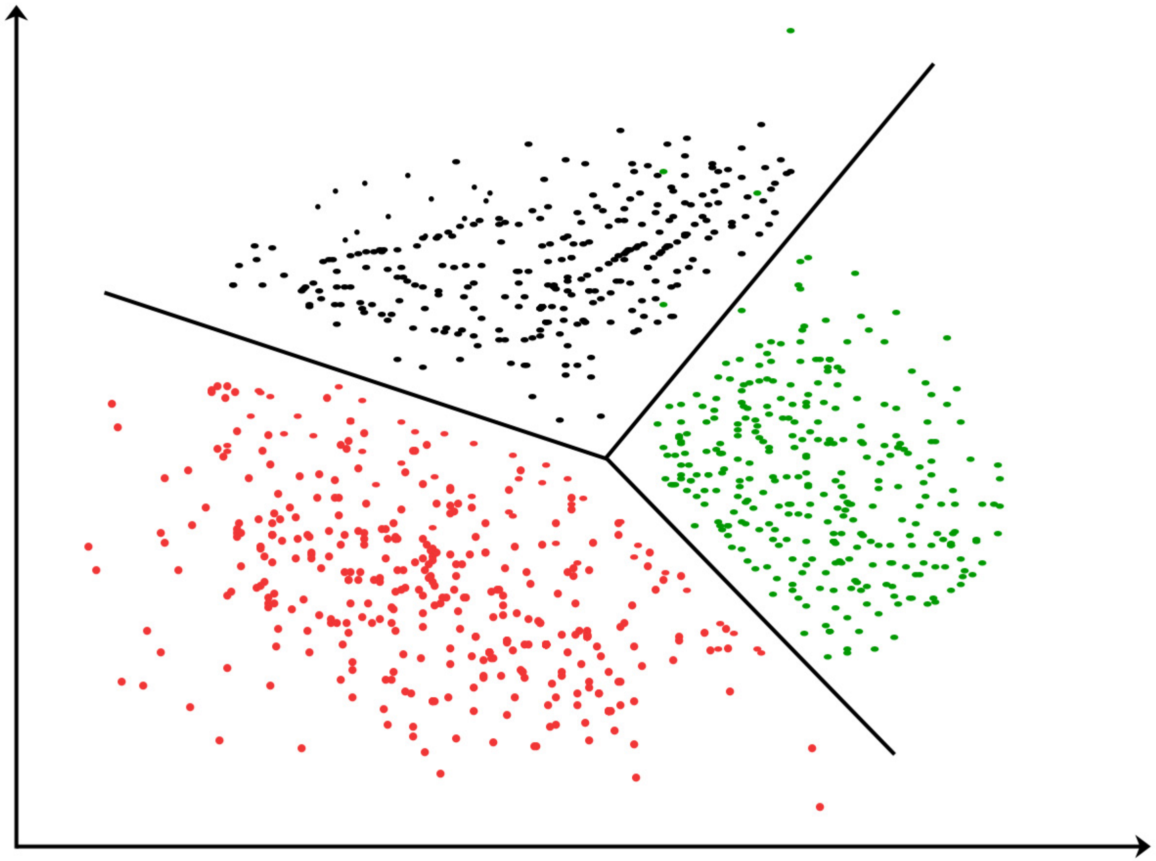
Clustering time series

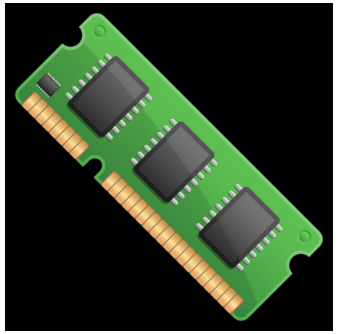
Jaehyoung Hong

Time-series clustering becomes important recently

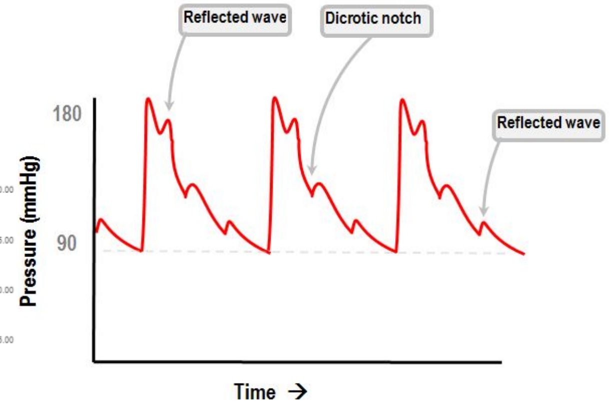


Clustering : Similarity & Difference

Memory



Processor



Time-series clustering needs some important characteristics

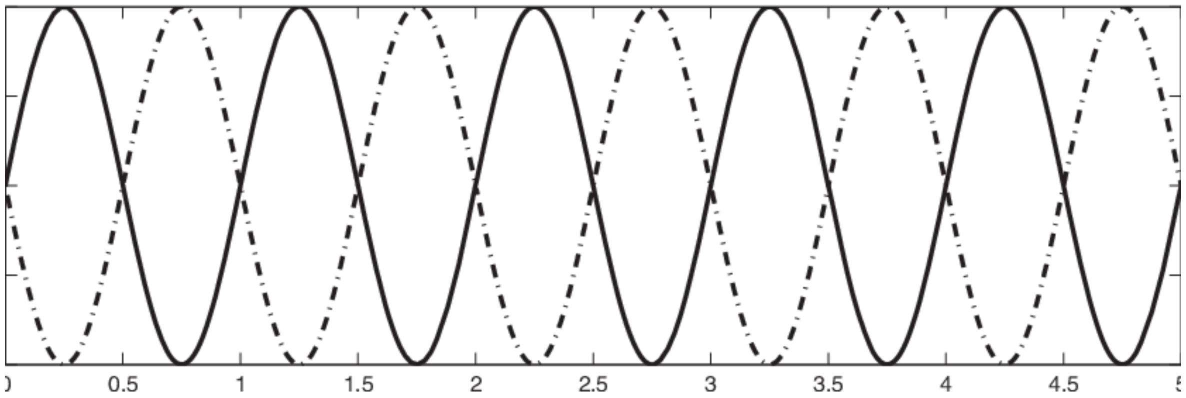
