

(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, \dots, S_N .

state at time t is q_t

$q_t = S_i$ means
system in state S_i .

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

first-order Markov model.

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

the future is independent of the past, except for
the preceding time step.

Continue with first-order Markov model.

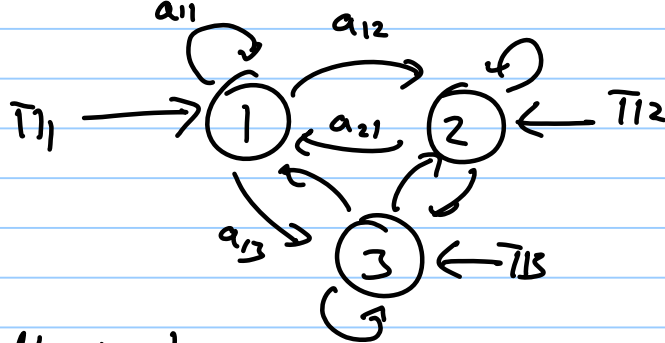
transition probabilities $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$

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

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

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

Initial prob $\pi = [0.5, 0.2, 0.3]$
 Trans. 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 $O = \{S_1, S_1, S_3, S_3\}$
 $P(O | A, \pi) = p(S_1) p(S_1 | S_1) p(S_3 | S_1) p(S_3 | S_3)$
 $= \pi_1 \cdot a_{11} \cdot a_{13} \cdot 0.8 = (0.5)(0.4)(0.3)(0.8) = 0.048$

13)

Learning Parameters for HMM.

Suppose we have K sequences of length T .
 q_t^k is state at time t of k^{th} sequence.

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

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

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 obtained by ML (max likelihood)

$$\hat{A}, \hat{\pi} = \text{ARG MAX}_{\pi} \prod_{k=1}^K P(O = Q_k | A, \pi)$$

(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 $O_t \in \{v_1 \dots v_m\}$

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

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

i.e. two sources of stochasticity

the observation is stochastic. $b_j(m)$

the transition is stochastic. a_{ij} .

Back to the urn analogy:

Let the urn contain balls with different colours. \rightarrow 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

$$S = \{S_1, \dots, S_N\}$$

2. M : No of observation symbols in alphabet

$$V = \{v_1, v_2, \dots, v_M\}$$

3. State transition probabilities:

$$A = [a_{ij}] \quad a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$$

4. Observation probabilities:

$$B = [b_{j(m)}], \quad b_{j(m)} = P(O_t = v_m | q_t = S_j)$$

5. Initial state probabilities:

$$\pi = [\pi_i] \quad \text{where } \pi_i = P(q_1 = S_i)$$

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

Three Basic Problems

(1.) Given a model λ , evaluate the

prob $P(O|\lambda)$ of any sequence $O = (O_1, O_2, \dots, O_T)$

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

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

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

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

(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

$$P(O, Q|\lambda) = P(q_1) \prod_{t=2}^T P(q_t | q_{t-1}) \prod_{t=1}^T P(O_t | q_t)$$
$$= \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \dots a_{q_{T-1} q_T} b_{q_T}(O_T)$$

We can compute

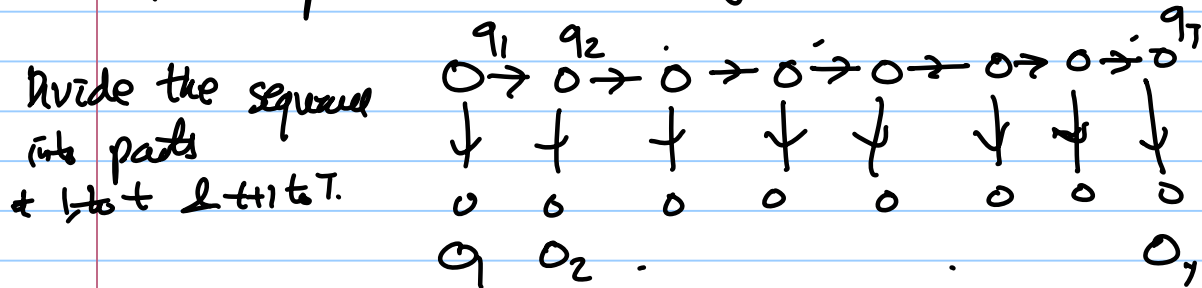
$$P(O|\lambda) = \sum_{\text{all possible } Q} P(O, Q|\lambda)$$

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 $P(O|A)$ called the forward-backward procedure (essentially - dynamic programming). This exploits the structure of the distribution



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

$$\alpha_t(i) = P(O_1, \dots, O_t, q_t = S_i | A)$$

This can be computed recursively:

Initialization \rightarrow

$$\begin{aligned} \alpha_1(i) &= P(O_1, q_1 = S_i | A) \\ &= P(O_1 | q_1 = S_i, A) P(q_1 = S_i | A) \\ &= \pi_i b_i(O_1) \end{aligned}$$

recursion

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

see book for details

(8)

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

$$\beta_T(i) = 1$$

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

(9)

HMM's

Finding the state sequence. - 2nd Problem

Again exploit the linear structure.

Greedy

Define $\delta_t(i)$ is prob of state s_i at time t given O and λ .

$$\begin{aligned}\delta_t(i) &= P(q_t = s_i | O, \lambda) \\ &= \frac{P(O | q_t = s_i, \lambda) P(q_t = s_i | \lambda)}{P(O | \lambda)} \\ &= \frac{\alpha_t(i) \beta_t(i)}{\sum_{j=1}^N \alpha_t(j) \beta_t(j)} \quad \parallel\end{aligned}$$

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

$\alpha_t(i)$ 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 $q_t^* = \arg \max_i \delta_t(i)$ 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)

Def'n

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

$$S_t(i) = \max_{q_1, \dots, q_{t-1}} P(q_1, q_2, \dots, q_{t-1}, q_t = S_i, O_1, \dots, O_t | \lambda)$$

calculate this recursively:

1. Initialize: $S_1(i) = \pi_i b_i(O_1), \psi_1(i) = 0.$

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

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

3. Termination:

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

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

4. Path (state sequence) backtracking:

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

Intuition $\psi_t(j)$ keeps track of the state that maximizes $S_t(j)$ at time $t-1$
Same complexity $O(N^2T)$.

(11)

HMM'SLearning 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.

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

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

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

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

(12)

HMM'S

Baum-Welch algorithm \rightarrow EM.

At each iteration,

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

M-step recalculate λ given
 $\xi_t(i,j)$ & $\delta_t(i)$

Alternate the two steps until convergence.

Indicator variables z_i^t

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

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

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

Estimate them in the E-step as

$$E[z_i^t] = \delta_t(i)$$

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

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 $\sum_t \delta_t(i)$

(13)

HMM'sThis gives transition probabilities from S_i to S_j

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(ij)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

(soft counts instead of real counts.)

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

For multiple observation sequences.

$$\chi = \{O^k\}_{k=1}^K$$

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

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

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

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

(14)

HMM's

Recap:

We have given algorithm to solve the three problems:

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

(2) compute $Q^* = \text{ARG MAX}_Q P(Q|O, \lambda)$

(3.) compute $\lambda^* = \text{ARG MAX}_\lambda P(\lambda|O)$

$P(O|\lambda)$ 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.

$$\lambda_1^* \dots \lambda_n^* = \text{ARG MAX}_\lambda P(\lambda^1|\lambda), P(\lambda^2|\lambda) \dots P(\lambda^n|\lambda)$$

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 state space using Markov processes (later in course)