• Markov models
• Hidden Markov model (HMM)
• Forward-backward algorithm
• Viterbi decoding (dynamic programming)
• Hidden semi-Markov model (HSMM)
• GMM/HMM/HSMM with dynamic features (Trajectory-HMM)
With a first order Markov model, the joint distribution of a sequence of states is assumed to be of the form

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

and we thus have

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

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

With a second order Markov model, the same joint distribution is assumed to be of the form

$$P(s_1, s_2, \ldots, s_T) = P(s_1)P(s_2|s_1) \prod_{t=3}^{T} P(s_t|s_{t-1}, s_{t-2})$$
Markov models - Parameters

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** $\mathbf{A}$ is defined, where

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

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

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

The elements of a $n$-step transition matrix $\mathbf{A}(n)$ are defined as $a_{i,j}(n) = \mathcal{P}(s_{t+n} = j \mid s_t = i)$, representing the probability to get from state $i$ to $j$ in exactly $n$ steps.

We then have $\mathbf{A}(1) = \mathbf{A}$ and $a_{i,j}(m + n) = \sum_{k=1}^{K} a_{i,k}(m)a_{k,j}(n)$. 
In other words, the probability of getting from $i$ to $j$ in $m+n$ steps is simply the probability of getting from state $i$ to $k$ in $m$ steps, and then from state $k$ to $j$ in $n$ steps, summed up over all $k$.

We can write the above as a matrix multiplication

$$A(m+n) = A(m)A(n)$$

We then have

$$A(n) = A A(n-1) = A A A(n-2) = \ldots = A^n$$

Thus, we can simulate $n$ steps of a Markov chain by raising the transition matrix at the power of $n$. 
Markov models in language modeling

One important application of Markov models is to make statistical language models, which are probability distributions over sequences of words.

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

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

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

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

In the general case, these are called **n-gram** models.
Markov models in 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 essay writing
The model can be used to sample from $\mathcal{P}(s_1, s_2, \ldots, s_t)$ to generate artificial text.
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.

Markov models in language modeling

Source: http://www.fit.vutbr.cz/~imikolov/rnnlm/gen-4gram.txt
MLE of transition matrix in Markov models

A Markov model is described by \( \Theta^{mm} = \{\{a_{i,j}\}_{j=1}^K, \Pi_i\}_{i=1}^K \), where the transition probabilities \( a_{i,j} \) are usually stored in a matrix \( A \).

The probability of a sequence \( \xi_{1:T} \) of length \( T \) is given by

\[
\mathcal{P}(\xi_{1:T} | \Theta^{mm}) = \prod (\Pi_1) A(\xi_1, \xi_2) A(\xi_2, \xi_3) \ldots A(\xi_{T-1}, \xi_T) \\
= \prod_{i=1}^K (\Pi_i)^{I(\xi_1 = i)} \prod_{t=2}^T \prod_{i=1}^K \prod_{j=1}^K (a_{i,j})^{I(\xi_{t-1} = i, \xi_t = j)}
\]

The log-likelihood of a set of \( M \) sequences of length \( T_m \) is given by

\[
\sum_{m=1}^M \log \mathcal{P}(\xi_{m,1:T_m} | \Theta^{mm}) = \sum_{i=1}^K N_i^1 \log \Pi_i + \sum_{i=1}^K \sum_{j=1}^K N_{i,j} \log a_{i,j}
\]

with \( N_i^1 = \sum_{m=1}^M I(\xi_{m,1} = i) \), \( N_{i,j} = \sum_{m=1}^M \sum_{t=2}^T I(\xi_{m,t-1} = i, \xi_{m,t} = j) \)
MLE of transition matrix in Markov models

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

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

These results can be extended in a straightforward way to higher order Markov models.

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)
demo_HMM01.m

You can think of an HMM as:
• a Markov chain with stochastic measurements
• a GMM with latent variables changing over time
Inference problems associated with HMMs

Probability of an observed sequence
\[ \mathcal{P}(\xi_{1:T}) = \mathcal{P}(\xi_1, \xi_2, \ldots, \xi_T) \leftarrow \text{Use of forward variable} \]

Probability of the latent variables

- Filtering \quad \rightarrow \quad \text{Use of forward or backward variables}
  \[ \mathcal{P}(s_t \mid \xi_{1:t}) = \mathcal{P}(s_t \mid \xi_1, \xi_2, \ldots, \xi_t) \leftarrow \text{forward comp.} \]

- Prediction
  \[ \mathcal{P}(s_{t+1} \mid \xi_{1:t}) = \mathcal{P}(s_{t+1} \mid \xi_1, \xi_2, \ldots, \xi_t) \leftarrow \text{forward comp.} \]

- Smoothing \quad \rightarrow \quad \text{Forward-backward algorithm}
  \[ \mathcal{P}(s_t \mid \xi_{1:T}) = \mathcal{P}(s_t \mid \xi_1, \xi_2, \ldots, \xi_T) \]

- MAP estimation \quad \rightarrow \quad \text{Viterbi decoding}
  \[ \mathcal{P}(s_{1:T} \mid \xi_{1:T}) = \mathcal{P}(s_1, s_2, \ldots, s_T \mid \xi_1, \xi_2, \ldots, \xi_T) \]
Emission/output distributions in HMM

Discrete tables

Gaussian distribution

Mixture of Gaussians

GMM with latent variable $z_t$ depending on the conditional distribution $\mathcal{P}(z_t | z_{t-1})$
Transition matrix structure in HMM

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

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

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

$\mathcal{N}(\mu_{1,1}, \Sigma_{1,1})$, $\mathcal{N}(\mu_{1,2}, \Sigma_{1,2})$, $\mathcal{N}(\mu_{2,1}, \Sigma_{2,1})$, $\mathcal{N}(\mu_{2,2}, \Sigma_{2,2})$
HMM - Domains 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 (but other data that depend on the sequence are)

