Hidden Markov Models

Le Anh Cuong

Reference:

(CMSC 723: Introduction to Computational Linguistics)

Reading

- Chapter 5, 6 [1]
- Chapter 9 [2]

Outline

- Hidden Markov Models
- The three fundamental problems for HMM
- HMMs : Implementation, properties, and variants

Hidden Markov Model (HMM)

- HMMs allow you to estimate probabilities of unobserved events
- Given plain text, which underlying parameters generated the surface
- E.g., in speech recognition, the observed data is the acoustic signal and the words are the hidden parameters

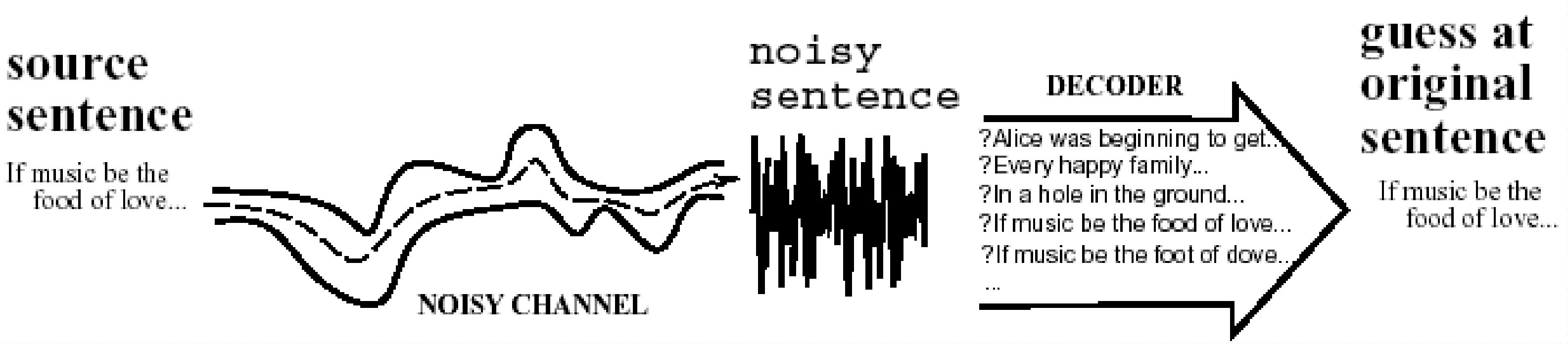
HMMs and their Usage

- HMMs are very common in Computational Linguistics:
  - Speech recognition (observed: acoustic signal, hidden: words)
  - Handwriting recognition (observed: image, hidden: words)
  - Part-of-speech tagging (observed: words, hidden: part-of-speech tags)
  - Machine translation (observed: foreign words, hidden: words in target language)

Noisy Channel Model

- In speech recognition you observe an acoustic signal ($A=a_1,\ldots,a_n$) and you want to determine the most likely sequence of words ($W=w_1,\ldots,w_n$): $P(W \mid A)$

Noisy Channel in a Picture



Noisy Channel Model

- Assume that the acoustic signal (A) is already segmented wrt word boundaries
- P(W | A) could be computed as

$$P(W \mid A) = \prod_{a_i} \max_{w_i} P(w_i \mid a_i)$$

- Problem: Finding the most likely word corresponding to a acoustic representation depends on the context
- E.g., /'pre-z&ns / could mean “presents” or “presence” depending on the context

Noisy Channel Model

- Given a candidate sequence W we need to compute P(W) and combine it with P(W | A)
- Applying Bayes' rule:

$$\operatorname*{argmax}_{W} P(W \mid A) = \operatorname*{argmax}_{W} \frac{P(A \mid W) P(W)}{P(A)}$$

- The denominator P(A) can be dropped, because it is constant for all W

Decoding

The decoder combines evidence from

- The likelihood: P(A | W)

  This can be approximated as:

$$P(A \mid W) \approx \prod_{i=1}^{n} P(a_i \mid w_i)$$

- The prior: P(W)

  This can be approximated as:

$$P(W) \approx P(w_1) \prod_{i=2}^{n} P(w_i \mid w_{i-1})$$

Search Space

- Given a word-segmented acoustic sequence list all candidates

'bot	ik-'spen-siv	'pre-z&ns
boat $P('bot \mid bald)$	excessive	presidents
bald $P(inactive \mid bald)$	expensive	presence
bold	expressive	presents
bought	inactive	press

