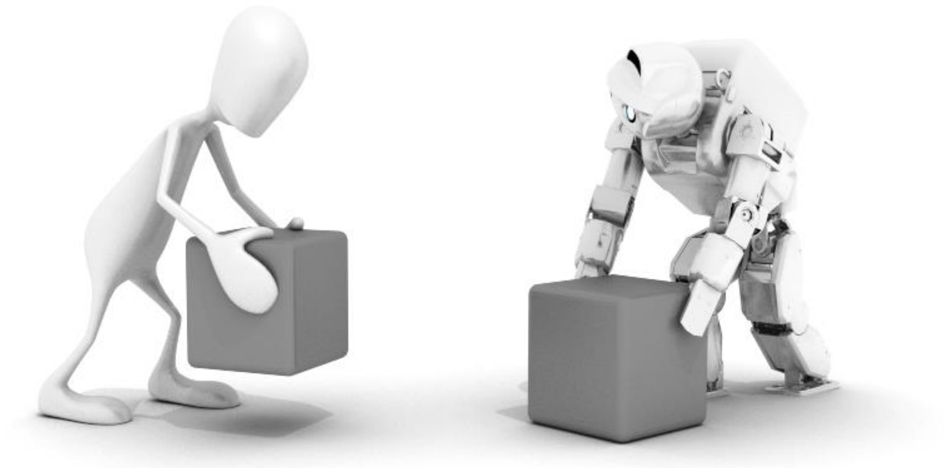
EE613 - Machine Learning for Engineers

HIDDEN MARKOV MODELS

Sylvain Calinon
Robot Learning and Interaction Group
Idiap Research Institute
Oct 12, 2023

Robot Learning and Interaction group

Idiap Research Institute



Learning from demonstration

EE613 labs:

Tobias Löw

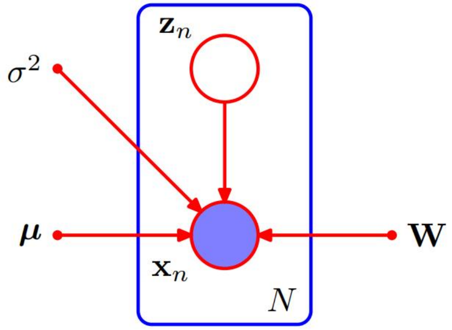
EE613 schedule

Thu. 21.09.2023	(C) 1. ML introduction
Thu. 28.09.2023	(C) 2. Bayesian 1 (C) 3. Bayesian 2
Thu. 12.10.2023	(C) 4. Hidden Markov Models
Thu. 19.10.2023	(C) 5. Dimensionality reduction
Thu. 26.10.2023	(C) 6. Decision trees
Thu. 02.11.2023	(C) 7. Linear regression
Thu. 09.11.2023	(C) 8. Nonlinear regression
Thu. 16.11.2023	(C) 9. Kernel Methods - SVM
Thu. 23.11.2023	(C) 10. Tensor factorization
Thu. 30.11.2023	(C) 11. Deep learning 1
Thu. 07.12.2023	(C) 12. Deep learning 2
Thu. 14.12.2023	(C) 13. Deep learning 3
Thu. 21.12.2023	(C) 14. Deep learning 4

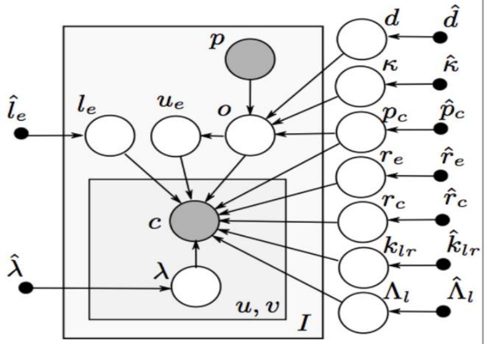
Recap from 28/09 lecture (Jean-Marc)

Overview

- Graphical models fundamentals
  - Bayesian networks, representations
  - conditional independence
  - undirected graphical models

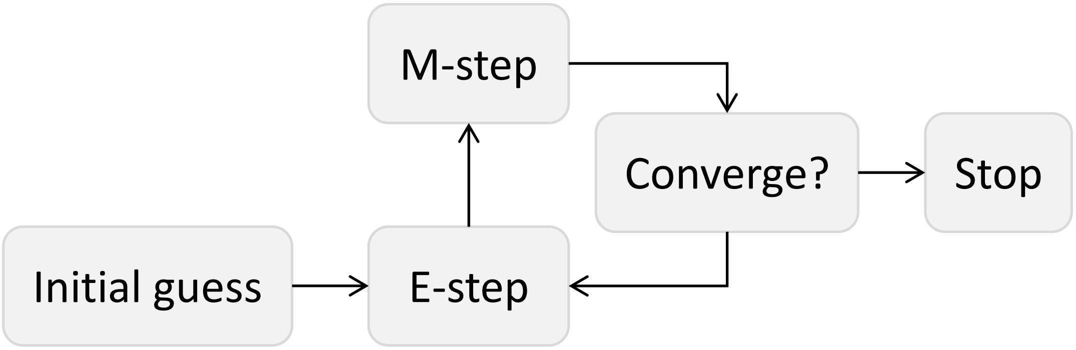


- Learning
  - ML, MAP, Bayesian
  - the EM algorithm, latent variable models
    - Gaussian Mixture Model (GMM)
    - Hidden Markov Model (HMM)

