L07 – Approximate Inference via Sampling

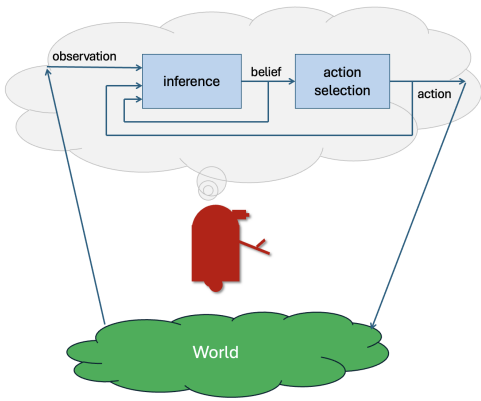
AIMA 13.4, Barber 27.1, 27.6

What you should know after this lecture

- Ancestral sampling in directed models
- Gibbs sampling in Bayes nets and factor graphs
- Intro to more general MCMC methods
- Particle filters

Probabilistic belief representation

- Belief is a probability distribution : $B \in \mathcal{P}(\mathcal{S})$
  (an element of the set of all distributions over $\mathcal{S}$)
- Important questions:
  - What is $p_B(\text{event})$?
  - What is the most likely state $\text{argmax}_s\, p_B(s)$?



Approximate inference

What if your network is highly connected? Exact inference is expensive.
Approximations exist!

- In a loopy factor graph, perform multiple rounds of message passing
  - Not guaranteed to converge
  - If Gaussian loopy BP converges, the means will be correct; under some conditions, the covariances will be, as well.
  - Otherwise, not too much we can say.
- Sampling methods try to draw samples from $P(X_1, \ldots, X_n)$ and compute answers to queries from them. Generally they are consistent: estimates converge in the limit to the true answers, but can take a long time

Sampling in Bayes nets

Easy to do ancestral sampling to get samples of any unconditional marginal or joint $\bar{v} \sim P_\alpha(\bar{V})$ when α is a BN

1. Sort nodes in α into topological order so that all nodes $pa(V)$ come before V in the ordering.
2. For $i = 1$ to M `// number of samples
   - For $j = 1$ to N `// number of nodes in network
     $x_j^i = \text{sample}(P_\alpha(V_j \mid pa(V_j) = x_j^i[pa(V_j)]))$
3. Use $\{x^i\}_{i=1..M}$ to estimate whatever you want, e.g.

$$\hat{P}_\alpha(V_k = 1, V_j = 0) = \frac{1}{M}\mathbb{I}(x_k^i = 1 \wedge x_j^i = 0)$$

where $\mathbb{I}(a) = 1$ if a else 0

Conditional sampling

What if we want $P_\alpha(V \mid E = e)$? Two general approaches:

- Rejection sampling: do ancestral sampling, but throw away all examples in which $E \neq e$.
  - Can be very slow if $P(E = e)$ is small.
- Importance sampling: sample from an easier distribution Q, but reweight the samples to compute your result
  - Let Q be a distribution over same domain as desired distribution P and $\{x^i\}_{1,\ldots,M} \sim Q(x)$
  - Then,

$$E_{x \sim P}[f(x)] \approx \frac{1}{Z} \sum_{x^i \sim Q} \frac{P(x^i)}{Q(x^i)} f(x^i)$$

  - Necessary that $Q(x) > 0$ for any x where $P(x) > 0$.
  - Have to be able to evaluate $P(x)$ and $Q(x)$
  - If P and Q are very different, you will need large M to get a good estimate.

Bayes net importance sampling

In Bayes nets, let $Z = V \setminus E$ (the unobserved nodes)

- Fix all $E = e$ and then use ancestral sampling to get samples from

$$Q(Z) = \prod_i P(Z_i \mid \text{pa}(Z_i))$$

- Importance weights

$$\frac{P(z \mid e)}{Q(z)} \propto \frac{P(z, e)}{Q(z)} = \frac{\prod_i P(z_i \mid \text{pa}(Z_i)) \prod_j P(e_j \mid \text{pa}(E_j))}{\prod_i P(z_i \mid \text{pa}(Z_i)}$$
$$= \prod_j P(e_j \mid \text{pa}(E_j))$$

The name importance sampling also used in a context of sampling from continuous distributions for a different method.

