19 Approximate Inference: Importance Sampling, Particle Filters

Thus far, we have focused on developing inference algorithms, exact and approximate, for discrete graphical models or Gaussian graphical model. The exact method involved Elimination algorithm and Junction Tree for generic graph while Belief Propagation (BP) (sum-product/max-product) for Tree graphical model. The approximate inference method involved variational approximation, graph partitioning and Markov Chain Monte Carlo (MCMC) method. However, we did not discuss the scenario where variables are continuous, and not necessarily Gaussian. Let us understand difficulty involved in such a scenario.

Why approximate? Consider the scenario of a continuous valued Hidden Markov Model (HMM). Recall that HMM is a tree-structured and hence from graphical structure perspective, BP is exact. Therefore, it is not clear what is the need for approximation. To that end, recall that when the variables were jointly Gaussian, BP can be easily implemented because

(a) the messages in BP were Gaussian,

(b) the marginals were Gaussian,

(c) and we did not need to directly evaluate integrals, instead, we propagated means and covariance parameters.

One of the difficulties with applying BP to general distributions is that we have to pass distributions as messages, but describing a distribution can be quite complex, which means that sending a message can be expensive. In the Gaussian case, we avoided this difficulty because Gaussians are simply parametrized by a mean and covariance. The other key was that the messages turned out to be Gaussian, so everything stayed in the same family of distributions. Unfortunately, this does not happen for arbitrary continuous distributions.

Gaussian models are widely used and powerful, but may not always the right choice. Therefore, it warrants us considering the following general continuous HMM[1]:

$$\begin{aligned} x_0 &\sim p_{0|-1}(\cdot) : \text{initialization} \\ x_{n+1}|x_n &\sim p_{x_{n+1}|x_n}(\cdot) : \text{dynamics} \\ y_n|x_n &\sim p_{y_n|x_n}(\cdot) : \text{observations} \end{aligned}$$

[1]For exposition, we'll assume that all of the variables are scalars, but everything explained here naturally extends to vector valued variables.

We're interested in the marginal of x_n after observing $y_0, \ldots, y_n$ (i.e. $p_{n|n}(x_n|y_0, \ldots, y_n)$) also called the *posterior marginal*. Recall that we can calculate it exactly via BP in an iterative manner in two steps:

Prediction: Transitioning to a new state

$$p_{n+1|n}(x_{n+1}|y_0, \ldots, y_n) = \int_{x_n} p_{x_{n+1}|x_n}(x_{n+1}|x_n) \overbrace{p_{n|n}(x_n|y_0, \ldots, y_n)}^{\text{previous iteration}} \, dx_n$$

Update: Folding in a new observation

$$p_{n+1|n+1}(x_{n+1}|y_0, \ldots, y_{n+1}) = \frac{1}{Z} p_{y_{n+1}|x_{n+1}}(y_{n+1}|x_{n+1}) p_{n+1|n}(x_{n+1}|y_0, \ldots, y_n)$$

where Z is the partition function which is an implicit integral.

We use the shorthand notation $p_{n|m}$ to denote $p_{x_n|y_0,\ldots,y_m}(x_n|y_0, \ldots, y_m)$. In practice, it is very hard to use these steps *exactly*, because both of the steps involve calculating an integral. In the Gaussian case, these steps reduced to simple linear algebra, but in general calculating the integrals for arbitrary distributions is intractable, so we use approximations instead and this leads to *approximate inference*.

This is precisely the type of approximation that we discuss in this lecture. We shall do so in the context of a tree graph structure offered by HMM. The method is called *particle filtering* and can be seen as *sequential MCMC* building upon *importance sampling*. This lecture develops method of *particle filtering* for HMM. It should be noted that an adaptation of MCMC (using appropriate Metropolis Hasting rule for continuous setting) can be used for approximate inference as well in such a scenario. However, *particle filtering* allows exploitation of graphical structure like HMM for efficiency.

