Review

* Hidden Markov models (HMMs)

Observations: $O_t$

States: $S_t$

* Parameters

$T_{ij} = P(S_{t+1} = j | S_t = i)$

$A_{ij} = P(S_{t+1} = j | S_t = i)$

$B_{ik} = P(O_t = k | S_t = i)$

* Key computations

1. How to compute likelihood $P(O_1, O_2, ..., O_T)$?
2. How to compute $\arg\max_{S_1, ..., S_T} P(S_1, ..., S_T | O_1, ..., O_T)$?
3. How to estimate (learn) $T_{ij}$, $A_{ij}$, $B_{ik}$?

* HW problem: belief updating
  - recursion for $P(S_t | O_1, O_2, ..., O_t)$
  - important for real-time monitoring

(2) Computing most likely state sequence

$l_{i*}^* = \max_{S_1, ..., S_T} \log P(S_1, S_2, ..., S_T, S_T = s, O_1, ..., O_T)$

Recursion: $l_{j+1}^* = \max_i [l_{ij}^* + \log A_{ij}] + \log B_{ji}(O_{t+1})$

How to derive $S_t^*$ from $l_t^*$?

Discrete state sequence → real-valued matrix

* Record most likely state transitions:

$\Phi_{t+1}(j) = \arg\max_i [l_i^* + \log A_{ij}]$

What is most likely state at time $t$ given state $j$ at time $t+1$ with observations $O_1, O_2, ..., O_T$.

* Compute $S^*$ by backtracking:

$S_T^* = \arg\max_i [l_i^*]$  

$S_t^* = \Phi_{t+1}(S_{t+1}^*) \quad t = T-1, T-2, ..., 1$

$S^* = S_1^*, S_2^*, ..., S_T^*$ is known as Viterbi path.

Viterbi algorithm is instance of dynamic programming.

(3) Learning in HMMs.

Given: sequence of observations $O_1, O_T$  
Goal: estimate $(T_{ij}, A_{ij}, B_{ik})$ to maximize $P(O_1, O_2, ..., O_T)$

Fixed: number of hidden states $n$. 
* Shared CPTs to estimate:  \( T_i = P(S_i = i) \)
   \( a_{ij} = P(S_{t+1} = j \mid S_t = i) \)
   \( b_{ik} = P(O_t = k \mid S_t = i) \)

* E-step

Compute \( P(S_1 = i \mid O_1, \ldots, O_T) \)

\[
P(S_{t+1} = j \mid O_1, \ldots, O_T)
\]

\[
P(S_t = i, O_t = k \mid O_1, \ldots, O_T) = P(S_t = i \mid O_1, \ldots, O_T) \times P(O_t = k \mid S_t = i)
\]

(analogy: \( p \alpha , \times , V \))

* Compute posterior probabilities

Analogous to \( \alpha_{it} = P(O_1, O_2, \ldots, O_t \mid S = i) \) before and up to time \( t \)

define \( \beta_{it} = P(O_{t+1}, O_{t+2}, \ldots, O_T \mid S = i) \) after time \( t \)

Starting at \( t+1 \) conditioning on \( S_t = i \)

- Recursion for \( \beta_{it} \):

1. base case \( \beta_{it} = P(\ldots \mid S_t = i) \) ?
   \[ \beta_{it} = 1 \text{ for all } i. \]

2. backwards step

   \[ \beta_{it} = P(O_{t+1}, \ldots, O_T \mid S_t = i) \]
   \[ = \sum_{j=1}^{n} P(O_{t+1}, \ldots, O_T, S_{t+1} = j \mid S_t = i) \text{ marginalization} \]
   \[ = \sum_{j=1}^{n} P(O_{t+1}, \ldots, O_T \mid S_{t+1} = j, S_t = i) P(S_{t+1} = j \mid S_t = i) \text{ product rule} \]
   \[ = \sum_{j=1}^{n} P(O_{t+1} \mid S_{t+1} = j) P(O_t, \ldots, O_T \mid S_{t+1} = j, O_{t+1}) P(S_{t+1} = j \mid S_t = i) \text{ product rule} \]
   \[ = \sum_{j=1}^{n} a_{ij} b_j \beta_{j,t+1} \]

Computing \( \alpha \) \& \( \beta \) matrix \( \leftrightarrow \) "forward-backward" algorithm \( \text{o.k.a. Baum-Welch} \)

- Posterior probabilities in E-step:

\[
P(S_t = i, S_{t+1} = j \mid O_1, \ldots, O_T) = \frac{P(S_t = i, S_{t+1} = j, O_1, \ldots, O_T)}{P(O_1, \ldots, O_T)}
\]

\[
P(O_1, \ldots, O_T, S_t = i) P(S_{t+1} = j \mid S_t = i, O_1, \ldots, O_T) P(O_{t+1} \mid S_{t+1} = i, O_1, \ldots, O_T)
\]

\[= \frac{\alpha_{it} a_{ij} b_j \beta_{j,t+1}}{\sum_{i, j} \alpha_{it} a_{ij} \beta_{j,t+1}} \]