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 likely many, many others...
HMM - Domains of application

It is in some applications common to consider that the hidden states have some desired meaning, and then try to estimate the hidden states from the observations.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Hidden state</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech</td>
<td>Words</td>
<td>Spectogram</td>
</tr>
<tr>
<td>Part-of-speech tagging</td>
<td>Noun/ verb/ etc</td>
<td>Words</td>
</tr>
<tr>
<td>Gene finding</td>
<td>Intron/ exon/ non-coding</td>
<td>DNA</td>
</tr>
<tr>
<td>Sequence alignment</td>
<td>Insert/ delete/ match</td>
<td>Amino acids</td>
</tr>
<tr>
<td>Robotics</td>
<td>Motion primitives</td>
<td>Joint angles</td>
</tr>
</tbody>
</table>
**HMM - Domains of application**

**Automatic speech recognition**

$\xi_t$ can represent features extracted from the speech signal, and $S_t$ can represent the word being spoken. The transition model $P(S_t | S_{t-1})$ represents the language model, and the observation model $P(\xi_t | S_t)$ represents the acoustic model.

**Activity recognition**

$\xi_t$ can represent features extracted from a video frame, and $S_t$ is the class of activity the person is engaged in (e.g., running, walking, sitting, etc.).

**Part of speech tagging**

$\xi_t$ can represent a word, and $S_t$ represents its part of speech (noun, verb, adjective, etc.).

**Gene finding**

$\xi_t$ can represent the DNA nucleotides (A,T,G,C), and $S_t$ represents whether we are inside a gene-coding region or not.

**Protein sequence alignment**

$\xi_t$ can represent an amino acid, and $S_t$ represents whether this matches the latent consensus sequence at this location.
Hidden Markov model (HMM)

\[ \Theta^{\text{GMM}} = \{\pi_i, \mu_i, \Sigma_i\}_{i=1}^{K} \]

\[ \Theta^{\text{HMM}} = \{\{a_{i,j}\}_{j=1}^{K}, \Pi_i, \mu_i, \Sigma_i\}_{i=1}^{K} \]

We will from now on consider a single Gaussian as state output

\[ \pi_i = 1 \]
Useful intermediary variables in HMM

**Forward variable**

\[ \alpha_{t,i}^{\text{HMM}} = P(s_t = i, \xi_{1:t}) \]

**Backward variable**

\[ \beta_{t,i}^{\text{HMM}} = P(\xi_{t+1:T} | s_t = i) \]

**Smoothed node marginals**

\[ \gamma_{t,i}^{\text{HMM}} = P(s_t = i | \xi_{1:T}) \]

**Smoothed edge marginals**

\[ \zeta_{t,i,j}^{\text{HMM}} = P(s_t = i, s_{t+1} = j | \xi_{1:T}) \]
Forward algorithm

The probability to be in state $i$ at time step $t$ given the partial observation $\xi_{1:t} = \{\xi_1, \xi_2, \ldots, \xi_t\}$ can be computed with the forward variable

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

which can be used to compute

$$\mathcal{P}(s_t = i \mid \xi_{1:t}) = \frac{\mathcal{P}(s_t = i, \xi_{1:t})}{\mathcal{P}(\xi_{1:t})} = \frac{\alpha_{t,i}^{\text{HMM}}}{\sum_{k=1}^{K} \alpha_{t,k}^{\text{HMM}}}$$

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

The recursion can be derived by using the chain rule and writing

\[
\mathcal{P}(s_t, \xi_{1:t}) = \sum_{s_{t-1}=1}^K \mathcal{P}(s_t, s_{t-1}, \xi_{1:t})
\]

\[
\mathcal{P}(a, b) = \frac{\mathcal{P}(b|a)\mathcal{P}(a)}{= \sum_{s_{t-1}=1}^K \mathcal{P}(\xi_t \mid s_t, s_{t-1}, \xi_{1:t-1}) \mathcal{P}(s_t \mid s_{t-1}, \xi_{1:t-1}) \mathcal{P}(s_{t-1}, \xi_{1:t-1})}
\]

Since \( \xi_t \) is conditionally dependent only on \( s_t \), and \( s_t \) is conditionally dependent only on \( s_{t-1} \), the above relation simplifies to

\[
\mathcal{P}(s_t, \xi_{1:t}) = \mathcal{P}(\xi_t \mid s_t) \sum_{s_{t-1}=1}^K \mathcal{P}(s_t \mid s_{t-1}) \mathcal{P}(s_{t-1}, \xi_{1:t-1})
\]

Since \( \mathcal{P}(\xi_t \mid s_t) \) and \( \mathcal{P}(s_t \mid s_{t-1}) \) are the emission and transition probabilities, \( \mathcal{P}(s_t, \xi_{1:t}) \) can be computed from \( \mathcal{P}(s_{t-1}, \xi_{1:t-1}) \).
Forward algorithm

\[ \alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t=i, \xi_{1:t}) \]

The \textit{forward} variable can thus be computed recursively with

\[ \alpha_{t,i}^{\text{HMM}} = \left( \sum_{j=1}^{K} \alpha_{t-1,j}^{\text{HMM}} a_{j,i} \right) \mathcal{N}(\xi_t | \mu_i, \Sigma_i) \]

by starting from

\[ \alpha_{1,i}^{\text{HMM}} = \prod_i \mathcal{N}(\xi_1 | \mu_i, \Sigma_i) \]

It can be used to evaluate trajectories by computing the likelihood

\[ \mathcal{P}(\xi | \Theta^{\text{HMM}}) = \sum_{i=1}^{K} \alpha_{T,i}^{\text{HMM}} \]
Forward algorithm

\[ \alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i, \xi_1:t) \]

\[ \alpha_{t,i}^{\text{HMM}} = \left( \sum_{j=1}^{K} \alpha_{t-1,j}^{\text{HMM}} a_{j,i} \right) \mathcal{N}(\xi_t | \mu_i, \Sigma_i) \text{ with } \alpha_{1,i}^{\text{HMM}} = \prod_i \mathcal{N}(\xi_1 | \mu_i, \Sigma_i) \]
Influence of transition probabilities against emission probabilities in HMM

\[
\alpha_{t,i}^{\text{HMM}} = \left( \sum_{j=1}^{K} \alpha_{t-1,j}^{\text{HMM}} a_{j,i} \right) \mathcal{N}(\xi_t | \mu_i, \Sigma_i)
\]