Markov chains

- set $\mathcal{S}$ of states; S_t is a random variable representing the state at time t; $s_i \in \mathcal{S}$ is a possible state
- initial state distribution $p(S_0) = \pi_0$ (row vector)
- transition distribution $P(S_t = s_j \mid S_{t-1} = s_i) = P_{ij}$

We can ask questions like:

- If $S_0 \sim \pi_0$, what is the distribution on S_1?
  Ans: $\pi_0 P$ (check dimensions! be sure it makes sense!)
- What's the probability it will hit s_9 before it hits s_3?
- What is the limiting behavior?
  - Could absorb into a single state and never escape
  - Could enter a deterministic cycle
  - Could have a stationary distribution: $\pi = \lim_{t\to\infty} P^t$ with the property that $\pi P = \pi$ independent of π_0
    Guaranteed if no 0's in P (read about ergodicity)
  - Fun facts: π is an eigenvector of P and the second largest eigenvalue governs the convergence rate

Gibbs sampling

Can be applied in Bayes nets but easiest to think of in factor graphs. Simple type of Markov chain Monte Carlo (MCMC).

- Define a Markov chain where
  - States are assignments of values to all variables
  - The stationary distribution of the chain is the desired distribution $P(V_1, \ldots, V_n)$
  - Samples will be identically distributed but not necessarily independent (because temporally correlated)
- To do estimation:
  - Run the chain for a while and throw those samples away ("burn in" phase) so we are in the stationary distribution
  - Keep (every kth) sample and use them to estimate the quantity of interest

Gibbs sampling in graphical model

1. Initialize values $\bar{v} = (v_1, \ldots, v_N)$ at random
2. Loop
   - Choose i randomly from $1, \ldots, N$
   - Set v_i to a sample from

     $$P(V_i \mid (V \setminus V_i) = (\bar{v} \setminus v_i)) = P(V_i \mid \text{mb}(V_i)) = \bar{v}_{\text{mb}(V_i)})$$

     where $\text{mb}(V_i)$ is the set of variables in the Markov blanket of V_i and $v_{\text{mb}(V_i)}$ is their values in the current assignment $\bar{v}$
3. Use the $\bar{v}$ samples to estimate quantity of interest.

Markov blanket: In a factor graph, it's the neighboring nodes

$$P(V_i \mid \text{mb}(V_i) = \text{mb}(v_i)) = \prod_{\phi \in N(V_i)} \phi[\text{mb}(v_i)]$$

where $\phi[\text{mb}(v_i)]$ is the vector of values for variable V_i that remains after selecting the other dimensions of factor ϕ to have their associated values in $\text{mb}(v_i)$.

Example: one step

Assume a factor graph with binary variables A, B, C, D and

- $\phi_{AB} = [[1, 10], [10, 1]]$
- $\phi_{AC} = [[1, 2], [3, 4]]$
- $\phi_{AD} = [[5, 2], [1, 1]]$

Assume

- the current assignment is $(1, 1, 1, 0)$
- we pick variable A to update
- we construct a distribution on A by
  - Finding all the factors mentioning A
  - For each of those factors, use the current assignments $B = 1$, $C = 1$, $D = 0$ to select out each factor's opinion about A
  - This gives us $[10, 1]$, $[2, 4]$, $[5, 1]$
  - Table multiplication gives us $[100, 4]$
- with probability $100/104$, we change A to have value 0

Gibbs sampling properties

If there are no 0 entries in the factors then, the chain is ergodic, which means

- the chain is aperiodic
- every state is reachable with non-zero probability from every other state

It's not too hard to prove that the joint distribution encoded by the network is the stationary distribution of the Markov chain induced by Gibbs sampling.
Fine in discrete / continuous, loopy graphs.
Read about Metropolis/Hastings (a more general class of algorithms) in AIMA4e to learn more.