Very large data

- ECG (heart late) : 1 (GB/hour)
- Typical weblog : 5 (GB/week)
- Space shuttle database : 200 (GB/day)

Raw data comparison is inefficient

Make model / Transform to statistic

Anti-phase



Carefully chosen distance is needed

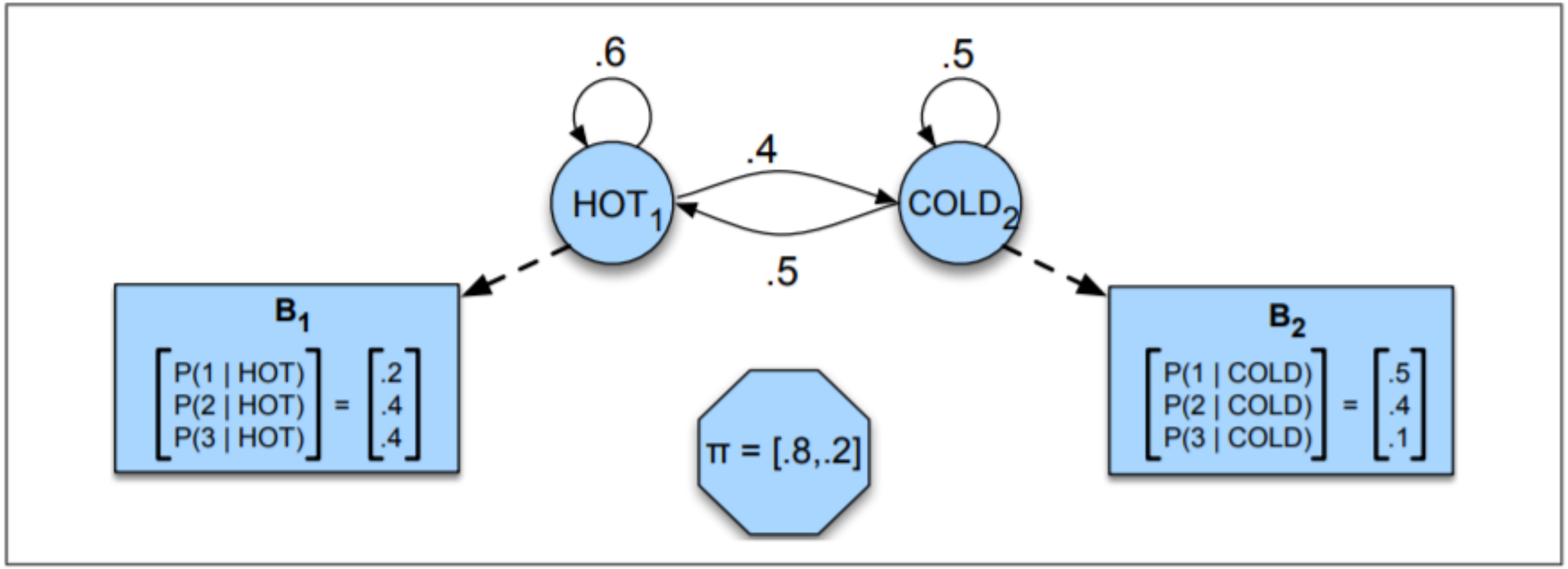
Outline of clustering with hidden markov model (HMM)