- Compute the most likely path

Markov Assumption

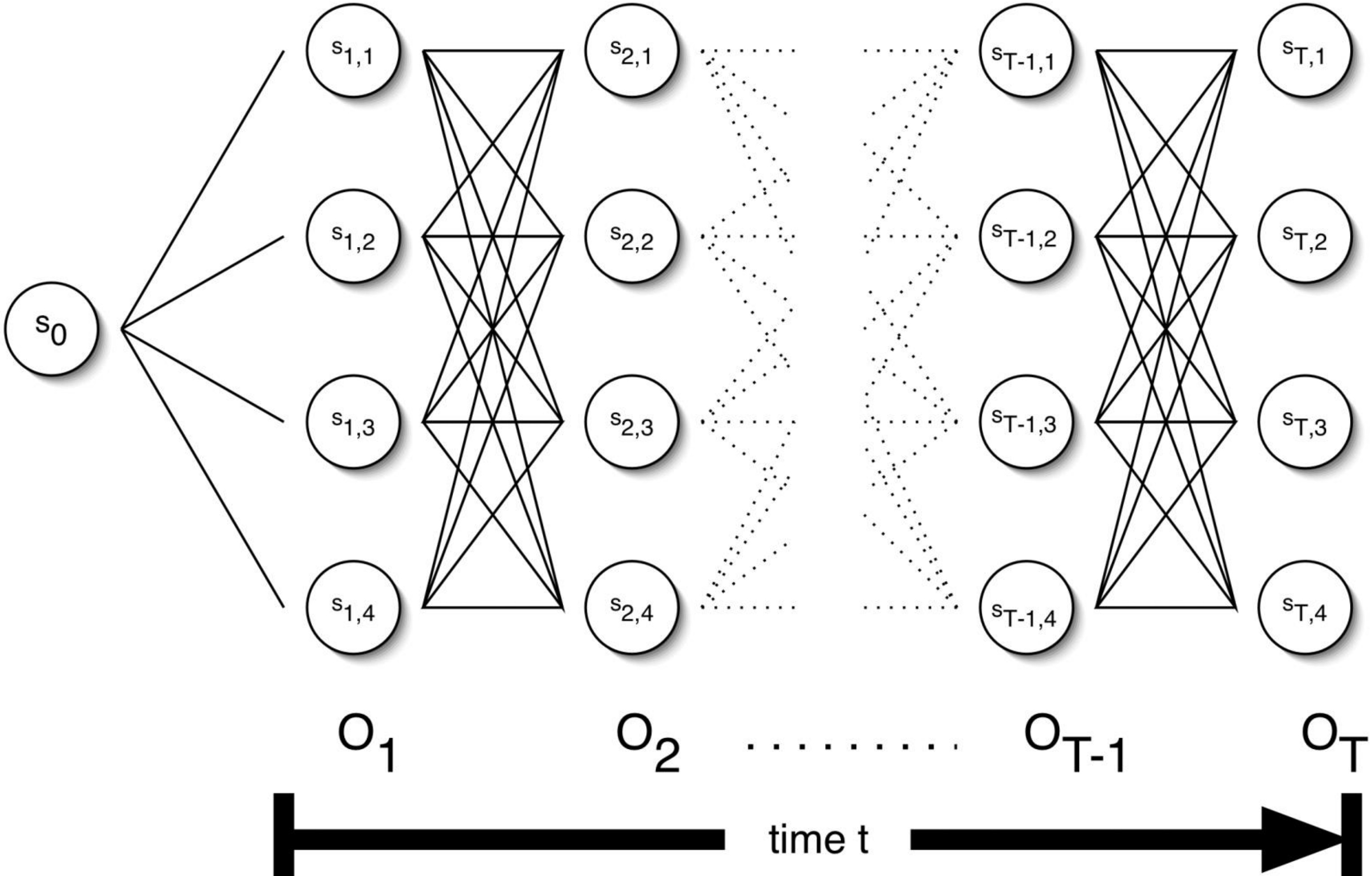
- The Markov assumption states that probability of the occurrence of word w_i at time t depends only on occurrence of word w_{i-1} at time t-1
  - Chain rule:

$$P(w_1,\ldots,w_n) = \prod_{i=2}^{n} P(w_i \mid w_1,\ldots,w_{i-1})$$

  - Markov assumption:

$$P(w_1,\ldots,w_n) \approx \prod_{i=2}^{n} P(w_i \mid w_{i-1})$$

The Trellis



s_0
$s_{1,1}$
$s_{1,2}$
$s_{1,3}$
$s_{1,4}$
$s_{2,1}$
$s_{2,2}$
$s_{2,3}$
$s_{2,4}$
$s_{T-1,1}$
$s_{T-1,2}$
$s_{T-1,3}$
$s_{T-1,4}$
$s_{T,1}$
$s_{T,2}$
$s_{T,3}$
$s_{T,4}$
o_1
o_2
o_{T-1}
o_T
time t

