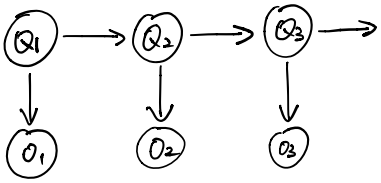


is also a directed graphical model.

Settings : * Q_t : hidden units random variable, q_t : observed hidden unit

O_t : observation random variable, o_t : observed observation.

* N : # of possible states, denoted by $S_1, S_2, \cdots, S_N$
* M : # of possible observation, denoted by $V_1, V_2, \cdots, V_M$
* $\underset{\sim}{\pi} = (\pi_1, \pi_2, \cdots, \pi_N)$ where $\pi_i = P(Q_1 = S_i)$ initial state.
* $\underset{=}{A}$: transition matrix, where $A_{ij} = P(Q_t = S_j \mid Q_{t-1} = S_i)$.
* $\underset{=}{B}$: observation probability matrix,

where $B_{jk} \equiv b_j(k) = P(O_t = V_k \mid Q_t = S_j)$.

we put $\lambda = (\underset{\sim}{\pi}, \underset{=}{A}, \underset{=}{B})$ as parameter of HMM.

* Markov Property :

$$P(Q_t \mid Q_{1:t-1}, O_{1:t-1}) = P(Q_t \mid Q_{t-1}) \quad \forall t$$

* Joint Probability :

$$P(q_{1:T}, o_{1:T}) = P(q_1) \cdot P(o_1 \mid q_1) \cdot \prod_{t=2}^{T} P(q_t \mid q_{t-1}) \cdot P(o_t \mid q_t)$$

$$= \pi_1 \cdot B_{11} \cdot \prod_{t=1}^{T} A_{q_{t-1}, t} \cdot B_{tt}$$

We can sample a sequence from $\lambda = (\underset{\sim}{\pi}, \underset{=}{A}, \underset{=}{B})$.

Three Basic Problems in HMM Learning.

* Note N, M are fixed in all three basic problems

1. Evaluation Problem.

Given fixed parameter $\lambda = (\underset{\sim}{\pi}, \underset{=}{A}, \underset{=}{B})$, evaluate the probability of an observed sequence $O_{1:T}$

i.e. $P(O_{1:T}|\lambda)$.

Both λ and $O_{1:T}$ are fixed

Example * given λ, $O_{1:T}$ and $O'_{1:T}$, which sequence is more likely to happen under λ?

* given $O_{1:T}$ and λ, λ', which parameter is a better fit to $O_{1:T}$

Solve by Dynamic Programming

2. Decoding problem:

Given fixed λ and fixed $O_{1:T}$, what are the best hidden states sequence $(q_{1:T})$ corresponding to these observations?

Solve by Dynamic Programming.

3. Learning Parameters:

Given N, M and $O_{1:T}$, what is best parameters $\lambda = (\underset{\sim}{\pi}, \underset{=}{A}, \underset{=}{B})$?

Find a λ that maximises the likelihood function $P(O_{1:T}|\lambda)$.

(if T is large enough, one training sample is enough to learn λ)

Solve by EM

1: Evaluation Problem:

Given $\mathcal{N}, M, \lambda, O_{1:T}$. Compute $P(O_{1:T}|\lambda)$

Let Ω be the set of all possible sequence of hidden states.

From the law of total probability:

$$P(O_{1:T}|\lambda) = \sum_{Q_{1:T}\in\Omega} P(O_{1:T}, q_{1:T}|\lambda)$$

$$= \sum_{Q_{1:T}\in\Omega} P(O_{1:T}|q_{1:T},\lambda)\cdot P(q_{1:T}|\lambda). \quad \cdots \text{(pentagram)}$$

where $P(O_{1:T}|q_{1:T},\lambda) = \prod_{t=1}^{T} P(o_t|q_t,\lambda) = \prod_{t=1}^{T} B_{tt}$

$$P(q_{1:T}|\lambda) = \pi_1 \cdot \prod_{t=2}^{T} A_{t-1,t}$$

Note we need to generate $|\Omega|$ sequences of hidden states and observations. Each state can take values in $\mathcal{N}$ possible states and there are T time stamps. $|\Omega| = \mathcal{N}^T$

This makes (pentagram) too complex to solve.

