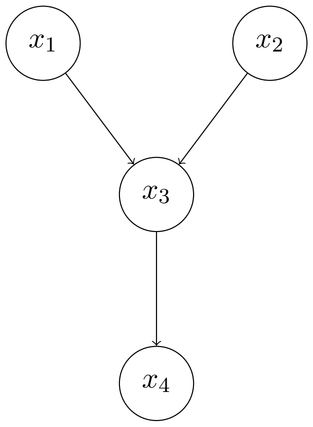
CS 181 Spring 2020 Section 9
Variable Elimination, HMMs, and Kalman Filters

1 Variable Elimination in Bayesian Networks

Recall that a Bayesian network is a graphical model that represents random variables and their dependencies using a directed acyclic graph. They allow us to efficiently model joint distributions over many variables by taking advantage of the local dependencies between variables, and they form the foundation of other models that we'll explore today.

In this section, we discuss an inference algorithm called variable elimination. Consider the Bayesian network we saw in lecture last week:



Assume that all of the random variables are Bernoulli, meaning their domain is $\{0,1\}$ with domain size $k=2$. In this network, we can encode the joint distribution as

$$p(x_1,x_2,x_3,x_4) = p(x_1)p(x_2)p(x_3|x_1,x_2)p(x_4|x_3)$$

If we wanted to calculate the marginal distribution of x_4 that is, have x_4 be our query without any evidence (conditioned on variables), we could naively marginalize out all other variables:

$$\begin{aligned}
p(x_4) &= \sum_{x_1}\sum_{x_2}\sum_{x_3} p(x_1,x_2,x_3,x_4) \\
&= \sum_{x_1}\sum_{x_2}\sum_{x_3} p(x_1)p(x_2)p(x_3|x_1,x_2)p(x_4|x_3)
\end{aligned}$$

To calculate these sums we would need to multiply two k-dimensional vectors for each of the $k^3 = 8$ possible combinations of x_1, x_2, x_3. In general, the number of combinations grows exponentially in the number of variables.

Note that Bayesian nets encode dependencies between variables, which we can use to calculate the marginal distribution more efficiently. By reordering the sums and eliminating one variable at a time, we derive the variable elimination procedure:

$$\begin{aligned}
p(x_4) &= \sum_{x_1}\sum_{x_2}\sum_{x_3} p(x_1)p(x_2)p(x_3|x_1,x_2)p(x_4|x_3) \\
&= \sum_{x_3} p(x_4|x_3) \sum_{x_2} p(x_2) \sum_{x_1} p(x_3|x_1,x_2)p(x_1) \\
&= \sum_{x_3} p(x_4|x_3) \sum_{x_2} p(x_2)p(x_3|x_2) \\
&= \sum_{x_3} p(x_4|x_3)p(x_3) \\
&= p(x_4)
\end{aligned}$$

Here, we eliminate x_1 using a k by k matrix $g_1(x_3, x_2)$, then x_2 with a K-dimensional vector $g_2(x_3)$, and lastly x_3, which results in a final K-dimensional vector of probabilities for x_4. Notice that we have a poly-tree, and we're eliminating leaves first and working towards our query variable, x_4.

Alternatively, we could have eliminated variables in a different order:

$$\begin{aligned}
p(x_4) &= \sum_{x_1}\sum_{x_2}\sum_{x_3} p(x_1)p(x_2)p(x_3|x_1,x_2)p(x_4|x_3) \\
&= \sum_{x_1} p(x_1) \sum_{x_2} p(x_2) \sum_{x_3} p(x_3|x_1,x_2)p(x_4|x_3) \\
&= \sum_{x_1} p(x_1) \sum_{x_2} p(x_2)p(x_4|x_1,x_2) \\
&= \sum_{x_1} p(x_1)p(x_4|x_1) \\
&= p(x_4)
\end{aligned}$$

Here, we eliminate x_3, then x_2, then x_1. Notice that the ordering matters: eliminating x_3 first results in a $kxkxk$ object $g(x_1, x_2, x_4)$.

