Markov property

Up to now:

- Assumed statistical independence of data
- For $X = \{x_1, x_2, \dots, x_n\}$:

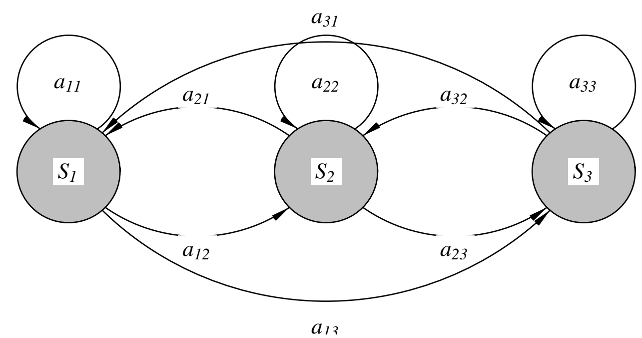
$$P(x_i | x_j) = P(x_j), \forall i \neq j .$$

Now, Markov assumption:

$$P(x_t | x_{t-1}, x_{t-2}, \dots, x_1) = P(x_t | x_{t-1})$$

Observable Markov models

- **Can directly observe state (nothing hidden)**
- N **states with probabilistic transitions** a_{ij}
- **Three-state example:**



Observable Markov model parameters

- q_t **= state of system at time** t
- **State-transition matrix:**

$$a_{ij} \equiv P(q_t = S_j | q_{t-1} = S_i), \; i, j \in \{1, \dots, N\}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix}$$

- **Initial-state probabilities:** $\pi_i = P(q_1 = S_i)$

Observable Markov models

- **Markov chain is completely defined by** $\{A, \pi\}$ **.**
- **Assume Markov chain is stationary (**$\{A, \pi\}$ **are fixed).**

Probability of observation sequence

Assume observation sequence O of length T:

$$O = \{q_1, q_2, \ldots, q_T\}$$

and a Markov model $\lambda = \{A, \pi\}$.

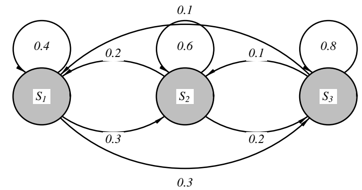
Computing $P(O|\lambda)$:

$$P(O|\lambda) = P(q_1) \times P(q_2|q_1) \times \ldots \times P(q_T|q_{T-1})$$

$$P(O|\lambda) = \pi_{q_1} \times a_{q_1 q_2} \times \ldots \times a_{(q_{T-1})q_T}$$

Example

$$A = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}, \pi = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.4 \end{bmatrix},$$



$$O = \{S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3\}$$

Probability $P(O|\lambda)$:

$$P(O|\lambda) = \pi_3 a_{33} a_{33} a_{31} a_{11} a_{13} a_{32} a_{23}$$

$$P(O|\lambda) = 0.4 \times 0.8 \times 0.8 \times 0.1 \times 0.4 \times 0.3 \times 0.1 \times 0.2$$

$$P(O|\lambda) \approx 6.144 \times 10^{-5}$$

Maximum-likelihood estimate

Given:

- Observation state sequence O

what are the maximum-likelihood estimates of $\{A, \pi\}$?

Maximum-likelihood estimate

Given:

- Observation state sequence O

what are the maximum-likelihood estimates of $\{A, \pi\}$?

Maximum-likelihood estimates:

$$a_{ij} = \frac{\text{number of transitions from state } S_i \text{ to state } S_j}{\text{number of transitions from state } S_i}$$

$$\pi_i = \begin{cases} 1, & \text{if } q_1 = S_i \\ 0, & \text{otherwise} \end{cases}$$

Example

$O = \{22333113333333333333333333331122222331213333333$
$33332132332223322222222222113312222222221112221113$
$33333\}$

Number of transitions from S_i to S_j:

$$\hat{A} = \begin{bmatrix} 7 & 4 & 5 \\ 5 & 27 & 4 \\ 4 & 4 & 40 \end{bmatrix} \Rightarrow \text{ normalize rows } \Rightarrow$$