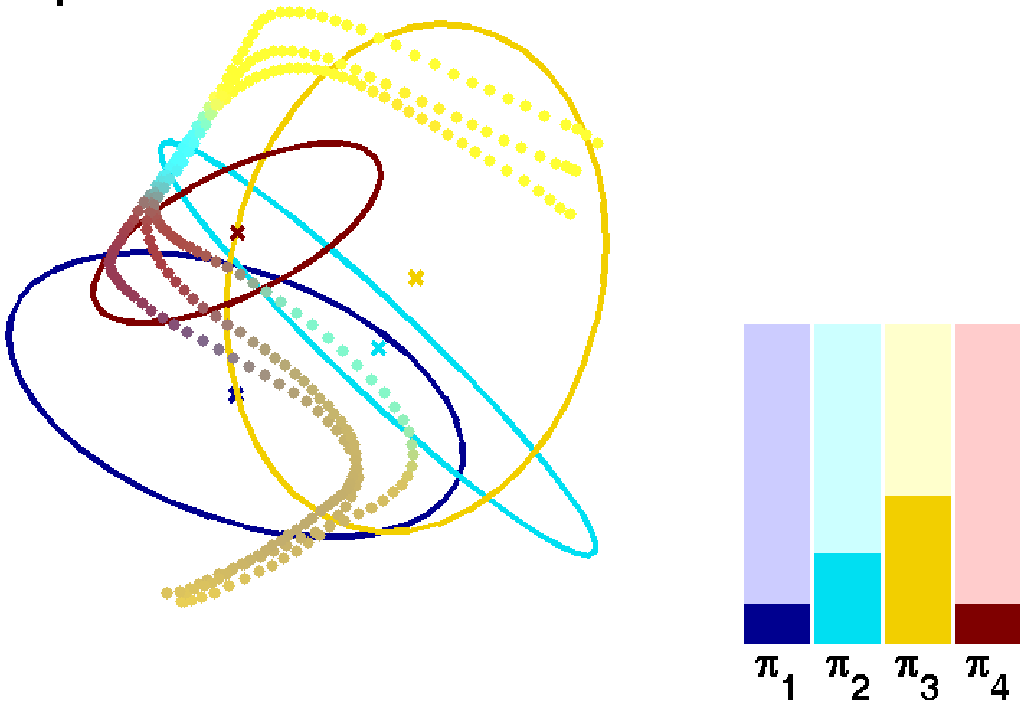


- Later : PCA, Probabilistic PCA

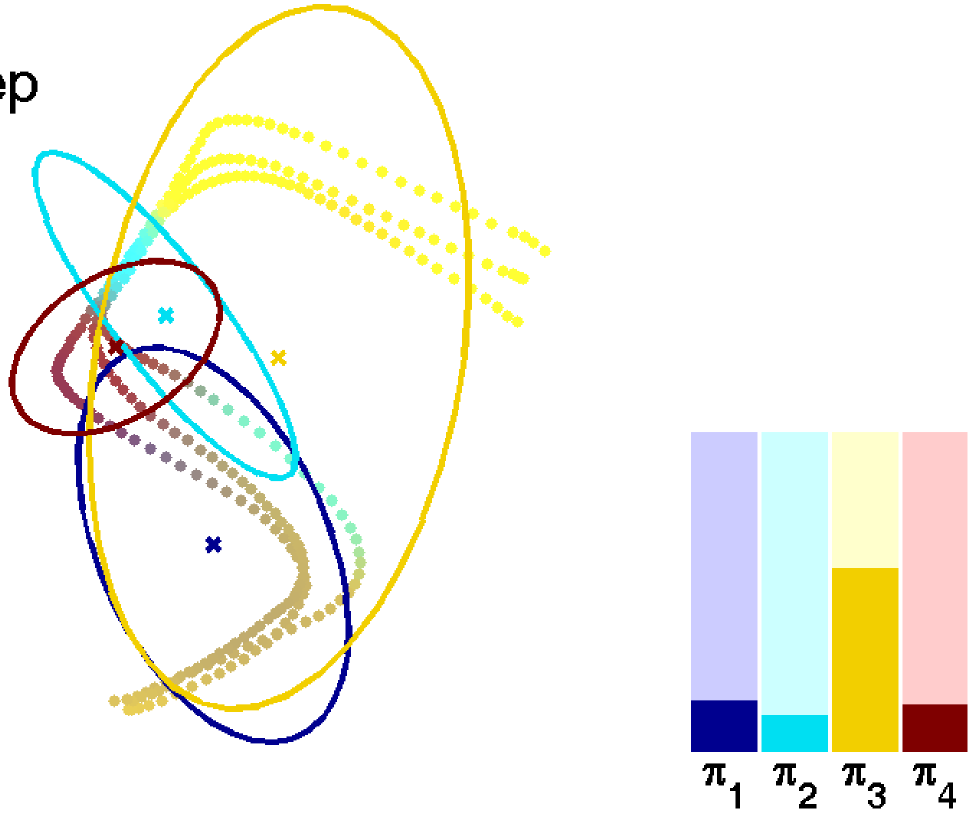
Recap: EM for GMM



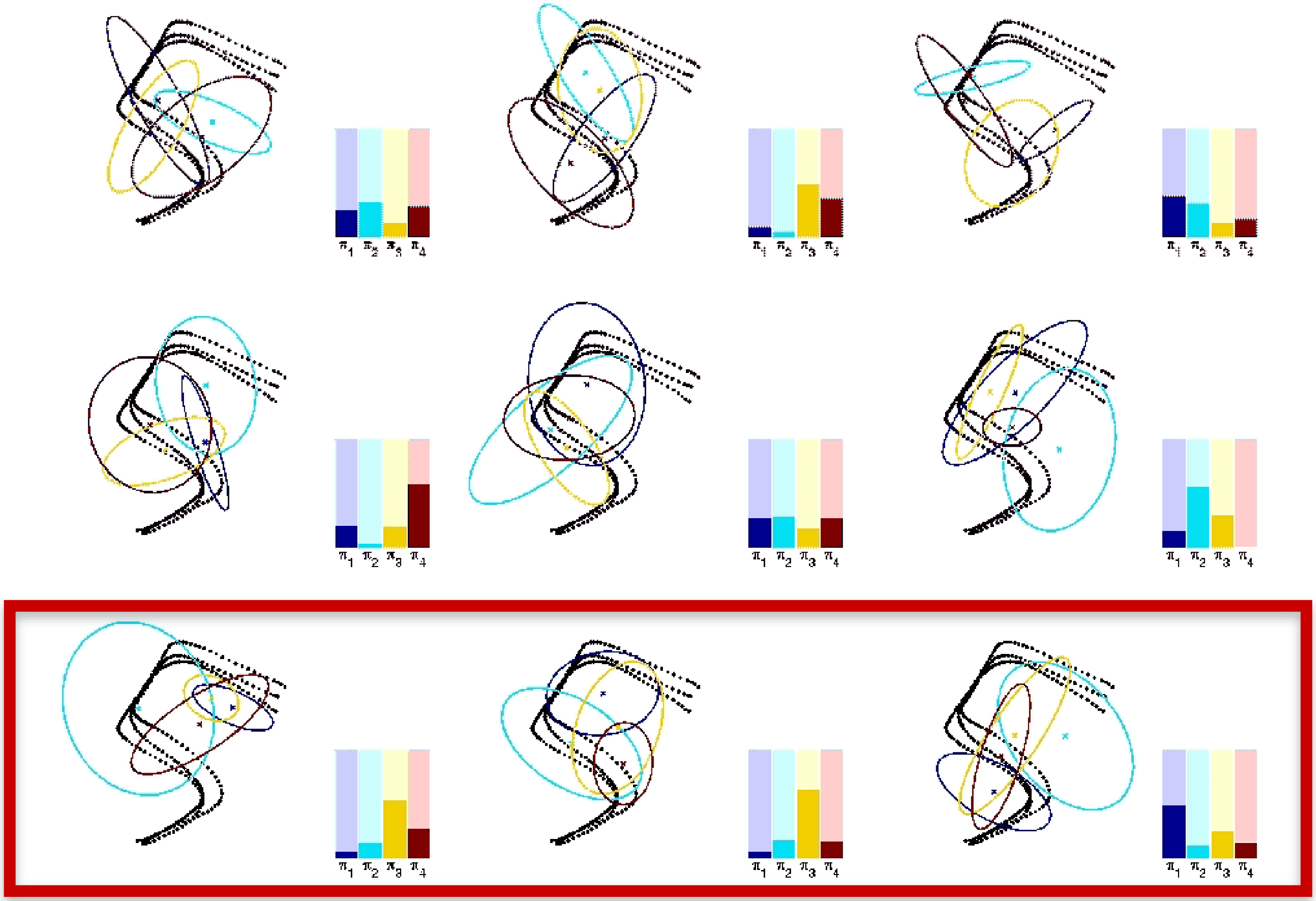
E-step



M-step

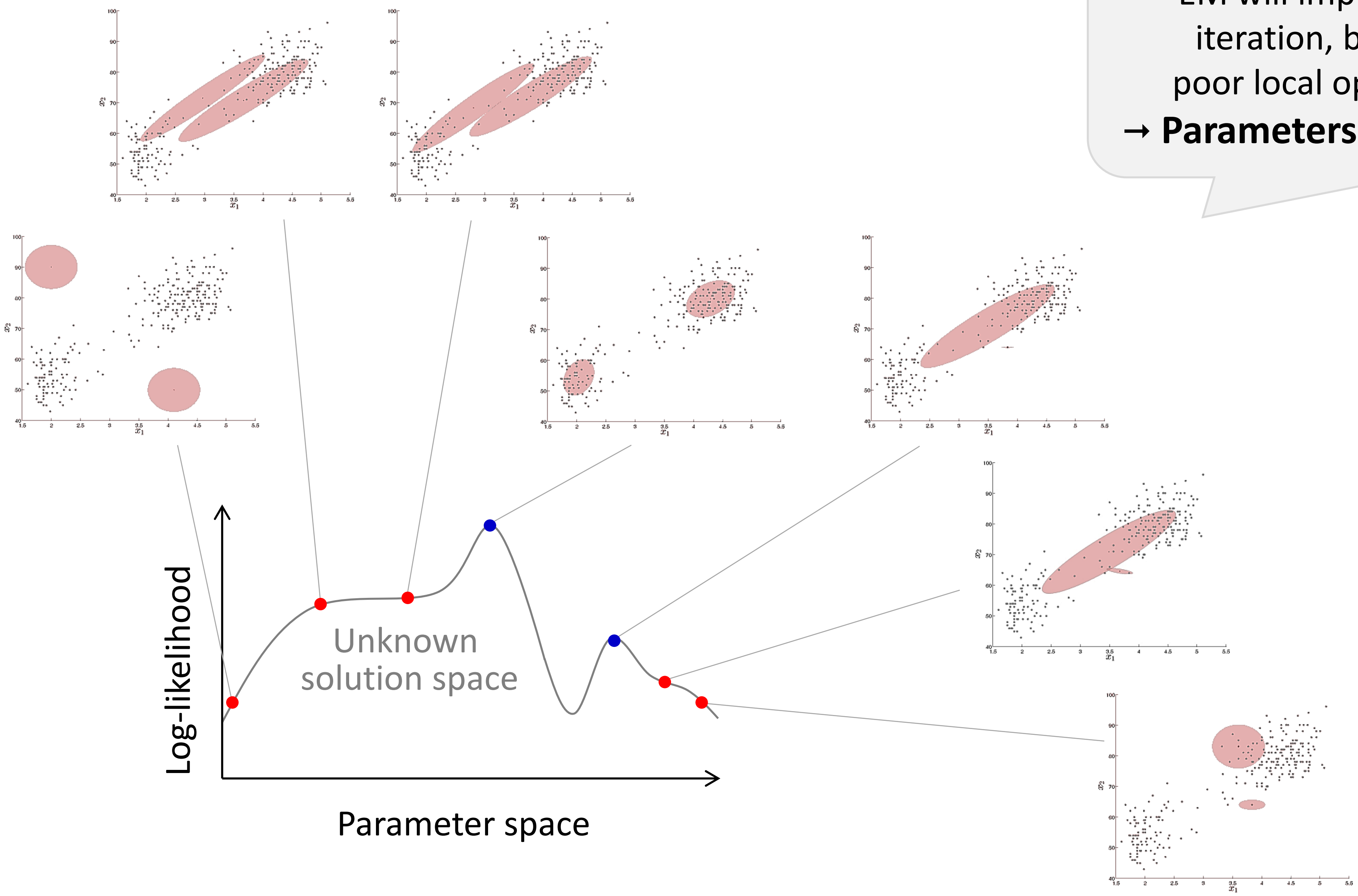


Recap: Local optima issue when using EM for GMM



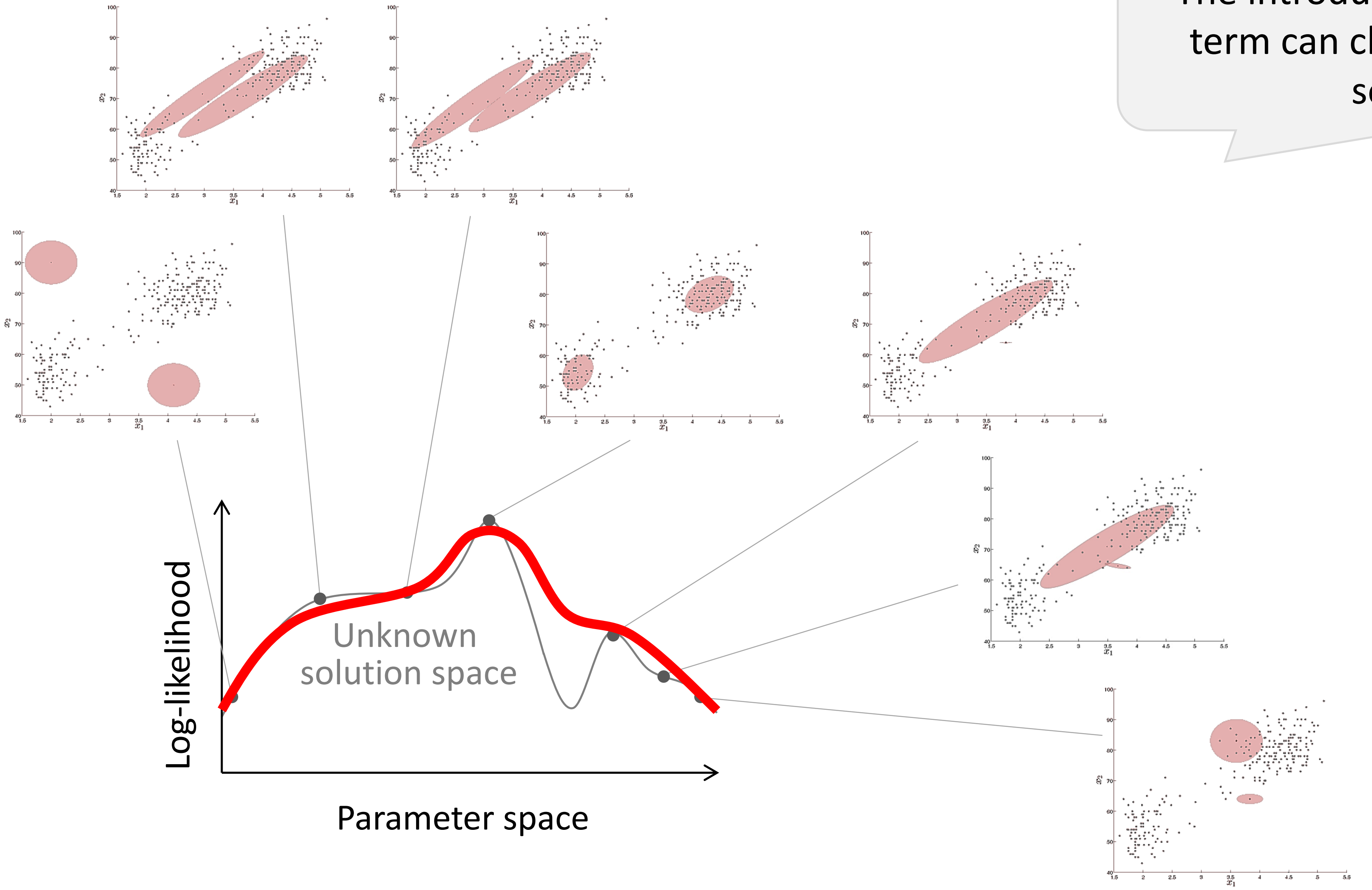
Recap: Local optima issue when using EM for GMM

EM will improve the likelihood at each iteration, but it can get trapped into poor local optima in the solution space
→ **Parameters initialization is important!**

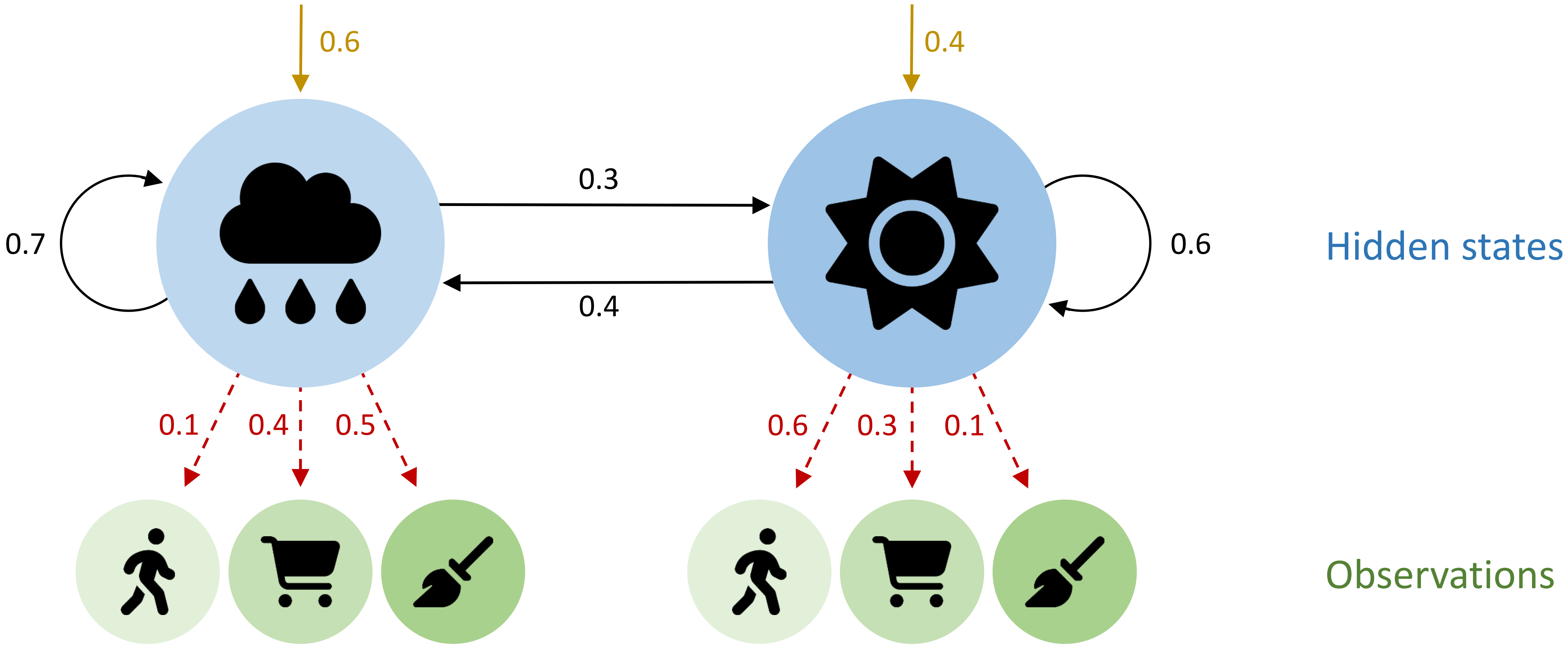


Recap: Local optima issue when using EM for GMM

The introduction of a regularization term can change the shape of the solution space

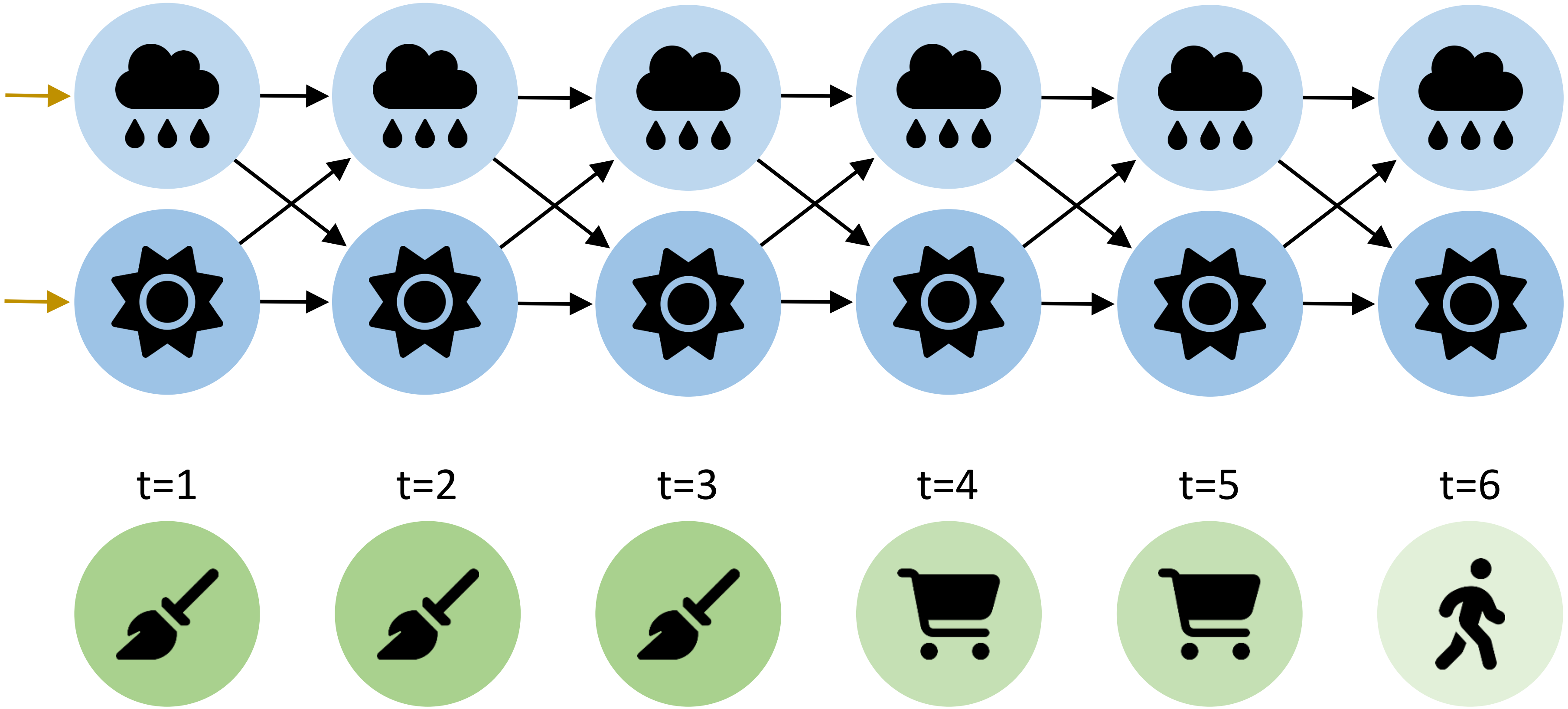


Hidden Markov Model (HMM)



Lattice representation of a sequence of observations (with the possible associated hidden states)

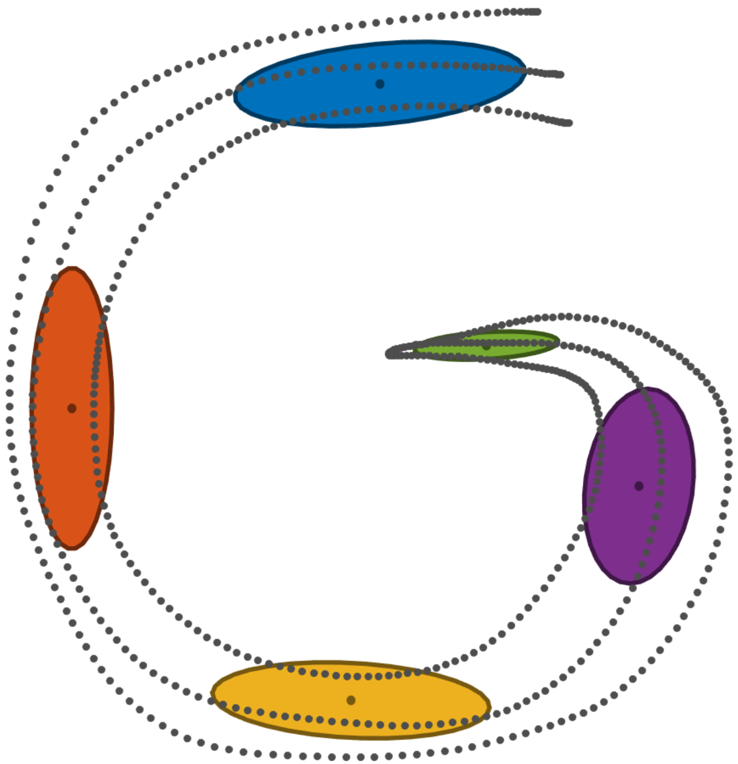
Dictionary of possible observations

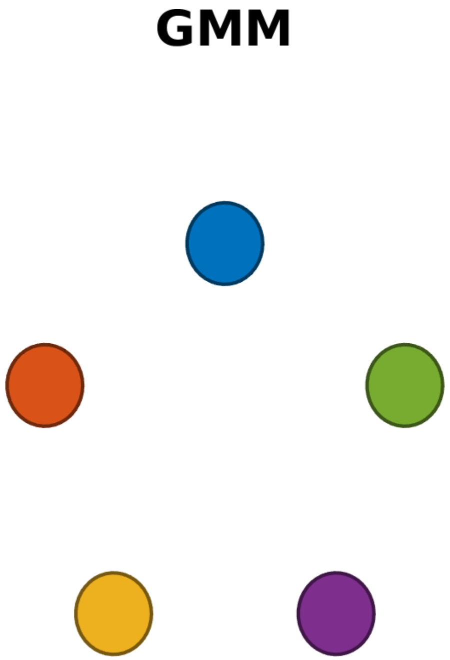


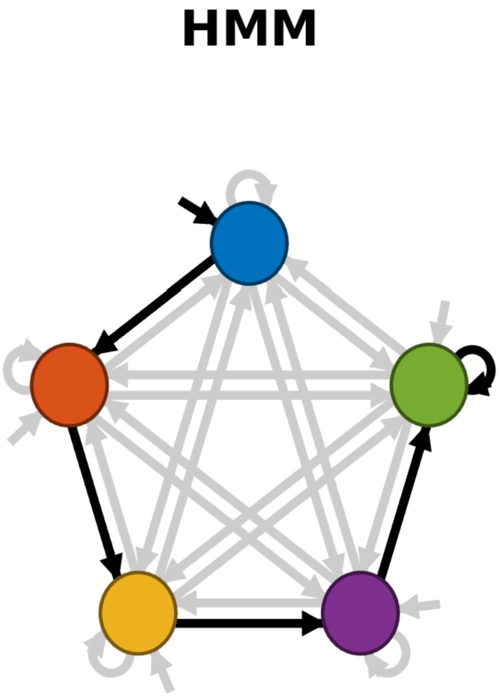
Options for the underlying hidden states $s_{1:6}$

Observed sequence $\xi_{1:6}$

GMM vs HMM







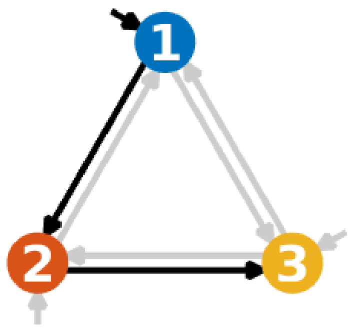
→ You can think of an HMM either as:

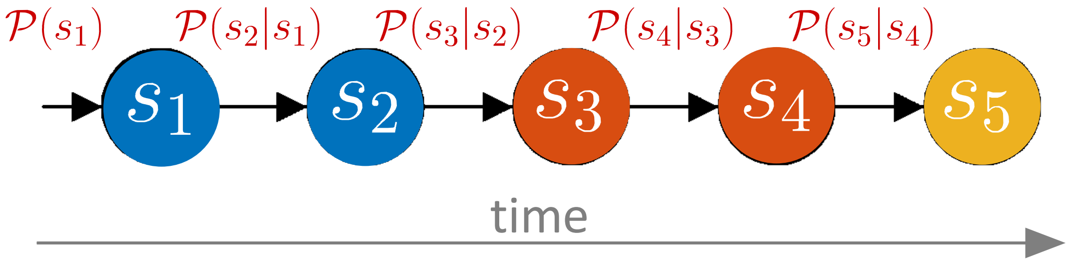
- a Markov chain with stochastic measurements
- a GMM with transition between the Gaussians

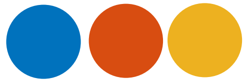
Outline of today's lecture

- Markov models
- Hidden Markov model (HMM)
- Forward-backward algorithm
- Viterbi decoding (dynamic programming)
- Hidden semi-Markov model (HSMM)
- HMM with dynamic features (Trajectory-HMM)

Markov models





Dictionary: 

With a **first order Markov model**, the joint distribution of a sequence of states is assumed to be of the form

$$\mathcal{P}(s_1, s_2, \ldots, s_T) = \mathcal{P}(s_1) \prod_{t=2}^{T} \mathcal{P}(s_t|s_{t-1})$$

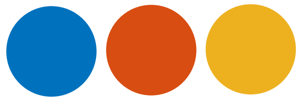
and we thus have

$$\mathcal{P}(s_t|s_1, s_2, \ldots, s_{t-1}) = \mathcal{P}(s_t|s_{t-1})$$

In most applications, the conditional distributions $\mathcal{P}(s_t|s_{t-1})$ will be assumed to be **stationary** (**homogeneous Markov chain**).