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
P(T \mid \lambda_i) \arrow[r, leftrightarrow, "\text{Distance}", "\text{(well-defined)}"'] & P(T \mid \lambda_j) \\
T_i \arrow[u, "\text{HMM}"] \arrow[r, leftrightarrow, "\text{Distance}", "\text{(ill-defined)}"'] & T_j \arrow[u, "\text{HMM}"']
\end{tikzcd}
$$

- $P(T|\lambda_i)$: Probability of observe time-series T when hidden markov model has parameter $\lambda_i = (A_i, B_i)$
- A_i : Matrix of transition probability
- B_i : Matrix of emission probability

Popular clustering method
(Partition around medoids)

Hidden markov model allows us to talk about *both observed & hidden* events



Number of ice cream (1~3; Observed) → Weather (H or C; Hidden)

HMM based on the Markov chain

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Markov Assumption: $P(q_i = a | q_1 ... q_{i-1}) = P(q_i = a | q_{i-1})$

Output Independence: $P(o_i | q_1 \ldots q_i, \ldots, q_T, o_1, \ldots, o_i, \ldots, o_T) = P(o_i | q_i)$

Three fundamental problems of HMM is key point of clustering idea

Problem 1 (Likelihood):	Given an HMM $\lambda = (A,B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Problem 2 (Decoding):	Given an observation sequence O and an HMM $\lambda = (A,B)$, discover the best hidden state sequence Q.
Problem 3 (Learning):	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

<Forward Algorithm>

<Viterbi Algorithm>

<Forward-Backward Algorithm>

$$
\begin{tikzcd}
P(T \mid \lambda_i) \arrow[rr, leftrightarrow, "\text{Distance}", "\text{(well-defined)}"'] & & P(T \mid \lambda_j) \\
T_i \arrow[u, "\text{HMM}"] \arrow[rr, leftrightarrow, "\text{Distance}", "\text{(ill-defined)}"'] & & T_j \arrow[u, "\text{HMM}"']
\end{tikzcd}
$$

We can compute the likelihood of a particular observation by using the forward algorithm

Computing Likelihood: Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$.

$$P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i)$$

$$P(3\ 1\ 3|\text{hot hot cold}) = P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})$$

(Fully determined by emission probability)

$$P(O,Q) = P(O|Q) \times P(Q) = \prod_{i=1}^{T} P(o_i|q_i) \times \prod_{i=1}^{T} P(q_i|q_{i-1})$$

$$\begin{aligned} P(3\ 1\ 3, \text{hot hot cold}) = {} & P(\text{hot}|\text{start}) \times P(\text{hot}|\text{hot}) \times P(\text{cold}|\text{hot}) \\ & \times P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold}) \end{aligned}$$

(Determined by transition and emission probability)