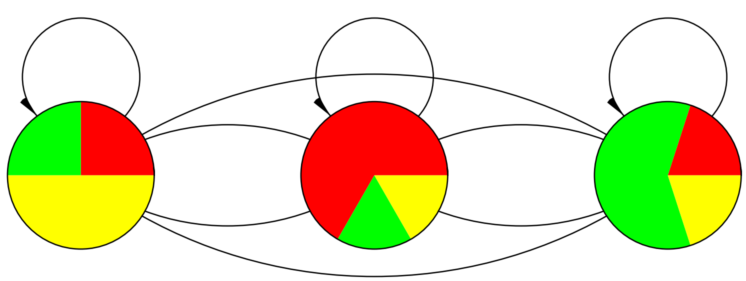
$$A^* \approx \begin{bmatrix} 0.438 & 0.250 & 0.313 \\ 0.139 & 0.750 & 0.111 \\ 0.085 & 0.085 & 0.830 \end{bmatrix}$$

Hidden Markov models (HMMs)

- **Underlying state no longer directly observable.**
- **Each state has probability distribution of observables associated with it.**
- **Two types:**
  - *Discrete output*
  - *Continuous output*

Discrete-output HMM example

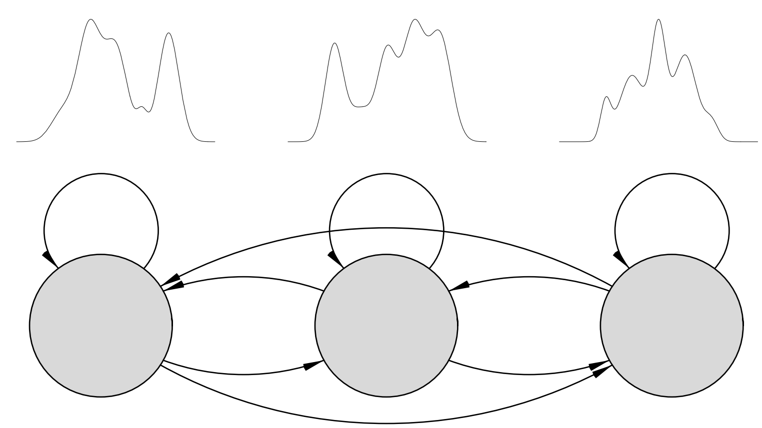
Hidden Markov model:



Sample observation sequence:

Continuous-output HMM

Hidden Markov model:



Sample observation sequence: $\mathbf{X} = \{\mathbf{x}_t\}, t = \{1, \ldots, T\}$.

Why discrete-output HMMs?

- **Much more computationally efficient**
- **Combine with VQ to model/classify real data**

Hidden Markov model applications

1. Speech recognition
2. Language modeling
3. Gesture recognition (e.g. sign language)
4. Hand-writing recognition
5. Facial-expression recognition (e.g. sign language)
6. Human skill modeling (e.g. surgical procedures)
7. Human control strategy analysis (e.g. driving)
8. Robot control (e.g. autonomous driving)
9. And others...

Discrete-output HMM parameters

Definitions:

- S_i = state i ; q_t = state of system at time t
- v_k = observable k ; O_t = observable at time t

HMM parameters:

- $N \times N$ state-transition matrix A :

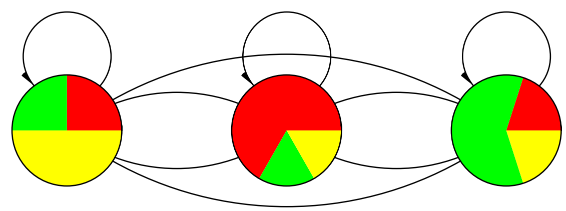
$$a_{ij} \equiv P(q_t = S_j | q_{t-1} = S_i) \text{ , } i, j \in \{1, \ldots, N\}$$

- $N \times L$ output probability distribution matrix B :

$$b_j(k) \equiv P(O_t = v_k | q_t = S_j) \text{ , } j \in \{1, \ldots, N\} \text{ , } k \in \{1, \ldots, k\}$$

Discrete-output HMM example

Hidden Markov model:



B **output probability matrix:**

$$B = \begin{bmatrix} 1/4 & 3/5 & 1/5 \\ 1/4 & 1/5 & 3/5 \\ 1/2 & 1/5 & 1/5 \end{bmatrix} \begin{matrix} \text{(red)} \\ \text{(green)} \\ \text{(yellow)} \end{matrix}$$

Hidden Markov models: 3 basic problems

Definitions:

- $O = \{O_t\}, t = \{1, \ldots, T\}$
- $\lambda = \{A, B, \pi\}$ = hidden Markov model

1. Evaluation: Compute $P(O|\lambda)$.

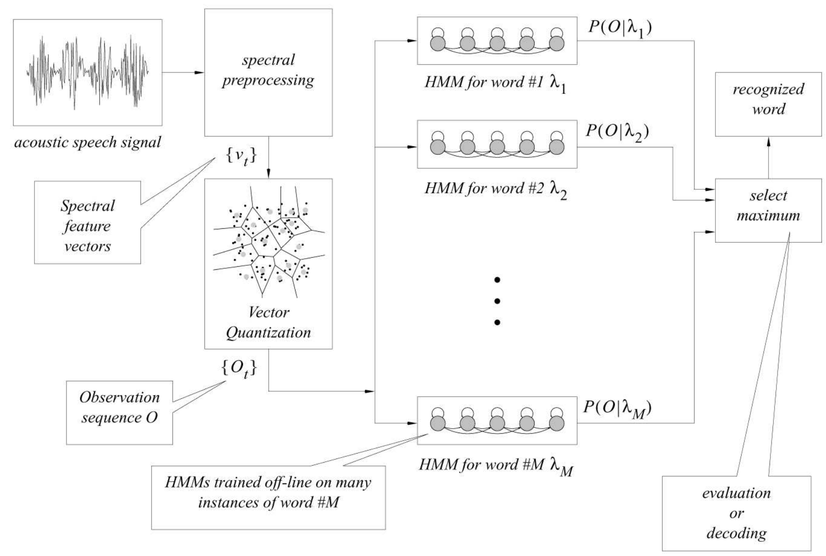
2. Decoding: Compute most likely state sequence Q^* **:**

$$Q^* = \{q_t\}, t = \{1, \ldots, T\}.$$

3. Training: Compute maximum-likelihood estimate λ^* **:**

$$P(O|\lambda) \le P(O|\lambda^*), \forall \lambda$$

Sample HMM system



Evaluation problem

Given:

- $O = \{O_t\}, t = \{1, \ldots, T\}$
- $\lambda = \{A, B, \pi\}$ = hidden Markov model

Compute: $P(O|\lambda)$

Obstacle: don't know underlying state sequence $Q = \{q_t\}$

Evaluation problem

For a specific underlying state sequence:

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

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T} = \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1} q_t}$$

(same as observable Markov model)

$$P(O, Q|\lambda) = P(O|Q, \lambda) P(Q|\lambda)$$

$$P(O|\lambda) = \sum_Q P(O, Q|\lambda) = \sum_Q P(O|Q, \lambda) P(Q|\lambda)$$

Evaluation problem

$$P(O|\lambda) = \sum_Q P(O, Q|\lambda) = \sum_Q P(O|Q, \lambda)P(Q|\lambda)$$

What's the problem with this?

Evaluation problem

$$P(O|\lambda) = \sum_Q P(O, Q|\lambda) = \sum_Q P(O|Q, \lambda)P(Q|\lambda)$$

- N^T **possible state sequences**
- $2TN^T$ **total operations**
- **Example:**

  $N = 5, T = 100$

  $2 \times 100 \times 5^{100} \approx 10^{72}$ operations.

Conclusion: need more efficient procedure...

Forward-backward algorithm

- **Efficient formulation of evaluation problem (eliminate repeated computations).**
- **Also useful for maximum-likelihood estimation (training) problem**

Define:

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

$\beta_t(i) = P(O_{t+1}, \ldots, O_T|q_t = S_i, \lambda)$ *(backward variable)*

Forward algorithm: Initialization (step 1)

$$\alpha_1(i) = P(O_1, q_1 = S_i|\lambda)$$

$$\alpha_1(i) = P(O_1|q_1 = S_i, \lambda)P(q_1 = S_i|\lambda)$$

$$\alpha_1(i) = \pi_i b_i(O_1), \; i \in \{1, \ldots, N\}.$$

Forward algorithm: Induction (step 2)

