Hidden Markov Models

Adapted from
Dr Catherine Sweeney-Reed’s slides
Summary

- Introduction
- Description
- Central problems in HMM modelling
- Extensions
- Demonstration
Specification of an HMM

- $N$ - number of states
  - $Q = \{q_1; q_2; \ldots; q_T\}$ - set of states
- $M$ - the number of symbols (observables)
  - $O = \{o_1; o_2; \ldots; o_T\}$ - set of symbols
Specification of an HMM

- **A** - the state transition probability matrix
  \[ a_{ij} = P(q_{t+1} = j|q_t = i) \]

- **B** - observation probability distribution
  \[ b_j(k) = P(o_t = k|q_t = j) \quad i \leq k \leq M \]

- **\( \pi \)** - the initial state distribution
Specification of an HMM

Full HMM is thus specified as a triplet:

\[ \lambda = (A, B, \pi) \]
Central problems in HMM modelling

- **Problem 1**

  Evaluation:
  - Probability of occurrence of a particular observation sequence, $O = \{o_1, \ldots, o_k\}$, given the model
  - $P(O|\lambda)$
  - Complicated – hidden states
  - Useful in sequence classification
Central problems in HMM modelling

- **Problem 2**
  - **Decoding:**
    - Optimal state sequence to produce given observations, \( O = \{o_1, \ldots, o_k\} \), given model
    - Optimality criterion
    - Useful in recognition problems
Central problems in HMM modelling

**Problem 3**

Learning:

- Determine optimum model, given a training set of observations
- Find $\lambda$, such that $P(O|\lambda)$ is maximal
Problem 1: Naïve solution

- State sequence $Q = (q_1, \ldots q_T)$
- Assume independent observations:

$$P(O \mid q, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) = b_{q_1}(o_1)b_{q_2}(o_2)\ldots b_{q_T}(o_T)$$

NB Observations are mutually independent, given the hidden states. (Joint distribution of independent variables factorises into marginal distributions of the independent variables.)
Problem 1: Naïve solution

- Observe that:

\[ P(q \mid \lambda) = \prod_{q_1} a_{q_1} q_2 a_{q_2} q_3 \ldots a_{q_{T-1}} q_T \]

- And that:

\[ P(O \mid \lambda) = \sum_{q} P(O \mid q, \lambda) P(q \mid \lambda) \]
Problem 1: Naïve solution

Finally get:

$$P(O \mid \lambda) = \sum_q P(O \mid q, \lambda)P(q \mid \lambda)$$

NB:
- The above sum is over all state paths
- There are $N^T$ states paths, each ‘costing’ $O(T)$ calculations, leading to $O(TN^T)$ time complexity.
Problem 1: Efficient solution

Forward algorithm:

Define auxiliary forward variable $\alpha$:

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

$\alpha_t(i)$ is the probability of observing a partial sequence of observables $o_1,...,o_t$ such that at time $t$, state $q_t=i$.
Problem 1: Efficient solution

- Recursive algorithm:
  - Initialise:
    \[ \alpha_1(i) = \pi_i b_i(o_1) \]
  - Calculate:
    \[ \alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \]
  - Obtain:
    \[ P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \]

Complexity is \( O(N^2T) \)
Problem 1: Alternative solution

Backward algorithm:

- Define auxiliary forward variable $\beta$:

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

$\beta_t(i)$ – the probability of observing a sequence of observables $o_{t+1}, \ldots, o_T$ given state $q_t = i$ at time $t$, and $\lambda$
Problem 1: Alternative solution

- Recursive algorithm:
  - Initialise:
    \[ \beta_T(j) = 1 \]
  - Calculate:
    \[ \beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j)a_{ij}b_j(o_{t+1}) \]
  - Terminate:
    \[ p(O|\lambda) = \sum_{i=1}^{N} \beta_1(i) \quad t = T - 1, \ldots, 1 \]

Complexity is \( O(N^2T) \)
Problem 2: Decoding

- Choose state sequence to maximise probability of observation sequence
- Viterbi algorithm - inductive algorithm that keeps the best state sequence at each instance
Problem 2: Decoding

Viterbi algorithm:

- State sequence to maximise $P(O,Q|\lambda)$:

$$P(q_1, q_2, ..., q_T \mid O, \lambda)$$

- Define auxiliary variable $\delta$:

$$\delta_t(i) = \max_{q} P(q_1, q_2, ..., q_t = i, o_1, o_2, ..., o_t \mid \lambda)$$