Remarks. It should be noted that there are other approaches known in the literature. For example, we can modify the Kalman Filter algorithm that we derived earlier to handle *nonlinear dynamical systems* (NLDS) by linearizing the state space equations about the current state estimate; in this case, it is called the *Extended Kalman Filter (EKF)*. The EKF has a limited scope of applicability, but when it can be applied, it is very useful. In fact, the EKF was used by the Apollo missions. Unfortunately, it is hard to analyze theoretically. We shall not discuss EKF here.

19.1 Particle Filter for HMMs with General Continuous Distribution

Consider an HMM with general continuous state and observations distributions as depicted in Figure 1: we want to estimate hidden states $x_0, \ldots, x_n$ given associated observations $y_0, \ldots, y_n$. Concretely, find $p_{x_i|y_0,\ldots,y_n}(x_k|y_0, \ldots, y_n)$, $0 \leq i \leq n$. To that end, let us gather what we know.

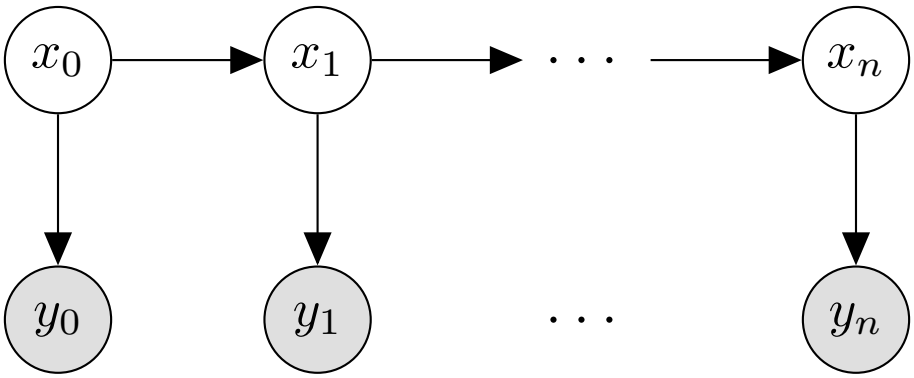

Figure 1: Example of an HMM.

We can sample from $p_{\mathsf{x}_i}(\cdot)$ by sampling from $p_{\mathsf{x}_0,\ldots,\mathsf{x}_n}(\cdot)$, $0 \le i \le n$. Since $p_{\mathsf{x}_i}(\cdot)$ is the marginal of $p_{\mathsf{x}_0,\ldots,\mathsf{x}_n}(\cdot)$ for $0 \le i \le n$, simply extracting the i^{th} coordinate of samples from $p_{\mathsf{x}_0,\ldots,\mathsf{x}_n}(\cdot)$ is the same as sampling from $p_{\mathsf{x}_i}(\cdot)$. In other words, suppose we take S samples from $p_{\mathsf{x}_0,\ldots,\mathsf{x}_n}(\cdot)$,

$$\{(x_0(s), \ldots, x_n(s)\}_{s=1}^S. \tag{1}$$

Then extracting the i^{th} coordinate,

$$\{x_i(s)\}_{s=1}^S \tag{2}$$

is a set of S samples from $p_{\mathsf{x}_i}(\cdot)$. Likewise, suppose we have a set of observations $y_0, \ldots, y_n$; then extracting the i^{th} coordinate from samples from $p_{\mathsf{x}_0,\ldots,\mathsf{x}_n|\mathsf{y}_0,\ldots,\mathsf{y}_n}(\cdot|y_0, \ldots, y_n)$ yields samples from $p_{\mathsf{x}_i|\mathsf{y}_0,\ldots,\mathsf{y}_n}(\cdot|y_0, \ldots, y_n)$.

