(1) # Hidden Markov Models (HMM's)

Note Title 5/18/2008

Recall — (i) representation/structure, (ii) inference, and (iii) learning.
So far, we have mostly considered (i) & (ii)
now we address learning.

HMM, developed for speech. Can be applied to vision
if we have a 1-D structure or can approximate a
1-D structure.

## Discrete Markov Processes

N-distinct states $S_1 \ldots, S_N$.
state at time $t$ is $q_t$ $\qquad$ $q_t = S_i$ means system in state $S_i$.

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

first-order Markov model.

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

the future is independent of the past, except for the preceeding time state.

Continue with First-order Markov model.

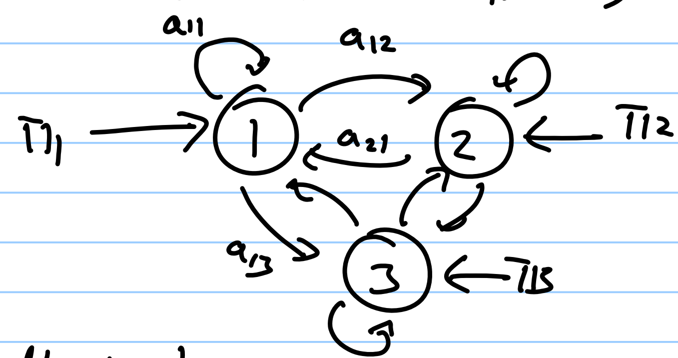
transition probabilities $\quad a_{ij} = P(q_{t+1} = S_j \mid q_t = S_i)$

$$a_{ij} \geq 0 \ \& \ \sum_{j=1}^{N} a_{ij} = 1, \text{ for all } i.$$

transition prob. is independent of time.

(2)

Initial probability. $\pi_i \equiv P(q_i = S_i)$ $\sum_{i=1}^{N} \pi_i = 1$




In an observable Markov model, we can directly observe the states $(q_t)$.
(This enables us to learn the transition probs)

Observation sequence $O = Q = \{q_1 \ldots q_T\}$

$P(O = Q \mid A, \pi) = p(q_1) \prod_{t=2}^{T} p(q_t \mid q_{t-1}) = \pi_{q_1} a_{q_1 q_2} \cdots a_{q_{T-1} q_T}$.

Example: Urns with 3 types of ball
$S_1 = $ red, $S_2 = $ blue, $S_3 = $ green || state: the urn we draw the ball from.

Initial prob $\pi = [0.5, 0.2, 0.3]$

Transition. $a_{ij}$ $A = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$

sequence $O = \{S_1, S_1, S_3, S_3\}$

$p(O \mid A, \pi) = p(S_1) P(S_1 \mid S_1) P(S_3 \mid S_1) P(S_3 \mid S_3)$
$= \pi_1 \cdot a_{11} \cdot a_{13} \cdot a_{33} = (0.5)(0.4)(0.3)(0.8)$
$= 0.048.$

13)

## Learning Parameter for HMM.

Suppose we have K sequences of length T.

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

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

$$\hat{a}_{ij} = \frac{\#[\text{transitions from } S_i \text{ to } S_j]}{\#[\text{transitions from } S_i]}$$

$$= \frac{\sum_k \sum_{t=1}^{T-1} I(q_t^k = S_i \text{ and } q_{t+1}^k = S_j)}{\sum_k \sum_{t=1}^{T-1} I(q_t^k = S_i)}$$

E.G. $\hat{a}_{12}$ 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 obtain by ML (max likelihood)

$$\hat{A}, \hat{\pi} = \text{ARG MAX} \prod_{k=1} P(O = Q_k \mid A, \pi).$$

(4)

# Hidden Markov Models HMM's

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

states $q_t \in \{S_1 \dots S_N\}$
observable $O_t \in \{v_1 \dots v_M\}$

$$b_j(m) \equiv P(O_t = v_m \mid q_t = S_j)$$

observation probability that we observe $v_m$ if the state is $S_j$.

i.e. two sources of stochasticity
the observation is stocastic. $b_j(m)$
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
Urn 2 mostly blue
Urn 3 mostly green.

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

(5)

# HMM. Formalize.

## Elements:

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

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

## Three Basic Problems

(1.) Given a model $\lambda$, evaluate the prob $P(O \mid \lambda)$ of any sequence $O = (O_1, O_2 \ldots O_T)$

(2.) Given a model and observation sequence $O$, find state sequence $Q = \{q_1, q_2 \ldots q_T\}$, which has highest probability of generating $O$

find $Q^*$ s.t. $Q^* = \text{ARG MAX } P(Q \mid O, \lambda)$

(3) Given training set of sequences $\mathcal{X} = \{O^k\}_k$

find $\lambda^* = \text{ARG MAX } P(\mathcal{X} \mid \lambda)$.