Solution: Dynamic Programming

Target: find $P(O_{1:T}|\lambda)$.

Note O_t only depends on Q_t.

Q_t only depends on Q_{t-1}

$\Rightarrow P(O_{1:T}|\lambda) \nearrow P(O_{1:T-1}|\lambda)$, $\searrow P(O_T|\lambda)$.

Then combine them by enumerating all possible states of Q_{T-1} and Q_T complexity: $\mathcal{N}^2$

Similarly, $P(O_{1:T-1}|\lambda) \nearrow P(O_{1:T-2}|\lambda)$, $\searrow P(O_{T-1}|\lambda)$.

This then becomes a Dynamic Programming Setting.

1.1 Forward variable and algorithm

$$P(O_{1:T}|\lambda) = \sum_{i=1}^{N} P(O_{1:T}, Q_T = S_i|\lambda).$$

$$= \sum_{i=1}^{N} P(O_{1:T-1}, Q_T = S_i|\lambda)\, B_{iT}$$

then: $P(O_{1:T-1}, Q_T = S_i|\lambda)$

$$= \sum_{j=1}^{N} P(O_{1:T-1}, Q_T = S_i, Q_{T-1} = S_j|\lambda).$$

$$= \sum_{j=1}^{N} P(O_{1:T-1}, Q_{T-1} = S_j|\lambda) \cdot P(Q_T = S_i|O_{1:T-1}, Q_{T-1} = S_j|\lambda)$$

$$= \sum_{j=1}^{N} P(O_{1:T-1}, Q_{T-1} = S_j|\lambda) \cdot P(Q_T = S_i|Q_{T-1} = S_j, \lambda)$$

$$= \sum_{j=1}^{N} P(O_{1:T-1}, Q_{T-1} = S_j|\lambda) \cdot A_{ji}$$

Note that:

$$P(O_{1:T-1}|\lambda) = \sum_{j=1}^{N} P(O_{1:T-1}, Q_{T-1} = j|\lambda)$$

if we can compute $P(O_{1:T-1}, Q_{T-1} = j|\lambda)$ $\forall j \in \{1, \cdots, N\}$.

we can evaluate $P(O_{1:T-1}|\lambda)$ hence $P(O_{1:T}|\lambda)$ for time T.

The FORWARD ALGORITHM:

$$\alpha_t(i) = P(O_{1:t}, Q_t = S_i|\lambda)$$

the recursion:

$$\alpha_{t+1}(i) = \left(\sum_{j=1}^{N} \alpha_t(j) \cdot A_{ji}\right) \cdot B_{i,t+1}$$

Pseudo-Code:

1. Initialise: $\alpha_1(i) = \pi_i B_{i1}$ $\forall$ $i \in \{1, \cdots, N\}$.
2. Forward recursion: $\forall$ $t \in \{1, 2, \cdots, T-1\}$, $\forall$ $i \in \{1, \cdots, N\}$.

$$\alpha_{t+1}(i) = \left(\sum_{j=1}^{N} \alpha_t(j) \cdot A_{ji}\right) \cdot B_{i,t+1}$$

3. Output: $P(O_{1:T}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

Complexity here is $O(TN^2)$, much better than N^T

1.2 Backward variable and algorithm

When at time t, look back:

$\alpha_i(t)$ is the probability that $Q_t = S_i$ and $O_{1:t}$ are observed.

What about the future? Still have $O_{t+1:T}$.

Define: $\beta_t(i) = P(O_{t+1:T} | Q_t = S_i, \lambda)$

$\beta_t(i)$ is the probability of observing future output sequence $O_{t+1:T}$ if $Q_t = S_i$ at time t

Recursion relationship:

$$\beta_t(i) = \sum_{j=1}^{N} A_{ij} B_{j,t+1} \beta_{t+1}(j).$$

This must be initialized at $t = T$ and move backward.

and $\beta_T(i)$ is the probability of observing NOTHING after time T give $Q_T = S_i$. Hence $\beta_T(i) = 1$.

Finally $P(O_{1:T} | \lambda) = \sum_{i=1}^{N} \pi_i B_{i1} \beta_1(i)$.