Parameters of an HMM

- States: A set of states $S=s_1,\ldots,s_n$
- Transition probabilities: $A= a_{1,1},a_{1,2},\ldots,a_{n,n}$ Each $a_{i,j}$ represents the probability of transitioning from state s_i to s_j.
- Emission probabilities: a set B of functions of the form $b_i(o_t)$ which is the probability of observation o_t being emitted by s_i
- Initial state distribution: π_i is the probability that s_i is a start state

The Three Basic HMM Problems

- Problem 1 (Evaluation): Given the observation sequence $O=o_1,\ldots,o_T$ and an HMM model

  $$\lambda = (A, B, \pi)$$

  , how do we compute the probability of O given the model?
- Problem 2 (Decoding): Given the observation sequence $O=o_1,\ldots,o_T$ and an HMM model

  $$\lambda = (A, B, \pi)$$

  , how do we find the state sequence that best explains the observations?

The Three Basic HMM Problems

- Problem 3 (Learning): How do we adjust the model parameters

$$\lambda = (A, B, \pi)$$

to maximize

$$P(O \mid \lambda)$$

Problem 1: Probability of an Observation Sequence

- What is $P(O \mid \lambda)$?
- The probability of a observation sequence is the sum of the probabilities of all possible state sequences in the HMM.
- Naïve computation is very expensive. Given T observations and N states, there are N^T possible state sequences.
- Even small HMMs, e.g. T=10 and N=10, contain 10 billion different paths
- Solution to this and problem 2 is to use dynamic programming

