Recall (i) representation/structure (ii) inference, and (iii) learning. So far, we have mostly considered (i) and (ii). Now we address learning, which is developed for speech. Can be applied to vision if we have a 1-D structure or can approximate 1-D structure.

Discrete Markov Processes

An *N*-disturbed state $S_1, ..., S_N$.

State at time $t$ is $q_t$. $q_t = S_i$ means system is in state $S_i$.

$P(q_{t+1} = S_j | q_t = S_i, q_{t-1} = S_k, \ldots)$

first-order Markov model.

$P(q_{t+1} = S_j | q_t = S_i, q_{t-1} = S_k, \ldots) = P(q_{t+1} = S_j, q_t = S_i)$

Future is independent of the past, except for the preceding time state.

Continue with first-order Markov state.

Transition probabilities $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$

$a_{ij} > 0$ and $\sum_j a_{ij} = 1$, for all $i$.

Transition probability is independent of time.
Initial probability. \( \Pi_i = P(q_i = s_i) \) \( \sum_{i=1}^n \Pi_i = 1 \)

In an observable Markov model, we can directly observe the states \( \{q_i\} \).
(This enables us to learn the transition probs.)

Observed sequence \( O = Q = \langle q_1, \ldots, q_T \rangle \)
\[
P(O = \mathbf{q} | \Lambda, \Pi) = \Pi q_1 \prod_{t=2}^{T} P(q_t | q_{t-1}) = \prod_{t=2}^{T} \Pi q_t q_{t-1} \]

Example: Urns with 3 colors of balls
\( S_1 = \text{red} \), \( S_2 = \text{blue} \), \( S_3 = \text{green} \) \( \{\text{state: the urn we draw the ball from}\} \)

Initial prob \( \Pi = [0.5, 0.2, 0.3] \)

Transition matrix \( \Lambda = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \)

Sequence \( O = \langle s_1, s_2, s_3, s_4 \rangle \)
\[
P(\mathbf{O} | \Lambda, \Pi) = P(s_1) P(s_1 | s_2) P(s_3 | s_2) P(s_4 | s_3) = \Pi_1 \cdot 0.4 \cdot 0.2 \cdot 0.1 \cdot 0.1 = 0.008.\]
Learning Parameters for HMM.

Suppose we have $K$ sequences of length $T$.

$q_t^k$ is state at time $t$ of $k$th sequence.

$$
\hat{\pi}_i = \frac{\# \text{[sequence starting with } S_i]}{\# \text{[number of sequences]}} = \frac{1}{K} \sum_{k=1}^{K} I(q_1^k = S_i)
$$

$$
\hat{\alpha}_{ij} = \frac{\# \text{[transition from } S_i \text{ to } S_j]}{\# \text{[transition from } S_i]} = \frac{1}{K} \sum_{k=1}^{K} \sum_{t=1}^{T-1} I(q_t^k = S_i \text{ and } q_{t+1}^k = S_j)
$$

\text{e.g. } \hat{\alpha}_{12} \text{ is no. of times a blue ball follows a red ball divided by the total no. of red balls.}

Note: these learning formula are intuitive, but it is important to realize that they are obtained by ML (max likelihood).

$$
\hat{A}, \hat{\pi} = \arg \max \prod_{k=1}^{K} P(O = Q_k | A, \pi)
$$
(4) **Hidden Markov Model**

States are not directly observable, but we have an observation from each state.

- State: \( q_t \in \{ S_1, \ldots, S_n \} \)
- Observe: \( O_t \in \{ v_1, \ldots, v_m \} \)

\[ b_j(m) = P(O_t = v_m | q_t = S_j) \]

The observation probability. We observe \( v_m \) if the state is \( S_j \).

I.e. two sources of stochasticity:

- The observation is stochastic: \( b_j(m) \)
- The transition is stochastic: \( a_{ij} \).

