CSC 412/2506:
Probabilistic Learning and Reasoning

Week 6 - 1/2: Hidden Markov Models

Michal Malyska

University of Toronto

Overview

- Hidden Markov Models
- Forward / Backward Algorithm
- Viterbi Algorithm

Sequential data

We generally assume data was i.i.d, however this may be a poor assumption:

- Sequential data is common in time-series modelling (e.g. stock prices, speech, video analysis) or ordered (e.g. textual data, gene sequences).
- Recall the general joint factorization via the chain rule

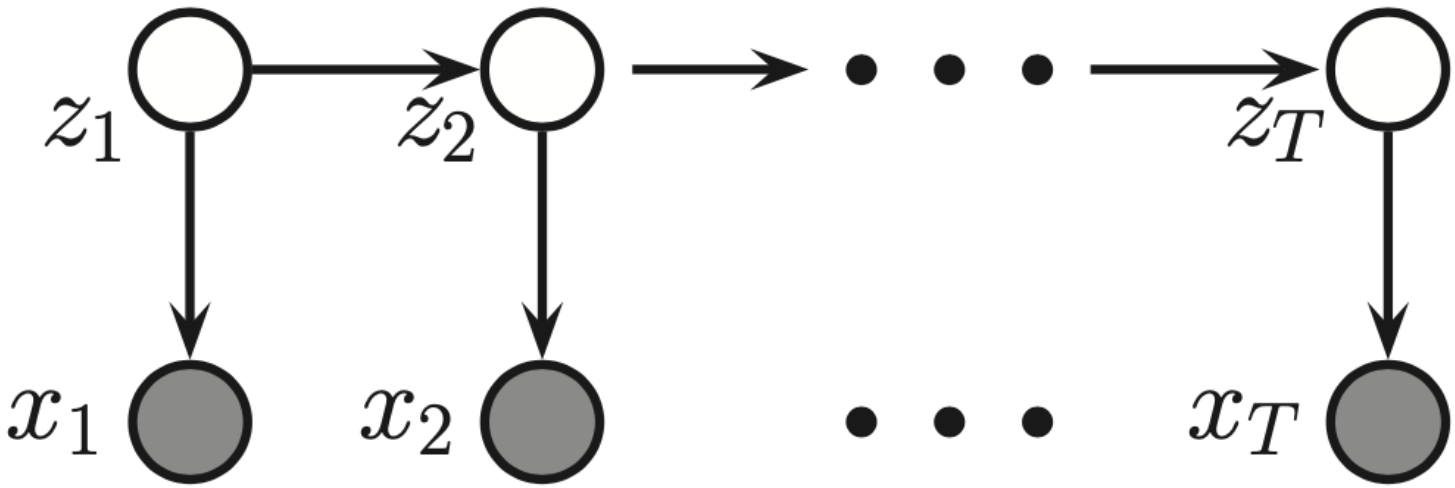
$$p(x_{1:T}) = \prod_{t=1}^{T} p(x_t|x_{t-1}, ..., x_1) \quad \text{where } p(x_1|x_0) = p(x_1).$$

- But this quickly becomes intractable for high-dimensional data -each factor requires exponentially many parameters to specify as a function of T.
- So we **made** the simplifying assumption that our data can be modeled as a **first-order Markov chain**

$$p(x_t|x_{1:t-1}) = p(x_t|x_{t-1})$$

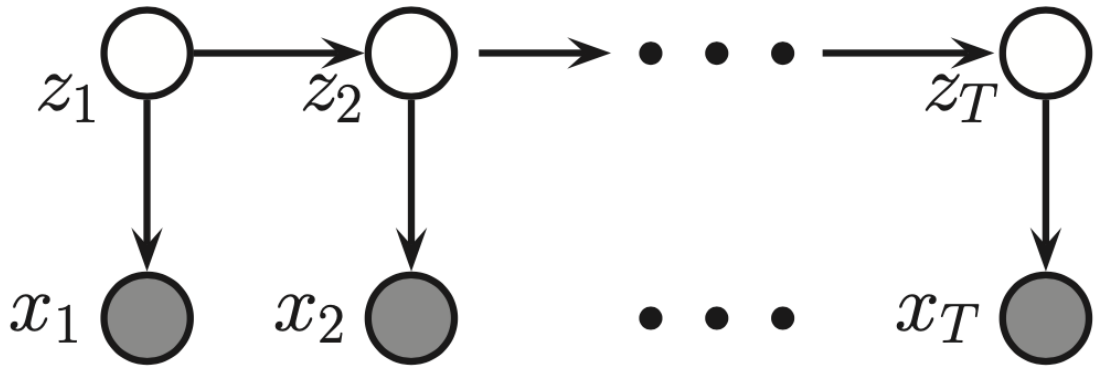
Sequential data

- In certain cases, Markov chain assumption is also restrictive.
- The state of our variables is fully observed. Hence, we introduce Hidden Markov Models