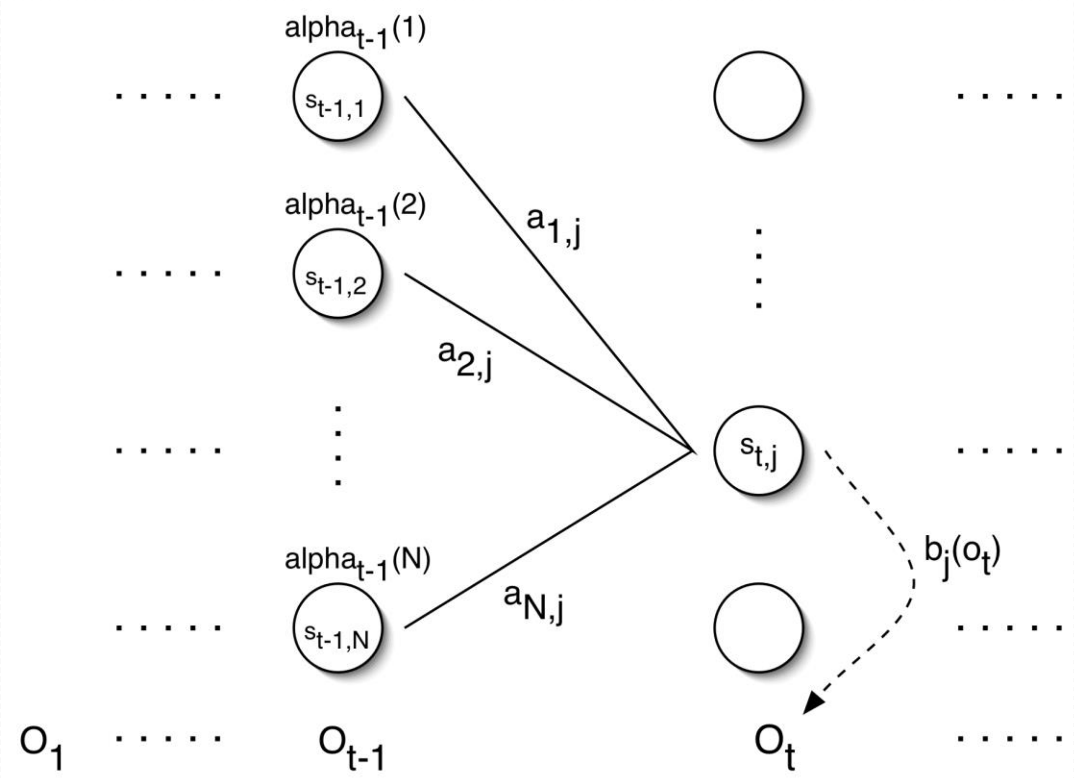
Forward Probabilities

- What is the probability that, given an HMM λ, at time t the state is i and the partial observation $o_1 \ldots o_t$ has been generated?

$$\alpha_t(i) = P(o_1 \ldots o_t, q_t = s_i \mid \lambda)$$

Forward Probabilities

$$\alpha_t(i) = P(o_1 \ldots o_t, q_t = s_i \mid \lambda)$$



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

Forward Algorithm

- Initialization: $$\alpha_1(i) = \pi_i b_i(o_1) \quad 1 \le i \le N$$

- Induction:

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t) \quad 2 \le t \le T, 1 \le j \le N$$

- Termination:

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

Forward Algorithm Complexity

- In the naïve approach to solving problem 1 it takes on the order of $2T*N^T$ computations
- The forward algorithm takes on the order of N^2T computations

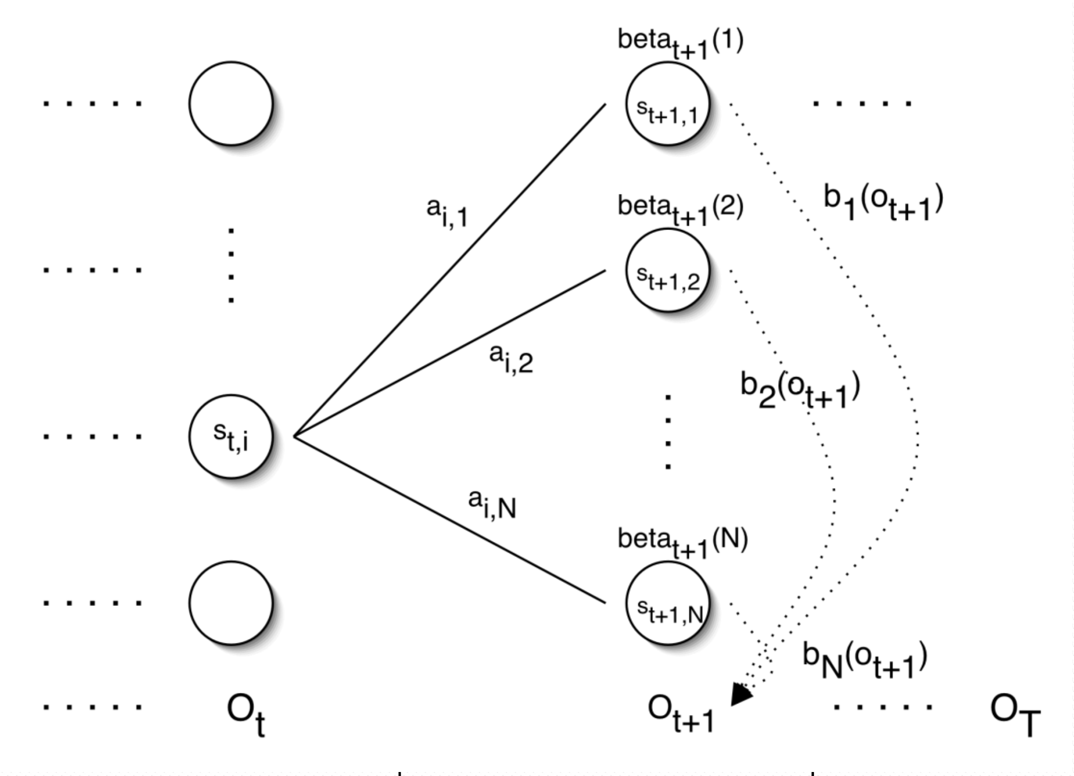
Backward Probabilities

- Analogous to the forward probability, just in the other direction
- What is the probability that given an HMM λ and given the state at time t is i, the partial observation $o_{t+1} \ldots o_T$ is generated?