We can compute the likelihood of a particular observation by using the forward algorithm

$$P(O) = \sum_{Q} P(O,Q) = \sum_{Q} P(O|Q)P(Q)$$

$$\Rightarrow \quad P(3\ 1\ 3) = P(3\ 1\ 3, \text{cold cold cold}) + P(3\ 1\ 3, \text{cold cold hot}) + P(3\ 1\ 3, \text{hot hot cold}) + \ldots$$

- Problem : N^T possible hidden sequence → Too large

- Forward algorithm : $O(N^2T)$

We can compute the likelihood of a particular observation by using the forward algorithm

$\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j|\lambda)$: probability of being in state j after seeing the first t observations

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

$\alpha_{t-1}(i)$	the **previous forward path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

We can compute the likelihood of a particular observation by using the forward algorithm

function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$forward[s,1] \leftarrow \pi_s * b_s(o_1)$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**

$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$

$$forwardprob \leftarrow \sum_{s=1}^{N} forward[s,T] \quad \text{; termination step}$$

return *forwardprob*

1. Initialization:

$$\alpha_1(j) = \pi_j b_j(o_1) \;\; 1 \leq j \leq N$$

2. Recursion:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

We can find most probable sequence of hidden states by using Viterbi algorithm

Decoding: Given as input an HMM $\lambda = (A,B)$ and a sequence of observations $O = o_1, o_2, ..., o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \ldots q_T$.

- Naïve method : For all possible hidden sequence, compute likelihood

$$v_t(j) = \max_{q_1,\ldots,q_{t-1}} P(q_1 \ldots q_{t-1}, o_1, o_2 \ldots o_t, q_t = j | \lambda)$$

$$\Rightarrow \quad v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

We can find most probable sequence of hidden states by using Viterbi algorithm

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
  $viterbi[s,1] \leftarrow \pi_s \, * \, b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} \; viterbi[s',t-1] \, * \, a_{s',s} \, * \, b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} \; viterbi[s',t-1] \, * \, a_{s',s} \, * \, b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} \; viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} \; viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer
return *bestpath*, *bestpathprob*

Finally, we can give a formal definition of the Viterbi recursion as follows:

1. **Initialization:**

$$v_1(j) = \pi_j b_j(o_1) \qquad 1 \le j \le N$$
$$bt_1(j) = 0 \qquad 1 \le j \le N$$

2. **Recursion**

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$
$$bt_t(j) = \operatorname{argmax}_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$

3. **Termination:**

$$\text{The best score:} \quad P* = \max_{i=1}^{N} v_T(i)$$
$$\text{The start of backtrace:} \quad q_T* = \operatorname{argmax}_{i=1}^{N} v_T(i)$$

We can learn the parameters of HMM by using forward-backward algorithm

Learning: Given an observation sequence O and the set of possible states in the HMM, learn the HMM parameters A and B.

3	3	2	1	1	2	1	2	3
hot	hot	cold	cold	cold	cold	cold	hot	hot

$$\pi_h = 1/3 \quad \pi_c = 2/3$$

$$P(1|hot) = 0/4 = 0 \quad p(1|cold) = 3/5 = .6$$

$$p(hot|hot) = 2/3 \quad p(cold|hot) = 1/3 \quad P(2|hot) = 1/4 = .25 \quad p(2|cold = 2/5 = .4$$

$$p(cold|cold) = 2/3 \quad p(hot|cold) = 1/3 \quad P(3|hot) = 3/4 = .75 \quad p(3|cold) = 0$$

<A : transition prob>

<B : emission prob>

We can learn the parameters of HMM by using forward-backward algorithm

$\beta_t(i) = P(o_{t+1}, o_{t+2} \ldots o_T | q_t = i, \lambda)$: backward probability

1. **Initialization:**

$$\beta_T(i) = 1, \quad 1 \le i \le N$$

2. **Recursion**

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j), \quad 1 \le i \le N, 1 \le t < T$$

3. **Termination:**

$$P(O|\lambda) = \sum_{j=1}^{N} \pi_j\, b_j(o_1)\, \beta_1(j)$$

We can learn the parameters of HMM by using forward-backward algorithm

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j | O, \lambda)$$ (summation of it gives numerator)

$$\text{not-quite-}\xi_t(i,j) = P(q_t = i, q_{t+1} = j, O | \lambda)$$

$$\text{not-quite-}\xi_t(i,j) = \alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

$$P(O|\lambda) = \sum_{j=1}^{N} \alpha_t(j)\beta_t(j)$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

