# Discrete Markov Processes

Previous lectures

- Assume that the samples are i.i.d. (independent and identically distributed)
- But data often appears in sequences
  - There is dependence (stochastic) between different elements of the sequence

## Discrete Markov Processes

$N$-distinct state $s_1, \ldots, s_N$

State at time $t$ : $q_t$

$q_t = s_i$ : 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

$$P(q_{t+1} = s_j \mid q_t = s_i, q_{t-1} = s_k, \ldots) = P(q_{t+1} = s_j \mid q_t = s_i)$$

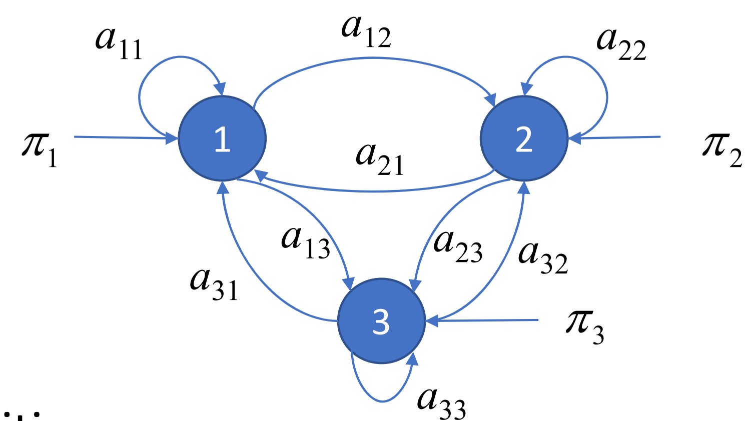
The future is independent of the past, except for the proceeding time state

Transition probability $a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i)$

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

Transition probability is independent of time

# Observable Markov Model




Initial probability $\pi_i \equiv P(q_i = s_i)$

In an observable Markov model, we can directly observe the states $\{q_t\}$

This enables us to learn the transition probabilities

Observation sequence $O = Q = \{q_1, \dots, q_T\}$

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

# Observable Markov Model

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 probability: $\pi = [0.5, 0.2, 0.3]$

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 $O=\{s_1, s_1, s_3, s_3\}$

$$P(O \mid A, \pi) = P(s_1)P(s_1 \mid s_1)P(s_3 \mid s_1)P(s_3 \mid s_3)$$
$$= \pi_1 \cdot a_{11} \cdot a_{13} \cdot a_{33} = 0.5 \times 0.4 \times 0.3 \times 0.8 = 0.048$$

# Learning Parameters for HMM

Suppose we have $K$ sequence of length $T$ ➡ $q_t$: state at time $t$ of $k^{\text{th}}$ sequence

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

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

E.G. $\hat{a}_{ij}$ is no. of times a blue ball is followed 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 (maximum likelihood)

$$\hat{A}, \hat{\pi} = \arg\max \prod_{k=1}^{K} P(O = Q_k \mid A, \pi)$$

# Hidden Markov Models (HMMs)

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

state $q_t \in \{s_1, \ldots, s_N\}$

observable $O_t \in \{v_1, \ldots, v_M\}$

$b_j(m) \equiv P(O_t = v_m \mid q_t = s_j)$ : observation prob. that we observe $v_m$ if the state is $s_j$

Two sources of stochasticity:

The observation $b_j(m)$ is stochastic
The transition $a_{ij}$ is stochastic

Back to the urn analogy: Let the urn contain balls with different colors

E.G. Urn: mostly red, Urn2: mostly blue, Urn3: mostly green

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

# Hidden Markov Models

**Elements:**

1. N: Number of states $S = \{s_1, \ldots, s_N\}$
2. M: Number of observation symbols in alphabet $V = \{v_1, \ldots, v_M\}$
3. State transition probability $A = \{a_{ij}\}, a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i)$
4. Observation probabilities $B = \{b_j(m)\}, b_j(m) = P(O_t = v_m \mid q_t = s_j)$
5. Initial state probabilities $\pi = \{\pi_i\}, \pi_i = P(q_1 = s_j)$

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

## Three Basic Problems

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

(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 $O$: $Q^*=\arg\max_Q P(Q|O,\lambda)$

(3) Given training et of sequence $X=\{O^k\}$, find $\lambda^*=\arg\max P(X|\lambda)$

# HMMs – Problem 1. Evaluation

Given an observation $O=(O_1, O_2, \ldots O_T)$ and a state sequence $Q$, the probability of observing $O$ given $Q$ is

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

But we don't know $Q$

The prior probability of state sequence is $P(O \mid \lambda) = P(q_1) \prod_{t=2}^{T} P(q_t \mid q_{t-1}) = \pi_{q_1} a_{q_1 q_2} \cdots a_{q_{T-1} q_T}$

Joint probability

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

We can compute $P(O \mid \lambda) = \sum_{Q} P(O, Q \mid \lambda)$

But this summation is impractical directly, because there are too many possible $Q$ ($|Q| = N^T$)

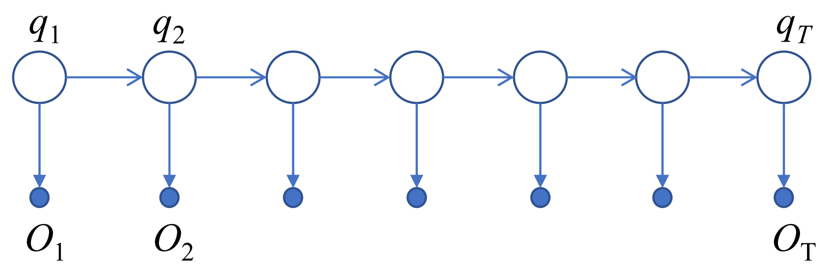
# HMMs – Problem 1. Evaluation

But there is an efficient procedure to calculate $P(O|\lambda)$ called the forward-backward procedure (essentially – dynamic programming)