Markov models – Parameters

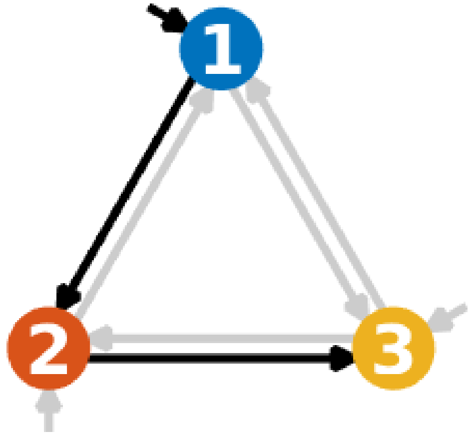
K *possible states*



Dictionary

The **initial state distribution** is defined by

$$\Pi_i = \mathcal{P}(s_1 = i) \quad \text{with} \quad \sum_{i=1}^{K} \Pi_i = 1$$



A **transition matrix** $\boldsymbol{A}$ is defined, with elements

$$a_{i,j} = \mathcal{P}(s_{t+1} = j \mid s_t = i)$$

defining the probability of getting from state i to state j in one step.

$$\begin{array}{c|ccc|} & 1 & 2 & 3 \\ \hline 1 & a_{1,1} & a_{1,2} & a_{1,3} \\ 2 & a_{2,1} & a_{2,2} & a_{2,3} \\ 3 & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array}$$

Constraint: each row of the matrix sums to one, $\sum_{j=1}^{K} a_{i,j} = 1$.

Example: language modeling

We define the state space to be all the words in some language.

The marginal probabilities $\mathcal{P}(s_t = k)$ are called **unigram** statistics.

For a first-order Markov model, $\mathcal{P}(s_t = k \mid s_{t-1} = j)$ is called a **bigram** model.

For a second-order Markov model, $\mathcal{P}(s_t = k \mid s_{t-1} = j, s_{t-2} = i)$ is called a **trigram** model, etc.

In the general case, these are called ***n*-gram** models.

Example: language modeling

Sentence completion

The model can predict the next word given the previous words in a sentence. This can be used to reduce the amount of typing required (e.g., mobile devices).

Data compression

The model can be used to define an encoding scheme, by assigning codewords to more probable strings. The more accurate the predictive model, the fewer the number of bits is required to store the data.

Text classification

The model can be used as a class-conditional density and/or generative classifier.

Automatic writing

The model can be used to sample from $\mathcal{P}(s_1, s_2, \ldots, s_t)$ to generate artificial text.

Example: language modeling

SAYS IT'S NOT IN THE CARDS LEGENDARY RECONNAISSANCE BY ROLLIE DEMOCRACIES UNSUSTAINABLE COULD STRIKE REDLINING VISITS TO PROFIT BOOKING WAIT HERE AT MADISON SQUARE GARDEN COUNTY COURTHOUSE WHERE HE HAD BEEN DONE IN THREE ALREADY IN ANY WAY IN WHICH A TEACHER …

Example of text generated from a 4-gram model, trained on a corpus of 400 million words.

The first 4 words are specified by hand, the model generates the 5th word, and then the results are fed back into the model.

Source: http://www.fit.vutbr.cz/~imikolov/rnnlm/gen-4gram.txt

MLE of transition matrix in Markov models

A Markov model is described by $\mathbf{\Theta}^{\text{MM}} = \left\{\{a_{i,j}\}_{j=1}^{K}, \Pi_i\right\}_{i=1}^{K}$, where the transition probabilities $a_{i,j}$ are stored in a matrix $\boldsymbol{A}$.

The maximum likelihood estimate (MLE) of the parameters can be computed with the normalized counts

$$\hat{\Pi}_i = \frac{N_i}{\sum_{k=1}^{K} N_k}, \qquad \hat{a}_{i,j} = \frac{N_{i,j}}{\sum_{k=1}^{K} N_{i,k}}$$

These results can be extended to higher order Markov models, but since an n-gram models has $O(K^n)$ parameters, special care needs to be taken with overfitting.

For example, with a bi-gram model and 50,000 words in the dictionary, there are 2.5 billion parameters to estimate, and it is unlikely that all possible transitions will be observed in the training data.

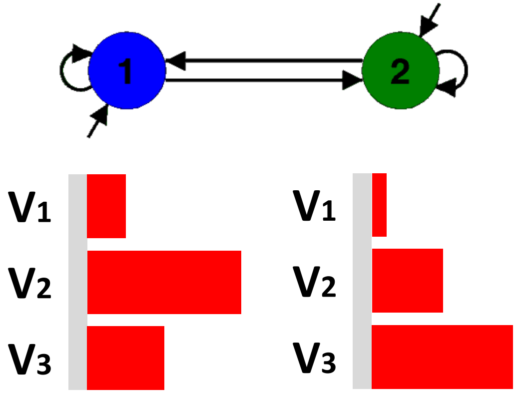
Hidden Markov model (HMM)

Python notebook:
demo_HMM.ipynb

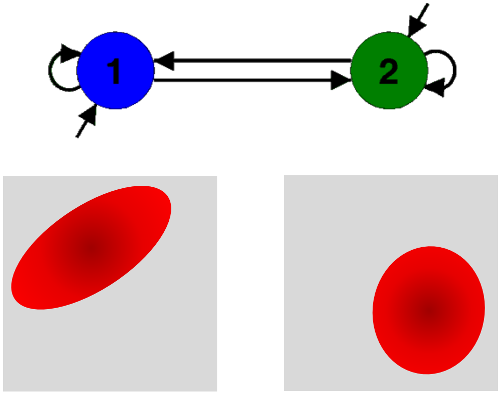
Matlab code:
demo_HMM01.m

Emission/output distributions in HMM

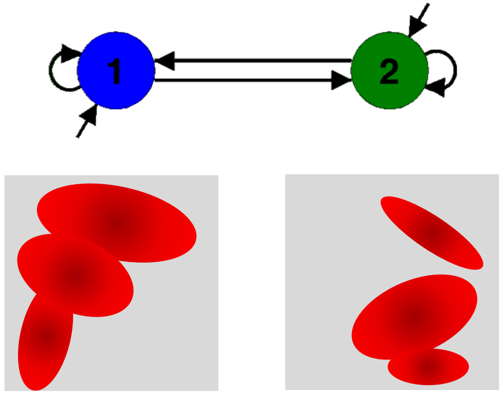
Discrete tables



Gaussian distribution

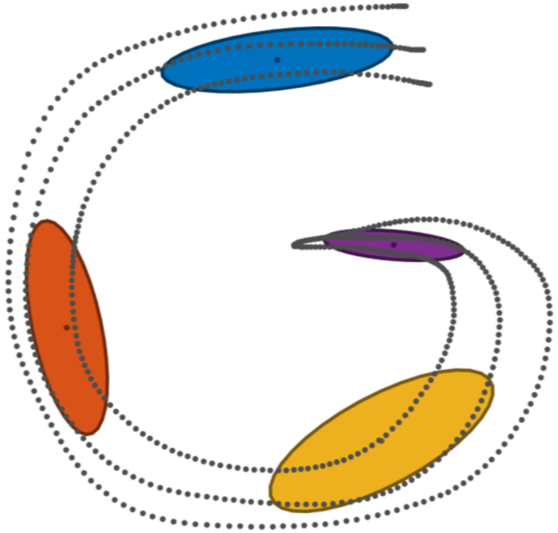


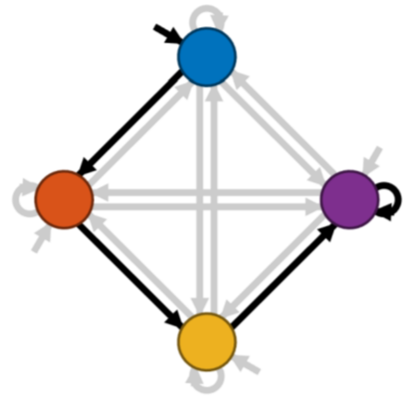
Mixture of Gaussians



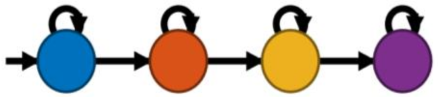
GMM with latent variable $\boldsymbol{z}_t$ depending on the conditional distribution $\mathcal{P}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$

Transition matrix structures in HMM

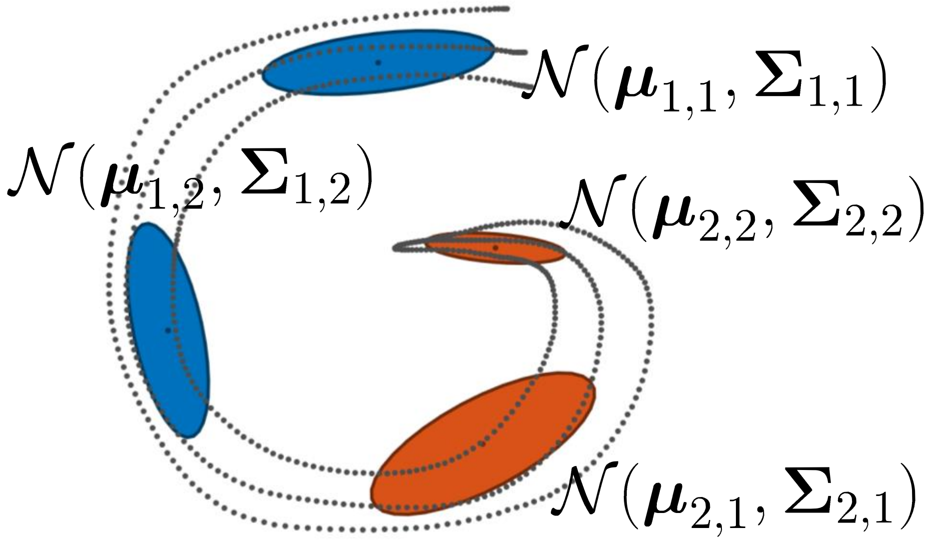




$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix}$$



$$\begin{bmatrix} a_{1,1} & a_{1,2} & 0 & 0 \\ 0 & a_{2,2} & a_{2,3} & 0 \\ 0 & 0 & a_{3,3} & a_{3,4} \\ 0 & 0 & 0 & a_{4,4} \end{bmatrix}$$



$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ 0 & a_{2,2} \end{bmatrix}$$

HMM – Examples of application

HMM is used in many fields as a tool for **time series or sequences analysis**, and in fields where the goal is to recover a data sequence that is not immediately observable:

Speech recognition
Speech synthesis
Part-of-speech tagging
Natural language modeling
Machine translation
Gene prediction
Molecule kinetic analysis
DNA motif discovery
Alignment of bio-sequences (e.g., proteins)
Metamorphic virus detection
Document separation in scanning solutions

Cryptoanalysis
Activity recognition
Protein folding
Human motion science
Online handwriting recognition
Robotics

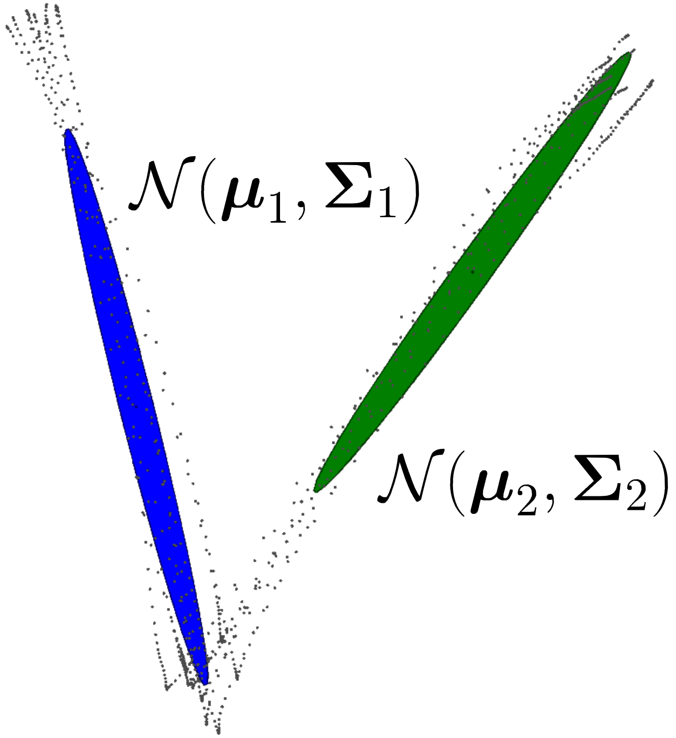
and many, many others...

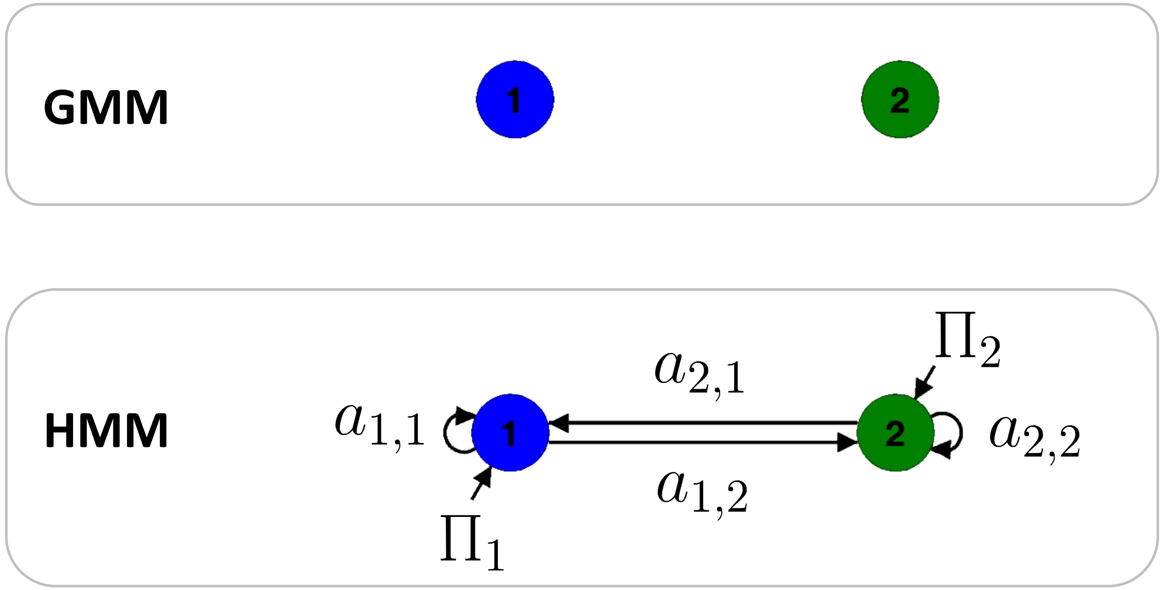
HMM parameters

$$\boldsymbol{\Theta}^{\text{GMM}} = \{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K}$$

$$\boldsymbol{\Theta}^{\text{HMM}} = \{\{a_{i,j}\}_{j=1}^{K}, \Pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K} \qquad \pi_i = 1$$

From now on, we will consider
a single Gaussian as state output





Inference problems associated with HMMs

Probability of an observed sequence

$$\mathcal{P}(\boldsymbol{\xi}_{1:T}) = \mathcal{P}(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_T)$$

Probability of the latent variables

Filtering

$$\mathcal{P}(s_t \,|\, \boldsymbol{\xi}_{1:t}) = \mathcal{P}(s_t \,|\, \boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_t)$$

Prediction

$$\mathcal{P}(s_{t+1} \,|\, \boldsymbol{\xi}_{1:t}) = \mathcal{P}(s_{t+1} \,|\, \boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_t)$$

Smoothing $\rightarrow$ *Forward-backward algorithm*

$$\mathcal{P}(s_t \,|\, \boldsymbol{\xi}_{1:T}) = \mathcal{P}(s_t \,|\, \boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_T)$$

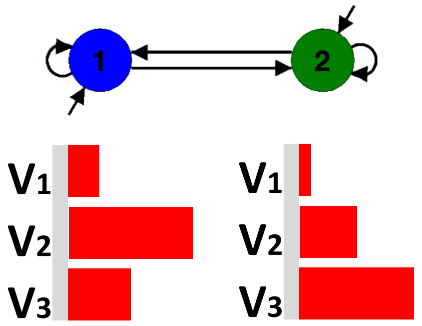
MAP estimation $\rightarrow$ *Viterbi decoding*

$$\mathcal{P}(s_{1:T} \,|\, \boldsymbol{\xi}_{1:T}) = \mathcal{P}(s_1, s_2, \ldots, s_T \,|\, \boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_T)$$

Intermediary variables that we will require in HMM

How to estimate the parameters of an HMM?