We can learn the parameters of HMM by using forward-backward algorithm

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{j=1}^{N} \alpha_t(j) \beta_t(j)}$$

$$\Rightarrow \quad \hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

We can learn the parameters of HMM by using forward-backward algorithm

<recomputing observation probability>

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

$$\gamma_t(j) = P(q_t = j|O,\lambda)$$ (summation of it gives numerator)

$$\gamma_t(j) = \frac{P(q_t = j, O|\lambda)}{P(O|\lambda)}$$

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)}$$

We can learn the parameters of HMM by using forward-backward algorithm

<recomputing observation probability>

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

$$\Rightarrow \quad \hat{b}_j(v_k) = \frac{\sum_{t=1\ s.t.O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

We can learn the parameters of HMM by using forward-backward algorithm

function FORWARD-BACKWARD(*observations* of len T, *output vocabulary* V, *hidden state set* Q) **returns** *HMM=(A,B)*

initialize A and B
iterate until convergence
E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \ \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \ \forall\, t,\ i, \text{ and } j$$

M-step

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

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 \text{ s.t. } O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return A, B

Define distance for clustering using KL-divergence

- Learning : find $\lambda = (A, B)$
- Likelihood : find $P(T|\lambda)$

KL-divergence

$$D_{KL}(P(T \mid \lambda_i); P(T \mid \lambda_j)) \equiv \int dT\, P(T \mid \lambda_i) \log \frac{P(T \mid \lambda_i)}{P(T \mid \lambda_j)}$$

Monte-Carlo

$$D_{KL}(P(T \mid \lambda_i); P(T \mid \lambda_j)) \approx \frac{1}{n} \sum_{\alpha=1}^{N} \log \frac{P(T_\alpha \mid \lambda_i)}{P(T_\alpha \mid \lambda_j)}$$

Need large N

One point approximation

$$D_{KL}(P(T \mid \lambda_i); P(T \mid \lambda_j)) \approx \log \frac{P(T_i \mid \lambda_i)}{P(T_i \mid \lambda_j)}$$

Too extreme

Define distance for clustering using KL-divergence

- The observed data set is sufficiently representative of the universe of possible trajectories

$$P(T \mid \lambda) \;\rightarrow\; \tilde{P}_\lambda \equiv \frac{1}{Z_\lambda} \{P(T_1 \mid \lambda), P(T_2 \mid \lambda), \ldots, P(T_N \mid \lambda)\} \equiv \left\{\tilde{P}(T_1 \mid \lambda), \tilde{P}(T_2 \mid \lambda), \ldots, \tilde{P}(T_N \mid \lambda)\right\}$$

Where $Z_\lambda = \sum_{i=1}^{N} P(T_i|\lambda)$

$$D_{KL}(\lambda_i; \lambda_j) \equiv D_{KL}(\tilde{P}_{\lambda_i}; \tilde{P}_{\lambda_j}) = \sum_{i=1}^{N} \tilde{P}(T_i \mid \lambda_i) \log \frac{\tilde{P}(T_i \mid \lambda_i)}{\tilde{P}(T_i \mid \lambda_j)}$$

$$D_{ij} \equiv D(T_i; T_j) \equiv \frac{1}{2} \left(D_{KL}(\lambda_i; \lambda_j) + D_{KL}(\lambda_j; \lambda_i)\right)$$

Partitioning around medoids (PAM) is clustering method similar to k-means clustering

- K-means clustering : Choose K-center → Assign data point to nearest center
  → Recompute centers as mean of data points in each cluster