Filtering: What if your system isn't conjugate?

- Gaussian errors, but non-linear dynamics: extended Kalman filter
- Somewhat non-Gaussian errors, non-linear dynamics: unscented Kalman filter
- Arbitrary model: particle filter

Filtering by sampling

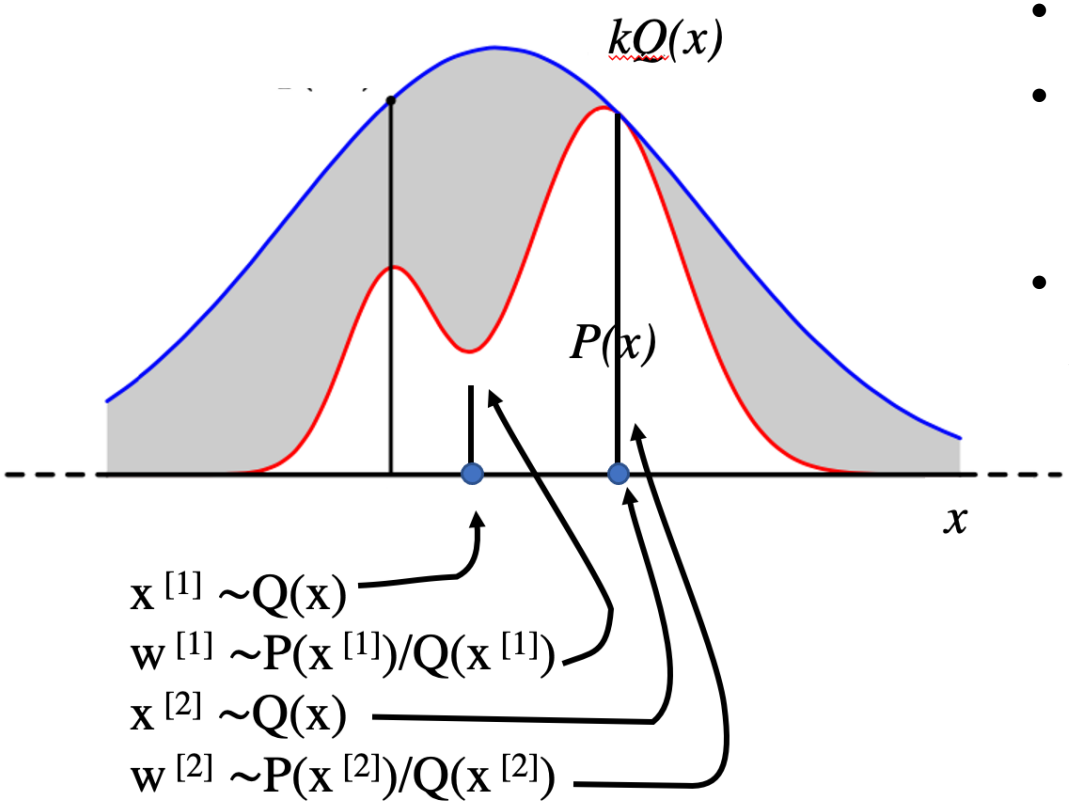
Importance Sampling (AIMA 13.4, Barber 27.6)

Keep all the samples, but weight them! Works in continuous space, too!