In general, the computational cost of variable elimination depends on the number of variables in these intermediate factors, in particular the largest object computed ("tree-width").

1.1 Exercise: Variable Elimination

Consider the Bayesian network described in above, and assume the following Conditional Probability Table (CPT). Let $x_i \in \{0,1\}$ denote the values that variable X_i can take. Our goal is to find $p(x_4)$.

x_1	$p(x_1)$
0	0.3
1	0.7

x_2	$p(x_2)$
0	0.6
1	0.4

x_3	x_1	x_2	$p(x_3 \vert x_1, x_2)$
0	0	0	0.5
0	0	1	0.2
0	1	0	0.9
0	1	1	0.5
1	0	0	0.5
1	0	1	0.8
1	1	0	0.1
1	1	1	0.5

x_4	x_3	$p(x_4 \vert x_3)$
0	0	0.7
0	1	0.1
1	0	0.3
1	1	0.9

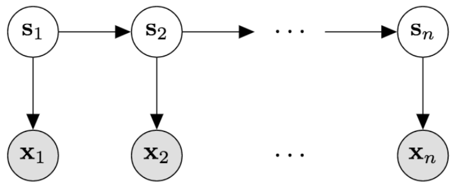
1. Eliminate X_1 first. Draw the resulting Bayesian network and compute the CPT.
2. Eliminate X_3 first. Draw the resulting Bayesian network and compute the CPT.
3. How many sum-product calculations do each of these variable elimination orders require? Which one is preferable?

Solution:

2 Hidden Markov Models

A Hidden Markov Model (HMM) is useful for inferring a sequence of unknown or hidden states from a corresponding sequence of observed evidence.

2.1 Graphical Model



Consider a sequence of one-hot encoded states $\mathbf{s}_1$,...,$\mathbf{s}_n$ where $\mathbf{s}_t \in \{S_k\}_{k=1}^{c}$, and a corresponding sequence of observations ($\mathbf{x}_1$,...,$\mathbf{x}_n$) where $\mathbf{x}_t \in \{O_j\}_{j=1}^{m}$. Each state can be one of c possible states, and each observation can be one of m possible observations. Note that N is the number of data points (each of which is a sequence), where n is the length of a sequence (assume all sequences are the same length).

2.2 Model Assumptions

HMMs are characterized by and allow us to reason about the following joint distribution

$$p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n, \mathbf{x}_1, \ldots, \mathbf{x}_n) = p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n) p(\mathbf{x}_1, \ldots, \mathbf{x}_n \mid \boldsymbol{s}_1, \ldots, \boldsymbol{s}_n)$$

However, it's not immediately obvious how we should optimize this model, and the following assumptions make this easier:

- The future hidden state is independent of past hidden states given the present (Markov Property):

  $p(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_1, \ldots \boldsymbol{s}_t, \mathbf{x}_1, \ldots, \mathbf{x}_t) = p(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t)$

- Observations only depend on the present hidden state:

  $p(\mathbf{x}_t \mid \boldsymbol{s}_1, \ldots, \boldsymbol{s}_t, \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) = p(\mathbf{x}_t \mid \boldsymbol{s}_t)$

Notice that the above assumptions allow us to factor the joint as follows:

$$\begin{aligned} p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n, \mathbf{x}_1, \ldots, \mathbf{x}_n) &= \\ p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n) p(\mathbf{x}_1, \ldots, \mathbf{x}_n \mid \boldsymbol{s}_1, \ldots, \boldsymbol{s}_n) &= \\ p(\boldsymbol{s}_1) \prod_{t=1}^{n-1} p(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t) \prod_{t=1}^{n} p(\mathbf{x}_t \mid \boldsymbol{s}_t) & \end{aligned}$$

2.3 Exercise: When to Use HMMs (Source: CMU)

For each of the following scenarios, is it appropriate to use a Hidden Markov Model? Why or why not? What would the observed data be in each case, and what would the hidden states capture?