Pseudo-Code:

1. Initialization: $\beta_T(i) = 1$, $\forall i \in \{1, \cdots, N\}$.
2. Backward Recursion: $\forall t \in \{T-1, T-2, \cdots, 2, 1\}$ and $\forall i \in \{1, \cdots, N\}$

$$\beta_t(i) = \sum_{j=1}^{N} A_{ij} B_{j,t+1} \beta_{t+1}(j).$$

3. Output

$$P(O_{1:T} | \lambda) = \sum_{i=1}^{N} \pi_i B_{i1} \beta_1(i).$$

Note: Forward and Backward algorithms are calculating the same thing: $P(O_{1:T} | \lambda)$. so they must match.

2. Decoding Problem: Find the Best Hidden States.

Idea: find the maximum likelihood state for each time step independently.

i.e. $\forall\ t \in \{1, \cdots, T\}$, find $P(Q_t = S_i \mid O_{1:T}, \lambda)$.

Define: $\gamma_t(i) = P(Q_t = S_i \mid O_{1:T}, \lambda)$

Set: $q_t = \underset{i \in \{1,\cdots,N\}}{\operatorname{argmax}}\ \gamma_t(i)$

Note: $\gamma_t(i)$ is the probability of Q_t being S_i when we observed the complete sequence $O_{1:T}$

$$\gamma_t(i) = P(Q_t = S_i \mid O_{1:T}, \lambda) = \frac{P(Q_t = S_i, O_{1:T} \mid \lambda)}{P(O_{1:T} \mid \lambda)}$$

→ Byproduct: $P(O_{1:T} \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i) \cdot \beta_t(i)$. $\forall\ t$.

$$= \frac{P(O_{1:t}, O_{t+1:T} \mid Q_t = S_i, \lambda) \cdot P(Q_t = S_i \mid \lambda)}{\sum_{j=1}^{N} P(Q_t = S_j, O_{1:T} \mid \lambda)}$$

$$= \frac{P(O_{1:t} \mid Q_t = S_i, \lambda) \cdot P(O_{t+1:T} \mid Q = S_i, \lambda) \cdot P(Q_t = S_i \mid \lambda)}{\sum_{j=1}^{N} P(Q_t = S_j, O_{1:T} \mid \lambda)}$$

$$= \frac{P(O_{t+1:T} \mid Q_t = S_i, \lambda) \cdot P(Q_t = S_i, O_{1:t} \mid \lambda)}{\sum_{j=1}^{N} P(Q_t = S_j, O_{1:T} \mid \lambda)}$$

($P(O_{t+1:T} \mid Q_t = S_i, \lambda) = \beta_t(i)$, $P(Q_t = S_i, O_{1:t} \mid \lambda) = \alpha_t(i)$)

$$= \frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j) \cdot \beta_t(j)}$$

To compute γ, we first set $\gamma_t(i) = \alpha_t(i) \cdot \beta_t(i)$. then l_1 normalise $\gamma\ \forall\ t$. Note we can find q_t by finding the maximum element in unnormalised γ values. Since normalisation constant $P(O_{1:T} \mid \lambda)$ will not change the index of largest γ value at time t.

Practical Problem: This finds the best state $\forall\ t \in \{1, \cdots, T\}$ independently. which may lead to the state sequences that should not appear.

e.g. we set $S_1, S_3 = 0$. but if we use γ to decode, we will find solutions for $S_1' \neq 0$ and $S_3' \neq 0$. Hence this has serious problems

2.1 Viterbi Decoding

To eliminate the previous problem, we use jointly decode the entire optimal state sequence:

$$q_{1:t} = \arg\max_{Q_{1:T}} P(Q_{1:T} \mid O_{1:T}, \lambda) = \arg\max_{Q_{1:T}} P(Q_{1:T}, O_{1:T} \mid \lambda).$$

Dynamic Programing Solution:

The recursive relationship:

Given $O_{1:t}$, it is useful to know the optimal paths for N subproblems:

$$\max_{Q_{1:t-1}} P(Q_{1:t-1}, O_{1:t}, Q_t = S_i \mid \lambda), \quad i \in \{1, \cdots, N\}.$$

We can solve at least the following two problems by using the answer to these N problems.

1. The optimal hidden state sequence till time t.

   if the i^* subproblem has the largest probability (likelihood), then the optimal parameter of this subproblem $q_{1:t-1}$ plus $q_t = i^*$ is the optimal hidden state sequence for $O_{1:t}$.