We can sample from $p_{\mathsf{x}_0,\ldots,\mathsf{x}_n}(\cdot)$ by using the Markov property. Note that by the Markov property of our model, x_i is conditionally independent of $\mathsf{x}_0, \ldots, \mathsf{x}_{i-2}$ given x_{i-1}. Therefore, in order to generate s^{th} sample from $p_{\mathsf{x}_0,\ldots,\mathsf{x}_n}(\cdot)$, we first sample

$$x_0(s) \sim p_{\mathsf{x}_0}(\cdot), \tag{3}$$

and then for all $i = 1, \ldots, n$, we sample

$$x_i(s) \sim p_{\mathsf{x}_i|\mathsf{x}_{i-1}}(\cdot|x_{i-1}(s)). \tag{4}$$

We can approximate expectations with samples. Suppose we wish to compute an expectation of some function $f(\cdot)$ with respect to (marginal) distribution of x_i, $\mathbb{E}_{p_{\mathsf{x}_i}}[f(\mathsf{x}_i)]$. Given a set of S samples $\{x_i(s)\}_{s=1}^S$ from $p_{\mathsf{x}_i}(\cdot)$, by the strong law of large numbers, we can approximate the expectation as

$$\frac{1}{S}\sum_{s=1}^{S} f(x_i(s)) \to \mathbb{E}_{p_{\mathsf{x}_i}}[f(\mathsf{x}_n)] \text{ as } S \to \infty. \tag{5}$$

This approximation is valid for a large class of functions, including all bounded measurable functions. For example, if f is the indicator function $f(x) = \mathbb{1}[x \in A]$ for a set A, this can be used to compute $p_{\mathsf{x}_i}(\mathsf{x}_i \in A)$:

$$\frac{1}{S}\sum_{s=1}^{S} \mathbb{1}[x_i(s) \in A] \to \mathbb{E}_{p_{\mathsf{x}_i}}[\mathbb{1}[\mathsf{x}_i \in A]] = p_{\mathsf{x}_i}(\mathsf{x}_i \in A) \text{ as } S \to \infty. \tag{6}$$

However, we can't sample directly from $p_{\mathsf{x}_0,\ldots,\mathsf{x}_n|\mathsf{y}_0,\ldots,\mathsf{y}_n}(\cdot|y_0,\ldots,y_n)$. Of course, we aren't very interested in the prior $p_{\mathsf{x}_0,\ldots,\mathsf{x}_n}(\cdot)$; we have observations $y_0,\ldots,y_n$ from the HMM and want to incorporate this information by using the posterior distribution $p_{\mathsf{x}_0,\ldots,\mathsf{x}_n|\mathsf{y}_0,\ldots,\mathsf{y}_n}(\cdot|y_0,\ldots,y_n)$. By the rules of conditional probability, the posterior given observations $y_0,\ldots,y_n$ may be expressed as (using notation, $v_0^n = (v_0,\ldots,v_n)$ for any vector v)[2]

$$p_{\mathsf{x}_0,\ldots,\mathsf{x}_n|\mathsf{y}_0,\ldots,\mathsf{y}_n}(x_0^n|y_0^n) = \frac{p_{\mathsf{y}_0,\ldots,\mathsf{y}_n|\mathsf{x}_0,\ldots,\mathsf{x}_n}(y_0^n|x_0^n)p_{\mathsf{x}_0,\ldots,\mathsf{x}_n}(x_0^n)}{p_{\mathsf{y}_0,\ldots,\mathsf{y}_n}(y_0^n)}.$$

Given the model description, we know $p_{\mathsf{x}_0,\ldots,\mathsf{x}_n}$ as well as $p_{\mathsf{y}_0,\ldots,\mathsf{y}_n|\mathsf{x}_0,\ldots,\mathsf{x}_n}$, both of which factorize nicely. However, we may not know $p_{\mathsf{y}_0,\ldots,\mathsf{y}_n}(\cdot)$. Therefore, given observation y_0^n, the conditional density of $\mathsf{x}_0,\ldots,\mathsf{x}_n$ takes form

$$p_{\mathsf{x}_0,\ldots,\mathsf{x}_n|\mathsf{y}_0,\ldots,\mathsf{y}_n}(x_0^n|y_0^n) = \frac{1}{Z}\big(p_{\mathsf{x}_0,\ldots,\mathsf{x}_n}(x_0^n)\big)\big(\prod_{i=0}^{n} p_{\mathsf{y}_i|\mathsf{x}_i}(y_i|x_i)\big). \tag{7}$$

In above, Z represents unknown 'normalization constant'. Therefore, the challenge is, can we sample from a distribution that has an unknown normalization constant? Importance sampling will rescue us from this challenge as explained next.

