Contents

1 Continuous Probability Densities

Motivation

In real robots we often deal with continuous variables and continuous state spaces. Hence, we need to extend the notion of probability to continuous variables.

1.1 Continuous Probability Densities

In robotics we typically need to model a belief over continuous, multivariate random variables $x \in \mathbb{R}^n$. We do this using **probability density functions** (PDFs) $p(x)$ over the variables x, satisfying

$$\int p(x)dx = 1. \tag{1}$$

In terms of notation, for continuous variables we use lowercase letters for random variables, and uppercase letters to denote sets of them. We drop the notational conventions of making distinctions between random variables X and their realized values x, which helped us get used to thinking about probabilities, but will get in the way of clarity below.

A **unimodal** density has a single maximum, its **mode**. In general, however, a density can have multiple modes, in which case we speak of a **multimodal** density. The **mean** of a density is defined as

$$\mu = \int xp(x)dx$$

irrespective of whether the density is unimodal or multimodal.

1.2 Gaussian Densities

A Gaussian probability density on a scalar x given **mean** μ and **variance** σ^2 is given by

$$\mathcal{N}(x;\mu,\sigma)^2 = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\}. \tag{2}$$

This density is also known as the "bell curve". This density is very popular in robotics because it is so simple: it is the just the negative exponential of a quadratic around μ. In a Bayesian probability framework we interpret densities as knowledge, and the the **standard deviation** σ indicates the uncertainty we have about the quantity x.

The other reason of the Gaussian's popularity derives from the **central limit theorem**. This theorem says that the probability of the sum of a number of random variables, no matter what the density is on them, will tend to a Gaussian density. And, it does not have to be many random variables either: summing 4 random variables distributed randomly over an interval yields a cubic density, which already matches a Gaussian quite nicely.

A **multivariate Gaussian density** on $x \in \mathbb{R}^n$is obtained by extending the notion of a quadratic to multiple dimensions. We define the squared **Mahalanobis distance,**

$$\|x-\mu\|_\Sigma^2 \overset{\Delta}{=} (x-\mu)^\top \Sigma^{-1} (x-\mu) \tag{3}$$

where $\mu \in \mathbb{R}^n$ is the mean. The quantity Σ is an $n \times n$ **covariance matrix**, a symmetric matrix indicating uncertainty about the mean. The Mahalanobis distance is nothing but a weighted Euclidean distance, and is the multivariate equivalent of the scalar squared distance

$$\|x-\mu\|_{\sigma^2}^2 \overset{\Delta}{=} \left(\frac{x-\mu}{\sigma}\right)^2$$

but using the matrix inverse to weight this distance metric in an n-dimensional space. Armed with this, the equation for the multivariate Gaussian is

$$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left\{-\frac{1}{2}\|x-\mu\|_\Sigma^2\right\}, \tag{4}$$

where the term $|2\pi\Sigma|$ in the normalization factor denotes the determinant of $2\pi\Sigma$.

Gaussian densities, whether scalar or multivariate, have some nice properties. A Gaussian density is unimodal and the mean μ is also its mode. In addition, any scalar marginal of a multivariate Gaussian is also a Gaussian. In fact, the probability density $P(y)$ of *any* linear combination $y = Hx$, with y an m-dimensional vector and H an $m \times n$ matrix, is also Gaussian with mean $H\mu$ and covariance $H\Sigma H^T$.

Exercise

1. Given a 2-dimensional density on (x, y), with mean $\mu = (\bar{x}, \bar{y})$ and covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_x^2 & r \\ r & \sigma_y^2 \end{bmatrix}$$

what is the variance of the marginals $p(x)$ and $p(y)$? Use the linearity property above.

2. Given the same 2D density, what is the mean and variance of the sum $z = x + y$?

3. Deeper thinking: what happens to the density if we push it through a nonlinear function, e.g., $z = \sqrt{x^2 + y^2}$, the Euclidean norm of the vector (x, y)? A qualitative answer is asked for.