2. The optimal $q_{1:t+1}$ for $O_{1:t+1}$ can be divided into: $q_{1:t-1}$, q_t, q_{t+1}. If we know $q_t = i^*$, the $q_{1:t-1}$ part can be found by the i^*-th subproblem. By Markov Assumption, we only need to consider N possible transitions $S_{q_t} \to S_{q_{t+1}}$ to decide which state is optimal for Q_{t+1}.

Define: $\delta_t(i) = \max_{Q_{1:t-1}} P(Q_{1:t-1}, O_{1:t}, Q_t = S_i \mid \lambda)$.

Recursive: $\delta_{t+1}(i) = \max_{j \in \{1, \cdots, N\}} \left(\delta_t(j) A_{ji} B_{i, t+1} \right)$. ---- ⊗

where $\delta_t(j)$ is the probability of the j-th subproblem.

A_{ji} transmits from S_j to S_i at time t and $t+1$ respectively.

$B_{i,t+1}$ is the probability of observing O_{t+1} when state is S_i.

Then $\delta_t(j) A_{ji} B_{i,t+1}$ is the probability of the optimal state sequence when we know $Q_t = S_j$ for an observation sequence $O_{1:t+1}$.

Forward Recursion: start with $t = 1$

After δ values are calculated, we find q_T by:

$$q_T = \underset{i \in \{1,\cdots,N\}}{\operatorname{argmax}} \delta_T(i)$$

From eqn ⊗, if we know the optimal state at time $t+1$ is q_{t+1}, we just need to find which j leads to the largest $\delta_t(j) A_{ji} B_{i,t+1}$. Then S_j is the optimal state for Q_t. That is, we need to record the optimal transitions from t to $t+1$.

We use $\psi_{t+1}(i)$ to denote the optimal state at time t if the optimal state is i at time $t+1$, hence:

$$\psi_{t+1}(i) = \underset{j \in \{1,\cdots,N\}}{\operatorname{argmax}} \left(\delta_t(j) A_{ji} B_{i,t+1}\right) = \underset{j \in \{1,\cdots,N\}}{\operatorname{argmax}} \left(\delta_t(j) A_{ji}\right)$$

?

Pseudo-Code: (Viterbi decoding)

1. Initialise: $\delta_1(i) = \pi_i B_{i1}$, $\psi_1(i) = 0$, $\forall\, i \in \{1,\cdots,N\}$.

2. Recursion: For $t = 2, 3, \cdots, T-1$, and $\forall\, i \in \{1,\cdots,N\}$

$$\delta_{t+1}(i) = \max_{j \in \{1,\cdots,N\}} \left(\delta_t(j) A_{ji} B_{i,t+1}\right).$$

$$\psi_{t+1}(i) = \max_{j \in \{1,\cdots,N\}} \left(\delta_t(j) A_{ji}\right).$$

3. Output: the optimal state q_T: $q_T = \underset{i \in \{1,\cdots,N\}}{\operatorname{argmax}} \delta_T(i)$.

and the rest of optimal path is determined by: for $t = T-1, T-2, \cdots, 2, 1$,

$$q_t = \psi_{t+1}(q_{t+1}).$$

Complexity $O(TN^2)$.

3. Learning Parameters in HMM.

Given N, M, $O_{1:T}$, learn $\lambda = (\pi, A, B)$

Classical Algo: Baum-Welch. (Maximum likelihood technique). (EM)

$$\lambda^* = \arg\max_{\lambda} P(O_{1:T}|\lambda).$$

Iterative algorithm:

1. Random initialize $\lambda^{(1)}$, compute $\ell^{(1)} = \log P(O_{1:T}|\lambda)$.
2. Update parameters to get $\lambda^{(2)}$.

⋮

Define new variable $\xi_t(i,j) = P(Q_t = S_i, Q_{t+1} = S_j | O_{1:T}, \lambda)$.