Other posteriors in E-step

\[
P(S_t = i \mid O_1, \ldots, O_T) = \sum_{j=1}^{n} P(S_t = i, S_{t+1} = j \mid O_1, \ldots, O_T)
\]

\[
P(S_t = i \mid O_1, \ldots, O_T) \text{ special case with } t = 1
\]
* M-step updates

\[ T_i \leftarrow P(S_t = i \mid O_1, \ldots, O_T) \]

\[ a_{ij} \leftarrow \frac{\sum_{t=1}^{T} P(S_t = i, S_{t+1} \neq j \mid O_1, \ldots, O_T)}{\sum_{t=1}^{T} P(S_t = i \mid O_1, \ldots, O_T)} \]

\[ b_{ik} \leftarrow \frac{\sum_{t=1}^{T} P(S_t = i \mid O_1, \ldots, O_T) I(k, O_t)}{\sum_{t=1}^{T} P(S_t = i \mid O_1, \ldots, O_T)} \]

* Complexity of HMM computations

(i) to compute \( P(O_1, O_2, \ldots, O_T) \)

(ii) to decode \( S^* = \arg \max_S P(S \mid O) \) \( O(n^T) \)

iii) parameter update of EM

**Multivariate Gaussian distributions**

* Random variable \( \mathbf{x} \in \mathbb{R}^n \) (real-valued vector)

* Probability density function (PDF)

\[ P(\mathbf{x}) = \frac{1}{(2\pi)^{n/2} \sqrt{\det(\Sigma)}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right\} \]

* Parameters

\( \mu = \mathbb{E}[\mathbf{x}] = \int P(\mathbf{x}) \mathbf{x} \ d\mathbf{x} \)

\( \Sigma_{ij} = \mathbb{E}[(\mathbf{x} - \mu)(\mathbf{x} - \mu)^T] = \int P(\mathbf{x})(x_i - \mu_i)(x_j - \mu_j) \ d\mathbf{x} \)

Shorthand: \( P(\mathbf{x}) = \mathcal{N}(\mathbf{x} ; \mu, \Sigma) \)

* Mathematical properties:

(i) if \( P(\mathbf{x}) \) and \( P(\mathbf{y}) \) are Gaussian PDFs over \( \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \),

then \( P(ax + \beta y) \) is also Gaussian.

where \( a, \beta \) are linear scalar coefficients.

(ii) if \( P(\mathbf{x}) \) is Gaussian over \( \mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n \),

then all marginals \( P(x_i), P(x_i, x_j), \ldots \) and all conditionals \( P(x_i | x_j), P(x_i, x_j | x_k), \ldots \) are also Gaussian.

* Maximum likelihood estimation

Given i.i.d. data \( \{ \mathbf{x}_t \}_{t=1}^T \) where \( \mathbf{x}_t \in \mathbb{R}^n \), how to choose \( \mu, \Sigma \)

to maximize

\[ P(\text{data}) = \prod_{t=1}^{T} \mathcal{N}(\mathbf{x}_t ; \mu, \Sigma) \]

- Log-likelihood

\[ \mathcal{L} = \log P(\text{data}) = \sum_{t=1}^{T} \log \mathcal{N}(\mathbf{x}_t ; \mu, \Sigma) \]
To maximize:\n\[ \frac{\partial L}{\partial \mu} = 0, \quad \frac{\partial L}{\partial \Sigma} = 0. \]
\[ \mu = \frac{1}{n} \sum_{i} x_i \text{ sample mean} \]
\[ \Sigma = \frac{1}{n} \sum_{i} (x_i - \mu)(x_i - \mu)^T \text{ sample covariance.} \]

**Clustering**

*Inputs of \( \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \) with \( \vec{x} \in \mathbb{R}^k \)

*Goal: partition inputs into \( k \) clusters

**Gaussian mixture model**

* DAG

\[ Z \in \{1, 2, \ldots, k\} \]

(observed)

(hidden)

cluster label

* CPTs

\[ P(Z = i) \text{ fraction of data in cluster } i \]

\[ P(X^i | Z = i) = N(\mu_i, \Sigma_i) \text{ cluster-dependent means and covariance matrices} \]

* Aside: ML estimation for complete data \( \hat{f}(\vec{x}_i, Z_i) \)

Let \( T_i = \sum_{t=1}^{T} I(Z_t = i) \text{ count of label } i \)

\[ P(Z_i) = \frac{1}{n} \]

\[ \mu_i = \frac{1}{n} \sum_{i} x_i I(Z_t = i) \]

\[ \Sigma_{ij} = \frac{1}{n} \sum_{i} (x_i - \mu_i)(x_i - \mu_i)^T I(Z_t = i) \]

Ex: \( n=2 \) dimensions

\[ \mu_1 \]

\[ \mu_2 \]

\[ k=3 \text{ clusters} \]