Topics

- Structured probability models
  - Markov models
  - Hidden markov models
Markov chain: review

- A first order (homogeneous) Markov chain:

The initial state $s_0$ is drawn from $P_0(s_0)$. Successive states are drawn from the one step transition probabilities $P_1(s_{t+1}|s_t)$.
Markov chain: properties

\[ P(s_t \mid s_{t-1}) \]

\[ P_0(s) \]

\[ P_1(s_t \mid s_{t-1}) \]

\[ s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots \]

- If there exists a finite \( k \) such that any state \( i \) can lead to any other state \( j \) after exactly \( k \) steps, the Markov chain is \textit{ergodic}:

\[ P(s_{t+k} = j \mid s_t = i) > 0 \] for all \( i, j \) and some finite \( k \)
Markov chains

- Problems we have to solve
  1. Prediction
  2. Estimation

- **Prediction**: what is the probability distribution over the possible states $s_{t+k}$ at time $t+k$ if we start from $s_t = i$?

\[
P_1(s_{t+1} | s_t = i) \\
P_2(s_{t+2} | s_t = i) = \sum_{s_{t+1}} P_1(s_{t+1} | s_t = i) P_1(s_{t+2} | s_{t+1}) \\
\ldots \\
P_k(s_{t+k} | s_t = i) = \sum_{s_{t+k-1}} P_{k-1}(s_{t+k-1} | s_t = i) P_1(s_{t+k} | s_{t+k-1})
\]

where $P_k(s' | s)$ is the k-step transition probability matrix.
Markov chain: estimation

- We need to estimate the initial state distribution $P_0(s_0)$ and the transition probabilities $P_1(s'|s)$
- Estimation from $L$ observed sequences of different lengths

\[ s_0^{(1)} \rightarrow s_1^{(1)} \rightarrow s_2^{(1)} \rightarrow \ldots \rightarrow s_{n_1}^{(1)} \]

\[ \ldots \]

\[ s_0^{(L)} \rightarrow s_1^{(L)} \rightarrow s_2^{(L)} \rightarrow \ldots \rightarrow s_{n_L}^{(L)} \]

Maximum likelihood estimates (observed fractions)

\[ \hat{P}_0(s_0 = i) = \frac{1}{L} \sum_{l=1}^{L} \delta(s_0^{(l)}, i) \]

where $\delta(x, y) = 1$ if $x = y$ and zero otherwise
Markov chain: estimation

\[ s_0^{(1)} \rightarrow s_1^{(1)} \rightarrow s_2^{(1)} \rightarrow \ldots \rightarrow s_{n_1}^{(1)} \]

\[ \ldots \]

\[ s_0^{(L)} \rightarrow s_1^{(L)} \rightarrow s_2^{(L)} \rightarrow \ldots \rightarrow s_{n_L}^{(L)} \]

- The transition probabilities are obtained as observed fractions of transitions out of a specific state

Joint estimate over successive states

\[ \hat{P}_{s,s'}(s = i, s' = j) = \frac{1}{\left( \sum_{l=1}^{L} n_l \right)} \sum_{l=1}^{L} \sum_{t=0}^{n_l-1} \delta(s_t^{(l)}, i)\delta(s_{t+1}^{(l)}, j) \]

and the transition probability estimates

\[ \hat{P}_1(s' = j \mid s = i) = \frac{\hat{P}_{s,s'}(s = i, s' = j)}{\sum_k \hat{P}_{s,s'}(s = i, s' = k)} \]
Markov chain: estimation

• Can we simply estimate Markov chains from a single long sequence?

\[ s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_n \]

– Ergodicity?
– What about the initial state distribution \( \hat{P}_0(s_0) \)?
Clustering by dynamics

- We can cluster time course signals by means of comparing their dynamics, where the dynamics is captured by a Markov chain model
  - system behavior monitoring (anomaly detection)
  - biosequences
  - etc.

- There are still many ways of using the Markov chain models for clustering (e.g., what is the clustering metric?)

- The approach we follow here is to derive a criterion for determining whether two (or more) sequences should be in the same cluster
Cluster criterion

• How can we tell whether two arbitrary sequences
  \[ S^{(1)} = \{s_0^{(1)}, \ldots, s_{n_1}^{(1)}\} \text{ and } S^{(2)} = \{s_0^{(2)}, \ldots, s_{n_2}^{(2)}\} \]
  should be in the same cluster?