$\rightarrow$ Maximum of expected complete data log-likelihood $\mathcal{Q}(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{\text{old}})$



How to compute $\frac{\partial \mathcal{Q}}{\partial \Pi_i}=0$, $\frac{\partial \mathcal{Q}}{\partial a_{i,j}}=0$, $\frac{\partial \mathcal{Q}}{\partial \boldsymbol{\mu}_i}=0$ and $\frac{\partial \mathcal{Q}}{\partial \boldsymbol{\Sigma}_i}=0$?

$\rightarrow$ Requires to compute $\zeta_{t,i,j}^{\text{HMM}} = \mathcal{P}(s_t{=}i, s_{t+1}{=}j \,|\, \boldsymbol{\xi}_{1:T})$

$\rightarrow$ Requires to compute $\gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t{=}i \,|\, \boldsymbol{\xi}_{1:T})$

Observations $\boldsymbol{\xi}_{1:T}$

How to compute $\zeta_{t,i,j}^{\text{HMM}}$ and $\gamma_{t,i}^{\text{HMM}}$?

$\rightarrow$ Requires to compute $\alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t{=}i, \boldsymbol{\xi}_{1:t})$

$\rightarrow$ Requires to compute $\beta_{t,i}^{\text{HMM}} = \mathcal{P}(\boldsymbol{\xi}_{t+1:T} \,|\, s_t{=}i)$

$\max\limits_{\boldsymbol{\Theta}} \mathcal{Q}(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{\text{old}})$

$\zeta_{t,i,j}^{\text{HMM}}$ $\gamma_{t,i}^{\text{HMM}}$

$\alpha_{t,i}^{\text{HMM}}$ $\beta_{t,i}^{\text{HMM}}$

EM for HMM (Baum-Welch algorithm)

E-step: compute $\gamma^{\text{HMM}}_{m,t,i}$ and $\zeta^{\text{HMM}}_{m,t,i,j}$

M-step:

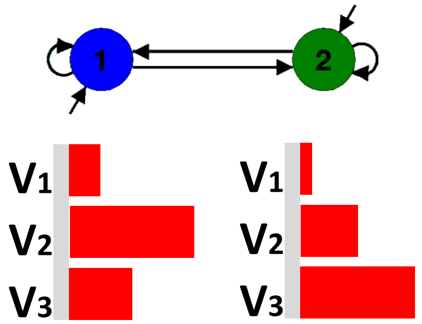
$$\Pi_i \leftarrow \frac{\sum_{m=1}^{M} \gamma^{\text{HMM}}_{m,1,i}}{M} = \frac{\text{Total number of times in i at time step 1}}{\text{Total number of trajectories}}$$

$$a_{i,j} \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m-1} \zeta^{\text{HMM}}_{m,t,i,j}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m-1} \gamma^{\text{HMM}}_{m,t,i}} = \frac{\text{Total number of transitions from i to j}}{\text{Total number of times in i (and transit to anything else)}}$$

$$\boldsymbol{\mu}_i \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma^{\text{HMM}}_{m,t,i}\, \boldsymbol{\xi}_{m,t}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma^{\text{HMM}}_{m,t,i}}$$

result similar to GMM

$$\boldsymbol{\Sigma}_i \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma^{\text{HMM}}_{m,t,i}\, (\boldsymbol{\xi}_{m,t} - \boldsymbol{\mu}_i)(\boldsymbol{\xi}_{m,t} - \boldsymbol{\mu}_i)^{\top}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma^{\text{HMM}}_{m,t,i}}$$



K Gaussians

M sequences

T_m points per sequences

The update rules can be interpreted as normalized counts, with several types of weighted averages required in the computation.

$$\alpha^{\text{HMM}}_{t,i} = \mathcal{P}(s_t\!=\!i, \boldsymbol{\xi}_{1:t})$$

$$\beta^{\text{HMM}}_{t,i} = \mathcal{P}(\boldsymbol{\xi}_{t+1:T} \,|\, s_t\!=\!i)$$

$$\gamma^{\text{HMM}}_{t,i} = \mathcal{P}(s_t\!=\!i \,|\, \boldsymbol{\xi}_{1:T})$$

$$\zeta^{\text{HMM}}_{t,i,j} = \mathcal{P}(s_t\!=\!i, s_{t+1}\!=\!j \,|\, \boldsymbol{\xi}_{1:T})$$

Useful intermediary variables in HMM

Forward variable $\alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i, \boldsymbol{\xi}_{1:t})$

Backward variable $\beta_{t,i}^{\text{HMM}} = \mathcal{P}(\boldsymbol{\xi}_{t+1:T} \mid s_t = i)$

Smoothed node marginals $\gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i \mid \boldsymbol{\xi}_{1:T})$

Smoothed edge marginals $\zeta_{t,i,j}^{\text{HMM}} = \mathcal{P}(s_t = i, s_{t+1} = j \mid \boldsymbol{\xi}_{1:T})$

Forward algorithm

$\alpha^{\text{HMM}}_{t,i} = \mathcal{P}(s_t\!=\!i, \boldsymbol{\xi}_{1:t})$

The probability to be in state i at time step t and to observe $\boldsymbol{\xi}_{1:t}\!=\!\{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_t\}$ can be computed with the **forward variable**

$$\alpha^{\text{HMM}}_{t,i} = \mathcal{P}(s_t\!=\!i, \boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_t) = \mathcal{P}(s_t\!=\!i, \boldsymbol{\xi}_{1:t})$$

which can be used to compute

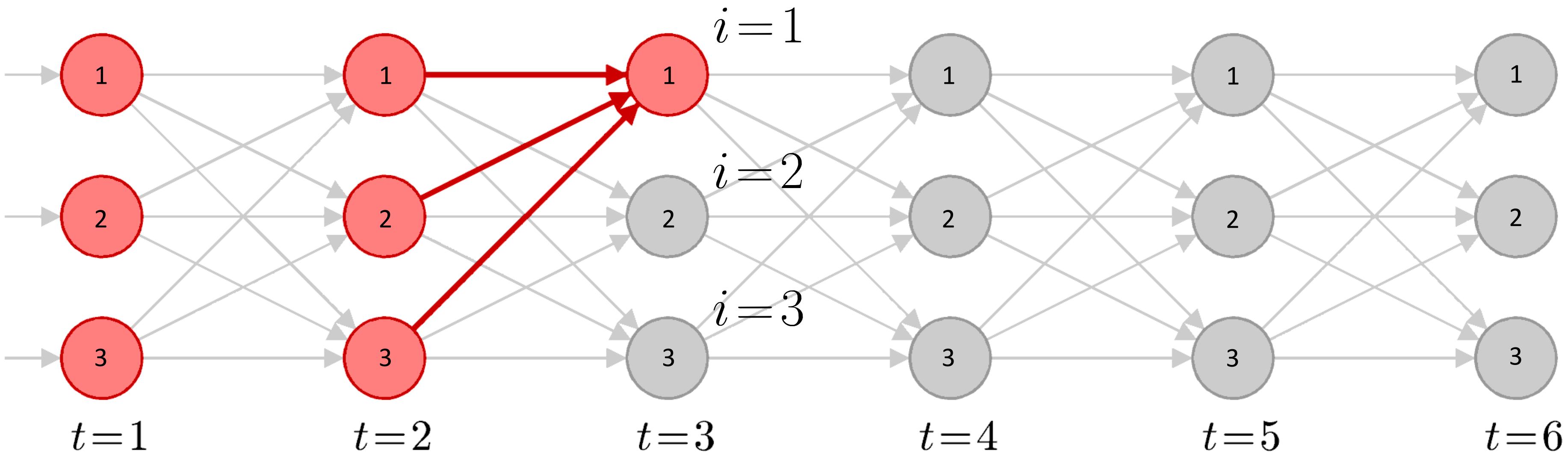
$$\mathcal{P}(s_t\!=\!i \,|\, \boldsymbol{\xi}_{1:t}) = \frac{\mathcal{P}(s_t\!=\!i, \boldsymbol{\xi}_{1:t})}{\mathcal{P}(\boldsymbol{\xi}_{1:t})} = \frac{\alpha^{\text{HMM}}_{t,i}}{\sum_{k=1}^{K} \alpha^{\text{HMM}}_{t,k}}$$

The direct computation would require marginalizing over all possible state sequences $\{s_1, s_2, \ldots, s_{t-1}\}$, which would grow exponentially with t.

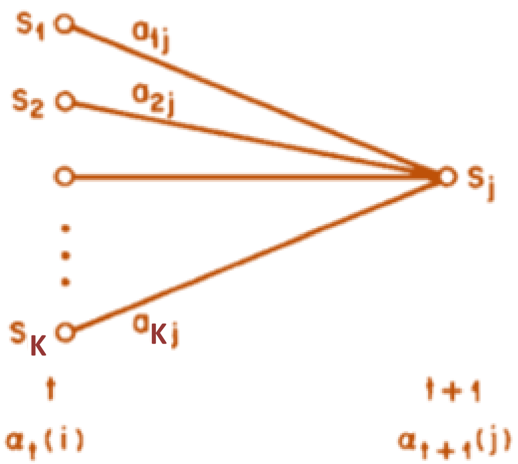
The forward algorithm takes advantage of the conditional independence rules of the HMM to perform the calculation recursively.

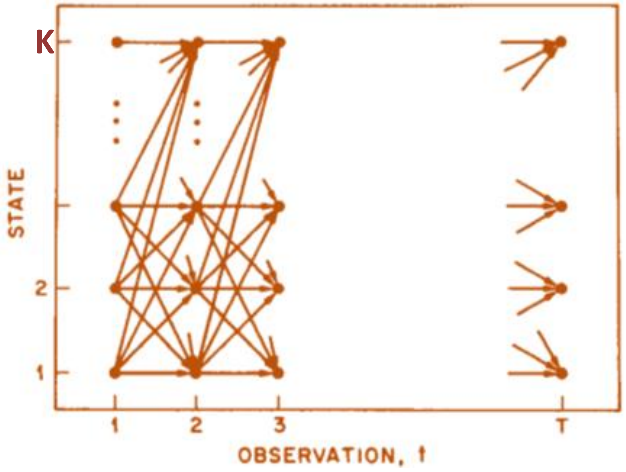
Forward algorithm

$$\alpha^{\text{HMM}}_{t,i} = \mathcal{P}(s_t\!=\!i, \boldsymbol{\xi}_{1:t})$$



$$\alpha^{\text{HMM}}_{t,i} = \Big(\sum_{j=1}^{K} \alpha^{\text{HMM}}_{t-1,j}\; a_{j,i} \Big)\, \mathcal{N}\big(\boldsymbol{\xi}_t \,|\, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\big) \quad \text{with} \quad \alpha^{\text{HMM}}_{1,i} = \Pi_i\, \mathcal{N}\big(\boldsymbol{\xi}_1 \,|\, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\big)$$





It can be used to evaluate trajectories by computing the likelihood

$$\mathcal{P}(\boldsymbol{\xi} \,|\, \boldsymbol{\Theta}^{\text{HMM}}) = \sum_{i=1}^{K} \alpha^{\text{HMM}}_{T,i}$$

Useful intermediary variables in HMM

Forward variable $\alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i, \boldsymbol{\xi}_{1:t})$

Backward variable $\beta_{t,i}^{\text{HMM}} = \mathcal{P}(\boldsymbol{\xi}_{t+1:T} \,|\, s_t = i)$

Smoothed node marginals $\gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i \,|\, \boldsymbol{\xi}_{1:T})$

Smoothed edge marginals $\zeta_{t,i,j}^{\text{HMM}} = \mathcal{P}(s_t = i, s_{t+1} = j \,|\, \boldsymbol{\xi}_{1:T})$

Backward algorithm

$$\beta_{t,i}^{\text{HMM}} = \mathcal{P}(\boldsymbol{\xi}_{t+1:T} \,|\, s_t = i)$$

Similarly, we can define a **backward variable** starting from the boundary condition

$$\beta_{T,i}^{\text{HMM}} = 1$$

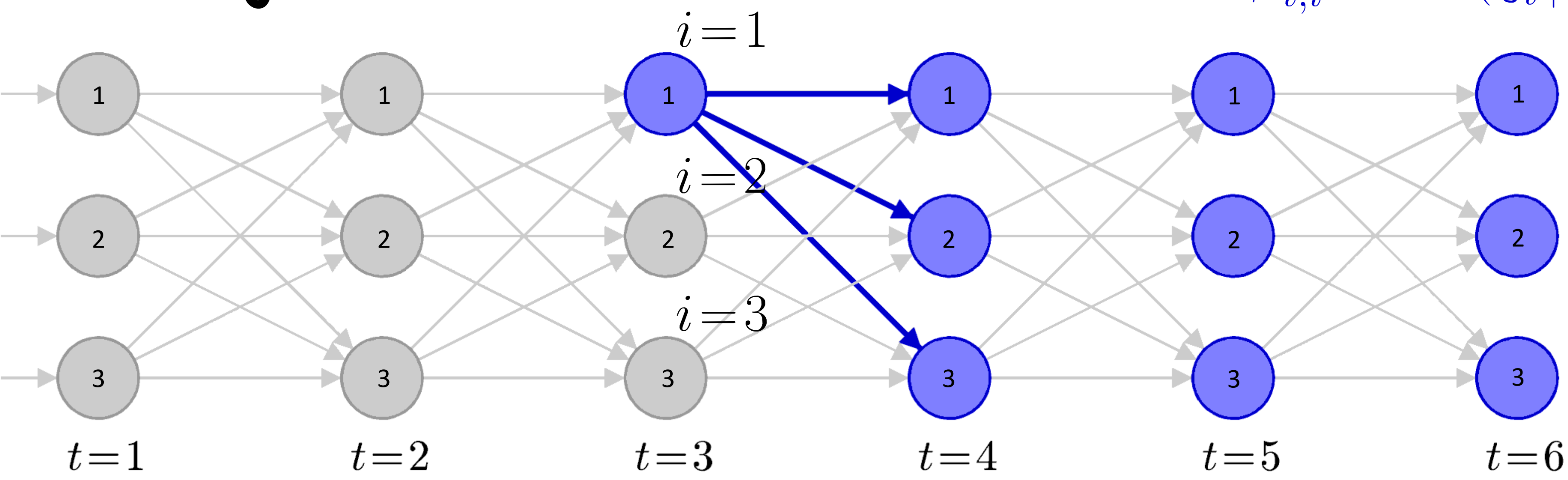
and computed as

$$\beta_{t,i}^{\text{HMM}} = \sum_{j=1}^{K} a_{i,j}\, \mathcal{N}\big(\boldsymbol{\xi}_{t+1} |\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\big)\; \beta_{t+1,j}^{\text{HMM}}$$

corresponding to the probability of the partial observation $\{\boldsymbol{\xi}_{t+1}, \ldots, \boldsymbol{\xi}_{T-1}, \boldsymbol{\xi}_T\}$, knowing that we are in state i at time step t.

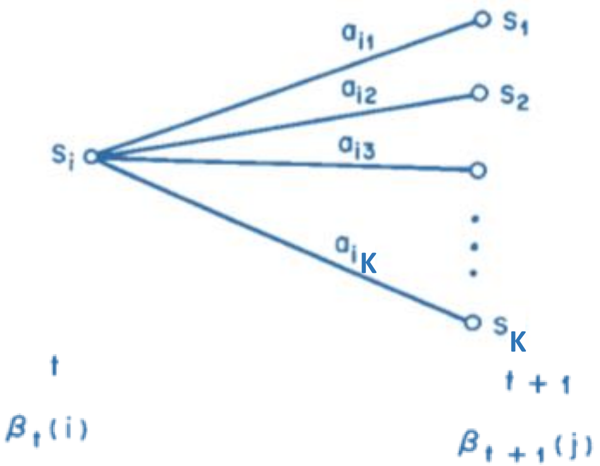
Backward algorithm

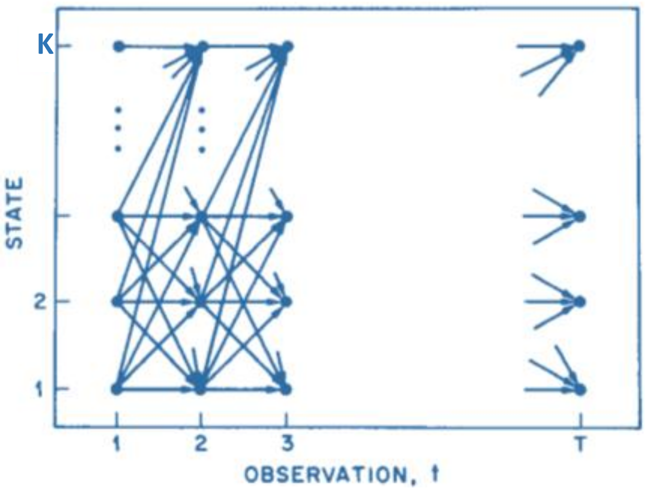
$$\beta^{\text{HMM}}_{t,i} = \mathcal{P}(\boldsymbol{\xi}_{t+1:T} \,|\, s_t \!=\! i)$$



$$\beta^{\text{HMM}}_{t,i} = \sum_{j=1}^{K} a_{i,j}\, \mathcal{N}\big(\boldsymbol{\xi}_{t+1} |\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\big)\; \beta^{\text{HMM}}_{t+1,j} \quad \text{with} \quad \beta^{\text{HMM}}_{T,i} = 1$$

boundary condition, corresponding to $\mathcal{P}(\emptyset \,|\, s_T \!=\! i) = 1, \quad \forall i$





Useful intermediary variables in HMM

These variable are sometimes called ***smoothed variables*** as they combine forward and backward probabilities in the computation.