- Partitioning around medoids (PAM) : Medoids must be the data points in cluster

Application to 1225 disease data shows 4 clusters

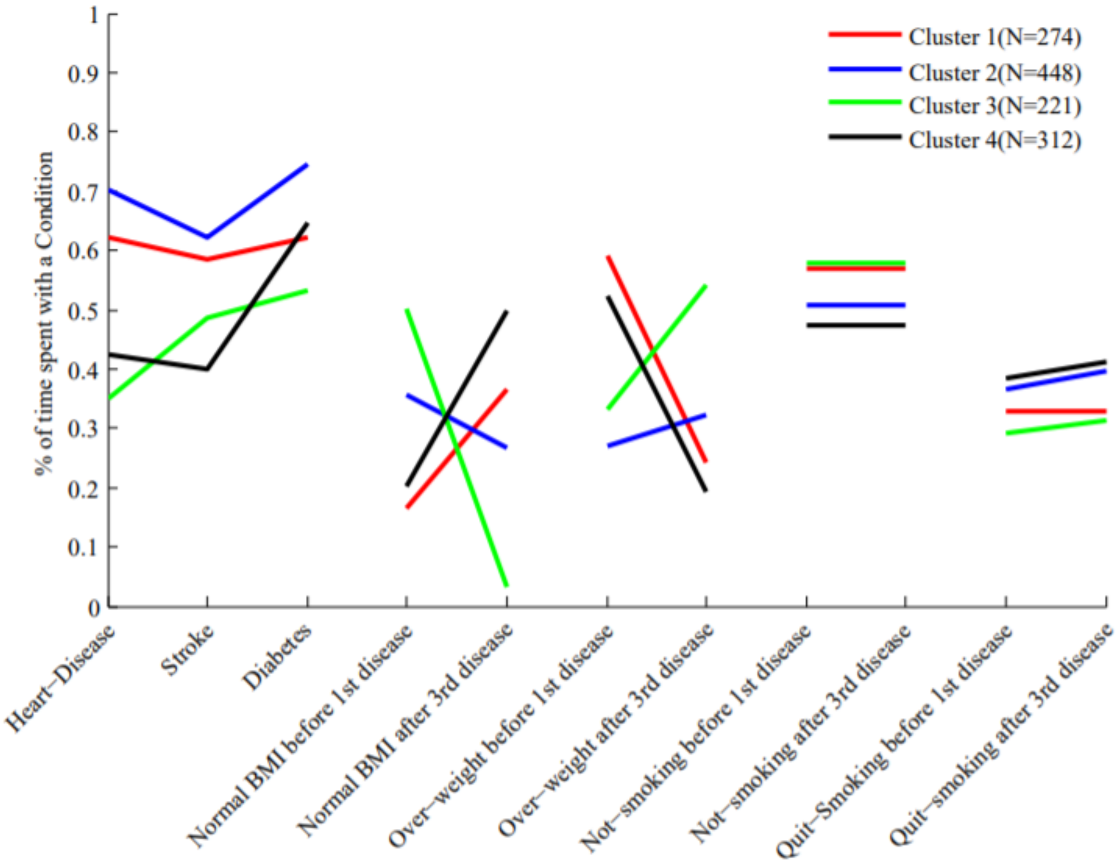
- 1225 patient = time-series / length = 18 / all patients develops three chronic condition / using obesity and smoking as covariates of hidden sequence

	Disease onset	Normal BMI	Overweight	Quit smoking
Cluster 1	Heart disease, stroke and diabetes almost simultaneously	Mostly overweight/obese before 1st disease	Some weight loss after 3rd disease	No change in smoking behaviour after 3rd disease
Cluster 2	Diabetes, heart disease and then stroke	Significantly overweight/obese before 1st disease	Some weight gain after 3rd disease	Mild increase in quitting smoking after 3rd disease
Cluster 3	Diabetes, stroke and then heart disease	Half time normal BMI before 1st disease	Significant weight gain after 3rd disease	Mild increase in quitting after 3rd disease
Cluster 4	Diabetes and then heart disease and stroke	Mostly overweight/obese before 1st disease	Significant weight loss after 3rd disease	Mild increase in quitting after 3rd disease

Number of clusters are chosen by DB / Dunn index

Application to 1225 disease data shows 4 clusters

- 1225 patient = time-series / length = 18 / all patients develops three chronic condition / using obesity and smoking as covariates of hidden sequence



Number of Hidden states	Number of clusters	Correlation
2 Hidden states	$K = 2$	0.74
3 Hidden states	$\boldsymbol{K = 4}$	**0.41**
4 Hidden states	$K = 3$	0.47