1. Stock market price data
2. Recommendations on a database of movie reviews
3. Daily precipitation data in Boston
4. Optical character recognition for identifying words

Solution:

2.4 Parameterization

- $\boldsymbol{\theta} \in \mathbb{R}^c$: defines the prior distribution over initial hidden states
- $\mathbf{T} \in \mathbb{R}^{c \times c}$: transition matrix where T_{kj} is the probability of transitioning from S_k to S_j
- $\{\boldsymbol{\pi}\}_{k=1}^{c}$: conditional probabilities of observations given hidden states such that $p(\mathbf{x}_t = O_j | \mathbf{s}_t = S_k; \{\boldsymbol{\pi}\}) = \pi_{kj}$. $\forall k \; \boldsymbol{\pi}_k \in \mathbb{R}^m$.

First, we need to estimate the parameters from the data, which we can do with a variant of EM. Then, with our trained HMM, we are able to perform several inference tasks on our data.

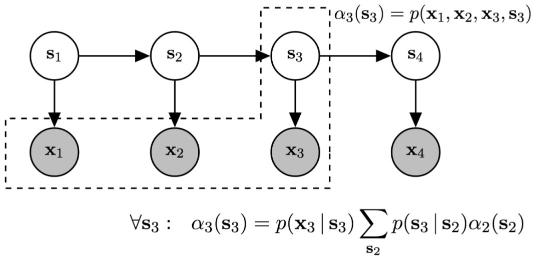
2.5 EM for HMMs

Given data points $\{\mathbf{x}^i\}_{i=1}^N$ defined by sequences $(x_1^i, \dots, x_n^i)$ of length n represented as row vectors, we want to infer the parameters $\{\mathbf{T}, \boldsymbol{\theta}, \{\boldsymbol{\pi}_k\}\}$. Had we been given the true states, we could easily compute joint probability $p(\mathbf{x}^i, \mathbf{s}^i)$ and write the complete-data log likelihood, and maximize with respect to the parameters. Instead, we need to estimate state distributions and parameters iteratively.

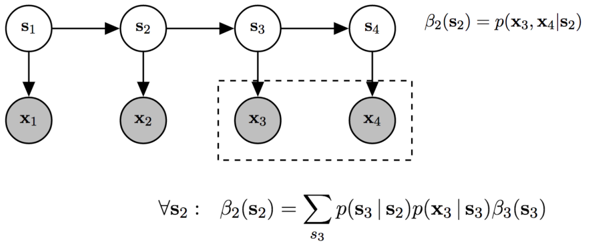
2.5.1 Forward-Backward Algorithm

The HMM model is characterized by the joint distribution $p(\boldsymbol{s}_1, \dots, \boldsymbol{s}_n, \mathbf{x}_1, \dots, \mathbf{x}_n)$, which means that many of our training and inference tasks require marginalization to obtain conditionals. Thus, naive algorithms can be expensive (they require lots of nested summations over states), and we use EM instead. We define the recurrence relations $\alpha_t(\boldsymbol{s}_t)$ and $\beta_t(\boldsymbol{s}_t)$ in the E-Step:

- $\alpha_t(\mathbf{s}_t)$ represents the joint probability of observations $1, \dots, t$ and state t. α_t can be defined in terms of α_{t-1}. We moove **forwards** through the sequence to calculate the α's
- $\beta_t(\mathbf{s}_t)$ represents the joint probability of observations $t+1, \dots, n$ conditioned on state t. β_t can be defined in terms of β_{t+1}. We move **backwards** through the sequence to calculate the β's.