You can think of their roles as passing "messages" from left to right, and from right to left, and then combining the information at each node.

Forward variable $\alpha_{t,i}^{\mathrm{HMM}} = \mathcal{P}$

Backward variable $\beta_{t,i}^{\mathrm{HMM}} = \mathcal{P}(\boldsymbol{\xi}_{t-} \quad | \, s_t$

Smoothed node marginals $\gamma_{t,i}^{\mathrm{HMM}} = \mathcal{P}(s_t = i \,|\, \boldsymbol{\xi}_{1:T})$

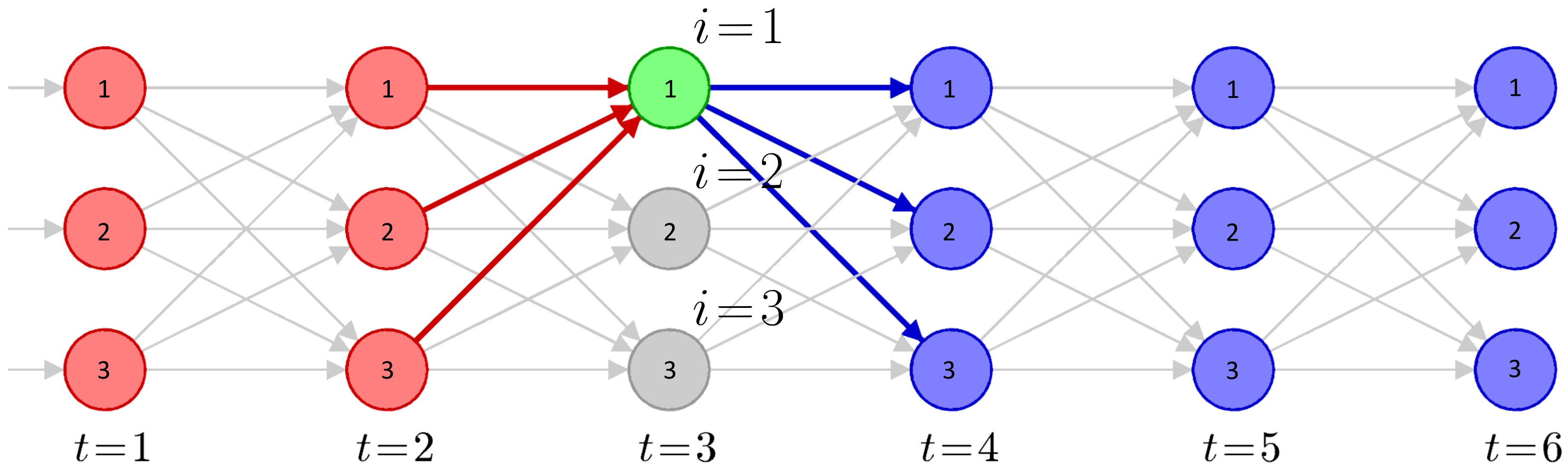
Smoothed edge marginals $\zeta_{t,i,j}^{\mathrm{HMM}} = \mathcal{P}(s_t = i, \, s_{t+1} = j \,|\, \boldsymbol{\xi}_{1:T})$

Smoothed node marginals

$$\gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i \,|\, \boldsymbol{\xi}_{1:T})$$

$$\alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i, \boldsymbol{\xi}_{1:t})$$

$$\beta_{t,i}^{\text{HMM}} = \mathcal{P}(\boldsymbol{\xi}_{t+1:T} \,|\, s_t = i)$$



$$\gamma_{t,i}^{\text{HMM}} = \frac{\alpha_{t,i}^{\text{HMM}} \; \beta_{t,i}^{\text{HMM}}}{\sum\limits_{k=1}^{K} \alpha_{t,k}^{\text{HMM}} \; \beta_{t,k}^{\text{HMM}}} = \frac{\alpha_{t,i}^{\text{HMM}} \; \beta_{t,i}^{\text{HMM}}}{\mathcal{P}(\boldsymbol{\xi})}$$

Useful intermediary variables in HMM

Forward variable $\alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i, \boldsymbol{\xi}_{1:t})$

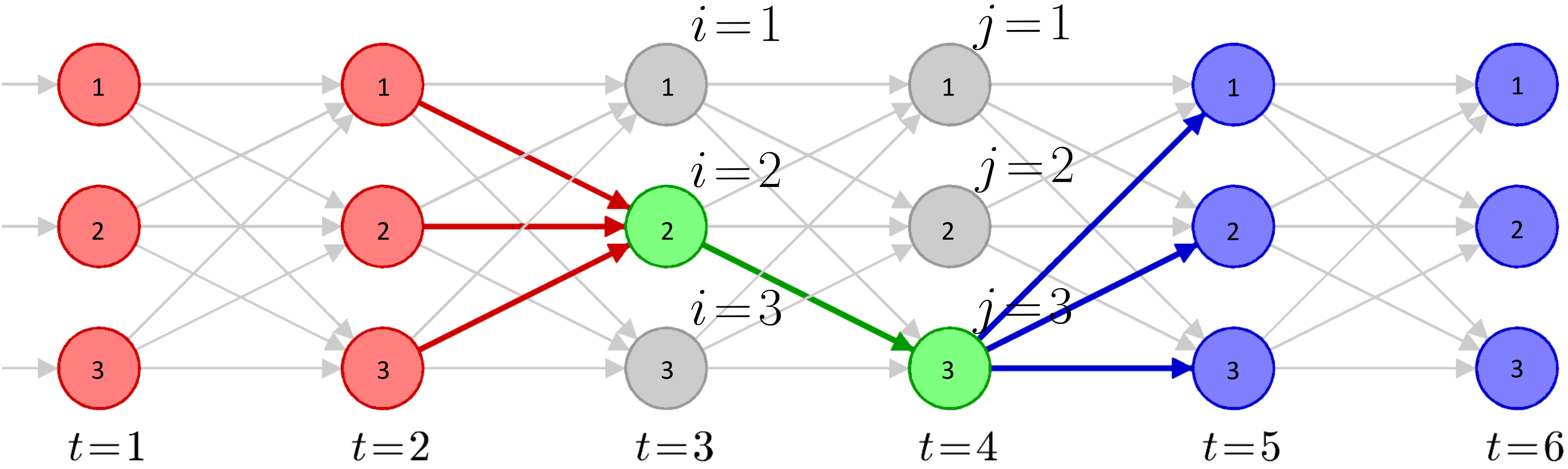
Backward variable $\beta_{t,i}^{\text{HMM}} = \mathcal{P}(\boldsymbol{\xi}_{t+1:T} \mid s_t = i)$

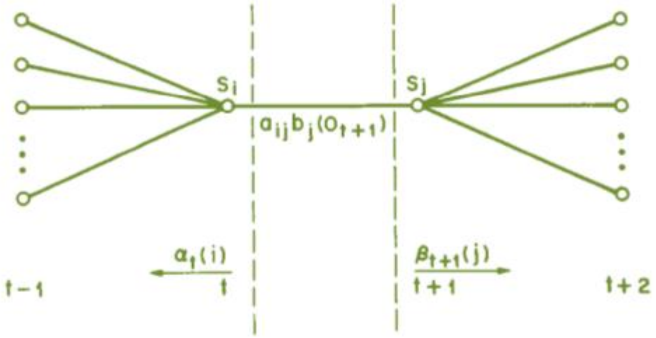
Smoothed node marginals $\gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i \mid \boldsymbol{\xi}_{1:T})$

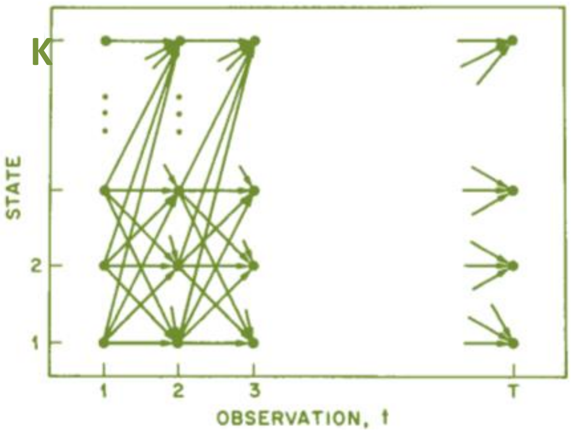
Smoothed edge marginals $\zeta_{t,i,j}^{\text{HMM}} = \mathcal{P}(s_t = i, s_{t+1} = j \mid \boldsymbol{\xi}_{1:T})$

Smoothed edge marginals

$$\zeta^{\text{HMM}}_{t,i,j} = \mathcal{P}(s_t{=}i, s_{t+1}{=}j \,|\, \boldsymbol{\xi}_{1:T})$$







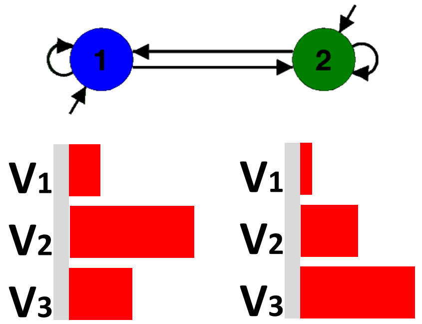
$$\zeta^{\text{HMM}}_{t,i,j} = \frac{\alpha^{\text{HMM}}_{t,i}\; a_{i,j}\; \mathcal{N}\big(\boldsymbol{\xi}_{t+1} \,|\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\big)\; \beta^{\text{HMM}}_{t+1,j}}{\sum_{k=1}^{K}\sum_{l=1}^{K} \alpha^{\text{HMM}}_{t,k}\; a_{k,l}\; \mathcal{N}\big(\boldsymbol{\xi}_{t+1} \,|\, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\big)\; \beta^{\text{HMM}}_{t+1,l}}$$

$$= \frac{\alpha^{\text{HMM}}_{t,i}\; a_{i,j}\; \mathcal{N}\big(\boldsymbol{\xi}_{t+1} \,|\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\big)\; \beta^{\text{HMM}}_{t+1,j}}{\mathcal{P}(\boldsymbol{\xi})}$$

EM for HMM (Baum-Welch algorithm)

K Gaussians
M sequences
T_m points per sequences

E-step: compute $\gamma^{\text{HMM}}_{m,t,i}$ and $\zeta^{\text{HMM}}_{m,t,i,j}$



M-step:

$$\Pi_i \leftarrow \frac{\sum_{m=1}^{M} \gamma^{\text{HMM}}_{m,1,i}}{M} = \frac{\text{Total number of times in i at time step 1}}{\text{Total number of trajectories}}$$

$$a_{i,j} \leftarrow \frac{\sum_{m=1}^{M}\sum_{t=1}^{T_m-1} \zeta^{\text{HMM}}_{m,t,i,j}}{\sum_{m=1}^{M}\sum_{t=1}^{T_m-1} \gamma^{\text{HMM}}_{m,t,i}} = \frac{\text{Total number of transitions from i to j}}{\text{Total number of times in i (and transit to anything else)}}$$

$$\boldsymbol{\mu}_i \leftarrow \frac{\sum_{m=1}^{M}\sum_{t=1}^{T_m} \gamma^{\text{HMM}}_{m,t,i}\, \boldsymbol{\xi}_{m,t}}{\sum_{m=1}^{M}\sum_{t=1}^{T_m} \gamma^{\text{HMM}}_{m,t,i}}$$

result similar to GMM

$$\boldsymbol{\Sigma}_i \leftarrow \frac{\sum_{m=1}^{M}\sum_{t=1}^{T_m} \gamma^{\text{HMM}}_{m,t,i}\, (\boldsymbol{\xi}_{m,t} - \boldsymbol{\mu}_i)(\boldsymbol{\xi}_{m,t} - \boldsymbol{\mu}_i)^{\top}}{\sum_{m=1}^{M}\sum_{t=1}^{T_m} \gamma^{\text{HMM}}_{m,t,i}}$$

The update rules can be interpreted as normalized counts, with several types of weighted averages required in the computation.

$$\alpha^{\text{HMM}}_{t,i} = \mathcal{P}(s_t = i, \boldsymbol{\xi}_{1:t})$$

$$\beta^{\text{HMM}}_{t,i} = \mathcal{P}(\boldsymbol{\xi}_{t+1:T} \,|\, s_t = i)$$

$$\gamma^{\text{HMM}}_{t,i} = \mathcal{P}(s_t = i \,|\, \boldsymbol{\xi}_{1:T})$$

$$\zeta^{\text{HMM}}_{t,i,j} = \mathcal{P}(s_t = i, s_{t+1} = j \,|\, \boldsymbol{\xi}_{1:T})$$

Numerical underflow issue in HMM

For long sequences, the forward and backward variables can quickly get very low, likely exceeding the precision range of the computer.

A simple scaling procedure is to multiply $\alpha_{t,i}^{\text{HMM}}$ by a factor independent of i, and divide $\beta_{t,i}^{\text{HMM}}$ by the same factor so that they are cancelled in the forward-backward computation.

The computation can be kept within reasonable bounds by setting the scaling factor

$$c_t = \frac{1}{\sum_{i=1}^{K} \alpha_{t,i}^{\text{HMM}}}$$

Numerical underflow issue in HMM

This issue is sometimes not covered in textbooks, although it remains very important for practical implementation of HMM!

$$c_t = \frac{1}{\sum_{i=1}^{K} \alpha_{t,i}^{\text{HMM}}}$$

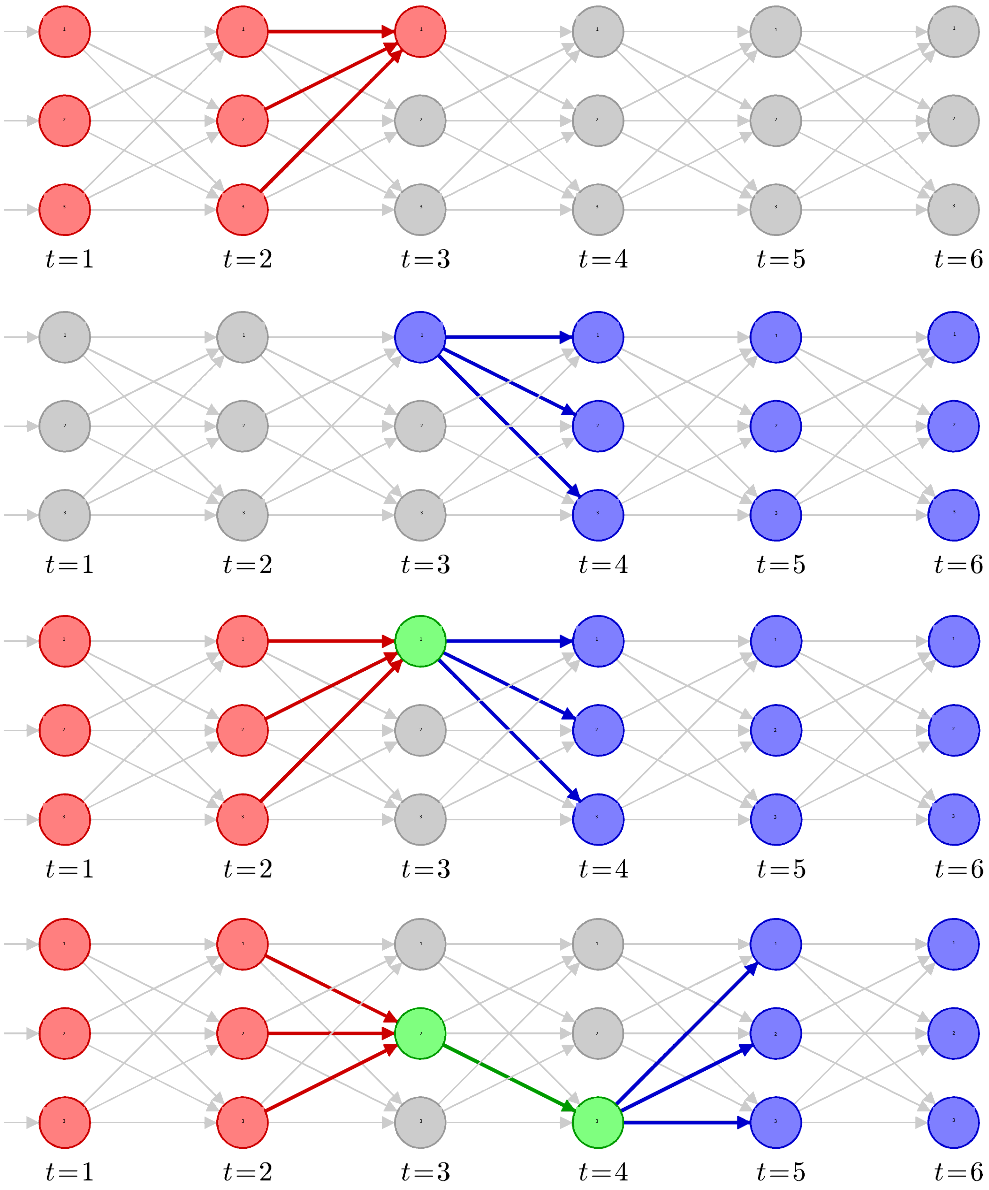
We have by induction

$$\hat{\alpha}_{t,i}^{\text{HMM}} = \left(\prod_{s=1}^{t} c_s\right) \alpha_{t,i}^{\text{HMM}}, \quad \hat{\beta}_{t,i}^{\text{HMM}} = \left(\prod_{s=t}^{T} c_s\right) \beta_{t,i}^{\text{HMM}}$$