Learned transition probabilities

Transition probabilities manually set

The color of each datapoint corresponds to the value of the forward variable $\alpha$
Useful intermediary variables in HMM

**Forward variable**

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

**Backward variable**

$$\beta_{t,i}^{\text{HMM}} = \mathcal{P}(\xi_{t+1:T} \mid s_t = i)$$

**Smoothed node marginals**

$$\gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i \mid \xi_{1:T})$$

**Smoothed edge marginals**

$$\zeta_{t,i,j}^{\text{HMM}} = \mathcal{P}(s_t = i, s_{t+1} = j \mid \xi_{1:T})$$
Backward algorithm

\[ \beta_{t,i}^{\text{HMM}} = \mathcal{P}(\xi_{t+1:T} \mid s_t = i) \]

Similarly, we can define a **backward variable** starting from

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

and computed as

\[ \beta_{t,i}^{\text{HMM}} = \sum_{j=1}^{K} a_{i,j} \mathcal{N}(\xi_{t+1} \mid \mu_j, \Sigma_j) \beta_{t+1,j}^{\text{HMM}} \]

corresponding to the probability of the partial observation \( \{\xi_{t+1}, \ldots, \xi_{T-1}, \xi_T\} \), knowing that we are in state \( i \) at time step \( t \).
Backward algorithm

$$\beta_{t,i}^{\text{HMM}} = \mathcal{P}(\xi_{t+1:T} \mid s_t = i)$$

$$\beta_{t,i}^{\text{HMM}} = \sum_{j=1}^{K} a_{i,j} \mathcal{N}(\xi_{t+1} \mid \mu_j, \Sigma_j) \beta_{t+1,j}^{\text{HMM}} \quad \text{with} \quad \beta_{T,i}^{\text{HMM}} = 1$$
Useful intermediary variables in HMM

Forward variable

\[ \alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i, \xi_{1:t}) \]

Backward variable

\[ \beta_{t,i}^{\text{HMM}} = \mathcal{P}(\xi_{t+1:T} \mid s_t = i) \]

Smoothed node marginals

\[ \gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i \mid \xi_{1:T}) \]

Smoothed edge marginals

\[ \zeta_{t,i,j}^{\text{HMM}} = \mathcal{P}(s_t = i, s_{t+1} = j \mid \xi_{1:T}) \]
Smoothed node marginals

Given the full observation $\xi = \{\xi_1, \xi_2, \ldots, \xi_T\}$, the probability of $\xi_t$ to be in state $i$ at time step $t$ is

$$\gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i \mid \xi_{1:T}) = \frac{\mathcal{P}(s_t = i, \xi_{1:T})}{\mathcal{P}(\xi_{1:T})}$$

\[
\mathcal{P}(a, b) = \mathcal{P}(b \mid a) \mathcal{P}(a)
\]

\[
= \frac{\mathcal{P}(\xi_{1:T} \mid s_t = i) \mathcal{P}(s_t = i)}{\mathcal{P}(\xi_{1:T})}
\]

\[
\mathcal{P}(b \mid a) = \frac{\mathcal{P}(a, b)}{\mathcal{P}(a)}
\]

Conditional independence property

\[
\mathcal{P}(a, b) = \mathcal{P}(b \mid a) \mathcal{P}(a)
\]

\[
= \frac{\mathcal{P}(s_t = i, \xi_{1:t}) \mathcal{P}(\xi_{t+1:T} \mid s_t = i) \mathcal{P}(s_t = i)}{\mathcal{P}(\xi_{1:T})}
\]

\[
\alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i, \xi_{1:t})
\]

\[
= \frac{\alpha_{t,i}^{\text{HMM}} \beta_{t,i}^{\text{HMM}}}{\sum_{k=1}^{K} \alpha_{t,k}^{\text{HMM}} \beta_{t,k}^{\text{HMM}}}
\]

\[
\beta_{t,i}^{\text{HMM}} = \mathcal{P}(\xi_{t+1:T} \mid s_t = i)
\]
Smoothed node marginals

\[ \gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i \mid \xi_{1:T}) \]

\[
\gamma_{t,i}^{\text{HMM}} = \frac{\alpha_{t,i}^{\text{HMM}} \beta_{t,i}^{\text{HMM}}}{\sum_{k=1}^{K} \alpha_{t,k}^{\text{HMM}} \beta_{t,k}^{\text{HMM}}} = \frac{\alpha_{t,i}^{\text{HMM}} \beta_{t,i}^{\text{HMM}}}{\mathcal{P}(\xi)}
\]
Useful intermediary variables in HMM

Forward variable

$$\alpha_{t,i}^{HMM} = \mathcal{P}(s_t = i, \xi_{1:t})$$

Backward variable

$$\beta_{t,i}^{HMM} = \mathcal{P}(\xi_{t+1:T} \mid s_t = i)$$

Smoothed node marginals

$$\gamma_{t,i}^{HMM} = \mathcal{P}(s_t = i \mid \xi_{1:T})$$

Smoothed edge marginals

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

\[ 
\zeta_{t,i,j}^{\text{HMM}} = \mathcal{P}(s_t = i, s_{t+1} = j \mid \xi_{1:T}) 
\]