- Weight the samples by how much the proposal and target match for the given sample
- When we get a sample in a region where the proposal and target match: weight the sample highly
- When we get a sample in a region where the proposal and target don't match much: give the sample little weight
- Introduce weights

$$w(x) = \frac{P(x)}{Q(x)}$$

Importance Sampling



- If we have unnormalized proposal and target $\tilde{p}$ and $\tilde{q}$, then we have weights $\tilde{w}$.
- It's easy to prove that

$$w(x) = \frac{\tilde{w}(x)}{\sum_x \tilde{w}(x)}$$

- Given importance-weighted samples, we can compute the statistics as before, but with weights:

$$E[x] = \sum w(x^{[i]})x^{[i]}$$

$$E[(x-\mu)^2] = \sum w(x^{[i]})(x^{[i]} - \bar{x})^2$$

Sampling Importance Resampling (SIR) (Barber 27.6)

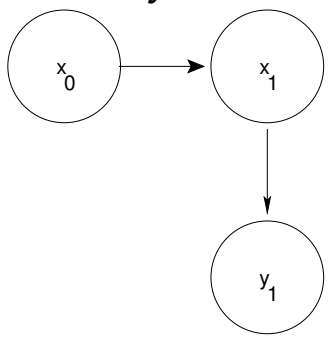
- Imagine that we wanted to infer the distribution $P(\mathbf{x}_{0:T}|\mathbf{y}_{0:T})$
  Bad idea:
  - Sample from some multivariate Gaussian proposal distribution $Q(\mathbf{x}_{0:T}) = N(0, \Sigma)$ where Σ is T dimensional.
  - Weight each sample according to the product of $P(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $P(\mathbf{y}_t|\mathbf{x}_t)$
  - Recover the statistics from the weighted samples.
- If the distance between q and p is large (e.g., Kullback-Leibler divergence), our weights can grow very small
- We may have a lot of samples, but few of them are particularly useful
- Idea: importance **re**-sampling:
  1. Sample from $Q(\mathbf{x})$
  2. Compute weights $w(\mathbf{x}) = P(\mathbf{x})/Q(\mathbf{x})$
  3. Sample from the discrete distribution of sampled $\mathbf{x} \sim w(\mathbf{x})$.

Particle filter

- Use samples as pseudo representation of a distribution ("non-parametric")
- Constant time per update step
- Weight of samples drops exponentially over time (because they are being generated without dependence on the observations)
- Instead, throw away samples with low weight and generate new ones as we go.

Filtering using Sampling: Sequential Importance Resampling

- Recall during filtering, the distribution we want is $P(\mathbf{x}_t|\mathbf{y}_{0:t})$ for each t
- If $T = 1$, then we have a 3-node Bayes net:



- We can get samples over the two latent variables $\mathbf{x}_{0:1}$, assuming we have a prior over $\mathbf{x}_0$.
  1. Sample a value of $\mathbf{x}_0$ according to its prior
  2. Sample a value of $\mathbf{x}_1$ according to the sampled value of $\mathbf{x}_0$ and the noisy dynamics model
  3. Store combined sampled values $\mathbf{x}_0$ and $\mathbf{x}_1$ as a single "particle"
  4. Repeat until happy
- Two problems: what do we do about $P(\mathbf{y}_1|\mathbf{x}_1)$, and how do we marginalize out $\mathbf{x}_0$?

Particle Filtering (Barber 27.6)

- Sampling from $P(\mathbf{y}_1|\mathbf{x}_1)$ doesn't help — we have $\mathbf{y}_1$.
- We need to sample from $P(\mathbf{x}_1, \mathbf{x}_0|\mathbf{y}_1)$
- Bayes rule to the rescue!

This is our target:

$$P(\mathbf{x}_0, \mathbf{x}_1|\mathbf{y}_1) = \alpha P(\mathbf{y}_1|\mathbf{x}_1)P(\mathbf{x}_1, \mathbf{x}_0) \quad \text{(Where did } \mathbf{x}_0 \text{ go in the first term?)}$$

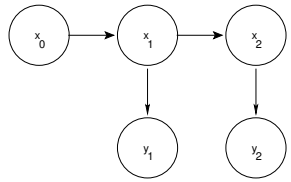
What if this was our proposal?