With this, the numerator and denominator will cancel out when used in the re-estimation formulas. For example

$$\gamma_{t,i}^{\text{HMM}} = \frac{\hat{\alpha}_{t,i}^{\text{HMM}} \hat{\beta}_{t,i}^{\text{HMM}}}{\sum_{k=1}^{K} \hat{\alpha}_{t,k}^{\text{HMM}} \hat{\beta}_{t,k}^{\text{HMM}}} = \frac{\left(\prod_{s=1}^{t} c_s\right)\left(\prod_{s=t}^{T} c_s\right) \alpha_{t,i}^{\text{HMM}} \beta_{t,i}^{\text{HMM}}}{\left(\prod_{s=1}^{t} c_s\right)\left(\prod_{s=t}^{T} c_s\right) \sum_{k=1}^{K} \alpha_{t,k}^{\text{HMM}} \beta_{t,k}^{\text{HMM}}} = \frac{\alpha_{t,i}^{\text{HMM}} \beta_{t,i}^{\text{HMM}}}{\sum_{k=1}^{K} \alpha_{t,k}^{\text{HMM}} \beta_{t,k}^{\text{HMM}}}$$

Why did we introduce these intermediary variables in HMM?



Forward variable

$$\alpha^{\mathrm{HMM}}_{t,i} = \mathcal{P}(s_t = i, \boldsymbol{\xi}_{1:t})$$

Backward variable

$$\beta^{\mathrm{HMM}}_{t,i} = \mathcal{P}(\boldsymbol{\xi}_{t+1:T} \mid s_t = i)$$

Smoothed node marginals

$$\gamma^{\mathrm{HMM}}_{t,i} = \mathcal{P}(s_t = i \mid \boldsymbol{\xi}_{1:T})$$

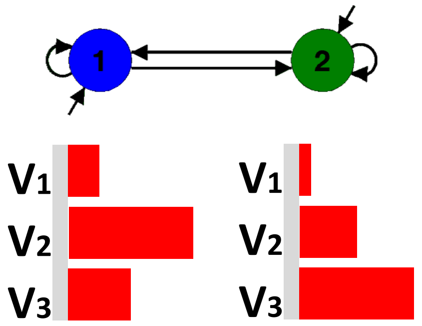
Smoothed edge marginals

$$\zeta^{\mathrm{HMM}}_{t,i,j} = \mathcal{P}(s_t = i, s_{t+1} = j \mid \boldsymbol{\xi}_{1:T})$$

Why did we introduce these intermediary variables in HMM?

How to estimate the parameters of an HMM?

$\rightarrow$ Maximum of expected complete data log-likelihood $\mathcal{Q}(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{\text{old}})$



How to compute $\frac{\partial \mathcal{Q}}{\partial \Pi_i}=0$, $\frac{\partial \mathcal{Q}}{\partial a_{i,j}}=0$, $\frac{\partial \mathcal{Q}}{\partial \boldsymbol{\mu}_i}=0$ and $\frac{\partial \mathcal{Q}}{\partial \boldsymbol{\Sigma}_i}=0$?

$\rightarrow$ Requires to compute $\zeta_{t,i,j}^{\text{HMM}} = \mathcal{P}(s_t{=}i, s_{t+1}{=}j \,|\, \boldsymbol{\xi}_{1:T})$

$\rightarrow$ Requires to compute $\gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t{=}i \,|\, \boldsymbol{\xi}_{1:T})$

Observations $\boldsymbol{\xi}_{1:T}$

How to compute $\zeta_{t,i,j}^{\text{HMM}}$ and $\gamma_{t,i}^{\text{HMM}}$?

$\rightarrow$ Requires to compute $\alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t{=}i, \boldsymbol{\xi}_{1:t})$

$\rightarrow$ Requires to compute $\beta_{t,i}^{\text{HMM}} = \mathcal{P}(\boldsymbol{\xi}_{t+1:T} \,|\, s_t{=}i)$

$\max_{\boldsymbol{\Theta}} \mathcal{Q}(\boldsymbol{\Theta}, \boldsymbol{\Theta}^{\text{old}})$

$\zeta_{t,i,j}^{\text{HMM}}$ $\gamma_{t,i}^{\text{HMM}}$

$\alpha_{t,i}^{\text{HMM}}$ $\beta_{t,i}^{\text{HMM}}$

Viterbi decoding (MAP vs MPE estimates)

Maximum a posteriori

Most probable explanation

Python notebook:
demo_HMM.ipynb

Matlab code:
demo_HMM_Viterbi01.m

Viterbi decoding (MAP vs MPE estimates)

Maximum a posteriori — Most probable explanation

The (jointly) most probable sequence of states $\hat{\boldsymbol{s}}^{\text{MAP}}$ is not necessarily the same as the sequence of (marginally) most probable states $\hat{\boldsymbol{s}}^{\text{MPE}}$

$$\hat{\boldsymbol{s}}^{\text{MAP}} = \arg\max_{\{s_1,s_2,\ldots,s_T\}} \mathcal{P}(\boldsymbol{s}|\boldsymbol{\xi})$$

$$\hat{\boldsymbol{s}}^{\text{MPE}} = \left\{ \arg\max_{s_1} \overbrace{\mathcal{P}(s_1|\boldsymbol{\xi})}^{\gamma_1^{\text{HMM}}}, \arg\max_{s_2} \overbrace{\mathcal{P}(s_2|\boldsymbol{\xi})}^{\gamma_2^{\text{HMM}}}, \ldots, \arg\max_{s_T} \overbrace{\mathcal{P}(s_T|\boldsymbol{\xi})}^{\gamma_T^{\text{HMM}}} \right\}$$

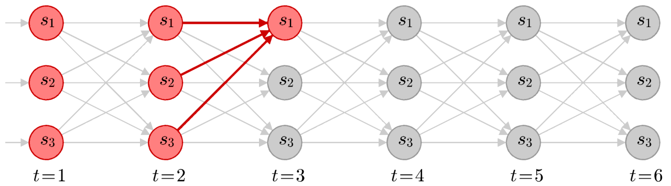
$$\gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i \,|\, \boldsymbol{\xi}_{1:T})$$

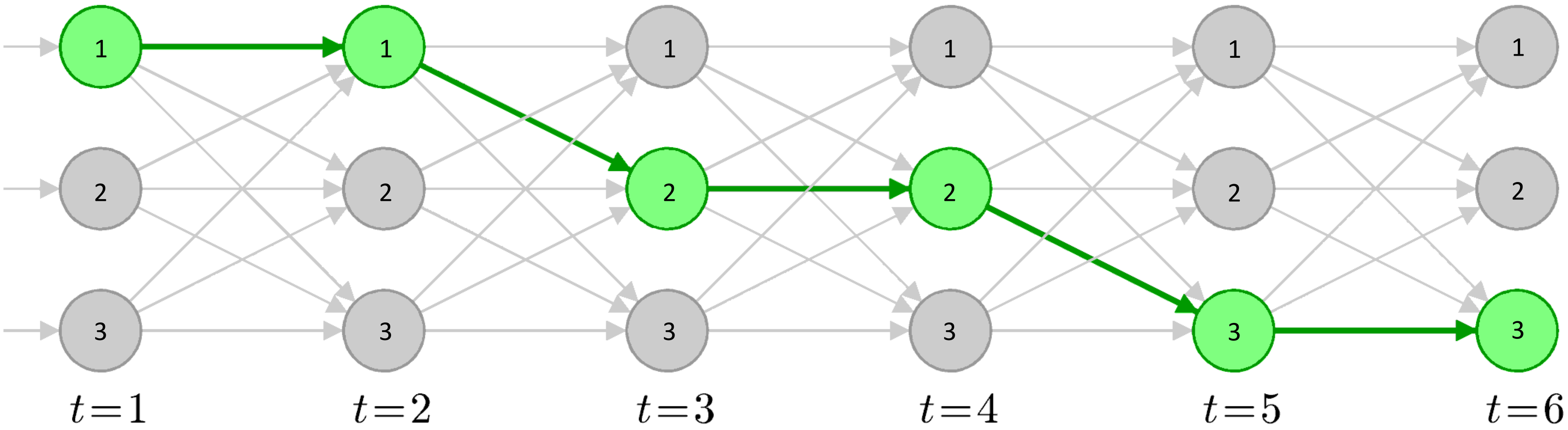
$\hat{\boldsymbol{s}}^{\text{MAP}}$ can be computed with the **Viterbi algorithm**, employing the max operator in a forward pass, followed by a backward pass using a **fast traceback procedure** to recover the most probable path.

Viterbi decoding

Forward algorithm:

$$\alpha^{\text{HMM}}_{t,i} = \Big(\sum_{j=1}^{K} \alpha^{\text{HMM}}_{t-1,j} \, a_{j,i} \Big) \, \mathcal{N}(\boldsymbol{\xi}_t \,|\, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$





Forward recursion:

$$\begin{cases} \delta_{t,i} = \max\limits_{j} \big(\delta_{t-1,j} \; a_{j,i}\big) \, \mathcal{N}\big(\boldsymbol{\xi}_t | \, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\big) \\ \Psi_{t,i} = \arg\max\limits_{j} \big(\delta_{t-1,j} \; a_{j,i}\big) \end{cases}$$

← probability of ending in state i at time step t by taking the most probable path

← indices that keep track of the states j that maximized $\delta_{t,i}$.

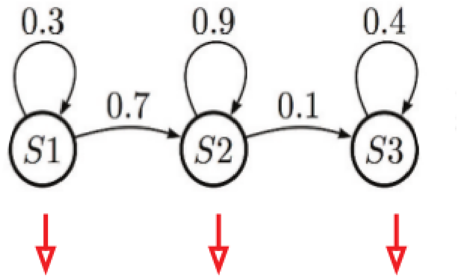
Backtracking: $\hat{s}^{\text{MAP}}_t = \Psi_{t+1,\, \hat{s}^{\text{MAP}}_{t+1}}$

Viterbi decoding - Example

$$\delta_{t,i} = \max_j \left(\delta_{t-1,j}\, a_{j,i}\right)\, b(\boldsymbol{\xi}_t|i)$$

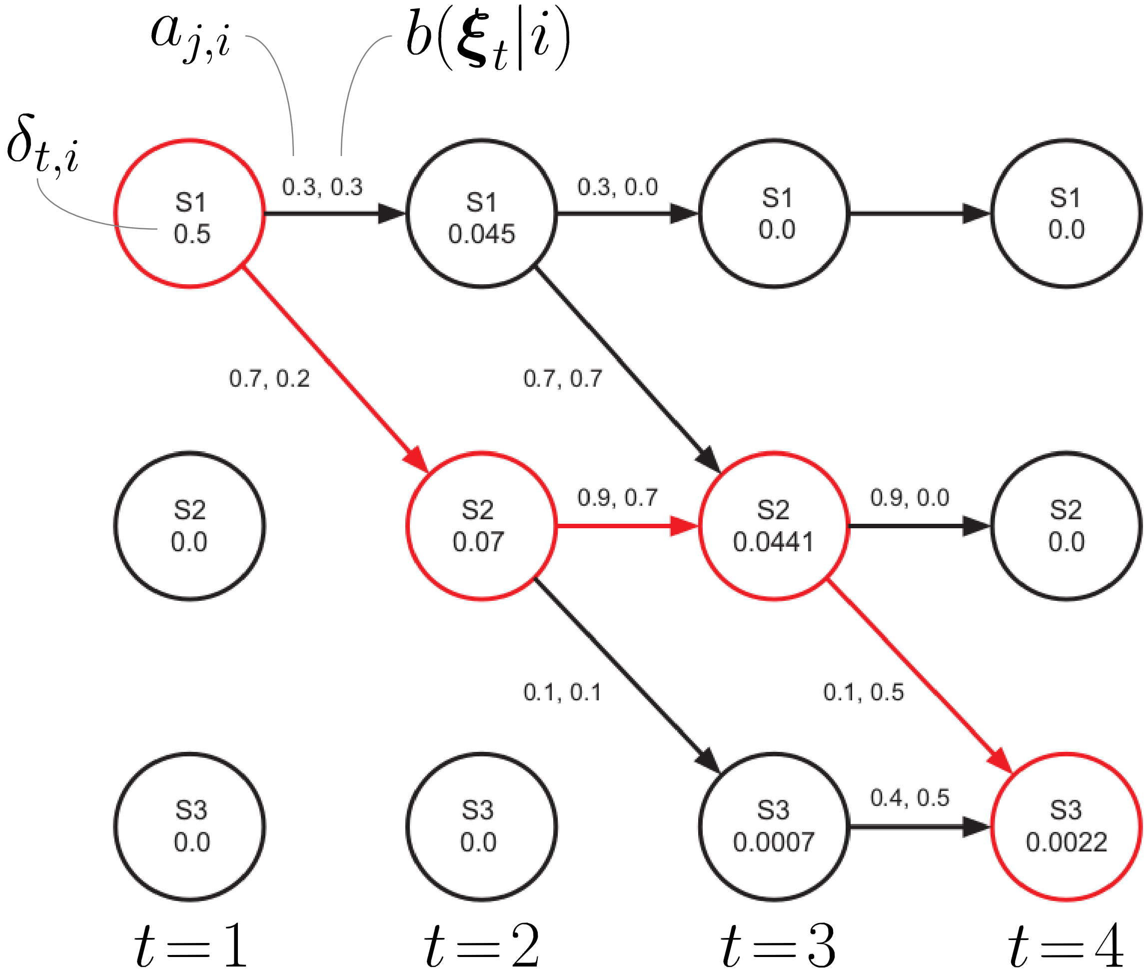
$$\text{with } \delta_{1,i} = \Pi_i\, \mathcal{N}\left(\boldsymbol{\xi}_1|\, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)$$

$$\boldsymbol{\Pi} = [1, 0, 0]$$



Dictionary

C1	0.5	0	0
C2	0.2	0	0
C3	0.3	0.2	0
C4	0	0.7	0.1
C5	0	0.1	0
C6	0	0	0.5
C7	0	0	0.4



$$\text{Observation: } \boldsymbol{\xi} = \{C1, C3, C4, C6\}$$

Figure adapted from Kevin P. Murphy (2012), Machine Learning: A Probabilistic Perspective

Numerical underflow issue in Viterbi

Similarly to the forward-backward variables in HMM, we have to take care about potential numerical underflow when implementing Viterbi decoding.

A simple way is to normalize $\delta_{t,i}$ at each time step t by multiplying it with

$$c_t = \frac{1}{\sum_{i=1}^{K} \delta_{t,i}}$$

similarly as in the computation of the forward-backward variables.

Such scaling will not affect the maximum.

Numerical underflow issue in Viterbi

Alternatively, we can work in the log domain. We then have

$$\begin{aligned}\log \delta_{t,i} &= \max_{\boldsymbol{s}_{1:t-1}} \log \mathcal{P}(\boldsymbol{s}_{1:t-1}, s_t\!=\!i \,|\, \boldsymbol{\xi}_{1:t}) \\ &= \max_{j}\Big(\log \delta_{t-1,j} + \log a_{i,j}\Big) + \log \mathcal{N}\Big(\boldsymbol{\xi}_t |\ \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\Big)\end{aligned}$$

With high dimensional Gaussians as emission distributions, the Viterbi computation with log can result in a **significant speedup**, since computing $\log \mathcal{P}(\boldsymbol{\xi}_t|s_t)$ can be much faster than computing $\mathcal{P}(\boldsymbol{\xi}_t|s_t)$.

When training HMMs, the Viterbi algorithm can be also used (instead of the forward-backward variables) in the E step of the EM procedure.