$\alpha_3(\mathbf{s}_3) = p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{s}_3)$
$\mathbf{s}_1$
$\mathbf{s}_2$
$\mathbf{s}_3$
$\mathbf{s}_4$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
$\forall \mathbf{s}_3 : \quad \alpha_3(\mathbf{s}_3) = p(\mathbf{x}_3 \mid \mathbf{s}_3) \sum_{\mathbf{s}_2} p(\mathbf{s}_3 \mid \mathbf{s}_2) \alpha_2(\mathbf{s}_2)$

(a) alpha


$\beta_2(\mathbf{s}_2) = p(\mathbf{x}_3, \mathbf{x}_4 | \mathbf{s}_2)$
$\mathbf{s}_1$
$\mathbf{s}_2$
$\mathbf{s}_3$
$\mathbf{s}_4$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
$\forall \mathbf{s}_2 : \quad \beta_2(\mathbf{s}_2) = \sum_{s_3} p(\mathbf{s}_3 \mid \mathbf{s}_2) p(\mathbf{x}_3 \mid \mathbf{s}_3) \beta_3(\mathbf{s}_3)$

(b) beta

Note that the probabilities we use for calculating α and β are given by the parameters that we fix in the E-Step.

$$\forall \mathbf{s}_t : \quad \alpha_t(\mathbf{s}_t) = \begin{cases} p(\mathbf{x}_t \mid \mathbf{s}_t) \sum_{\mathbf{s}_{t-1}} p(\mathbf{s}_t \mid \mathbf{s}_{t-1}) \alpha_{t-1}(\mathbf{s}_{t-1}) & \text{if } 1 < t \leq n \\ p(\mathbf{x}_1 \mid \mathbf{s}_1) p(\mathbf{s}_1) & \text{o.w.} \end{cases}$$

$$\forall \mathbf{s}_t : \quad \beta_t(\mathbf{s}_t) = \begin{cases} \sum_{\mathbf{s}_{t+1}} p(\mathbf{s}_{t+1} \mid \mathbf{s}_t) p(\mathbf{x}_{t+1} \mid \mathbf{s}_{t+1}) \beta_{t+1}(\mathbf{s}_{t+1}) & \text{if } 1 \leq t < n \\ 1 & \text{o.w.} \end{cases}$$

2.5.2 Inference Patterns with $\boldsymbol{\alpha}, \boldsymbol{\beta}$

The following patterns are useful for inference with a trained HMM as well as during the E-Step:

- $\alpha_t(\mathbf{s}_t)\beta_t(\mathbf{s}_t) = p(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{s}_t) \propto p(\mathbf{s}_t | \mathbf{x}_1, \dots, \mathbf{x}_n)$
- joint of observations: $p(\mathbf{x}_1, \dots, \mathbf{x}_n) = \sum_{\mathbf{s}_t} \alpha_t(\mathbf{s}_t)\beta_t(\mathbf{s}_t)$ (for any t)
- smoothing: $p(\mathbf{s}_t \mid \mathbf{x}_1, \dots, \mathbf{x}_n) \propto p(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{s}_t) = \alpha_t(\mathbf{s}_t)\beta_t(\mathbf{s}_t)$
- prediction: $p(\mathbf{x}_{n+1} \mid \mathbf{x}_1, \dots, \mathbf{x}_n) \propto \sum_{\mathbf{s}_n, \mathbf{s}_{n+1}} \alpha_n(\mathbf{s}_n) p(\mathbf{s}_{n+1} \mid \mathbf{s}_n) p(\mathbf{x}_{n+1} \mid \mathbf{s}_{n+1})$
- transition: $p(\mathbf{s}_t, \mathbf{s}_{t+1} \mid \mathbf{x}_1, \dots, \mathbf{x}_n) \propto \alpha_t(\mathbf{s}_t) p(\mathbf{s}_{t+1} \mid \mathbf{s}_t) p(\mathbf{x}_{t+1} \mid \mathbf{s}_{t+1}) \beta_{t+1}(\mathbf{s}_{t+1})$

2.5.3 E-Step

The goal of the expectation step is to compute the expected values of the hidden states given a fixed set of parameters $\mathbf{w} = \{\mathbf{T}, \boldsymbol{\theta}, \{\boldsymbol{\pi}_k\}\}$. That is, we estimate the state distribution for $\mathbf{s}_1^i, \ldots, \mathbf{s}_n^i$ given $\mathbf{x}^i$.