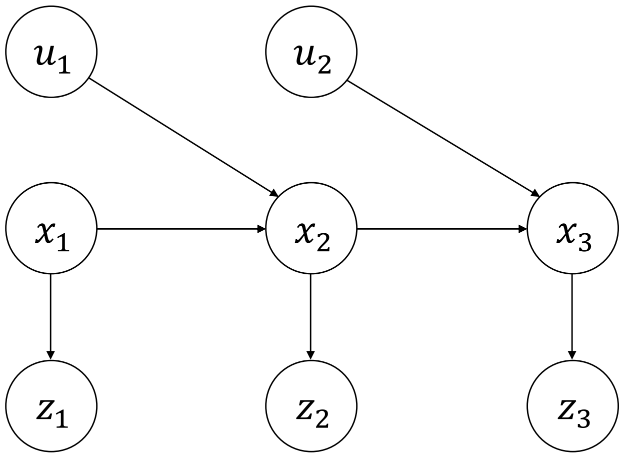

Figure 1: Continuous Bayes net modeling a robot with continuous states x_t, continuous measurements z_t, and continuous controls u_t.

1.3 Bayes Nets and Mixture Models

Continuous Bayes nets are exactly like discrete Bayes nets, except with continuous variables. Most of the concepts generalize effortlessly, and we will forego very formal definitions where we can.

An example crucial to the robotics domain is shown in Figure 1, which is the continuous equivalent of the dynamic Bayes net from Part I. In the figure, the continuous Markov chain backbone represents the evolution of the continuous state x_t over time, conditioned on the **controls** u_t. Notice we use new terminology here: we say *controls* rather than actions in this new, continuous world. For example, for the Duckiebot, the control u might be two-dimensional and represent the wheel speeds of the left and right wheels, respectively. Finally, the continuous measurements z_t at the bottom are conditioned on the states x_t.

We can mix and match discrete and continuous variables. A particularly simple Bayes net is a **Gaussian mixture model**. The joint density is

$$p(x, C) = p(x|C)P(C)$$

where $P(C)$ is a PMF on a discrete variable C that chooses between different Gaussians, , and $p(x|C)$ is the corresponding Gaussian mixture component. Even though each Gaussian density is unimodal, the marginal $P(x)$ is a multimodal density

$$p(x) = \sum_c p(x|C = c)p(C = c)$$

where the sum is over components. An example for a two-component mixture is shown in Figure 2.

Continuous probability densities present a representational challenge. We can no longer specify CPTs: either an equation needs to be available, as with the Gaussian density above, or an arbitrary density has to be somehow approximated. One such approximation is exactly using mixture densities: we can "mix" many simpler densities, like Gaussians, to approximate a more complicated density. This is known as **Parzen window density estimation**.

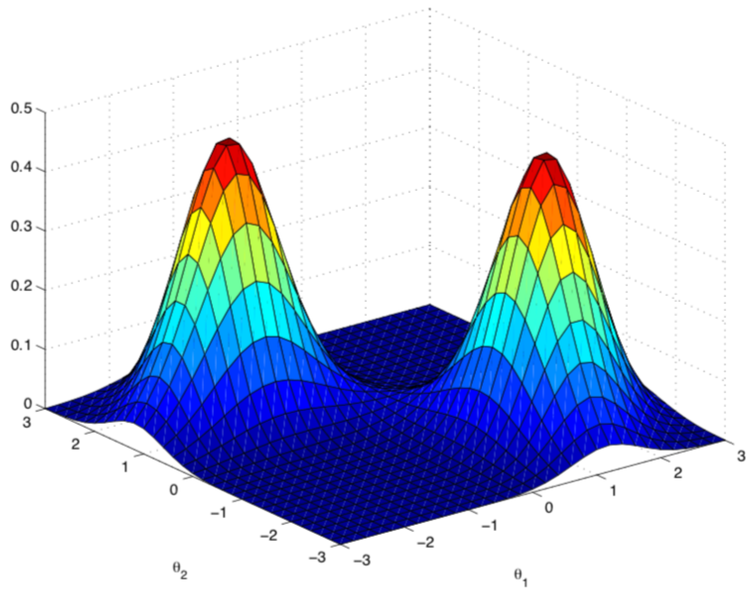

Figure 2: Gaussian mixture density with two components of about equal weight.