• We can compare (approximate) description lengths of either
  encoding the sequencies separately or jointly
  \[ DL^{(1)} + DL^{(2)} \geq DL^{(1+2)} \]
  where \( DL^{(1+2)} \) uses the same Markov chain for both
  sequencies while \( DL^{(1)} \) and \( DL^{(2)} \) are based on different
  models.
Cluster criterion cont’d

• Approximate description lengths:

\[
DL^{(1)} + DL^{(2)} = -\log P(S^{(1)}|\hat{\theta}_1) + \frac{d}{2}\log(n_1)
- \log P(S^{(2)}|\hat{\theta}_2) + \frac{d}{2}\log(n_2)
\]

\[
DL^{(1+2)} = -\log P(S^{(1)}|\hat{\theta}) - \log P(S^{(2)}|\hat{\theta})
+ \frac{d}{2}\log(n_1 + n_2)
\]

where the maximum likelihood parameter estimates \(\hat{\theta}_1, \hat{\theta}_2,\) and \(\hat{\theta}\) each include the initial state distribution and the transition probabilities; \(d = 3\) for binary sequences.

• We are essentially testing here whether the two sequences have the same first order Markov dynamics.
Simple example

- Four binary sequences of length 50:
  1. 001001100100010100000100001110110101010100...
  2. 01011111110100110101000010000001010111001...
  3. 11010110000001101100100011011111010111101...
  4. 1101010111101011110111101101101101000101...

Evaluations:

\[
\begin{align*}
DL^{(1)} + DL^{(2)} - DL^{(1+2)} &= 6.6 \text{ bits} \\
DL^{(1+2)} + DL^{(3+4)} - DL^{(1+2+3+4)} &= -0.9 \text{ bits}
\end{align*}
\]

Agglomerative hierarchical clustering with Euclidean distance would give (((2, 3), 4), 1)
Beyond Markov chains

- Potential problems with using Markov chains
  - if the state is continuous
  - if we cannot fully determine what the current state is (e.g., due to noisy observations)
  - if the state is an abstraction and never directly observable

- We need to augment the markov chain with a model that relates the states to observables
A hidden Markov model (HMM) is a model where we generate a sequence of outputs in addition to the Markov state sequence

\[
\begin{array}{cccc}
    s_0 & \rightarrow & s_1 & \rightarrow & s_2 & \rightarrow & \ldots \\
    \downarrow & & \downarrow & & \downarrow & & \\
    x_0 & & x_1 & & x_2 & & \\
\end{array}
\]

- number of states \( m \)
- initial state distribution \( P_0(s_0) \)
- state transition model \( P_1(s_{t+1} | s_t) \)
- output model \( P_o(x_t | s_t) \) (discrete or continuous)

This is a latent variable model in the sense that we will only observe the outputs \( \{x_0, x_1, \ldots, x_n\} \); the state sequence remains “hidden”
HMM example

• Two states 1 and 2; observations are tosses of unbiased coins

\[ P_0(s = 1) = 0.5, \quad P_0(s = 2) = 0.5 \]
\[ P_1(s' = 1|s = 1) = 0, \quad P_1(s' = 2|s = 1) = 1 \]
\[ P_1(s' = 1|s = 2) = 0, \quad P_1(s' = 2|s = 2) = 1 \]
\[ P_o(x = \text{heads}|s = 1) = 0.5, \quad P_o(x = \text{tails}|s = 1) = 0.5 \]
\[ P_o(x = \text{heads}|s = 2) = 0.5, \quad P_o(x = \text{tails}|s = 2) = 0.5 \]

• This model is \textit{unidentifiable} in the sense that the particular hidden state Markov chain has no effect on the observations
HMM example: biased coins

- Two states 1 and 2; outputs are tosses of *biased* coins

\[
P_0(s = 1) = 0.5, \quad P_0(s = 2) = 0.5
\]
\[
P_1(s' = 1|s = 1) = 0, \quad P_1(s' = 2|s = 1) = 1
\]
\[
P_1(s' = 1|s = 2) = 0, \quad P_1(s' = 2|s = 2) = 1
\]
\[
P_o(x = \text{heads}|s = 1) = 0.25, \quad P_o(x = \text{tails}|s = 1) = 0.75
\]
\[
P_o(x = \text{heads}|s = 2) = 0.75, \quad P_o(x = \text{tails}|s = 2) = 0.25
\]

- What type of output sequences do we get from this HMM model?
HMM example

- Continuous output model: $x = [x_1, x_2]$, $P_o(x|s)$ is a Gaussian with mean and covariance depending on the underlying state $s$. Each state is initially equally likely.

- How does this compare to a mixture of four Gaussians model?
HMM problems

• There are several problems we have to solve
  1. How do we evaluate the probability that our model generated the observation sequence \( \{x_0, x_1, \ldots, x_n\} \)?
     – forward-backward algorithm
  2. How do we uncover the most likely hidden state sequence corresponding to these observations?
     – dynamic programming
  3. How do we adapt the parameters of the HMM to better account for the observations?
     – the EM-algorithm
Probability of observed data