$$\begin{aligned} Q(\mathbf{x}_0, \mathbf{x}_1) &= P(\mathbf{x}_0, \mathbf{x}_1) \\ \frac{P(\mathbf{x}_0, \mathbf{x}_1|\mathbf{y}_1)}{Q(\mathbf{x}_0, \mathbf{x}_1)} &= \alpha \frac{P(\mathbf{y}_1|\mathbf{x}_1)P(\mathbf{x}_0, \mathbf{x}_1)}{P(\mathbf{x}_0, \mathbf{x}_1)} \\ \Rightarrow w(\mathbf{x}_0, \mathbf{x}_1) &= \alpha P(\mathbf{y}_1|\mathbf{x}_1) \end{aligned}$$

- What do we do about $\mathbf{x}_0$? How do we marginalize it out?
  - We can marginalize out $\mathbf{x}_0$ by resampling $p(\mathbf{x}_0, \mathbf{x}_1)$ according to the weights, and then dropping $\mathbf{x}_0$.

Particle Filtering

- If we move to $k = 2$, we have a 5 node Bayes net



- We could just run the whole process from $\mathbf{x}_0$, but recall that $\mathbf{x}_T \perp\!\!\!\perp \mathbf{x}_{0:T-2}, \mathbf{y}_{0:T-1} | \mathbf{x}_{T-1}$.
- This independence means that if we have a distribution over $\mathbf{x}_{T-1} | \mathbf{y}_{0:T-1}$, we can discard the history $\mathbf{x}_{0:T-2}, \mathbf{y}_{0:T-1}$ from the particle.
- Therefore, for each new time step, we run the algorithm one step from $\mathbf{x}_{T-1}$ to get samples over $\mathbf{x}_T$, weight by the new observation $\mathbf{y}_T$ and resample.
- This gives us the following algorithm:

PARTICLE-FILTER($\mathbf{x}_{T-1}^{[i]}, \mathbf{y}_T$)

  for i from 1 to n

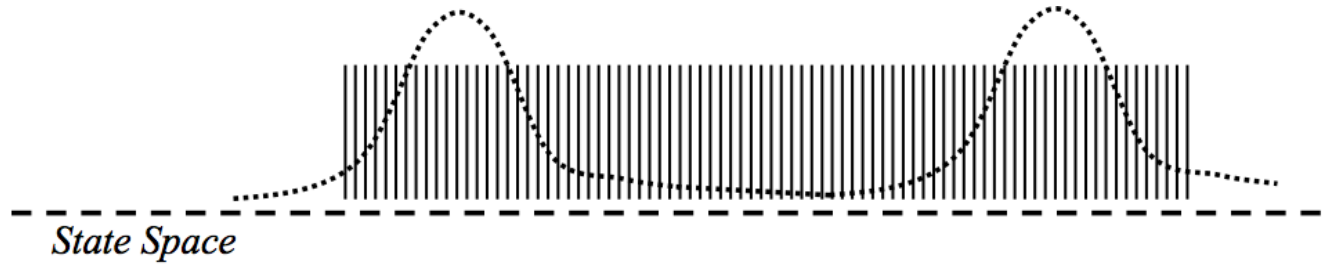
    Sample $\mathbf{x}_T^{[i]} \sim P(\mathbf{x}_T | \mathbf{x}_{T-1}^{[i]})$

    Compute $w^{[i]} = P(\mathbf{y}_T | \mathbf{x}_{T-1}^{[i]}, \mathbf{x}_T^{[i]})$

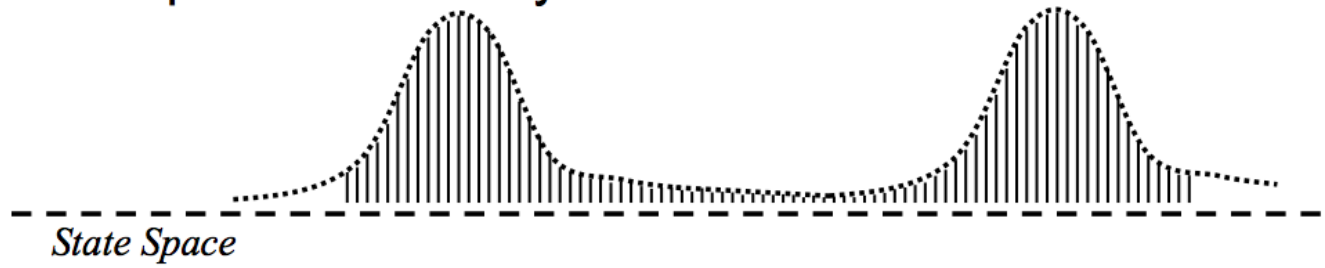
  Generate $x_T^{[1\ldots n]}$ samples from $\{\mathbf{x}_{T-1}^{[i]}, \mathbf{x}_T^{[i]}, w^{[i]}\} \sim w$