19.2 Importance Sampling

Let us re-state the challenge abstractly: we are given a distribution with density μ such that

$$\mu(x) = \frac{q(x)}{Z}, \tag{8}$$

where $q(x)$ is a known function and Z is unknown normalization constant. We want to sample from μ so as to estimate

$$\mathbb{E}_{\mu}[f(x)] = \int f(x)\mu(x)\mathrm{d}x. \tag{9}$$

We can sample from a known distribution with density ν, potentially very different from μ. Importance sampling provides means to solves this problem:

[2]Here, we assume that we have well-defined densities to be able to write conditional densities as expressed.

- Sample $x(1), x(2), \ldots, x(S)$ from known distribution with density ν.
- Compute $w(1), \ldots, w(S)$ as

$$w(s) = \frac{q(x(s))}{\nu(x(s))}. \tag{10}$$

- Output estimation for $\mathbb{E}_\mu[f(x)]$ as

$$\hat{E}(S) = \frac{\sum_{s=1}^{S} w(s) f(x(s))}{\sum_{s=1}^{S} w(s)}. \tag{11}$$

Next we provide justification for this method: as $S \to \infty$, the estimation $\hat{E}(S)$ converges to the desired value $\mathbb{E}_\mu[f(x)]$. To that end, we need following definition: let support of distribution with density p be

$$\mathsf{supp}(p) = \{x : p(x) > 0\}. \tag{12}$$

Theorem 1. *Let* $\mathsf{supp}(\mu) \subseteq \mathsf{supp}(\nu)$. *Then as* $S \to \infty$,

$$\hat{E}(S) \to \mathbb{E}_\mu[f(x)], \qquad \textit{with probability } 1. \tag{13}$$

Proof. Using strong law of large numbers, it follows that with probability 1,

$$\begin{aligned}
\frac{1}{S}\sum_{s=1}^{S} w(s) f(x(s)) &\to \mathbb{E}_\nu\left[\frac{q(x)}{\nu(x)} f(x)\right] \\
&= \int_{\mathsf{supp}(\nu)} \frac{q(x)}{\nu(x)} f(x)\nu(x)\, \mathrm{d}x \\
&= \int_{\mathsf{supp}(\mu)} Z\mu(x) f(x)\, \mathrm{d}x \\
&= Z\mathbb{E}_\mu[f(x)],
\end{aligned} \tag{14}$$

where we have used the fact that $q(x) = 0$ for $x \notin \mathsf{supp}(\mu)$. Using similar argument, it follows that with probability 1,

$$\frac{1}{S}\sum_{s=1}^{S} w(s) \to Z\mathbb{E}_\mu[1] = Z. \tag{15}$$

This leads to the desired conclusion. □

Remarks. It is worth noting that as long as $\mathsf{supp}(\mu)$ is contained in $\mathsf{supp}(\nu)$, the estimation converges irrespective of choice of ν. This is quite remarkable. Alas, it comes at a cost. The choice of ν determines the variance of the estimator and hence the number of samples S required to obtain good estimation.

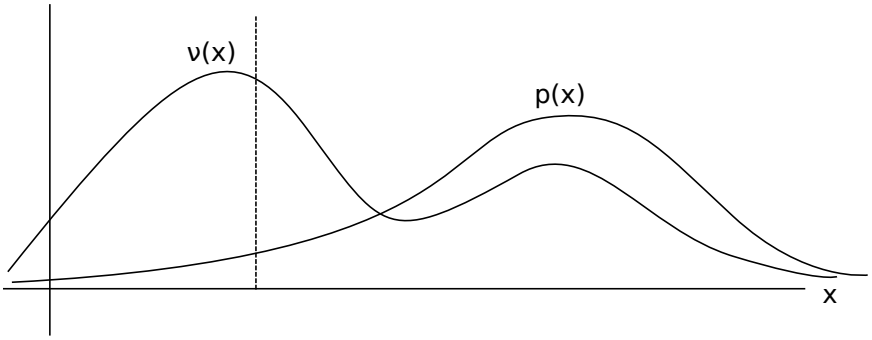

Figure 2: Depiction of importance sampling. $\nu(x)$ is the proposal distribution and $\mu(x)$ is the distribution we're trying to sample from. On average we will have to draw many samples from ν to get a sample where ν has low probability, but as we can see in this example, that may be where μ has most of its probability mass.