1.4 Continuous Measurement Models

In many cases it is both justified and convenient to model measurements as corrupted by zero-mean Gaussian noise. For example, a bearing measurement from a given pose $x \in SE(2)$ to a given 2D landmark l would be modeled as

$$z = h(x, l) + \eta, \tag{5}$$

where $h(.)$ is a **measurement prediction function**, and the noise η is drawn from a zero-mean Gaussian density with **measurement covariance** R. This yields the following conditional density $p(z|x, l)$ on the measurement z:

$$p(z|x, l) = \mathcal{N}(z; h(x, l), R) = \frac{1}{\sqrt{|2\pi R|}} \exp\left\{-\frac{1}{2}\left\|h(x, l) - z\right\|_R^2\right\}. \tag{6}$$

The measurement functions $h(.)$ are often nonlinear in practical robotics applications. Still, while they depend on the actual sensor used, they are typically not difficult to reason about or write down. The measurement function for a 2D bearing measurement is simply

$$h(x, l) = atan2(l_y - x_y, l_x - x_x) - x_\theta, \tag{7}$$

where $atan2$ is the well-known two-argument arctangent variant. Hence, the final **probabilistic measurement model** $p(z|x, l)$ is obtained as

$$p(z|x, l) = \frac{1}{\sqrt{|2\pi R|}} \exp\left\{-\frac{1}{2}\left\|atan2(l_y - x_y, l_x - x_x) - x_\theta - z\right\|_R^2\right\}. \tag{8}$$

Note that we will not *always* assume Gaussian measurement noise: to cope with the occasional data association mistake, we can use robust measurement densities, with heavier tails than a Gaussian density.

1.5 Continuous Motion Models

For a robot operating in the plane, probabilisitic **motion models** are densities of the form $p(x_{t+1}|x_t)$, specifying a **probabilistic motion model** which the robot is assumed to obey. This *could* be derived from odometry measurements, in which case we would proceed exactly as described above. Alternatively, such a motion model could arise from known control inputs u_t. In practice, we often use a conditional Gaussian assumption,

$$p(x_{t+1}|x_t, u_t) = \frac{1}{\sqrt{|2\pi Q|}} \exp\left\{-\frac{1}{2}\|g(x_t, u_t) - x_{t+1}\|_Q^2\right\}, \tag{9}$$

where $g(.)$ is a motion model, and Q a covariance matrix of the appropriate dimensionality, e.g., 3×3 in the case of robots operating in the plane.

Summary

We briefly summarize what we learned in this section:

1. Continuous probability densities generalize the notion of probability distributions to continuous random variables.
2. The scalar and multivariate Gaussian densities are useful and relatively simple.
3. Bayes nets generalize as well, and even allow for discrete and continuous variables in the same Bayes net, as in the case of mixture densities.
4. Continuous measurement models typically have a measurement prediction corrupted by noise, often modeled as Gaussian.
5. Continuous measurement models follow a similar pattern, but are conditioned on controls.

2 Monte Carlo Inference

Motivation

With nonlinear dynamics and/or measurement models exact inference is computationally intractable. One way out is to perform approximate inference using sampling. One of the best known such algorithms are particle filters, including Monte Carlo Localization.

2.1 Simulating from a Continuous Bayes Net

Sampling from a continuous Bayes net is done using ancestral sampling, as before: we topologically sort the DAG, and then sample the conditional densities in topological sort order. For the dynamic Bayes net from Figure 1, the topological sort trivially follows the temporal ordering, and we sample first within the red, then green, and then the blue time-slice, as shown in Figure 3.

In each slice, the story is pretty much the same. For example, in the second (green) slice we will

1. Sample the state x_2 from the continuous motion model $p(x_2|x_1, u_1)$, by evaluating $g(x_1, u_1)$, and adding Gaussian noise with mean $\mu = 0$ and covariance Q.