Hidden Markov Models (HMMs)

- HMMs hide the temporal dependence by keeping it in the unobserved state.
- No assumptions on the temporal dependence of observations is made.
- For each observation x_t, we associate a corresponding unobserved hidden/latent variable z_t



- The joint distribution of the model becomes

$$p(x_{1:T}, z_{1:T}) = p(z_1) \prod_{t=2}^{T} p(z_t|z_{t-1}) \prod_{t=1}^{T} p(x_t|z_t)$$

Hidden Markov Models (HMMs)

Unlike simple Markov chains, the observations are not limited by a Markov assumption of any order. Assuming we have a homogeneous model, we only have to know three sets of distributions

1. **Initial distribution**: $\pi(i) = p(z_1 = i)$. The probability of the first hidden variable being in state i (often denoted π)
2. **Transition distribution**:
   $\Psi(i, j) = p(z_{t+1} = j | z_t = i) \quad i \in \{1, ..., k\}$. The probability of moving from hidden state i to hidden state j.
3. **Emission probability**: $\psi_t(i) = p(x_t | z_t = i)$. The probability of an observed random variable x given the state of the hidden variable that "emitted" it.

HMMs: Objectives

We consider the following objectives:

1. Compute the probability of a latent sequence given an observation sequence.
   That is, we want to be able to compute $p(z_{1:t}|x_{1:t})$. This is achieved with the **Forward-Backward algorithm**.
2. Infer the most likely sequence of hidden states.
   That is, we want to be able to compute

$$z^\star = \underset{z_{1:T}}{\text{argmax}}\ p(z_{1:T}|x_{1:T}).$$

   This is achieved using the **Viterbi algorithm**.

Forward algorithm

- The goal is to recursively compute the filtered marginals,

$$\alpha_t(j) = p(z_t = j | x_{1:t})$$

  in an HMM,
- assuming that we know the initial $p(z_1)$, transition $p(z_t|z_{t-1})$, and emission $p(x_t|z_t)$ probabilities $\forall t \in [1, T]$.
- This is a step in the **forward-backward algorithm**.

Forward algorithm

The algorithm has two steps:

- First one is the prediction step, in which we compute the one-step-ahead predictive density; this acts as the new prior for time t:

$$p(z_t = j|x_{1:t-1}) = \sum_i p(z_t = j|z_{t-1} = i)p(z_{t-1} = i|x_{1:t-1})$$
$$= \sum_i \Psi(i,j)\alpha_{t-1}(i)$$

- Next one is the update step,

$$\begin{aligned}\alpha_t(j) =& p(z_t = j|x_{1:t}) = p(z_t = j|x_{1:t-1}, x_t) \\ \propto& p(x_t|z_t = j, x_{1:t-1})p(z_t = j|x_{1:t-1}) \\ \propto& p(x_t|z_t = j)p(z_t = j|x_{1:t-1}) = \psi_t(j)p(z_t = j|x_{1:t-1})\end{aligned}$$

where the normalizing constant is

$$Z_t = p(x_t|x_{1:t-1}) = \sum_j p(z_t = j|x_{1:t-1})p(x_t|z_t = j)$$

Forward algorithm

- This process is called the predict-update cycle.
- Using matrix notation, we can write the update in the following simple form:

$$\alpha_t \propto \psi_t \odot (\Psi^T \alpha_{t-1})$$

where

- $\psi_t(j) = p(x_t|z_t = j)$ is the local evidence at time t,
- $\Psi(i, j) = p(z_t = j|z_{t-1} = i)$ is the transition matrix,
- and $\odot$ is the Hadamard (entrywise) product.

Forward-Backward algorithm

- The Forward-backward algorithm is used to efficiently estimate the latent sequence given an observation sequence under a HMM.
- That is, we want to compute

$$p(z_t|x_{1:T}) \quad \forall_t \in [1, T]$$

assuming that we know the initial $p(z_1)$, transition $p(z_t|z_{t-1})$, and emission $p(x_t|z_t)$ probabilities $\forall t \in [1, T]$.

Forward-Backward algorithm

This task of hidden state inference breaks down into the following:

- **Filtering**: compute posterior over current hidden state, $p(z_t|x_{1:t})$.
- **Prediction**: compute posterior over future hidden state, $p(z_{t+k}|x_{1:t})$.
- **Smoothing**: compute posterior over past hidden state, $p(z_k|x_{1:t}) \quad 1 < k < t$.