$$\alpha_{t+1}(j) = P(O_1, \ldots, O_{t+1}, q_{t+1} = S_j | \lambda)$$

$$\alpha_{t+1}(j) =$$

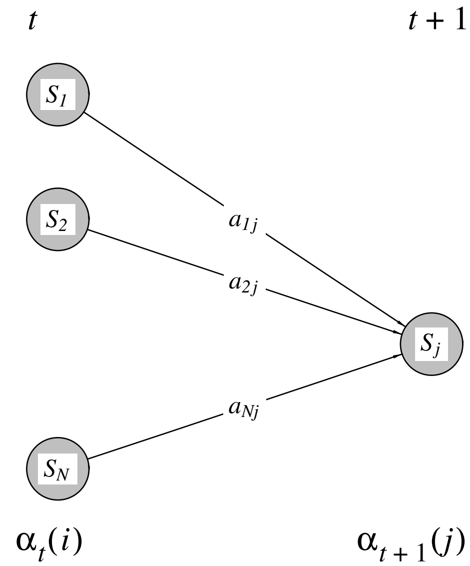
$$\left[\sum_{i=1}^{N} P(O_1, \ldots, O_t, q_t = S_i | \lambda) P(q_{t+1} = S_j | q_t = S_i, \lambda)\right] \times$$

$$P(O_{t+1} | q_{t+1} = S_j, \lambda)$$

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right) b_j(O_{t+1}) \text{ , } t \in \{1, \ldots, T\} \text{ ,}$$

$$j \in \{1, \ldots, N\} \text{ .}$$

Forward algorithm: induction diagram



Forward algorithm: Completion (step 3)

By definition:

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

so that:

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

- $N^2 T$ **total operations (compare to** $2TN^T$**)**
- **For** $N = 5$**,** $T = 100$**:**

  $5^2 \times 100 = 2500$ operations.

Forward algorithm (complete)

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

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1) \text{ , } i \in \{1, \ldots, N\} \text{ .}$$

2. Induction:

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right) b_j(O_{t+1}) \text{ , } t \in \{1, \ldots, T\} \text{ .}$$

3. Completion:

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

Backward algorithm (complete)

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

1. Initialization:

$$\beta_T(i) \equiv 1 \,,\; i \in \{1, \ldots, N\} \,.$$

2. Induction:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \,,\; t \in \{T-1, \ldots, 1\} \,.$$

3. Completion:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(i) \beta_t(i) \,,\; \forall t \,.$$

Hidden Markov models: 3 basic problems

Definitions:

- $O = \{O_t\} \,,\; t = \{1, \ldots, T\}$
- $\lambda = \{A, B, \pi\}$ = hidden Markov model

1. Evaluation: Compute $P(O|\lambda)$ **.**

2. Decoding: Compute most likely state sequence Q^* **:**

$$Q^* = \{q_t\} \,,\; t = \{1, \ldots, T\} \,.$$

3. Training: Compute maximum-likelihood estimate λ^* **:**

$$P(O|\lambda) \le P(O|\lambda^*) \,,\; \forall \lambda$$

Training problem: maximum-likelihood estimates

Suppose I knew the underlying state sequence $Q = \{q_t\}$ **corresponding to observation sequence** $O = \{O_t\}$ **.**

Maximum-likelihood estimates:

$$a_{ij} = \frac{\text{number of transitions from state } S_i \text{ to state } S_j}{\text{number of transitions from state } S_i}$$ *(same as for observable Markov models)*

$$b_j(k) = \frac{\text{number of times in state } S_j \text{ and observing symbol } v_k}{\text{number of times in state } S_j}$$

$$\pi_i = (\text{number of times in state } S_i \text{ at time } t = 1)$$

Training problem: maximum-likelihood estimates

When we don't know underlying state sequence:

$$\bar{a}_{ij} = \frac{\text{expected \# of transitions from state } S_i \text{ to state } S_j}{\text{expected \# of transitions from state } S_i}$$

$$\bar{b}_j(k) = \frac{\text{expected \# of times in state } S_j \text{ \& observing symbol } v_k}{\text{expected \# of times in state } S_j}$$

$$\bar{\pi}_i = (\text{expected \# of times in state } S_i \text{ at time } t = 1)$$

How do we get this? EM, but of course...

EM derivation of maximum-likelihodd estimates

We'll derive the update formula for a_{ij}.