(6) HMM's

## Problem 1. Evaluation.

Given an observation $O = \{O_1, O_2 \dots O_T\}$ and a state sequence $Q = \{q_1, q_2 \dots q_T\}$ the prob. of observing O given Q is

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(O_t|q_t,\lambda) = b_{q_1}(O_1) \cdot b_{q_2}(O_2) \dots b_{q_T}(O_T)$$

But we don't know Q.
the prior prob. of the state sequence is

$$P(Q|\lambda) = P(q_1) \prod_{t=2}^{T} P(q_t|q_{t-1}) = \pi_{q_1} a_{q_1 q_2} \dots a_{q_{T-1} q_T}$$

Joint probability is

$$P(O,Q|\lambda) = P(q_1) \prod_{t=2}^{T} P(q_t|q_{t-1}) \prod_{t=1}^{T} P(O_t|q_t)$$

$$= \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \dots \dots a_{q_{T-1} q_T} b_{q_T}(O_T)$$

We can compute

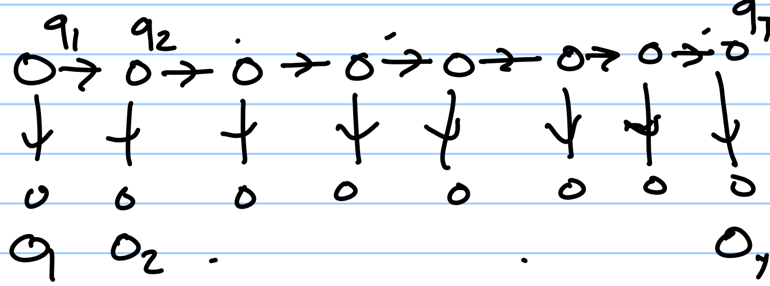
$$P(O|\lambda) = \sum_{\text{all possible } Q} P(O,Q|\lambda)$$

But this summation is impractical directly, because there are too many possible Q. $N^T$.

(7) HMM's

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

Divide the sequence into parts $t$ 1, to $t$ & $t+1$ to $T$.




Forward variable $\alpha_t(i)$ is prob. of observing the partial sequence $\{O_1 \ldots O_t\}$ and being in state $S_t$ at time $t$, (given the model $\lambda$)

$$\alpha_t(i) \equiv P(O_1 \ldots O_t, q_t = S_i | \lambda)$$

This can be computed recursively:

Initialization $\rightarrow$ $\alpha_1(t) = P(O_1, q_1 = S_i | \lambda)$
$= P(O_1 | q_1 = S_i, \lambda) P(q_1 = S_i | \lambda)$
$= \pi_i b_i(O_1)$

recursion $\alpha_{t+1}(j) = \left[ \sum_{j=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1})$

see book for details

# HMM's

Intuition: $\alpha_t(i)$ explains first $t$ observations 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**, $P(O|\lambda) = \sum_{i=1}^{N} P(O, q_T = S_i | \lambda)$

$$= \sum_{i=1}^{N} \alpha_T(i)$$

Computing $\alpha_t(i)$ 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 $\beta_t(i)$

$$\beta_t(i) \equiv P(O_{t+1}, \dots, O_T | q_t = S_i, \lambda)$$

initialize $\beta_T(i) = 1$

recursion. $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$

(9)

# HMM's

## Finding the State Sequence. – 2nd Problem

Again exploit the linear structure.

Greedy

Define $\gamma_t(i)$ is prob of state $S_i$ at time t given O and $\lambda$.

$$\gamma_t(i) = P(q_t = S_i \mid O, \lambda)$$
$$= \frac{P(O \mid q_t = S_i, \lambda)\, P(q_t = S_i \mid \lambda)}{P(O \mid \lambda)}$$
$$= \frac{\alpha_t(i)\,\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\,\beta_t(j)}$$

normalization ensures that $\sum_i \gamma_t(i) = 1$.

$\alpha_t(i)$ forward variable explains the starting part of the sequence until time t ending in $S_i$, 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 $q_t^* = \arg\max_i \gamma_t(i)$ for each t.