Let the $c \times 1$ vector $\mathbf{q}_t^i = (q_{t1}^i, \ldots, q_{tc}^i)$ represent $\mathbf{x}^i$'s distribution over states for time t under the current parameters. Let $\mathbf{Q}_{t,t+1}^i$ be the $c \times c$ matrix of transition probabilities under the current parameters. Then

- α's and β's are defined in terms of fixed parameters.
- $\mathbf{q}$'s are defined in terms of α's and β's
- Calculate $q_{tk}^i = p(\mathbf{s}_t^i = S_k | \mathbf{x}^i; \mathbf{w})$ for all t and k (use smoothing eq. just above)
- Calculate $q_{t,t+1,k,\ell}^i = p(\mathbf{s}_t^i = S_k, \mathbf{s}_{t+1}^i = S_\ell | \mathbf{x}^i; \mathbf{w})$ (use transition eq. just above)

2.5.4 M-Step

Now we need to update our parameters to maximize the expected complete-data log likelihood $\mathbb{E}_{\mathbf{S}}[\ln p(\mathbf{x}, \mathbf{S}; \mathbf{w})]$. Applying the appropriate Lagrange multipliers and maximizing with respect to each of the parameters of interest, we recover the following update equations:

$$\hat{N}_{1k} = \sum_{i=1}^{N} q_{1k}^i \text{ (first period)} \quad \text{and more generally} \quad \hat{N}_k = \sum_{i=1}^{N} \sum_{t=1}^{n} q_{tk}^i \text{ (all periods)}$$

$$\hat{N}_{-nk} = \sum_{i=1}^{N} \sum_{t=1}^{n-1} q_{tk}^i \text{ (without last period)}$$

$$\hat{N}_{k\ell} = \sum_{i=1}^{N} \sum_{t=1}^{n-1} q_{t,t+1,k,\ell}^i \text{ (transitions)}$$

$$\hat{N}_{kj} = \sum_{i=1}^{N} \sum_{t=1}^{n} q_{tk}^i x_{tj}^i \text{ (observations)}$$

$$\hat{\theta}_k = \frac{\hat{N}_{1k}}{N} \quad \hat{\pi}_{kj} = \frac{\hat{N}_{kj}}{\hat{N}_k} \quad \hat{t}_{k\ell} = \frac{\hat{N}_{k\ell}}{\hat{N}_{-nk}}$$

2.6 Exercise: Parameter Estimation in Supervised HMMs

You are trying to predict the weather using an HMM. The hidden states are the weather of the day, which may be sunny or rainy, and the observable states are the color of the clouds, which can be white or gray. You have data on the weather and clouds from one sequence of four days (note: the hidden states are observed here):

Day	Weather	Clouds
1	Sunny	White
2	Rainy	Gray
3	Rainy	Gray
4	Sunny	Gray

1. Draw a graphical model representing the HMM.
2. Give the values of N, n, c and of the one-hot vectors $\mathbf{s}_1^1, \ldots, \mathbf{s}_4^1, \mathbf{x}_1^1, \ldots, \mathbf{x}_4^1$.
3. Estimate and interpret the values of the parameters $\boldsymbol{\theta}, \mathbf{T}, \{\boldsymbol{\pi}_k\}_{k=1}^c$ using the MLE estimators for the supervised HMM:

$$\hat{\theta}_k = \frac{N_{1k}}{N}, \quad \hat{t}_{kl} = \frac{N_{kl}}{N_{-nk}}, \quad \hat{\pi}_{kj} = \frac{N_{kj}}{N_k}$$

$$N_k = \sum_{i=1}^{N}\sum_{t=1}^{n} s_{tk}^i, \quad N_{1k} = \sum_{i=1}^{N} s_{1,k}^i, \quad N_{-nk} = \sum_{i=1}^{N}\sum_{t=1}^{n-1} s_{tk}^i$$

$$N_{kl} = \sum_{i=1}^{N}\sum_{t=1}^{n-1} s_{t,k}^i s_{t+1,l}^i, \quad N_{kj} = \sum_{i=1}^{N}\sum_{t=1}^{n} s_{tk}^i x_{tj}^i$$