This exploits the Markov structure of the distribution

Divide the sequence into parts

($1$ to $t$) &($t$+1 to $T$)




Forward variable $\alpha_t(i)$ is probability of observing the partial sequence and being in state $S_t$ at time $t$, (given the model $\lambda$): $\alpha_t(i) = P(O_1,\dots,O_t, q_t = s_i \mid \lambda)$

This can be computed recursively

Initialization: $\alpha_1(i) = P(O_1, q_1 = s_i \mid \lambda)$

$$= P(O_1 \mid q_1 = s_i, \lambda) P(q_1 = s_i \mid \lambda)$$

$$= \prod_i b_i(O_1)$$

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

# HMMs – Problem 1. Evaluation

Intuition: $\alpha_t(i)$ explains first $t$ observations and ends in state $s_i$

× probability $a_{ij}$ to get to state $s_j$ at $t+1$

× probability of generating ($t+1$)th observation $b_j(O_t+1)$

Then sum over all possible states $s_i$ at time $t$

$$P(O \mid \lambda) = \sum_{i=1}^{N} P(O, q_T = s_i \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Computing $\alpha_t(i)$ is $O(N^2T)$

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

An alternative algorithm (which we need later) is backward variable $\beta_t(i) \equiv P(O_{t+1}, \ldots O_T \mid q_t = s_i, \lambda)$

Finalize recursion: $\beta_T(i) = 1$

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

JOHNS HOPKINS UNIVERSITY

# HMMs – Problem 2. Finding the state sequence

Again, exploit the linear structure

Greedy Define $\delta_t(i)$ in probability of state $s_i$ at time t given $O$ and $\lambda$

$$\delta_t(i) = P(q_t = s_t \mid O, \lambda) = \frac{P(O \mid q_t = s_i, \lambda) P(q_t = s_i \mid \lambda)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} a_t(j)\beta_t(j)}$$

Forward variable $\alpha_t(i)$ 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 $a_{ij} = 0$

# HMMs – Viterbi Algorithm (Dynamic Programming)

Define $\delta_t(i)$ is the probability of the highest probability path that accounts for all the first t observations and ends in $s_i$

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

Calculate recursively

1. Initialize $s_1(i) = \pi_i b_i(O_1), \psi_1(i) = 0$
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 $p^* = \max_i s_T(i)$

   $q_T^* = \arg\max_i s_T(i)$
4. Path (state sequence) backtracking: $q_T^* = \psi_{t+1}(q_{t+1}^*), t = T-1, T-2, \ldots, 1$

Intuition

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

Same complexity $O(N^2T)$

# HMMs – Baum-Welch algorithm (EM)

At each iteration,

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

M-step Recalculate $\lambda$ given $\zeta_t(i,j)$ & $\gamma_t(i)$

Alternate the two steps until convergence

Indicator variables $Z_i^t = \begin{cases} 1, \text{if } q_t = s_i \\ 0, \text{otherwise} \end{cases}$ 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 $E[Z_i^t] = \gamma_t(i)$

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

In M-step, count the expected number of transitions from $s_i$ to $s_j$ $\left(\sum_t \zeta_t(i,j)\right)$

and total number of transitions from $s_i$ $\left(\sum_t \gamma_t(i)\right)$

JOHNS HOPKINS UNIVERSITY

# HMMs – Baum-Welch algorithm (EM)

This gives transition probability from $s_i$ to $s_j$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \zeta_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad \hat{b}_j = \frac{\sum_{t=1}^{T} \gamma_t(j) I(O_t = v_m)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Soft counts instead of real counts

For multiple observation sequences:

$$X = \{O^k : k = 1, \ldots, K\}$$

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

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

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

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

## HMMs -- Recapulation

We have given algorithm to solve the three problems:

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

(2) Compute $Q^* = \arg\max P(Q \mid O, \lambda)$

(3) Compute $\lambda^* = \arg\max P(X \mid \lambda)$

$P(O|\lambda)$ 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 $P(O \mid \lambda_1) > P(O \mid \lambda_2)$

model 2, otherwise

I.E. detect which model generates the sequences

This for multiple models with training data for each

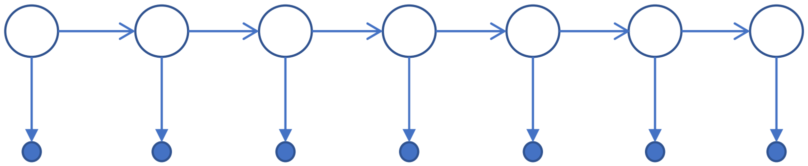
$$\lambda_1^*, \ldots, \lambda_w^* = \arg\max_X P(X^1 \mid \lambda) P(X^2 \mid \lambda) \ldots P(X^w \mid \lambda)$$

Use this to build speech recognition system

Further extensions of HMMs are described in the book

The basic idea is to exploit the one-dimensional structure of the model

Enables dynamic programming to perform rapid computation



EM algorithm for learning the model parameters

Multiple models – model selection