Given the full observation \( \xi = \{\xi_1, \xi_2, \ldots, \xi_T\} \), the probability to be in state \( i \) at time step \( t \) and in state \( j \) at time step \( t+1 \) is

\[ 
\zeta_{t,i,j}^{\text{HMM}} = \mathcal{P}(s_t = i, s_{t+1} = j \mid \xi_{1:T}) 
\]

\[ 
= \frac{\mathcal{P}(s_t = i, s_{t+1} = j, \xi_{1:T})}{\mathcal{P}(\xi_{1:T})} 
\]

\[ 
\mathcal{P}(b \mid a) = \frac{\mathcal{P}(a,b)}{\mathcal{P}(a)} 
\]

\[ 
= \frac{\alpha_{t,i}^{\text{HMM}} a_{i,j} \mathcal{N}(\xi_{t+1} \mid \mu_j, \Sigma_j) \beta_{t+1,j}^{\text{HMM}}}{\sum_{k=1}^{K} \sum_{l=1}^{K} \alpha_{t,k}^{\text{HMM}} a_{k,l} \mathcal{N}(\xi_{t+1} \mid \mu_l, \Sigma_l) \beta_{t+1,l}^{\text{HMM}}} 
\]

Note that we have \( \gamma_{t,i}^{\text{HMM}} = \sum_{j=1}^{K} \zeta_{t,i,j}^{\text{HMM}} \)
Smoothed edge marginals $\zeta_{t,i,j}^{\text{HMM}} = \mathcal{P}(s_t=i, s_{t+1}=j | \xi_{1:T})$

This result can be retrieved by rewriting the numerator with Bayes rules and the conditional independence properties of the model

$$\mathcal{P}(s_t, s_{t+1}, \xi_{1:T}) = \mathcal{P}(\xi_{1:T} | s_t, s_{t+1}) \mathcal{P}(s_t, s_{t+1})$$

$$= \mathcal{P}(\xi_{1:t} | s_t, s_{t+1}) \mathcal{P}(\xi_{t+1} | s_t, s_{t+1}) \mathcal{P}(\xi_{t+2:T} | s_{t+1}) \mathcal{P}(s_{t+1} | s_t) \mathcal{P}(s_t)$$

**Conditional independence property**

$$= \mathcal{P}(\xi_{1:t} | s_t) \mathcal{P}(\xi_{t+1} | s_{t+1}) \mathcal{P}(\xi_{t+2:T} | s_{t+1}) \mathcal{P}(s_{t+1} | s_t) \mathcal{P}(s_t)$$

$$= \mathcal{P}(s_t, \xi_{1:t}) \underbrace{\mathcal{P}(\xi_{t+1} | s_{t+1})}_{\alpha_{t,i}^{\text{HMM}}} \underbrace{\mathcal{N}(\xi_{t+1} | \mu_j, \Sigma_j)}_{\beta_{t+1,j}^{\text{HMM}}} \mathcal{P}(s_{t+1} | s_t)$$

$$\mathcal{P}(\xi_{t+1} | s_{t+1} = j) = \mathcal{N}(\xi_{t+1} | \mu_j, \Sigma_j)$$

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

$$\alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i, \xi_{1:t}) = \mathcal{P}(\xi_{1:t} | s_t = i) \mathcal{P}(s_t = i)$$

$$\beta_{t+1,j}^{\text{HMM}} = \mathcal{P}(\xi_{t+2:T} | s_{t+1} = j)$$

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

\[
\zeta_{t,i,j}^{\text{HMM}} = \frac{\alpha_{t,i}^{\text{HMM}} a_{i,j} \mathcal{N}(\xi_{t+1} | \mu_j, \Sigma_j) \beta_{t+1,j}^{\text{HMM}}}{\sum_{k=1}^{K} \sum_{l=1}^{K} \alpha_{t,k}^{\text{HMM}} a_{k,l} \mathcal{N}(\xi_{t+1} | \mu_l, \Sigma_l) \beta_{t+1,l}^{\text{HMM}}} 
\]

\[
= \frac{\alpha_{t,i}^{\text{HMM}} a_{i,j} \mathcal{N}(\xi_{t+1} | \mu_j, \Sigma_j) \beta_{t+1,j}^{\text{HMM}}}{\mathcal{P}(\xi)} 
\]
**EM for HMM**

The expected complete data log-likelihood is

\[ Q(\Theta, \Theta^{old}) = \mathbb{E} \left[ \sum_{m=1}^{M} \sum_{t=1}^{T_m} \log \mathcal{P}(\xi_{m,t}, s_t | \Theta) \mid \xi, \Theta^{old} \right] \]

\[ = \sum_{i=1}^{K} \mathbb{E}[N_{i}^{1}] \log \Pi_i + \sum_{i=1}^{K} \sum_{j=1}^{K} \mathbb{E}[N_{i,j}] \log a_{i,j} \]

\[ + \sum_{m=1}^{M} \sum_{t=1}^{T_m} \sum_{i=1}^{K} \mathcal{P}(s_t = i \mid \xi_m, \Theta^{old}) \log \mathcal{N}(\xi_{m,t} \mid \mu_i, \Sigma_i) \]

with expected counts given by

\[ \mathbb{E}[N_{i}^{1}] = \sum_{m=1}^{M} \mathcal{P}(s_{m,1} = i \mid \xi_m, \Theta^{old}) = \sum_{m=1}^{M} \gamma_{m,1,i}^{HMM} \]

\[ \mathbb{E}[N_{i,j}] = \sum_{m=1}^{M} \sum_{t=1}^{T_m-1} \mathcal{P}(s_{m,t} = i, s_{m,t+1} = j \mid \xi_m, \Theta^{old}) = \sum_{m=1}^{M} \sum_{t=1}^{T_m-1} \zeta_{m,t,i,j}^{HMM} \]
EM for HMM

By setting

\[
\frac{\partial Q(\Theta, \Theta^{\text{old}})}{\partial \Pi_i} = 0 \quad \frac{\partial Q(\Theta, \Theta^{\text{old}})}{\partial a_{i,j}} = 0
\]

a result similar to the case of Markov models is obtained.