  return $x_T^{[1\ldots n]}$

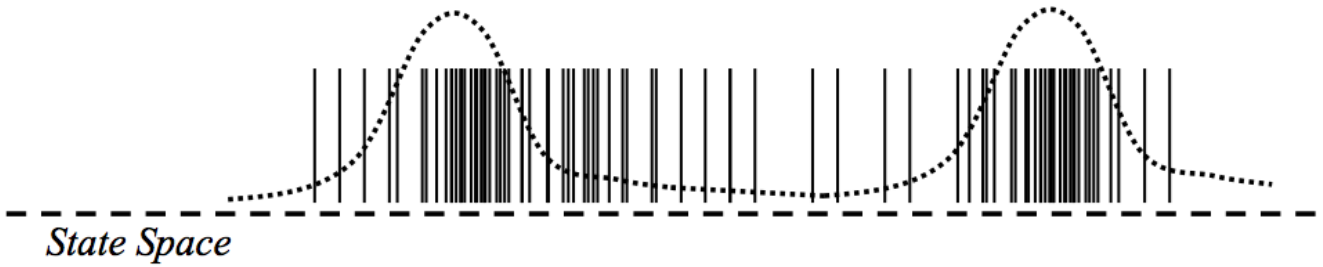
Particle filtering



- Sample from an easy function



- Compute importance weights



- Resample particles from particle set, according to importance weights

Particle filtering – some implementation issues

- You may not want to resample from the weighted particles on every step
  - Resampling may cause important but currently low-probability particles to be lost.
  - One option is to resample only after a certain amount of time when some of the particle weights are consistently very low.
- Need to be careful only to incorporate observations when the dynamics make the observations independent
  - Do not let the particle filter incorporate observations from the same location.
  - This will lead to convergence to a point estimate

Particle filtering – some implementation issues

- Another problem can be accidental particle death, when all the particles are too similar and have very low weight.
- If the measurement likelihood is strongly peaked, only a few particles may have a likely importance weight — these particles will get resampled often, leading to a pool of samples with low diversity $\Rightarrow$ coarsely sampled approximation to the posterior
- To create some particle diversity, after resampling step, may want to add an additional perturbation by sampling a small noise term to be added to the samples.
- May also want to mix in particles from a distribution other than the prior
    - Sample from uniform over the state space : akin to fictitious noise in the Kalman filter: prevents the estimator from becoming too sure due to unmodelled approximations
    - Sample some particles from measurement model, compute importance weights from the dynamics. "Hybrid MCL" (Thrun et al. 2001)

Particle Filtering – Complexity

- There are few useful bounds for sampling techniques, and there is no formal bound on the number of particles required to get good performance
- When there are many local minima in the posterior distribution, need to make sure that you have enough particles
  - Hard to do in general
  - Can use KLD sampling to determine online when more samples are needed [Fox, 2003]
- A "simulated annealing" approach can be used, where the measurements are assumed much noisier than in truth, and the assumed noise is gradually reduced as the distribution converges to a consistent estimate
- Often used for global estimation of the position with no prior knowledge, before "tracking" can begin

So many other important ideas!

- Rao-Blackwellization: particles over some variables, and continuous distributions inside the particles over others.
- Dynamic Bayesian networks: factor states within a time step, and express transitions as a more general Bayes net.