Hidden semi-Markov model (HSMM)

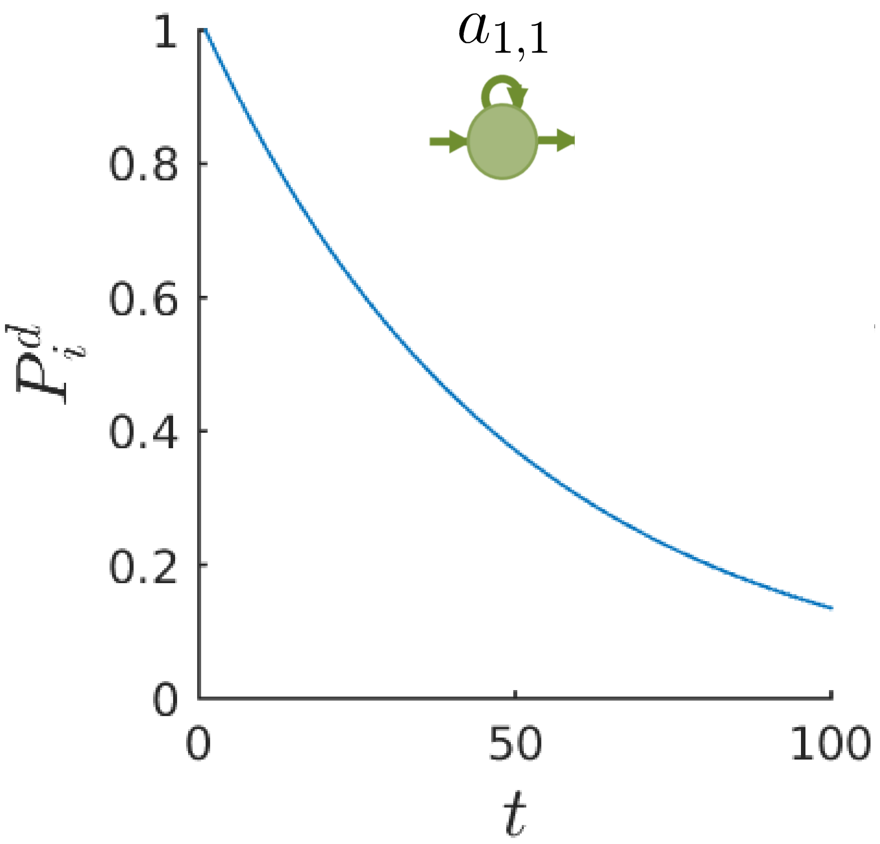
Python notebook:
demo_HSMM.ipynb

Matlab code:
demo_HSMM01.m

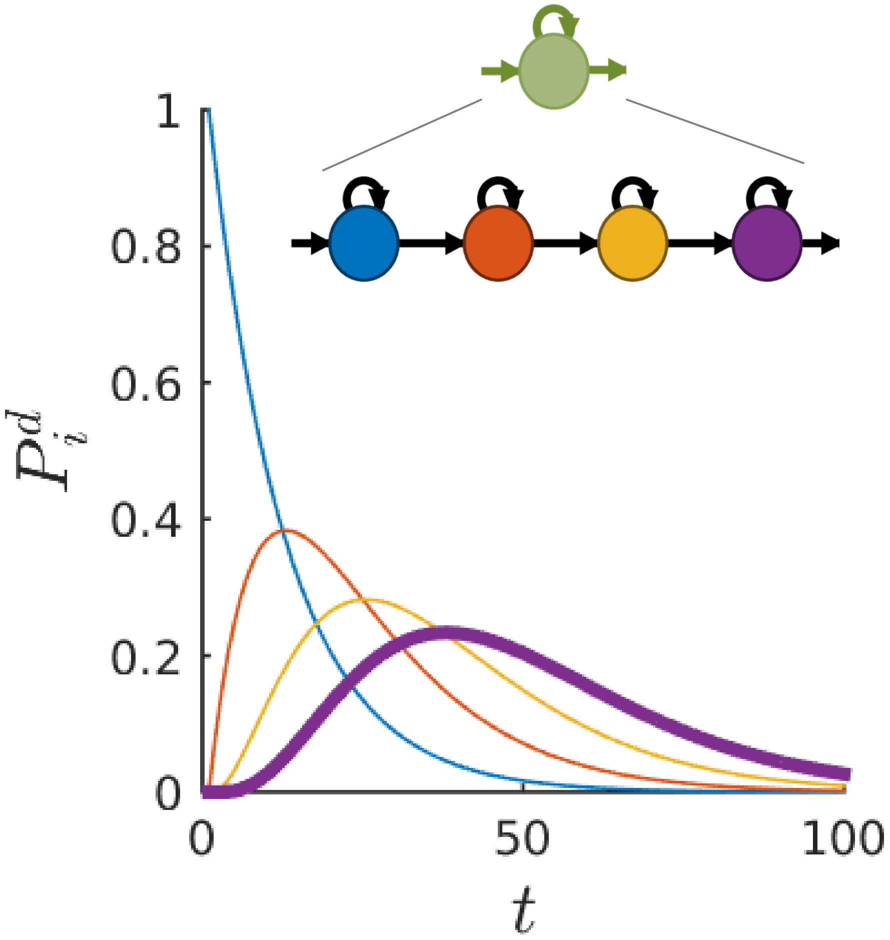
State duration probability in standard HMM

The state duration follows a geometric distribution

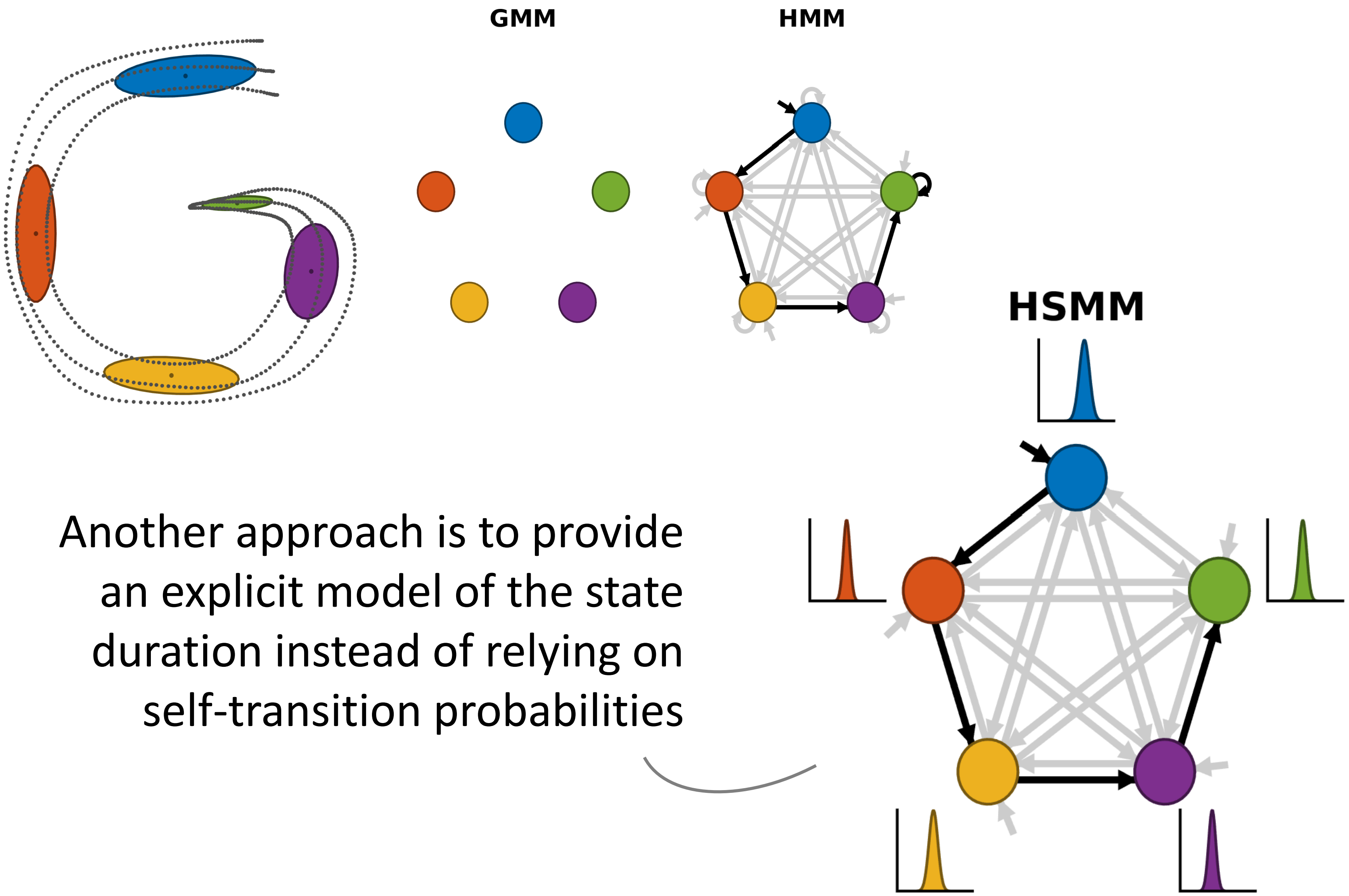
$$\mathcal{P}(d) = a_{i,i}^{d-1}(1 - a_{i,i})$$



Naive approach: By artificially duplicating the number of states while keeping the same emission distribution, other state duration distributions can be modeled



Hidden semi-Markov model (HSMM)



Another approach is to provide an explicit model of the state duration instead of relying on self-transition probabilities

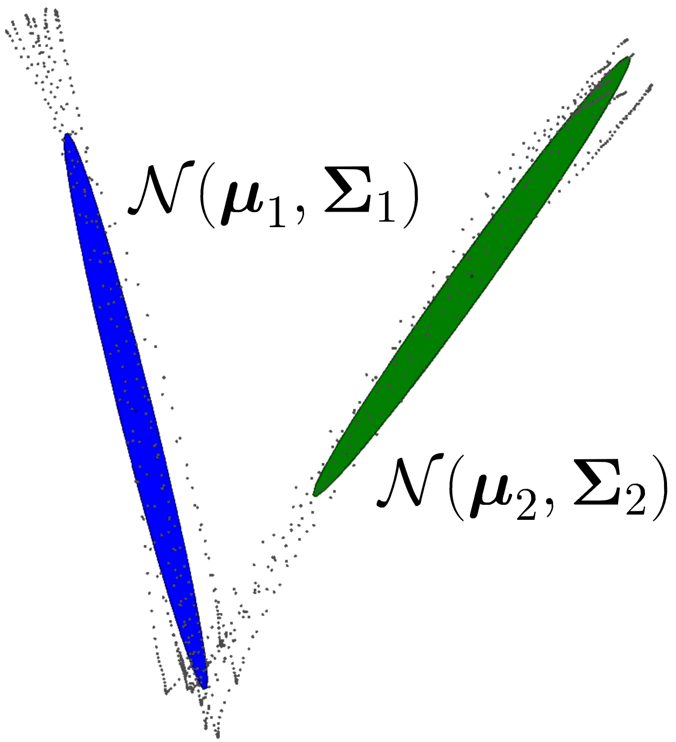
Hidden semi-Markov model (HSMM)

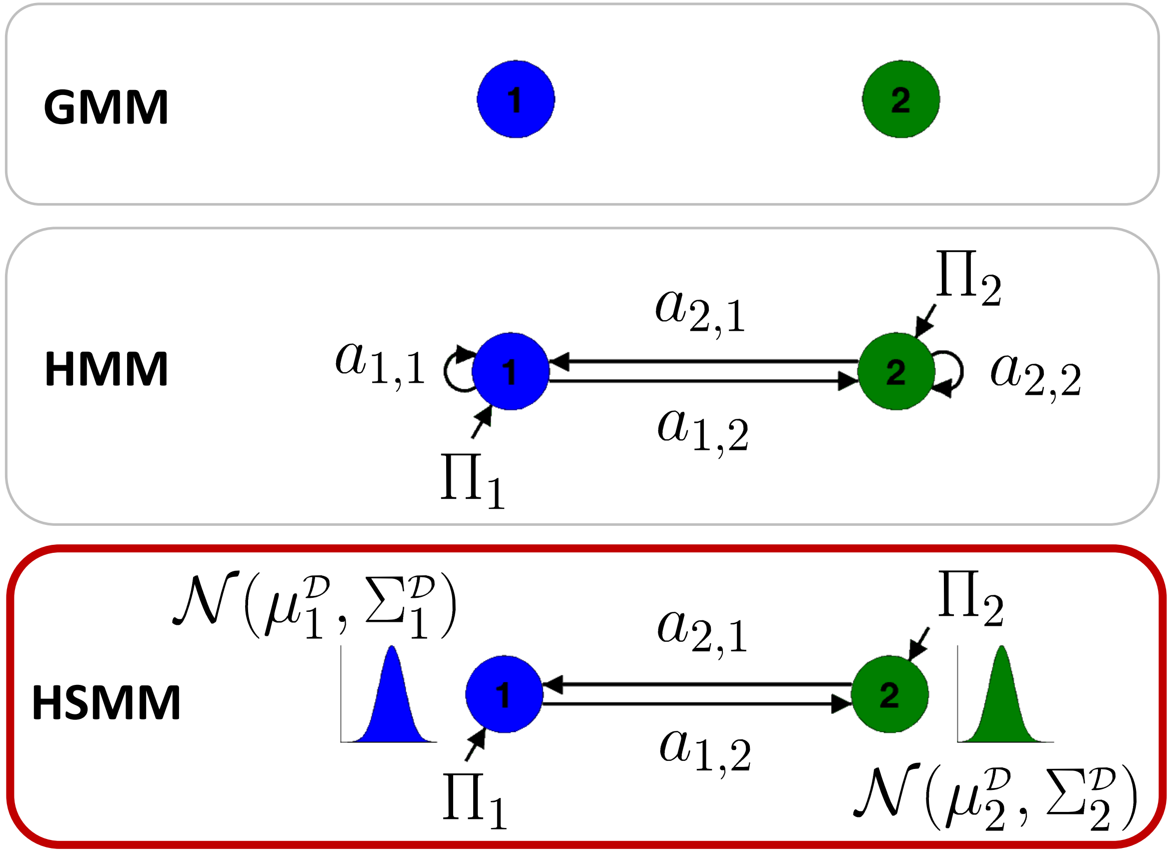
$$\boldsymbol{\Theta}^{\text{GMM}} = \{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^K$$

$$\boldsymbol{\Theta}^{\text{HMM}} = \{\{a_{i,j}\}_{j=1}^K, \Pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^K$$

$$\boldsymbol{\Theta}^{\text{HSMM}} = \{\{a_{i,j}\}_{j=1, j\neq i}^K, \Pi_i, \mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^K$$

Parametric duration distribution

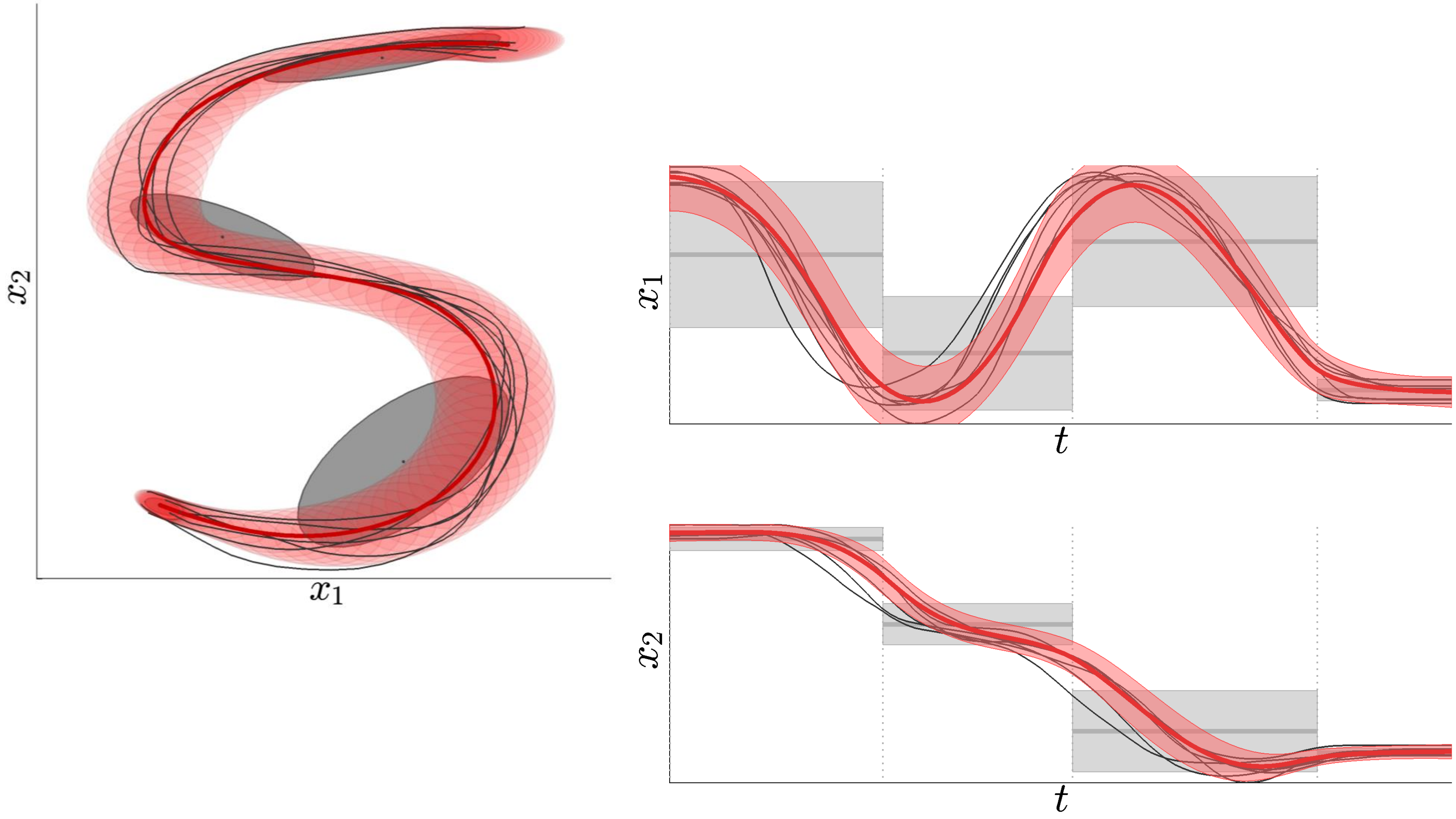




HMM with dynamic features
(Trajectory-HMM)

Matlab code:
demo_trajHSMM01.m

HMM with dynamic features



HMM with dynamic features

For the encoding of movements, velocity and acceleration can be used as dynamic features. By considering an Euler approximation, the velocity is computed as

$$\dot{\boldsymbol{x}}_t = \frac{\boldsymbol{x}_{t+1} - \boldsymbol{x}_t}{\Delta t}$$

where $\boldsymbol{x}_t$ is a multivariate position vector.