But, this ignores the relations between neighboring states. It may be inconsistent $q_t^* = S_i$, $q_{t+1}^* = S_j$ but $a_{ij} = 0$

(10)

# HMM's

Viterbi Algorithm (Dynamic Programming)

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

$$\delta_t(i) = \max_{q_1 \cdots q_{t-1}} P(q_1, q_2 .., q_{t-1}, q_t = S_i, O_1 \ldots O_t \mid \lambda)$$

calculate this recursively:

1. Initialize: $\delta_1(i) = \pi_i b_i(O_1)$, $\psi_1(i) = 0$.

2. Recursion. $\delta_t(j) = \max_i \delta_{t-1}(i) a_{ij} \, b_j(O_t)$

   $\psi_t(j) = \arg\max_i \delta_{t-1}(i) a_{ij}$.

3. Termination:

   $p^* = \max_i \delta_T(i)$

   $q_T^* = \arg\max_i \delta_T(i)$

4. Path (state sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2 \ldots 1$$

Intuition $\psi_t(j)$ keeps track of the state that maximizing $\delta_t(j)$ at time $t-1$

Same complexity $O(N^2 T)$.

(11)

# HMM's

## Learning Model Parameters. Problem 3.

$\chi = \{ O^k \}_{k=1}^{K}$ set of sequences.

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

$$\lambda^* = \underset{\lambda}{\text{ARG MAX}}\ P(\chi|\lambda).$$

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

<u>Defn.</u> $\xi_t(i,j)$ prob. of being in state $S_i$ at time $t$ and in state $S_j$ at $t+1$ given observation $O$ and $\lambda$:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_k \sum_l \alpha_t(k)\, a_{kl}\, b_l(O_{t+1})\, \beta_{t+1}(l)}$$

<u>Note:</u> if Markov model is observable, then both $\gamma_t(i)$ & $\xi_t(i,j)$ are 0/1

# HMM's

Baum-Welch algorithm → EM.

At each iteration,

E-step compute $\xi_t(i,j)$ & $\gamma_t(i)$ given current $\lambda = (A, B, \pi)$

M-step recalculate $\lambda$ given $\xi_t(i,j)$ & $\gamma_t(i)$

Alternate the two steps until convergence.

Indicator variables $z_i^t$

$$z_i^t = \begin{cases} 1, & \text{if } q_t = S_i \\ 0, & \text{otherwise} \end{cases}$$

and

$$z_{ij}^t = \begin{cases} 1, & \text{if } q_t = S_i \ \& \ q_{t+1} = S_j \\ 0, & \text{otherwise.} \end{cases}$$

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

Estimate them in the E-step as

$$E[z_i^t] = \gamma_t(i)$$
$$E[z_{ij}^t] = \xi_t(i,j)$$

In M-step, count the expected no. of transitions from $S_i$ to $S_j$ $\sum_t \xi_t(i,j)$ and total no. of transitions from $S_i$ is $\sum_t \gamma_t(i)$

(13)

# HMM's

This gives transition probabilities from $S_i$ to $S_j$

$$\hat{a}_{ij} = \sum_{t=1}^{T-1} \xi_t(i,j) \Big/ \sum_{t=1}^{T-1} \gamma_t(i)$$

(soft counts instead of real counts.)

$$\hat{b}_j(m) = \frac{\sum_{t=1}^{T} \gamma_t(j)\, I(O_t = v_m)}{\sum_{t=1}^{T} \gamma_t(j)}$$

For multiple observation sequences.

$$\mathcal{X} = \{O^k\}_{k=1}^{K}$$

$$P(\mathcal{X}|\lambda) = \prod_{k=1}^{K} P(O^k|\lambda).$$

$$\hat{a}_{ij} = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \xi_t^k(i,j)}{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \gamma_t^k(i)}$$

$$\hat{b}_j(m) = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \gamma_t^k(j)\, I(O_t^k = v_m)}{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \gamma_t^k(j)}$$

$$\hat{\pi}_i = \sum_{k=1}^{K} \gamma_1^k(i) \Big/ K$$

# HMM's

**Recap:**

We have given algorithm to solve the three problem:

(1) compute $P(O|\lambda)$

(2) compute $Q^* = \arg\max P(Q|O,\lambda)$

(3.) compute $\lambda^* = \arg\max P(\chi|\lambda)$

$P(O|\lambda)$ is used for model selection

Suppose we have two alternative models for the data $P(O|\lambda_1)$ $P(O|\lambda_2)$

select model 1 if $P(O|\lambda_1) > P(O|\lambda_2)$

model 2 otherwise.

i.e. detect which model generated the sequences.

Do this for multiple models with training data for each.

$$\lambda_1^*, \dots \lambda_N^* = \arg\max_{\lambda} P(\chi^1|\lambda), P(\chi^2|\lambda) \dots P(\chi^N|\lambda)$$

Model selection.

Use this to build speech and vision recognition system.

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

- Similar methods (DP & EM) can be used to train Stochastic Context Free Grammars.

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