**Back to the urn analogy:**

Let the urn contain balls with different colours. → E.G. urn 1 mostly red

\[ \begin{align*}
\text{Urn 1} & \quad 	ext{mostly red} \\
\text{Urn 2} & \quad 	ext{mostly blue} \\
\text{Urn 3} & \quad 	ext{mostly green}
\end{align*} \]

The observation is the ball colour, but we don’t know which urn it comes from (the state).
Elements:

1. N: No. of states
   \[ S = \{ S_1, \ldots, S_N \} \]
2. \( \mathbf{N} \): No. of observation symbols in alphabet
   \[ \mathbf{V} = \{ v_1, v_2, \ldots, v_M \} \]
3. State transition probabilities:
   \[ A = \{ a_{ij} \} \quad a_{ij} = P(q_{t+1} = S_j | q_t = S_i) \]
4. Observation probabilities:
   \[ \mathbf{B} = \{ b_j(m) \} \quad b_j(m) = P(o_t = v_m | q_t = S_j) \]
5. Initial state probability:
   \[ \pi = \{ \pi_i \} \quad \text{where} \quad \pi_i = P(q_1 = S_i) \]

\( \lambda = (A, \mathbf{B}, \pi) \) specifies the parameter set of an HMM.

Three basic problems:

1. Given a model \( \lambda \), evaluate the
   probability \( P(O|\lambda) \) of any sequence \( O = (o_1, o_2, \ldots, o_T) \).
2. Given a model and observation sequence, find state sequence \( Q = (q_1, q_2, \ldots, q_T) \), which has highest probability of generating \( O \).
   \[ q^* = \arg \max P(O|q^*, \lambda) \]
   \[ Q^* = \arg \max P(Q|O, \lambda) \]
3. Given training set of sequences \( \mathcal{X} = \{ O^k \} \), find
   \[ \lambda^* = \arg \max P(\mathcal{X} | \lambda) \].
Problem 1. Evaluation

Given an observation \( O = \{o_1, o_2, \ldots, o_T\} \)
and a state sequence \( Q = \{q_1, q_2, \ldots, q_T\} \)
the prob. of observing \( O \) given \( Q \) is

\[
P(O|Q, \lambda) = \prod_{t=1}^{T} P(D_{t}|q_t, \lambda) = b_{q_1}(o_1) \cdot b_{q_2}(o_2) \cdots b_{q_T}(o_T)
\]

But we don't know \( Q \).

The prior prob. of the state sequence \( Q \)

\[
P(Q|\lambda) = P(q_1) \cdot \prod_{t=2}^{T} P(q_t|q_{t-1}) = a_{q_1} \cdot \prod_{t=2}^{T} a_{q_t|q_{t-1}}
\]

Joint prob. is

\[
P(O, Q|\lambda) = P(q_1) \cdot \prod_{t=2}^{T} P(q_t|q_{t-1}) \cdot \prod_{t=1}^{T} P(o_t|q_t)
\]

\[
= \prod_{t=1}^{T} b_{q_t}(o_t) \cdot \prod_{t=2}^{T} a_{q_t|q_{t-1}} \cdot b_{q_1}(o_1)
\]

We can compute

\[
P(O|\lambda) = \sum_{all \ possible \ Q} P(O, Q|\lambda)
\]

But this summation is impractical directly, because there are too many possible \( Q \).
(7) \[ \text{HMM's} \]

But there is an efficient procedure to calculate \( P(012) \) called the forward-backward procedure (essentially - dynamic programming). This exploits the structure of the distribution.

Divide the sequence \( 0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow \ldots \) into parts that are independent.

Forward variable \( \alpha_t(i) \) is prob. of observing the partial sequence \( <0_1, \ldots, 0_t> \) and being in state \( S_i \) at time \( t \), given the model \( A \):

\[ \alpha_t(i) = P(0_1, \ldots, 0_t, q_t = S_i | \lambda) \]

This can be computed recursively:

Initially:

\[ \alpha_1(i) = P(0_1, q_1 = S_i | \lambda) = P(0_1, q_1 = S_i, \lambda) P(q_1 = S_i | \lambda) = \prod_i b_i(0_1) \]

Recursion:

\[ \alpha_{t+1}(j) = \left[ \sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] \cdot b_j(0_{t+1}) \]

See book for details.
HMM's

Intuition: $a_t(i)$ explains first + observation and ends in state $S_i$ multiply by prob $a_{ij}$ to get to state $S_j$ at $t+1$ multiply by prob of generating $(t+1)^{th}$ observation $b_j(O_{t+1})$ then sum over all possible states $S_i$ at time $t$.