Hidden variables:

$$y_t(i,j) = \begin{cases} 1, \text{ at time } t\text{, the system transitions from } S_i \text{ to } S_j \\ 0, \text{ otherwise} \end{cases}$$

$$y(i,j) = \text{number of transitions from state } S_i \text{ to state } S_j$$

Note:

$$y(i,j) = \sum_{t=1}^{T-1} y_t(i,j) .$$

EM derivation for A

More hidden variables:

$$y_t(i) = \begin{cases} 1, \text{ at time } t\text{, the system transitions from state } S_i \\ 0, \text{ otherwise} \end{cases}$$

$$y(i) = \text{number of transitions from state } S_i$$

Note:

$$y(i) = \sum_{t=1}^{T-1} y_t(i) .$$

EM derivation for A

Maximum-likelihood estimate in terms of hidden variables:

$$a_{ij} = \frac{\text{number of transitions from state } S_i \text{ to state } S_j}{\text{number of transitions from state } S_i}$$

$$a_{ij} = \frac{y(i,j)}{y(i)} = \frac{\sum_{t=1}^{T-1} y_t(i,j)}{\sum_{t=1}^{T-1} y_t(i)}$$

EM derivation for A

Replace hidden variables with E[hidden variables]:

$$a_{ij} = \frac{y(i,j)}{y(i)} = \frac{\sum_{t=1}^{T-1} y_t(i,j)}{\sum_{t=1}^{T-1} y_t(i)}$$

$$\bar{a}_{ij} = \frac{E[y(i,j)]}{E[y(i)]} = \frac{\sum_{t=1}^{T-1} E[y_t(i,j)]}{\sum_{t=1}^{T-1} E[y_t(i)]}$$

EM derivation for A

Expression for $E[y_t(i,j)]$:

$$E[y_t(i,j)] = [0 \cdot P(y_t(i,j) = 0) + 1 \cdot P(y_t(i,j) = 1)]$$

$$E[y_t(i,j)] = P(y_t(i,j) = 1)$$

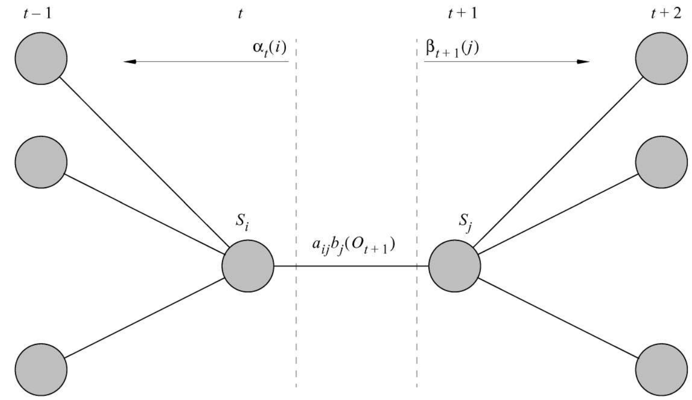
$$P(y_t(i,j) = 1) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$$

Using Bayes theorem:

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

EM derivation for A

$$E[y_t(i,j)] = \frac{P(q_t = S_i, q_{t+1} = S_j, O | \lambda)}{P(O | \lambda)}$$



Expression for $E[y_t(i,j)]$

$$E[y_t(i,j)] = \frac{P(q_t = S_i, q_{t+1} = S_j, O | \lambda)}{P(O | \lambda)}$$

So:

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

$$\beta_{t+1}(j) = P(O_{t+2}, \ldots, O_T | q_{t+1} = S_j, \lambda)$$

$$E[y_t(i,j)] = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O | \lambda)}$$

Expression for $E[y_t(i)]$

$$E[y_t(i)] = [0 \cdot P(y_t(i) = 0) + 1 \cdot P(y_t(i) = 1)]$$

$$E[y_t(i)] = P(y_t(i) = 1)$$

$$P(y_t(i) = 1) = P(q_t = S_i | O, \lambda)$$

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

$$P(q_t = S_i | O, \lambda) = \sum_{j=1}^{N} E[y_t(i,j)]$$

EM update formula for A

Iterative update for elements of A state-transition matrix:

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} E[y_t(i,j)]}{\sum_{t=1}^{T-1}\sum_{j=1}^{N} E[y_t(i,j)]}$$

where,

$$E[y_t(i,j)] = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)}$$

Simplified expression for EM update formula: single observation sequence

$$\bar{a}_{ij} = \frac{\frac{1}{P(O|\lambda)}\sum_{t=1}^{T-1} \alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\frac{1}{P(O|\lambda)}\sum_{t=1}^{T-1}\alpha_t(i)\beta_t(i)}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{t=1}^{T-1}\alpha_t(i)\beta_t(i)}$$