19.3 Particle Filtering: HMM

Now we shall utilize Importance Sampling for obtaining samples from distribution $p_{x_0,\ldots,x_n|y_0,\ldots,y_n}(x_0^n|y_0^n)$, which in above notation corresponds to μ. Therefore, the function q is given by

$$q(x_0^n) = p_{x_0,\ldots,x_n}(x_0^n)\big(\prod_{i=0}^{n} p_{y_i|x_i}(y_i|x_i)\big). \tag{16}$$

The known distribution that we shall utilize to estimate the unknown distribution is none other than $p_{x_0,\ldots,x_n}(\cdot)$, which in above notation is ν. Therefore, for any given x_0^n, the weight function w is given by

$$\begin{aligned} w_0^n \equiv w(x_0^n) &= \frac{q(x_0^n)}{\nu(x_0^n)} \\ &= \prod_{i=0}^{n} p_{y_i|x_i}(y_i|x_i). \end{aligned} \tag{17}$$

In summary, we estimate $\mathbb{E}_{p_{x_0,\ldots,x_n|y_0,\ldots,y_n}}[f(x_0,\ldots,x_n)|y_0,\ldots,y_n]$ as follows.

1. Obtain S i.i.d. samples, $x_0^n(s)$, $1 \le s \le S$, as per $p_{x_0,\ldots,x_n}$.
2. Compute corresponding weights $w_0^n(s) = \prod_{i=0}^{n} p(y_i|x_i(s)), \quad 1 \le s \le S$.
3. Output estimation of $\mathbb{E}_{p_{x_0,\ldots,x_n|y_0,\ldots,y_n}}[f(x_0,\ldots,x_n)|y_0,\ldots,y_n]$ as

$$\frac{\sum_{s=1}^{S} w_0^n(s) f(x_0^n(s))}{\sum_{s=1}^{S} w_0^n(s)}. \tag{18}$$

Sequential implementation. It is worth observing that the above described algorithm can be sequentially implemented. Specifically, given the samples $x_0^n(s)$ and corresponding weight function $w_0^n(s)$, we can obtain $x_0^{n+1}(s)$ and $w_0^{n+1}(s)$ as follows:

- Sample $x_{n+1}(s) \sim p_{x_{n+1}|x_n}(\cdot|x_n(s))$.
- Compute $w_0^{n+1}(s) = w_0^n(s) \times p_{y_{n+1}|x_{n+1}}(y_{n+1}|x_{n+1}(s))$.

Such a sequential implementation is particularly useful in various settings, just like sequential inference using BP.

19.4 Particle Filtering: Extending to Trees

In this section, we'll show how we can use a particle approach to do approximate inference in general tree structured graphs. As in the previous section, we'll break down the BP equations into two steps and show how particles are updated in each step. The HMM structure was particularly nice and we'll see how the additional flexibility of general trees makes the computation more difficult and how the tools we've learnt so far can surmount these issues.

Recall that for general tree structured graphs, the BP message equations were

$$m_{i\to j}(x_j) = \int_{x_i} \psi_{ij}\phi_i(x_i) \prod_{l\in N(i)\setminus j} m_{l\to i}(x_i)\, dx_i.$$

We can write this as two steps

$$\phi_{ij}(x_i) \triangleq \phi_i(x_i) \prod_{l\in N(i)\setminus j} m_{l\to i}(x_i) \tag{19}$$

$$m_{i\to j}(x_j) = \int_{x_i} \psi_{ij}(x_i, x_j)\phi_{ij}(x_i)\, dx_i. \tag{20}$$