Finally, $P(O_{1:T}) = \sum_{i=1}^{N} P(O_{1:T}, q_t = S_i) = \sum_{i=1}^{N} a_t(i)$

Computing $a_t(i)$ is $O(N^2T)$

This solves the first problem – computing the prob of generating the data given the model.

An alternative algorithm (which we need later) is backward variable $\beta_t(i)$

$\beta_t(i) = P(O_{t+1}, \ldots, O_T | q_t = S_i, \theta)$

Initialize $\beta_T(i) = 1$

Recurrent $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$
HMM's

Finding the State Sequence. - 2nd Problem

Again exploit the linear structure.

Greedy

Define \( \gamma_t(i) \) to be prob of state \( S_t \) at time \( t \) given \( O \) and \( \lambda \).

\[
\gamma_t(i) = \frac{P(q_t = S_t \mid O, \lambda)}{P(O \mid \lambda)} = \frac{P(q_t = S_t, O \mid \lambda)}{P(O \mid \lambda)} = \frac{\delta_t(i) \beta_t(i)}{\sum_{j=1}^{N} \delta_t(j) \beta_t(j)} \quad \text{normalizes to ensure } \sum_{i=1}^{N} \gamma_t(i) = 1.
\]

\( \delta_t(i) \) forward variable explains the starting part of the sequence up to time \( t \) ending in \( S_t, \) backward variable \( \beta_t(i) \) explains the remaining part of the sequence up to time \( T. \)

We can try to estimate the state by choosing \( q_t^* = \arg \max_i \delta_t(i) \) for each \( t. \)

But, this ignores the relations between neighboring states. It may be inconsistent that \( q_t^* = S_i, q_{t+1}^* = S_j \) but \( a_{ij} = 0 \)
HMM's

Viterbi Algorithm (Dynamic Programming)

Definition: $S_t(i)$ is the prob of the highest probability path that accounts for all the first $t$ observations and ends in $S_i$:

$$S_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}, q_t} P(q_1, q_2, \ldots, q_{t-1}, q_t = S_i, 0, 0, \ldots, 0, 1)$$

Calculate this recursively:

1. Initialization: $S_1(i) = \pi_i b_i(q_{01}), \psi_1(1) = 0$.

2. Recursion:
   $$S_t(j) = \max_i S_{t-1}(i) a_{ij} b_j(q_{0j})$$
   $$\psi_t(j) = \arg \max_i S_{t-1}(i) a_{ij}$$

3. Termination:
   $$\pi^* = \max_i S_T(i)$$
   $$q_t^* = \arg \max_i S_T(i)$$

4. Path (state sequence) backtracking:
   $$q^{\pi}_{t+1} = \psi_{t+1}(q^{\pi}_t), t = T-1, T-2, \ldots, 1$$

Intuition: $\psi_t(j)$ keeps track of the state that maximizes $S_t(j)$ at time $t-1$ with complexity $O(n^2)$. 
HMM's

Learning Model Parameters. Problem 3.

\[ X = \{ \theta_k \}_{k=1}^{K}, \text{ set of sequences} \]

\[ P(X, \theta) = \prod_{k=1}^{K} P(O_k | \theta_k) \]

\[ X^* = \arg \max_{X} P(X, \theta). \]

This is performed by a combination of EM and dynamic programming.