$\xi_t(i,j)$ is the expected proportion of transition from S_i (time t) to S_j (time $t+1$)

Then it is natural to use $\xi_t(i,j)$ to update the value for A_{ij} in $\lambda^{(r+1)}$

($\xi_t(i,j)$ computed based on $\lambda^{(r)}$)

$$\xi_t(i,j) \cdot P(O_{1:T}|\lambda) = P(Q_t = S_i, Q_{t+1} = S_j, O_{1:T}|\lambda)$$

Then we have $\xi_t(i,j) = \dfrac{\alpha_t(i) A_{ij} B_{j,t+1} \beta(j)}{P(O_{1:T}|\lambda)}$

Since $\xi_t(i,j)$ is a probability, we have $\sum_{i=1}^{N}\sum_{j=1}^{N} \xi_t(i,j) = 1$

Hence $\sum_{i=1}^{N}\sum_{j=1}^{N} \dfrac{\alpha_t(i) A_{ij} B_{j,t+1} \beta(j)}{P(O_{1:T}|\lambda)} = 1$

$$\Rightarrow P(O_{1:T}|\lambda) = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) A_{ij} B_{j,t+1} \beta(j). \quad \forall\, t \in \{1, 2, \cdots, T-1\}$$

We also get that: $\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$.

Recursive Relationship: $\xi_t(i,j) = \dfrac{\alpha_t(i) \cdot A_{ij} B_{j,t+1} \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) A_{ij} B_{j,t+1} \beta_{t+1}(j)}$. ... Responsibility in EM.

The parameters $\lambda = (\underline{\pi}, \underline{\underline{A}}, \underline{\underline{B}})$ can be updated using γ and ξ.

* Since $\gamma_1(i)$ is the expected proportion of $Q_1 = S_i$, we update π_i using $\gamma_1(i)$.

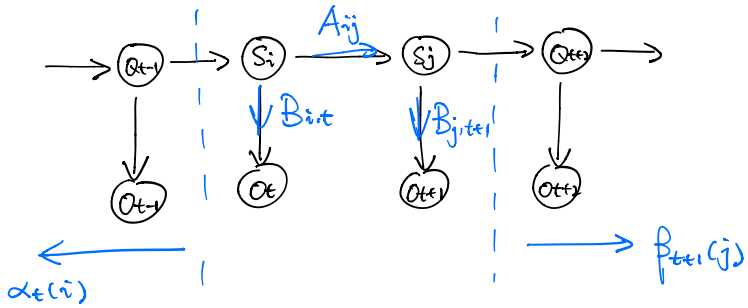
* $A_{ij} = \dfrac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$.

Interpretation:

- $\xi_t(i,j)$ is the expected probability of transition from S_i to S_j at t. Hence the expected number of transitions from S_i to S_j in the training sequence is $\sum_{t=1}^{T-1} \xi_t(i,j)$.
- $\sum_{t=1}^{T-1} \gamma_t(i)$ is the expected number of times that any hidden states being S_i

Hence A_{ij}, the transition probability from S_i to S_j, is the proportion of transitions $S_i \to S_j$ in all transitions starting from S_i.

* $B_{jk} = \dfrac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} [\![O_t = k]\!] \gamma_t(j)}$, where $[\![\cdot]\!]$ is the indicator function.



Summary :

	Definition	Recursion
α	$\alpha_t(i) = P(O_{1:T}, Q_t = S_i \mid \lambda)$	$\alpha_{t+1}(i) = \left(\sum_{j=1}^{N} \alpha_t(j) A_{ji}\right) B_{i,t+1}$
β	$\beta_t(i) = P(O_{t+1:T} \mid Q_t = S_i, \lambda)$	$\beta_t(i) = \sum_{j=1}^{N} A_{ij} B_{j,t+1} \beta_{t+1}(j)$
γ	$\gamma_t(i) = P(Q_t = S_i \mid O_{1:T}, \lambda)$	$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$
δ	$\delta_t(i) = \max_{Q_{1:t-1}} P(Q_{1:t-1}, O_{1:t}, Q_t = S_i \mid \lambda)$	$\delta_{t+1}(i) = \max_{j \in \{1,\dots,N\}} \left(\delta_t(j) A_{ji} B_{i,t+1}\right)$
ξ	$\xi_t(i,j) = P(Q_t = S_i, Q_{t+1} = S_j \mid O_{1:T}, \lambda)$	$\xi_t(i,j) = \frac{\alpha_t(i) \cdot A_{ij} B_{j,t+1} \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) A_{ij} B_{j,t+1} \beta_{t+1}(j)}$