$$\beta_t(i) = P(o_{t+1} \ldots o_T \mid q_t = s_i, \lambda)$$

Backward Probabilities

$$\beta_t(i) = P(o_{t+1} \ldots o_T \mid q_t = s_i, \lambda)$$



$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \right]$$

Backward Algorithm

- Initialization:

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

- Induction:

$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \right] \quad t = T-1 \ldots 1, 1 \le i \le N$$

- Termination:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \pi_i \beta_1(i)$$

Problem 2: Decoding

- The solution to Problem 1 (Evaluation) gives us the sum of all paths through an HMM efficiently.
- For Problem 2, we wan to find the path with the highest probability.
- We want to find the state sequence $Q=q_1 \ldots q_T$, such that

$$Q = \arg\max_{Q'} P(Q' \mid O, \lambda)$$

Viterbi Algorithm

- Similar to computing the forward probabilities, but instead of summing over transitions from incoming states, compute the maximum
- Forward:
- Viterbi Recursion:

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

$$\delta_t(j) = \left[\max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij} \right] b_j(o_t)$$

Viterbi Algorithm

- Initialization: $\delta_1(i) = \pi_i b_j(o_1) \quad 1 \le i \le N$
- Induction:

$$\delta_t(j) = \left[\max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij}\right] b_j(o_t)$$

$$\psi_t(j) = \left\lfloor \underset{1 \le i \le N}{\arg\max}\, \delta_{t-1}(i)\, a_{ij} \right\rfloor \quad 2 \le t \le T, 1 \le j \le N$$

- Termination:

$$p^* = \max_{1 \le i \le N} \delta_T(i) \qquad q_T^* = \underset{1 \le i \le N}{\arg\max}\, \delta_T(i)$$

- Read out path:

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

Problem 3: Learning

- Up to now we've assumed that we know the underlying model $\lambda = (A, B, \pi)$
- Often these parameters are estimated on annotated training data, which has two drawbacks:
  - Annotation is difficult and/or expensive
  - Training data is different from the current data
- We want to maximize the parameters with respect to the current data, i.e., we're looking for a model λ', such that

$$\lambda' = \arg\max_{\lambda} P(O \mid \lambda)$$

Problem 3: Learning

- Unfortunately, there is no known way to analytically find a global maximum, i.e., a model λ', such that

$$\lambda' = \arg\max_{\lambda} P(O \mid \lambda)$$

- But it is possible to find a local maximum
- Given an initial model λ, we can always find a model λ', such that

$$P(O \mid \lambda') \geq P(O \mid \lambda)$$

Parameter Re-estimation

- Use the forward-backward (or Baum-Welch) algorithm, which is a hill-climbing algorithm
- Using an initial parameter instantiation, the forward-backward algorithm iteratively re-estimates the parameters and improves the probability that given observation are generated by the new parameters

Parameter Re-estimation

- Three parameters need to be re-estimated:
  - Initial state distribution: π_i
  - Transition probabilities: $a_{i,j}$
  - Emission probabilities: $b_i(o_t)$