The maximum likelihood estimate of the initial state distribution and transition probability parameters can thus be computed with the normalized counts

\[
\hat{\Pi}_i = \frac{\mathbb{E}[N^1_i]}{\sum_{k=1}^{K} \mathbb{E}[N^1_k]} = \frac{\mathbb{E}[N^1_i]}{M}, \quad \hat{a}_{i,j} = \frac{\mathbb{E}[N_{i,j}]}{\sum_{k=1}^{K} \mathbb{E}[N_{i,k}]} = \frac{\mathbb{E}[N_{i,j}]}{\mathbb{E}[N_i]}
\]

with \( \mathbb{E}[N_i] = \sum_{m=1}^{M} \sum_{t=1}^{T_m} P(s_{m,t} = i \mid \xi_m, \Theta^{\text{old}}) = \sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma^{\text{HMM}}_{m,t,i} \)
EM for HMM - Summary

M-step:

\[ \Pi_i \leftarrow \frac{\sum_{m=1}^{M} \gamma_{m,1,i}^{\text{HMM}}}{M} \]

Total number of times in i at time step 1
Total number of trajectories

\[ a_{i,j} \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_{m-1}} \zeta_{m,t,i,j}^{\text{HMM}}}{\sum_{m=1}^{M} \sum_{t=1}^{T_{m-1}} \gamma_{m,t,i}^{\text{HMM}}} \]

Total number of transitions from i to j
Total number of times in i (and transit to anything else)

\[ \mu_i \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_{m}} \gamma_{m,t,i}^{\text{HMM}} \xi_{m,t}}{\sum_{m=1}^{M} \sum_{t=1}^{T_{m}} \gamma_{m,t,i}^{\text{HMM}}} \]

By setting \( \frac{\partial Q(\Theta, \Theta^{old})}{\partial \mu_i} = 0 \) and \( \frac{\partial Q(\Theta, \Theta^{old})}{\partial \Sigma_i} = 0 \), a result similar to GMM is obtained.

\[ \Sigma_i \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_{m}} \gamma_{m,t,i}^{\text{HMM}} (\xi_{m,t} - \mu_i)(\xi_{m,t} - \mu_i)^\top}{\sum_{m=1}^{M} \sum_{t=1}^{T_{m}} \gamma_{m,t,i}^{\text{HMM}}} \]
EM for HMM - Summary

\( K \) Gaussians

\( M \) trajectories

\( T_m \) points per traj.

M-step:

\[
\Pi_i \leftarrow \frac{\sum_{m=1}^{M} \gamma_{m,1,i}^{HMM}}{M}
\]

\[
\alpha_{i,j} \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m-1} \zeta_{m,t,i,j}^{HMM}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m-1} \gamma_{m,t,i}^{HMM}}
\]

\[
\mu_i \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}^{HMM} \xi_{m,t}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}^{HMM}}
\]

\[
\Sigma_i \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}^{HMM} (\xi_{m,t} - \mu_i)(\xi_{m,t} - \mu_i)^T}{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}^{HMM}}
\]

These results can be formally retrieved with EM (also called Baum-Welch algorithm in the context of HMM).

We gave here the interpretation in terms of normalized counts, with several weighted averages computed based on the responsibility of each datapoint in each cluster.
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 basic scaling procedure is to multiply $\alpha_{t,i}^{HMM}$ by a factor independent of $i$, and divide $\beta_{t,i}^{HMM}$ 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

$$c_t = \frac{1}{\sum_{i=1}^{K} \alpha_{t,i}^{HMM}}$$
Numerical underflow issue in HMM

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

We can see by induction that

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

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

$$\gamma_{t,i}^{\text{HMM}} = \frac{\hat{\alpha}_{t,i}^{\text{HMM}} \hat{\beta}_{t,i}^{\text{HMM}}}{\sum_{k=1}^{K} \hat{\alpha}_{t,k}^{\text{HMM}} \hat{\beta}_{t,k}^{\text{HMM}}} = \frac{\left( \prod_{s=1}^{t} c_s \right) \left( \prod_{s=t}^{T} c_s \right) \alpha_{t,i}^{\text{HMM}} \beta_{t,i}^{\text{HMM}}}{\sum_{k=1}^{K} \alpha_{t,k}^{\text{HMM}} \beta_{t,k}^{\text{HMM}}} = \frac{\alpha_{t,i}^{\text{HMM}} \beta_{t,i}^{\text{HMM}}}{\sum_{k=1}^{K} \alpha_{t,k}^{\text{HMM}} \beta_{t,k}^{\text{HMM}}}$$
Summary - Why did we introduce these four intermediary variables in HMM?

**Forward variable**

\[ \alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i, \xi_{1:t}) \]

**Backward variable**

\[ \beta_{t,i}^{\text{HMM}} = \mathcal{P}(\xi_{t+1:T} \mid s_t = i) \]

**Smoothed node marginals**

\[ \gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t = i \mid \xi_{1:T}) \]

**Smoothed edge marginals**

\[ \zeta_{t,i,j}^{\text{HMM}} = \mathcal{P}(s_t = i, s_{t+1} = j \mid \xi_{1:T}) \]
Summary - Why did we introduce these four intermediary variables in HMM?

How to estimate the parameters of an HMM?

→ Maximum of expected complete data log-likelihood \( Q(\Theta, \Theta^{\text{old}}) \)

How to compute \( \frac{\partial Q}{\partial \Pi_i} = 0, \frac{\partial Q}{\partial a_{i,j}} = 0, \frac{\partial Q}{\partial \mu_i} = 0 \) and \( \frac{\partial Q}{\partial \Sigma_i} = 0 \)?