$\delta_t(i)$ – the probability of the most probable path ending in state $q_t=i$
Problem 2: Decoding

- Recurrent property:

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

- Algorithm:
  1. Initialise:

\[
\delta_1(i) = \pi_i b_i(o_1) \quad 1 \leq i \leq N
\]

\[
\psi_1(i) = 0
\]
Problem 2: Decoding

2. Recursion:

\[ \delta_t(j) = \max_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij}) b_j(o_t) \]

\[ \psi_t(j) = \arg \max_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij}) \quad 2 \leq t \leq T, 1 \leq j \leq N \]

3. Terminate:

\[ P^* = \max_{1 \leq i \leq N} \delta_T(i) \]

\[ q_T^* = \arg \max_{1 \leq i \leq N} \delta_T(i) \]

P* gives the state-optimised probability

Q* is the optimal state sequence

(Q* = \{q1*, q2*, ..., qT*\})
Problem 2: Decoding

4. Backtrack state sequence:

\[ q_t^* = \psi_{t+1}(q_{t+1}^*) \quad t + T - 1, T - 2, ..., 1 \]

O(N^2T) time complexity
Problem 3: Learning

- Training HMM to encode obs seq such that HMM should identify a similar obs seq in future
- Find $\lambda = (A, B, \pi)$, maximising $P(O | \lambda)$
- General algorithm:
  - Initialise: $\lambda_0$
  - Compute new model $\lambda$, using $\lambda_0$ and observed sequence $O$
  - Then $\lambda_o \leftarrow \lambda$
  - Repeat steps 2 and 3 until:

\[
\log P(O | \lambda) - \log P(O | \lambda_0) < d
\]
Problem 3: Learning

Step 1 of Baum-Welch algorithm:

Let $\xi(i,j)$ be a probability of being in state $i$ at time $t$ and at state $j$ at time $t+1$, given $\lambda$ and $O$ seq

$$\xi(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O | \lambda)}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}$$
Problem 3: Learning

Operations required for the computation of the joint event that the system is in state $S_i$ and time $t$ and State $S_j$ at time $t+1$
Problem 3: Learning

- Let $\gamma_t(i)$ be a probability of being in state $i$ at time $t$, given $O$

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

- $\sum_{i=1}^{T-1} \gamma_t(i)$ - expected no. of transitions from state $i$

- $\sum_{i=1}^{T-1} \xi_t(i)$ - expected no. of transitions $i \rightarrow j$
Problem 3: Learning

Step 2 of Baum-Welch algorithm:

- $\hat{\pi} = \gamma_1(i)$ the expected frequency of state $i$ at time $t=1$

- $\hat{a}_{ij} = \frac{\sum \xi_t(i,j)}{\sum \gamma_t(i)}$ ratio of expected no. of transitions from state $i$ to $j$ over expected no. of transitions from state $i$

- $\hat{b}_j(k) = \frac{\sum_{t,o=1}^t \gamma_t(j)}{\sum \gamma_t(j)}$ ratio of expected no. of times in state $j$ observing symbol $k$ over expected no. of times in state $j$
Problem 3: Learning

- Baum-Welch algorithm uses the forward and backward algorithms to calculate the auxiliary variables $\alpha, \beta$
- B-W algorithm is a special case of the EM algorithm:
  - E-step: calculation of $\xi$ and $\gamma$
  - M-step: iterative calculation of $\hat{\pi}, \hat{a}_{ij}, \hat{b}_j(k)$
- Practical issues:
  - Can get stuck in local maxima
  - Numerical problems – log and scaling
Extensions

- Problem-specific:
  - Left to right HMM (speech recognition)
  - Profile HMM (bioinformatics)
Extensions

General machine learning:
- Factorial HMM
- Coupled HMM
- Hierarchical HMM
- Input-output HMM
- Switching state systems
- Hybrid HMM (HMM +NN)
- Special case of graphical models
  - Bayesian nets
  - Dynamical Bayesian nets
Examples

Coupled HMM

Extensions

Factorial HMM
HMMs – Sleep Staging

- Flexer, Sykacek, Rezek, and Dorffner (2000)
- Observation sequence: EEG data
- Fit model to data according to 3 sleep stages to produce continuous probabilities: P(wake), P(deep), and P(REM)
- Hidden states correspond with recognised sleep stages. 3 continuous probability plots, giving P of each at every second
HMMs – Sleep Staging

Manual scoring of sleep stages

Staging by HMM

Probability plots for the 3 stages
Excel

- Demonstration of a working HMM implemented in Excel
Further Reading