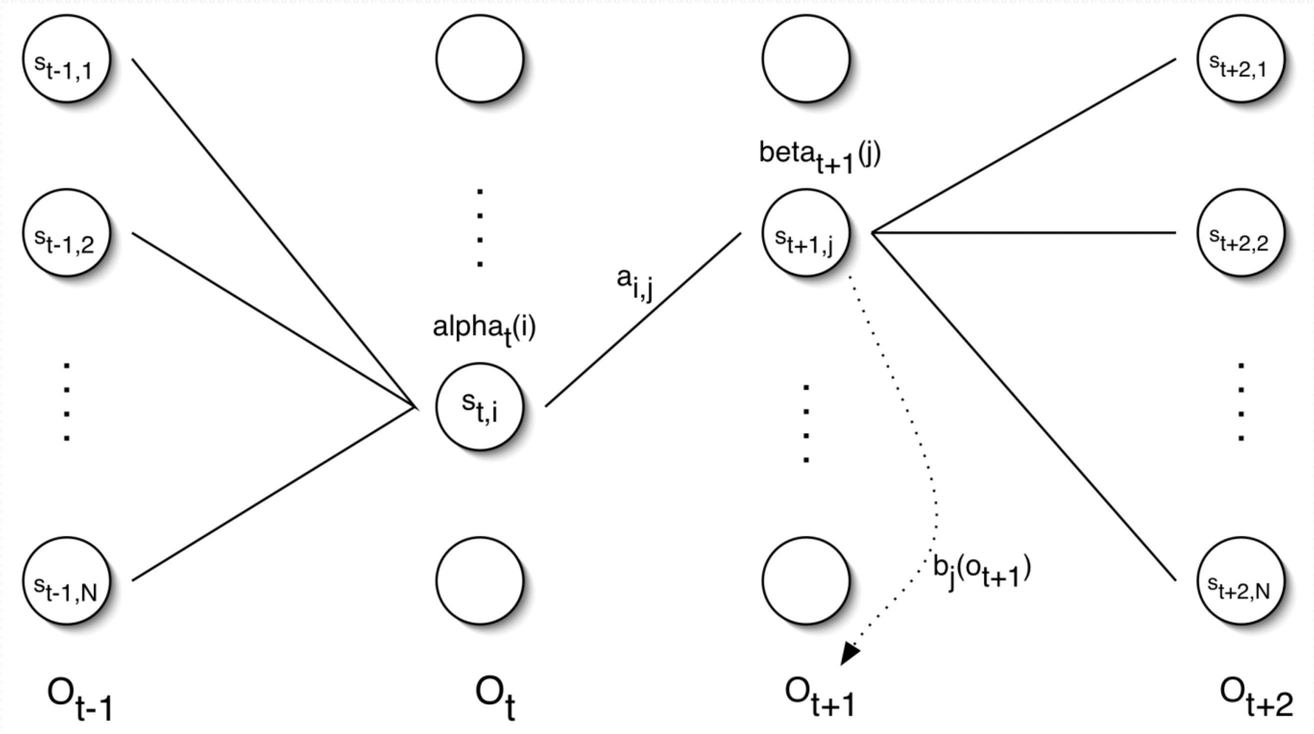
Re-estimating Transition Probabilities

- What's the probability of being in state s_i at time t and going to state s_j, given the current model and parameters?

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda)$$

Re-estimating Transition Probabilities

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda)$$



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

Re-estimating Transition Probabilities

- The intuition behind the re-estimation equation for transition probabilities is

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

- Formally:

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j'=1}^{N} \xi_t(i,j')}$$

Re-estimating Transition Probabilities

- Defining $$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

  As the probability of being in state s_i, given the complete observation O

- We can say: $$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

Review of Probabilities

- Forward probability: $\alpha_t(i)$

  The probability of being in state s_i, given the partial observation $o_1,\ldots,o_t$
- Backward probability: $\beta_t(i)$

  The probability of being in state s_i, given the partial observation $o_{t+1},\ldots,o_T$
- Transition probability: $\xi_t(i,j)$

  The probability of going from state s_i, to state s_j, given the complete observation $o_1,\ldots,o_T$
- State probability: $\gamma_t(i)$

  The probability of being in state s_i, given the complete observation $o_1,\ldots,o_T$

Re-estimating Initial State Probabilities

- Initial state distribution: π_i is the probability that s_i is a start state
- Re-estimation is easy:

$$\hat{\pi}_i = \text{expected number of times in state } s_i \text{ at time } 1$$

- Formally:

$$\hat{\pi}_i = \gamma_1(i)$$

Re-estimation of Emission Probabilities

- Emission probabilities are re-estimated as

$$\hat{b}_i(k) = \frac{\text{expected number of times in state } s_i \text{ and observe symbol } v_k}{\text{expected number of times in state } s_i}$$

- Formally:

$$\hat{b}_i(k) = \frac{\sum_{t=1}^{T} \delta(o_t, v_k)\, \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$$

Where

$$\delta(o_t, v_k) = 1, \text{ if } o_t = v_k, \text{ and 0 otherwise}$$

Note that δ here is the Kronecker delta function and is not related to the δ in the discussion of the Viterbi algorithm!!

The Updated Model

- Coming from $\lambda = (A, B, \pi)$
- we get to $\lambda' = (\hat{A}, \hat{B}, \hat{\pi})$

by the following update rules:

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad \hat{b}_i(k) = \frac{\sum_{t=1}^{T} \delta(o_t, v_k)\gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \qquad \hat{\pi}_i = \gamma_1(i)$$

Expectation Maximization

- The forward-backward algorithm is an instance of the more general EM algorithm
  - The E Step: Compute the forward and backward probabilities for a given model
  - The M Step: Re-estimate the model parameters