Define: \( \xi(t, i, j) \) prob. of being in state \( s_i \) at time \( t \)

and \( s_j \) at \( t+1 \), given observation \( o_t \) and \( o_{t+1} \):

\[ \xi(t, i, j) = P(q_t = s_i, q_{t+1} = s_j | O_t, O_{t+1}) \]

\[ \xi(t, i, j) = \frac{a_{ij} \beta_{ij}(O_t) \alpha_{ij}(O_t+1) \beta_{ij}(O_{t+1})}{\sum_{k} \sum_{l} a_{kl} \alpha_{kl}(O_t) \beta_{kl}(O_t+1) \beta_{kl}(O_{t+1})} \]

Note: if Markov model is observable, then both \( \alpha(t, i) \) & \( \xi(t, i, j) \) are 0/1.
HMM's

Baum-Welch algorithm $\rightarrow$ EM.

At each iteration,

E-step: compute $\xi_t(i,j)$ & $\pi_t(i)$

given current $\lambda = (A,B,T)$

M-step: recompute $\pi$ given

$\xi_t(i,j) & \pi_t(i)$

Alternate the two steps until convergence.

Indicator variables $z^t$:

$z^t_i = \begin{cases} 1, & \text{if } q_t = s_i \\ 0, & \text{otherwise} \end{cases}$

and $z^t_{i,j} = \begin{cases} 1, & \text{if } q_t = s_i \text{ and } q_{t+1} = s_j \\ 0, & \text{otherwise} \end{cases}$

(Note, these are 0/1 in case of observable Markov model)

Estimate them in the E-step as

$E[z^t_i] = \pi_t(i)$

$E[z^t_{i,j}] = \xi_t(i,j)$

In M-step, count the expected no. of transitions from $s_i$ to $s_j$ $\xi_t(i,j)$ and total no. of transitions from $s_i$ $\xi_t$ $\pi_t(i)$
$\hat{a}_{ij} = \frac{1}{\sum_{t=1}^{T} \gamma_{t}^{k} (i,j)} \sum_{t=1}^{T} \delta_{t}^{k} (c) \ (\text{real counts})$

$\hat{b}_{j}(m) = \frac{1}{\sum_{t=1}^{T} \delta_{t}^{k} (j) \ I \ (Q_{t}=u_{m})} \sum_{t=1}^{T} \delta_{t}^{k} (j)$

For multiple observation sequences:

$\chi = \langle O^{k} \rangle_{k=1}^{K}$

$P (\chi | \eta) = \prod_{k=1}^{K} P (O^{k} | \eta) .$

$\hat{a}_{ij} = \frac{1}{\sum_{k=1}^{K} \sum_{t=1}^{T} \gamma_{t}^{k} (i,j)} \sum_{k=1}^{K} \sum_{t=1}^{T} \delta_{t}^{k} (c) \gamma_{t}^{k} (j) \ I \ (O_{t}^{k}=u_{m})$

$\hat{b}_{j}(m) = \frac{1}{\sum_{k=1}^{K} \sum_{t=1}^{T} \gamma_{t}^{k} (j) \ I \ (Q_{t}=u_{m})} \sum_{k=1}^{K} \sum_{t=1}^{T} \gamma_{t}^{k} (j)$

$\hat{c}_i = \frac{1}{\sum_{k=1}^{K} \delta_{1}^{k} (i)}$
Recap:
We have given algorithm to solve the three problem:

1. Compute $P(O12)$
2. Compute $Q^* = \arg\max P(O102)$
3. Compute $\lambda^* = \arg\max P(X12)$

$P(O12)$ is used for model selection.
Suppose we have two alternative models for the data $P(O12_1), P(O12_2)$
select model 1 if $P(O12_1) > P(O12_2)$
model 2 otherwise.

 Decide which model generated the sequence.
Do this for multiple models with training data for each.

$\lambda_1, \ldots, \lambda_n = \arg\max P(X^1\lambda), P(X^n\lambda)$

Model selection
Use this to build speech and vision recognition system.

The HMM gives a good example of how the three key elements: (i) representation, (ii) inference, and (iii) learning are combined.

- Similar models (DP and EM) can be used to train stochastic context-free grammars.

- Can extend to learn the statespace using Dirichlet processes (later in course)