L14 – Continuous factored models: Approximations

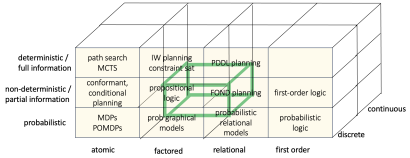
AIMA 14.5.3 KAlg 19.6 Barber 27.6

What you should know after this lecture

- Approximations to Kalman filtering
- Particle filtering

Probabilistic reasoning about partially-specified world states

Factored states
Continuous-valued factors
Approximate inference



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

Extended Kalman filter

- Assume system with non-linearity limited to **f** and **h**

$$\begin{aligned} \mathbf{x}_t &= \mathbf{f}(\mathbf{x}_{t-1}) + \mathbf{w} & \mathbf{w}_t &\sim N(0, W) \\ \mathbf{y}_t &= \mathbf{h}(\mathbf{x}_t) + \mathbf{v}_t & \mathbf{v}_t &\sim N(0, R) \end{aligned}$$

- Taylor series expansion about the current state estimate $\hat{\mathbf{x}}$:

$$\begin{aligned} \mathbf{f}(\mathbf{x}_{t-1}) &= \mathbf{f}(\hat{\mathbf{x}}_{t-1|t-1}) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{t-1|t-1}} (\mathbf{x}_t - \hat{\mathbf{x}}_{t-1|t-1}) + \dots \\ \mathbf{h}(\mathbf{x}_t) &= \mathbf{h}(\hat{\mathbf{x}}_{t|t-1}) + \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_t} (\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1}) + \dots \end{aligned}$$

assumes that all partial derivatives exist.

$$A(\hat{\mathbf{x}}_{t-1|t-1}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{t-1|t-1}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots \\ & & \ddots \end{bmatrix}_{\mathbf{x}_t}, \qquad H(\hat{\mathbf{x}}_{t|t-1}) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\Big|_{\hat{\mathbf{x}}_{t|t-1}}$$

- Note that A and H are now time-varying!

Illustrative Example of EKF Weaknesses

- Shows possible reasons for EKF issues (Julier and Uhlmann [1994])
- Consider motion of vehicle following arc $\mathbf{x} = [x(t) y(t) \psi(t)]^T$, velocity $V(t)$ and radius of curvature $Ra(t)$
  - Velocity disturbed by a zero-mean uncorrelated process.
  - Covariance ellipse in oriented in direction of travel (Fig. 1).
- Later, 1/4 way around circle, Fig. 2 shows true position and uncertainty ellipse at time $t + 1$ — covariance ellipse has been expanded and rotated.

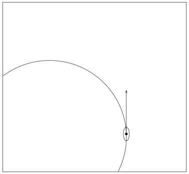


Fig. 1: Mean and covariance of a vehicle at time $t = m$

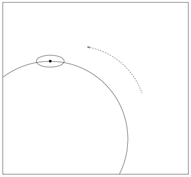


Fig. 2: True mean and covariance prediction to time $t = m + \Delta t$

Illustrative Example of EKF Issues

- EKF predicts covariance using linearized A matrix — equivalent to a constant velocity model tangent to the circle at time t.
- Effect shown in Fig. 3 — mean predicted around arc, but covariance ellipse predicted linearly in direction of travel ⇒ EKF loses critical information that largest component of uncertainty is in direction of travel.
- Error can be corrected by adding process noise to system (see Fig. 4), but at the expense of performance.

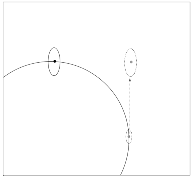


Fig. 3: EKF prediction of mean with linear covariance propagation

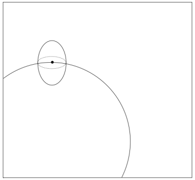


Fig. 4: EKF prediction "adjusted" to compensate for linearisation error

Unscented Kalman filter

- Example of a more general idea: assumed density filtering
- Even if your posterior doesn't have the same parametric form as your prior, find the distribution in the family of the prior that is in some sense closest to the actual posterior
- This is an example of variational inference.
- One strategy is moment matching: estimate mean and covariance of posterior, and pretend it's a Gaussian with those moments. (Can match more moments if that's helpful.)
- UKF is a very simple and even more approximate version of this idea.
- It doesn't stink!

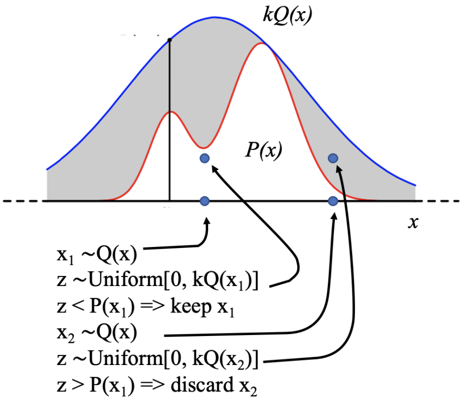
Sampling in continuous problems

- Rejection sampling
- Importance sampling
- Particle filter

Rejection Sampling (Barber 27.1)

- We want to sample from <u>target distribution</u> P, but can't
- Need to be able to evaluate P up to a normalizing constant, i.e., we should be able to evaluate $\tilde{P}(x)$ where $P(x) = \frac{1}{Z}\tilde{P}(x)$ even if we do not know Z.
- We <u>can</u> sample from <u>proposal distribution</u> $Q(x)$
- Pick a constant k such that $kQ(x) \geqslant \tilde{P}(x)$ for all x
- To sample from $P(x)$:
  - Sample $x^{[i]}$ from $Q(x)$
  - Sample z from the rejection probability distribution $[0, kQ(x^{[i]})]$
  - Keep sample $x^{[i]}$ if $z \leqslant \tilde{P}(z)$, otherwise reject $x^{[i]}$