Solution:

2.7 Exercise: EM for HMMs

You are trying to model a toy's state using an HMM. At each time step, the toy can be active (state 1) or inactive (state 2), but you can only observe the color of the indicator light, which can be red (observation state 1) or green (observation state 2). You have collected data from one sequence:

Time	Light
1	Green
2	Red
3	Green

You initialize your EM with $\boldsymbol{\theta} = [\frac{1}{2}\ \frac{1}{2}]^\top, \mathbf{T} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix}, \boldsymbol{\pi}_1 = [\frac{1}{4}\ \frac{3}{4}]^\top, \boldsymbol{\pi}_2 = [\frac{3}{4}\ \frac{1}{4}]^\top$.

1. Compute $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3$ for the forward-backward algorithm using the initial parameter values.
2. How is $\mathbf{q}_t^1$ defined? Compute the values of $\mathbf{q}_1^1, \mathbf{q}_2^1$ using the α and β values.
3. How is $\mathbf{Q}_{t,t+1}^1$ defined? Compute the value of $\mathbf{Q}_{1,2}^1$ using the α and β values.

During EM, at one point you obtain the following values after the E step:

$$\mathbf{q}_1^1 = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \end{bmatrix}^\top, \quad \mathbf{q}_2^1 = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \end{bmatrix}^\top, \quad \mathbf{q}_3^1 = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \end{bmatrix}^\top$$

$$\mathbf{Q}_{1,2}^1 = \begin{bmatrix} \frac{1}{6} & \frac{1}{2} \\ \frac{1}{6} & \frac{1}{6} \end{bmatrix}, \quad \mathbf{Q}_{2,3}^1 = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} \end{bmatrix}$$

1. Use the above values to compute $\hat{N}_k, \hat{N}_{kl}, \hat{N}_{kj}$.
2. Complete the M step by updating the parameters $\boldsymbol{\theta}, \mathbf{T}, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2$.

Solution:

3 Kalman Filters

Now consider the following dynamical system model:

$$z_{t+1} = \Phi z_t + \epsilon_t$$

$$x_t = A z_t + \gamma_t$$

where z are the hidden variables and x are the observed measurements. Φ and A are known constants, while ϵ and γ are random variables drawn from the following normal distributions:

$$\epsilon_t \sim \mathcal{N}(\mu_\epsilon, \sigma_\epsilon^2)$$

$$\gamma_t \sim \mathcal{N}(\mu_\gamma, \sigma_\gamma^2)$$

This is called a (one-dimensional) linear Gaussian state-space model. It is closely related to an HMM – try drawing out the graphical model! – but here the hidden states and the observations are now continuous and normally distributed. Linear Gaussian state-space models have convenient mathematical properties and can be used to describe noisy measurements of a moving object (e.g. missiles, rodents, hands), market fluctuations, etc.

The Kalman filter is an algorithm to perform filtering in linear Gaussian state-space models, i.e. to find the distribution of z_t given observations $x_1, ..., x_t$. The distribution of $z_t \mid x_1, ..., x_s$ will be $\mathcal{N}(\mu_{t|s}, \sigma^2_{t|s})$. If we start with $\mu_{t-1|t-1}$ and $\sigma^2_{t-1|t-1}$, the algorithm tells us to

1. Define the distribution of $z_t \mid x_1, ..., x_{t-1}$ by computing $\mu_{t|t-1}$ and $\sigma^2_{t|t-1}$. This is called the prediction step.
2. Define the distribution of $z_t \mid x_1, ..., x_t$ by computing $\mu_{t|t}$ and $\sigma^2_{t|t}$. This is called the update step.

The Kalman filter alternates between prediction and update steps, assimilating observations one at a time. It requires one forward pass through the data, and is analogous to obtaining the α's in an HMM.