The probability of interest, $p(z_t|x_{1:T})$ is computed using a forward and backward recursion

- **Forward Recursion**: $p(z_t|x_{1:t})$
- **Backward Recursion**: $p(x_{1+t:T}|z_t)$

Forward-Backward algorithm

We can break the chain into two parts, the past and the future, by conditioning on z_t:

- We have

$$\begin{aligned}\gamma_t = p(z_t|x_{1:T}) \propto & p(z_t, x_{1:T}) \\ = & p(z_t, x_{1:t})p(x_{t+1:T}|z_t, x_{1:t}) \\ = & p(z_t, x_{1:t})p(x_{t+1:T}|z_t) \\ \propto & (\text{Forward Recursion})(\text{Backward Recursion})\end{aligned}$$

- The third line is arrived at by noting the conditional independence $x_{t+1:T} \perp x_{1:t}|z_t$.
- We know how to perform forward recursion from the previous part.

Backward recursion

In the backward pass,

$$
\begin{aligned}
\beta_t(i) &= p(x_{t+1:T}|z_t = i) \\
&= \sum_j p(z_{t+1} = j, x_{t+1:T}|z_t = i) \\
&= \sum_j p(x_{t+2:T}|z_{t+1} = j, z_t = i, x_{t+1}) p(x_{t+1}|z_{t+1} = j, z_t = i) p(z_{t+1} = j|z_t = i) \\
&= \sum_j p(x_{t+2:T}|z_{t+1} = j) p(x_{t+1}|z_{t+1} = j) p(z_{t+1} = j|z_t = i) \\
&= \sum_j \beta_{t+1}(j) \psi_{t+1}(j) \Psi(i, j)
\end{aligned}
$$

- Notice that our backward recursion contains our emission, $\psi_{t+1} = p(x_{t+1}|z_{t+1})$ and transition, $\Psi = p(z_{t+1}|z_t)$ probabilities.

Backward recursion

- In vector notation

$$\beta_t = \Psi(\psi_{t+1} \odot \beta_{t+1})$$

  where $\beta_T(i) = 1$.

- Once we have the forward and the backward steps complete, we can compute

$$\gamma_t(i) \propto \alpha_t(i)\beta_t(i).$$

  which is called the **forward-backward algorithm**.

- Recall

$$\begin{aligned}\gamma_t = p(z_t|x_{1:T}) \propto & p(z_t, x_{1:t})p(x_{t+1:T}|z_t) \\ \propto & (\text{Forward Recursion})(\text{Backward Recursion})\end{aligned}$$

Viterbi algorithm

- The Viterbi algorithm (Viterbi 1967) is used to compute the most probable sequence.

$$\hat{z} = \arg\max_{z_{1:T}} \; p(z_{1:T}|x_{1:T})$$

- Since this is MAP inference, we might think of replacing sum-operators with max-operators, just like we did in sum-product and max-product.
- But this, in general, will lead to incorrect results.
- In Viterbi algorithm, the forward pass does use max- product, but the backwards pass uses a traceback procedure to recover the most probable path.

Viterbi algorithm

- Let's define

$$\delta_t(j) = \max_{z_1,..,z_{t-1}} p(z_{1:t-1}, z_t = j | x_{1:t})$$

  which is the probability of ending up in state j at time t, by taking the most probable path.
- We notice that

$$\begin{aligned}
\delta_t(j) &= \max_{z_1,..,z_{t-1}} p(z_{1:t-1}, z_t = j | x_{1:t}) \\
&\propto \max_{z_1,..,z_{t-1}} p(z_{1:t-2}, z_{t-1} = i | x_{1:t-1}) p(z_t = j | z_{t-1} = i) p(x_t | z_t = j) \\
&= \max_i \delta_{t-1}(i) \Psi(i,j) \psi_t(j)
\end{aligned}$$

- Let's keep track of the most likely previous state,

$$\theta_t(j) = \arg\max_i \delta_{t-1}(i) \Psi(i,j) \psi_t(j).$$

Viterbi algorithm

- Initialize the algorithm with

$$\delta_1(j) = \pi_j \psi_1(j).$$

  where $\pi_j = p(z_1 = j)$
- and terminate with

$$z_T^* = \arg\max_i \delta_T(i)$$

- Then, we compute the most probable sequence of states using traceback:

$$z_t^* = \theta_{t+1}(z_{t+1}^*)$$

Summary

- HMMs hide the temporal dependence by keeping it in the unobserved state.
- No assumptions on the temporal dependence of observations is made.
- Forward-backward algorithm can be used to find "beliefs"
- Viterbi algorithm can be used to do MAP.
- Next lecture: Variational inference.