→ Requires to compute \( \zeta_{t,i,j}^{\text{HMM}} = \mathcal{P}(s_t=i, s_{t+1}=j \mid \xi_{1:T}) \)

→ Requires to compute \( \gamma_{t,i}^{\text{HMM}} = \mathcal{P}(s_t=i \mid \xi_{1:T}) \)

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

→ Requires to compute \( \alpha_{t,i}^{\text{HMM}} = \mathcal{P}(s_t=i, \xi_{1:t}) \)

→ Requires to compute \( \beta_{t,i}^{\text{HMM}} = \mathcal{P}(\xi_{t+1:T} \mid s_t=i) \)
Viterbi decoding

(MAP vs MPE estimates)

Maximum a posteriori  
Most probable explanation

[demo_HMM_Viterbi01.m]

Viterbi decoding (MAP vs MPE estimates)

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

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

\[
\hat{s}^{\text{MPE}} = \left\{ \arg \max_{s_1} \mathcal{P}(s_1|\xi), \arg \max_{s_2} \mathcal{P}(s_2|\xi), \ldots, \arg \max_{s_T} \mathcal{P}(s_T|\xi) \right\}
\]

If there is a unique mode, they can provide the same result, but not in the general case with multiple equally probably assignments.

While \( \hat{s}^{\text{MPE}} \) can be computed by replacing the sum operator with a max operator in \( \gamma^{\text{HMM}} \), the Viterbi algorithm used to compute \( \hat{s}^{\text{MAP}} \) employs the max operator in a forward pass, followed by a backward pass using a traceback procedure to recover the most probable path.
Viterbi decoding (MAP vs MPE estimates)
Viterbi decoding - Algorithm

Initialization:
\[ \delta_{1,i} = \Pi_i \mathcal{N}(\xi_1 | \mu_i, \Sigma_i) \]
\[ \Psi_{1,i} = 0 \]

Recursion:
\[ \delta_{t,i} = \max_j (\delta_{t-1,j} a_{j,i}) \mathcal{N}(\xi_t | \mu_i, \Sigma_i) \]
\[ \Psi_{t,i} = \arg \max_j (\delta_{t-1,j} a_{j,i}) \quad \forall t \in \{2, 3, \ldots, T\} \]

Termination:
\[ \hat{s}^{MAP}_T = \arg \max_j \delta_{T,j} \]

Backtracking:
\[ \hat{s}^{MAP}_t = \Psi_{t+1, \hat{s}^{MAP}_{t+1}} \quad \forall t \in \{T-1, T-2, \ldots, 1\} \]

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

It tells us the most likely previous state on the most probable path to \( s_t = i \)

Here, \( \delta_{t,i} = \max_{s_{1:t-1}} \mathcal{P}(s_{1:t-1}, s_t = i | \xi_{1:t}) \), and \( \Psi_{t,i} \) are state indices that keep track of the states \( j \) that maximized \( \delta_{t,i} \).
Viterbi decoding - Trellis representation

\[
d_{t,i} = \max_j (\delta_{t-1,j} a_{j,i}) \mathcal{N}(\xi_t | \mu_i, \Sigma_i)
\]

\[
\Psi_{t,i} = \arg \max_j (\delta_{t-1,j} a_{j,i})
\]

\[
\hat{s}_{t}^{\text{MAP}} = \Psi_{t+1, \hat{s}_{t+1}^{\text{MAP}}}
\]
Numerical underflow issue in Viterbi

Similarly to the forward-backward variables in HMM, we have to worry about numerical underflow in the practical implementation of Viterbi decoding.

One way is to normalize $\delta_{t,i}$ at each time step $t$ 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

\[
\log \delta_{t,i} = \max_{s_{1:t-1}} \log \mathcal{P}(s_{1:t-1}, s_t = i \mid \xi_{1:t})
\]

\[
= \max_j \left( \log \delta_{t-1,j} + \log a_{i,j} \right) + \log \mathcal{N}(\xi_t \mid \mu_i, \Sigma_i)
\]

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

It is for this reason that it is also common to use the Viterbi algorithm in the E step of the EM procedure instead of the forward-backward variables when training high dimensional systems based on HMMs.
Viterbi decoding - Example

\[ \xi = \{C1, C3, C4, C6\} \]

\[ \Pi = [1, 0, 0] \]

\[ \delta_{t,i} = \max_j (\delta_{t-1,j} a_{j,i}) \ b(\xi_t | i) \]

\[ a_{j,i} \]

\[ b(\xi_t | i) \]

\[ \delta_{t,i} \]

\[ S1 \ 0.5 \]

\[ S1 \ 0.045 \]

\[ S1 \ 0.0 \]

\[ S2 \ 0.0 \]

\[ S2 \ 0.07 \]

\[ S2 \ 0.0441 \]

\[ S3 \ 0.0 \]

\[ S3 \ 0.0 \]

\[ S3 \ 0.0007 \]

\[ S3 \ 0.0022 \]

\[ t = 1 \]

\[ t = 2 \]

\[ t = 3 \]

\[ t = 4 \]
Hidden semi-Markov model (HSMM)

demo_HSMMM01.m

State duration probability in standard HMM

The state duration follows a geometric distribution

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

By artificially duplicating the number of states while keeping the same emission distribution, other state duration distributions can be modeled.
Another approach is to provide an explicit model of the state duration instead of relying on self-transition probabilities.
Hidden semi-Markov model (HSMM)

\[ \Theta^{\text{GMM}} = \{ \pi_i, \mu_i, \Sigma_i \}_{i=1}^K \]

\[ \Theta^{\text{HMM}} = \{ \{ a_{i,j} \}_{j=1}^K, \Pi_i, \mu_i, \Sigma_i \}_{i=1}^K \]

\[ \Theta^{\text{HSMM}} = \{ \{ a_{i,j} \}_{j=1, j \neq i}^K, \Pi_i, \mu_i^P, \Sigma_i^P, \mu_i, \Sigma_i \}_{i=1}^K \]

Single Gaussian as state output

\[ \pi_i \]