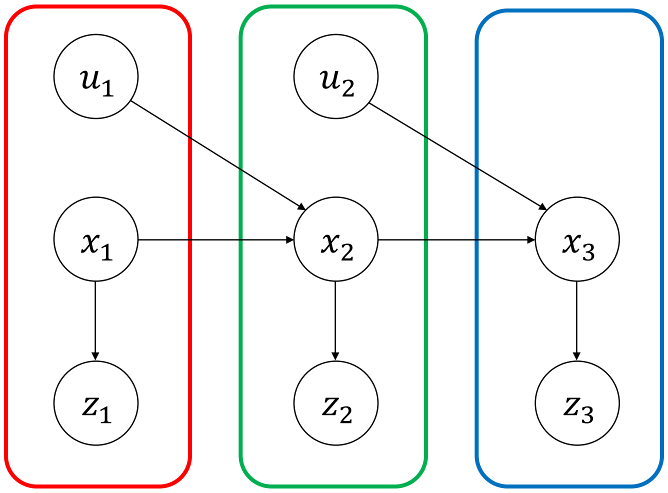

Figure 3: Ancestral sampling in a continuous Bayes net.

2. Sample the measurement z_2 from the continuous measurement model $p(z_2|x_2)$, and adding zero-mean Gaussian noise with measurement covariance R.
3. Sample a control u_2 that we will use in the next time-slice.

The only exception is the first slice, where the first state x_1 will be sample from the first-state prior $P(x_1)$.

Figure 4 shows an example where we sample from the motion-model only, giving rise to so-called "banana-shaped" densities. What is happening is mysterious at first, until you realize that the red dots in the figure only show the marginal of the robot position, not the full 3-dimensional density on the pose (x, y, θ). Even though the in this simplistic example the motion models are simply "move 2 meters", or "turn left and then move two meters", the Gaussian additive noise can make the resulting densities decidedly non-Gaussian after several iterations. The reason is the noise on the heading θ: this leads to non-linear effects over time.

2.2 Sampling as an Approximate Representation

TBD: refer to the slides for now.

2.3 Importance Sampling

Sampling from arbitrary densities is not easy in general. Gaussian densities, which are relatively easy to sample from, are the exception rather than the rule.

A simple method is **importance sampling**. Given any **target density** $p(x)$, we instead sample from a (hopefully easier) **proposal density** $q(x)$, and then produce *weighted samples* with **importance weights**

$$\pi^{(r)} = \frac{q(x^{(r)})}{q(x^{(r)})}$$

where $x^{(r)}$ is a sample (with index r) from $q(x)$. After sampling the required number, the importance weights can be normalized to sum up to 1.

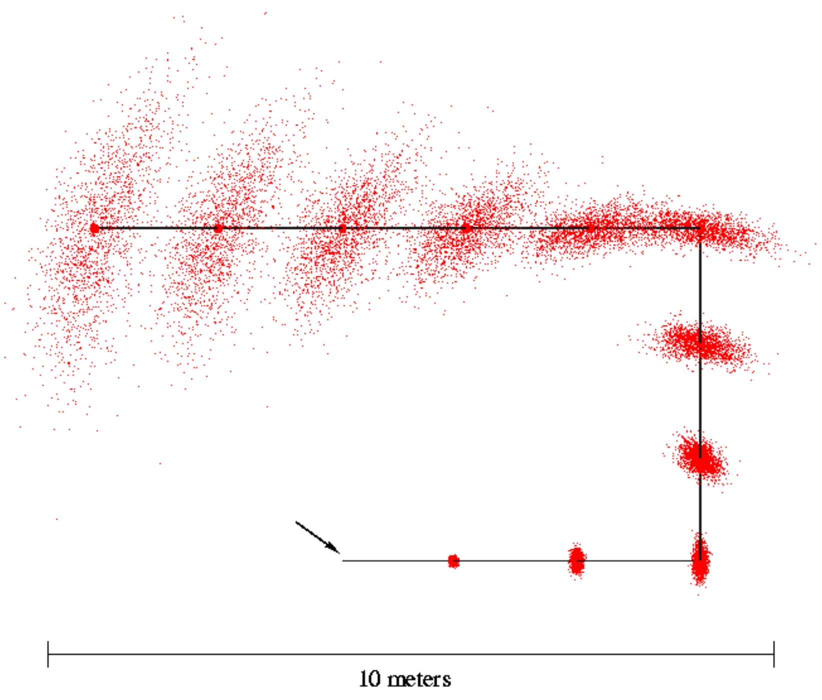

Figure 4: Sampling from the motion-model only, giving rise to so-called "banana-shaped" densities.