• In principle computing the probability of the observed sequence involves summing over exponentially many possible hidden state sequences

\[ P(x_0, \ldots, x_n) = \sum_{s_0, \ldots, s_n} \text{Prob. of obs. and a hidden state sequence} \]

\[ P_0(s_0)P_1(x_0|s_0) \cdots P_1(s_n|s_{n-1})P_0(x_n|s_n) \]

\[
\begin{align*}
  s_0 & \rightarrow s_1 & \rightarrow s_2 & \rightarrow \ldots & \rightarrow s_{n-1} & \rightarrow s_n \\
  \downarrow & & & & & \\
  x_0 & \rightarrow x_1 & \rightarrow x_2 & \rightarrow x_{n-1} & \rightarrow x_n
\end{align*}
\]

• We can, however, exploit the structure of the model to evaluate the probability much more efficiently
Forward-backward algorithm

\[ s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_{n-1} \rightarrow s_n \]
\[ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \]
\[ x_0 \quad x_1 \quad x_2 \quad \ldots \quad x_{n-1} \quad x_n \]

- **Forward (predictive) probabilities** \( \alpha_t(i) \):
  \[
  \alpha_t(i) = P(x_0, \ldots, x_t, s_t = i)
  \]
  \[
  \frac{\alpha_t(i)}{\sum_j \alpha_t(j)} = P(s_t = i | x_0, \ldots, x_t)
  \]

- **Backward propabilities** \( \beta_t(i) \):
  \[
  \beta_t(i) = P(x_{t+1}, \ldots, x_n | s_t = i)
  \]
  (evidence about the current state from future observations)

- Both can be updated *recursively*
Recursive forward updates

\begin{center}
\begin{tikzpicture}[inner sep = 0.5cm]
\node [cylindrical, draw] (s0) at (0,0) {$s_0$};
\node [cylindrical, draw] (s1) at (1.5,0) {$s_1$};
\node [cylindrical, draw] (s2) at (3,0) {$s_2$};
\node [circle, draw] (x0) at (0,-1) {$x_0 = heads$};
\node [circle, draw] (x1) at (1.5,-1) {$x_1 = tails$};
\node [circle, draw] (x2) at (3,-1) {$x_2 = heads$};
\draw [->] (s0) to (x0);
\draw [->] (s0) to (s1);
\draw [->] (s1) to (x1);
\draw [->] (s1) to (s2);
\draw [->] (s2) to (x2);
\end{tikzpicture}
\end{center}

- Forward recursion: $\alpha_t(i) = P(x_0, \ldots, x_t, s_t = i)$

\begin{align*}
\alpha_0(1) & = P_0(1) P_o(heads|1) \\
\alpha_0(2) & = P_0(2) P_o(heads|2) \\
\alpha_1(1) & = \left[ \alpha_0(1) P_1(1|1) + \alpha_0(2) P_1(1|2) \right] P_o(tails|1) \\
\alpha_1(2) & = \left[ \alpha_0(1) P_1(2|1) + \alpha_0(2) P_1(2|2) \right] P_o(tails|2)
\end{align*}
Recursive forward updates cont’d

$S_0$, $S_1$, $S_2$

$x_0 = heads$, $x_1 = tails$, $x_2 = heads$

- Recursive updates for $\alpha_t(i) = P(x_0, \ldots, x_t, s_t = i)$:

$$\alpha_0(i) = P_0(s_0 = i) P_o(x_0 | s_0 = i)$$

$$\alpha_t(i) = \left[ \sum_j \alpha_{t-1}(j) P_1(s_t = i | s_{t-1} = j) \right] P_o(x_t | s_t = i)$$
Recursive backward updates

\[ s_0 \quad s_1 \quad s_2 \]
\[ x_0 = \text{heads}, \quad x_1 = \text{tails}, \quad x_2 = \text{heads} \]

- Backward recursion: \( \beta_t(i) = P(x_{t+1}, \ldots, x_n | s_t = i) \)

\[
\beta_2(1) = 1 \\
\beta_2(2) = 1 \\
\beta_1(1) = P_1(1|1)P_o(\text{heads}|1)\beta_2(1) + P_1(2|1)P_o(\text{heads}|2)\beta_2(2) \\
\beta_1(2) = P_1(1|2)P_o(\text{heads}|1)\beta_2(1) + P_1(2|2)P_o(\text{heads}|2)\beta_2(2)
\]
Recursive backward updates cont’d

\[ \begin{align*}
x_0 &= \text{heads}, \quad x_1 = \text{tails}, \quad x_2 = \text{heads} \\
\beta_n(i) &= 1 \\
\sum_j P_1(s_t = j | s_{t-1} = i) P_o(x_t | s_t = j) \beta_t(j)
\end{align*} \]

- Recursive updates for \( \beta_t(i) = P(x_{t+1}, \ldots, x_n | s_t = i) \):

- Recursive backward updates cont’d

\[ \begin{align*}
\beta_n(i) &= 1 \\
\beta_{t-1}(i) &= \sum_j P_1(s_t = j | s_{t-1} = i) P_o(x_t | s_t = j) \beta_t(j)
\end{align*} \]