Single Gaussian as state output

\[ \mathcal{N}(\mu_1, \Sigma_1) \]

\[ \mathcal{N}(\mu_2, \Sigma_2) \]

\[ \mathcal{N}(\mu_1^P, \Sigma_1^P) \]

\[ \mathcal{N}(\mu_2^P, \Sigma_2^P) \]
Hidden semi-Markov model (HSMM)

While the HMM computes the *forward* variable as

\[ \alpha_{t,i}^{\text{HMM}} = \sum_{j=1}^{K} \alpha_{t-1,j}^{\text{HMM}} \ a_{j,i} \ \mathcal{N}_{t,i}, \quad \text{with} \quad \mathcal{N}_{t,i} = \mathcal{N}(\xi_t | \mu_i, \Sigma_i) \]

the HSMM requires the evaluation of

\[ \alpha_{t,i}^{\text{HSMM}} = \sum_{d=1}^{d_{\text{max}}} \sum_{j=1}^{K} \alpha_{t-1,j}^{\text{HSMM}} \ a_{(j,d),(i,d)} \ \mathcal{N}_{(t,d),i} \]

which requires the system to keep an history of length \( d_{\text{max}} \).

\( a_{(j,d),(i,d)} \) is the probability to be in state \( i \) at iterations \([t+1, \ t+d]\) knowing that we were in state \( j \) at iterations \([t-d+1, \ t]\).

\( \mathcal{N}_{(t,d),i} \) is the probability to observe \( \{\xi_{t-d+1}, \xi_{t-d+2}, \ldots, \xi_t\} \) knowing that we were in state \( i \) at iterations \([t-d+1, \ t]\).
Hidden semi-Markov model (HSMM)

An explicit-duration HSMM with, for example*, a Gaussian parametrization of the duration $\mathcal{N}_{d,i}^p = \mathcal{N}(d | \mu_i^p, \Sigma_i^p)$ assumes that

$$a_{(j,d),(i,d)} = a_{j,i} \mathcal{N}_{d,i}^p$$

and

$$\mathcal{N}_{(t,d),i} = \prod_{s=t-d+1}^{t} \mathcal{N}_{s,i}$$

which corresponds to the assumption that the state duration is dependent on the current state and independent on the previous state, and that the outputs are conditionally independent.

* Used here only for convenience purpose, since other distributions from the exponential family would better model positive counts (e.g., log-normals or gamma distributions).
Hidden semi-Markov model (HSMM)

The probability to be in state $i$ at time step $t$ given the partial observation $\xi_{1:t} = \{\xi_1, \xi_2, \ldots, \xi_t\}$ can then be recursively computed

$$
\mathcal{P}(s_t = i \mid \xi_{1:t}) = \frac{\alpha_{t,i}^{\text{HSMM}}}{\sum_{k=1}^{K} \alpha_{t,k}^{\text{HSMM}}}, \quad \text{with}
$$

$$
\alpha_{t,i}^{\text{HSMM}} = \sum_{d=1}^{d_{\max}} \sum_{j=1}^{K} \alpha_{t-d,j}^{\text{HSMM}} a_{j,i} \mathcal{N}_{d,i}^D \prod_{s=t-d+1}^{t} \mathcal{N}_{s,i}, \quad \text{where}
$$

$$
\mathcal{N}_{d,i}^D = \mathcal{N}(d \mid \mu_i^D, \Sigma_i^D) \quad \text{and} \quad \mathcal{N}_{s,i} = \mathcal{N}(\xi_s \mid \mu_i, \Sigma_i)$$
Hidden semi-Markov model (HSMM)

For $t < d_{\text{max}}$, the initialization is given by

$$\alpha_{1,i}^{\text{HSMM}} = \prod_i \mathcal{N}_{1,i}^D \mathcal{N}_{1,i}$$

$$\alpha_{2,i}^{\text{HSMM}} = \prod_i \mathcal{N}_{2,i}^D \prod_{s=1}^{2} \mathcal{N}_{s,i} + \sum_{j=1}^{K} \alpha_{1,j}^{\text{HSMM}} a_{j,i} \mathcal{N}_{1,i}^D \mathcal{N}_{2,i}$$

$$\alpha_{3,i}^{\text{HSMM}} = \prod_i \mathcal{N}_{3,i}^D \prod_{s=1}^{3} \mathcal{N}_{s,i} + \sum_{j=1}^{K} \sum_{d=1}^{2} \sum_{s=4-d}^{3} \alpha_{3-d,j}^{\text{HSMM}} a_{j,i} \mathcal{N}_{d,i}^D \prod_{s=4-d}^{3} \mathcal{N}_{s,i}$$

which corresponds to an update rule for $t < d_{\text{max}}$ written as

$$\alpha_{t,i}^{\text{HSMM}} = \prod_i \mathcal{N}_{t,i}^D \prod_{s=1}^{t} \mathcal{N}_{s,i} + \sum_{j=1}^{K} \sum_{d=1}^{t-1} \sum_{s=t-d+1}^{t} \alpha_{t-d,j}^{\text{HSMM}} a_{j,i} \mathcal{N}_{d,i}^D \prod_{s=t-d+1}^{t} \mathcal{N}_{s,i}$$
GMM/HMM/HSMM with dynamic features

(Trajectory-HMM)

demo_trajHSM01.m

GMM/HMM/HSMM with dynamic features
In the field of speech processing, the extraction of statistics from both static and dynamic features within an HMM is very common.

In particular, it can be used in speech synthesis to avoid discontinuities in the generated speech spectra.

The synthesized speech then becomes natural and smooth even when a small number of Gaussians is used.

This is achieved by coordinating the distributions of both static and dynamic features (the dynamic features are often called delta and delta-delta parameters).

In speech processing, these parameters usually correspond to the evolution of mel-frequency cepstral coefficients characterizing the power spectrum of a sound, but the same approach can be used with any form of continuous signals.
GMM/HMM/HSMM with dynamic features

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

$$\dot{x}_t = \frac{x_{t+1} - x_t}{\Delta t}$$

where $x_t$ is a multivariate position vector.