From the messages, the marginals can be computed as

$$p_i(x_i) \propto \phi_i(x_i) \prod_{l\in N(i)} m_{l\to i}(x_i).$$

Naturally, we'll use weighted particle sets to represent messages $m_{i\to j}$ and beliefs ϕ_{ij}. For the details to work out nicely, we require some additional normalization conditions on the potentials

$$\int_{x_i} \phi_i(x_i)\, dx_i < \infty$$

$$\int_{x_j} \psi_{ij}(x_i, x_j)\, dx_j < \infty \text{ for all } x_i$$

The first issue that we run into is how to calculate ϕ_{ij}. If we represent $m_{l\to i}$ as a weighted particle set, we have to answer what it means to multiply weighted particle sets? Another way of thinking about a weighted particle set approximation to $m_{l\to i}$ is

$$m_{l\to i}(x_i) \approx \sum_s w_{l\to i}(s)\delta(x_i - x_{l\to i}(s)).$$

where δ is the Kronecker delta[3]. Then it is clear that multiplication of weighted particle sets, just means multiplication of approximations. But this does not solve the problem, consider $m_{l\to i}(x_i) \times m_{l'\to i}(x_i)$

$$\begin{aligned} m_{l\to i}(x_i) \times m_{l'\to i}(x_i) &\approx \sum_{s,s'} w_{l\to i}(s)\delta(x_i - x_{l\to i}(s))w_{l'\to i}(s')\delta(x_i - x_{l'\to i}(s')) \\ &= \sum_{s,s'} w_{l\to i}(s)w_{l'\to i}(s') \times \delta(x_i - x_{l\to i}(s))\delta(x_i - x_{l'\to i}(s')) \end{aligned}$$

The distributions are continuous, so $x_{l\to i}(s) \neq x_{l'\to i}(s')$ with probability 1, hence the product would be 0. The solution to this problem is to place a *Gaussian kernel* around each particle. Explicitly, instead of approximating $m_{l\to i}$ as a sum of delta functions, we'll approximate it as

$$m_{l\to i}(x_i) \approx \sum_s w_{l\to i}(s)\mathtt{N}(x_i; x_{l\to i}(s), J^{-1}_{l\to i}),$$

where $J_{l\to i}$ is an information matrix and whole sum is referred to as a *mixture of Gaussians*. This takes care of computing the beliefs.

As a result, ϕ_{ij} is a product of $d-1$ mixtures of Gaussians where d is the degree of i. It isn't hard to see that ϕ_{ij} is itself a mixture of Gaussians, but with S^{d-1} components. So, computing the integral in the message equation and generating new particles $x_{i\to j}(s)$ boils down to sampling from a mixture of S^{d-1} Gaussians. Sampling from a mixture of Gaussians is usually not difficult, unfortunately, the mixture has exponentially components, so in this case it is no small task. One approach to this problem is to use MCMC, either Metropolis-Hastings or Gibbs sampling[4]. This gives us the new particles for our message, and we can iterate to compute weighted particle approximations for the rest of the messages and marginals.

To summarize, we could extend particle methods to BP on general tree structures in a natural way. In doing so, we had to overcome two obstacles: "multiplying" weighted particle sets and sampling from a product of mixtures of Gaussians. We overcame the first issue by replacing the δ functions with Gaussians. We tackled the

[3]The Kronecker delta has the special property that if f is a function then $\int_x f(x)\delta(x)\,dx = f(0)$. It essentially picks out the value of f at 0.

[4]See "Efficient Multiscale Sampling From Products of Gaussian Mixtures" Ihler et al. (2003) for more information about this approach.

second issue using familiar tools, MCMC. The end result is an approximate inference algorithm to calculate posterior marginals on general tree structured graphs with continuous distributions!

19.4.1 Concluding Remarks

In the big picture, we combined our BP algorithm with importance sampling to do approximate inference of marginals for continuous distribution. This rounds out our toolbox of algorithms; now we can infer posterior marginals for discrete and continuous distributions. Of course, we could have just used the Monte Carlo methods from the previous lecture to simulate the whole joint distribution. The problem with that approach is that it would be very slow and inefficient compared to the particle filtering approach.

6.438 Algorithms for Inference

Fall 2014