Rejection Sampling



kQ(x)
P(x)
x
$x_1 \sim Q(x)$
$z \sim \text{Uniform}[0, kQ(x_1)]$
$z < P(x_1) \Rightarrow$ keep x_1
$x_2 \sim Q(x)$
$z \sim \text{Uniform}[0, kQ(x_2)]$
$z > P(x_1) \Rightarrow$ discard x_2

- The remaining samples $x^{[i]}$ are distributed according to $P(x)$, and we can compute statistics from these samples:
  $\hat{\mathbf{x}} = E[x] = \frac{1}{n}\sum x^{[i]}$,
  $E[(x-\mu)^2] = \frac{1}{n}\sum(x^{[i]} - \hat{\mathbf{x}})^2$.
- Problems:
  - Can be hard to pick k
  - If the distance between q and p is large (e.g., Kullback-Leibler divergence), then many samples will be rejected $\Rightarrow$ inefficient
  - This problem grows exponentially worse with the dimensionality of x

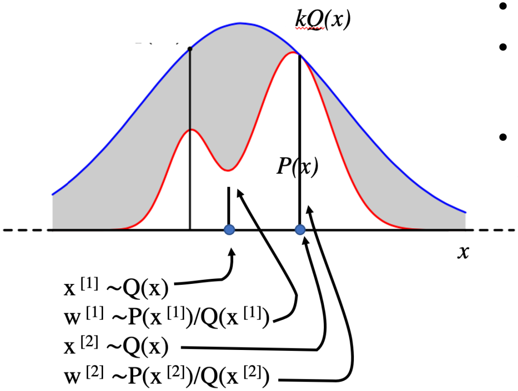
Importance Sampling (AIMA 13.4, Barber 27.6)

Keep all the samples, but weight them!

- Weight the samples by how much the proposal and target match <u>for the given sample</u>
- 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



$kQ(x)$
$P(x)$
x
$x^{[1]} \sim Q(x)$
$w^{[1]} \sim P(x^{[1]})/Q(x^{[1]})$
$x^{[2]} \sim Q(x)$
$w^{[2]} \sim P(x^{[2]})/Q(x^{[2]})$

- 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:

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

Sampling Importance Resampling (SIR) (Barber 27.6)

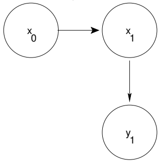
- Imagine that we wanted to infer the distribution $P(\mathbf{x}_{0:T}|\mathbf{y}_{0:T})$
  - Sample from some multivariate Gaussian proposal distribution $Q(\mathbf{x}_{0:T}) = N(0, \Sigma)$ where Σ is T dimensional.
  - Weight each sample according to the product of $P(\mathbf{x}_t|\mathbf{x}_{t-1})$ 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), then we're no longer losing samples, but our weights can grow very small.
  - Seems likely if $Q(\mathbf{x}_{0:T}) = N(0, \Sigma)$ and $\tilde{P}((\mathbf{x}_{0:T})) = \prod_t P(\mathbf{x}_t|\mathbf{x}_{t-1})P(\mathbf{y}_t|\mathbf{x}_t)$
- We may have a lot of samples, but few of them are particularly useful
- Combine rejection sampling and importance 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 $\mathbf{x} \sim w(\mathbf{x})$.
- Also eliminates the need to pick k in rejection sampling

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 $\mathbf{x}_0$. (We will see this as the Prior-Sample algorithm shortly.)
  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 $\mathsf{P}(\mathbf{y}_1|\mathbf{x}_1)$ doesn't help — we <u>have</u> $\mathbf{y}_1$.
- We need to sample from $\mathsf{P}(\mathbf{x}_1, \mathbf{x}_0|\mathbf{y}_1)$
- Bayes rule to the rescue!

This is our target:

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

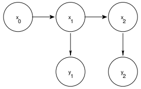
What if this was our proposal?

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

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

Particle Filtering

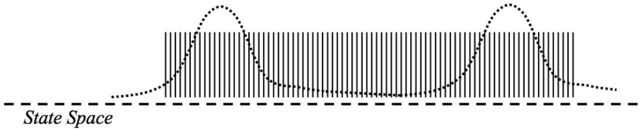
- If we move to $k = 2$, we have a 5 node Bayes net



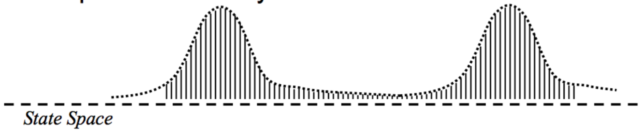
- We could just run the whole process from $\mathbf{x}_0$, but recall that $\mathbf{x}_T \perp\!\!\!\perp \mathbf{x}_{0:T-2}, \mathbf{y}_{0:T-1} | \mathbf{x}_{T-1}$.
- This independence means that if we have a distribution over $\mathbf{x}_{T-1} | \mathbf{y}_{0:T-1}$, we can discard the history $\mathbf{x}_{0:T-2}, \mathbf{y}_{0:T-1}$ 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 $x_T^{[1 \ldots n]}$
```

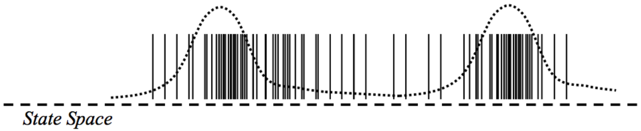
Particle filtering



- 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 ⇒ 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 <u>too</u> sure due to unmodelled approximations
    - Sample some particles from measurement model, compute importance weights from the dynamics. "Hybrid MCL"

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 <u>many</u> 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 <u>more</u> 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.

Next week

- Midterm on Monday!
- MDPs