The acceleration is similarly computed as

$$\ddot{x}_t = \frac{\dot{x}_{t+1} - \dot{x}_t}{\Delta t} = \frac{x_{t+2} - 2x_{t+1} + x_t}{\Delta t^2}$$
GMM/HMM/HSMM with dynamic features

A vector $\zeta_t$ will be used to represent the concatenated position, velocity and acceleration vectors at time step $t$

$$\zeta_t = \begin{bmatrix} x_t \\ \dot{x}_t \\ \ddot{x}_t \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ -\frac{1}{\Delta t} I & \frac{1}{\Delta t^2} I & \frac{1}{\Delta t^2} I \\ \frac{1}{\Delta t^2} I & -\frac{1}{\Delta t^2} I & \frac{1}{\Delta t^2} I \end{bmatrix} \begin{bmatrix} x_t \\ x_{t+1} \\ x_{t+2} \end{bmatrix}$$

Here, the number of derivatives will be set up to acceleration ($C = 3$), but the results can easy be generalized to a higher or lower number of derivatives.
GMM/HMM/HSMM with dynamic features

\( \zeta \) and \( x \) are then defined as large vectors concatenating \( \zeta_t \) and \( x_t \) for all time steps

\[
\begin{bmatrix}
\zeta_1 \\
\zeta_2 \\
\vdots \\
\zeta_T
\end{bmatrix}
\quad \quad
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_T
\end{bmatrix}
\]

Similarly to the matrix operator defined in the previous slide for a single time step, a large sparse matrix \( \Phi \) can be defined so that

\[ \zeta = \Phi x \]
GMM/HMM/HSMM with dynamic features

\[ \begin{bmatrix} \zeta \\ \vdots \\ \mathbf{x}_t \\ \dot{\mathbf{x}}_t \\ \ddot{\mathbf{x}}_t \\ \mathbf{x}_{t+1} \\ \dot{\mathbf{x}}_{t+1} \\ \ddot{\mathbf{x}}_{t+1} \\ \vdots \end{bmatrix} = \begin{bmatrix} \Phi \\ \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ \mathbf{x}_{t+1} \\ \mathbf{x}_{t+2} \\ \mathbf{x}_{t+3} \\ \vdots \end{bmatrix} \]

\( \zeta \in \mathbb{R}^{DCT} \)  
(C=3 here) 

\( \Phi \in \mathbb{R}^{DCT \times DT} \) 

\( \mathbf{x} \in \mathbb{R}^{DT} \)

Note that \( \Phi \) is a big sparse matrix that can automatically be constructed through the use of Kronecker products.
GMM/HMM/HSMM with dynamic features

D=1, C=3, T=10

D=2, C=3, T=10

D=2, C=3, T=100

Sparse matrices $\Phi \in \mathbb{R}^{DCT \times DT}$
GMM/HMM/HSMM with dynamic features

For a sequence of states \( s = \{s_1, s_2, \ldots, s_T\} \) of \( T \) time steps, with discrete states \( s_t \in \{1, \ldots, K\} \), the likelihood of a movement \( \zeta = \Phi x \) is given by

\[
\mathcal{P}(\zeta | s) = \prod_{t=1}^{T} \mathcal{N}(\zeta_t | \mu_{s_t}, \Sigma_{s_t})
\]

where \( \mu_{s_t} \) and \( \Sigma_{s_t} \) are the center and covariance of state \( s_t \) at time step \( t \). This product can be rewritten as

\[
\mathcal{P}(\Phi x | s) = \mathcal{N}(\Phi x | \mu_s, \Sigma_s)
\]

with \( \mu_s = \begin{bmatrix} \mu_{s_1} \\ \mu_{s_2} \\ \vdots \\ \mu_{s_T} \end{bmatrix} \) and \( \Sigma_s = \begin{bmatrix} \Sigma_{s_1} & 0 & \cdots & 0 \\ 0 & \Sigma_{s_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Sigma_{s_T} \end{bmatrix} \)
GMM/HMM/HSMM with dynamic features

By using the relation $\zeta = \Phi x$, we seek to retrieve a trajectory

$$\hat{x} = \arg \max_x \log \mathcal{P}(\Phi x \mid s)$$

Equating to zero the derivative of

$$\log \mathcal{P}(\Phi x \mid s) = -\frac{1}{2}(\Phi x - \mu_s)\Sigma_s^{-1}(\Phi x - \mu_s)$$

$$\frac{\partial}{\partial x} x^\top A x = (A + A^\top) x$$

with respect to $x$ yields

$$\Phi^\top \Sigma_s^{-1}(\Phi x - \mu_s) = 0$$

$$\iff \hat{x} = (\Phi^\top \Sigma_s^{-1} \Phi)^{-1} \Phi^\top \Sigma_s^{-1} \mu_s$$
The covariance error of this estimate is given by

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

where $\sigma$ is a scale factor.

The resulting Gaussian $\mathcal{N}(\hat{x}, \hat{\Sigma}^x)$ forms a trajectory distribution, where $\hat{x} \in \mathbb{R}^{DT}$ is a trajectory stored in a vector form.
GMM/HMM/HSMM with dynamic features

The trajectory distribution can be multimodal (multiple path options)
Main references

Hidden Markov model (HMM)

Hidden semi-Markov model (HSMM)

GMM/HMM/HSMM with dynamic features (Trajectory HMM)
General textbooks

Machine Learning
A Probabilistic Perspective
Kevin P. Murphy

Pattern Recognition
and Machine Learning
Christopher M. Bishop

Finite Mixture Models
Geoffrey McLachlan
David Peel

WILEY SERIES IN PROBABILITY AND STATISTICS
Advanced research topics  (not covered in the course)

Bayesian Nonparametrics applied to HMM (model selection)

Spectral learning of HMM parameters

Task-parameterized HMM