Importance sampling can be used as a simple approximate implementation of Bayes rule. In this case, the target density is $p(x|z)$, and we can use the prior $p(x)$ as the proposal density. Hence, the weights can be computed as

$$\pi^{(r)} = \frac{p(x^{(r)}|z)}{p(x^{(r)})} \propto \frac{l(x^{(r)}; z)p(x^{(r)})}{p(x^{(r)})} = l(x^{(r)}; z)$$

where we used $p(x|z) \propto l(x; z)p(x)$, with $l(x; z)$ the likelihood of x given z. Hence, when implementing Bayes law by proposing from the prior, the weights are simply the measurement likelihoods $l(x^{(r)}; z)$.

As an aside: sampling from a factor graph can be done using Gibbs sampling, but this is computationally heavy and is out of the scope of this course.

2.4 Particle Filters and Monte Carlo Localization

There are two ways to intuitively derive particle filtering:

1. Let us assume that we have a set of samples on X_{t-1}, i.e., $\{x_{t-1}^{(s)}\} \sim p(x_{t-1}|Z^{t-1})$. We create a tiny Bayes net, $x_{t-1} \to x_t \to z_t$, and just extend the sample set as in ancestral sampling, obtaining $\left\{(x_{t-1}, x_t)^{(s)}\right\} \sim p(x_{t-1}, x_t|Z^{t-1})$. We then marginalize to $\{x_t^{(s)}\} \sim p(x_t|Z^{t-1})$, weight by the likelihoods $l(x_t^{(s)}; z_t)$ to obtain $\{x_t^{(s)}, \pi_t^{(s)}\} \sim p(X_t|Z^t)$. Finally, resample to get back to an unweighted set of particles for the next step. In summary, given the approximate posterior at the previous time step $\{x_{t-1}^{(s)}\} \sim p(x_{t-1}|Z^{t-1})$, we
   (a) extend to include the next state to obtain $\left\{(x_{t-1}, x_t)^{(s)}\right\} \sim p(x_{t-1}, x_t|Z^{t-1})$;

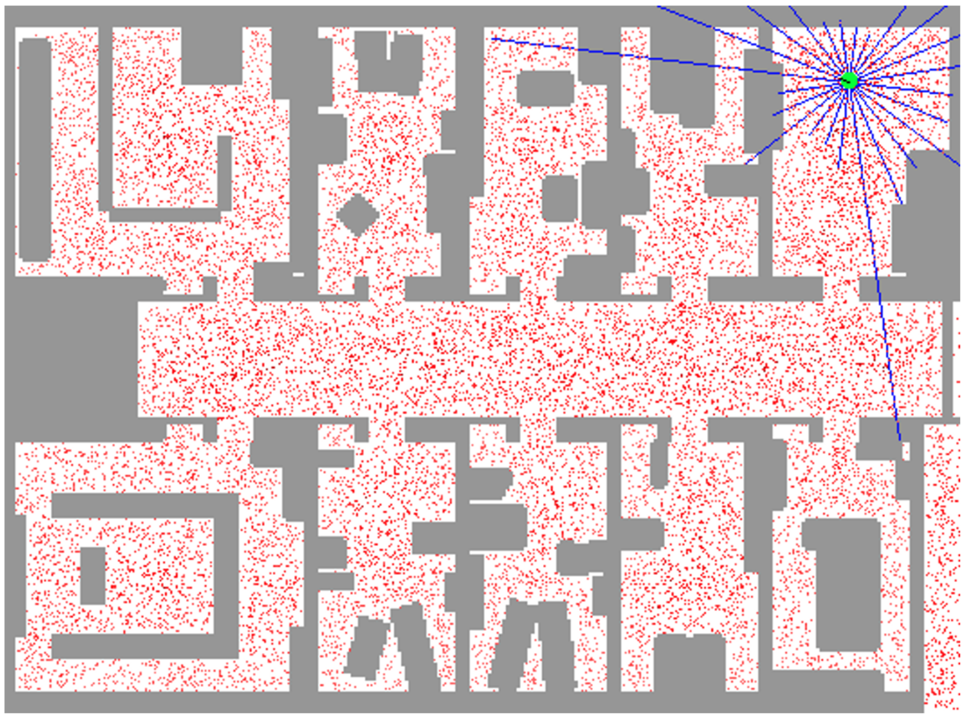

Figure 5: Monte Carlo localization

(b) marginalize to obtain the approximate predictive density $\{x_t^{(s)}\} \sim p(x_t|Z^{t-1})$;

