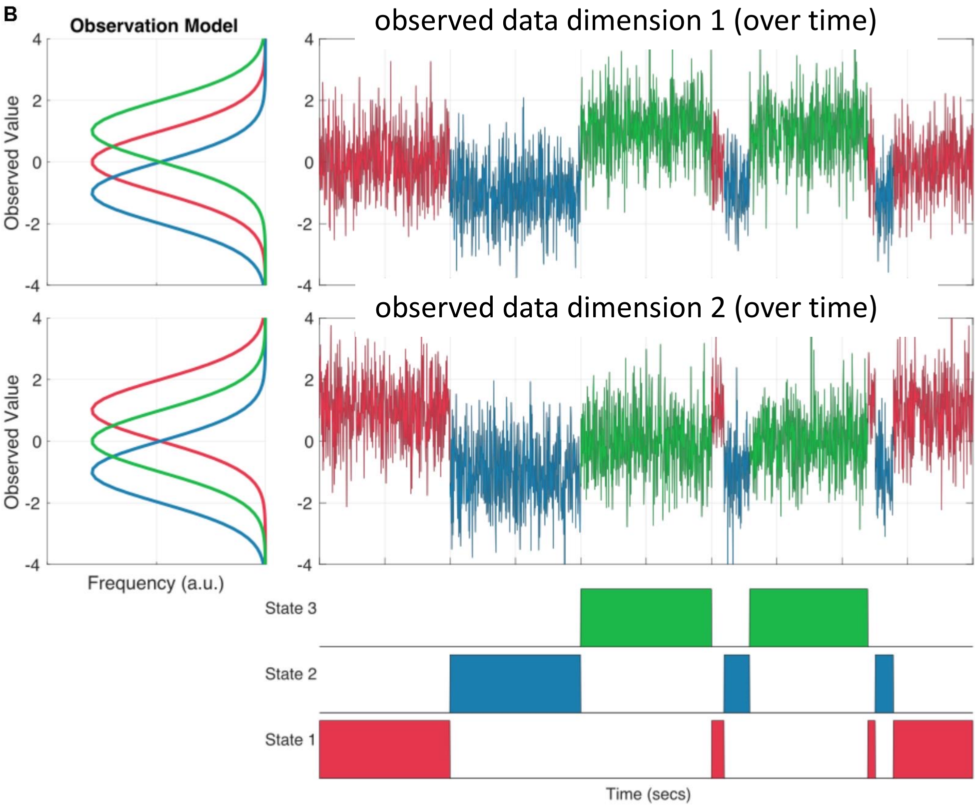


Hidden Markov Models – Continuous Data

- 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





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

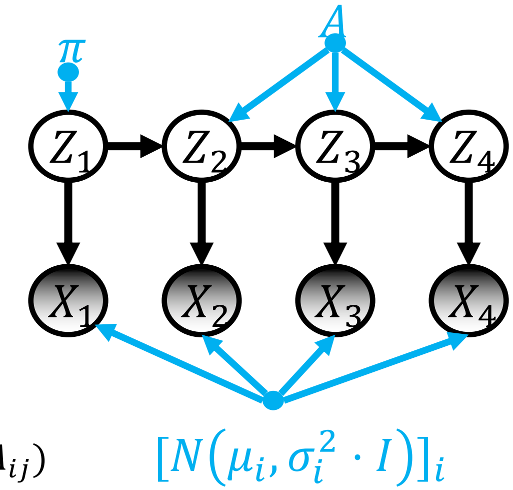
Hidden Markov Models – Continuous Data

- 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.:

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

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

Parameter Learning – Continuous Case



- The expected joint log-likelihood is:

$$E_{P(Z_{1:T}|X_{1:T}\boldsymbol{\theta}^{old})}[lnP(X_{1:T}, Z_{1:T}|\boldsymbol{\theta})] = \sum_k \mathrm{P}(Z_1 = k|X_{1:T}, \boldsymbol{\theta}^{old}) \log(\pi_k)$$

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

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

- Again one can solve $\boldsymbol{\theta}^{new} = \mathrm{argmax}_{\boldsymbol{\theta}} \mathbb{E}_{P(Z_{1:T}|X_{1:T},\boldsymbol{\theta}^{old})}[\log P(X_{1:T}, Z_{1:T}|\boldsymbol{\theta})]$ 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.

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

  - GMM: observations X_i are independent
  - HMM: observations X_t are conditional independent given $\boldsymbol{Z}$
    for the estimate above, we assumed $P(Z_{1:T}|X_{1:T}, \boldsymbol{\theta}^{old})$ is fixed/given

Overview of Tasks concerning HMMs

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

$T = sequence\ length$

$K = \#possible\ states\ for\ Z_t$

Questions – HMM

1. Does the sequence $[Z_1, \dots, Z_T]$ fullfill the Markov property ? Why ?

2. Does the sequence $[X_1, \dots, X_T]$ fullfill the Markov property ? Why ?

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

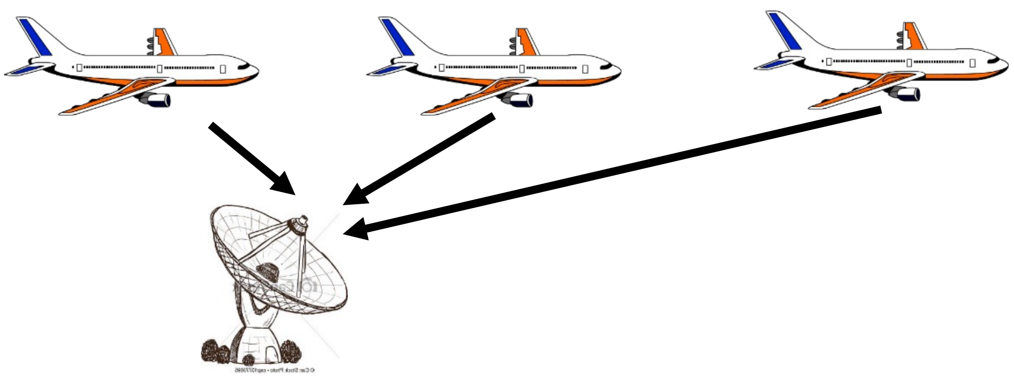
- Example: Tracking
  - Z_t : physical vector quantities (e.g. position, velocity, etc.) at time-step t
  - X_t: observed noisy measurements of airplane location at time-step t

$$\Pr(Z_{t+1}|Z_t) = N(Z_{t+1}|f(Z_t), \sigma_z^2)$$

f: given Z_t predicts Z_{t+1} ,e.g., using laws of motion

$$\Pr(X_t|Z_t) = N(X_t|g(Z_t), \sigma_x^2)$$

g: sensor measurement based on Z_t



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

Neural Hidden Markov Models / Deep State Space Models

$$\Pr(Z_{t+1}|Z_t) = N(Z_{t+1}|f(Z_t), \sigma_z^2)$$

$$\Pr(X_t|Z_t) = N(X_t|g(Z_t), \sigma_x^2)$$

- There are no constraints on f and g
- In particular, they can be neural networks f_ψ and g_ϕ with parameters ψ, ϕ
- $\boldsymbol{\theta} = \{\psi, \phi\}$ can then be optimized via gradient ascent in the M step of the EM algorithm
- This principle can also be applied to non-Gaussian distributions

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