Expression for EM update formula: multiple observation sequences

Multiple observation sequences: $\mathbf{O} = \{O^{(1)}, \ldots, O^{(M)}\}$

$$\bar{a}_{ij} = \frac{\sum_{m=1}^{M}\frac{1}{P(O^{(m)}|\lambda)}\sum_{t=1}^{T_m-1} \alpha_t^m(i)a_{ij}b_j(O_{t+1}^{(m)})\beta_{t+1}^m(j)}{\sum_{m=1}^{M}\frac{1}{P(O^{(m)}|\lambda)}\sum_{t=1}^{T_m-1}\alpha_t^m(i)\beta_t^m(i)}$$

Hidden Markov models: 3 basic problems

Definitions:

- $O = \{O_t\}, t = \{1, \ldots, T\}$
- $\lambda = \{A, B, \pi\}$ = hidden Markov model

1. Evaluation: Compute $P(O|\lambda)$**.**

2. Decoding: Compute most likely state sequence Q^* **:**

$$Q^* = \{q_t\}, t = \{1, \ldots, T\}.$$

3. Training: Compute maximum-likelihood estimate λ^* **:**

$$P(O|\lambda) \le P(O|\lambda^*), \forall\lambda$$

Decoding problem

Given:

- $O = \{O_t\}, t = \{1, \ldots, T\}$
- $\lambda = \{A, B, \pi\}$ = hidden Markov model

Compute:

- $Q^* = \{q_t\}, t = \{1, \ldots, T\}$ s.t.:

  $P(Q^*|O, \lambda) > P(Q|O, \lambda), \forall Q \neq Q^*$.

Viterbi algorithm

From basic probability theory:

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

Therefore:

Maximizing $P(Q, O|\lambda)$ implies maximizing $P(Q|O, \lambda)$.

Viterbi algorithm (continued):

Definition:

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

(what does this mean?)

Recursive relationship:

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] b_j(O_{t+1})$$

(why?)

Basis of Viterbi algorithm...

Viterbi algorithm (continued):

Initialization:

$$\delta_1(i) = P(q_1 = S_i, O_1|\lambda)$$

$$\delta_1(i) = \pi_i b_i(O_1), i \in \{1, \ldots, N\}$$

Induction:

$$\delta_t(j) = [\max_i \delta_{t-1}(i) a_{ij}] b_j(O_t)$$

$$\psi_t(j) = \operatorname*{argmax}_i [\delta_{t-1}(i) a_{ij}], j \in \{1, \ldots, N\},$$

$$t \in \{2, \ldots, T\}$$

Viterbi algorithm (continued):

Termination:

$$P^* = \max_{Q} P(Q, O|\lambda)$$

$$P^* = \max_{i} \delta_T(i)$$

$$q_T^* = \operatorname{argmax}_{i} \delta_T(i)$$

Path (state-sequence) back tracking:

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

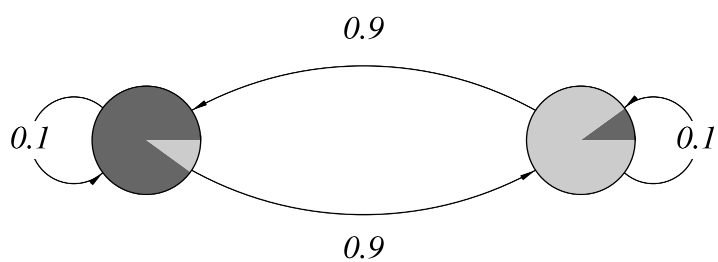
$$Q^* = \{q_t^*\}, \ t \in \{1, \ldots, T\}.$$

Additional topics (covered in notes)

- **Update equation for output probability matrix B (very similar to A matrix).**
- **Scaling of update equations to prevent numerical underflow.**
- **Continuous-output HMMs**
  - *Much more computationally intensive*
  - *Table lookup replaced by complex pdf evaluation at every step of forward-backward algorithm*

Simple training example

Generated observation sequence of length 10,000 from:



$$A = \begin{bmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \end{bmatrix}, \ B = \begin{bmatrix} 0.1 & 0.9 \\ 0.9 & 0.1 \end{bmatrix}, \ \pi = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

Trained models for different number of states