The acceleration is similarly computed as

$$\ddot{\boldsymbol{x}}_t = \frac{\dot{\boldsymbol{x}}_{t+1} - \dot{\boldsymbol{x}}_t}{\Delta t} = \frac{\boldsymbol{x}_{t+2} - 2\boldsymbol{x}_{t+1} + \boldsymbol{x}_t}{\Delta t^2}$$

HMM with dynamic features

$$\dot{\boldsymbol{x}}_t = \frac{\boldsymbol{x}_{t+1} - \boldsymbol{x}_t}{\Delta t}, \quad \ddot{\boldsymbol{x}}_t = \frac{\boldsymbol{x}_{t+2} - 2\boldsymbol{x}_{t+1} + \boldsymbol{x}_t}{\Delta t^2}$$

A vector $\boldsymbol{\zeta}_t$ will be used to represent the concatenated position, velocity and acceleration vectors at time step t

$$\boldsymbol{\zeta}_t = \begin{bmatrix} \boldsymbol{x}_t \\ \dot{\boldsymbol{x}}_t \\ \ddot{\boldsymbol{x}}_t \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ -\frac{1}{\Delta t}\boldsymbol{I} & \frac{1}{\Delta t}\boldsymbol{I} & \boldsymbol{0} \\ \frac{1}{\Delta t^2}\boldsymbol{I} & -\frac{2}{\Delta t^2}\boldsymbol{I} & \frac{1}{\Delta t^2}\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t+1} \\ \boldsymbol{x}_{t+2} \end{bmatrix}$$

Here, the number of derivatives will be set up to acceleration ($C=3$), but the same approach can be applied to a different number of derivatives.

A GMM/HMM/HSMM with centers $\{\boldsymbol{\mu}_i\}_{i=1}^K$ and covariance matrices $\{\boldsymbol{\Sigma}_i\}_{i=1}^K$ is first fit to the dataset $[\boldsymbol{\zeta}_1, \boldsymbol{\zeta}_2, \ldots, \boldsymbol{\zeta}_T]$.

HMM with dynamic features

$$\boldsymbol{\zeta}_t = \begin{bmatrix} \boldsymbol{x}_t \\ \dot{\boldsymbol{x}}_t \\ \ddot{\boldsymbol{x}}_t \end{bmatrix}$$

$\boldsymbol{\zeta}$ and $\boldsymbol{x}$ are defined as large vectors concatenating $\boldsymbol{\zeta}_t$ and $\boldsymbol{x}_t$ for all time steps

$$\boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{\zeta}_1 \\ \boldsymbol{\zeta}_2 \\ \vdots \\ \boldsymbol{\zeta}_T \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}$$

Similarly to the matrix operator defined in the previous slide for a single time step, a large sparse matrix $\boldsymbol{\Phi}$ can be defined so that

$$\boldsymbol{\zeta} = \boldsymbol{\Phi}\boldsymbol{x}$$

HMM with dynamic features

D *dimensions*
C *derivatives*
T *time steps*

$$
\begin{bmatrix} \vdots \\ \boldsymbol{x}_t \\ \dot{\boldsymbol{x}}_t \\ \ddot{\boldsymbol{x}}_t \\ \boldsymbol{x}_{t+1} \\ \dot{\boldsymbol{x}}_{t+1} \\ \ddot{\boldsymbol{x}}_{t+1} \\ \vdots \end{bmatrix}
=
\begin{bmatrix}
\ddots & \vdots & \vdots & \vdots & \iddots \\
\cdots & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \cdots \\
\cdots & -\frac{1}{\Delta t}\boldsymbol{I} & \frac{1}{\Delta t}\boldsymbol{I} & \boldsymbol{0} & \cdots \\
\cdots & \frac{1}{\Delta t^2}\boldsymbol{I} & -\frac{2}{\Delta t^2}\boldsymbol{I} & \frac{1}{\Delta t^2}\boldsymbol{I} & \cdots \\
 & \cdots & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\
 & \cdots & -\frac{1}{\Delta t}\boldsymbol{I} & \frac{1}{\Delta t}\boldsymbol{I} & \boldsymbol{0} \\
 & \cdots & \frac{1}{\Delta t^2}\boldsymbol{I} & -\frac{2}{\Delta t^2}\boldsymbol{I} & \frac{1}{\Delta t^2}\boldsymbol{I}
\end{bmatrix}
\begin{bmatrix} \vdots \\ \boldsymbol{x}_t \\ \boldsymbol{x}_{t+1} \\ \boldsymbol{x}_{t+2} \\ \boldsymbol{x}_{t+3} \\ \vdots \end{bmatrix}
$$

$\boldsymbol{\zeta} \in \mathbb{R}^{DCT}$ $\qquad$ $\boldsymbol{\Phi} \in \mathbb{R}^{DCT \times DT}$ $\qquad$ $\boldsymbol{x} \in \mathbb{R}^{DT}$

(C=3 here)

Large sparse matrix

HMM with dynamic features

By providing explicitely a sequence of states $\boldsymbol{s} = \{s_1, s_2, \dots, s_T\}$ of T time steps (e.g., retrieved by Viterbi or specified manually), with discrete states $s_t \in \{1, \dots, K\}$, the likelihood of a movement $\boldsymbol{\zeta} = \boldsymbol{\Phi x}$ is given by

$$\mathcal{P}(\boldsymbol{\zeta}|\boldsymbol{s}) = \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{\zeta}_t \,|\, \boldsymbol{\mu}_{s_t}, \boldsymbol{\Sigma}_{s_t})$$

where $\boldsymbol{\mu}_{s_t}$ and $\boldsymbol{\Sigma}_{s_t}$ are the center and covariance of state s_t at time step t.

This product can be rewritten as

$$\mathcal{P}(\boldsymbol{\zeta}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{\zeta} \,|\, \boldsymbol{\mu}_{\boldsymbol{s}}, \boldsymbol{\Sigma}_{\boldsymbol{s}})$$

$$\text{with} \quad \boldsymbol{\mu}_{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{\mu}_{s_1} \\ \boldsymbol{\mu}_{s_2} \\ \vdots \\ \boldsymbol{\mu}_{s_T} \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{\Sigma}_{s_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{s_2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\Sigma}_{s_T} \end{bmatrix}$$

HMM with dynamic features

For example, for a sequence of states $\boldsymbol{s} = \{1, 1, 2, 2, 3, 3, 3, 4\}$ with $K = 4$ and $T = 8$, we have

$$\boldsymbol{\mu}_{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \boldsymbol{\mu}_2 \\ \boldsymbol{\mu}_3 \\ \boldsymbol{\mu}_3 \\ \boldsymbol{\mu}_3 \\ \boldsymbol{\mu}_4 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{\Sigma}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_4 \end{bmatrix}$$

$$\boldsymbol{\mu}_{\boldsymbol{s}} \in \mathbb{R}^{DCT} \qquad \boldsymbol{\Sigma}_{\boldsymbol{s}} \in \mathbb{R}^{DCT \times DCT}$$

HMM with dynamic features

By using the relation $\boldsymbol{\zeta} = \boldsymbol{\Phi}\boldsymbol{x}$, we want to retrieve a trajectory

$$\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} \; \log \mathcal{P}(\boldsymbol{\Phi}\boldsymbol{x} \,|\, \boldsymbol{s})$$

$$\mathcal{N}(\boldsymbol{\Phi}\boldsymbol{x} \,|\, \boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s) = (2\pi)^{-\frac{DCT}{2}} |\boldsymbol{\Sigma}_s|^{-\frac{1}{2}} \; \exp\Big(-\frac{1}{2}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{\mu}_s)^{\top}\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{\mu}_s)\Big)$$

Equating to zero the derivative of

$$\log \mathcal{P}(\boldsymbol{\Phi}\boldsymbol{x} \,|\, \boldsymbol{s}) = -\frac{1}{2}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{\mu}_s)^{\top}\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{\mu}_s) - \frac{1}{2}\log|\boldsymbol{\Sigma}_s| - \frac{DCT}{2}\log(2\pi)$$

with respect to $\boldsymbol{x}$ yields

$$\boldsymbol{\Phi}^{\top}\boldsymbol{\Sigma}_s^{-1}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{\mu}_s) = \boldsymbol{0}$$

$$\iff \quad \hat{\boldsymbol{x}} = \left(\boldsymbol{\Phi}^{\top}\boldsymbol{\Sigma}_s^{-1}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\top}\boldsymbol{\Sigma}_s^{-1}\boldsymbol{\mu}_s$$

Weighted
least squares!

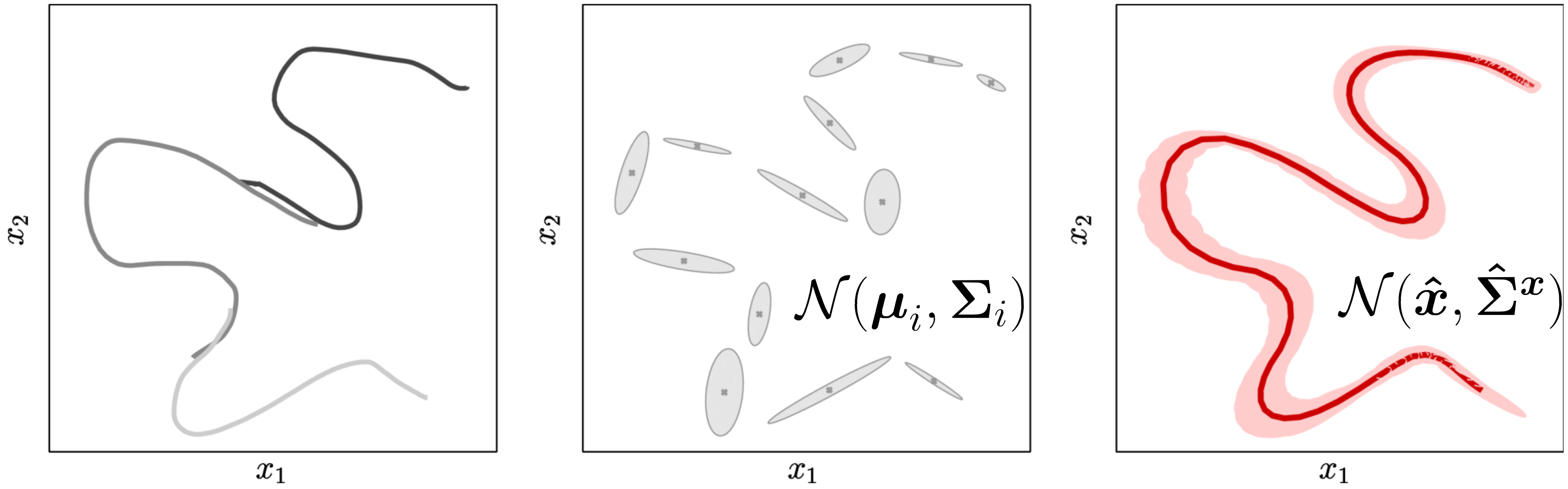
HMM with dynamic features

The residual error of this estimate is given by the covariance matrix

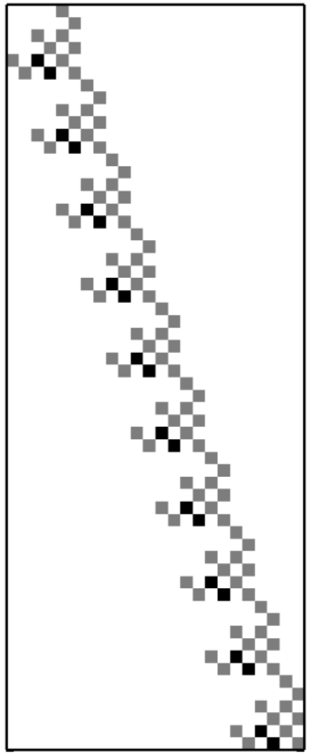
$$\hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}} = \sigma\left(\boldsymbol{\Phi}^{\top}\boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1}\boldsymbol{\Phi}\right)^{-1}$$

where σ is a scaling factor.

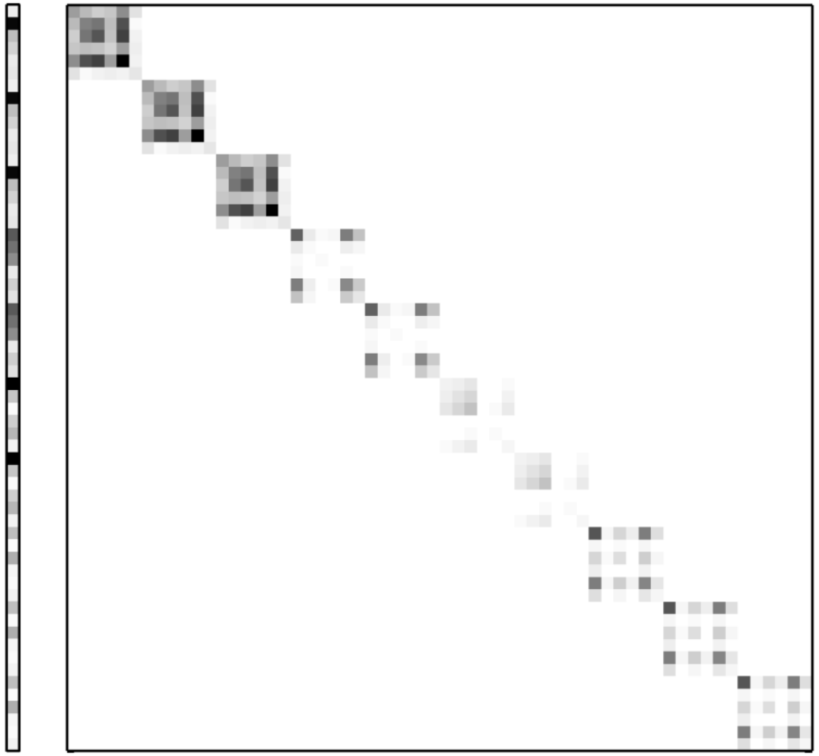
The resulting Gaussian $\mathcal{N}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}})$ forms a trajectory distribution, where $\hat{\boldsymbol{x}} \in \mathbb{R}^{DT}$ is an average trajectory stored in a vector form.



HMM with dynamic features - Summary



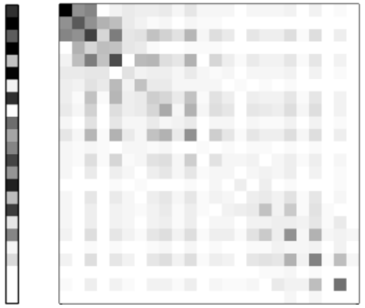
$\mathbf{\Phi}$



$\mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s)$

$$\hat{\boldsymbol{x}} = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_s^{-1} \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_s^{-1} \boldsymbol{\mu}_s$$

$$\hat{\boldsymbol{\Sigma}}^x = \sigma \left(\boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_s^{-1} \boldsymbol{\Phi}\right)^{-1}$$



$\mathcal{N}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\Sigma}}^x)$

References

Hidden Markov model (HMM)

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE, 77:2:257–285, February 1989

Hidden semi-Markov model (HSMM)

S.-Z. Yu. Hidden semi-Markov models. Artificial Intelligence, 174:215–243, 2010

S. E. Levinson. Continuously variable duration hidden Markov models for automatic speech recognition. Computer Speech & Language, 1(1):29–45, 1986

HMM with dynamic features (Trajectory HMM)

S. Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. IEEE Trans. on Acoustics, Speech, and Signal Processing, 34(1):52–59, 1986

H. Zen, K. Tokuda, and T. Kitamura. Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences. Computer Speech and Language, 21(1):153–173, 2007