MACHINE LEARNING FOR SIGNAL PROCESSING

5-4-2025

Sriram Ganapathy
LEAP lab, Electrical Engineering, Indian Institute of Science,
sriramg@iisc.ac.in

Viveka Salinamakki, Varada R.
LEAP lab, Electrical Engineering, Indian Institute of Science

http://leap.ee.iisc.ac.in/sriram/teaching/MLSP25/

Graphical Models

Graphical Models

- What are graphs
  - ✳ What are graphical models
- Directed and Undirected graphs
- Conditional independence

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*
  - $Q = \{q_1; q_2; : : : ;q_T\}$ *- set of states*
- *M - the number of symbols (observables)*
  - $O = \{o_1; o_2; : : : ;o_T\}$ *- set of symbols*

Specification of an HMM

- A - *the state transition probability matrix*
  - $aij = P(q_{t+1} = j|q_t = i)$
- B- *observation probability distribution*
  - $b_j(k) = P(o_t = k|q_t = j) \quad i \leq k \leq M$
- *π - 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

- <u>Problem 3</u>

  Learning:
  - Determine optimum model, given a training set of observations
  - Find λ, such that P(O|λ) is maximal

Problem 1: Naïve solution

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

$$P(O \mid q, \lambda) = \prod_{i=1}^{T} P(o_t \mid q_t, \lambda) = b_{q1}(o_1) b_{q2}(o_2) \ldots b_{qT}(o_T)$$

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 :

$$P(q \mid \lambda) = \pi_{q1} a_{q1q2} a_{q2q3} \ldots a_{qT-1qT}$$

- 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 α:

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

$\alpha_t(i)$ is the probability of observing a partial sequence of observables $o_1,\ldots o_t$ such that at time t, state $q_t=i$

Central problems

Problem 1: Efficient solution

- Recursive algorithm:
  - Initialise: $$\alpha_1(i) = \pi_i b_i(o_1)$$
  - Calculate:

(Partial obs seq to *t* AND state *i* at *t*) x (transition to *j* at *t+1*) x (sensor)

  - Obtain: $$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(o_{t+1})$$

Sum, as can reach *j* from any preceding state

α incorporates partial obs seq to *t*

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

Sum of different ways of getting obs seq

Complexity is $O(N^2T)$

Problem 1: Alternative solution

Backward algorithm:

- Define auxiliary forward variable β:

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

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

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:

$$p(O \mid \lambda) = \sum_{i=1}^{N} \beta_1(i) \qquad t = T-1,\ldots,1$$

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

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

- Define auxiliary variable δ:

$$\delta_t(i) = \max_q P(q_1, q_2, \ldots, q_t = i, o_1, o_2, \ldots o_t \mid \lambda)$$

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

Problem 2: Decoding

To get state seq, need to keep track of argument to maximise this, for each *t* and *j*. Done via the array $\psi_t(j)$.

- Recurrent property:

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

- Algorithm:
  - 1. Initialise:

$$\delta_1(i) = \pi_i b_i(o_1) \qquad 1 \le i \le N$$

$$\psi_1(i) = 0$$

Problem 2: Decoding

Central problems

- 2. Recursion:

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

$$\psi_t(j) = \arg\max_{1 \le i \le N}(\delta_{t-1}(i)a_{ij}) \qquad 2 \le t \le T, 1 \le j \le N$$

- 3. Terminate:

$$P^* = \max_{1 \le i \le N} \delta_T(i)$$

P* gives the state-optimised probability

$$q_T^* = \arg\max_{1 \le i \le N} \delta_T(i)$$

Q* is the optimal state sequence
(Q* = {q1*,q2*,…,qT*})

Problem 2: Decoding

Central problems

- 4. Backtrack state sequence:

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

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 λ=(A,B,π), maximising P(O|λ)
- General algorithm:
  - Initialise: λ_0
  - Compute new model λ, using λ_0 and observed sequence O
  - Then $\lambda_0 \leftarrow \lambda$
  - Repeat steps 2 and 3 until:

$$\log P(O \mid \lambda) - \log P(O \mid \lambda_0) < d$$

Problem 3: Learning

Step 1 of Baum-Welch algorithm:

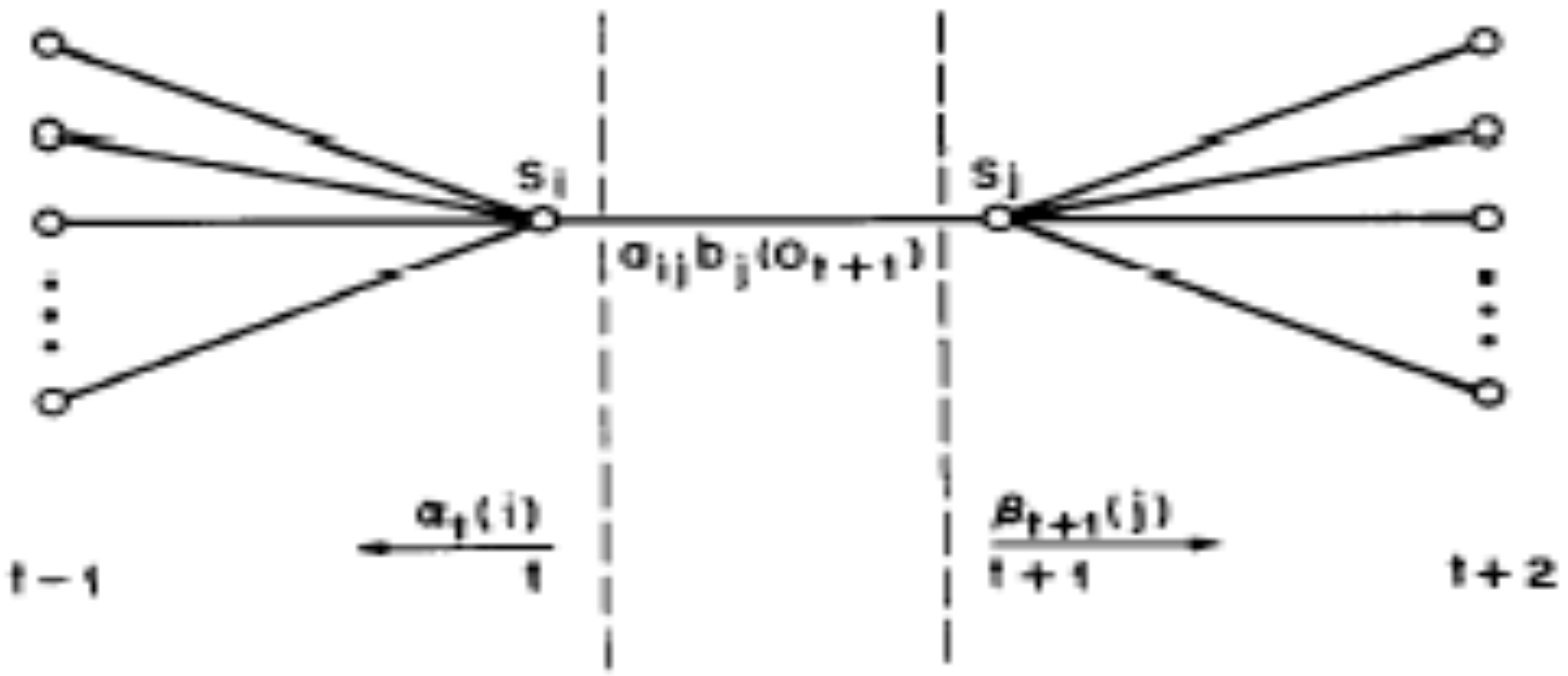
- Let *ξ(i,j)* be a probability of being in state *i* at time *t* and at state *j* at time *t+1*, given λ and O seq

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

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

Problem 3: Learning

Central problems



Operations required for the computation of the joint event that the system is in state Si and time t and State Sj at time t+1

Problem 3: Learning

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

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

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

Problem 3: Learning

Central problems

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*
- $\hat{b}_j(k) = \frac{\sum_{t,o_t=k} \gamma_t(j)}{\sum \gamma_t(j)}$ 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 α,β
- B-W algorithm is a special case of the EM algorithm:
  - E-step: calculation of ξ and γ
  - 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

STORY OF MLSP 2025

EM algorithm

Decision Theory

Generative Modeling

Gaussian Modeling

Gaussian Mixture Modeling

Classification Problem

Function Modeling

Linear Models for Regression and Classification

Kernel Machines & Max-margin classifiers

Support Vector Machines

Data Representations PCA, LDA

Discriminative Modeling

Gradient Descent

Neural Networks

Learning Rules & Normalization

Deep neural architectures

भारतीय विज्ञान संस्थान INDIAN INSTITUTE OF SCIENCE

LEAP

STORY OF MLSP 2025

Feed-forward Models

Learning with Regularization

Ensembling

Deep neural architectures

Convolutional Neural N/w

Other Topics

Recurrent Neural N/w

Attention & Transformers

Graphical Models

HMMs

Content Delivery

Theory and Mathematical Foundation

Intuition and Analysis

Implementation and Understanding

COURSE STRUCTURE (ROUGH SCHEDULE)

- Introduction to real world data and signals - text, speech, image, video.
- Dimensionality reduction - principal components, linear discriminants.
- Decision theory for pattern recognition, ML and MAP methods, Bias-variance trade-off, model assessment, cross-validation, estimating generalization error.
- Generative modeling and density estimation - Gaussian and mixture Gaussian models, kernel density estimators, hidden Markov models. Expectation Maximization.
- Linear regression and kernel methods. Regularization methods.
- Discriminative modeling - support vector machines, decision trees and random forest classifiers, bagging and boosting.
- Neural networks: gradient descent optimization and back propagation, regularization in neural networks, dropout. normalization methods.
- Introduction to deep learning - feedforward, convolutional and recurrent networks, practical considerations in deep learning.
- Introduction to transformer models - self and cross attention, encoder and decoder architectures, autoregressive decoding.

DATES OF VARIOUS RITUALS

- 3 Assignments spread over 3 months (roughly one assignment every 3 weeks). ✓
- February second half - Midterm ✓
- February 4th week - project topic and team finalization and proposal submission. [1 and 2 person teams]. ✓
- March 3rd week - Project MidTerm Presentations. ✓
- April 3rd week - Final Exam
- April last week - Project Final Presentations

THANK YOU

Sriram Ganapathy and TA team
LEAP lab, C328, EE, IISc
sriramg@iisc.ac.in