(c) weight by the likelihoods $l(x_t^{(s)}; z_t)$ to obtain $\{x_t^{(s)}, \pi_t^{(s)}\} \sim p(x_t|Z^t)$;

(d) resample to get back an unweighted approximation $\{x_t^{(r)}\} \sim p(x_t|Z^t)$.

2. Let us assume that we have a *weighted* set of samples by marginalizing out x, i.e., $\{x_{t-1}^{(r)}, \pi_{t-1}^{(r)}\} \sim p(x_{t-1}|Z^{t-1})$. We can form a mixture density that approximates the predictive density $p(x_t|Z^{t-1}) \approx \sum_r \pi_{t-1}^{(r)} p(x_t|x_{t-1}^{(r)})$, sample from it to get $\{x_t^{(s)}\} \sim p(x_t|Z^{t-1})$, and then weight the samples using the likelihood: $\{x_t^{(s)}, \pi_t^{(s)}\} \sim P(x_t|Z^t)$, where $\pi_t^{(s)} = l(x_t^{(s)}; z_t)$. In summary, given the approximate posterior using weighted samples $\{x_{t-1}^{(r)}, \pi_{t-1}^{(r)}\} \sim p(x_{t-1}|Z^{t-1})$, we

   (a) approximate the predictive density as a mixture density $p(x_t|Z^{t-1}) \approx \sum_r \pi_{t-1}^{(r)} p(x_t|x_{t-1}^{(r)})$;

   (b) sample from it to get $\{x_t^{(s)}\} \sim p(x_t|Z^{t-1})$;

   (c) weight the samples using the likelihood: $\{x_t^{(s)}, \pi_t^{(s)}\} \sim P(x_t|Z^t)$, where $\pi_t^{(s)} = l(x_t^{(s)}; z_t)$.

The former derivation is a straight extension of importance sampling as in the previous section, but the resampling step is somewhat magical. The latter derivation shows that the resampling can be interpreted as approximating the predictive density for X_t. Of course, both derivations lead to the exact same algorithm.

Exercise

1. Show that both algorithms are the same.

2. Implement a 1D version of the particle filter in python.

2.5 Connection with the Elimination Algorithm*

Factor graphs were not harmed in the making of this filter. But, you can interpret the weighted samples as a factor $f_1(x_{t-1})$ on x_{t-1}, then add a motion model to x_t and a likelihood factor on x_t.

A continuous version of eliminating x_{t-1}, if this was tractable, would form the product factor

$$\psi(x_{t-1}, x_t) = f_1(x_{t-1})p(x_t|x_{t-1})$$

and try to re-write it as $p(x_{t-1}|x_t)\tau(x_t)$. We saw in the discrete sum-product algorithm that this is done by marginalization. Doing this here, we obtain the new factor

$$\tau(x_t) = \sum f_1(x_{t-1})p(x_t|x_{t-1}),$$

which can be recognized as approximating the predictive density $p(x_t|Z^{t-1})$. In the elimination step we also usually calculate the conditional $p(x_{t-1}|x_t)$ but we do not need it here.

Then, eliminating x_t would form the product $\tau(x_t)l(x_t; z_t)$. This is hard if we would like to modify mixture components exactly, but easy if we first sample from them: we just re-weight then with the likelihood, and we are done. Hence, the factor graph interpretation closely aligns with the second derivation above.

Summary

We briefly summarize what we learned in this section:

1. Simulating Continuous Bayes Nets
2. Sampling as Approximation
3. Importance Sampling
4. Particle Filters and Monte Carlo Localization
5. Monte Carlo Localization & the Elimination Algorithm